République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université 8Mai 1945 – Guelma
Faculté des sciences et de la Technologie
Département d'Electronique et Télécommunications

Mémoire de fin d'étude
pour l'obtention du diplôme de Master Académique

Domaine : **Sciences et Technologie**
Filière : **Electronique**
Spécialité : **Systèmes Electroniques**

Commande neuronale direct avec modèle inverse en utilisant le perceptron multicouche

---

Présenté par :

LEBED TOUFIK

MAIZ SEYFEDDINE SALEH

Sous la direction de :

DR. NEMISSI MOHAMED

Juin 2017

REMERCEMENTS

Toute notre gratitude et remerciements avant tout vont à Allah Qui nous a donné la santé, la force, la patience le courage et la Volonté de finir ce travail.

Nos vifs remerciements vont à notre encadreur le Dr. NEMISSI MOHAMED pour son encadrement et ses Conseils scientifiques tout le long de ce travail.

Aussi

Nos remerciements vont aux membres du jury d'avoir honoré notre Soutenance et pour l'effort fourni afin de juger ce modeste travail.

Nos remerciements vont également à tous nos enseignants qui ont, Contribués à notre formation.

DEDICACES

Je dédie ce modeste travail :

À ma très chère mère, mon très cher père,

À mon frère, et mes sœurs,

À toute ma famille,

À l'ensemble de mes amis.

Lebed toufik

Dédicace

Je voudrais dédier cet humble travail à toute ma famille, à ma chère maman qui ne m'a jamais quitté,et qui a veille à ce que je sois et ce que je suis devenu maintenant.

À mon cher pére qui peut etre fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse dieux faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobeles, l'éducation et le soutient permanent venu de toi.

À mes très chères sœurs Nadjah, Sana, Mouna, Sabrina, Radja, Bassma

À Mon très chèr frère Riad,

À tous mes cousines sur tous Salah, Boubaker qui sont comme mes fréres.

À tous mes amis: Hassan , Amine, Iassa, Charaf, Islam, Mohamed...

Et Je dédie ce mémoire à tous mes enseignants ,

À tous ceux qui m'aiment...

SEYF.

Résumé

Ce travail consiste à exploiter les capacités d'apprentissage, de calcul parallèle et d'approximation des réseaux de neurones pour reproduire la dynamique inverse des systèmes non-linéaires, puis à l'utilisation du modèle neuronal inverse ainsi formé comme étant un contrôleur direct de ce système.

La méthode étudiée s'effectue en trois étapes essentielles : i) création des données d'apprentissage ; ii) apprentissage des réseaux de neurones afin de créer un modèle permettant de reproduire la dynamique inverse du système ; iii) l'application du modèle obtenu pour la commande du système.

SOMMAIRE

CHAPITRE II : Commande par les réseaux de neurones

INTRODUCTION GÉNÉRALE

Introduction générale :

La complexité des applications actuelles comportant beaucoup de non-linéarités, des dynamiques difficile a modélisé, des bruits non mesurables exigent a la théorie de contrôle d'intégrer de nouveaux concepts qui s'inscrit dans le cadre de la commande intelligente. L'objectif est d'introduire de nouveaux mécanismes permettant une commande plus simple, capable de s'adapter à des variations de l'environnement et démontrant des capacités d'apprentissage. Les réseaux de neurone artificiels font partie de cette tendance actuelle.

La commande par les réseaux de neurones comporte généralement une étape d'identification et une étape de contrôle. L'identification consiste à élaborer un modèle neuronal qui est une estimation du processus à commander et cela au moyen d'une phase d'apprentissage. Celle-ci peut être soit préalable (hors ligne), ou bien elle peut se faire intégralement en ligne. La commande utilise les connaissances acquises pendant la phase d'identification et/ou de l'apprentissage en ligne pour élaborer des signaux de commande.

L'objectif de ce travail est d'exploiter les capacités des réseaux de neurones, notamment celle de l'apprentissage pour reproduire la dynamique inverse des systèmes non-linéaires, puis à l'utilisation du modèle neuronal inverse ainsi formé comme étant un contrôleur direct de ce système. Nous utilisons le perceptron multicouche qui est le réseau de neurones le plus utilisé et le plus simple et qui est aussi très efficace.

Le présent mémoire comporte trois chapitres.

Le premier chapitre consiste en une introduction aux réseaux de neurones. Nous décrivons leurs principes de modélisation, leur apprentissage et leurs architectures. Nous présentons également à la fin de ce chapitre quelques avantages et inconvénient des réseaux de neurones.

Le deuxième chapitre constitue une introduction à la commande neuronale. Nous présentons les motivations d'utilisation des réseaux de neurones dans le domaine de la commande et quelques approches de la commande neuronale, puis nous présentons la méthode étudiée : la commande neuronale avec modèle inverse.

Le troisième chapitre présente les résultats obtenus de la commande neuronale avec modèle inverse en utilisant le perceptron multicouche sur un système non linéaire.

Finalement, une conclusion générale conclue ce mémoire.

CHAPITRE I

GÉNÉRALITÉ SUR LES RÉSEAUX DE NEURONES

I.1. Introduction :

Les dernières années ont vu un développement technologique puissant dans des domaines divers, et il y a eu un accroissement de besoin pour le contrôle et la gestion des systèmes complexes qui introduisent d'énorme calculs et un nombre de variables important ; d'où la nécessité de chercher de nouvelles méthodes pour une gestion plus souple et moins coûteuse en temps de calculs et en manipulation des variables dont le nombre ne cesse d'augmenter. Pour cela, on s'est intéressé de plus en plus aux systèmes qui apprennent, en utilisant des modélisations des neurones biologiques.

Les réseaux de neurones artificiels ont été étudiés pendant plusieurs années dans le but d'imiter les performances du cerveau de l'être vivant.

Inspirés des réseaux neuromimétique biologiques, ils existent plusieurs modèles de réseaux de neurones artificiels, et chaque modèle se prête bien pour une application particulière (classification, reconnaissance, contrôle ...etc.) ; Mais leurs utilisations restent limitées dans quelques applications, et il reste beaucoup de domaines où les réseaux de neurones n'ont pas trouvé de solutions, telle que la planification par exemple. Les meilleurs systèmes à réseaux de neurones restent assez loin d'imiter des performances telles que celles de l'être humain.

I.2. Historique :

Dès 1890, W. James, célèbre psychologue américain introduit le concept de mémoire associative et propose ce qui deviendra une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones connue plus tard sous le nom de loi de Hebb.

En 1943 J. Mc Culloch et W. Pitts laissent leurs noms à une modélisation du neurone biologique (un neurone au comportement binaire). Ceux sont les premiers à montrer que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes (tout au moins au niveau théorique).

En 1949 D. Hebb, physiologiste américain explique le conditionnement chez l'animal par les propriétés des neurones eux-mêmes. Ainsi, un conditionnement de type pavlovien tel que, nourrir tous les jours à la même heure un chien, entraîne chez cet animal la sécrétion de salive à cette heure précise même en l'absence de nourriture. La loi de modification des propriétés des connexions entre neurones qu'il propose explique en partie ce type de résultats expérimentaux.

En 1957 F. Rosenblatt développe le modèle du Perceptron. Il construit le premier neuro-ordinateur basé sur ce modèle et l'applique au domaine de la reconnaissance de formes.

En 1960 : B. Widrow, un automaticien, développe le modèle Adaline (Adaptative Linear Element). Dans sa structure, le modèle ressemble au Perceptron, cependant la loi d'apprentissage est différente.

En 1969, M. Minsky et S. Papert publient un ouvrage qui met en exergue les limitations théoriques du perceptron. Limitations alors connues, notamment concernant l'impossibilité de traiter par ce modèle des problèmes non linéaires. Ils étendent implicitement ces limitations à tous modèles de réseaux de neurones artificiels. Leur objectif est atteint, il y a abandon financier des recherches dans le domaine (surtout aux U.S.A.), les chercheurs se tournent principalement vers les systèmes à bases de règles.

Du 1967 jusqu'à 1982 S. Grossberg, T. Kohonen, toutes les recherches ne sont pas interrompues elles se poursuivent sous le couvert de divers domaines comme : le traitement adaptatif du signal, la reconnaissance de formes, la modélisation en neurobiologie, etc.

En 1982 : J. J. Hopfield est un physicien reconnu à qui l'on doit le renouveau d'intérêt pour les réseaux de neurones artificiels met en avant l'isomorphisme de son modèle avec le modèle d'ising (modèle des verres de spins). Cette idée va drainer un flot de physiciens vers les réseaux de neurones artificiels.

En 1983 : La machine de Boltzmann est le premier modèle connu apte à traiter de manière satisfaisante les limitations recensées dans le cas du perceptron. Mais l'utilisation pratique s'avère difficile, la convergence de l'algorithme étant extrêmement longue.

En 1985 : La rétro-propagation de gradient apparaît nous avons la possibilité de réaliser une fonction non linéaire d'entrée/sortie sur un réseau en décomposant cette fonction en une suite d'étapes linéairement séparables [1].

I.3.Domaines d'application :

Aujourd'hui, les réseaux de neurones ont de nombreuses applications dans des domaines très variés :

- Traitement d'image : compression d'images, reconnaissance de caractères et de signatures, reconnaissance de formes et de motifs, cryptage, classification, ...
- Traitement du signal : traitement de la parole, identification de sources, filtrage, classification, ...
- Contrôle : diagnostic de pannes, commande de processus, contrôle qualité, robotique.
- Optimisation : allocation de ressources, planification, régulation de trafic, gestion, finance, etc. ...
- Simulation : simulation boîte noire, prévisions météorologiques.
- Classification d'espèces animales étant donnée une analyse ADN.
- Modélisation de l'apprentissage et perfectionnement des méthodes de l'enseignement.
- Approximation d'une fonction inconnue ou modélisation d'une fonction connue mais complexe à calculer avec précision [2].

I.4. Principes de modélisation des réseaux de neurones:

I.4.1. Le neurone biologique :

Les biologistes estiment que le système nerveux compte plus de 100 milliards de neurones interconnectés. Bien que les neurones ne soient pas tous identiques, leur forme et certaines caractéristiques permettent de les répartir en quelques grandes classes. En effet, il est aussi important de savoir que les neurones n'ont pas tous un comportement similaire en fonction de leur position dans le cerveau. Le neurone biologique présenté à la figure I.1 peut être décomposé en quatre principales parties : le corps cellulaire, les dendrites, l'axone et la synapse [3].

a) Dendrites

Chaque neurone possède une chevelure de dendrite qui forme une sorte d'arborescence autour du corps cellulaire, elles permettent aux neurones de capter les signaux qui parviennent de l'extérieur.

b) Corps cellulaire

Dans la plupart des cas, la forme du corps cellulaire dépend de sa position dans le cerveau, elle peut être pyramidale ou sphérique. Il inclut le noyau du neurone et effectue les transformations biochimiques necessaires a la vie du neurone.

c) L'axone

L'axone est la fibre nerveuse qui permet la transposition des signaux émis par le neurone généralement un axone est plus long que les dendrites, ainsi se ramifie à son extrémité, là où il communique avec les autres neurones.

d) Les synapses

Ce sont des jonctions entre deux neurones et qui sont essentielles dans le fonctionnement du système nerveux.

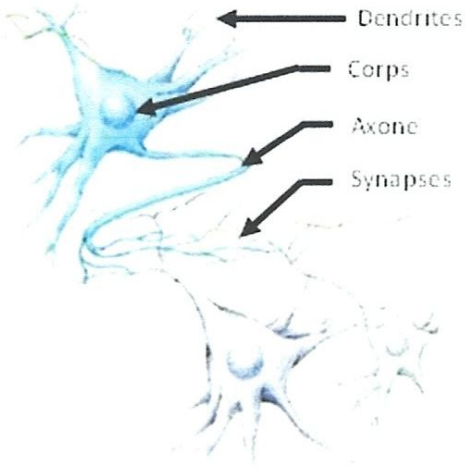


Figure I.1 : Schéma d'un neurone biologique.

