

**République Algérienne Démocratique et Populaire**  
**Ministère de l'Enseignement Supérieure et de la Recherche Scientifique**  
**Université Abderrahmane Mira**  
**Faculté de la Technologie**



**Département d'Automatique, Télécommunication et d'Electronique**

## **Projet de Fin d'Etudes**

Pour l'obtention du diplôme de Master

Filière : Automatique

Spécialité : Automatique et Système

### **Thème**

**Commande Des Systèmes Par Réseaux De Neurones Artificielles**

**Préparé par :**

- CHEMLAL Lahna
- CHEKABA Fairouz

**Dirigé par :**

HADDAR Hocine

**Examiné par :**

Mr MENDIL.B

Mr NAIT MOHAND.N

**Année universitaire : 2022/2023**



## Remerciement

Nous tenons à remercier vivement le bon dieu tout puissant de nous avoir donné la patience, la volonté et le courage pour accomplir ce travail.

Nous tenons à remercier chaleureusement notre promoteur, Messieurs HEDDAR Hocine, pour ces conseils judicieux, sa patience, surtout pour sa conscience, son sérieux et pour le fait qu'il n'a ménagé aucun effort pour mettre à notre disposition tous les moyens nécessaires à l'accomplissement de ce travail.

Nous exprimons nos sincères remerciements au nombre de Jury Mr NAIT Mohand.N et Mr MENDIL.B pour avoir accepté d'évaluer ce modeste travail avec enthousiasme.

Nous remercierons tous les enseignants de département d'Automatique, Télécommunication et d'Electronique de l'université Bejaïa qui ont contribué à notre formation.

Et pour terminer, Nous présentons également nos sincères remerciements à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail et à celle ou celui que nous étourderie

aura omis dans cette page . . .





## Dédicaces

Avec tous mes sentiments de respect, avec l'expérience  
De ma reconnaissance, je dédie ma remise de diplôme et ma  
Joie à mon paradis, à la prunelle de mes yeux, à la source  
De ma joie et mon bonheur, ma lune et le fil d'espoir qui allumer mon  
Chemin, ma moitié, Maman.

A celui qui m'a fait une femme, ma source de vie, d'amour et  
D'affection, à mon support qui était toujours à mes côtés pour me  
Soutenir et m'encourager, à mon prince papa.

A mon frère Bakhli pour l'amour qu'i me réserve.

A mes sœurs Sonia, Djedjiga qui n'ont pas cessée de me conseiller,  
Encourager et soutenir tout au long de mes études.

A mon adorable petite sœur Foufa, qui sait toujours comment  
Procurer la joie et le bonheur pour toute la famille.

A mon encadreur Mr Heddar

Sans oublier mon binôme Fairouz pour son soutien moral sa patience  
Et sa compréhension tout au long de ce projet.

A tous mes collègues de promotion master 2 automatique et systèmes  
Sans oublier mes chers collègues de licence à l'université de Bouira

Lydia M et Lydia T et Mimi.

A tout ce qui ont participé à ma réussite et

A tous qui m'aime.

**CHEMLAL Lahna**





## Dédicaces

J'ai le grand plaisir de dédier le fruit de mes 23 ans :

A ma très chère mère, qui me donne toujours l'espoir de

Vivre et qui n'a jamais cessé de prier pour moi.

A mon très cher père, pour ses encouragements, son soutien,

Surtout pour son amour et son sacrifice afin que rien n'entrave le

Déroulement de mes études.

A mes chers frères : Aïmed, Zinedine et AD Kader pour leur amour,

Confiance et leur soutien inconditionnel tout au long de ma scolarité.

A Mes chères sœurs : Salima et Nadjet en signe de gratitude et

D'amour pour leur encouragements, brouté et vivacité.

A mon encadreur Mr Heddar

A mes meilleurs amis : Laila, Nawel, ...

A tous mes collègues de promotion master 2 Automatique

A tout qui m'aide et compulse ce Projet fin d'étude

En fin, je remercie ma binôme Lahna qui a contribué à la réalisation

De ce modeste travail.

Et A tout ce qui ont participé à ma réussite et

A tous qui m'aime.

**CHEKABA Fairouz.**





## Sommaire

<b>Sommaire</b> .....	I
<b>Liste des Figures</b> .....	III
<b>Liste des Tableaux</b> .....	V
<b>Liste des Abréviations</b> .....	VI
<b>Introduction générale</b> .....	1
 <b>CHAPITRE I : Les Réseaux De Neurone Artificiel</b> 	
I.1.Introduction : .....	3
I.2. Historique : [1] .....	3
I.3. Définition d'un réseau de neurone : .....	4
I.4. Notions de neurophysiologie : .....	5
I.4.1. Neurone biologique : .....	5
I.4.2.Neurone formel (Artificiel) : .....	5
I.4.2.1. La modélisation d'un neurone formel : .....	6
I.4.2.2.Composition d'un réseau de neurones : .....	7
I.4.2.3.Fonction d'activation : .....	7
I.5.Architecture des réseaux de neurones artificiels (RNA) : .....	9
I.5.1. Les réseaux non bouclés : .....	9
I.5.2.Les réseaux bouclés ou récurrents : .....	9
I.6.Apprentissage des réseaux de neurones : .....	10
I.6.1. Définition : .....	10
I.6.2.Types d'apprentissage des réseaux de neurones : .....	11
I.6.2.1.Apprentissage supervisé : .....	11
I.6.2.2.Apprentissage non-supervisé : .....	11
I.6.3.Algorithmes d'apprentissage du perceptron multicouche : .....	12
I.6.3.1. L'algorithme de la rétro propagation du gradient : .....	12
II.6.3.2.L'algorithme de Levenberg-Marquardt : .....	13
II.7.Réseaux de neurones dans le contrôle des systèmes : .....	14

II.7.1. Stabilisation du système non linéaire par un réseau de neurones :.....	14
II.8. Applications d'aujourd'hui : .....	15
II.8. Conclusion : .....	16
<b>CHAPITRE II : Commande des systèmes</b>	
II.1. Introduction :.....	17
II.2. La Commande PID :.....	17
II.3. Commande par retour d'état : .....	19
II.3.1. Structure de la loi de commande [8] :.....	19
II.3.2. Contrôlabilité :.....	19
II.3.3. Placement de pôles par retour d'état .....	19
II.3.3.1. Principe : .....	20
II.3.3.2. Calcul du gain du retour d'état :.....	21
II.4. Commande linéaire quadratique (LQR) :.....	22
II.4.1. Présentation de la méthode LQR :.....	22
II.4.2 Le critère du compromis :.....	23
II.4.3 Choix des matrices de pondérations Q et R :.....	23
II.4.4 Recherche du gain du retour K :.....	24
II.5. Commande par réseaux de neurones artificiels :.....	24
II.5.1. Commande adaptative par modèle de référence (MRAC) :.....	24
II.5.2 Contrôle neuronal adaptatif :.....	25
II.5.2.1 La commande neuronale adaptative directe :.....	25
II.5.2.2 La commande neuronale adaptative indirecte : .....	26
II.5.3 Identification neuronale :.....	27
II.5.3.1. Identification des systèmes :.....	27
II.5.3.1.1. Étape qualitative :.....	28
II.5.3.1.2. Étape quantitative :.....	30
II.5.3.2 Structure d'identification :.....	30
II.5.3.2.1. Modélisation directe :.....	31



---

II.5.3.2.2 Modélisation Inverse :.....	32
II.6. Conclusion : .....	33
<b>Chapitre III : Commande neuronale des systèmes dynamiques.</b>	
III.1. Introduction :.....	33
III.2. Commande NARMA-L2 : .....	33
Exemple 2 :.....	41
III.3. Commande neuronale à base de modèle de référence : .....	51
III.4. Conclusion : .....	62
<b>Conclusion générale</b> .....	64
<b>Bibliographie</b> .....	
<b>Résumé</b> .....	

Liste des Figures

**Figure.I-1** : schéma de neurone biologique .....5

**Figure.I-2** : modèle d'un neurone formel ..... 6

**Figure.I-3** : Architecture d'un réseau de neurone. ....7

**Figure.I-4** : Réseau de neurones non bouclé.....9

**Figure.I-5** :Réseau de neurones bouclé..... 10

**Figure.I-6** :Forme canonique de l'apprentissage supervisé. .... 11

**Figure.I-7** :Forme canonique de l'apprentissage non-supervisé..... 12

**Figure.II-1**:Commande PID ..... 18

**Figure.II-2**:Principe du placement du pole ..... 20

**Figure.II-3**:Structure MRAC indirecte /RP ..... 25

**Figure.II-4** : Schéma de structure de la commande neuronale adaptative directe..... 26

**Figure.II-5** : Structure de la commande neuronale adaptative Indirecte ..... 27

**Figure.II-6** : les modèles d'identification entrée-sortie ..... 29

**Figure.II-7** : Structure d'identification série-parallèle..... 31

**Figure.II-8** : Structure d'identification parallèle..... 31

**Figure.II-9** :Structure de modélisation inverse (par apprentissage généralisé). .... 32

**Figure.III-1** : Modèle neuronal de type NARMA-L2..... 35

**Figure.III-2** : Contrôleur neuronal de type NARMA-L2 ..... 35

**Figure.III-3** :Schéma bloc de la commande par réseau de neurones sous Simulink ..... 36

**Figure.III-4** :Simulation en boucle fermée de la commande NARMA-L2 pour le cas de contrôleur linéaire exacte ..... 38

**Figure.III-5** : Simulation en boucle fermée de la commande NARMA-L2 pour le cas linéaire avec prise en considération de la linéarité ..... 39

**Figure.III-6** :Simulation de la commande NARMA-L2 pour le cas linéaire. .... 41

**Figure.III-7** : Réponse du système linéaire utilisé avec le contrôleur NARMA-L2 non linéaire avec 10 neurones dans les couches 1 et 3..... 42

**Figure.III-8** : Modèle neuronal non cyclique (Feedforward) utilisé pour l'apprentissage ..... 43

**Figure.III-9** : Les entrées et sortie du système non-linéaire utilisés pour l'apprentissage..... 44

**Figure.III-10** : Evolution de la moyenne de la somme des carrées de l'erreur à travers les différentes époques d'apprentissage pour le modèle du système non linéaire ..... 45

**Figure.III-11** : Histogramme de l'erreur du modèle du système non linéaire ..... 46

**Figure.III-12** : Régression linéaire entre les sorties du système non linéaire et son modèle neuronal. .... 47

<b>Figure.III-13</b> : Auto corrélation de l'erreur du modèle neuronal du système non linéaire .....	48
<b>Figure.III-14</b> : Inter corrélation entre l'erreur et l'entrée du modèle neuronal du système non linéaire.....	49
<b>Figure.III-15</b> : Réponse en boucle fermée du système non linéaire avec contrôleur NARMA-L2 (cas de référence égale au signal d'apprentissage. ....	50
<b>Figure.III-16</b> : Réponse en boucle fermée du système non linéaire contrôlé par NARMA-L2 (cas d'une entrée différente du signal d'apprentissage .....	52
<b>Figure.III-17</b> : Structure de la Commande Neuronale à modèle de référence de systèmes dynamiques .....	53
<b>Figure.III-18</b> : Structure du réseau de neurone utilisé comme modèle pour le système.....	53
<b>Figure.III-19</b> : Bloc diagramme utilisé pour la génération des données d'apprentissage (a. le système, b. Le contrôleur, c. Structure en boucle fermée).....	55
<b>Figure.III-20</b> : Signaux utilisés pour l'apprentissage du modèle.....	55
<b>Figure.III-20 .b</b> : Evolution du MSE pendant l'entraînement du modèle neuronal en boucle ouverte. ....	56
<b>Figure III-20 .c</b> : Evolution de l'erreur quadratique moyenne pendant l'apprentissage en boucle fermée ainsi que la régression linéaire entre la sortie du réseau de neurone et la sortie du système. ....	57
<b>Figure III-20 .d</b> : Réponse du système et réponse du modèle neuronal ainsi que l'erreur entre eux.....	58
<b>Figure III-21</b> : Réseau de neurone contenant le modèle et le contrôleur .....	59
<b>Figure III-22</b> : Entrée et sortie du modèle de référence utilisées pour l'entraînement du contrôleur .....	60
<b>Figure III-22.b</b> : Structure en boucle ouverte utilisée pour le premier apprentissage.....	61
<b>Figure.III-23</b> : Résultats de l'apprentissage en boucle fermée : a. Evolution de moyenne des carrés de l'erreur "MSE", b. Histogramme de l'erreur, c. Régression linéaire entre la réponse du modèle de référence et celle du réseau de neurone. ....	62
<b>Figure.III-24</b> : Réponse du réseau de neurone en boucle fermée avec celle du modèle de référence ainsi que l'erreur entre les deux.....	63

**Liste des Tableaux**

**Tableau I-1** : Modélisation des éléments du neurone biologique..... 6

**Tableau I-2** : Fonction de transfert ..... 7

## Liste des Abréviations

- PID Proportional Integral Derivative
- LQR Linear Quadratic Regulator
- RNA Réseaux de Neurones Artificiels
- MRAC Model Reference Adaptive Control
- MLP Multi-Layer Perceptron
- RP Retro Propagation
- MSE Mean square error
- LM Levenberg Marquardt
- NARX Nonlinear Autoregressive Network with Exogenous Inputs
- NARMA Non Linear Auto Regressive Moving Average
- TANSIG tangent sigmoid function
- Det Determinant
- LTI Linéaire à Temps Invariant
- SISO Single Input Single Output
- PMC Perceptron Multi Couche
- TDL Tipt Delay Line

# **Introduction**

# **Générale**

# Introduction générale

---

Dans le domaine de l'automatique, l'étude des systèmes non linéaires reste un domaine d'étude très attractif au sein de la communauté des ingénieurs et chercheurs en automatisme. Contrairement aux systèmes linéaires pour lesquels l'automatisme fournit tout un ensemble de méthodes d'analyse et de synthèse des lois de commande, les systèmes non linéaires ne disposent pas d'outils et de méthodes générales d'analyse et de synthèse. En effet, les systèmes non linéaires ont des structures extrêmement diverses, une dynamique complexe et peuvent présenter toutes sortes de comportements étranges. Les études de stabilité sont généralement réalisées à l'aide d'un modèle mathématique décrivant la dynamique du système étudié, dont l'analyse est effectuée par plusieurs méthodes connues dans la littérature. Ces méthodes conduisent en fait à la détermination des conditions et de la stabilité du système étudié. Les conclusions sur la stabilité du système tirées des modèles descriptifs ne sont pas toujours directement applicables aux systèmes physiques réels, et il est nécessaire de souligner certaines hypothèses et contraintes de mise en œuvre en temps réel. Le concept de stabilité des systèmes non linéaires a fait l'objet d'une abondante littérature depuis le siècle dernier (Kraokovski, Lasalle, Poincaré). [24]

Les méthodes de contrôle classiques ont été largement utilisées dans de nombreux problèmes de contrôle industriel. Cependant, la plupart des systèmes physiques présentent des non-linéarités, et leurs paramètres sont souvent mal connus et/ou variables dans le temps. Pour le contrôle de tels systèmes, les méthodes de contrôle traditionnelles montrent leurs limites en termes de stabilité et de performances. Récemment, avec le développement des ordinateurs numériques, les ingénieurs en contrôle se sont intéressés à de nouvelles méthodes de contrôle, telles que le contrôle adaptatif, le contrôle prédictif, le contrôle robuste et les techniques basées sur l'intelligence artificielle. Dans ce dernier, le système est contrôlé par un réseau de neurones artificiels.

