988

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'Etude

Pour l'obtention du Diplôme de Master Académique

Domaine : **Sciences et Technologie**
Filière : **Electronique**
Spécialité : **Instrumentation**

Modélisation et Commande par les réseaux de neurone RBF : Etude sur des modèles théoriques

---

Présenté par :

KERDOUS DHIYA EDDINE

Sous la direction de :

Dr. Nemissi Mohamed

Juin 2018

REMERCIEMENTS

Au terme de ce travail, je tiens à exprimer ma profonde gratitude et mes sincères remerciements à mes tuteurs, et surtout mes collègues pour toute temps qu'ils m'ont consacré, leur directive précieuse, et pour la qualité de leur suivi durant toute la période de mon PFE.

Mes remerciements s'adressent également à mon encadrant Monsieur NEMISSI Mohammed, maître de conférences à l'Université de 08 mai 1945 de Guelma, pour avoir accepté de diriger ce travail. Son soutien, sa clairvoyance et ses compétences m'ont été d'une aide inestimable.

Je tiens à remercier sincèrement les membres du jury qui me font le grand honneur d'évaluer ce travail.

Mes plus vifs remerciements s'adressent aussi à tout le cadre professoral et administratif de département d'électronique et télécommunication.

Mes remerciements vont enfin à toute personne qui a contribué de Près ou de loin à l'élaboration de ce travail.

Résumé

L'objectif de ce travail est d'analyser les performances des réseaux de neurones à fonctions radiales dans la modélisation et la commande en se basant sur des modèles théoriques. Dans la modélisation, nous évaluons les capacités d'apprentissage de ce type de réseaux et nous analysons l'effet d'initialisation et d'adaptation de leurs paramètres. Dans la commande, nous étudions la possibilité d'utilisation de ces réseaux dans un système de commande adaptative avec modèle de référence. Dans ce type de commande, le comportement souhaité du système est spécifié par un modèle de référence, et les paramètres du contrôleur sont ajustés en fonction de la différence entre la réponse du système et celle du modèle. L'application des réseaux de neurones dans ce type de commande consiste à les introduire en tant que contrôleur et a utilisé l'erreur entre le système et le modèle comme fonction coût pour leur apprentissage.

Sommaire

Chapitre 02 : Commande adaptative et commande neuronale.

Chapitre 03 : Modélisation et commande avec les réseaux RBF

Listes des figures

Chapitre 01 : Introduction sur les réseaux neurones artificiels

Chapitre 02 : Commande adaptative et commande neuronale

Chapitre 03 : Modélisation et commande avec les réseaux RBF

Introduction générale

Le contrôle intelligent est un domaine complexe, stimulant et ayant une grande importance pratique et un potentiel important. Ce domaine, en plein développement, est apparu comme un domaine interdisciplinaire, entre les systèmes contrôlés par ordinateur et l'intelligence artificielle, à la fin des années soixante-dix et au début des années quatre-vingt lorsque l'infrastructure technique et théorique nécessaire en informatique et en temps réel devenait disponible.

La commande de processus au moyen des réseaux de neurones constitue un élément important dans ce domaine. Les réseaux de neurones sont de plus en plus populaires en raison de leur capacité à imiter certains processus créatifs du cerveau, quoique de façon simpliste, qui ne peuvent être imités par des méthodes mathématiques ou logiques existantes. De telles capacités sont essentielles pour résoudre de nombreux problèmes complexes, notamment celle exigées par le développement multidisciplinaire moderne.

Dans ce travail on s'intéresse à l'intégration des réseaux de neurones, en premier lieu, pour la modélisation, puis, dans les systèmes de contrôle adaptatif. Plus précisément, nous étudions la possibilité d'utilisation de ces réseaux dans un système de commande adaptative avec modèle de référence. En effet, dans ce type de commande, le comportement souhaité du système est spécifié par un modèle de référence, et le contrôleur est adapté de façon à minimiser l'erreur entre la réponse du système et celle du modèle. L'application des réseaux de neurones dans ce type de commande consiste à les introduire en tant que contrôleur et a utilisé l'erreur entre le système et le modèle comme fonction coût pour leur apprentissage.

L'ensemble des travaux de ce mémoire est regroupé dans 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 modèles de base de réseaux de neurones.

Le deuxième chapitre constitue une introduction à la commande adaptative et la commande neuronale. Nous évoquons les motivations d'intégration des réseaux de neurones dans le domaine du contrôle ainsi que quelques approches.

Le troisième chapitre analyse l'application des réseaux de neurones à fonction radiale dans la modélisation et la commande. Nous discutons les performances de ce type de réseaux de neurone en analysant les résultats obtenus par leurs applications sur des modèles théoriques

Finalement, une conclusion générale conclue ce mémoire.

CHAPITRE 1:

Introduction sur les réseaux neurones

Chapitre 01 :

Introduction sur les réseaux neurones

I.1 Introduction :

Ces dernières années ont connu une évolution technologique considérable nécessitant un besoin de plus en plus important pour le contrôle et la gestion des systèmes complexes qui introduisent d'énormes calculs et un nombre de variables important. Pour cela, les chercheurs sont à la recherche de nouvelles méthodes plus souple et moins couteuse. Ils s'intéressent alors de plus en plus aux systèmes intelligents qui peuvent apprendre, comme les réseaux de neurones artificiels.

Les réseaux de neurones artificiels se sont des neurones qui s'inspirent à des neurones biologiques, Ils sont des outils puissants qui peuvent utiliser dans plusieurs domaines comme la robotique, la reconnaissance de formes, le traitement de signal et la commande de processus.

Malgré la réussite des réseaux artificiels la communauté scientifique reste toujours loin de la mise en œuvre de machines qui peuvent de reproduire les capacités de calcul des systèmes nerveux, même les plus simples.

I.2. Historique sur les réseaux de neurone :

Le développement des réseaux de neurone a commencé en 1890 avec le psychologue américain W. James, qui a introduit le concept de mémoire associative, Il propose une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones. Cette règle sera connue sous le nom : la règle de Hebb.

En 1943, Mc Culloch et Pitts deux biophysiciens de l'université de Chicago. Ce sont les premiers qui démontrent que le réseau de neurone formel peut représenter n'importe quelle fonction logique et arithmétique.

Quelques années après, En 1949 neuropsychologue et psychologue canadien Donald Hebb explique le conditionnement chez l'animal par les propriétés des neurones eux-mêmes. La loi de modification des propriétés des connexions entre neurones qu'il propose démontre en partie ses résultats expérimentaux. Il introduit le terme connexionnisme pour parler de modèles massivement parallèles, connectés et proposé de nombreuses règles de mise à jour des poids dont la célèbre règle de Hebb.

En 1957, le psychologue américain Franck Rosenblatt propose le modèle de perceptron. Il fabrique le premier neuro-ordinateur basé sur ce modèle et l'applique au domaine de la reconnaissance de formes.

En 1960 le professeur américain Bernard Windrow de l'ingénierie électrique à l'Université de Stanford, Il a été un Co-inventeur de l'algorithme adaptatif du filtre des moindres carrés (LMS) de Widrow-Hoff avec son étudiant de doctorat à l'époque.

En 1969 Marvin Lee Minsky en collaboration avec le mathématicien Seymour Papert publie un ouvrage qui met en exergue les limitations théoriques du perceptron.

Du 1967 jusqu'à 1982 T. Kohnen, S. Grossberg et autres poursuivent les recherches dans les divers domaines comme : la reconnaissance de formes, le traitement adaptatif, la modélisation en neurobiologie, etc.

En 1982, le physicien américain John Joseph Hopfield conçoit un réseau des neurones récurrents, ce types de neurones utilisé en reconnaissance automatique de la parole ou de l'écriture manuscrite.

En 1983 la machine de Boltzmann est le premier modèle type de réseaux de neurone artificielle qui apte à traiter les limitations recensées dans le cas de perceptron.

I.3. Domaine d'application des réseaux de neurone :

Les réseaux de neurones aujourd'hui ont de nombreuses applications dans des domaines très variés [1][2][3]:

- 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.
- 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, ...
- Traitement automatique des langues : segmentation en mots, représentation sémantique des mots (plongements lexicaux), traduction automatique.
- Contrôle : diagnostic de pannes, commande de processus, contrôle qualité, robotique,

I.4. Les neurones biologiques :

I.4.1. Système nerveux :

Le système nerveux humain est responsable de l'envoi et la réception, et du traitement de flux de donnée, tous les muscles et les organes dépendent de ces flux pour fonctionner. Le cerveau humain est le meilleur modèle de la machine, Il est constitué d'un grand nombre d'unités biologiques élémentaire (1000 à 10000 synapse par neurone). [2]

I.4.2. Le neurone biologique :

Les biologistes estiment que le nombre de neurone plus de 100 milliards dans le système nerveux humain, ils sont interconnectés entre eux. Les neurones ne sont pas tous identiques, ils très important de savoir que les neurones n'ont pas un comportement similaire en fonction de leur position dans le cerveau.

Le neurone biologique se décompose en 4 parties lesquelles :

I.4.2.1 Les dendrites :

Ce sont les récepteurs principaux du neurone qui permettent de capter les signaux qui parviennent à l'extérieur et les transitent vers le corps cellulaire.

I.4.2.2 Corps cellulaire ou cellule somatique :

Est la partie centrale de neurone, sa forme dépend de sa position dans le cerveau, elle est peut-être une pyramide ou sphérique, le corps cellulaire inclut le noyau du neurone et effectue les transformations biochimiques nécessaires à la vie du neurone.

I.4.2.3L'axone :

L'axone est la fibre nerveuse et la sortie du neurone qui permet de transmettre les signaux électriques entre les neurones. L'axone possède plusieurs terminaisons nerveuses pour communiquer avec plusieurs neurones au même temps.

I.4.2.4Les synapses:

Ce sont des jonctions entre 2 neurones, ces jonctions sont essentielles dans le fonctionnement du système nerveux.

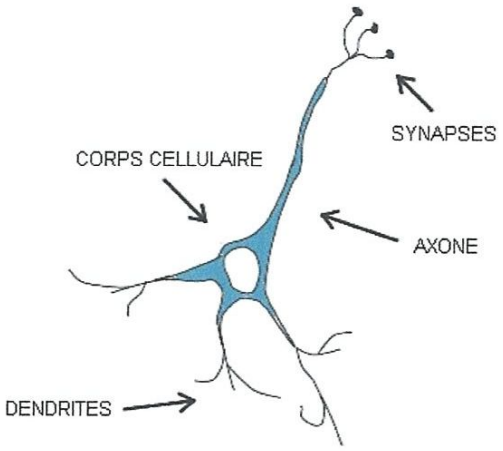


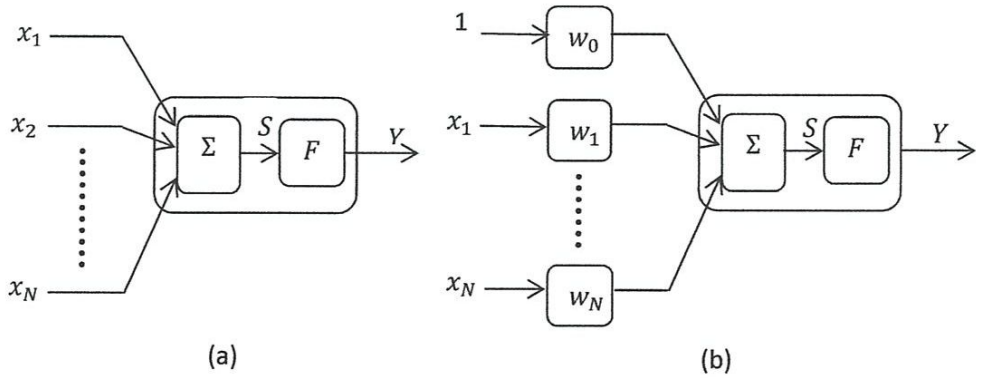
Figure I.1 : Schéma d'un neurone biologique

I.5. Le neurone formel:

Le neurone formel est une modélisation mathématique d'un réseau neurone biologique. Le premier modèle de neurone formel a été conçu aux années 40 par McCulloch et Pitts, Ce modèle calcule la somme pondérée des signaux, puis il déclenche une réponse si cette somme dépasse un certain seuil. Le modèle McCulloch et Pitts ne possède pas une règle apprentissage jusqu'à 1949 le neuropsychologue et psychologue D. Hebb a proposé le principe d'apprentissage. La valeur de sortie (Y) résulte la somme des entrées (x) pondérées par des coefficients w_j et du calcul d'une fonction d'activation (F) de cette somme pondérée. [2][4]

la formalisation mathématique de son comportement est donnée par :

$$Y = F(s) = F\left(w_0 + \sum_{n=1}^{N} w_n\, x_n\right) \qquad \text{(I.1)}$$



Le neurone formel
(a) Modèle de McCulloch et Pitts
(b) Modèle avec capacité d'apprentissage

Figure I.2 : Schéma d'un neurone formel

I.6.Fonction d'activation:

La fonction d'activation joue un rôle très essentiel dans le comportement du neurone artificiel. Cette fonction sous la forme d'équation mathématique souvent non linéaire est appliquée à la sortie du neurone artificiel. La fonction d'activation prend plusieurs formes selon le réseau [3][4]:

I.6.1.Fonction binaire à seuil:

La fonction à seuil, illustrée sur la figure I.3, est définie par :

$$h(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \qquad \text{(I.2)}$$

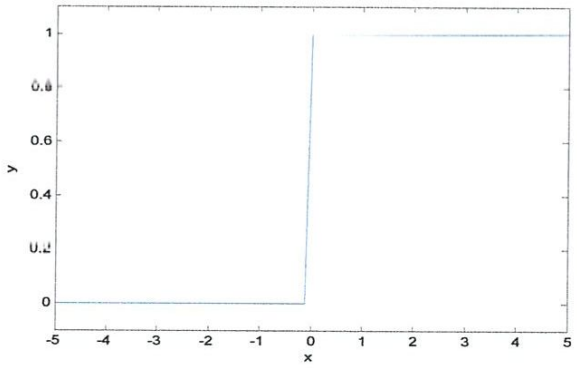


Figure I.3: Fonction binaire à seuil.