I.4.2.Le neurone formel:

Un neurone artificiel est une fonction mathématique conçu comme un modèle de neurones biologiques. Les neurones artificiels sont des unités constitutives dans un réseau neuronal artificiel. Selon le modèle utilisé, ils peuvent être appelés une unité semi-linéaire, neurone binaire, la fonction de seuil linéaire, ou simplement neurone.

Le neurone artificiel reçoit une ou plusieurs entrées (représentant les dendrites) et les résume pour produire une sortie (représentant l'axone d'un neurone). Habituellement, les valeurs de chaque nœud sont pondérées, et la somme est passée à travers une fonction non-linéaire connue en tant que fonction de l'activation ou de la fonction de transfert. Les fonctions de transfert ont généralement une forme sigmoïde, mais ils peuvent aussi prendre la forme d'autres fonctions non-linéaires, des fonctions linéaires par morceaux, ou fonctions en escalier. Ils sont aussi souvent monotones croissante, continue, dérivable et bornée.

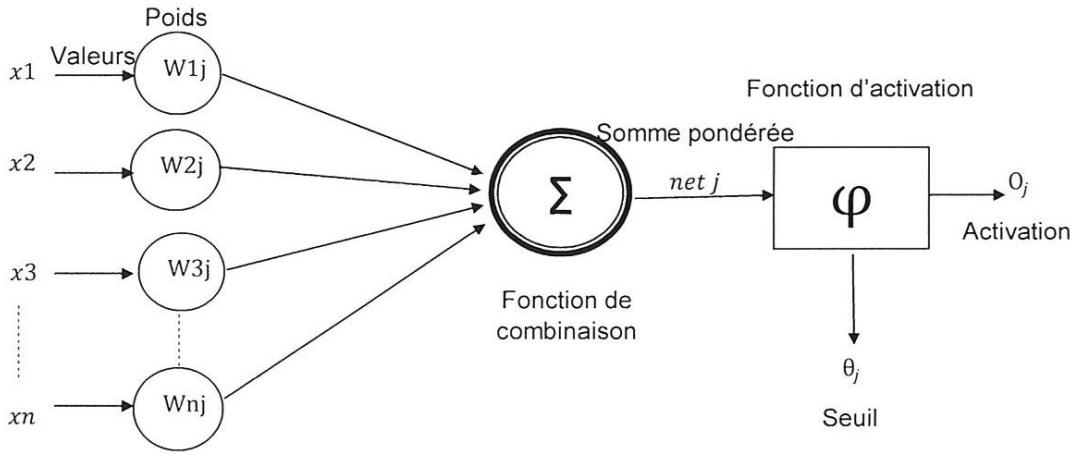


Figure I.2 : Structure générale d'un neurone.

I.4.3.Fonctions d'activation :

Il est clair que la fonction d'activation joue un rôle très important dans le comportement du neurone. Elle retourne une valeur représentative de l'activation du neurone, cette fonction a comme paramètre la somme pondérée des entrés ainsi que le seuil d'activation la nature de cette fonction diffèrent selon le réseau. On en compte divers types, parmi elles :

a) Fonction binaire à seuil

Fonction à seuil montrée par la figure I.3(a) et définie par :

$$h(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases} \qquad (I.1)$$

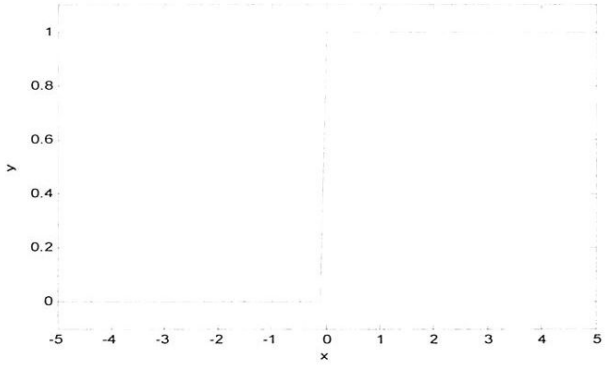


Figure I.3(a) : Fonction binaire à seuil.

b) Fonction signe

La fonction signe, montrée par la figure I.3(b), et définie par :

$$\text{sgn}(x) = \begin{cases} 1 & \text{si } x > 0 \\ -1 & \text{sinon} \end{cases} \qquad (I.2)$$

Le seuil introduit une non-linéarité dans le comportement du neurone, cependant, il limite la gamme des réponses possibles à deux valeurs.

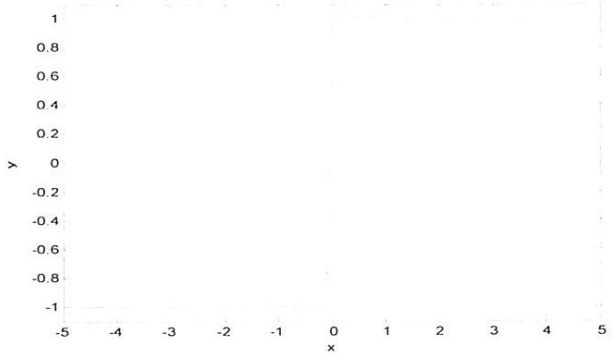


Figure I.3(b) : Fonction signe.

c) Fonction linéaire

C'est l'une des fonctions d'activations les plus simples, cette fonction est représentée par la figure I.3(c), sa fonction est définie par :

$$f(x) = x \qquad (I.3)$$

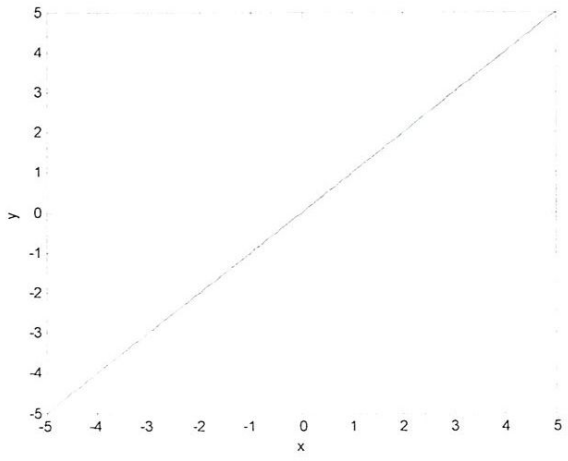


Figure I.3(c) : Fonction lineaire.

d) Fonction linéaire à seuil ou multi-seuils

On peut la définir comme suit :

$$h(x) = \begin{cases} x & \text{si } x \in [u, v] \\ v & \text{si } x < v \\ u & \text{si } x > v \end{cases} \qquad \text{(I.4)}$$

Cette fonction représente un compromis entre la fonction linéaire et la fonction seuil. Son graphe est représenté par la figure I.3(d) : entre ses deux barres de saturation, elle confère au neurone une gamme de réponses possibles. En modulant la pente de la linéarité, on affecte la plage de réponse du neurone.

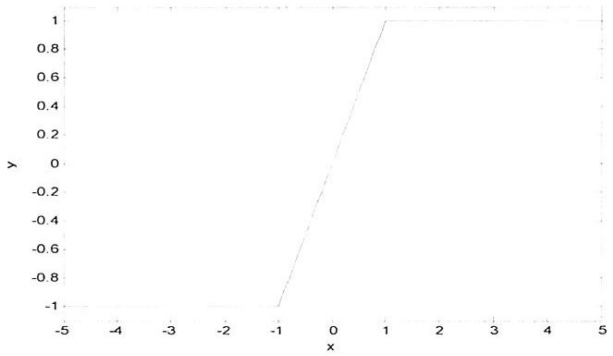


Figure I.3(d) : Fonction linéaire a seuil.

e) Fonction sigmoïdale

Elle est l'équivalent continu de la fonction linéaire représentée par la figure I.3(e). Etant continue, elle est dérivable, d'autant plus que sa dérivée est simple à calculer.

La fonction log sigmoïde est définie par :

$$\text{logsig}(x) = \frac{1}{1+e^{-x}} \qquad \text{(I.5.a)}$$

Et sa dérivée est :

$$\frac{d}{dx}(\text{logsig}(x)) = \text{logsig}(x)\,(1 - \text{logsig}(x)) \qquad \text{(I.5.b)}$$

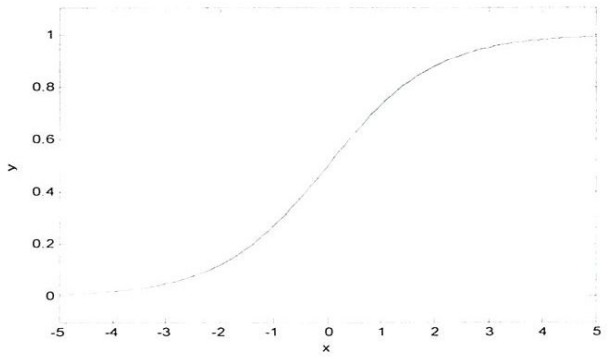


Figure I.3(e) : Fonction log sigmoïde.

f) Fonction tangente sigmoïde

La fonction tangente sigmoïde est celle montrée par la figure I.3(f). Elle est définie par :

$$\text{tansig(x)} = \frac{2}{(1+ e^{-2x})} - 1 \qquad \text{(I.6.a)}$$

Et sa dérivée est définie comme suit :

$$\frac{d}{dx}(\text{tansig(x)}) = \frac{4e^{-2x}}{(e^{-2x}+1)^2} \qquad \text{(I.6.b)}$$

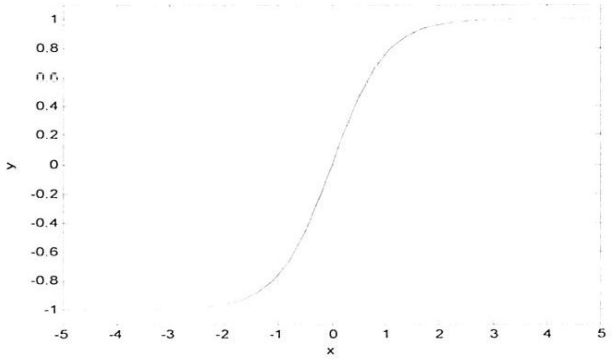


Figure I.3(f) : Fonction tangente sigmoïde.

I.5. Architecture des réseaux de neurones :

On distingue deux grands types d'architectures de réseaux de neurones : les réseaux de neurones non bouclés et les réseaux de neurones bouclés.

I.5.1. Les réseaux de neurones bouclés :

Les réseaux de neurones bouclés peuvent avoir une topologie de connections quelle conque, comprenant notamment des boucles qui ramènent aux entrées la valeur d'une ou plusieurs sorties.

La figure I.4 montre un exemple d'un réseau de neurone bouclé.

Pour chaque système soit causal, il faut évidemment qu'a toute boucle soit associe un retard. Un réseau de neurone bouclé à temps discret est donc régi par une ou plusieurs équations différentielles non linéaires, résultant de la composition des fonctions réalisées par chacun des neurones et des retards associés à chacune des connexions.

La forme la plus générale des équations régissant un réseau de neurones bouclé et appelée forme canonique :

$$x(k+1) = \varphi[x(k), u(k)] \tag{I.7}$$

$$y(k) = \psi[x(k), u(k)] \tag{I.8}$$

Où $x(k)$ est le vecteur des variables d'état à l'instant (discret) kT, $u(k)$ est le vecteur des entrées, $y(k)$ est le vecteur des sorties.