L'objectif de ce travail est l'étude et l'application d'une commande avec un réseau de neurones pour la stabilisation de systèmes non linéaires

Pour cela, le présent mémoire est organisé en trois chapitres qui sont résumés comme suit :

- Le premier chapitre expose les principes généraux des réseaux de neurones et de leur apprentissage.
- Le deuxième chapitre présente trois méthodes de commande classique : La commande PID, la commande par retour d'état, la commande linéaire quadratique. La quatrième méthode s'intéresse à la commande intelligente par réseau de neurones artificielle

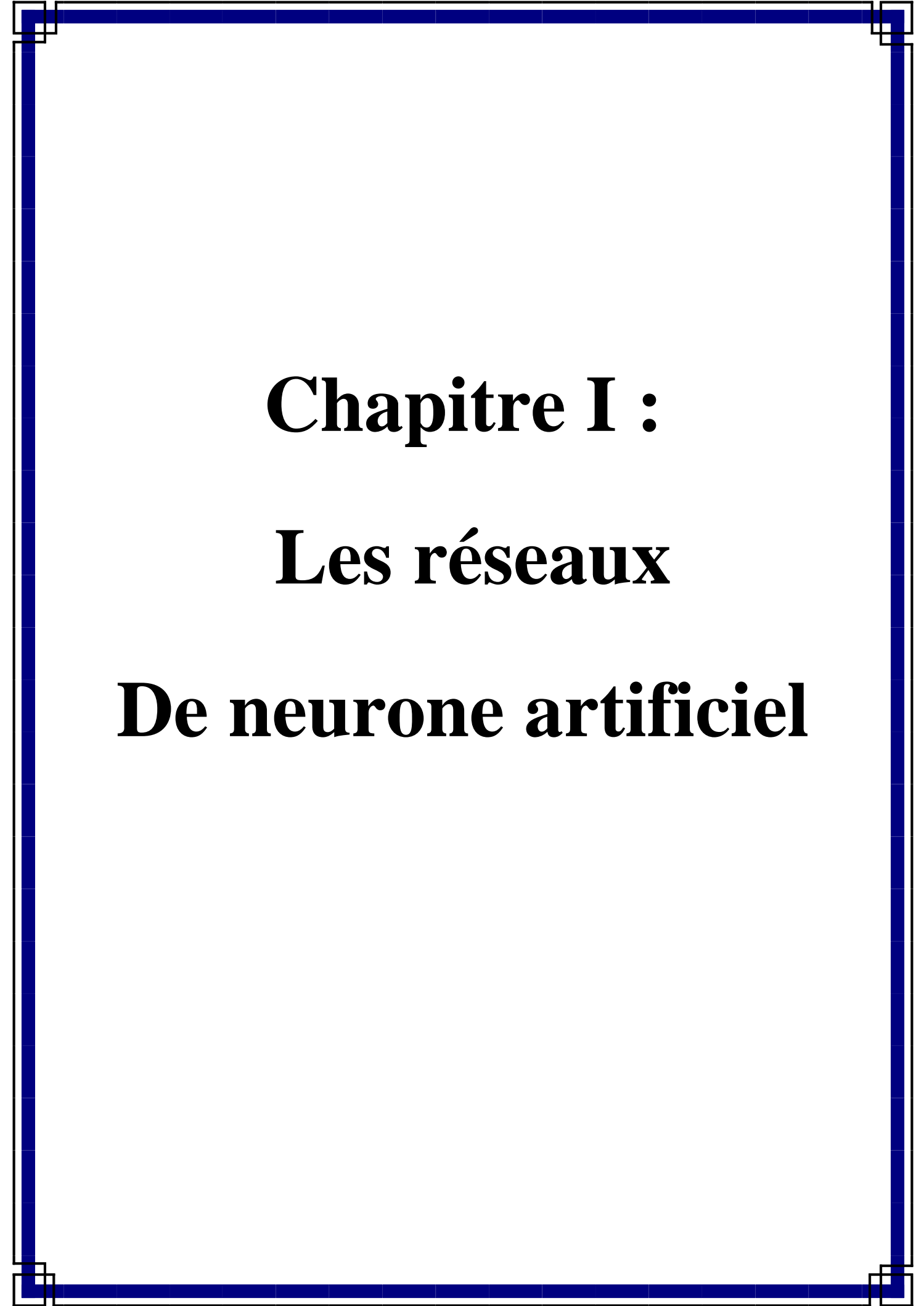
## Introduction Générale

---

➤ Le dernier chapitre est consacré à la commande de systèmes non-linéaires par réseaux de neurones.

Enfin une conclusion générale clôture le mémoire.





# **Chapitre I :**

## **Les réseaux**

### **De neurone artificiel**

# CHAPITRE I : Les Réseaux De Neurone Artificiel

---

## I.1.Introduction :

Les réseaux de neurones artificiels sont à l'origine du système nerveux humain. Leur développement découle du désir humain de comprendre et d'imiter les capacités du cerveau afin que les systèmes intégrés puissent effectuer des tâches au nom des humains.

Le réseau de neurones artificiels est constitué de plusieurs opérateurs non linéaires, à savoir des formes de neurones, qui fournissent des signaux de sortie en fonction d'un certain nombre de signaux d'entrée de poids différents. Ces neurones peuvent être connectés de diverses manières et après apprentissage peuvent être utilisés pour différentes applications telles que la surveillance des processus, la modélisation, la classification, la prédiction, le diagnostic, la reconnaissance de forme, etc.

Dans ce chapitre nous introduisons le concept de réseaux de neurones artificiels, montrant sa définition, leurs fonctions d'activation et divers composants du réseau, y compris différents types d'organisations et de connexions qui nous permettent d'avoir plusieurs structures de réseaux de neurones artificiels. Ces derniers doivent être dotés de techniques assurant la collecte d'informations auprès du monde extérieur, ou de simples règles d'apprentissage permettant au réseau de s'adapter à son environnement en ajustant et mettant à jour des poids représentatifs de la force des connexions.

## I.2. Historique : [1]

Les recherches menées dans le domaine du connexionnisme ont démarré avec la présentation **en 1943** par **W. McCulloch** et **W. Pitts** d'un modèle simplifié de neurone biologique communément appelé neurone formel. Ils montrèrent également théoriquement que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes.

**En 1949, D. Hebb** initie, dans son ouvrage "The Organization of Behavior", la notion d'apprentissage. Deux neurones entrant en activité simultanément vont être associés (c'est-à-dire que leurs contacts synaptiques vont être renforcés). On parle de loi de **Hebb** et d'associationnisme.

En 1958, **F. Rosenblatt** développe le modèle du Perceptron. C'est un réseau de neurones inspiré du système visuel. Il possède deux couches de neurones : une couche de perception (sert à recueillir les entrées) et une couche de décision. C'est le premier modèle pour lequel un processus d'apprentissage a pu être défini.

S'inspirant du perceptron, **Widrow** et **Hoff**, développent, dans la même période, le modèle de l'Adaline (Adaptive Linear Element). Ce dernier sera, par la suite, le modèle de base des réseaux de neurones multicouches.

En 1969, Les recherches sur les réseaux de neurones ont été pratiquement abandonnées lorsque **M. Minsky** et **S. Papert** ont publié leur livre « Perceptrons » (1969) et ont démontré les limites théoriques du perceptron, en particulier, l'impossibilité de traiter les problèmes non linéaires par ce modèle.

En 1982, **Hopfield** développe un modèle qui utilise des réseaux totalement connectés basés sur la règle de **Hebb** pour définir les notions d'attracteurs et de mémoire associative.

En 1984 c'est la découverte des cartes de **Kohonen** avec un algorithme non supervisé basé sur l'auto-organisation et suivi une année plus tard par la machine de **Boltzman** (1985).

Une révolution survient alors dans le domaine des réseaux de neurones artificiels : une nouvelle génération de réseaux de neurones, capables de traiter avec succès des phénomènes non-linéaires : le perceptron multicouche ne possède pas les défauts mis en évidence par **Minsky**. Proposé pour la première fois par **Werbos**, le Perceptron Multicouche apparaît en 1986 introduit par **Rumelhart**, et, simultanément, sous une appellation voisine, chez **LeCun** (1985). Ces systèmes reposent sur la rétro propagation du gradient de l'erreur dans des systèmes à plusieurs couches, chacune de type **Adaline** de **Bernard Widrow**, proche du perceptron de **Rumelhart**. [1]

De nos jours, l'utilisation des réseaux de neurones dans divers domaines ne cesse de croître. Les applications en sont multiples et variées.

### I.3. Définition d'un réseau de neurone :

Un réseau de neurones artificiels (RNA) est un ensemble formel de neurones (unités de calcul simples, nœuds de processeur) connectés en couches (ou sous-groupes) et fonctionnant en parallèle. Dans le réseau, chaque sous-groupe effectue un traitement indépendant et transmet les résultats de son analyse au sous-groupe suivant. Ainsi, les informations fournies au réseau vont se propager couche par couche, de la couche

d'entrée vers la couche de sortie, soit en ne passant pas soit en passant par plusieurs couches intermédiaires (appelées couches cachées).

En règle générale (sauf pour les couches d'entrée et de sortie), chaque neurone d'une couche est connecté à tous les neurones de la couche précédente et suivante. Les réseaux de neurones artificiels ont la capacité de stocker des connaissances empiriques et de les rendre utilisables. Les compétences de traitement du réseau (ainsi que les connaissances) seront stockées dans des poids synaptiques acquis par un processus d'adaptation ou d'apprentissage.[2]

### I.4. Notions de neurophysiologie :

#### I.4.1. Neurone biologique :

Un neurone recueille souvent des signaux provenant d'autres neurones du cerveau humain via une variété de petites structures appelées dendrites. Comme celle représentée à la figure I .1 Le neurone recueille souvent des signaux provenant d'autres neurones du cerveau humain via une variété de petites structures appelées dendrites. Le neurone transmet des images d'activité électrique le long d'un support long et fin appelé axone divisé en milliers de branches. Une structure appelée synapse est située à l'extrémité de chaque branche et convertit les effets électriques qui inhibent ou excitent l'activité de l'axone en effets électriques qui inhibent ou excitent l'activité des neurones connectés. Un neurone envoie une activité électrique à la base de son axone lorsqu'il reçoit une entrée excitatrice suffisamment supérieure à son entrée inhibitrice. [3]

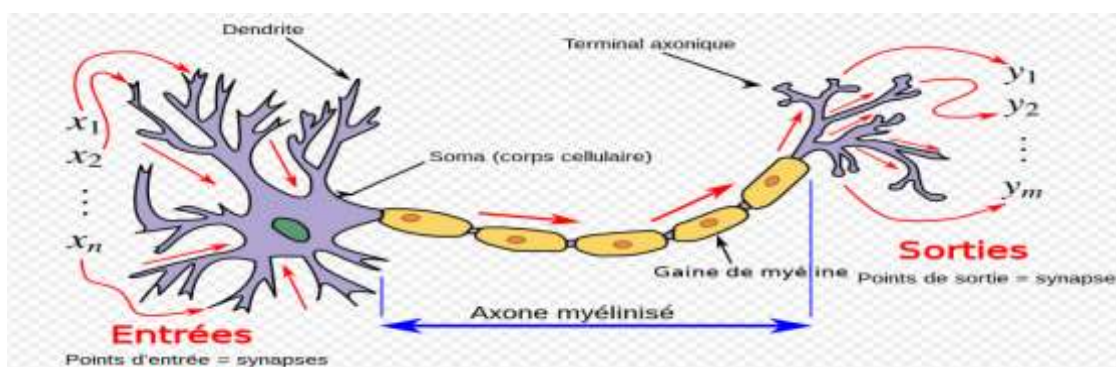


Figure I-1 : schéma de neurone biologique. [4]

#### I.4.2. Neurone formel (Artificiel) :

Un neurone artificiel (ou cellule) est un processeur de base. Il reçoit de nombreuses variables d'entrée de neurones appartenant à des niveaux amont. Chaque entrée est associée à un poids  $w$  représentant la force de la connexion. Chaque processeur élémentaire a une sortie qui se branche ensuite pour fournir des

variables à plusieurs neurones appartenant à des niveaux en aval. Chaque connexion est associée à un poids. Comme celle représentée à la figure I .2 suivante :

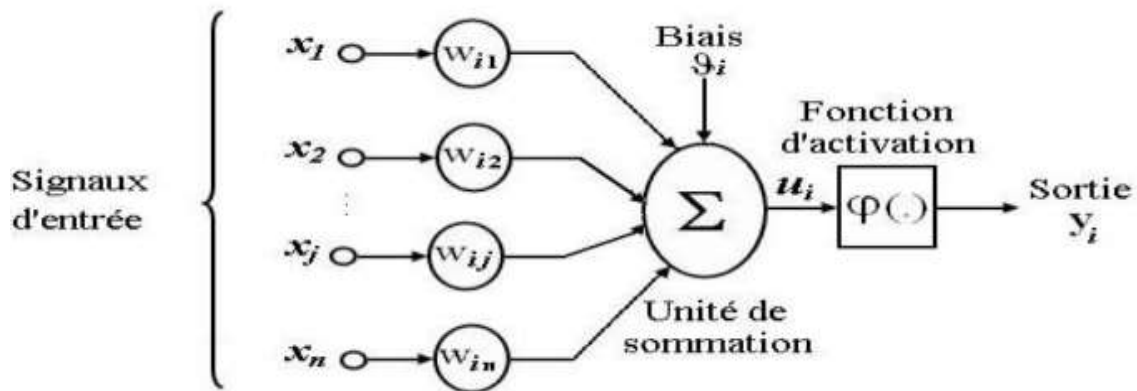


Figure I-2 : modèle d'un neurone formel. [5]