I.6.2.Fonction signe :

La fonction signe, illustrée sur la figure I.4, est définie par par :

$$sgn(x) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases} \qquad \text{(I.3)}$$

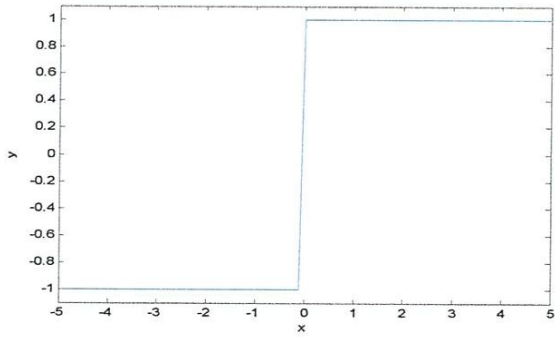


Figure I.4: Fonction signe

I.6.3.Fonction linéaire :

C'est l'une des fonctions d'activations les plus simples, représentée par la figure I.5.Est donnée par :

$$F(x) = x \tag{I.4}$$

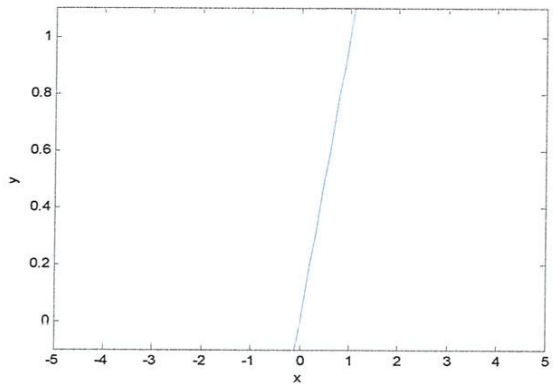


Figure I.5: Fonction linéaire.

I.6.4.Fonction linéaire à seuil ou multi-seuils :

Cette fonction est définie comme suit :

$$h(x) = \begin{cases} x\ si & x \in [u, v] \\ v\ si & x < v \\ u\ si & x > v \end{cases} \tag{I.5}$$

Cette fonction représente un compromis entre la fonction linéaire et la fonction seuil. Son graphe est représenté par la figure I.6.

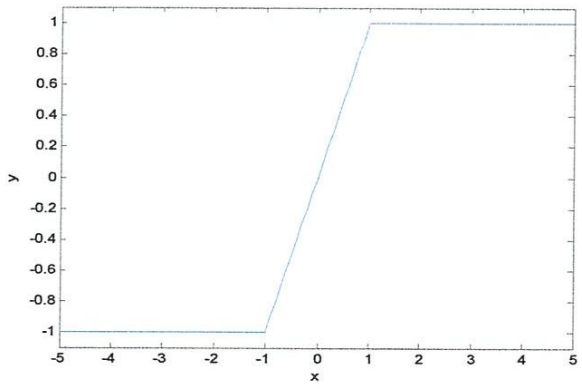


Figure I.6 : Fonction linéaire à seuil.

I.6.5.Fonction log sigmoïdale :

La fonction sigmoïde est l'une des fonctions d'activation les plus utilisées, elle est définie par :

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

Et sa dérivée est :

$$\frac{d}{dx}\big(\text{logsig}(x)\big) = \text{logsig}(x)\big(1 - \text{logsig}(x)\big) \qquad \text{(I.6.b)}$$

Cette fonction est montrée sur la figure I.7 :

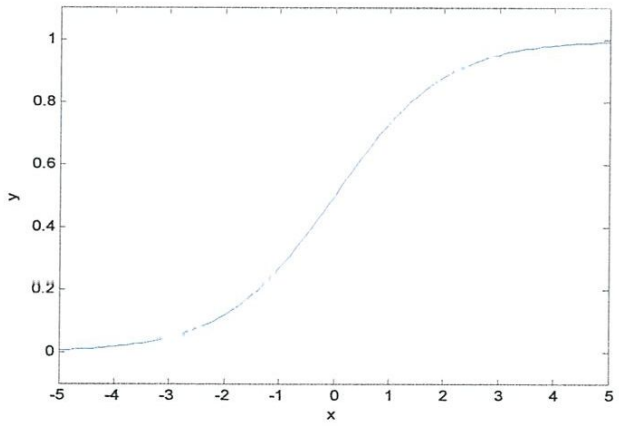


Figure I.7 : Fonction log sigmoïde

I.6.6 Fonction tangente sigmoïde :

La fonction tangente sigmoïde, illustrée dans la figure I.8, est définie par :

$$tansig(x) = \frac{2}{(1+e^{-2})} - 1 \qquad \text{(I.7)}$$

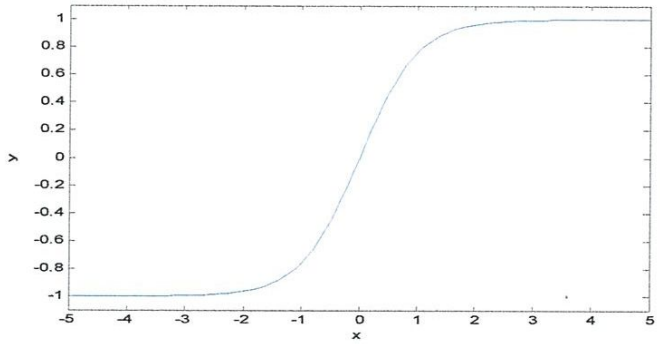


Figure I.8 : Fonction tangente sigmoïde.

I.6.7 Fonction à base radial (RBF) :

Les fonctions RBF ou les fonctions à base radiale ont été très utilisés depuis la fin des années quatre-vingt. La fonction RBF croit où décroît de façon monotone par rapport à la distance d'un point central. Sa forme générale est donnée par :

$$h = \exp\left(-\frac{\|x-c\|^2}{2r^2}\right) \qquad (I.8)$$

Les paramètres de cette fonction sont : le centre c le rayon r. Un exemple de la fonction RBF est représentée à la figure I.9 :

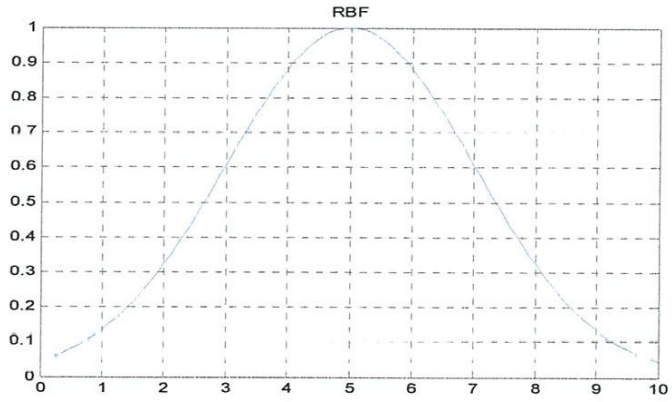


Figure I.9 : Fonction RBF.

I.7. Architecture des réseaux de neurones :

On distingue deux types d'architectures : réseaux de neurones non bouclés et réseaux de neurones bouclés (récurrents) :

I.7.1.Réseaux de neurones non bouclés :

Un réseau non bouclé est un réseau qui réalise une ou plusieurs fonctions algébriques de ses entrées, par composition des fonctions réalisées par chacun de ses entrées. ce type de réseaux est comprend souvent 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. Dans ce type de réseau l'information circule dans un seul sens ; de l'entrée vers la sortie (sens unidirectionnel)

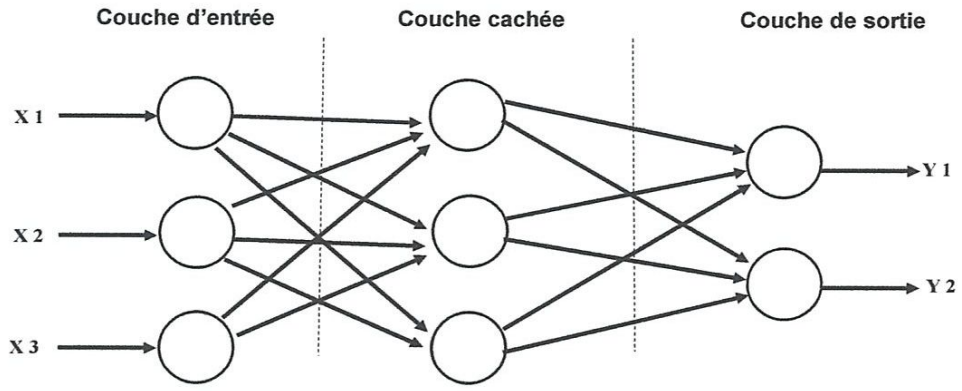


Figure I.10 : Réseaux de neurones non bouclés .

I.7.2.Réseaux de neurones bouclés :

Les réseaux neurones bouclés ou bien les réseaux à "Feed Back " possèdent une, ou plusieurs topologies de connexions, ce type des réseaux comprennent une, ou plusieurs valeurs de sortie qui ramènent aux entrées.

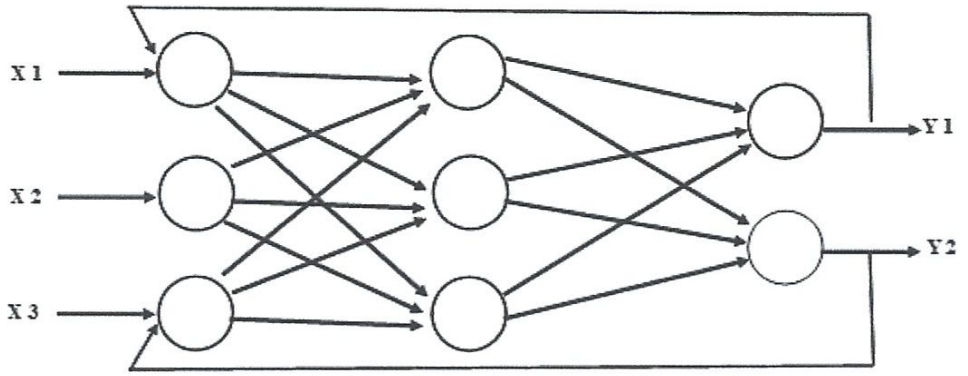


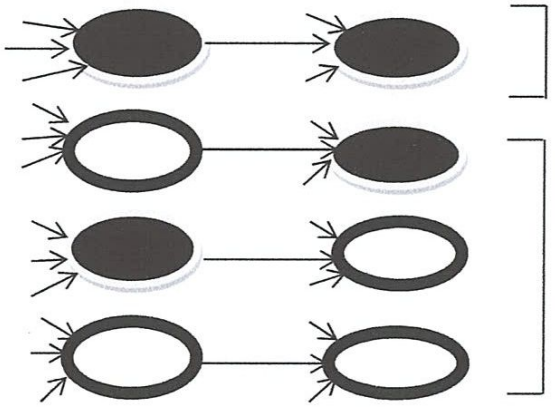
Figure I.11 : Réseaux de neurones bouclés

I.8. Apprentissage des réseaux de neurones :

L'apprentissage est une caractéristique essentielle des réseaux de neurone, cette caractéristique consiste à modifier les poids synaptiques jusqu'à le réseau puisse effectuer la tache désirée [1][3][4].

Chaque poids W_{ij} reliant un neurone i au neurone j à l'itération (r) est modifié selon l'équation suivante :

La règle de Hebb donne alors :



Connexion renforcée

Connexions non renforcées

I.8.2.2.La règle de Widrow-Hoff (règle de delta):

C'est une règle d'apprentissage supervisé utilisé dans le but de correction des erreurs observée en sortie. La règle de Widrow Hoff, ou la règle de delta, est basée sur la minimisation de l'erreur quadratique moyenne (Fonction de coût).