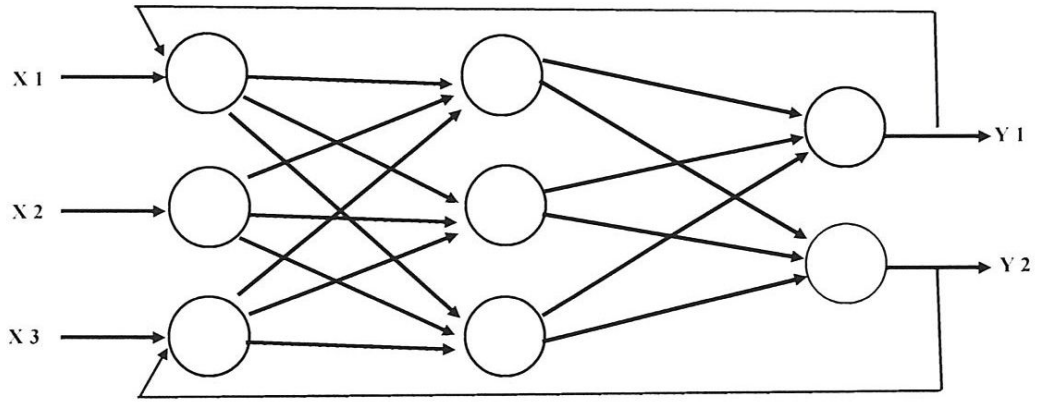


Figure I.4 : Réseaux de neurones bouclés.

I.5.2. Les réseaux de neurones non bouclés :

Un réseau de neurones non bouclé réalise une (ou plusieurs) fonctions algébriques de ses entrées, par composition des fonctions réalisées par chacun de ses neurones.

Un réseau de neurones non bouclé a une structure particulière, très fréquemment utilisée : il comprend une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Les neurones de chaque couche ne sont pas connectés entre eux. Cette structure est appelée perceptron multicouche. La figure I.5 représente un exemple de réseau non bouclé avec une seule couche cachée.

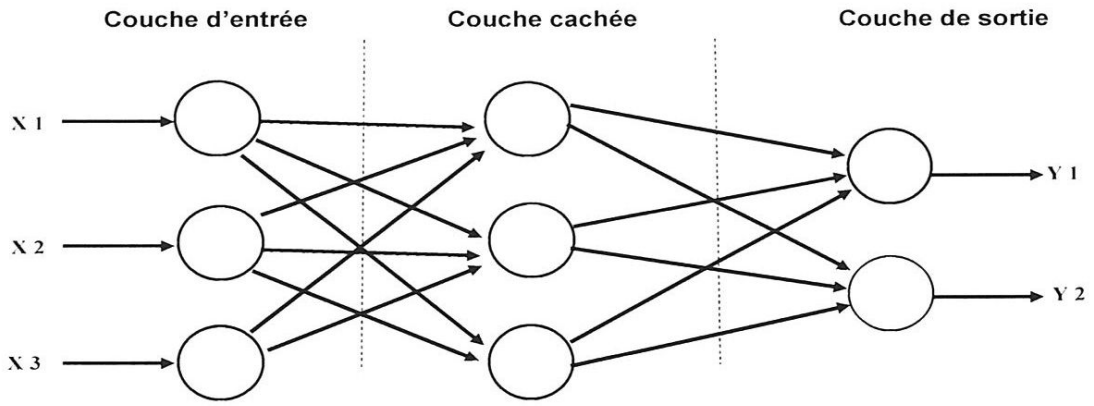


Figure I.5 : Réseaux de neurones non bouclés.

I.6. Apprentissage des réseaux de neurones :

I.6.1. Définition

L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples d'apprentissage.

Dans les algorithmes actuels, les variables modifiées pendant l'apprentissage sont les poids des connexions.

I.6.2. Protocoles d'apprentissage

Presque la totalité des réseaux de neurones ont en commun un même protocole d'apprentissage celui-ci comporte quatre étapes :

Etape 1 : Initialisation des poids synaptiques avec des petites valeurs aléatoires.

Etape 2 : Présentation de l'exemple d'entrée et propagation de l'activation des neurones.

Etape 3 : Calcule de l'erreur. Dans le cas d'un apprentissage supervisé cette erreur dépend de la différence entre l'activation des neurones et la sortie désirée.

Etape 4 : Calcul du vecteur de correction à partir des valeurs des erreurs, avec lequel on effectue la correction des poids synaptiques.

I.6.3. Règles d'apprentissage :

a) La règle de Hebb

La règle de Hebb est le premier mécanisme d'évolution proposé sur les synapses. Son interprétation pour les réseaux de neurones artificiels est la suivante :

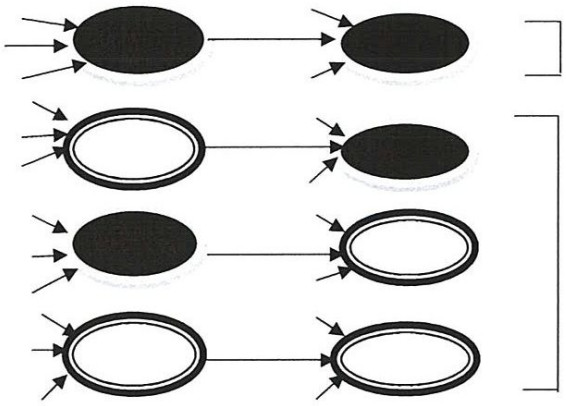
On considère que deux neurones connectés entre eux sont activé aux mêmes moments, la connexion qui les relie doit être renforcée et elle n'est pas modifiée, dans le cas contraire. C'est -à-dire que le poids Wij d'une connexion entre un neurone i et un neurone j augmente quand les deux neurones sont activés en même temps et il n'est pas modifié, dans le cas contraire.

Si nous prenons, à titre d'exemple, les conventions suivantes :

Cellule non activée

Cellule activée

La règle de Hebb donne alors :



Connexion renforcée

Connexions non renforcées

Ceci se traduit par :

Quand la cellule émettrice et la cellule réceptrice s'activent en même temps, il faut augmenter le poids de cette connexion lors de l'apprentissage. La connexion entre ces deux cellules devienne alors très forte. Si la cellule émettrice s'active sans que la cellule réceptrice ne le soit, ou si la cellule réceptrice s'active alors que la cellule émettrice n'était pas activée, cela traduit bien le fait que la connexion entre les deux n'est pas prépondérante dans le comportement de la cellule réceptrice on peut donc dans la phase d'apprentissage laisser un poids faible à cette connexion.

En se basant sur ce principe, Hebb a donné la règle d'apprentissage suivante :

$$Wij(t + \partial t) = Wij(t) + \mu AiAj \qquad (I.9)$$

Avec Wij (t) et $Wij(t + \partial t)$: les poids de la connexion entre le neurone i et le neurone j aux instants(t)et$(t + \partial t)$.

Ai Et Aj : L'activation du neurone i et l'activation du neurone j

μ : C'est le paramètre de l'intensité de l'apprentissage$(\mu > 0)$.

b) La règle de Widrow-Hoff

La règle d'apprentissage de Widrow-Hoff, ou des moindres carrés (LMS, Least Square Sum), est une règle d'apprentissage supervisé basée sur la correction d'erreurs observées en sortie. Cette règle consiste à minimiser une fonction coût caractérisée par l'erreur quadratique moyenne. Pour un ensemble d'apprentissage contenant Q paires entrée/sortie désirée $\{(X^{(q)}/T^{(q)})\}$, $q = 1, \dots Q$ où $X^{(q)}$ et $T^{(q)}$ représentent respectivement la $q^{ème}$ entrée et la $q^{ème}$ sortie désirée, l'erreur $(e(r))$ à l'itération r est donnée par :

$$e(r) = T(r) - Y(r) \tag{I.10}$$

Où : $Y(r)$ est la sortie calculée du réseau. La fonction coût est :

$$F(X) = e^2(r) \tag{I.11}$$

L'apprentissage selon la règle LMS consiste à calculer le gradient à chaque présentation d'un exemple d'apprentissage. Le changement de poids est alors :

$$\Delta w_{ij}(t) = -\eta \nabla F(X) \tag{I.11.a}$$

$$= -\eta \frac{\partial e^2(r)}{\partial w_{ij}} \tag{I.11.b}$$

Cette règle de correction permet donc aux neurones d'adapter leurs poids pour se rapprocher à une valeur désirée correspondante à chaque exemple présenté. Cette règle a été utilisée pour l'apprentissage de l'ADALINE dans lequel chaque neurone i corrige ses poids w_{ij} à l'itération r selon l'équation suivante :

$$\Delta w_{ij}(r) = \Delta w_{ij}(r-1) - \eta(t_i - y_i)x \tag{I.12}$$

Où : t_i et y_i sont respectivement la sortie désirée et la sortie calculée correspondantes au neurone $\mathbf{i}$; x est l'entrée et η est une constante positive appelée pas d'apprentissage.

I.6.2. Les types d'apprentissage :

Les techniques d'apprentissage se subdivisent en trois grandes familles :

a) Apprentissage supervisé

Pour ce type d'apprentissage (perceptron, Adaline, etc…), le réseau doit savoir qu'il a commis une erreur et il doit connaitre la réponse qu'il dû donner. Pour un stimulus présenté à la couche d'entrée du réseau, la réponse obtenue est comparée avec celle désirée et la modification et les ajustements à apporter aux poids sont déterminés en fonction de l'erreur commise par le réseau. Généralement, les règles d'apprentissage supervisé sont des formes de descente du gradient.

L'apprentissage supervisé nécessite donc la définition d'une base d'exemples d'apprentissage représentative. Chaque exemple présenté au réseau est un couple (entrée, sortie désirée). La minimisation de l'erreur entre la valeur de sortie et la valeur désirée est basée sur le principe de l'erreur quadratique.

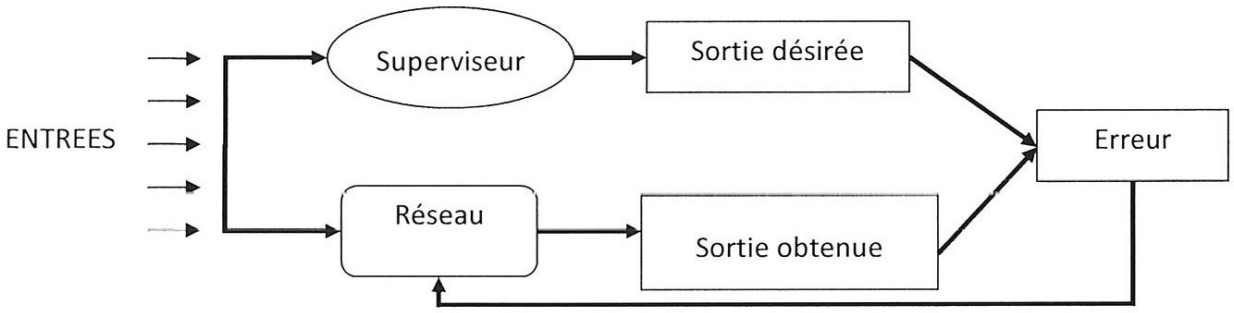


Figure I.6(a) : Apprentissage supervise.

b) Apprentissage semi-supervisé

L'apprentissage semi-supervisé suppose qu'un comportement de référence précis n'est pas disponible, mais qu'en revanche, il est possible d'obtenir des indications qualitatives (correcte/incorrecte) sur les performances du réseau.

c) Apprentissage non supervisé

Pour ce type d'apprentissage il s'agit d'atteindre l'ensemble des poids synaptiques pour lesquels le comportement du réseau est optimal. La modification et l'ajustement des poids se font en fonction d'un critère interne, indépendant de la relation entre le comportement du réseau et la tâche qui doit effectuer.

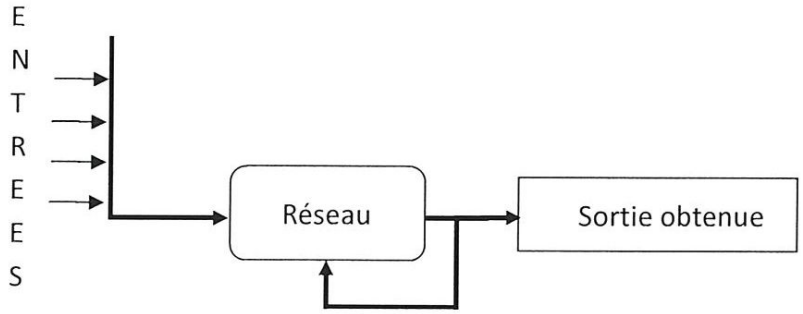


Figure I.6(b) : Apprentissage non supervisé.