$[x_1 \dots \dots x_n]$  : Représente les entrées de neurones ;

$[w_1 \dots \dots w_n]$  : Représente le nombre de couches ;

$S = \sum_{i=1}^n w_{i,k} \times x_n + b$  : Représente la valeur de la somme pondérée ;

$\varphi$  : Représente la fonction d'activation ;

$y = \varphi(S)$  : Représente la sortie de neurone ;

$b$  : Représente le biais interne ;

**I.4.2.1. La modélisation d'un neurone formel :**

La modélisation consiste à mettre en œuvre des systèmes de réseaux de neurones aux aspects non biologiques mais artificiels. Cela signifie que, selon les principes biologiques, il existe un élément correspondant à chaque élément composant un neurone biologique, modélisant ainsi chacun d'eux. Cette modélisation est résumée dans le tableau ci-dessous. Cela nous permet de voir clairement la transition entre les neurones biologiques et formels.[3]

Tableau I-1 : Modélisation des éléments du neurone biologique. [8]

Neurone biologique	Neurone artificiel
Synapses	Poids de connexions

Axones	Signal de sortie
Dendrite	Signal d'entrée
Somma	Fonction d'activation

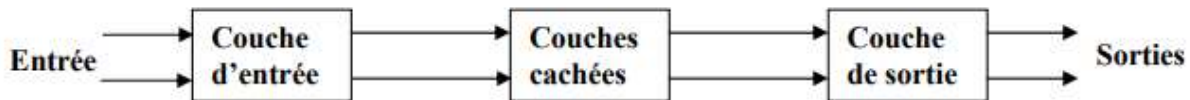
**I.4.2.2.Composition d'un réseau de neurones :**

Un RNA est constitué généralement de trois couches, comme illustre sur la figure I.3 :[6]

♦ **Une couche d'entrée** : Elle est constituée de l'ensemble des neurones du réseau qui reçoivent les données du problème. Sa taille est donc déterminée directement par le nombre de variables d'entrée.

♦ **Une couche de sortie** : Elle est constituée de l'ensemble des neurones de sortie du réseau. C'est cette couche-là qui fournit les résultats du problème.

♦ **Une ou plusieurs couches cachées** : Ce sont les couches qui se trouvent entre la couche d'entrée et la couche de sortie. Elles définissent l'activité interne du réseau. En général, les fonctions d'activations sont non linaires sur ces couches. [7]



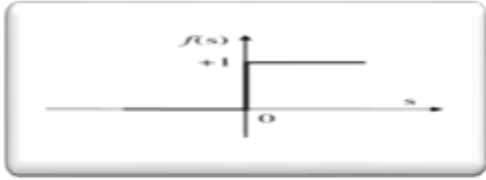

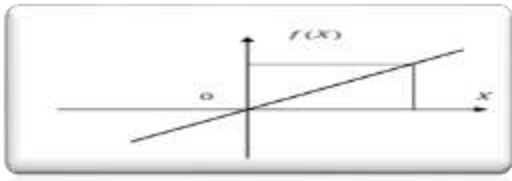
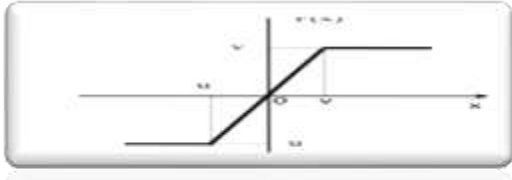


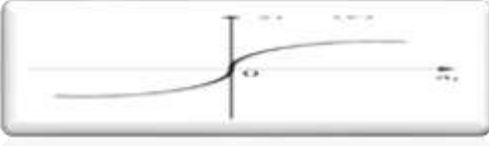

**Figure I-3 :** Architecture d'un réseau de neurone. [7]

**I.4.2.3.Fonction d'activation :**

La fonction d'activation permet de définir la réponse du neurone en fonction de son entrée totale, citons dans le tableau I.2 suivant quelques fonctions souvent utilisées [9] :

**Tableau I-2 :** Fonction de transfert. [10]

Désignation de la fonction :	L'équation mathématique :	Représentation graphique de la fonction :	Fonction MATLAB
------------------------------	---------------------------	---	-----------------

Seuil	$sl(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases}$		hardlim
Seuil symétrique	$sl(x) = \begin{cases} -1 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$		hardlims
Fonction linéaire	$F(x)=x$		purelin
Fonction linéaire à seuil ou multi-seuils	$F(x) = \begin{cases} x & \text{si } x \in [u, v] \\ v & \text{si } x \geq v \\ u & \text{si } x \leq u \end{cases}$		satlins
Linéaire saturée	$F(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ x & \text{si } 0 \leq x \leq 1 \\ 1 & \text{si } x \geq 1 \end{cases}$		satlin
Fonction sigmoïde	$f(x) = \frac{1}{1 + e^{-x}}$		logsig
Tangente hyperbolique	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$		tanh
Compétitive	$f(x) = \begin{cases} 1 & \text{si } x \text{ maximum} \\ 0 & \text{autrement} \end{cases}$		compet

## I.5. Architecture des réseaux de neurones artificiels (RNA) : [1]

Un neurone réalise une fonction non linéaire paramétrée de ses entrées. Un réseau de neurone est un maillage de plusieurs neurones organisés en couches.

On peut distinguer essentiellement deux familles de réseaux de neurones en fonction du chemin suivi par le signal qui le travers : Les réseaux non bouclés appelés aussi feed-forward et les réseaux bouclés.

### I.5.1. Les réseaux non bouclés :

Ces réseaux sont également appelés perceptrons ou réseaux de type statique, où les informations se propagent de la couche d'entrée à la couche de sortie sans revenir, en suivant les connexions on ne peut pas revenir au neurone de départ. (Fig.I.4).

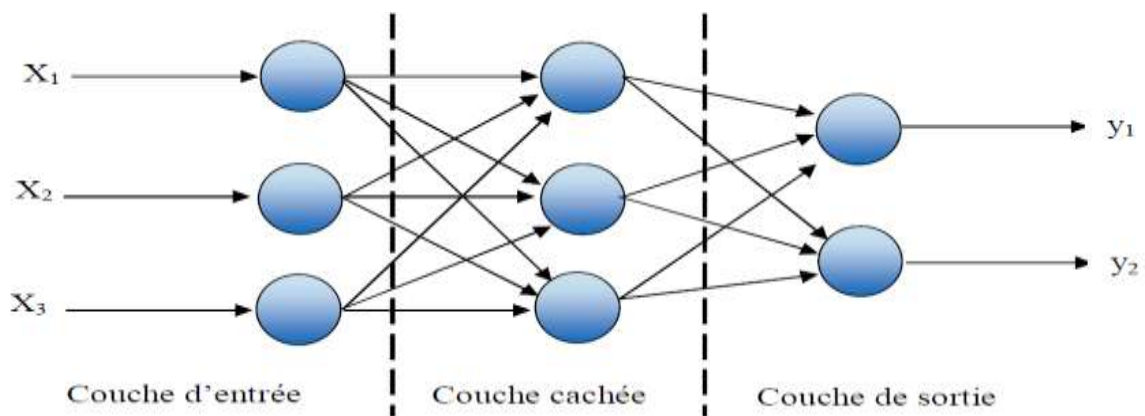
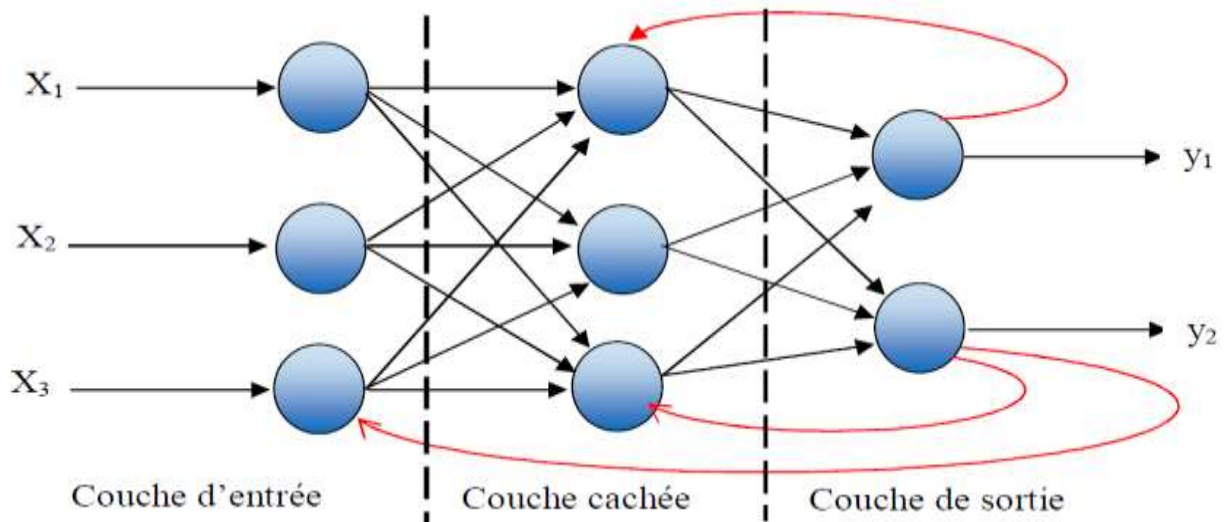


Figure I-4 : Réseau de neurones non bouclé. [1]

### I.5.2. Les réseaux bouclés ou récurrents :

Contrairement aux réseaux acycliques, les réseaux récurrents peuvent contenir des chemins récurrents qui traversent plusieurs fois le même neurone (Fig. I-5). En raison de cette structure récurrente, les stimuli entrants peuvent être partiellement ou totalement contestés par des états antérieurs du réseau ou par l'arrivée de stimuli ultérieurs. Ce type de réseau a donc une capacité théorique plus importante que les réseaux acycliques. Ainsi, les réseaux récurrents présentent une dynamique complexe due à de multiples rétroactions internes.





**Figure I-5 :** Réseau de neurones bouclé. [1].

## I.6.Apprentissage des réseaux de neurones :

Une fois l'architecture est choisie, elle doit subir une phase d'apprentissage qui est un point crucial du développement d'un réseau de neurone. Il s'agit d'une procédure adaptative par laquelle les connexions des neurones sont ajustées face à une source d'information, généralement par des algorithmes spécifiques.[7]

### I.6.1. Définition :

On appelle « apprentissage » des réseaux de neurones la procédure qui consiste à estimer les paramètres des neurones du réseau, afin que celui-ci remplisse au mieux la tâche qui lui est affectée.

Autrement dit, c'est une phase du développement du réseau de neurones durant laquelle on calcule les poids des neurones de telle manière que les sorties du réseau soient aussi proches que possible des sorties désirées. A la fin de cette phase, le réseau converge vers un fonctionnement adapté au problème qu'on désire résoudre, tout en fournissant, au préalable des exemples d'apprentissage. Ces derniers doivent être suffisamment représentatifs ; autrement dit ; il faudra qu'ils couvrent aussi exhaustivement que possible le domaine de fonctionnement désiré pour le réseau. Cette technique permet de conserver au réseau un comportement adapté malgré les fluctuations dans les données d'entrées. [7]

## I.6.2.Types d'apprentissage des réseaux de neurones :

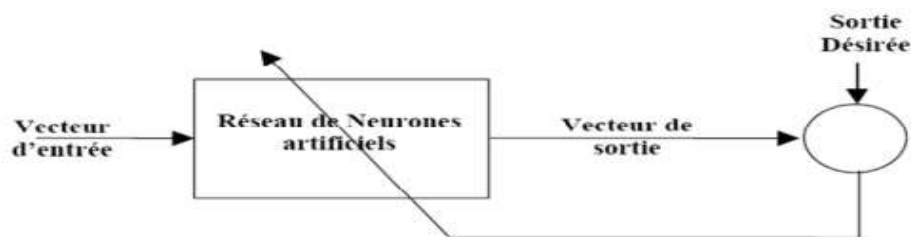
Dans le cadre de la définition précédemment citée et au niveau des algorithmes d'apprentissage (règles d'apprentissage), il a été défini deux grandes classes : apprentissage supervisé, apprentissage non-supervisé. Mais l'objectif fondamental de l'apprentissage demeure le même : soit la classification, l'approximation de fonction ou encore la prévision.

### I.6.2.1.Apprentissage supervisé :

L'apprentissage est dit supervisé lorsque les exemples sont constitués de couples de valeurs (Base d'apprentissage) du type : « valeur d'entrée, valeur de sortie désirée » (on présente au réseau de neurone les entrées et les sorties désirées correspondantes). (Comme dans le Fig.I.6).

Tout le problème de l'apprentissage supervisé consiste : étant donné un vecteur d'apprentissage de  $n$  couples  $(x_i, y_i)$ ,  $i=1 \dots, n$  ; à faire adapter le réseau par une comparaison entre les résultats qu'il a calculés, en fonction des entrées fournies et la réponse attendue en sortie. Ensuite, le réseau va modifier ses poids jusqu'à ce que le résultat soit le plus proche possible de la sortie désirée, correspondant à une entrée donnée. Autrement dit, c'est déterminer le vecteur des poids des neurones capables de prédire le même vecteur de sortie à partir du même vecteur d'entrée.

Cet apprentissage n'est possible que si les solutions sont connues pour les exemples de la base d'apprentissage.