Pour un ensemble d'apprentissage contenant Q paires entrée/sortie désirée $\{(X(q)\ /T(q))\}$ avec $q = 1,,,Q$ ou $X^{(q)}$ et $T^{(q)}$ représentent respectivement la $q^{ème}$ entrées et la $q^{ème}$ sorties désirée, l'erreur $(e(r))$ à l'itération r est donnée par

$$e(r) = T(r) - Y(r) \tag{I.11}$$

Où : $Y(r)$ est la sortie calculée du réseau

La fonction de coût est :

$$F(X) = e^2(r) \tag{I.12}$$

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

$$\Delta w_{ij} = -\eta \frac{\nabla F(X)}{\partial w_{ij}} = \eta \frac{\partial e^2(r)}{\partial w_{ij}} \tag{I.13.a}$$

Cette règle de correction permet donc aux neurones d'adapter leurs poids pour se rapprocher a Une valeurs 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 :

$$\Delta w_{ij}(r) = \Delta w_{ij}(r-1) - \eta(t_i - y_i)x \tag{I.13.b}$$

Où :t_i et y_i sont respectivement la sortie désirée et la sortie calculée correspondantes au neurone x est l'entrée et η est une constante positive appelée pas d'apprentissages.

$$W_{ij}(r) = W_{ij}(r-1) + \Delta W_{ij}(r-1) \quad \text{(I.9)}$$

I.8.1.Protocole d'apprentissage :

La plupart des réseaux de neurones possèdent le même protocole d'apprentissage, il comporte quatre étapes [4]:

Étape 1 **:** initialisation des poids synaptiques de neurone avec des faible valeurs.

Étape 2 : présentation de l'exemple d'entrée et propagation de l'activation des neurones.

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

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

I.8.2.Règles d'apprentissages :

I.8.2.1 La règle de Hebb:

Cette règle a été établie par le neuropsychologue et psychologue canadien Donald Hebb en 1949,la règle de Hebb est le premier mécanisme d'évolution basée sur synapses, Cette règle est souvent résumée par : « le poids synaptiques entre deux neurones est renforcé quand les deux neurones sont actifs au même temps » .

Le principe de la règle de Hebb est donnée par la relation suivante

$$w_{ij}(t+\delta t) = W_{ij}(t) + \beta \times A_i \times A_j \quad \text{(I.10)}$$

A_i, A_j : activation de neurone i et l'activation du neurones j.

β : le paramètre de l'intensité de l'apprentissage ($\beta > 0$).

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

Cellule activée

Cellule non activée

I.8.3 Les Types d'apprentissage :

I.8.3.1.Apprentissage supervisé :

Pour ce type d'apprentissage, on doit présenter au réseau des entrées, et à la fois les sorties désirées pour ces entrées. Le réseau doit ajuster ses poids de façon à minimisé l'écart entre la réponse désirée et la sortie du réseau. Cette procédure est répétée jusqu'à ce qu'un critère de performance soit satisfait. L'apprentissage supervisé nécessite donc la définition d'une base d'exemples d'apprentissage représentative. Chaque exemple présente au réseau est un couple (sortie désirée, entrée).

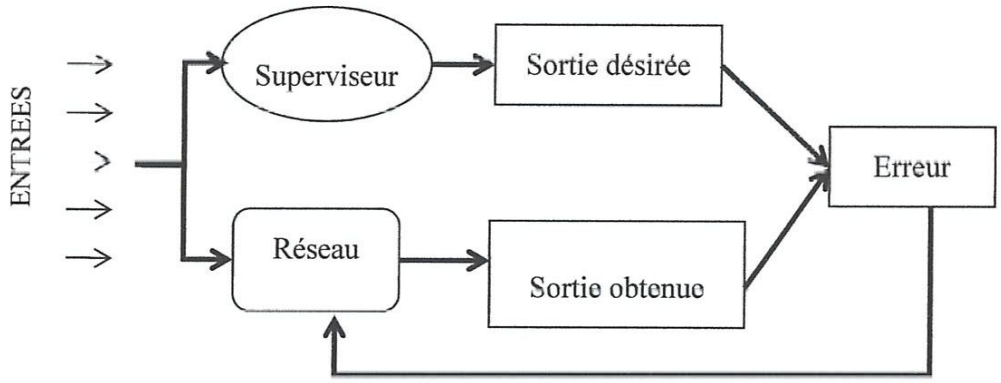


Figure I.12 : Apprentissage supervisé

I.8.3.2. Apprentissage non supervisé :

Pour ce type d'apprentissage, aucune information sur la réponse désirée n'est fournie au réseau. On présente une entrée au réseau et on le laisse évoluer librement jusqu'à un comportement optimal.

L'ajustement et la modification des poids tes indépendant de la relation entre la taches qui doit effectuer et le comportement du réseau.

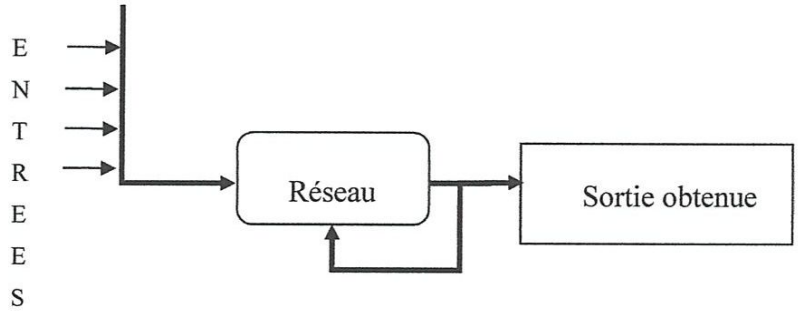


Figure I.13 : Apprentissage non supervisé

I.91. Conclusion :

Les réseaux de neurones ont connu un développement très important en matière de nouvelles architectures et de nouveaux algorithmes d'apprentissages. Dans ce chapitre nous avons donné quelques notions principales sur les réseaux de neurones.

Les réseaux de neurones possèdent certains avantages et inconvénients [2]:

Les avantages :

- La possibilité de la représentation de n'importe quelle fonction soit linéaire ou pas.
- Résistance au bruit ou au manque de fiabilité des données.
- capacité de calcul parallèle.

Les inconvénients :

- Boites noires.
- Manques de techniques d'initialisation.
- 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).

Chapitre 02 :

Commande adaptative et commande neuronale

II.1 Introduction :

Le contrôle adaptatif devient de plus en plus populaire dans de nombreux domaines de l'ingénierie et de la science, car les concepts de systèmes adaptatifs deviennent plus attrayants dans le développement d'applications avancées. Le contrôle adaptatif fait face à de nombreux défis importants, en particulier dans les applications non traditionnelles, telles que les systèmes en temps réel qui ne possèdent pas de modèles classiques précis. En effet, dans les systèmes modernes, il existe de nombreuses 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. Plusieurs méthode moderne, telles 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 problèmes 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 Commande adaptative :

II.2.1 Principe :

Dans le langage courant, «s'adapter» signifie changer un comportement pour se conformer à des nouvelles circonstances. Intuitivement, un contrôleur adaptatif est donc un contrôleur qui peut modifier son comportement en réponse à des changements dans la dynamique du processus et le caractère des perturbations [5]. Dans les systèmes de contrôle ordinaire à rétroaction, les mesures de réponse sont renvoyées vers le contrôleur par l'intermédiaire de la chaine de retour, mais les valeurs des paramètres du contrôleur en elles-mêmes sont inchangées pendant le fonctionnement. En contrôle adaptatif, les valeurs de paramètres sont modifiées en fonction de certains critères.

Un système de contrôle adaptatif est un système de contrôle à rétroaction dans lequel les valeurs de certains ou de tous les paramètres du contrôleur sont modifiées (adaptées) pendant le fonctionnement du système, en temps réel, sur la base de certaines mesures de performance. Un contrôleur adaptatif est donc un contrôleur qui peut modifier son comportement en réponse à des changements dans la dynamique du processus et le caractère des perturbations.

La figure (II.1) représente le schéma général d'un système de contrôle adaptatif [5]. Il peut être considéré comme ayant deux boucles. La première est une réaction normale avec le processus et le contrôleur. L'autre boucle est la boucle d'ajustement des paramètres.

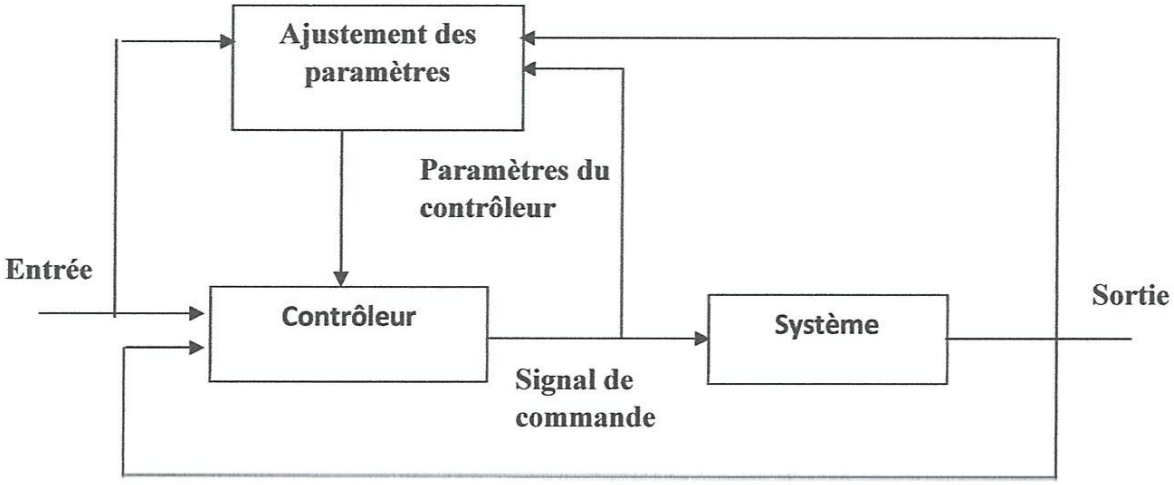


Figure II.1 : Schéma général de commande adaptative.

Il y a deux différentes grandes classes de systèmes de contrôle adaptatif : Le contrôle adaptatif direct et le contrôle adaptatif indirect [5][]. Dans les systèmes de contrôle adaptatifs directs, les paramètres du contrôleur sont directement mis à jour à partir d'une loi adaptative, tandis que dans les conceptions de contrôle adaptatif indirect (figure II.2), les paramètres du système sont d'abord estimés de façon adaptative puis les paramètres du contrôleur sont calculés à partir des paramètres estimés du système.

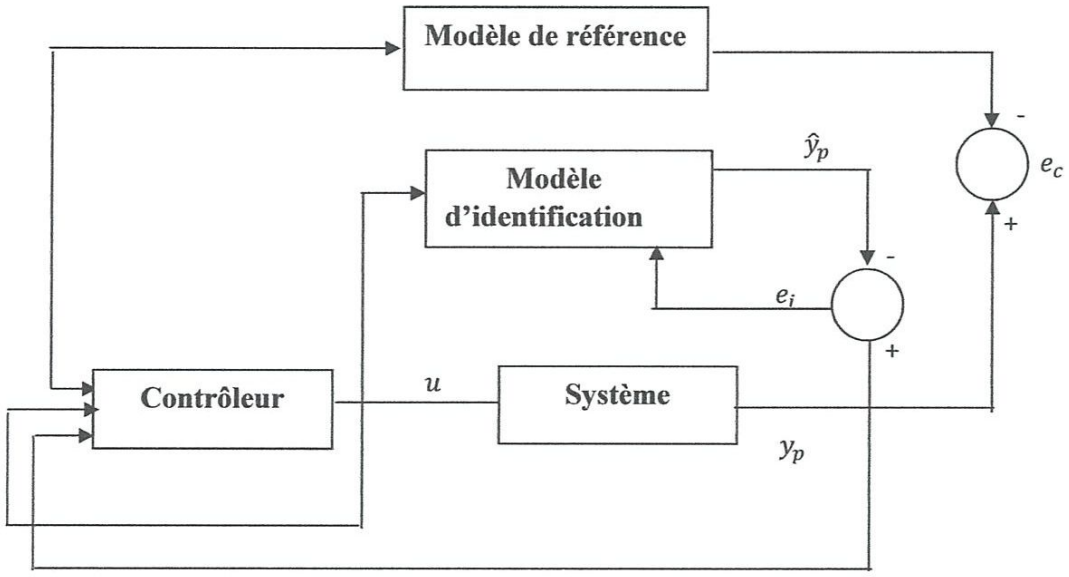


Figure II.2: Schéma général de la commande adaptative indirecte

II.2.2 Commande adaptative avec modèle de référence :

Dans la commande adaptative avec modèle de référence (MRAC, Model Reference Adaptive Control), le comportement souhaité du système est spécifié par un modèle, et les paramètres du contrôleur sont ajustés en fonction de la différence entre la réponse du système et celle du modèle. Les systèmes adaptatifs avec modèle de référence ont été dérivés à l'origine pour des systèmes déterministes à temps continu. Des extensions à des systèmes à temps discret et des systèmes avec des perturbations stochastiques.