I.7. Avantages et inconvénients des réseaux de neurones :

I.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.
- Résistance au bruit ou au manque de fiabilité des données.
- Simple à manipuler, beaucoup moins de travail personnel à fournir que dans l'analyse statistique classique. Ne demande pas de grandes compétences en mathématique, informatique ou statistique.
- 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 multi variables.

I.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 sont incompréhensibles pour l'utilisateur.

CHAPITRE II

COMMANDE PAR LES RÉSEAUX DE NEURONES

II.1. Introduction :

Dans de nombreuses applications du monde réel, il existe beaucoup de non-linéarités, des dynamiques difficile a modélisé, des bruits non mesurable, multi-retours, etc. Qui posent des problèmes aux ingénieurs en essayant de mettre en œuvre des stratégies de contrôle. Dernièrement, le développement de nouvelles stratégies de contrôle reposait en grande partie sur des théories de contrôle modernes et classiques. La théorie du contrôle moderne, telle que les techniques de contrôle adaptatif et optimal et la théorie classique de contrôle, repose principalement sur la linéarisation des systèmes.

Dans l'application de ces techniques, le développement de modèles mathématiques est d'une nécessité préalable. Cependant, une telle modélisation mathématique qui repose en grande partie sur les hypothèses de linéarisation des systèmes pourrait ne pas refléter les vraies propriétés physiques du système. Même si des modèles mathématiques complexes qui reflètent avec précision les relations physiques entrées-sorties du système peuvent être construits, la sensibilité des paramètres devrait être faible afin de rendre le système de contrôle utile. Ceci est plutôt difficile, sauf dans le cas des modèles linéairesable.

De ces faits, les spécifications requises pour les applications réelles des théories de contrôle nécessitent que les algorithmes de contrôle doivent être assez simples pour être mis en œuvre et être compris. Ces algorithmes devraient également avoir les propriétés suivantes, telles que la capacité d'apprentissage, la flexibilité, la robustesse et la non-linéarité. L'une des raisons pour lesquelles le contrôle flou est devenue si populaire maintenant qu'il est satisfaisant certains de ces facteurs.

Les réseaux de neurones sont actuellement aussi devenus populaire dans le domaine de contrôle. En effet, ils ont prouvé leur succès dans la résolution de problème de reconnaissance de formes. Les réseaux de neurones ont la capacité d'apprendre des fonctions entrées-sorties, donc ils fournissent des solutions plus simples aux problèmes complexes de contrôle. En outre, les neurones sont des éléments non linéaires et, par conséquent, les réseaux neuronaux sont fondamentalement des systèmes non linéaires qui peuvent être utilisés pour apprendre et résoudre des problèmes de contrôle non linéaire, ou les méthodes de contrôle traditionnelles et conventionnelles n'ont pas encore de solution. Ainsi, ces dernières années, le contrôle intelligent en général a été facilement acceptable pour les applications de contrôle réelles.

II.2. Motivations de la commande neuronale :

Le control neuronal est défini comme l'utilisation de réseaux neuronaux bien spécifiés - artificiels ou naturels - pour émettre des signaux de contrôle réels. Habituellement, lorsque nous parlons de «contrôle», il existe une forme d'intelligence associée au terme. Nous devons nous contrôler de faire des choses illégales.

A la fin des années 80, l'algorithme de la retro-propagation a été reconnu comme un outil efficace pour réaliser le mécanisme d'apprentissage. Dans presque tous les domaines de l'ingénierie, les chercheurs ont commencé à s'engager activement dans les applications des réseaux de neurones dans l'intention de trouver de meilleures solutions aux méthodes conventionnelles. Dans le domaine du contrôle, les réseaux de neurones ont été jugés aptes à résoudre des problèmes de contrôle non linéaires et complexes, où les méthodes de contrôle conventionnelles et traditionnelles n'ont pas encore de solution pratique. Du point de vue des auteurs, il existe plusieurs raisons qui ont motivé l'intérêt de recherche dans l'application des réseaux neuronaux dans le domaine de contrôle, en tant qu'alternatives aux méthodes de contrôle traditionnelles. Parmi ces raisons [3] :

II.2.1. Capacités d'apprentissage :

Les réseaux de neurones peuvent être entrainés pour apprendre toute fonction à condition que des informations suffisantes soient fournies pendant le processus d'apprentissage et que des modèles neuronaux soient judicieusement choisis. Ainsi, cette capacité d'auto-apprentissage des réseaux neuronaux élimine l'utilisation d'analyses mathématiques complexes et difficiles, nécessaires dans de nombreuses méthodes traditionnelles de contrôle adaptatif et optimal.

II.2.2. La non-linéarité :

L'incorporation de la fonction d'activation sigmoïdale semi-linéaire (ou de certaines fonctions non linéaires générales) dans les neurones cachés des réseaux de neurones multicouches offre une capacité de calcul non linéaire pour résoudre des problèmes de contrôle hautement non linéaires où les approches de contrôle traditionnelles n'ont pas encore de solution pratique. Surement, c'est l'avantage le plus important d'utiliser les réseaux de neurones du point de vue théorique de contrôle.

II.2.3. Non nécessité de connaissances a priori :

L'exigence d'une vaste information a priori concernant le système à contrôler, telle que la modélisation mathématique, est une nécessité préalable dans les techniques de contrôle adaptative et optimales traditionnelles avant qu'elles ne puissent être mises en œuvre. En raison de la capacité d'auto-apprentissage des réseaux neuronaux, de telles informations ne sont pas nécessaires pour les contrôleurs neuronaux. Ainsi, ces contrôleurs semblent pouvoir contrôler dans une gamme d'incertitude plus large.

II.2.4. Calcul parallèle :

Le parallélisme massif des réseaux de neurones offre de très grand capacité de calcul surtout lorsqu'elle est mise en œuvre à l'aide de puces neuronale ou de matériel parallèle. Par exemple, American Neurologix Inc. Ont développé un processeur neuronal (NLX420) déjà commercialisé dans les années 90s.

II.3. Approches de la commande neuronale :

II.3.1. Approches avant la retro propagation :

L'un des premiers travaux les plus importants sur le contrôle neuronal est celui de Widrow et Smith [4]. Ils ont montré que leur ADALINE est capable d'équilibrer un système de réception en copiant un contrôleur humain existant en forme de «contrôle supervisé». Albus [2] a proposé une nouvelle approche de contrôle d'un manipulateur (bras robotique) en utilisant une forme de réseau de neurones a base de table de consultation connue sous le nom de contrôleur d'articulation du modèle cérébelleux (CMAC, Cerebellar Model Articulation Controller). Miller et al. [6] Ont appliqué le CMAC dans le contrôle des robots et d'autres applications en temps réel à l'Université du New Hampshire. En 1983, Barto et al. [7] Ont proposé un système d'apprentissage adaptatif consistant en un élément de recherche associatif unique (ASE, single associative search element) et un élément critique adaptable (ACE, single adaptive critic element) appliqué pour équilibrer un système de pendule inversé comme indiqué dans le schéma synoptique simplifié de la figure II.1. Plusieurs variantes de cette approche d'affectation de crédit ont été étudiées et appliquées de manière indépendante par d'autres chercheurs.

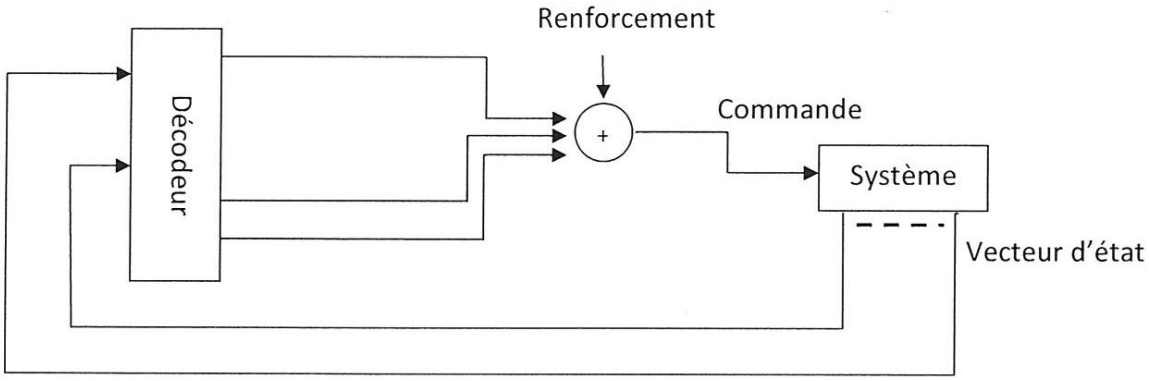


Figure II.1 : Méthode critique adaptative.

II.3.2. Commande neuronal série avec modèle inverse :

Le succès de l'algorithme de back-propagation pour entrainer des réseaux multicouches a crée une varie explosion sur l'application des réseaux neuronaux dans le domaine de contrôle. Différents schémas de contrôle neuronal basé sur l'algorithme de back-propagation ont été proposés, l'un des systèmes les plus appliqués est l'approche du modèle inverse direct. La popularité de cette approche est à cause de sa simplicité. Une fois qu'un réseau neuronal a été entrainé pour apprendre l'inverse d'un système à contrôler, il peut alors être configuré pour contrôler directement ce système.

L'idée de cette approche était peut-être empruntée du système auto-adapté traditionnel de contrôle [3] où le signal de contrôle du système peut être obtenu à partir de son modèle mathématique inverse en donnant la sortie désirée du système.

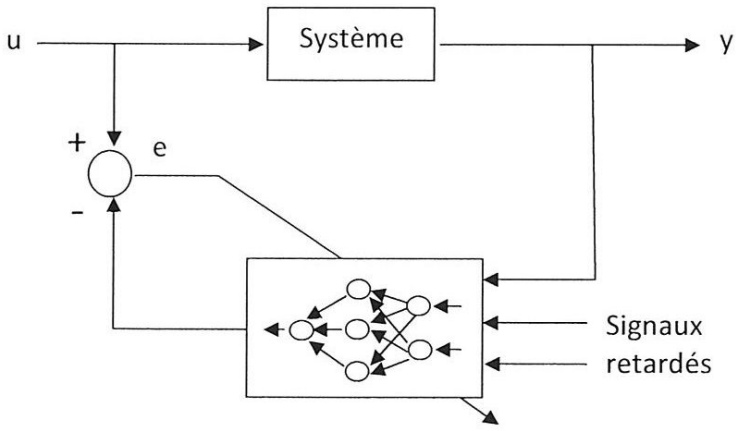


Figure II.2 : Architecture d'apprentissage généralisé.

Deux approches de commande par modèle inverse utilisant l'algorithme de la retro-propagation, appelé architecture d'apprentissage généralisée et spécialisée, ont été proposée [7].

Dans l'architecture d'apprentissage généralisée, comme le montre la figure II.2, le réseau est entrainé hors ligne en utilisant des données obtenues à partir des caractéristiques en boucle ouvertes ou fermées du système. Cette méthode est similaire à l'apprentissage par paquet dans les problèmes de reconnaissance des formes. Une fois entrainé, le réseau est configuré en tant que contrôleur direct dans un système conventionnel de contrôle de a rétroaction.

Dans la deuxième approche, l'architecture d'apprentissage spécialisée voir figure II.3, le réseau est entrainé de manière en ligne (ou «dirigé par objectif») où l'erreur est retro-propagée a travers le réseau à chaque échantillon. Le modèle de réseau neuronal est formé en ligne pour apprendre le modèle inverse du système par une rétro-propagation de l'erreur.

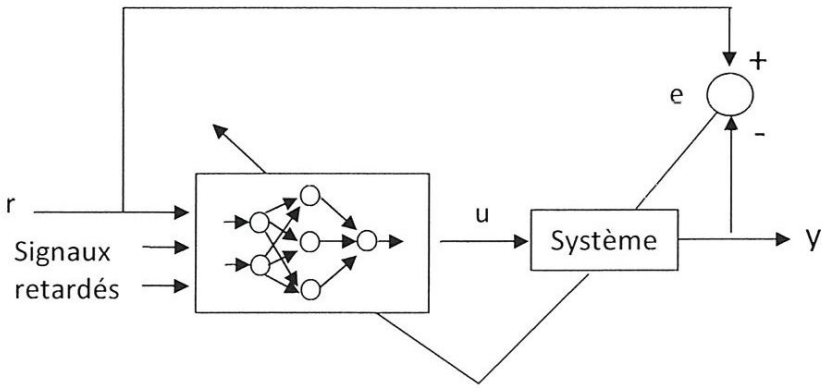


Figure II.3 : Architecture d'apprentissage spécialisée.