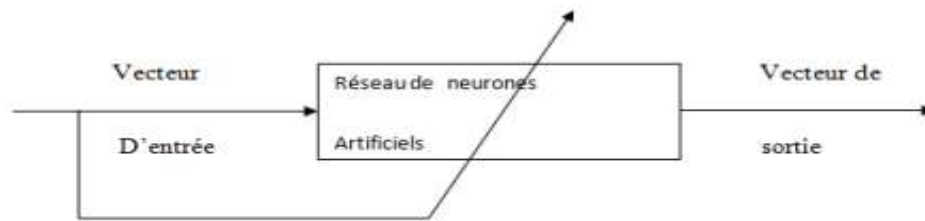


**Figure I-6 :** Forme canonique de l'apprentissage supervisé. [2]

### I.6.2.2.Apprentissage non-supervisé :

L'apprentissage est dit non supervisé lorsque seuls les vecteurs d'entrées sont disponibles, comme représenté à la figure I.7. Dans ce cas, les exemples présentés à l'entrée provoquent une auto-adaptation du réseau afin de produire des valeurs de sortie qui soient proches en réponse pour des valeurs d'entrées similaires.

L'apprentissage non supervisé est surtout utilisé pour le traitement du signal, l'analyse fonctionnelle, et plus souvent pour la classification. L'avantage de ce type d'apprentissage réside dans sa grande capacité d'adaptation reconnue comme auto-organisation (self-organizing).



**Figure I-7** : Forme canonique de l'apprentissage non-supervisé. [2]

### I.6.3. Algorithmes d'apprentissage du perceptron multicouche :

Il existe plusieurs algorithmes d'apprentissage du PMC (Perceptron Multicouche), parmi les plus utilisés [6] :

#### I.6.3.1. L'algorithme de la rétro propagation du gradient :

L'algorithme de rétro-propagation de gradient est un algorithme itératif qui vise à minimiser le critère d'erreur quadratique entre la sortie obtenue par un réseau multicouche et la sortie souhaitée. Cette minimisation est obtenue en configurant correctement les poids, et l'erreur est la différence entre la valeur attendue du neurone de sortie et la valeur qu'il calcule par propagation. En fait, l'algorithme requiert une fonction continue, non linéaire et différentiable comme fonction de transfert des neurones.

Mathématiquement, la méthode est basée sur l'algorithme de descente de gradient et utilise les règles de dérivation de fonctions différentiables. Dans cette méthode, les erreurs qui se produisent à la sortie du réseau sont propagées vers les couches cachées, d'où le nom de rétro-propagation. [6] [7] [2].

Le but de la méthode de rétro-propagation est d'ajuster les paramètres  $w_{ij}$  pour minimiser la fonction de coût.

#### ❖ Expression de la fonction de coût :

Afin de quantifier et mesurer la pertinence et du coup le taux d'erreur du réseau de neurones, nous avons désormais besoin de ce que l'on appelle une fonction de coût ou de perte ("loss or cost function" en anglais). La fonction la plus couramment utilisée, et dont nous sommes servis lors de cette étude est la fonction dite fonction d'erreur quadratique moyenne (MSE pour Mean Square Error) dont la définition est donnée comme suit :

Soit une base d'apprentissage constituée de  $N$  exemples. Pour chaque exemple  $k$  ( $k \leq N$ ), l'erreur  $e(k)$  est calculée comme étant la différence entre la cible  $t(k)$  et la valeur de sortie de réseau  $a(k)$  (Equation I-1).

$$e(k) = t(k) - a(k) \quad \text{I.1}$$

Pour tout l'ensemble  $N$  de l'apprentissage la fonction erreur MSE est donnée par l'Equation :

$$MSE = E_m = \frac{1}{N} \sum_{k=1}^N e(k)^2 = \frac{1}{N} \sum_{k=1}^N (t(k) - a(k))^2 \tag{I.2}$$

Avec :

$E_m$  : représente l'erreur quadratique commise au niveau de la couche de sortie du réseau ;

Plus la valeur de cette fonction de coût est petite, plus le modèle reproduit fidèlement les observations utilisées pour l'apprentissage. Le but de l'algorithme d'apprentissage consiste donc à ajuster les poids de réseau afin de réduire cette erreur quadratique à son minimum.

L'algorithme de la rétro-propagation du gradient de l'erreur se résume aux étapes suivantes :

- 1- Initialisation des poids de connexions à des valeurs aléatoires de faible grandeur ;
- 2- Présentation d'un couple (entrée, sortie désirée) de la base d'apprentissage ;
- 3- Présentation de la forme d'entrée sur la couche d'entrée du réseau ;
- 4- Calcul par propagation de la sortie du réseau avec l'expression suivante :

$$a(k) = \sum_{i=1}^n \sum_{j=1}^n w_{ij} y_j(k-1) \tag{I.3}$$

- 5- Calcul des différents signaux d'erreur des différentes couches :

- Calcul de l'erreur de la sortie :

$$\frac{\partial e_p(k)}{\partial w_i(k)} = -(t(k) - a(k)) \tag{I.4}$$

- Calcul de l'erreur dans les couches de sortie :

$$\frac{\partial E_m(w)}{\partial w_i(k)} = \sum_{p=1}^N \frac{\partial e_p(k)}{\partial y(k+1)} \frac{\partial y(k+1)}{\partial w_i(k)} \tag{I.5}$$

- 6- Mise à jour des matrices de connexions ;
- 7- Tant que l'erreur est trop importante, retourner à l'étape 2. Sinon aller à l'étape (8).
- 8- Fin.

### II.6.3.2.L'algorithme de Levenberg-Marquardt :

L'algorithme de Levenberg-Marquardt (LM) C'est un algorithme itératif de minimisation, permet d'obtenir une solution numérique au problème de minimisation d'une fonction, souvent non linéaire et dépendant de plusieurs variables. Il appartient à la classe des méthodes quasi-Newtoniennes, il obéit à la formule suivante de mise à jour des paramètres :

$$w_{k+1} = w_k - [H(w_k) + u_{k+1}I]^{-1}\nabla Ek(w_k)$$

Où :  $H(w_k)$ : La matrice Hessienne d'une fonction de coût appelé  $E$ ;  $I$  : La matrice identité,  $u_{k+1}$ : Un scalaire appelé pas,  $\nabla Ek(w_k)$  le gradient de la fonction de coût

Le calcul de l'inverse de la matrice  $[H(w_k) + u_{k+1}I]^{-1}$  peut s'effectuer par des méthodes d'inversion directe. L'algorithme de Levenberg-Marquardt est le plus rapide et assure la meilleure convergence vers un minimum de l'erreur quadratique, pour les problèmes d'approximation des fonctions où le nombre des poids du réseau est inférieur à cents. Quand le nombre de poids augmente l'efficacité de l'algorithme LM diminue.

## II.7.Réseaux de neurones dans le contrôle des systèmes :

En raison de leurs capacités d'approximation générales, les réseaux de neurones sont bien adaptés au contrôle des systèmes non linéaires. En effet, dans ce cas la fonction de contrôle est une fonction non linéaire et le but est d'approximer cette fonction par RNA (un réseau de neurones artificiels). Cette approximation est obtenue par l'apprentissage des poids du réseau, qui peut se faire hors ligne ou en ligne :

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

Plusieurs algorithmes de contrôle ont émergé basés sur ces deux structures de réseaux de neurones artificiels, qui peuvent être principalement divisés en :

- Contrôle inverse (basé sur l'erreur de contrôle),
- Contrôle basé sur l'erreur de sortie,
- Contrôle adaptatif et contrôle basé sur la critique adaptative.

### II.7.1.Stabilisation du système non linéaire par un réseau de neurones :

La détermination et le choix du réseau optimal pour un processus donné sont des problèmes ouverts, malgré l'existence de quelques travaux, qui permettent pour une vaste classe de réseaux, de déterminer l'architecture optimale. Avant de concevoir un réseau de neurones artificiels, il faut tout d'abord [11] :

→ Fixer le nombre de couches cachées : Mis à part les couches d'entrée et de sortie, l'analyste doit décider du nombre de couches intermédiaires ou cachées. Sans couche cachée, le réseau n'offre que de faibles possibilités d'adaptation ; avec une couche cachée, il est capable, avec un nombre suffisant de neurones, d'approximer toute fonction continue.

→ Déterminer le nombre de neurones par couches cachées Actuellement, il n'existe pas une loi qui nous dicte exactement le nombre de neurones nécessaires au niveau des couches cachées. Donc on ne sait pas comment construire le réseau, ni combien de neurones sont dans la couche cachée, ni combien de liens synaptiques. En effet, si le réseau possède un très grand nombre de poids et de neurones, le réseau est trop souple et si ce nombre est trop petit, le réseau est trop rigide et présente des mauvaises performances.

→ Test d'arrêt : La détermination du critère d'arrêt est cruciale dans la mesure où la convergence peut passer par des minima locaux. En effet, le test d'arrêt est la mesure des performances du réseau pour savoir si la convergence du réseau est atteinte. D'une façon générale, on cherche à arrêter l'algorithme si l'erreur  $E$  est minimale c'est-à-dire si le gradient de l'erreur est proche de zéro.

→ Taux d'apprentissage : Ce paramètre détermine la vitesse de convergence. Si la valeur de démarrage de  $\eta$  est grande, alors on aura un apprentissage très rapide mais au prix de la création des oscillations dans l'erreur totale qui empêcheront l'algorithme de converger vers un minimum désiré. Le réseau devient instable. Dans la plupart des cas si la fonction d'erreur possède plusieurs minimums locaux, le réseau subira un blocage dans l'un d'eux. Toutes ces conditions nous obligent à commencer l'apprentissage avec une petite valeur de  $\eta$ , si on veut attendre un minimum global même si l'apprentissage est long.

→ Seuil de tolérance : Ce paramètre critique détermine la précision de la réponse du réseau. Théoriquement, l'algorithme doit se terminer dès que le minimum de l'erreur commise par le réseau sera atteint, correspondant à un gradient nul, ce qui n'est jamais rencontré en pratique. C'est pourquoi on fixe à priori ce seuil afin d'arrêter l'apprentissage.

## II.8.Applications d'aujourd'hui :

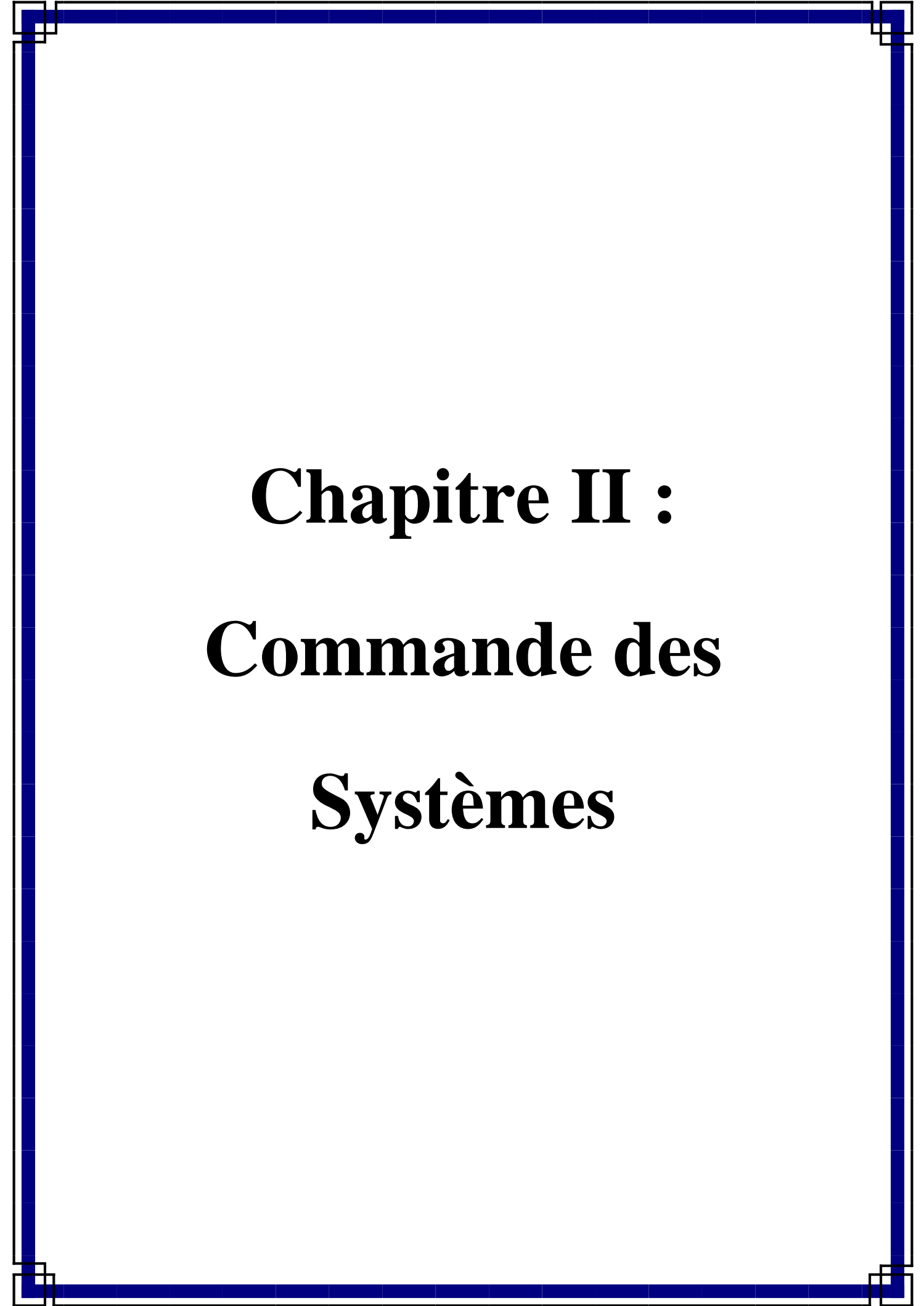
Les réseaux de neurones artificiels ont de nombreuses applications dans des secteurs très variés [3] :

- Traitement d'images : reconnaissance de caractères et de signatures, compression d'images, reconnaissance de forme, cryptage, classification, etc.
- Traitement du signal : filtrage, classification, identification de source, traitement de la parole...etc.
- Contrôle : commande de processus, diagnostic, contrôle qualité, asservissement des robots, systèmes de guidage automatique des automobiles et des avions...etc.
- Défense : guidage des missiles, suivi de cible, reconnaissance du visage, radar, sonar, lidar, compression de données, suppression du bruit...etc.
- Optimisation : planification, allocation de ressource, gestion et finances, etc.

- Simulation : simulation du vol, simulation de boîte noire, prévision météorologique ...etc.