Dans les systèmes de contrôle ordinaire à rétroaction, les mesures de réponse sont renvoyées vers le contrôleur par l'intermédiaire de la chaine de retour, mais les valeurs des paramètres du contrôleur en elles-mêmes sont inchangées pendant le fonctionnement. En contrôle adaptatif, les valeurs de paramètres sont modifiées en fonction de certains critères. Dans les systèmes MRAC, en particulier, la même entrée de référence qui est appliquée au système physique est également appliquée à un modèle de référence. La différence entre la réponse du système physique et la sortie du modèle de référence est l'erreur.

L'objectif idéal est de rendre cette erreur nulle à tout moment. Par conséquent, le système fonctionnera comme le modèle de référence. Le signal d'erreur est utilisé par le mécanisme d'adaptation pour déterminer les modifications nécessaires aux valeurs des paramètres du contrôleur afin d'atteindre cet objectif.

L'approche générale de MRAC est illustrée par le diagramme de la figure II.3. Le système a une boucle de réaction ordinaire composée du processus et du contrôleur et une autre boucle de rétroaction qui change les paramètres du contrôleur. Les paramètres sont modifiés sur la base du retour d'erreur, qui est la différence entre la sortie du système et la sortie du modèle de référence. La boucle de rétroaction ordinaire est appelée boucle interne, et la boucle de réglage des paramètres est appelée boucle externe. Le mécanisme d'ajustement des paramètres dans un système adaptatif avec modèle de référence peut être obtenu de deux manières : en utilisant une méthode de gradient ou en appliquant la théorie de la stabilité.

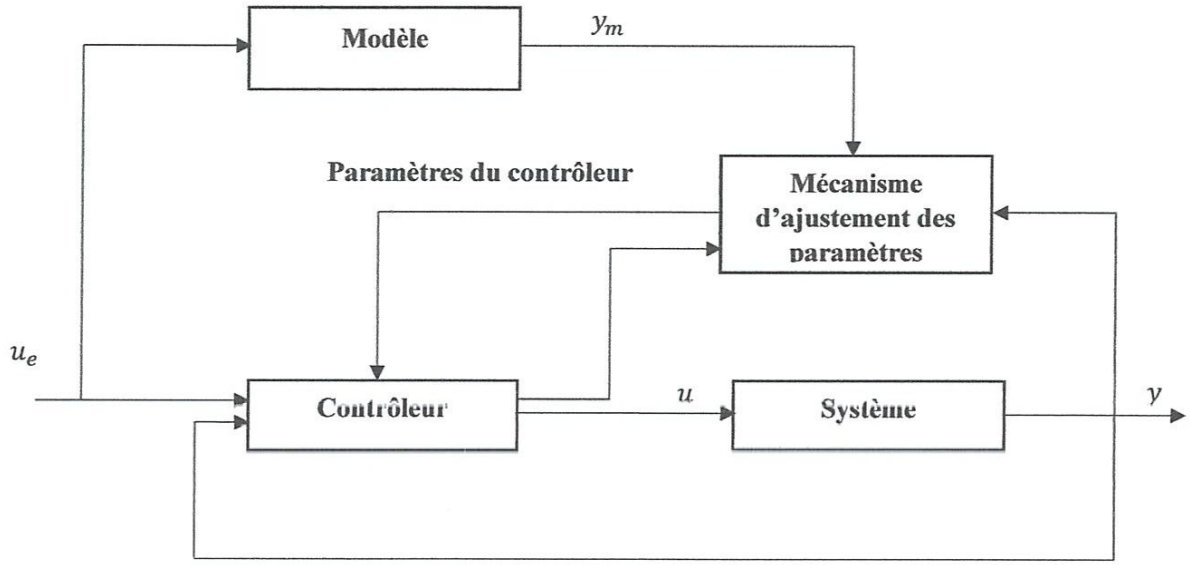


Figure II.3 : Commande adaptative avec modèle de référence .

Il est a noté que le modèle de référence est un modèle idéal qui génère une réponse souhaitée lorsqu'il est soumis à l'entrée de référence, au moins de manière asymptotique, c'est-à-dire l'erreur converge vers zéro. En ce sens, il s'agit simplement d'un moyen de spécification des performances et il se peut qu'il ne présente aucune ressemblance ou analogie avec le modèle analytique du processus lui-même. Par exemple, le modèle de référence peut être choisi en tant que système linéaire avec des propriétés d'amortissement et de bande passante souhaitées [5][6].

II.2.3 La règle MIT :

La règle du MIT est l'approche originale du MRAC. L'origine du nom est du fait qu'elle a été développé au laboratoire d'instrumentation (maintenant le laboratoire Draper) au MIT (Massachusetts Institute of Technology).

Pour présenter la règle MIT, considérons un système en boucle fermée dans lequel le contrôleur a un paramètre ajustable θ. La réponse en boucle fermée souhaitée est spécifiée par un modèle dont la sortie est y_m. Soit e_c l'erreur entre la sortie du système en boucle fermée et la sortie y_m du modèle. Une possibilité consiste à ajuster les paramètres de telle sorte que la fonction objective est minimisée. cette fonction est donnée par:

$$J(\theta) = \frac{1}{2} e_c^2 \qquad \text{(II.1)}$$

Pour rendre J petite, il est raisonnable de changer les paramètres dans la direction négative du gradient de J. le règle du MIT est alors donnée par :

$$\frac{\partial \theta}{\partial t} = -\eta \frac{\partial J}{\partial \theta} = -\eta e_c \frac{\partial e_c}{\partial \theta} \quad \text{(II.2)}$$

Dans cette règle, La dérivée partielle $\partial e_c / \partial \theta$, indique comment l'erreur est influencée par le paramètre réglable.

Il existe de nombreuses alternatives à la fonction objective donnée par l'Eq. (II.1). Par exemple, la fonction valeur absolue de l'erreur donnée par :

$$J(\theta) = |e| \quad \text{(II.3)}$$

La méthode du gradient donne :

$$\frac{\partial \theta}{\partial t} = -\eta \left(\frac{\partial e}{\partial \theta}\right) sign(e) \quad \text{(II.4)}$$

Il est a noté qu'il aussi possible d'ajuster de nombreux paramètres. L'équation (II.2) s'applique également dans ce cas. Le symbole θ devrait alors être interprété comme un vecteur et $\partial e / \partial \theta$ comme le gradient de l'erreur par rapport aux paramètres [5][6].

II. 3 Motivations de la commande neuronale :

Le control neuronal est défini comme l'utilisation de réseaux neuronaux artificiels pour émettre des signaux de contrôle réels. Habituellement, lorsque nous parlons de «contrôle», il existe une forme d'intelligence associée au terme.

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 [7] :

II.3.1. 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.3.1. 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.3.3. 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, les ces contrôleurs semblent pouvoir contrôler dans une plus large gamme d'incertitude.

II.3.4. Calcul parallèle :

Le parallélisme massif des réseaux de neurones offre de très grande 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.

II. 4. Premiers travaux sur la commande neuronale :

L'un des premiers travaux les importants sur le contrôle neuronal est celui de Widrow et Smith [8]. 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 [9] , ont proposé une nouvelle approche de contrôle d'un manipulateur (bras robotique) en utilisant une forme de réseau de neurones à base de table de consultation connue sous le nom de contrôleur d'articulation du modèle cérébelleux (CMAC, Cerebellar Model Articulation Controller). Miller et al, 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. 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 Dans la Figure II.4. Werbos a qualifié cette approche de «critique adaptatif» et plusieurs variantes de cette approche d'affectation de crédit ont été étudiées et appliquées de manière indépendante par d'autres (Gaines et Andreae, Witten, Werbos, Hollande, Anderson)

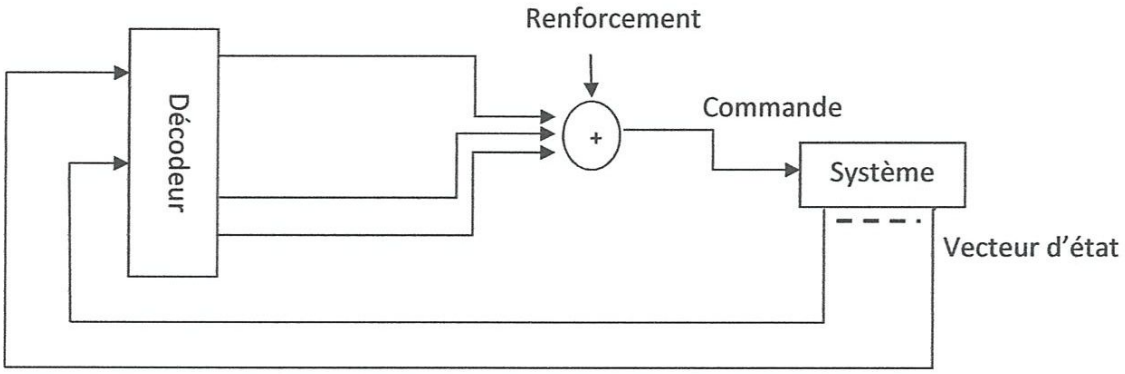


Figure II.4 : Méthode critique adaptative.

Le succès de l'algorithme de back-propagation pour entrainer des réseaux multicouches a créé 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 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 se système.

L'idée de ce schéma 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 sortie désirée du système.

Deux approches de commande par modèle inverse utilisant l'algorithme de la retro-propagation, appelé architecture d'apprentissage généralisée et spécialisée, ont été proposée [6].

Dans l'architecture d'apprentissage généralisée, comme le montre la figure II.5, 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 la rétroaction.

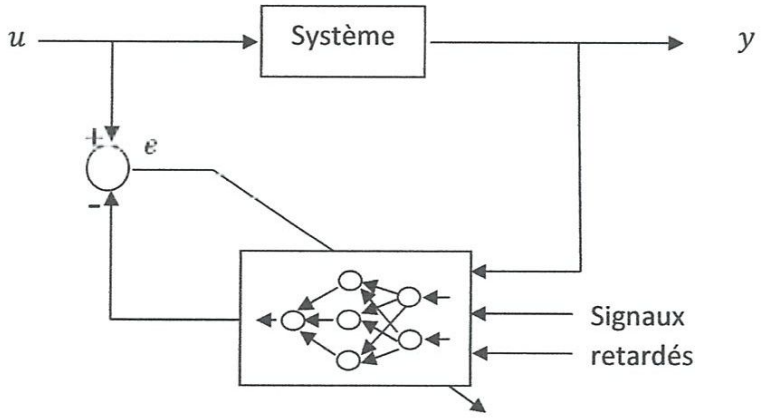


Figure II.5 : Architecture d'apprentissage généralisé

Dans la deuxième approche, l'architecture d'apprentissage spécialisée (Figure II. 6), le réseau est entrainé de manière en ligne (ou «dirigé par objectif») où l'erreur de performance est inversée par le réseau à chaque échantillon.

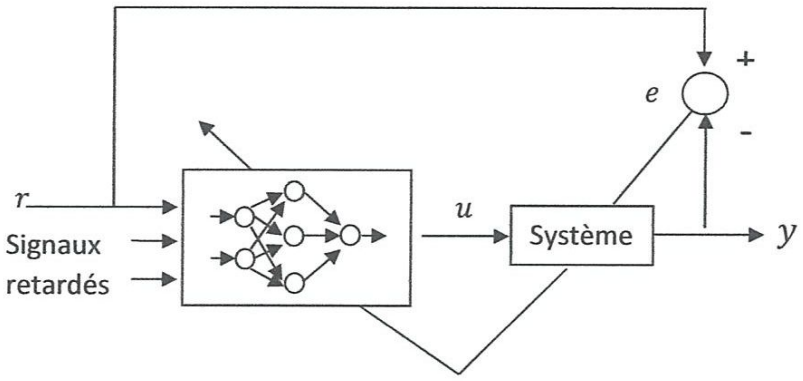


Figure II.6 : Architecture d'apprentissage spécialisée.

II. 5. Commande neurale avec modèle inverse :

II.5.1. Approche série :

Le contrôle neuronal direct avec modèle inverse (figure II.7), également appelé contrôle neuronal en série, est un système dans lequel le réseau de neurones permet réaliser la dynamique inverse du système [7].

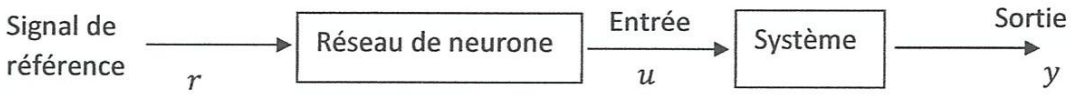


Figure II.7 : Diagramme bloque de la commande neuronale directe avec modèle inverse.

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

$$y = G_s(u) \tag{II.5}$$

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

$$u = G_s^{-1}(y) \tag{II.6}$$

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 :

$$y = G_s(u) = G_s\left(G_s^{-1}(r)\right) = r \tag{II.7}$$

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.8. Ce réseau sera entrainé en 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.