II.3.3. Commande neuronale parallèle avec modèle inverse :

Une autre approche de commande neuronale avec modèle inverse basée sur l'apprentissage en ligne en utilisant la retro-propagation est la commande parallèle appelée « apprentissage avec retour de l'erreur » de Kawato et al. [8] [9]. Dans cette approche, illustrée sur la figure II.4, le réseau de neurone est configuré en parallèle avec un contrôleur conventionnel à rétroaction. Le réseau est entrainé en ligne en répétant les cycles de trajectoires désirées où l'erreur de rétroaction est inversée à travers le réseau. La convergence est obtenue lorsque le réseau neuronal a appris l'inverse du système et prend alors le contrôle et élimine l'effet du contrôleur de rétroaction.

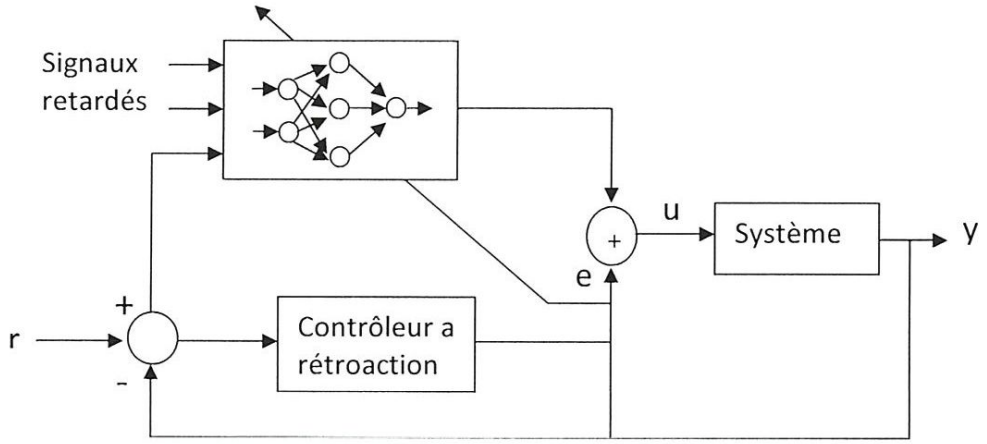


Figure II.4 : Commande parallèle basée sur l'apprentissage avec retour de l'erreur.

II.3.4. Commande neuronale indirect :

Dans cette approche, deux réseaux de neurones sont utilisés pour contrôler le système comme le montre la figure II.5. Le premier réseau de neurone est utilisé comme un modèle du système et l'autre comme contrôleur. Le réseau de neurones du modèle peut-être entrainé soit hors ligne en utilisant l'architecture d'apprentissage généralisée ou même en ligne.

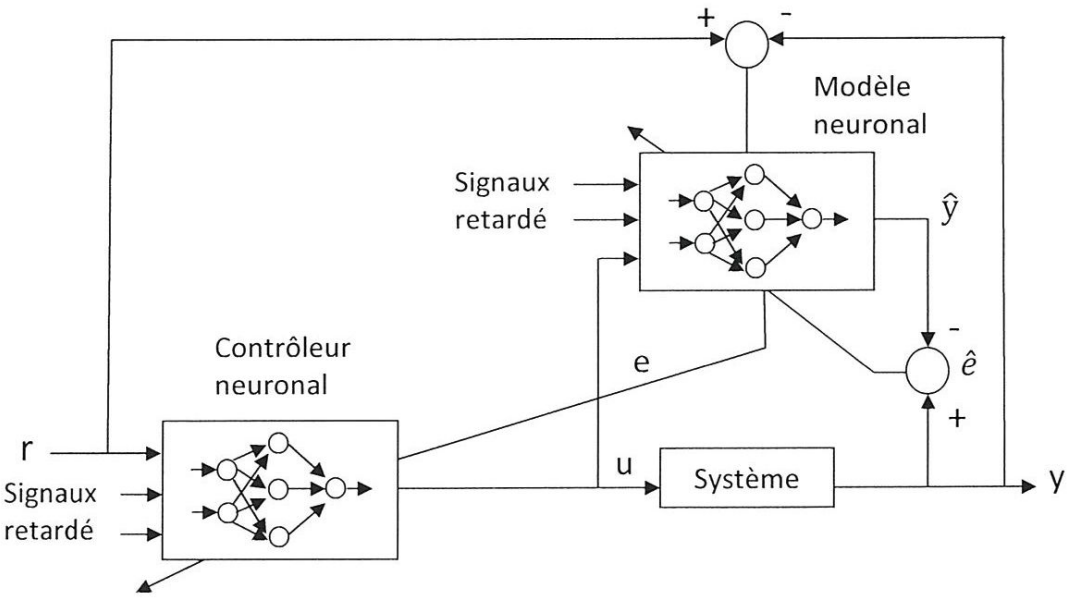


Figure II.5 : Commande neuronale indirect avec modèle et contrôleur neuronaux.

II.3.5. Commande neuronale auto-ajustable :

La configuration du contrôle neuronal auto-ajustable est illustrée sur la figure II.6 où un réseau neuronal est utilisé pour régler les paramètres d'un contrôleur conventionnel similaire au réglage effectué par un opérateur humain.

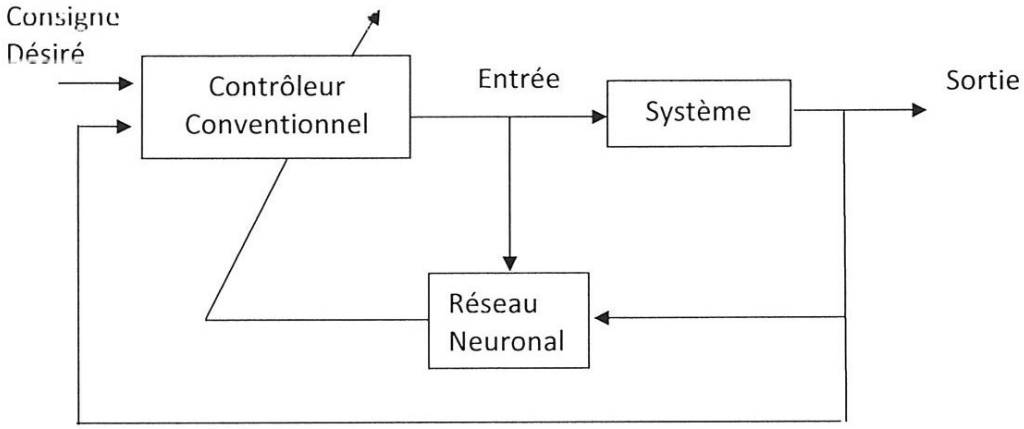


Figure II.6 : Configuration générale de la commande neuronale auto-ajustable.

Dans ce cas d'un opérateur humain, celui-ci a accumulé de l'expérience et des connaissances sur le système de contrôle, cependant, contrairement à un ordinateur, il est plutôt impossible pour l'opérateur de stocker l'historique des données antérieures du système pour toutes sortes de conditions de fonctionnement. Un ordinateur peut stocker ces informations facilement et

récupérer tout de suite. Par conséquent, si nous pouvons inclure l'expérience et les connaissances de l'opérateur dans un réseau neuronal et le former en fonction de l'historique des données passées, le réseau neuronal qualifié pourrait être utilisé comme moyen de régler en ligne les paramètres du contrôleur.

Cette approche a des applications directes sur de nombreuses techniques de contrôle traditionnelles qui incluent des méthodes de contrôle adaptatif. De nombreuses méthodes de contrôle adaptatif ont un certain nombre de paramètres définis par l'utilisateur qui doivent être sélectionnés ou réglés précédemment. Ceux-ci sont généralement choisis par essai et erreur. En intégrant un réseau neuronal dans la chaine de contrôle, il peut ensuite être utilisé pour régler ces paramètres d'une manière en ligne. Ainsi, cette stratégie de commande neuronale autonome a des applications possibles dans de nombreuses approches de contrôle traditionnelles.

II.3.6 Ajustement neuronale des paramètres d'un contrôleur PID :

Les contrôleurs PID ont une longue histoire dans l'ingénierie de contrôle et ils se sont avérés robustes, simples et stables pour de nombreuses applications réelles. Par exemple dans les années 90, il a été rapporté que le pourcentage des contrôleurs PID qui ont été utilisé au Japon dans des problèmes de contrôle réel est d'environ 84%. Ceci montre que les contrôleurs PID est très acceptable pour de nombreuses applications réelles en raison de sa simplicité de structure et de son principe facilement compréhensible.

Un contrôleur PID comporte trois paramètres : le gain proportionnel, la constante de temps intégral, la constante de temps dérivé. Ces paramètre sont ajusté afin d'obtenir la sortie souhaitée.

La figure II.7 montre une structure générale d'auto-ajustement des paramètres d'un PID où le réseau neuronal est utilisé à la place des opérateurs humains, de sorte que la fonction d'erreur est minimisée en ajustant les gains.

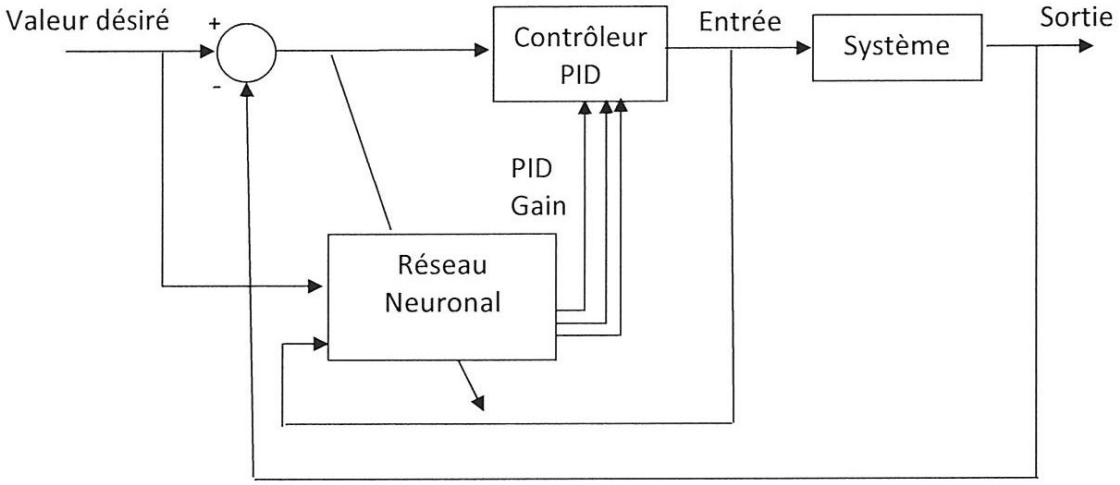


Figure II.7 : Configuration générale d'ajustement des paramètres d'un PID avec un réseau de neurones.

II.4. Commande neuronale directe avec modèle inverse :

Le contrôle neuronal direct avec modèle inverse (figure II.8), également appelé contrôle neuronal en série, est un système dans lequel le réseau de neurones permet de réaliser la dynamique inverse du système [10] [11].

Figure II.8 : Diagramme bloc de la commande neuronale directe avec modèle inverse.

Ainsi, si la fonction de transfert du système est G(u) alors :

$$y = G_s(u) \tag{II.1}$$

Alors, le réseau de neurones réalise la fonction inverse comme suit

$$u = G_s^{-1}(y) \tag{II.2}$$

Ou G_s^{-1} signifie l'inverse de G_s

Ainsi, si nous donnons le signal de référence r au réseau neuronal, alors la sortie y du système deviendrait r, c'est-à-dire :

$$y = G_s(u) = G_s\left(G_s^{-1}(r)\right) = r \qquad \text{(II.3)}$$

Où u est l'entrée du système correspondant au signal de référence r et y est la sortie du système pour u.