## **II.8. Conclusion :**

Afin d'utiliser les réseaux de neurones comme modèles pour représenter le comportement des systèmes non linéaires, nous menons une étude détaillée des réseaux de neurones artificiels. Nous avons introduit les éléments de base des réseaux de neurones, les neurones formels. Ces derniers ne sont que des modèles de neurones biologiques. Nous avons également conclu que la manière dont les neurones sont connectés permet d'obtenir plusieurs types d'architectures.



# **Chapitre II :**

# **Commande des**

# **Systemes**



# CHAPITRE II : Commande des systèmes

## II.1.Introduction :

Un système non linéaire est un ensemble d'équations non linéaires (telles que différentielles) qui décrivent l'évolution temporelle des variables constitutives du système sous le contrôle d'un nombre fini de variables indépendantes appelées entrées ou variables de commande, ou simplement commandes qui peuvent être choisies librement pour atteindre des objectifs spécifiques.

Les points d'entrée peuvent soit être choisis avec une case ouverte, ce qui signifie qu'ils ne dépendent que du temps, soit avec une case fermée, ce qui signifie qu'ils dépendent des variables mesurées, appelées observations, qui gardent une trace de l'état du système à un moment donné.

Nous allons dans ce deuxième chapitre, donner quelque technique de commande des systèmes.

## II.2.La Commande PID :

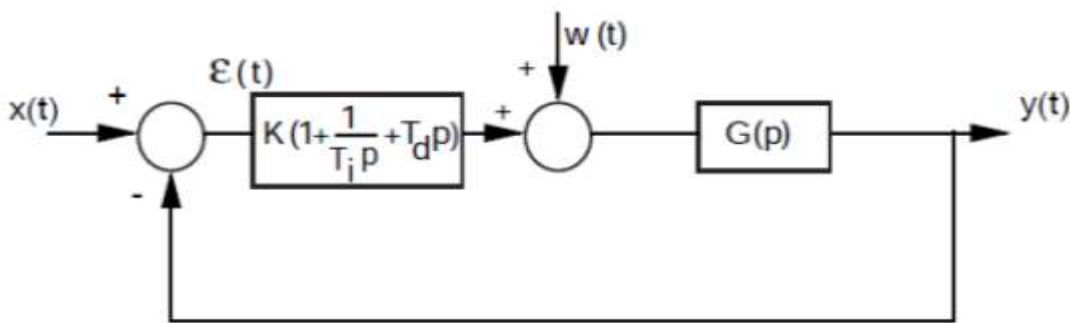
Les contrôleurs PID répondent à plus de 90% des besoins industriels, par exemple, des milliers de contrôleurs sont installés dans les usines pétrolières. Parmi les innombrables méthodes de réglage PID, cette section présente les méthodes les plus utilisées dans le contrôle de processus industriels. Aucune méthode de réglage n'existe et il est important de comprendre que c'est la connaissance du comportement du processus qui conduit à l'application de la méthode pour obtenir des paramètres PID réglés de manière satisfaisante. [12]

Le contrôleur peut être jugé selon les critères suivants :

- Le contrôleur doit être capable de maintenir la variable contrôlée à sa valeur de consigne.
- Le système en boucle fermée doit être asymptotiquement stable et présenter des performances satisfaisantes sur une large gamme de fréquences.
- L'effet des interférences doit être minime.
- La réponse aux changements de point de consigne doit être rapide et douce.
- Des actions de contrôle excessives doivent être évitées (la variable de contrôle  $u(t)$  ne doit pas être trop grande).
- Le système de contrôle doit être robuste : il doit être insensible aux variations du procédé et aux erreurs du modèle de procédé.

• **Correcteur proportionnel - intégral – dérivée :**

Les termes proportionnels et intégraux peuvent provoquer des dépassements de consigne et des oscillations. Cela signifie que pour les moteurs, par exemple, avec une polarité inversée, c'est loin d'être idéal. Pour limiter ce comportement indésirable, nous introduisons un troisième élément, le terme dérivé. Son action dépendra du signe et du taux de variation de l'erreur, et est à l'opposé de l'action proportionnelle. Au fur et à mesure que l'erreur diminue, elle devient dominante autour de la valeur souhaitée, le terme proportionnel devient moins efficace et l'intégrale évolue peu : elle ralentit alors le système, limite les dépassements et réduit le temps d'établissement. :



**Figure II-1 :** Commande PID. [8]

Le correcteur PID a pour but de combiner les effets positifs des trois corrections. La détermination des coefficients  $Kp$ ,  $Ki$ ,  $Kd$  du correcteur PID peut améliorer à la fois la précision ( $Kp$  et  $Kd$ ), la stabilité ( $Kd$ ) et la vitesse ( $Kp$ ,  $Kd$ ).

Le réglage PID est généralement assez compliqué, mais les méthodes de réglage pratiques donnent de bons résultats.

Il est régi par la relation suivante :

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \tag{II.1}$$

Sa fonction de transfert est donnée par :

$$C(p) = K_p + \frac{K_i}{p} + K_d p = K_p \left( 1 + T_d p + \frac{1}{T_i p} \right) \tag{II.2}$$

En général, on a les qualités suivantes du correcteur PID :

- L'action proportionnelle est souvent utilisée pour la vitesse du système fermé.
- L'action intégrale est utilisée pour améliorer l'erreur en régime permanent.
- L'action différentielle améliore la stabilité.

### II.3. Commande par retour d'état :

#### II.3.1. Structure de la loi de commande [8] :

Les équations du système en boucle fermée sont :

$$\begin{cases} \dot{x} = Ax(t) + Bu(t) \\ y(t) = Cx(t) + De(t) \\ u(t) = e(t) - Kx(t) \end{cases} \quad \text{II.3}$$

L'équation d'état du système en boucle fermée s'écrit :

$$\dot{x} = [Ax(t) + B[e(t) - Kx(t)]] = (A - BK)x(t) + Be(t) \quad \text{II.4}$$

Par conséquent, la matrice d'état du système en boucle fermée vaut  $(A-BK)$ .

La dynamique du système en boucle fermée est donc fixée par les valeurs propres de la matrice  $(A-BK)$  ; ces valeurs propres sont les racines de l'équation caractéristique :

$$\text{Det}(SI - (A - BK)) = 0 \quad \text{II.5}$$

#### II.3.2. Contrôlabilité :

Le système décrit par l'équation est dit contrôlable à  $t=t_0$  s'il est possible de construire un signal de commande sans contrainte qui transférera l'état initial à n'importe quel état final dans un intervalle fini de temps  $t_0 \leq t \leq t_1$

Si chaque état est contrôlable, alors le système est dit complètement contrôlable. La condition nécessaire et suffisante pour que le système soit complètement contrôlable, est que le rang de la matrice  $(n \times n)$  :

$$Cc = [Bc, AcBc \dots Ac^{n-1}Bc] \quad \text{II.6}$$

#### II.3.3. Placement de pôles par retour d'état :

Nous supposons que toutes les variables d'état sont mesurables et sont disponibles pour la rétroaction. Si le système considéré est complètement contrôlable, alors les pôles du système en boucle fermée peuvent être placés à n'importe quel endroit désiré au moyen d'un retour d'état à travers une matrice de gain approprié.

II.3.3.1. Principe :

Le principe est de déterminer une commande telle que les pôles du système en boucle fermée soient convenablement placés dans le plan complexe et satisfassent des spécifications d’amortissement, et de rapidité [8]. Les pôles de la fonction de transfert étant les valeurs propres de la matrice d’état.

Par conséquent, l’objectif est d’obtenir un contrôle qui modifie de manière appropriée la matrice d’état du système. C’est-à-dire imposer un comportement spécifié ou des propriétés souhaitées au système en appliquant des lois de commande qui placent les pôles du système en boucle fermée dans des positions permettant d’obtenir ces propriétés. La détermination des pôles en boucle fermée requis est basée sur des critères de réponse transitoire et/ou de réponse en fréquence tels que la vitesse, l’amortissement ou la bande passante, et les conditions de régime permanent.

En d’autres termes, il s’agit de trouver une loi de commande linéaire telle que les pôles d’un système de commande en boucle fermée avec cette loi coïncident exactement avec les racines du polynôme :

$$P(s) = s^n + p_n s^{n-1} + \dots + p_2 s + p_1 \tag{II.7}$$

La loi de commande qui satisfait l’objectif précédent est alors appelée : « *commande à placement de pôles* ».

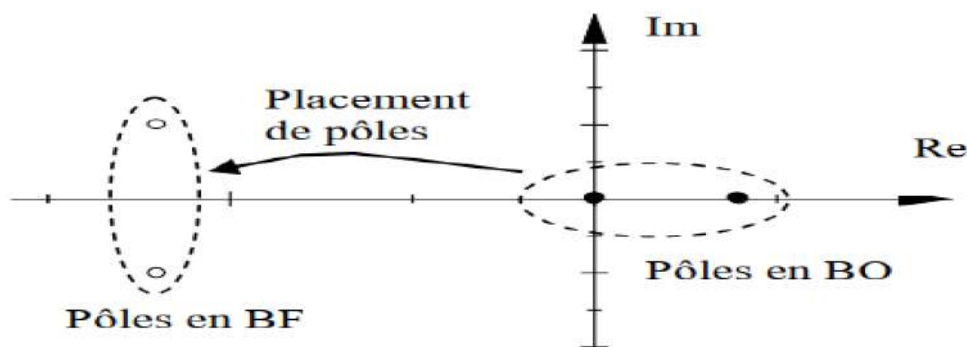


Figure II-2 : Principe du placement du pole [8]

En d’autres termes, le principe consiste à introduire de nouveaux paramètres dans le système afin que l’on puisse contrôler les positions des pôles du système en boucle fermée. Un système de contrôle en boucle fermée d’ordre n aura une équation caractéristique d’ordre n de la forme :

$$s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0 \tag{II.8}$$

**II.3.3.2. Calcul du gain du retour d'état :**

Le but est de calculer la matrice d'ajustement K pour imposer les pôles du système d'anneaux. Ce problème revient à imposer un polynôme caractéristique du système. Soit P(s) le polynôme recherché de degré n.

Pour calculer la matrice de gain K nécessaire pour le placement de pôles, plusieurs méthodes existent dans la littérature. Dans ce qui suit nous exposerons trois d'entre elles.

**a. Substitution directe :**

Pour les systèmes d'ordre inférieur ou égal à 3, on peut substituer le gain  $K = [k_1 \ k_2 \ \dots \ k_n]$  directement dans le polynôme caractéristique  $|sI - [A - BK]|$

Si le comportement désiré est spécifié à travers la sélection de n valeurs propres désirées  $\mu_1, \mu_2, \dots, \mu_n$  alors le polynôme caractéristique désiré en boucle fermée est donné par :

$$P_{BF}(s) = (s - \mu_1) * (s - \mu_2) \dots (s - \mu_n) = s^n + \alpha_{n-1} s^{n-1} + \dots + \alpha_1 s + \alpha_n \tag{II.9}$$

Comme le système en boucle fermée doit avoir comme pôles ceux du polynôme désiré, l'égalité suivante doit être vérifiée :

$$|sI - [A - BK]| = s^n + \alpha_{n-1} s^{n-1} + \dots + \alpha_1 s + \alpha_n \tag{II.10}$$

Ce qui donne comme solution les éléments du vecteur gain  $k_1, k_2, \dots, k_n$ .

**b. Forme contrôlable :**

Si le système est contrôlable (qui est la condition nécessaire et suffisante pour le placement des pôles), on peut définir la transformation d'état  $x = Tz$ , avec la matrice de transformation :

$$W = \begin{bmatrix} \alpha_{n-1} & \alpha_{n-1} & \alpha_1 & 1 \\ \alpha_{n-2} & \dots & \alpha_1 & 1 & 0 \\ & \dots & \dots & & \\ \alpha_1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$|sI - A| = s^n + \alpha_{n-1} s^{n-1} + \dots + \alpha_1 s + \alpha_n \tag{II.11}$$

Alors le vecteur gain est donné par la formule :

$$K = [\alpha_n - a_n \ \alpha_{n-1} - a_{n-1} \ \dots \ \alpha_1 - a_1] T^{-1} \tag{II.12}$$

**c. Formule d'Ackermann :**

La formule d'Ackermann pour la détermination de la matrice de gain K est donnée par l'équation suivante :

$$K = [0 \ \dots \ 0 \ 1] C c^{-1} \varphi(A) \tag{II.13}$$

Ou :

$$\varphi(A) = A^n + \alpha_1 A^{n-1} + \dots + \alpha_{n-1} A + \alpha_n \quad \text{II.14}$$

Nous venons de voir le principe de détermination des gains d'un système en boucle fermée par placement de pôles.

Les étapes du calcul de la commande sont alors les suivantes :

1. Calcul de la matrice  $(A - BL)$ .
2. Calcul du polynôme caractéristique de  $(A - BL)$ . Il vaut  $\det (sI - (A - BL))$ .
3. Identification du polynôme caractéristique de  $(A-BL)$  avec le dénominateur de la fonction de transfert de la boucle fermée.

## II.4.Commande linéaire quadratique (LQR) :

La commande linéaire quadratique est expliquée comme suit : LQ ou LQR pour un "contrôleur quadratique linéaire", le système est linéaire et le contrôle est quadratique, le meilleur contrôle est le retour d'état de la forme :

$$U = -K \cdot x(t) \quad \text{II.15}$$

Qui a pour but de minimiser Le critère quadratique  $J$  qui s'exprime par la formule suivante :

$$J = \int_0^{\infty} (x^T Q x + u^T R u) dt \quad \text{II.16}$$

$x$  représente le vecteur d'état,  $Q$  est une matrice de poids diagonaux d'ordre  $(n \times n)$  qui garantit la pénalité du vecteur d'état pour chaque variable d'état en fonction des coefficients attribués, et  $R$  est un scalaire représentant la pénalité d'énergie d'entrée. [14]

### II.4.1.Présentation de la méthode LQR :

Pour contrôler  $U$  linéairement grâce à  $x$ , on utilise la loi de commande :  $U(t) = -Kx(t)$

Le problème est de trouver le retour d'état de régulation optimal en termes de compromis vitesse, puissance et énergie de commande. Le problème est donc de trouver la matrice de gain pour le retour d'état  $K$ . Si vous travaillez en temps fini,  $R$  changera avec le temps. Néanmoins, nous constatons que  $R$  reste essentiellement constant. Il ne change qu'au début de la servocommande, par exemple lorsqu'un avion décolle ou atterrit. Nous avons donc divisé le problème en deux parties.