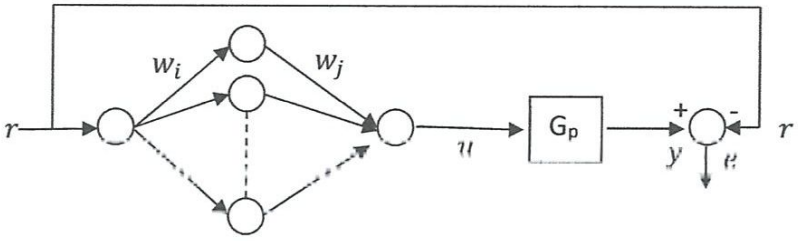


Figure II.8 : Commande basée sur un réseau avec une seule entrée

Ce réseau de neurones à la structure interne la plus simple où un seul 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 :

$$E = \frac{1}{2}(r - y)^2 \qquad \text{(II.8)}$$

Dans les applications réelles, les systèmes a contrôlé sont généralement dynamiques ou 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 entrés additionnelles sont des entrés passée et aussi des sorties passées du réseau.

II.5.2. Approche parallèle :

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. [12] [13]. Dans cette approche, illustrée sur la figure II.9, le réseau de neurone est configuré en parallèle avec un contrôleur conventionnel a 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.

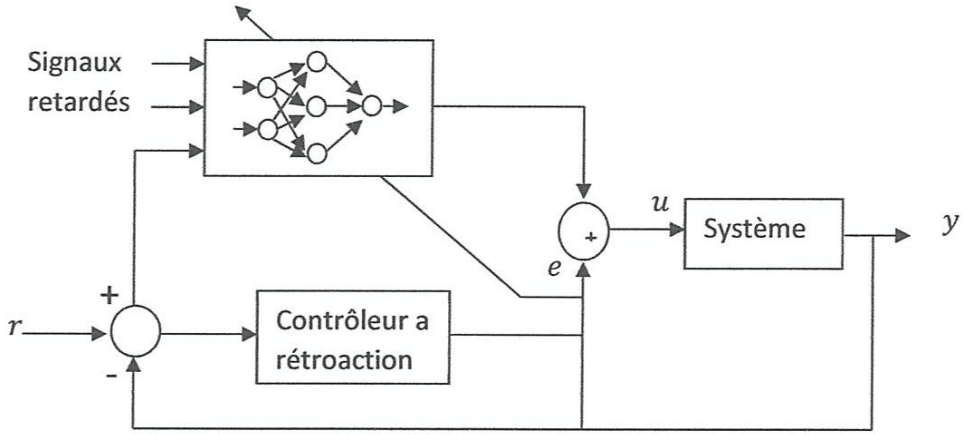


Figure II.9 : Commande parallèle appelée basée sur l'apprentissage avec retour de l'erreur

II. 5.3 Commande neuronale indirect :

Dans cette approche, deux réseaux de neurones sont utilisés pour contrôler le système comme le montre la figure II.10. 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.

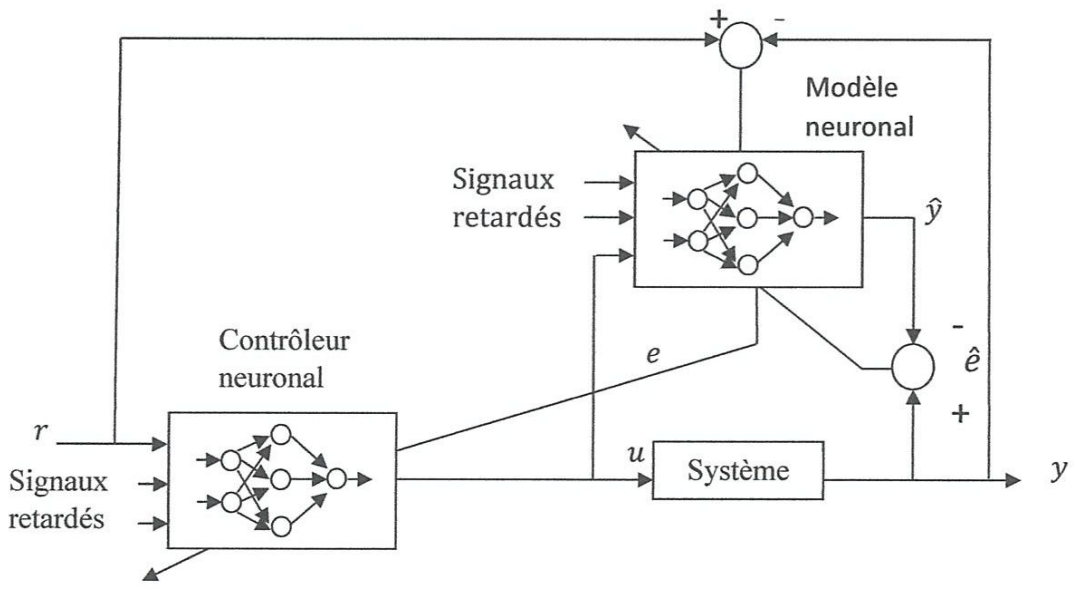


Figure II.10 : Commande neuronale indirect avec modèle et contrôleur

II.6. Commande neuronale auto-ajustable :

II.6.1 Principe :

La configuration du contrôle neuronal auto-ajustable est illustrée sur la figure II.11 où un réseau neuronal est utilisé pour régler les paramètres d'un contrôleur conventionnel similaire au réglage effectué par un opérateur humain.

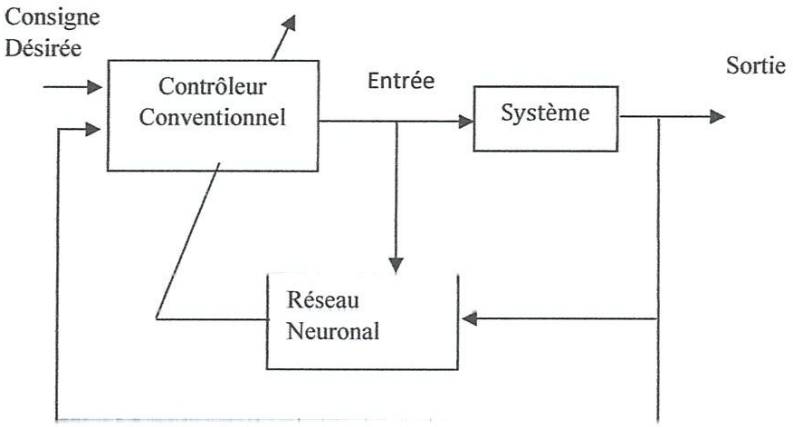


Figure II.11 : Configuration générale de la commande neuronale auto-ajustable.

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.6.2 Ajustement 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.13 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.

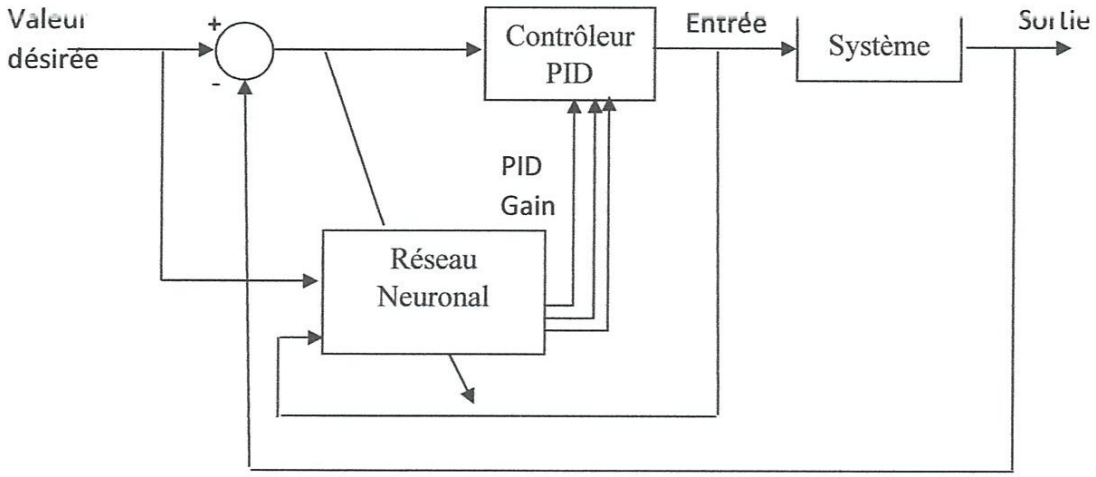


Figure II.13 : Configuration générale d'ajustement des paramètres d'un PID avec un réseau de neurones.

II. 7 Conclusion :

La plupart des systèmes de contrôle neuronal sont généralement basés sur les approches de conception suivantes [7]:

- 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.
- 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.
- Dans l'approche d'autoréglage, un réseau de neurones 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.
- Dans l'approche basée sur un modèle et contrôleur, le réseau de neurones maximise une certaine mesure de performance au fil du temps.
- Dans la commande adaptative, Le réseau de neurones permet d'approximer un contrôle optimal dans un environnement non linéaire et avec perturbations.

Chapitre 03 :

Modélisation et commande avec les réseaux RBF

III.1 Introduction :

Les réseaux de neurone à base radiale (RBF, Radial Basis Functions) sont apparus à la fin des années quatre-vingt. Ce type de réseau a été utilisé dans plusieurs domaines comme la reconnaissance de forme, la reconnaissance de la parole et l'interpolation. Dans ce travail, nous étudions ce type de réseaux de neurone dans le cadre la modélisation et la commande et des systèmes. Avant d'aborder ces thématiques, nous présentons brièvement l'architecture des réseaux de neurones de type RBF, l'effet des paramètres des fonctions gaussiennes (RBF) et les différents algorithmes d'apprentissage.

III.2 Les réseaux de neurone de type RBF :

III.2.1 Architecture des réseaux RBF :

Les réseaux de neurones RBF font partie des réseaux de neurones supervisés. Un réseau RBF est constitué de trois couches : une couche d'entrée qui retransmet les entrées sans modifications, une seule couche cachée qui contient les neurones RBF qui sont généralement des gaussiennes, et une couche de sortie dont les neurones sont généralement animés par une fonction d'activation linéaire. Chaque couche est complètement connectée à la suivante et il n'y a pas de connexions à l'intérieur d'une même couche.

Ce réseau est constitué de N neurones d'entrée, M neurones cachés et J neurones de sortie.

La sortie de m^{ième} neurone de la couche cachée est donnée par :

$$h_m^{(q)} = exp\left(-\frac{\|x^{(q)}-c_m\|^2}{2b_m^2}\right) \qquad \text{(III.1)}$$

c_m et b_m sont, respectivement, le centre et la largeur du m^{ième} neurone de la couche cachée. La sortie du j^{ième} neurone de la couche de sortie est donnée par :

$$y_j^{(q)} = \frac{1}{M}\sum_1^m w_{mj} \times h_m^{(q)} \qquad \text{(III.2)}$$

w_{mj} Sont les poids reliant la couche cachée à celle de la sortie.

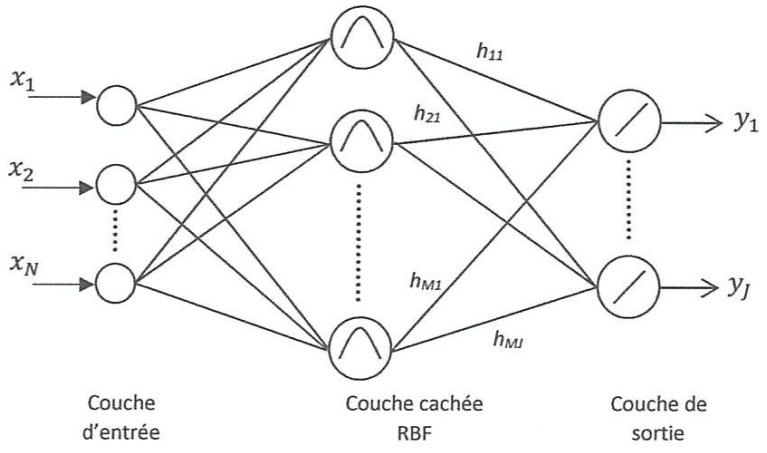


Figure III.1 : Un réseau de neurone RBF.

III.2.2. Effet des paramètres des fonction RBF:

L'effet des fonctions RBF de la couche cachée est lié aux centres c_m, la valeur des largeurs bj et le nombre des neurones cachés. Le principe de la conception c_m et b_m peut-être exprimé comme le suit [4][15]:

1. Le vecteur des centres c_j représente l'organisation des fonctions gaussiennes du réseau de neurones. Plus la valeur d'une entrée est proche de cj, plus la sensibilité est grande, sinon elle petite. Les centres devraient être choisis de façon de bien couvrir l'intervalle des entrées.

2. La valeur de largeur b_m représente la largeur de la fonction gaussienne. Plus la valeur b_m est grande, plus la fonction gaussienne est large. La largeur de la fonction gaussienne représente l'intervalle de couverture des entrées du réseau. Plus la fonction gaussienne est large, plus l'intervalle de couverture du réseau pour les entrées est grand. Autrement, plus l'intervalle de couverture est petit, plus les entrées sont moins représentées. Les valeurs des largeurs b_m devraient aussi être optimales pour avoir une meilleure couverture.