Pour réaliser cette configuration, nous pouvons utiliser le réseau neuronal de la figure II.9. Ce réseau sera entrainé e sorte que l'erreur quadratique soit minimisée. Si l'erreur devient presque égale à zéro, alors le réseau neuronal réalise la dynamique inverse du système.

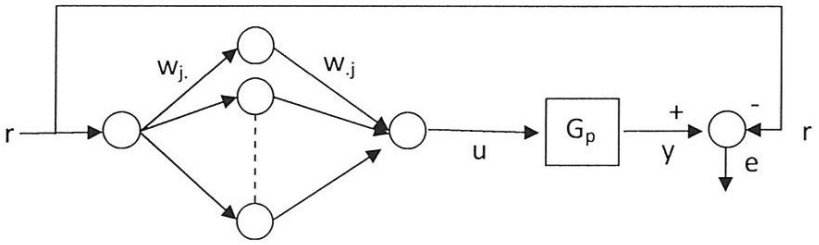


Figure II.9 : Commande basée sur un réseau avec une seule entrée.

Ce réseau de neurones a la structure interne la plus simple où un seule neurone dans les couche d'entrée et un seul neurone à la couche de sortie. L'entrée du réseau neuronal est r et sa sortie est u. La sortie du système doit être aussi proche que possible à r après l'apprentissage. Ainsi, la fonction d'erreur à minimiser est :

$$E = \frac{1}{2}(r - y)^2 \qquad \text{(II.4)}$$

Dans les applications réelles, les systèmes a contrôlé sont généralement dynamiques or le réseau de neurone de la figure II.9 est plutôt statique. Par conséquent, on doit fournir au réseau neuronal de nombreuses entrées additionnelles qui représente des informations passées. Ces entrées additionnelles sont des entrées passées et aussi des sorties passées du réseau. La figure II.10 représente un réseau avec p commandes passées et q sorties passées.

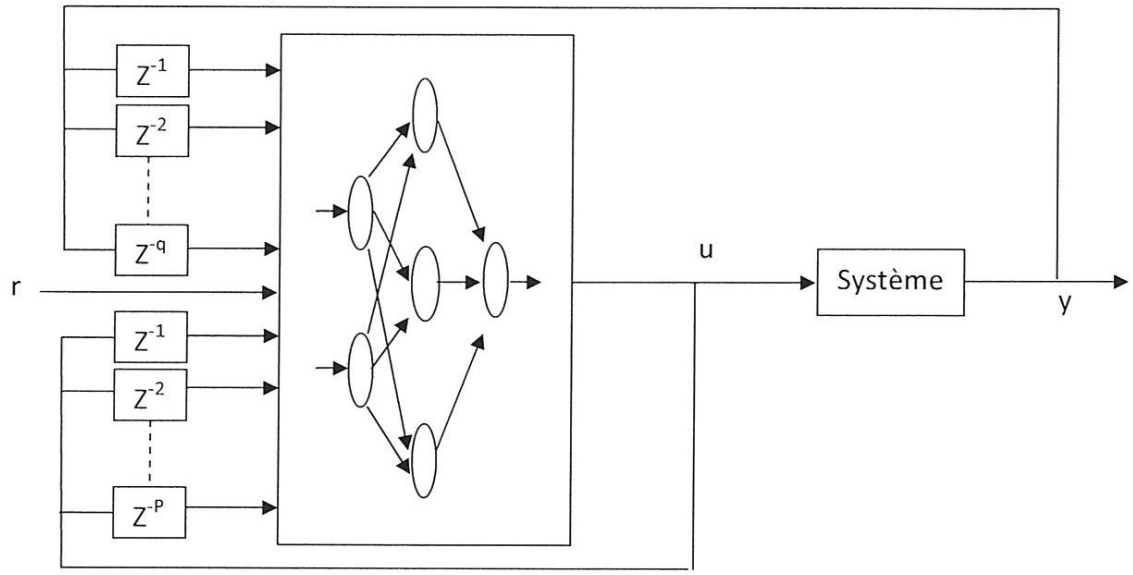


Figure II.10 : Commande basée sur un réseau avec plusieurs entrées : sortie et commande retardés.

II.5.Conclusion :

La plupart des systèmes de contrôle neuronal sont généralement basés sur les cinq approches de conception suivantes [3]:

i. Dans le contrôle en série, le réseau de neurone apprend comment générer directement les signaux de commande (qui seront utilisé comme entrées du système) en utilisant les signaux de référence désirés. Les signaux de commande obtenue permettent donc au système de produire les signaux de références.

ii. Dans les approches de contrôle parallèle, un réseau de neurones est utilisé pour compenser le signal de commande est fourni par un contrôleur conventionnel de sorte que la sortie du système puisse suivre la sortie souhaitée le plus près possible.

iii. Dans l'approche d'autoréglage, un réseau de neurone ajuste les paramètres de contrôle inclus dans un contrôleur conventionnel de sorte que la sortie du système suit le plus possible le signal de sortie souhaité.

iv. Dans l'approche basée sur un modèle et contrôleur, le réseau de neurone maximise une certaine mesure de performance au fil du temps.

v. Dans l'approche critique adaptative, on réalise un contrôle analogue au contrôle optimal dans un environnement avec bruits.

CHAPITRE III

APPLICATIONS

III.1 Introduction :

L'objectif de ce travail est d'exploiter les capacités d'apprentissage, de calcul parallèle et d'approximation des réseaux de neurones pour reproduire la dynamique inverse des systèmes non-linéaires, puis à l'utilisation du modèle neuronal inverse ainsi formé comme étant un contrôleur direct de ce système.

Dans ce chapitre nous appliquons cette méthode en utilisant un réseau de neurones de type perceptron multicouches qui est le réseau de neurones le plus simple et le plus utilisé. Nous effectuons des tests de simulation sur système non linéaire.

III. 2. Le Perceptron Multicouches :

III.2.1. Architecture du PMC :

Le PMC est une réalisation en série de perceptrons. Ce modèle comprend en plus de la couche d'entrée et la couche de sortie, une ou plusieurs couches cachées. Dans ce réseau, chaque neurone est connecté à tous les neurones de la couche précédente et la couche suivante. Les neurones de la même couche ne sont pas connectés entre eux.

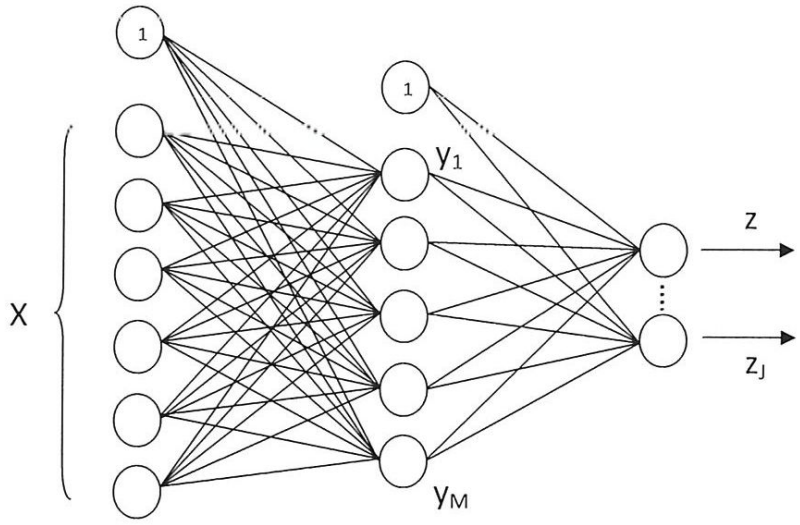


Figure III.1 : PMC avec M neurones cachées et J neurones de sorties.

La figure III.1 illustre un PMC, avec une seule couche cachée, comportant N neurones d'entrée, M neurones cachés et J neurones de sortie. Le $n^{ème}$ neurone d'entrée est relié avec le $m^{ème}$ neurone caché par le poids w_{nm} et le $m^{ème}$ neurone caché est relié avec le $j^{ème}$ neurone de sortie par le poids u_{mj}. A chaque présentation d'un exemple X $(x_1, x_2, \dots, x_N)$, les

composants de son vecteur caractéristique seront transmis aux neurones de la couche cachée. Les sorties de ces neurones $(y_1, y_2, \dots, y_M)$ seront à leur tour transmis aux neurones de la couche suivante (la couche de sortie). La sortie du $m^{ème}$ neurone caché est donnée par :

$$y_m = h(r_m) = h\left(\sum_{m=1}^{M} x_n w_{nm} + w_{0m}\right) \qquad \text{(III.1)}$$

Où : $h(.)$ est la fonction d'activation des neurones de la couche cachée.

Les neurones de la couche de sortie constituent la sortie du réseau. La sortie du $j^{ème}$ neurone est donnée par :

$$z_j = g(s_j) = g\left(\sum_{j=1}^{J} y_m u_{mj} + u_{0j}\right) \qquad \text{(III.2)}$$

Où : $g(.)$ est la fonction d'activation des neurones de sortie.

Dans les problèmes de classification, le nombre de neurones de la couche d'entrée du PMCest égal à la dimension de l'espace caractéristique parce que le rôle de cette couche est de présenter chaque composante du vecteur caractéristique aux neurones de la couche suivante.

Le nombre de neurones de la couche de sortie égale le nombre de classes, soit un neurone par classe (la sortie de ce neurone est 1 pour les exemples appartenant à la classe correspondante et les autres sont à 0).

En tant que classificateur, le PMC doit donc fournir une sortie Z $(z_1, z_2, \dots, z_J)$ correspondante à la classe de l'exemple X $(x_1, x_2, \dots, x_N)$ présenté en entrée.

Les capacités d'approximations du PMC ont été évoquées dans plusieurs travaux [2]. Dans ce contexte, il a été démontré qu'un réseau de neurones avec une seule couche cachée peut approximer n'importe quelle fonction. Par exemple, Cybenko a démontré qu'un PMC avec une seule couche cachée ayant une fonction d'activation continue et non linéaire, est suffisant pour approximer, au sens des moindres carrées, avec une erreur arbitrairement faible pour un ensemble donné d'apprentissage, n'importe quelle transformation continue représentée par cet exemple.

III.2. 2. Apprentissage du PMC par la Retro Propagation :

La rétro-propagation du gradient (Back Propagation) est l'une des méthodes les plus simples et les plus utilisées pour l'apprentissage des réseaux de neurones. C'est une extension de la règle d'apprentissage de Widrow appliquée aux réseaux monocouches. La RP (rétro-propagation) consiste donc à minimiser la distance entre la sortie calculée $Z^{(q)}$ et la sortie désirée $T^{(q)}$ correspondantes à chaque exemple d'apprentissage $X^{(q)}$. L'erreur quadratique est souvent employée comme étant la fonction coût de la RP. Pour un ensemble de Q exemples d'apprentissage, l'erreur quadratique totale est donnée par :

$$E = \sum_{q=1}^{Q} \sum_{j=1}^{J} \left(t_j^{(q)} - z_j^{(q)} \right)^2 \qquad \text{(III.3)}$$

L'algorithme de la RP est basé sur la modification des poids du réseau de façon à effectuer une descente de gradient sur la surface d'erreur. Au début de l'apprentissage, les poids sont initialisés avec des valeurs aléatoires et modifiés ensuite dans une direction qui réduira l'erreur.

La modification Δw d'un poids w est donnés par :

$$\Delta w = \eta \frac{\partial E}{\partial w} \qquad \text{(III.4)}$$

Où : η est une constante positive appelée pas d'apprentissage permettant de définir la taille des modifications des poids.

Il s'agit donc de prendre un pas dans l'espace des poids permettant de réduire la fonction coût.