- Trouvez la partie constante de  $K$  qui est utilisée pendant la majeure partie du temps d'asservissement.
- Trouver tout  $K$  correspondant à tout  $\sigma$  de la partie significative. Cependant, cela nécessiterait beaucoup de calculs car les calculs devraient être effectués dans l'ordre inverse. Pour cette raison, lorsqu'il s'agit d'asservissement non critique, nous choisissons souvent de l'ignorer.[14]

### II.4.2 Le critère du compromis :

**Vitesse de rejet de perturbations :**

$$J_x = \sum_{n=1}^{\infty} X_d^T(k) Q X_d(k) \quad \text{II.17}$$

**Énergie de commande :**

On peut évaluer l'énergie de commande par le critère :

$$J_u = \sum_{n=1}^{\infty} U^T(k) R U(k) \quad \text{II.18}$$

**Critère de compromis :**

$$J(x_0, k_0, u) = J_u + J_x = \sum_{n=1}^{\infty} (X_d^T(k) Q X_d(k) + U^T(k) R U(k)) \quad \text{II.19}$$

Où :

R et Q : Sont des matrices de pondérations symétriques définies positive.

$$u^t(k) R u(k) > 0, x^t(k) Q x(k) \geq 0 .$$

Dans le cas d'un problème LTI (linéaire à temps invariant) le gain du retour statique I est une constante, les matrices Q et R doivent être spécifiées : les performances de la commande dépendent fortement des valeurs numériques des coefficients de ces matrices.

L'opérateur doit choisir les matrices de pondération. Ce résultat est important du point de vue des applications, car il permet de mettre en œuvre une commande en boucle fermée. Fonction de l'état du système ; le gain de retour k est constant et ne dépend que des paramètres du système et des matrices de pondérations Q et R du critère d'optimisation.

### II.4.3 Choix des matrices de pondérations Q et R :

La commande LQR permet de calculer une loi de commande linéaire optimale grâce à une fonction de cout J, que LQR minimise. Q et R sont alors les matrices de pondération de x et u de la fonction de cout.

Cependant, Q et R doit être définie positive. Elles expriment les préférences de l'ingénieur pour le contrôle sur A et B. Pour commencer, on se limite à l'élaboration de matrices de pondération diagonales. Il n'existe pas de manière systématique pour les calculer car elles ne représentent que des préférences mais il existe certaines heuristiques pour se donner une valeur de départ selon la règle de Bryson. On peut ensuite affiner les matrices initiales jusqu'à un résultat satisfaisant avec la méthode de tâtonnement.[14]

$$R = \text{diag}(r_1, r_2 \dots r_m) ; Q = \text{diag}(q_1, q_2 \dots q_m)$$

II.20

$$q_i = \frac{1}{\sup(y_i)} ; i = 1, 2, \dots, n$$

$$r_i = \frac{1}{\sup(u_i)} ; i = 1, 2, \dots, m$$

#### II.4.4 Recherche du gain du retour K :

Pour trouver le gain du retour K en temps infini, on applique la formule suivante :

$$K = (R + B_d^T P B_d)^{-1} B_d^T P A_d \quad \text{II.21}$$

P est obtenu par la résolution de l'équation de Riccati :

$$P = Q + A_d^T (P - P B_d (R + B_d^T P B_d)^{-1} B_d^T P) A_d = 0 \quad \text{II.22}$$

En rapportant la commande u formulée en (II.15), dans l'équation d'état, on obtient l'équation différentielle décrivant le comportement du système en boucle fermée :

$$\dot{x}(k) = (A_d - B_d K x(k)) \quad \text{II.23}$$

### II.5. Commande par réseaux de neurones artificiels :

Les techniques de contrôle traditionnelles montrent souvent leurs faiblesses et leurs inefficacités dans la théorie du contrôle des systèmes dynamiques non linéaires, en particulier lorsque les systèmes à étudier sont fortement non linéaires et incertains. La recherche avancée a conduit au développement de nouvelles techniques de contrôle non linéaire telles que la commande adaptative. Cette dernière repose sur le principe d'estimation de paramètres et/ou de fonctions inconnues dans un modèle dynamique du système à contrôler grâce à des mécanismes d'adaptation appropriés [15].

Les réseaux de neurones artificiels (RNA) sont des approximateurs généraux non linéaires couramment utilisés dans les schémas de contrôle adaptatif pour estimer les incertitudes existantes.

#### II.5.1. Commande adaptative par modèle de référence (MRAC) :

Dans Adaptive Reference Model Control (MRAC), les spécifications sont exprimées en termes de modèles de référence. Ce dernier prend une entrée de référence  $r(k)$  et produit la réponse souhaitée  $y_m(k)$ . Le contrôleur a été formé pour minimiser les erreurs de contrôle [16] :

$$e_c(k) = [y(k) - y_m(k)].$$



Le principe de la méthode est illustré par la Fig.3. Le contrôleur est contrôlé par la boucle externe. Le simulateur neuronal est utilisé pour rétro propager l'erreur  $e_c(k)$  de la sortie du système à la sortie du contrôleur.

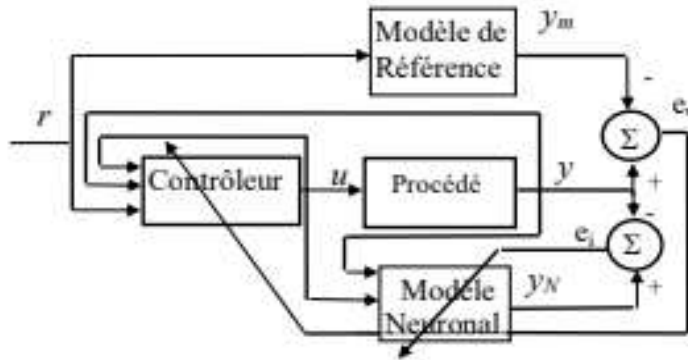


Figure II-3 : Structure MRAC indirecte /RP [16]

**II.5.2 Contrôle neuronal adaptatif :**

En général, les contrôleurs neuronaux adaptatifs sont de deux types : la commande neuronale adaptative directe et la commande neuronale adaptative indirecte. Dans la commande neuronale adaptative directe, le système neuronal est utilisé comme un contrôleur adaptatif. Cependant, dans la commande neuronale adaptative indirecte, Nous avons besoin d'identifier le système avant de le commander. [17] [18].

**II.5.2.1 La commande neuronale adaptative directe :**

La commande neuronale adaptative directe utilise un système neuronal comme contrôleur, placé en amont avec le système à commander. Comme représentée à la figure-4.

Prenons par exemple un système qu'on veut commander qui s'écrit sous la forme suivante :

$$y(k + 1) = f(y(k), \dots, y(k - n), u(k), \dots, u(k - m)) \tag{II.24}$$

Avec un modèle de référence :

$$y_m(k + 1) = f(y_m(k), r(k)) \tag{II.25}$$

La sortie du contrôleur neuronal représente une commande à appliquer à l'entrée du système, celle-ci est donnée par :

$$u(k + 1) = NN(y_m(k), y(k), u(k)) \tag{II.26}$$

Les paramètres du contrôleur neuronal sont directement ajustés en se basant sur l'erreur  $e_c$  :

$$e_c(k + 1) = y(k + 1) - y_m(k + 1) \tag{II.27}$$

Cette méthode de commande présente un inconvénient quand le système à contrôler est inconnu, puisque on ne peut pas calculer les changements des paramètres du contrôleur.

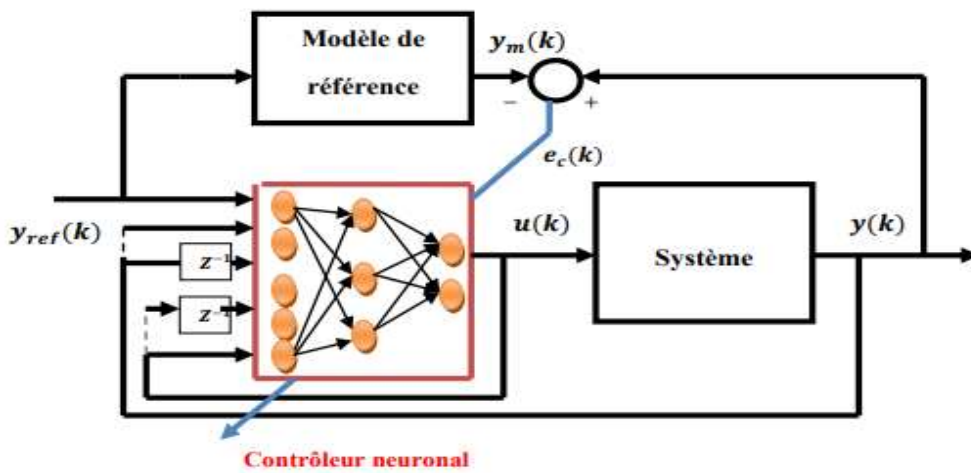


Figure II-4 : Schéma de structure de la commande neuronale adaptative directe. [17].

**II.5.2.2 La commande neuronale adaptative indirecte :**

Ce type de commande utilise une combinaison de deux réseaux de neurones. Le premier réseau est entraîné pour identifier le modèle dynamique du système. Le deuxième réseau de neurones est entraîné pour générer une commande nécessaire à appliquer au système pour que sa sortie suive celle du modèle de référence. Comme représentée sur la figure II.5.

Le signal de commande généré par le contrôleur neuronal  $u(k)$  est utilisé comme une entrée pour le modèle neuronal et le système commandé. L'erreur  $e_i(k)$  entre la sortie du système  $y(k)$  et celle du modèle  $\hat{y}(k)$  sera utilisée pour ajuster les paramètres du modèle neuronal d'une part, et d'autre part, l'erreur  $e_c(k)$  entre la sortie du modèle de référence  $y_m(k)$  et la sortie du système commandé sera utilisée pour adapter les paramètres du contrôleur neuronal. Ce qui signifie que la qualité de la poursuite dépend largement de la convergence du modèle identifié.

L'inconvénient de cette technique est la complexité de la structure de commande avec deux réseaux et le temps nécessaire aux calculs et aux ajustements des paramètres.

Si on prend un système non linéaire sous la forme :

$$y(k + 1) = f(y(k), \dots, y(k - n), u(k), \dots, u(k - m)) \tag{II.28}$$

La structure du modèle d'identification neuronal sera donnée par l'équation suivante :

$$\hat{y}(k + 1) = NN(y(k), \dots, y(k - n), u(k), \dots, u(k - m)) \tag{II.29}$$

Avec  $\hat{y}(k)$  la sortie du modèle d'identification.

L'ajustement des paramètres du modèle neuronal est basé sur l'erreur d'identification.

Les paramètres du contrôleur neuronal sont ajustés en utilisant l'erreur  $e_c(k + 1)$  :

Avec :

$$e_c(k + 1) = y(k + 1) - y_m(k + 1) \tag{II.30}$$

Nous utilisons alors le modèle neuronal identifié, quand le système à commander est inconnu, l'expression de l'erreur est :

$$e_c(k + 1) = \hat{y}(k + 1) - y_m(k + 1) \tag{II.31}$$

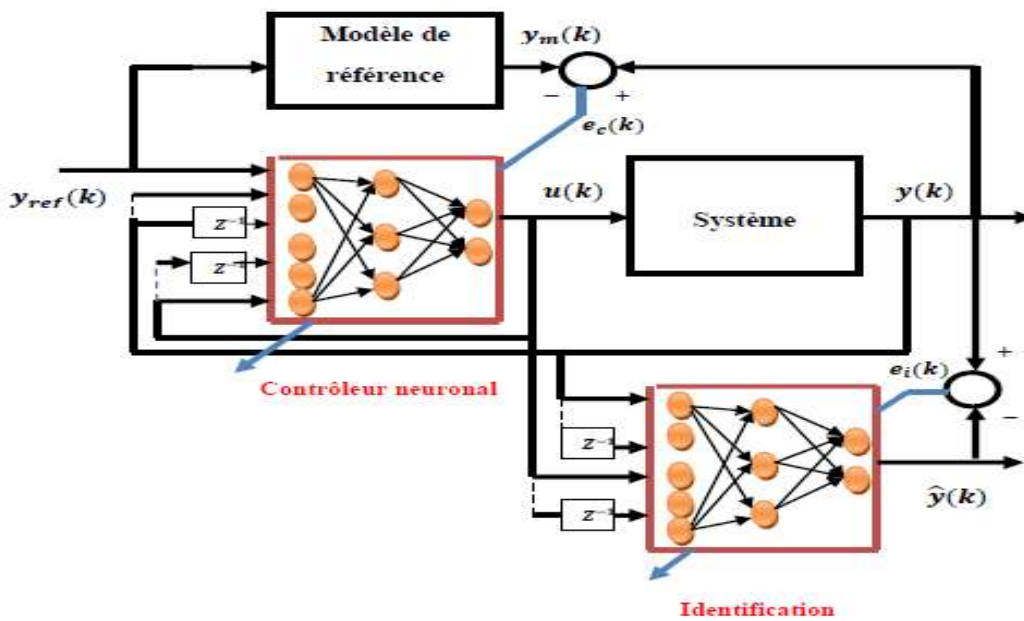


Figure II-5 : Structure de la commande neuronale adaptative Indirecte. [17]

### II.5.3 Identification neuronale :

#### II.5.3.1. Identification des systèmes :

L'identification se fait en général en deux étapes : l'étape qualitative (caractérisation) et l'étape quantitative (estimation des paramètres) [19]

II.5.3.1.1.Étape qualitative :

Elle consiste à corriger les formes des équations décrivant le processus, pour les systèmes non linéaires, NARENDRA a proposé quatre types de modèles d'identification entrée-sortie, elles sont décrites par les équations suivantes :

**Modèle I :**

$$y(k + 1) = \sum_{i=0}^{n-1} \alpha_i y(k - i) + g[u(k), u(k - 1), \dots, u(k - m + 1)] \quad \text{II.32}$$