III.2.3. Apprentissage RBF par rétropropagation :

L'apprentissage du RBF est supervisé. Il consiste à adapter les poids du réseau, de manière que le réseau soit capable de réaliser une tâche donnée (classification, modélisation, Commande, Etc...). Pour ce faire on doit avoir un ensemble d'exemples d'apprentissage constitués d'une suite de Q vecteurs de sorties désirées $T^{(q)}\ (t_1, t_2, \dots t_j)$. L'apprentissage consiste à minimiser l'erreur quadratique totale donnée par :

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

L'algorithme d'apprentissage d'un réseau RBF en utilisant la rétro propagation du gradient [14] permet d'ajuster les poids synaptiques, en propageant l'erreur de la couche de sortie vers la couche d'entrée (figure III.2).

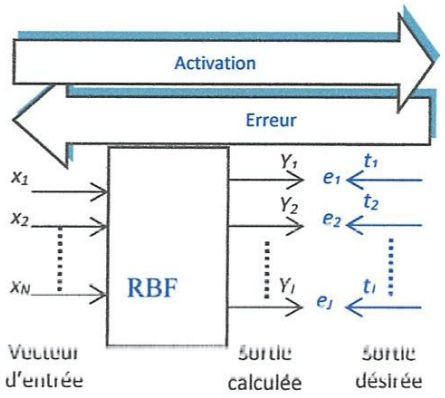


Figure III.2 : Apprentissage d'un réseau RBF par la rétro-propagation du gradient.

III.2.6 Algorithme générale d'apprentissage par rétro-propagation :

L'apprentissage des réseaux RBF est généralement composé de deux étapes distinctes :

- La première est l'initialisation et des paramètres des fonctions RBF : centres et largeurs. Les poids sont initialisés aléatoirement.
- La deuxième concerne l'apprentissage des poids de la couche de sortie des paramètres des fonctions RBF.

Algorithme générale d'apprentissage par rétro-propagation est donné comme suit [2][4]:

Etape 1 : Initialisation des données

N : Nombre d'exemple d'apprentissage
M : Nombre des neurones cachés
Initialisation les paramètres de RBF : C, b
Initialisation aléatoire des poids : w.
η: *Pas d'apprentissage.*
I : Nombre d'itération.
E_m: *Erreur minimale.*

Etape 2 : Apprentissage

Pour chaque itération i

Pour chaque exemple q

Pour chaque neurone m

Calcul des fonctions RBFs (sortie de couche cachée) :

$$h_m^{(q)} = exp\left(-\frac{\| x^{(q)} - c_m \|^2}{2{b_m}^2}\right)$$

Calcul de la sortie : $y = h \times w$

L'ajustement des poids :

$w(i) = w(i-1) + \Delta w$ *Avec* $\Delta w = \partial E/\partial w$

Pour chaque neurone m

L'ajustement des paramètres des RBFs :

$c_m(i) = c_m(i-1) + \Delta c_m(i)$ *Avec* $\Delta c_m(i) = \partial E/\partial c$

$b_m(i) = b_m(i-1) + \Delta b_m(i)$ *Avec* $\Delta b_m(i) = \partial E/\partial b$

III.2.4. Le pas d'apprentissage :

Dans le processus d'apprentissage d'un réseau de neurones par la rétro-propagation, le pas d'apprentissage définit la taille de la descente sur la surface de l'erreur et, par conséquent, la vitesse d'apprentissage. Si ce paramètre est choisi avec une petite valeur, l'adaptation des poids sera avec des petits pas et l'apprentissage sera lent. En revanche s'il est choisi grand, le réseau risquera de dépasser le minimum global et risquera aussi d'avoir des oscillations [2][4].

Théoriquement, le pas d'apprentissage optimal est celui qui mène au minimum d'erreur dans une seule époque d'apprentissage.

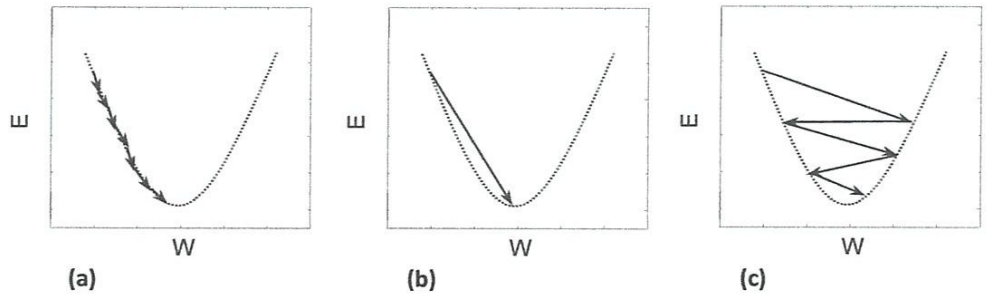


Figure III.3 : Descente de gradient dans un problème unidimensionnel avec différentes valeurs du pas d'apprentissage.

(a) $\eta < \eta_{opt}$ (b) $\eta = \eta_{opt}$ (c) $\eta > \eta_{opt}$

III.2.5 L'ajout d'un terme d'inertie (Le moment) :

Cette technique a été proposée par Rumelhart et al. [14]. Elle consiste à ajouter un terme d'inertie aux corrections apportées à chaque poids. L'adaptation du poids w_{nm} à l'itération $(t+1)$ sera de la forme :

$$w_{nm}(t+1) = w_{nm}(t) + \eta \nabla E\big(w_{nm}(t)\big) + \alpha \Delta\big(w_{nm}(t-1)\big)$$
$$= w_{nm}(t) - \eta \frac{\partial E(t)}{\partial w_{nm}} + \alpha\big(w_{nm}(t) - w_{nm}(t-1)\big) \qquad \text{(III.5)}$$

L'introduction du moment tente de corriger les surfaces d'erreur qui comportent souvent des plateaux (régions dans lesquelles la pente $dj(w)/dw$ est très petite).

III.3. Modélisation des systèmes avec les réseaux RBF

III.3.1 Principe :

Dans cette partie nous étudions la capacité de modélisation des réseaux RBF. on se base sur l'architecture de la figure III.4 Dans cette structure le signal d'entrée $u(k)$ est appliqué en même temps au système et au modèle neuronale. L'erreur $e(k)$ entre la sortie du système et celle du modèle est utilisée pour l'apprentissage du réseau.

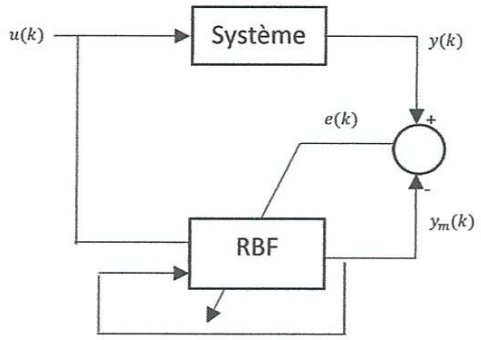


Figure III. 4 : Modélisation avec un réseau de neurone RBF.

Dans le réseau de neurones RBF, $x = [x_1, x_2, \dots x_n]$ est le vecteur d'entrée, et h_j est fonction gaussienne pour le réseau neurone j.

$$h_j = exp\left(-\frac{\| x(k) - c_j||^2|}{2{b_j}^2}\right)$$

Où $c = [c_{j1} \dots c_{jn}]$ est le vecteur de centre du réseau.

Le vecteur de largeur de la fonction gaussienne est

$$b = [b_1 ; \dots ; b_m] \qquad \text{(III.6)}$$

La valeur de poids est :

$$w = [w_1 ; .... ; w_m] \quad \text{(III.7)}$$

La sortie de RBF est

$$y_m = h_1.w_1 + ... h_j.w_j + h_m.w_m \quad \text{(III.8)}$$

La fonction à minimiser est l'erreur entre la sortie du système, y, et la sortie du modèle neuronale, y_m, donnée comme suit:

$$E(t) = \frac{1}{2} . (y(t) - y_m(t))^2 \quad \text{(III.9)}$$

L'adaptation des poids est donnée par [4][15]:

$$\Delta w_j = -\eta \frac{\partial E}{dw_j} = \eta (y(t) - y_m(t)) h_j \quad \text{(III.10)}$$

La mise à jour du poids w_j a l'instant t est donnée par :

$$w_j(t) = w_j(t-1) + \Delta w_j + \alpha (w_j(t-1) - w_j(t-2)) \quad \text{(III.11)}$$

L'adaptation des paramètres des fonctions RBFs sont donnée par :

$$\Delta b_j(t) = -\eta \frac{\partial E}{db_j} = \eta (y(t) - y_m(t)) w_j h_j \frac{\|x - c_j\|^2}{b_j^3} \quad \text{(III.12)}$$

La mise à jour de la largeur b_j a l'instant t est donnée par :

$$b_j(t) = b_j(t-1) + \Delta b_j(t) + \alpha (b_j(t-1) - b_j(t-2)) \quad \text{(III.13)}$$

$$\Delta c_{ji}(t) = -\eta . \frac{\partial E}{dc_{ji}} = \eta (y(t) - y_m(t)) w_j h_j \frac{xj - c_j}{b_j^2} \quad \text{(III.14)}$$

La mise à jour du centre c_{ji} a l'instant t est donnée par :

$$c_{ji}(t) = c_{ji}(t-1) + \Delta c_{ji}(t) + \alpha (c_{ji}(t-1) - c_{ji}(t-2)) \quad \text{(III.15)}$$

Dans les équations précédentes, η est le pas d'apprentissage et α est le moment $\eta \in [0,1]$ et $\alpha \in [0,1]$

III.3.2 Procédure d'apprentissage dans la modélisation :

Procédure d'apprentissage dans le cas de la modélisation est donné comme suit :

A chaque instant k :

Calcul du signal d'entrée $u(k)$

Calcul de la sortie du système $y(k)$

$$X = [u(k) \quad y_m(k-1)]$$

Pour chaque neurone caché j

Calcul de h_j *en utilisant Eq.III.1*

Calcul de la sortie du réseau $y_m(k)$ *en utilisant Eq.III.8*

Calcul de l'erreur $e_m(k) = y(k) - y_m(k)$

Pour chaque neurone caché j
Calcul de $\Delta w_j(k)$ *en utilisant Eq.III.10*
Calcul de $\Delta b_j(k)$ *en utilisant Eq.III.13*
Pour i=1 jusqu'à 2
Calcul de $\Delta c_{ji}(k)$ *en utilisant Eq.III.14*
Adaptation des paramètres des RBFs :
$c_{ji}(k)$ *En utilisant Eq.III.15*
$b_j(k)$ *En utilisant Eq.III.13*
Adaptation des poids :
$w_j(k)$ *En utilisant Eq.III.12*

III.3.3 Application sur 1er modèle :

Pour évaluer la capacité de modélisation des réseaux RBF, considérons le modèle non linéaire discret suivant [15][16][17] :

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

Le signal d'entrée est :

$$u(k) = sin(pi \times k \times te/25) \; + \; sin(pi \times k \times te/10)$$

Nous utilisons un réseau RBF avec 5 neurones cachées, donc d'architecture 2-5-1. Les valeurs initiales des poids sont choisies aléatoirement entre 0 et 1 et les paramètres initiaux de la gaussienne sont comme suit : les centres sont répartis sur l'intervalle $[-5\ 5]$:

$C = [-3 \quad -1.5 \quad 0 \quad 1{,}5 \quad 3]$ et la largeur $b = 1.5$.

La figure III.5 illustre les fonctions RBFs avant et après l'apprentissage. Nous remarquons que l'algorithme d'apprentissage a permis d'élargir les largeurs des fonctions gaussiennes (représentée en discontinue). Ceci permis d'améliorer la couverture de l'intervalle d'entrée.

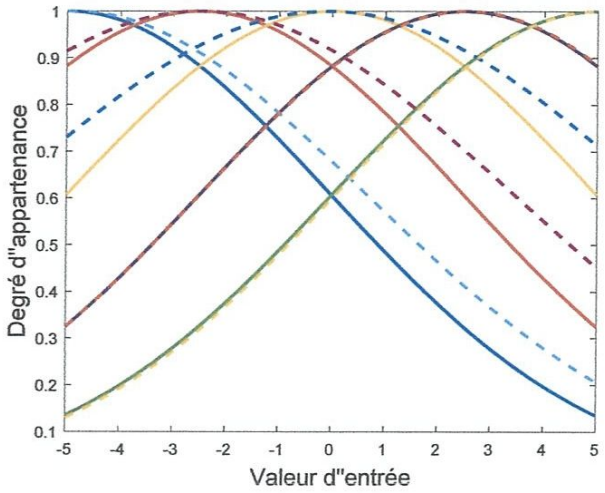


Figure III.5 : Les fonctions RBF du 1er modèle avant et après l'apprentissage

La figure III.6 illustre la réponse du système et l'erreur entre la sortie du modèle et du système. Nous remarquons que la sortie du réseau suit parfaitement le système. Nous remarquons aussi la présence de quelques pics dans l'erreur durant le premier cycle puis il disparaisse et l'erreur converge vers 0. Ceci démontre les capacités d'apprentissage de notre système.