Chaque poids est modifié à l'itération $(r+1)$ en fonction de sa valeur à l'itération (r) par :

$$w^{(r+1)} = w^{(r)} + \Delta w^{(r)} \qquad \text{(III.5)}$$

Pour un PMC avec une couche cachée, la mise à jour des poids de la couche cachée et ceux de la couche de sortie est donnée par :

$$u_{mj}^{(r+1)} = u_{mj}^{(r)} - \eta \frac{\partial E^{(r)}}{\partial u_{mj}} \qquad \text{(III.6)}$$

$$w_{nm}^{(r+1)} = w_{nm}^{(r)} - \eta \frac{\partial E^{(r)}}{\partial w_{nm}} \qquad \text{(III.7)}$$

Le développement de l'Eq. (III.6) et l'Eq. (III.7) donne les équations d'adaptation suivantes [1] [2]:

$$u_{mj}^{(r+1)} = u_{mj}^{(r)} + \eta_1 (t_j - z_j) \dot{g}(s_j) y_m \quad \text{(III.8)}$$

$$w_{nm}^{(r+1)} = w_{nm}^{(r)} + \eta_2 \left(\sum_{j=1}^{J} (t_j - z_j) \dot{g}(s_j) u_{mj} \right) \acute{h}(r_m) x_n \quad \text{(III.9)}$$

Où : $\dot{g}(s_j)$ et $\acute{h}(r_m)$ sont respectivement les dérivées des fonctions d'activation des neurones cachés et des neurones de sortie. Dans le cas des fonctions sigmoïde, les dérivées sont données par :

$$\dot{g}(s_j) = z_j(1 - z_j) \quad \text{(III.10)}$$

$$\grave{h}(r_m) = y_m(1 - y_m) \quad \text{(III.11)}$$

L'algorithme de la RP est donné par :

Etape 1 : introduction des données

$X^{(q)}$ (vecteurs d'entrée)

$T^{(q)}$ (vecteurs de sorties désirées)

Etape 2 : détermination des dimensions de données

N (nombre d'entrée qui sera le nombre de neurones de la couche d'entrée)

Q (nombre d'exemples d'apprentissage)

J (nombre de classes qui seront le nombre de neurones de la couche de sortie)

Initialisation des paramètres :

I (nombre d'itération)

η_1 et η_2 (pas d'apprentissage)

M (nombre de neurones de la couche cachée)

W_{mi} et W_{nm} (poids initiaux avec des valeurs aléatoires)

Etape 3 : Apprentissage

Pour i = 1 à I (pour chaque itération)

Pour q = 1 à Q (pour chaque exemple)

X = $X^{(q)}$ (introduction du vecteur d'entrée de l'exemple q)

T = $T^{(q)}$ (introduction du vecteur de sortie de l'exemple q)

Sous-programme de la mise à jour

Pour m = 1 à M

Pour n=1 à N

$$p_m = \left(\sum_{m=1}^{M} x_n w_{nm} + w_{0m} \right)$$

$$y_m = \frac{1}{1+ e^{-pm}}$$

Pour j = 1 à *J*

$$\mathrm{Sj} = \sum_{j=1}^{J} y_m u_{mj} + u_{0j}$$

$$\mathrm{Z_j} = \frac{1}{1+ e^{-sj}}$$

$$E(q) = \sum_{j=1}^{J} (t_j - y_j)^2$$

Pour m = 1 à M

Pour j = 1 à J

Ajustement de chaque poids w_{mj} selon (III.8)

Pour n = 1 à N

Ajustement de chaque poids w_{nm} selon (III.9)

$$E(i) = \sum_{q=1}^{Q} Ep(q)$$

Etape 4 : Si condition d'arrêt fin sinon retour étape 3.

III.3. Commande neuronale directe par modèle inverse :

La méthode étudiée s'effectue en trois étapes essentielles (figure III.2). Ceci comporte les étapes suivantes :

i) Création des données d'apprentissage en utilisant la SBPA.

ii) Apprentissage des réseaux de neurones afin de créer un modèle permettant de reproduire la dynamique inverse du système.

iii) L'application du modèle obtenu pour la commande du système.

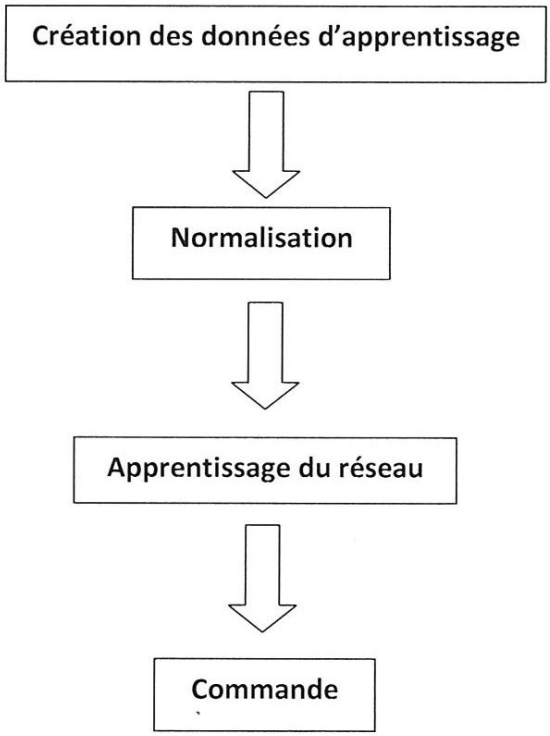


Figure III.2 : les étapes de la commande neuronale avec modèle inverse.

La figure III.3 illustre l'architecture du système de commande étudié qui réalise une commande neuronale par modèle inverse avec un PMC.

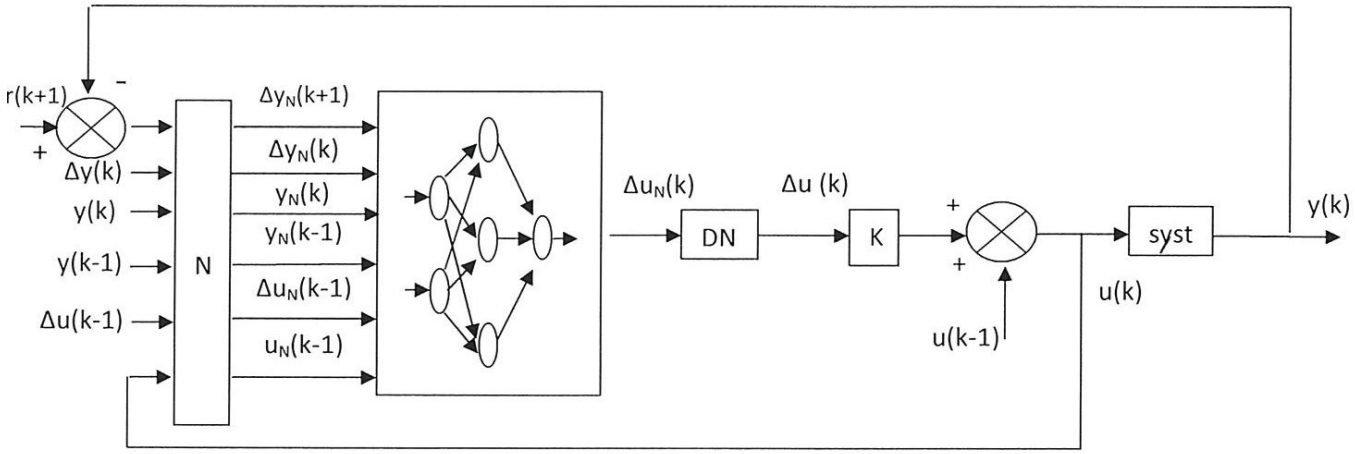


Figure III.3: Commande neuronale par modèle inverse avec un PMC.

III.4. Simulation et résultats :

Nous appliquons la méthode étudiée sur le système de Narendra et Parthasarathy qui est un processus dynamique non linéaire [12] données par :

$$y(k+1) = \frac{y(k)}{1 + y(k)^2} + u(k)^3 \qquad \text{(III.12)}$$

Ce système a été utilisé dans plusieurs travaux [13] [14]

III.4.1. Création des données d'apprentissage :

D'une façon générale, la conception d'un système de contrôle performant nécessite de connaître le modèle dynamique du procédé, qui décrit la relation entre les variations de la commande et les variations de la mesure. Dans la commande neuronale avec modèle inverse, il est aussi nécessaire de former un réseau de neurone permettant de bien modéliser la dynamique inverse du système. Pour ce faire, il faut avoir des données apprentissage suffisantes. C'est ainsi que nous utilisons la SBPA (Séquence Binaire Pseudo Aléatoire) qu'est un signal formé d'impulsions rectangulaires modulées aléatoirement en longueur, qui approxime un bruit blanc discret, donc riche en fréquence et de valeur moyenne nulle, ne modifiant pas le point de fonctionnement du procédé. La SBPA est un signal pseudo aléatoire car elle est caractérisée par une « longueur de séquence » à l'intérieur de laquelle les variations de la

largeur des impulsions varient aléatoirement, mais, sur un grand horizon de temps, elles sont périodiques. Le principe de la SBPA est illustré sur la figure III.4.

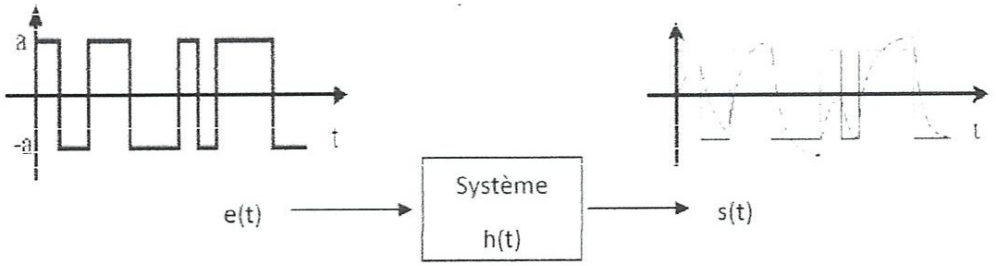


Figure III.4 : Principe de création des données avec SBPA.

La figure III.5 représente une partie de la réponse de notre système à un signal SBPA. Ces données seront utilisées pour l'apprentissage du réseau de neurones.

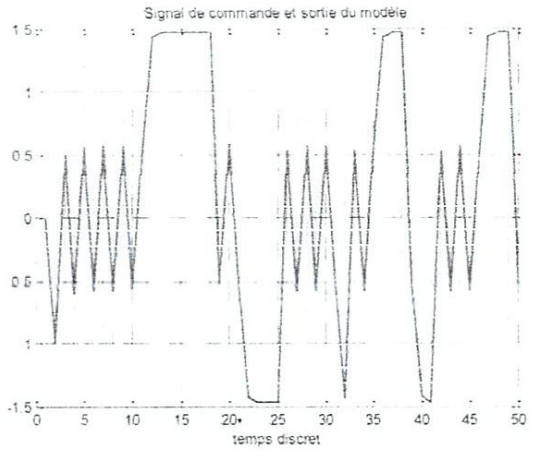


Figure III.5 : Réponse du système à la SBPA.

III.4.2. Apprentissage du réseau :

Nous utilisons un PMC avec 6 neurones de la couche d'entrée et un seul neurone de la couche de sortie (figure III.6). Pour les neurones de la couche cachée nous avons effectué plusieurs tests et nous avons constaté qu'à partir de 4 neurones cachés nous avons obtenu des résultats satisfaisants. Les 6 entrées sont : $\Delta y(k+1)$, $\Delta y(k)$, $y(k)$, $y(k-1)$, $u(k-1)$ et $\Delta u(k-1)$.