**Modèle II :**

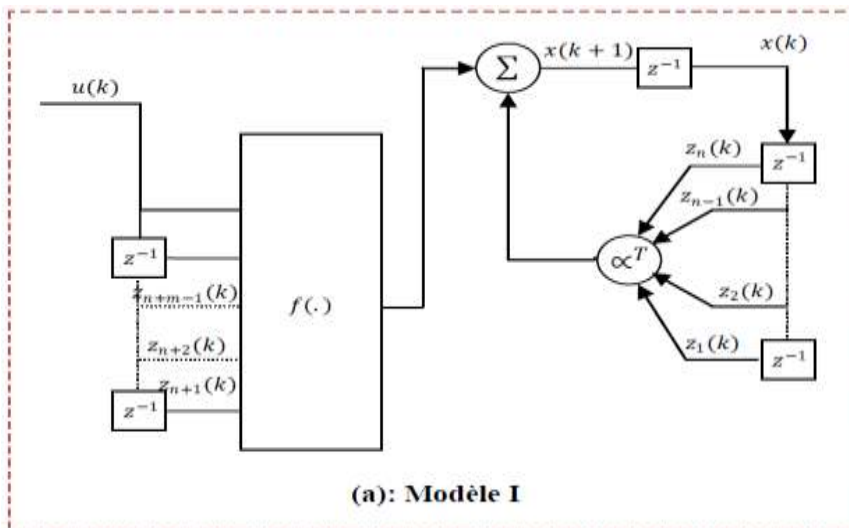
$$y(k + 1) = f[y(k), y(k - 1), \dots, y(k - n + 1)] + \sum_{i=0}^{m-1} \beta_i u(k - i) \quad \text{II.33}$$

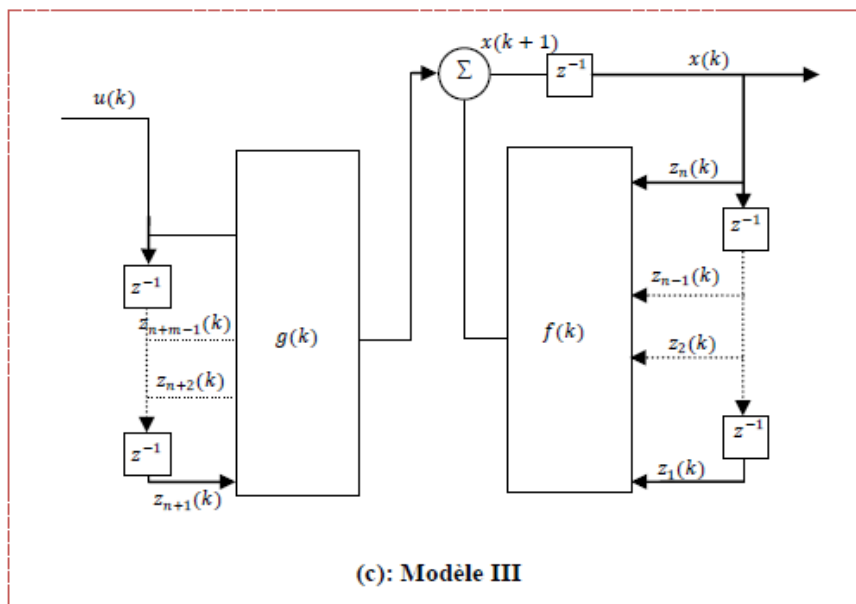
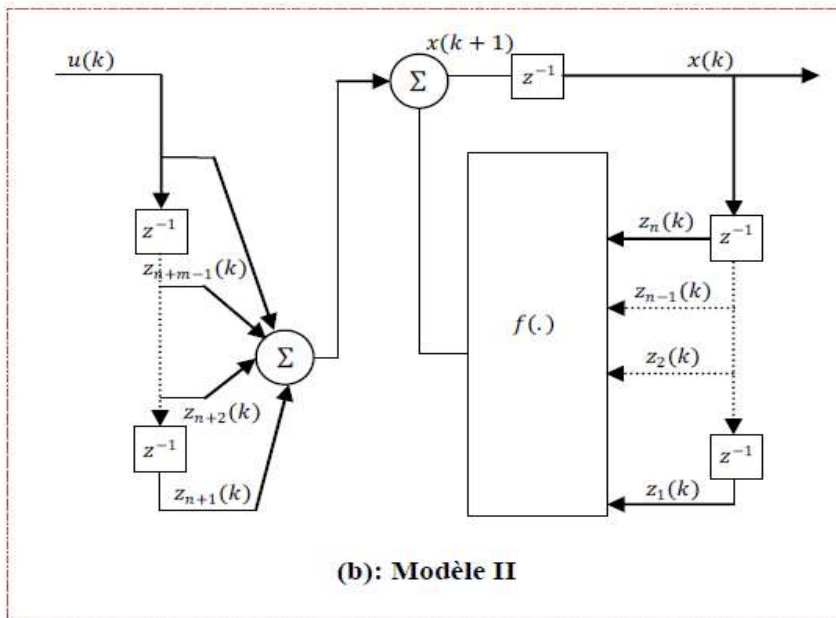
**Modèle III :**

$$y(k + 1) = f[y(k), y(k - 1), \dots, y(k - n + 1)] + g[u(k), u(k - 1), \dots, u(k - m + 1)] \quad \text{II.34}$$

**Modèle IV :**

$$y(k + 1) = f[y(k), y(k - 1), \dots, y(k - n + 1), u(k), u(k - 1), \dots, u(k - m + 1)] \quad \text{II.35}$$





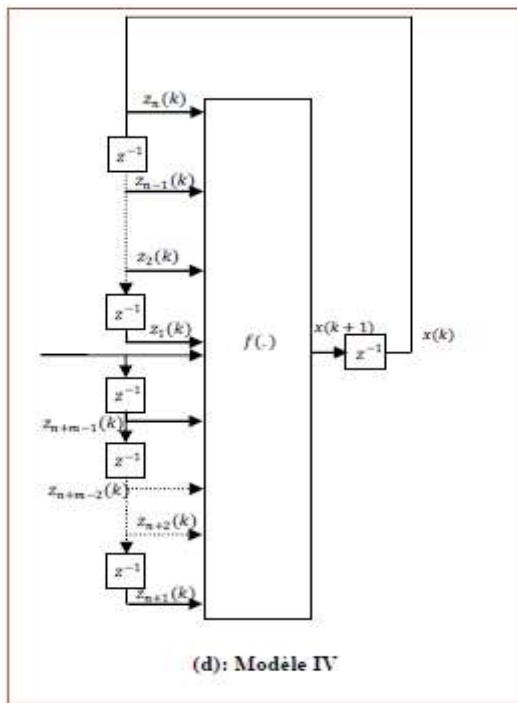


Figure II-6 : les modèles d'identification entrée-sortie

Où  $[u(k), y(k)]$  représente la paire d'entrée-sortie d'un système SISO à l'instant  $k$ , et  $m \leq n$ . Le modèle IV est la représentation de systèmes NARMA la plus générale.

De nouveaux modèles simplifiant les algorithmes de contrôle et de ces faits applicables à une large classe de systèmes non linéaires [19]. Ces deux modèles sont référés par modèle V et modèle VI. Ils sont décrits par :

**Modèle V :**

$$y(k + 1) = f[y(k), \dots, y(k - n + 1)] + \sum_{i=0}^{n-1} g_i[y(k), \dots, y(k - n + 1)]u(k - i) \quad \text{II.36}$$

**Modèle VI**

$$y(k + 1) = f[y(k), \dots, y(k - n + 1)], u(k - 1), \dots, u(k - n + 1)] + g[y(k), \dots, y(k - n + 1), u(k - 1), \dots, u(k - n + 1)]u(k) \quad \text{II.37}$$

**II.5.3.1.2.Étape quantitative :**

Elle consiste à déterminer les valeurs numériques des paramètres (poids des connexions) qui interviennent dans le modèle d'identification. On doit définir un critère qui exprime quantitativement l'écart entre le système et le modèle. Ce critère devra être minimisé [20].

**II.5.3.2 Structure d'identification :**

Le principe de l'identification neuronale consiste à placer un réseau de neurones en parallèle avec le système et à l'entraîner pour simuler le comportement dynamique de ce dernier. Il existe deux types d'identification : la modélisation directe et la modélisation inverse.

**II.5.3.2.1. Modélisation directe :**

La modélisation directe consiste à former un réseau de neurones pour simuler la dynamique directe d'un système. Le réseau est placé en parallèle avec le système. Pendant la formation, le système agit comme un enseignant, entraînant le réseau à imiter ses actions. Dans le cas des réseaux MLP, l'algorithme RP est généralement utilisé pour l'adaptation du poids.

On suppose que le système à identifier est uni varié et peut être décrit par le modèle

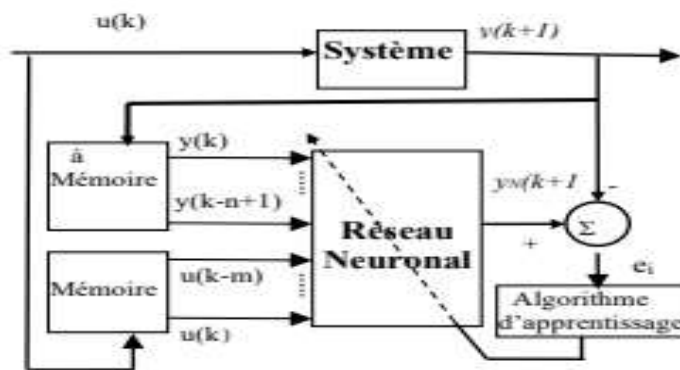
Suivant NARMA :

$$y(k + 1) = f(y(k), y(k - 2), \dots, y(k - n + 1); u(k), u(k - 1), \dots, u(k - m)) \quad \text{II.38}$$

Où  $f: R^{n+m} \rightarrow R$  est une fonction non linéaire et n est l'ordre du système. En s'inspirant des techniques classiques ; il est possible d'utiliser différentes structures d'identification.

- a. Identification Série-Parallèle :** Dans cette configuration, comme le montre la figure 7, le réseau est placé en parallèle avec le système, mais il reçoit le flux d'informations de sa sortie (c'est-à-dire en série). Le réseau de neurones reçoit un vecteur d'entrée de la forme :

$$X(k) = [y(k - 1), y(k - 2), \dots, y(k - n) ; u(k), u(k - 1), \dots, u(k - m)] \quad \text{II.39}$$



**Figure II-7 :** Structure d'identification série-parallèle. [16]

On peut voir que le réseau reçoit les valeurs passées de la sortie du système, plutôt que de sa propre sortie.

- b. Identification Parallèle :** Dans cette architecture, comme le montre la figure 8, le réseau est entièrement parallèle au système. Le réseau reçoit les valeurs passées de sa propre sortie. La forme du vecteur d'entrée :

$$X(k) = [y_n(k - 1), y_n(k - 2), \dots, y_n(k - n) ; u_n(k - 1), u_n(k - 2), u_n(k - m)] \quad \text{II.40}$$

L'avantage de cette structure est la précision et le rejet des interférences et du bruit de mesure associés à la consommation électrique réelle du système. Cependant, il n'y a aucune garantie de convergence. Il est

donc préférable d'utiliser un modèle série-parallèle. Généralement, l'identification du système commence par une identification série/parallèle. Si la sortie du modèle est suffisamment proche de la sortie du système, une structure parallèle peut être utilisée pour obtenir un modèle plus précis.

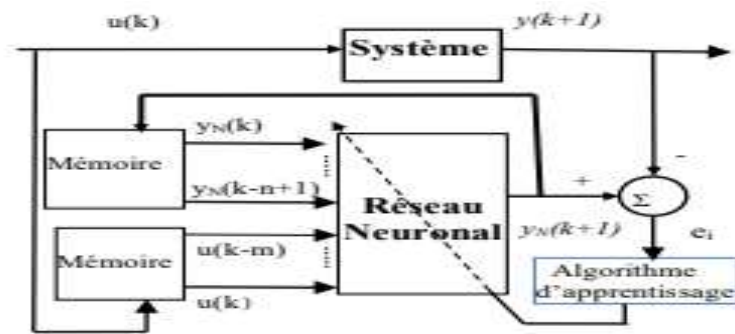


Figure II-8 : Structure d'identification parallèle.[16]

**II.5.3.2.2 Modélisation Inverse :**

La modélisation inverse forme un réseau de neurones pour émuler la dynamique inverse du système. Le but est d'utiliser le modèle obtenu comme contrôleur. La mise en cascade avec le système permet une fonction de transfert uniforme entre la trajectoire souhaitée et les performances réelles du système.

Le modèle inverse peut être exprimé par :

$$u(k) = f^{-1}(y(k + 1), y(k), \dots, y(k - n + 1); u(k - 1), u(k - m)) \tag{II.41}$$

Cette formule nécessite la connaissance de la valeur future de la sortie du système y (k+1). Cette valeur est remplacée par la valeur cible yd(k+1) supposée disponible à l'instant k. Cette approche est directement liée au contrôle de systèmes.



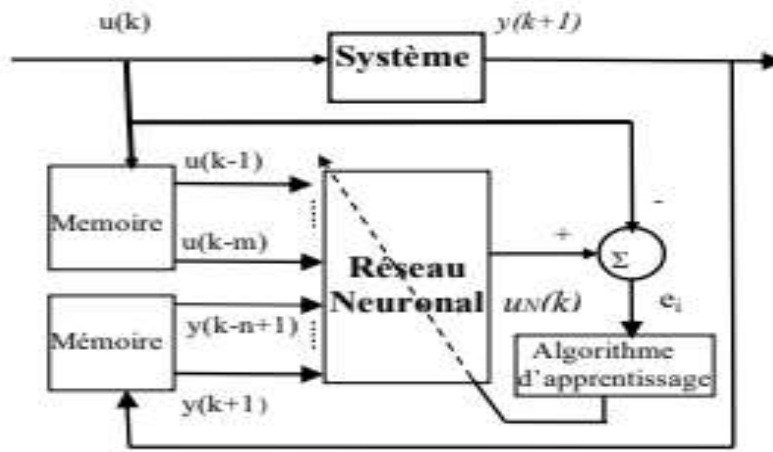


Figure II-9 : Structure de modélisation inverse (par apprentissage généralisé). [16]

**II.6. Conclusion :**

Dans ce chapitre, nous avons attaché une importance à la commande stabilisante les systèmes non linéaires. Dans un premier temps, un rappel sur trois techniques de commande classiques, la commande PID, la commande par retour d'état et la commande linéaire quadratique. Ensuite, nous nous sommes tournés vers la commande adaptative neuronale directe et indirecte, l'identification neuronale des systèmes non linéaires et sa structure.