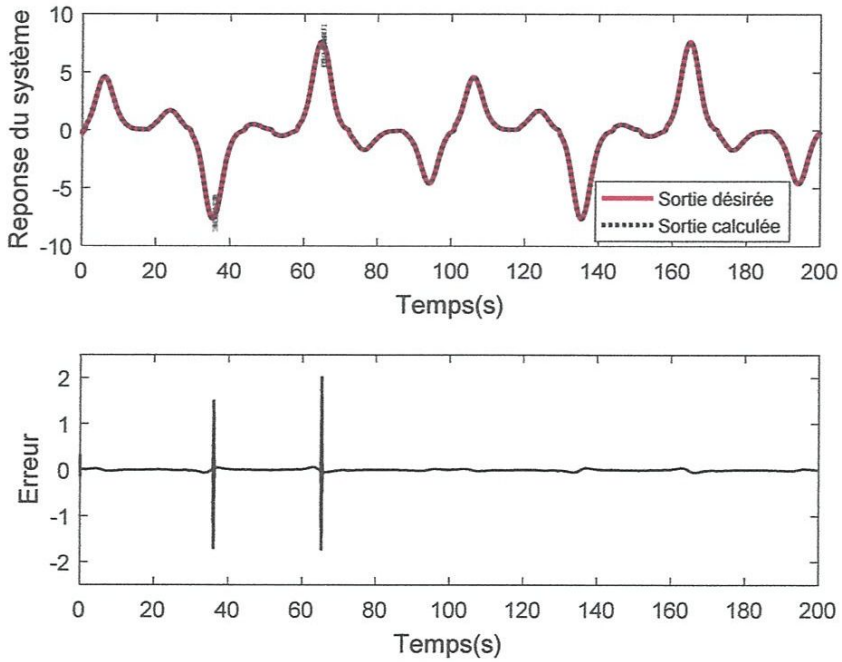


Figure III.6 : La réponse et l'erreur du système en utilisant le 1er modèle

Pour analyser l'effet du nombre des fonctions RBF utilisées, nous avons effectué un deuxième test avec différents nombres de RBFs, à savoir 3,5 et 7 RBFs. La figure III.7 représente les erreurs correspondantes aux différents nombres des RBFs, c.à.d. le nombre de

neurones dans la couche cachée. D'après les résultats obtenus nous voyons qu'il existe une relation inversement proportionnelle entre le nombre de couche cachées et l'erreur ; plus le nombre des RBFs augmente plus l'erreur diminue. Plus précisément, nous remarquons que le système permis de données des résultats satisfaisants à partir de 5 RBFs.

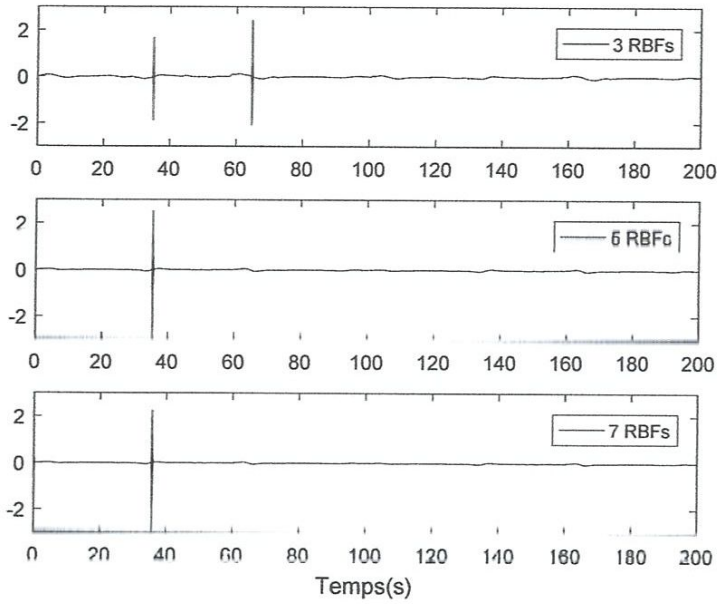


Figure III.7 : Erreurs correspondantes aux différents nombres des RBFs

Pour analyser l'effet de l'initialisation des RBFs, Nous avons réalisé un troisième test avec 3 RBFs. Dans la première configuration, les fonctions RBFs sont bien reparties tandis que dans la deuxième, la répartition est non uniforme. C.à.d., en choisissant un vecteur de centre qui ne recouvre pas tous l'intervalle des entrées. La figure III.8 représente les deux cas d'initialisation (répartie et non répartie) avant et après l'apprentissage. Nous remarquons que notre algorithme d'apprentissage améliore la répartition des fonctions gaussiennes sur l'intervalle des entrées.

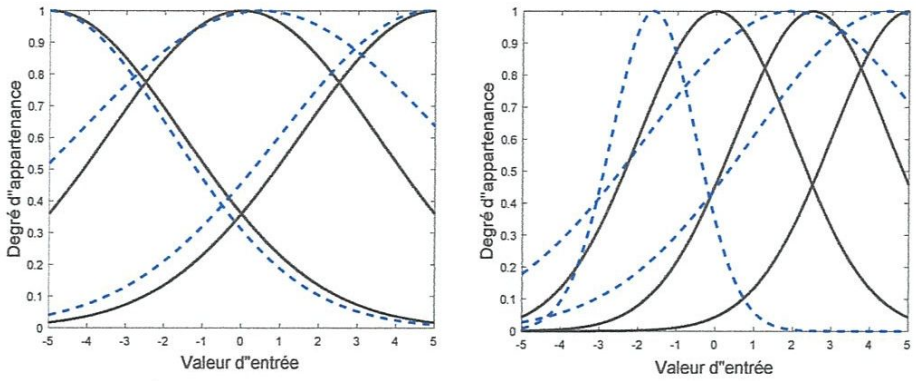


Figure III.8 Les fonctions RBFs avant et après l'apprentissage.
(a) : RBFs réparties, (b) : RBFs non réparties.

La figure III.9 représente l'erreur pour les trois cas ; non répartie, bien repartie, et réparties sans adaptation des paramètres de fonction gaussiennes (centres et largeurs).

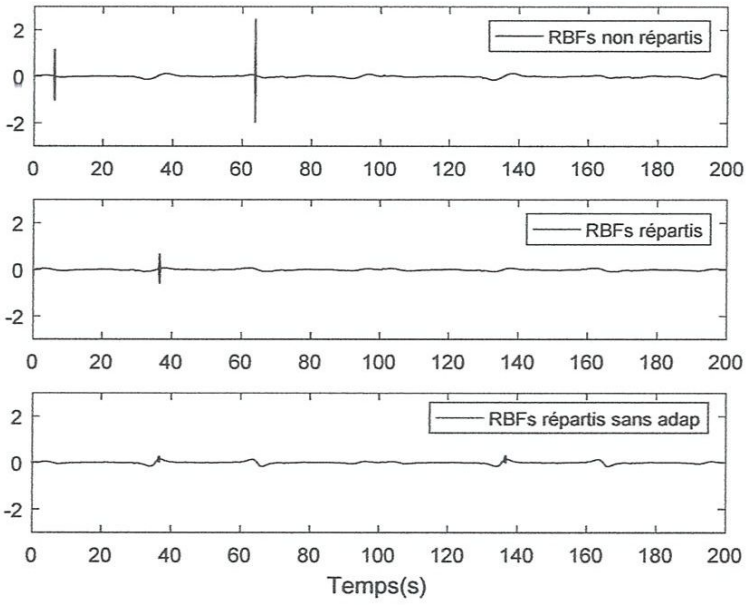


Figure III.9 : L'erreur du système dans les cas de RBFs :
Non répartie, bien repartie, et bien réparties sans adaptation

D'après La figure III.9, la distribution uniforme des fonctions gaussiennes sur l'intervalle des entrées rend le système mieux performant. Nous remarquons que l'apprentissage a permet d'améliorer les performances soit dans le cas des RBFs bien repartie ou non. Nous remarquons également que lorsque les RBFs sont bien réparties le système peut

donner des résultats satisfaisants sans adaptation des paramètres de fonctions gaussiennes (centre, largeur)

III.3.4 Application sur le 2éme modèle :

Pour évaluer davantage les capacités de modélisation des réseaux RBF, considérons un deuxième modèle [15][16]:

$$y(k) = u(k-1) + \frac{0.5y(k-1)(1-y(k-1))}{1+\exp(-0.25y(k-1))}$$

Le signal d'entrée est :

$$u(k) = sin(pi * k * te/50) * cos(pi * k * te/30)$$

Nous utilisons un réseau RBF avec 3 neurones cachées, donc d'architecture 2-3-1. Les valeurs initiales des poids sont choisies aléatoirement entre 0 et 1.

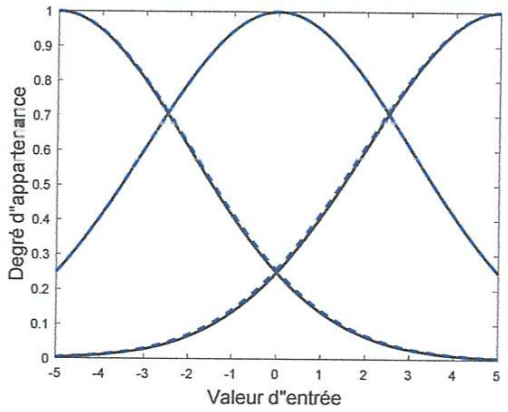


Figure III.10 : Les fonctions RBF avant et après l'apprentissage pour le 2ème modèle.

La figure III.10 illustre les fonctions RBFs avant et après apprentissage pour le 2ème modèle. Nous remarquons que l'algorithme d'apprentissage n'a pas changé les paramètres des fonctions gaussiennes.

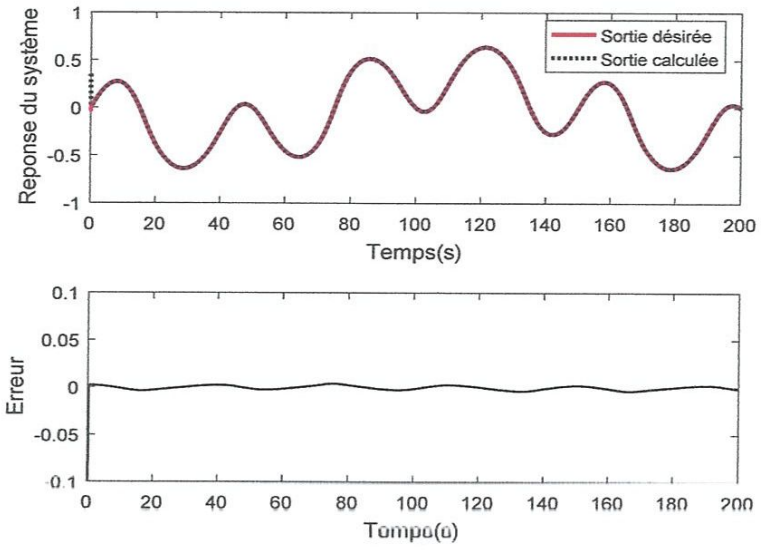


Figure III.11 : La réponse et l'erreur du système en utilisant le 2eme modèle.

La figure III.11 illustre la réponse du système et l'erreur entre la sortie du modèle et du système. Nous remarquons que la sortie du réseau suit parfaitement le système et l'erreur est très petite. Ceci démontre encore les capacités de modélisation et d'apprentissage des réseaux RBF.

III.4 Commande adaptative avec modèle de référence basée les réseaux RBF

III.4.1 Principe :

Dans cette partie nous étudions la possibilité d'utilisation des réseaux RBF dans un système de commande adaptative avec modèle de référence. L'objectif est d'exploiter les capacités d'apprentissage des réseaux RBF pour la mise en œuvre du contrôleur. Dans ce type de commande adaptative, expliquée dans le paragraphe II.2.2, le comportement souhaité du système est spécifié par un modèle, et les paramètres du contrôleur sont ajustés en fonction de la différence entre la réponse du système et celle du modèle. La figure III.12 illustre un système de commande adaptative avec modèle de référence dans lequel un réseau RBF est utilisé comme un contrôleur. Le signal d'entrée r est appliqué au modèle de référence et le signal u issue du contrôleur neuronal est appliqué au système. L'erreur entre la sortie du modèle de référence et le système est utilisé pour l'adaptation du réseau RBF.

La sortie du modèle de référence est $y_m(k)$, alors l'erreur de suivi, e_c , est donnée par:

$$e_c(k) = y_m(k) - y(k) \qquad \text{(III.16)}$$

La fonction objective utilisée est :

$$E(k) = \frac{1}{2} e_c(k)^2 \tag{III.17}$$

La sortie du contrôleur neuronale est donnée par

$$u = h_1.w_1 + \dots . h_j.w_j + h_m.w_m \tag{III.18}$$

Où m est le nombre de réseaux de neurones dans la couche cachée, w_j est la valeur de poids et h_j est sortie de la fonction gaussienne.