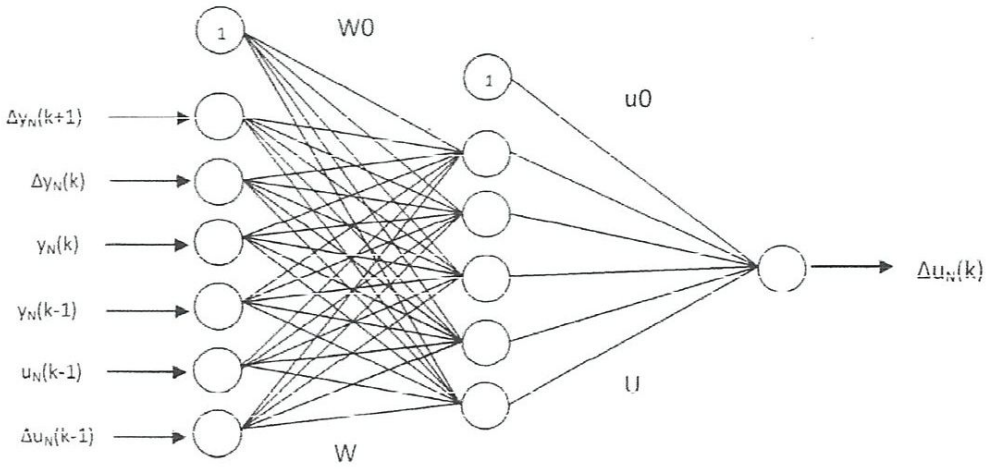


Figure III.6 : Architecture du PMC utilisé.

Nous avons effectué l'apprentissage du réseau en utilisant différentes valeurs de pas d'apprentissage (η) et nous constatons qu'à partir de 0.5 le réseau converge plus rapidement (figure III.7).

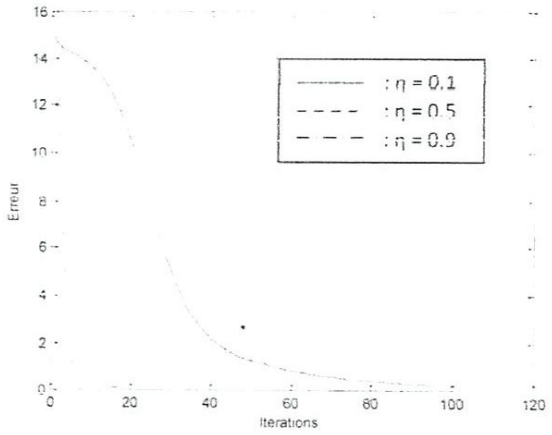


Figure III.7 : Evolution de l'erreur au cours de l'apprentissage pour η= 0.1, 0.5, 0.9

La figure III.8. Représente l'écart entre la sortie obtenue du réseau après l'apprentissage et la sortie désirée. Le réseau obtenu donne une réponse très similaire a la sortie du système.

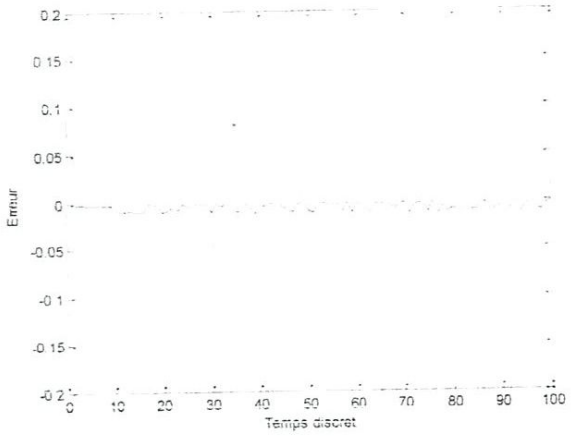


Figure III.8: Erreur entre la sortie calculée et la sortie désirée.

III.4.3. Application de la commande :

Pour évaluer les performances de commande du réseau obtenu après l'apprentissage nous appliquons le signal de référence de la figure III.9. Nous remarquons que le système à bien suivi la consigne.

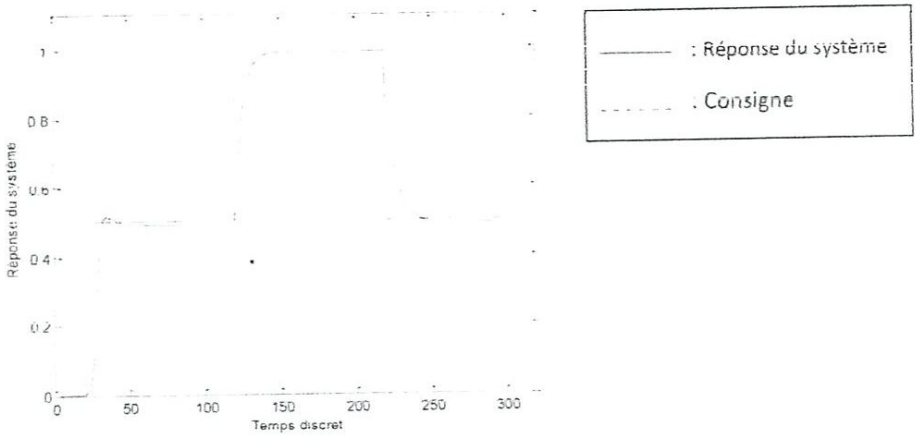


Figure III.9 : Réponse du système.

La figure III.10 : représente le signal de commande appliqué au système.

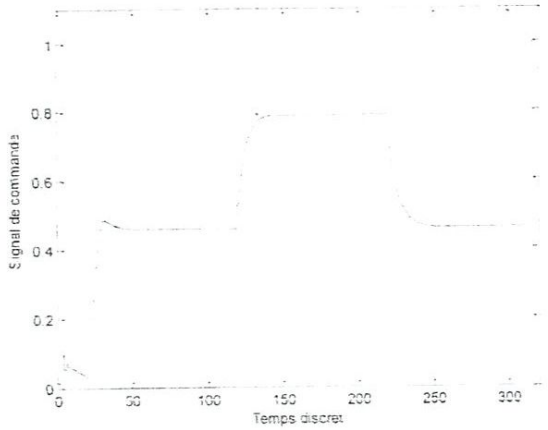


Figure III.10 : Signal de commande appliqué au système.

Pour évaluer la robustesse de notre système, nous avons réalisé un deuxième test avec des perturbations. Nous avons créé une perturbation de -0.25 à l'itération 120 et une autre de +0.25 à l'itération 350.La figure III.11 représente la réponse du système dans le cas aux perturbations, nous remarquons que le système est robuste aux perturbations.

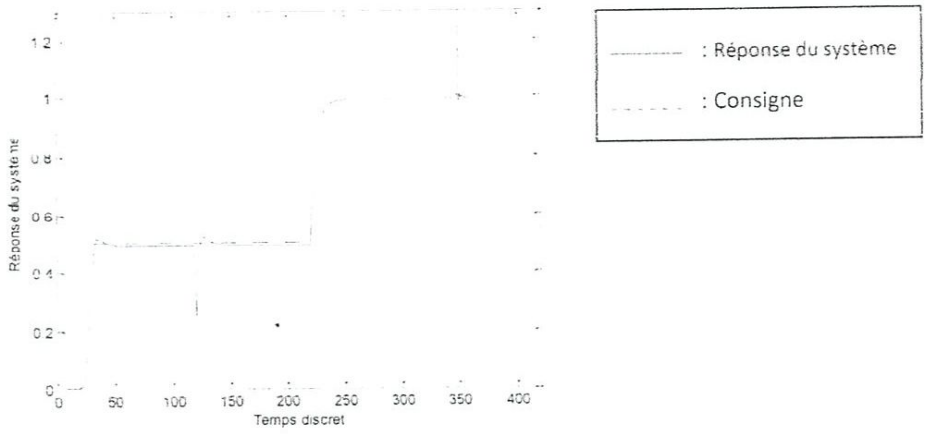


Figure III.11 : Réponse du système dans la présence de perturbations.

La figure III.12 : représente le signal de commande appliqué au système.

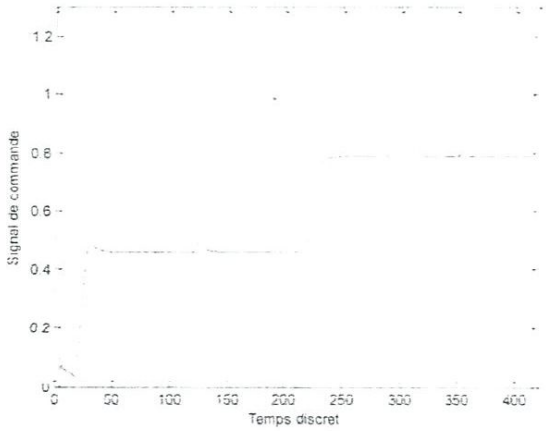


Figure III.12 : Signal de commande appliqué au système dans le cas de perturbations.

III.5 Conclusion :

Après avoir présenté l'architecture et l'apprentissage du le perceptron multicouche ainsi que les différents étapes de la commande neuronale avec modèle inverse, nous avons évalué les performances de cette commande neuronale en effectuant deux tests : sans et avec perturbations. Nous constatons que le système a bien suivi la consigne et qu'il montré de bon performance face aux perturbations.

CONCLUSION GÉNÉRALE

Conclusion générale :

L'objectif de ce travail était d'étudié la commande neuronale avec modèle inverse. L'objectif était d'exploiter les réseaux de neurones pour reproduire la dynamique inverse des systèmes puis à l'utilisation du modèle neuronal inverse ainsi formé comme étant un contrôleur direct de ce système. Nous avons utilisé perceptron multicouche qui est le réseau de neurones le plus utilisé, le plus simple et aussi qui très efficace.

Après avoir donné une introduction aux réseaux de neurones et à la commande neuronale dans les deux premiers chapitres, nous avons présenté, dans le troisième chapitre, les résultats obtenus de la commande neuronale avec modèle inverse en utilisant le perceptron multicouche sur un système non linéaire. Les résultats obtenus ont montré de bonne performance de cette méthode et une bonne robustesse aux perturbations.

BIBLIOGRAPHIE

[1] Personnaz, L. Rivals, I., "Réseaux de neurones formels pour la modélisation, la commande et la classification ", CNRS édition, Paris, 2003.

[2] Nemissi M., " Classification et reconnaissances des formes par algorithme hybrides" ; thèse de doctorat, Université du Guelma, 2009.

[3] Sigeru O., Marzuki K. and Rubiyah Y., "Neuro-Control and its Applications", Springer-Verlag, London, 1996.

[4] Widrow, B. and F.W. Smith, "Pattern-recognizing control systems", Proc. of Computer and Information Sciences, Washington D.C., Spartan, Washington, 1964.

[5] Albus, J.S., "A new approach in manipulator control: the cerebellar model articulation controller (CMAC)", Journal of Dynamic Systems, Measurement and Control, pp. 220-227,1975.

[6] Miller III, W.T., F.H. Glanz, and L.G. Kraft, "CMAC: An associative neural network alternative to backpropagation", Proc. of IEEE, Vol. 78, pp. 1561-1567, 1990.

[7] Barto, A.G., R.S. Sutton, and C.W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems", IEEE Trans. On Systems, Man, and Cybernetics, Vol. SMC-13, pp.834-846, 1983.

[8] Kawato, M., K. Furukawa, and R. Suzuki, "A hierarchical neural network model for control and learning of voluntary movement", Biological Cybernetics, Vol. 57, pp. 169-185, 1987.

[9] Kawato, M., Y. Uno, M. Isobe, and R. Suzuki, "Hierarchical neural network model for voluntary movement with application to robotics", IEEE Control Systems Magazine, Vol. 8, pp.8-16, 1988.

[10] Astrtim, K.J. and B. Witternmark, "Adaptive Control," Addison-Wesley, New York, 1989.

[11] Psaltis, D., A. Sideris, and A. Yamamura, "A Multilayered neural network controller", IEEE Control Systems Magazine, Vol. 8, pp.17-21,1988.

[12] K.S. Narendra, K. Parthasarathy, "Identification and control of dynamical systems using neural networks", IEEE Trans Neural Network, 1990, 1(1), pp. 4–27.

[13] C. F. Juang, P. H. Chang, "Designing fuzzy-rule-based systems using continuous Ant colony optimization", IEEE trans., Fuzzy Syst., vol. 18, no. 1, 2010, pp. 138-149.

[14] Talbi N.," Conception des Systèmes d'inférence Floue par des Approches Hybrides : Application pour la Commande et la Modélisation des Systèmes Non linéaires " ; thèse de doctorat, Université de Constantine, 2014.