Dans le réseau RBF, $x = [x_1 \dots ., x_n]^T$ est le vecteur d'entrée, $h = [h_1 \dots h_m]^T$, h_j est fonction gaussienne correspondante au $j^{ème}$ neurone caché donnée par :

$$h_j = exp\left(\frac{\| x(k) - c_j \|^2}{2b_j^2}\right) \tag{III.19}$$

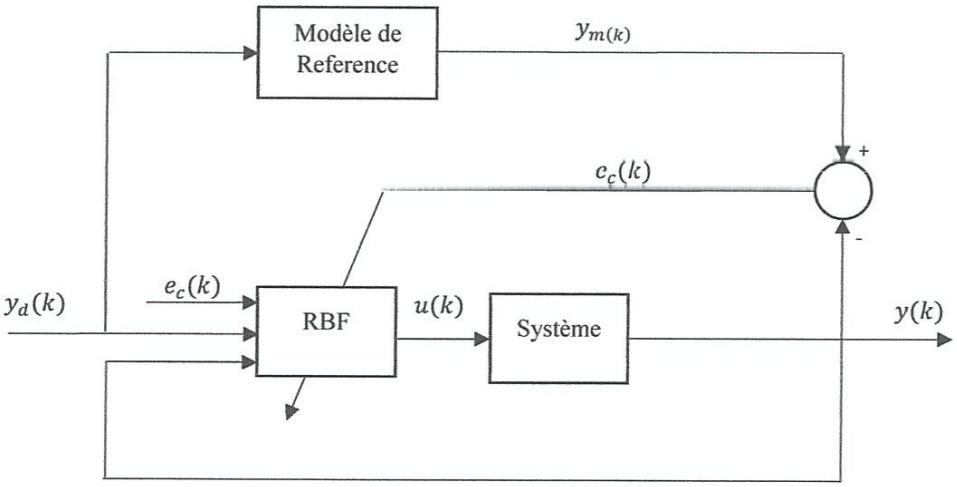


Figure III.12 : Commande adaptative avec modèle de référence basé sur un réseau RBF.

Selon la méthode de descente du gradient, l'adaptation des poids est effectué comme suit [15] :

$$\Delta w_j(k) = -\eta \frac{\partial E(k)}{\partial w} = -\eta e_c(k) \frac{\partial y(k)}{\partial u(k)} . h_j \tag{III.20}$$

La mise à jour du poids w_j a l'instant t est alors donnée par :

$$w_j(k) = w_j(k-1) + \Delta w_j(k) + \alpha \Delta w_j(k) \tag{III.21}$$

Où η est le pas d'apprentissage et α est le moment $\eta \in [0,1]$, $\alpha \in [0,1]$,

L'adaptation des largeurs de fonctions RBFs s'effectue comme suit :

$$\Delta b_j(\mathrm{k}) = \eta \frac{\partial E(k)}{db_j} = \frac{\partial y(k)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial b_j} \quad \text{(III.22)}$$

$$\Delta b_j(\mathrm{k}) = \eta . e_c(k) \frac{\partial y(k)}{\partial u(k)} w_j h_j \frac{\| x - c_{ji} \|^2}{b_j^3} \quad \text{(III.23)}$$

La mise à jour de la largeur b_j a l'instant k est alors donnée par :

$$b_j(\mathrm{k}) = b_j(k-1) + \eta \Delta b_j + \alpha(b_j(k-1) - b_j(k-2)) \quad \text{(III.24)}$$

$$\Delta c_{ji}(\mathrm{k}) = -\eta . \frac{\partial E(k)}{dc_{ji}} = \frac{\partial y(k)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial c_{ji}} \quad \text{(III.25)}$$

$$\Delta c_{ji}(\mathrm{k}) = \eta . e_c(k) \frac{\partial y(k)}{\partial u(k)} w_j h_j \frac{x_i - c_{ji}}{b_j^2} \quad \text{(III.26)}$$

L'adaptation des centres des fonctions RBFs s'effectue comme suit :

La mise à jour du centre c_{ij} a l'instant k est alors donnée par :

$$c_{ij}(k) = c_{ij}(k-1) + \eta \Delta c_{ij} + \alpha(c_{ij}(k-1) - c_{ij}(k-2)) \quad \text{(III.27)}$$

Dans les équations précédentes, η est le pas d'apprentissage et α est le moment $\eta \in [0,1]$ et $\alpha \in [0,1]$

III.4.2 Procédure d'apprentissage dans la commande :

La procédure de la commande adaptative avec modèle de référence basée sur un réseau RBF est donnée comme suit :

A chaque instant k

Calcul du signal d'entré $y_d(k)$
Calcul de la sortie du modèle de référence $y_m(k)$
Calcul de la sortie du système $y(k)$
$X = [y_d(k) \quad y(k-1) \quad e_c(k-1)]$
Pour chaque neurone caché j
Calcul de h_j *en utilisant Eq.III.19*
Calcul de la sortie du réseau $y_m(k)$ *en utilisant Eq.III.18*
Calcul de l'erreur $e_c(k) = y_m(k) - y(k)$
Pour chaque neurone caché j
Calcul de $\Delta w_j(k)$ *en utilisant Eq.III.20*
Calcul de $\Delta b_j(k)$ *en utilisant Eq.III.23*
Pour i=1 jusqu'à 3
Calcul de $\Delta c_{ji}(k)$ *en utilisant Eq.III.26*
Adaptation des paramètres des RBFs :
$c_{ji}(k)$ *En utilisant Eq.III.27*
$b_j(k)$ *En utilisant Eq.III.24*

Adaptation des poids :

w_j *En utilisant Eq.III21*

III.4.3 Application :

On considère le système non linéaire discret suivant [15][16] :

$$y(k) = -0.1y(k-1) + \frac{u(k-1)}{(1+y(k-1)^2)}$$

Le temps d'échantillonnage $t_s = 1ms$,

le modèle de référence est :

$$y_m(k) = 0.6\, y_m(k-1) + u(k)$$

le signal d'entrée est

$$y_d(k) = 0.5 \sin(2\pi k t_s)$$

La figure III.13 représente les sept fonctions gaussiennes utilisées par la conception du contrôleur neuronale. Nous avons initialisé ces fonctions réparties uniformément sur l'intervalle d'entrée afin d'obtenir une bonne couverture des entrées. Nous avons aussi adopté un apprentissage des poids seulement sont adaptation des fonctions RBFs afin d'accélérer l'apprentissage du contrôleur. En effet, la lenteur d'apprentissage du contrôleur constitue l'un des inconvénients de la commande adaptative et par l'adaptation des poids seulement on a essayé d'éviter ce problème.

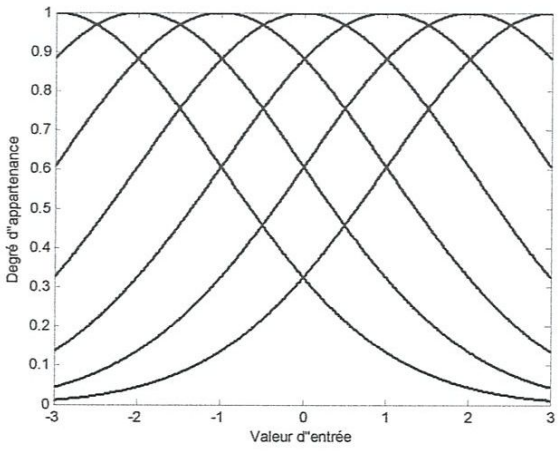


Figure III.13 : Les fonctions RBFs utilisés pour le contrôleur neuronal.

La figure III.14 illustre la réponse du système, Nous remarquons, qu'après quelque oscillation au début, la sortie du système, $y(k)$, suit parfaitement la sortie de modèle de référence $y_m(k)$. Ceci démontre les capacités d'apprentissage du contrôleur neuronal.

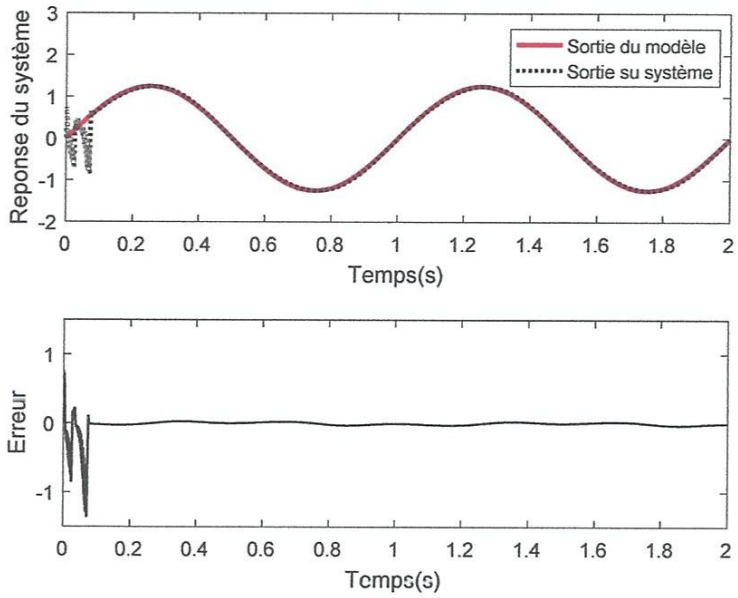


Figure III.14 : La réponse du système, du modèle et l'erreur.

III.5 Conclusion :

Dans ce chapitre nous avons étudié, en premier lieu, la modélisation par les réseaux de neurones RBFs en utilisant deux modèles théoriques. Nous avons analysé l'effet d'adaptation des paramètres des fonctions RBFs, l'effet de leur nombre et de leur initialisation. Nous avons tiré les remarques suivantes :

- les Réseaux RBF ont de bonnes capacités d'apprentissage. Ceci est caractérisé par convergence de l'erreur vers « 0 ».
- Lorsque les RBFs sont initialement bien réparties, le système peut donner des résultats satisfaisants sans adaptation des paramètres de fonctions gaussiennes (centres, largeurs)
- Le nombre des fonctions RBFs dépond du système.

Ensuite nous avons étudié la possibilité d'utilisation des réseaux RBF dans un système de commande adaptative avec modèle de référence. Nous avons constaté également de bonnes performances des réseaux RBF en tant que contrôleurs.

Conclusion générale

L'objectif de ce travail était d'analyser les performances des réseaux de neurones à fonction radiale dans la modélisation et la commande.

Nous avons étudié, en premier lieu, la modélisation par les réseaux de neurones RBFs en utilisant deux modèles théoriques. Nous avons analysé l'effet d'adaptation des paramètres des fonctions RBFs, l'effet de leur nombre et de leur initialisation. Nous avons constaté que les réseaux RBF ont de bonnes capacités d'apprentissage ce qui leur a permis de former de bon modèles. Nous avons remarqué que le processus d'apprentissage a permis de bien ajuster les paramètres des fonctions RBFs pour couvrir convenablement l'intervalle des entrée. Nous avons constaté également que l'initialisation des fonctions RBF est d'importance capital. En effet, lorsque ces fonctions sont initialisées avec une bonne répartition on peut se passer de leur adaptation ce qui permet de rendre l'apprentissage plus rapide.

Nous avons ensuite étudié la possibilité d'utilisation des réseaux RBF dans un système de commande adaptative avec modèle de référence. Dans ce type de commande adaptative, les performances désirées du système sont spécifiées par un modèle de référence, et les paramètres du contrôleur sont ajustés en fonction de la différence entre la réponse du système et celle du modèle. L'application des réseaux de neurones dans ce type de commande consiste à les introduire en tant que contrôleur et a utilisé l'erreur entre la sortie du système et celle du modèle comme fonction coût pour leur apprentissage. Après simulation, Nous avons constaté également de bonnes performances des réseaux RBF en tant que contrôleurs.

Références

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

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

[3] Samarasinghe S., "Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition", CRC press, 2006.

[4] Looney, Carl G. Pattern recognition using neural networks: theory and algorithms for engineers and scientists. New York: oxford university press, 1997.

[5] Åström, K. J., & Wittenmark, B., Adaptive Control. Courier Corporation, 2008.

[6] Tao, G, Adaptive control design and analysis (Vol. 37). John Wiley & Sons, 2003.

[7] Omatu, S., Khalid, M. B., & Yusof, R. Neuro-control and its applications. Springer-Verlag, 1996.

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

[9] 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.

[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] 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.

[13] 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.

[14] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representation by error propagation, Parallel distributed processing: exploration in the microstructure of cognition", D. E. Rumelhart and J. L. McClelland edition, MIT press Cambridge, pp. 318-362, 1986.

[15] Liu, J., "Radial Basis Function (RBF) neural network control for mechanical systems: design, analysis and Matlab simulation". Springer Science & Business Media, 2013.

[16] Narendra, K. S., & Parthasarathy, K., "Identification and control of dynamical systems using neural networks", IEEE Transactions on neural networks, vol. 1, no 1, p. 4-27. 1990

[17] Juang, C. F., Hung, C. W., & Hsu, C. H.,"Rule-based cooperative continuous ant colony optimization to improve the accuracy of fuzzy system design", IEEE Transactions on Fuzzy Systems, vol. 22, no 4, p. 723-735, 2014.