Enfin, il est conclu que les réseaux de neurones sont souvent utilisés dans l'identification et le contrôle des systèmes. Ils ont la propriété approchée générale de tout système dynamique non linéaire.

En général, il existe deux types de contrôleurs neuronaux adaptatifs : Contrôle neuronal adaptatif direct et contrôle neuronal adaptatif indirect. Dans le contrôle neuronal adaptatif direct, le système neuronal est utilisé comme contrôleur adaptatif. Cependant, dans le contrôle neuronal adaptatif indirect, nous devons identifier le système avant de pouvoir le contrôler.

**Chapitre III :**  
**Commande neuronale**  
**Des**  
**Systemes dynamiques**

# Chapitre III : Commande neuronale des systèmes dynamiques.

## III.1. Introduction :

Dans ce chapitre nous considérons deux approches pour l'utilisation des réseaux de neurones dans la commande de systèmes dynamiques. La première c'est la commande NARMA-L2 dans laquelle on utilise la linéarisation par feedback et la deuxième est la commande par modèle de référence MRAC. Pour les deux approches une étape d'identification est primordiale. Elle consiste à ajuster, par apprentissage, un réseau de neurone pour représenter un modèle du système à commander. Ensuite, le contrôleur est conçu directement à partir du modèle neuronal pour le cas de la commande NARMA-L2. Pour le cas du MRAC une deuxième étape d'identification est utilisée pour la conception du contrôleur.

Nous appliquons ces deux commandes à des exemples de systèmes dynamiques et nous analysons les résultats obtenus.

## III.2. Commande NARMA-L2 :

Il a été montré dans [21] et ailleurs qu'un système dynamique non linéaire à une seule entrée et une seule sortie peut être localement représenté par :

$$y(k+d) = f(y(k), y(k-1), \dots, y(k-n+1), u(k), u(k-1), \dots, u(k-n+1))$$

$n, d$  sont l'ordre et le degré relatif du système respectivement. Ce modèle est appelé modèle NARMA (Non Linear AutoRegressive Moving Average).

L'identification du système par réseau de neurone consiste à utiliser les données d'entrée et de sortie du système pour entraîner un réseau de neurone multicouches pour qu'il approxime la fonction  $f(\ )$ . Le modèle identifié sera dénoté :

$$\hat{y}(k+d) = N(y(k), y(k-1), \dots, y(k-n+1), u(k), u(k-1), \dots, u(k-n+1))$$

Pour commander le système, la commande  $u(k)$  à l'instant  $k$  doit être choisie de telle sorte que  $y(k+d) = y_r(k+d)$  ou  $y_r(k+d)$  est le signal de référence. Sous certaines conditions (théorème de

fonction implicite) la fonction inverse de la fonction  $f(\ )$  existe et elle peut alors être apprise par un réseau de neurone multicouche à partir des données mesurées d'entrée et de sortie ainsi que le signal de référence :

$$u(k) = G(y(k), y(k - 1), \dots, y(k - n + 1), y^*(k + d), u(k - 1), \dots, u(k - n + 1))$$

Vu que le contrôleur est dans la boucle avec un système dynamique, on est obligé d'utiliser la back propagation dynamique pour faire l'apprentissage. Les méthodes de gradient dynamique sont très lentes et nécessitent beaucoup plus de calcul que les méthodes statiques. En pratique, des approximations sont utilisées au lieu de la back propagation dynamique. Deux modèles approximatifs du système sont introduits pour simplifier le calcul de la commande. Ce sont le modèle NARMA-L1 et le modèle NARMA-L2 :

Le modèle NARMA-L1

$$y(k + d) = f_0(y(k), y(k - 1), \dots, y(k - n + 1)) + \sum_{i=0}^{n-1} g_i(y(k), y(k - 1), \dots, y(k - n + 1))u(k - i)$$

Le modèle NARMA-L2

$$y(k + d) = f_0(y(k), y(k - 1), \dots, y(k - n + 1), u(k - 1), \dots, u(k - n + 1)) + g_0(y(k), y(k - 1), \dots, y(k - n + 1), u(k - 1), \dots, u(k - n + 1))u(k)$$

Si le système est approximable par l'un de ces deux modèles, alors le calcul de la commande se trouve beaucoup plus simplifié. Dans la suite, nous considérons que les systèmes sont approximables par un modèle NARMA-L2 de la forme :

$$y(k + d) = f_0(y(k), y(k - 1), \dots, y(k - n + 1), u(k), u(k - 1), \dots, u(k - n + 1)) + g_0(y(k), y(k - 1), \dots, y(k - n + 1), u(k), u(k - 1), \dots, u(k - n + 1))u(k + 1)$$

Durant l'apprentissage, on fait entrainer deux réseaux de neurones pour apprendre les fonctions  $f_0(\ )$  et  $g_0(\ )$ . La commande est alors celle qui garantit que  $y(k + d) = y_r(k + d)$  et elle sera donnée par :

$$u(k + 1) = \frac{y_r(k + d) - f_0(y(k), y(k - 1), \dots, y(k - n + 1), u(k), u(k - 1), \dots, u(k - n + 1))}{g_0(y(k), y(k - 1), \dots, y(k - n + 1), u(k), u(k - 1), \dots, u(k - n + 1))}$$

Le modèle et le contrôleur NARMA-L2 sont représentés par les figures qui suivent

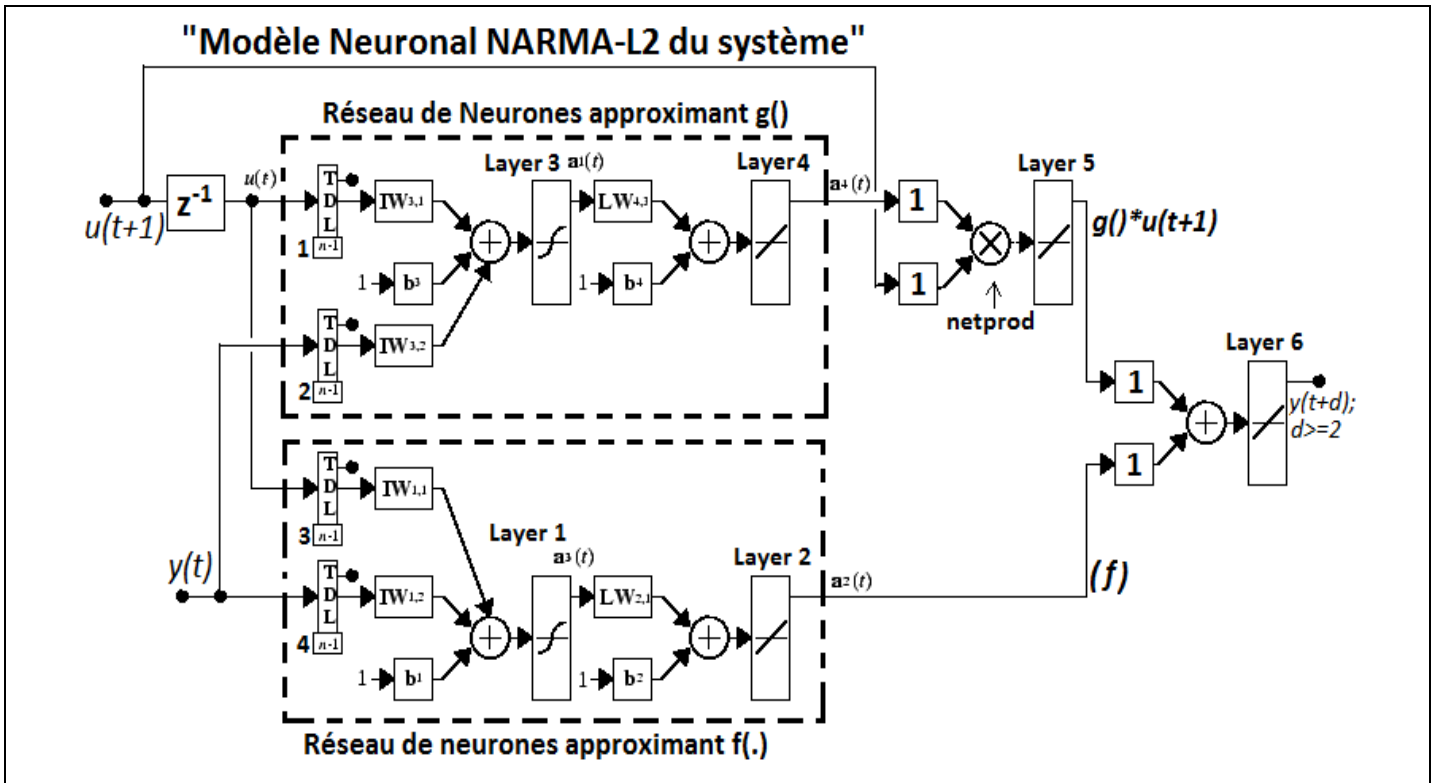


Figure (III-1) : Modèle neuronal de type NARMA-L2

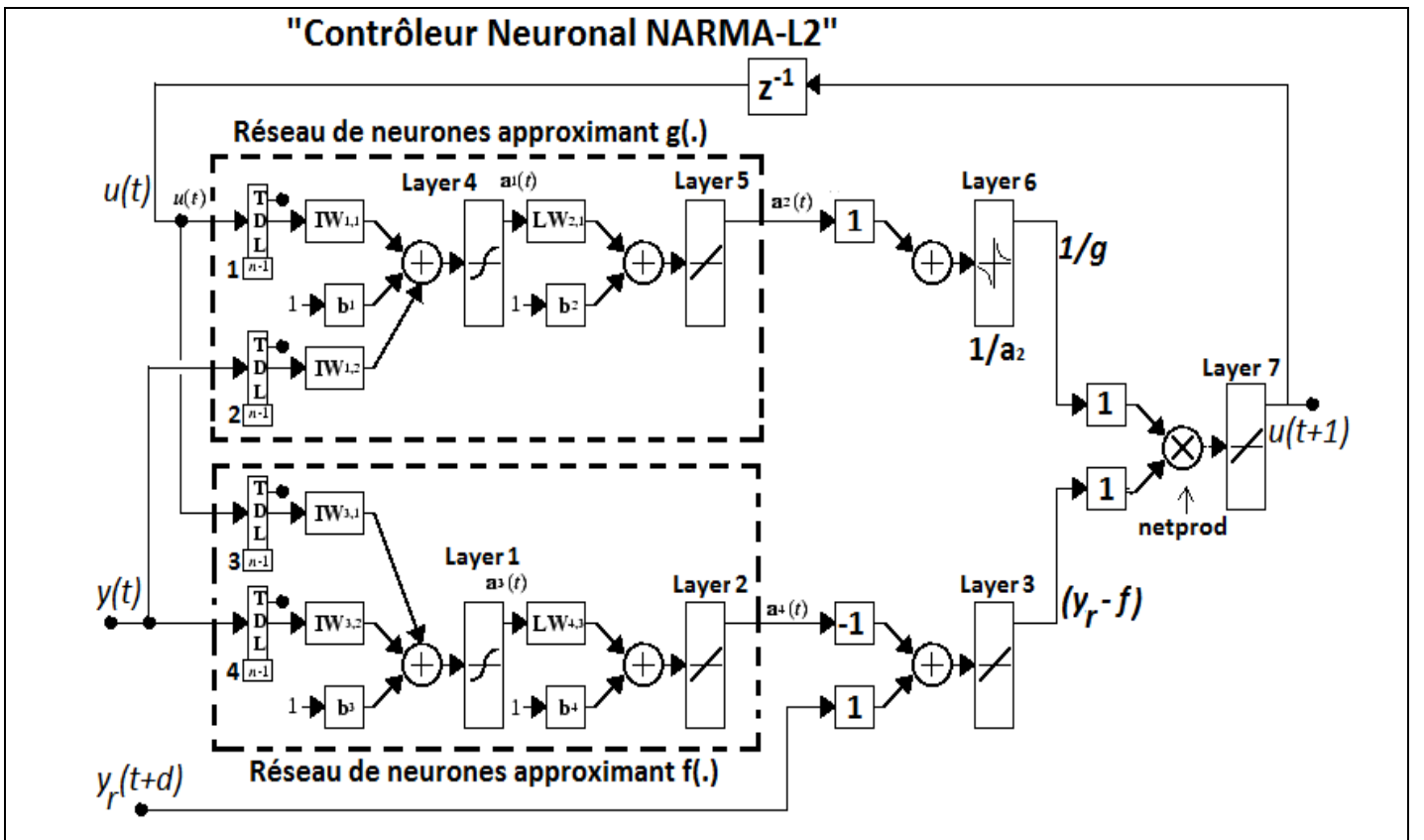


Figure (III-2) : Contrôleur neuronal de type NARMA-L2.

Au début nous allons appliquer le contrôleur neuronal de type NARMA-L2 à un exemple simple où le système est linéaire et est aussi dans la forme canonique pour l'application de ce type de contrôleur. Ceci va nous permettre de voir comment le contrôleur se comporte dans le cas idéal où la linéarité est prise en considération et aussi dans le cas où la linéarité n'est pas prise en considération. Le système de cet exemple est décrit par l'équation aux différences (récurrente) suivante :

$$y(n + 3) = \underbrace{0.5y(k) + 0.1y(k - 1) + 0.1y(k - 2) - u(k - 1)}_{f(\cdot)} + \underbrace{1}_{g(\cdot)} u(k + 1)$$

Dans cette expression, les fonctions  $f(\cdot)$  et  $g(\cdot)$  sont directement obtenues sans avoir besoin de faire l'identification. Les réseaux de neurones qui les modélisent sont des réseaux linéaires et leurs poids synaptiques sont obtenus directement de l'équation précédente. Nous avons utilisé des réseaux de neurones de la même structure que celle utilisée pour obtenir le contrôleur NARMA-L2 (sinon le système entier peut être modélisé par un seul neurone). Les poids synaptiques des couches 1 (pour modéliser  $f(\cdot)$ ) et 3 (pour modéliser  $g(\cdot)$ ) sont :

$$IW\{1,1\} = [0 \quad -1 \quad 0]; \quad IW\{1,2\} = [0.5 \quad 0.1 \quad 0.1]; \quad b\{1\} = 0$$

$$IW\{3,1\} = [0 \quad 0 \quad 0]; \quad IW\{3,2\} = [0 \quad 0 \quad 0]; \quad b\{3\} = 1$$

Nous avons créé un réseau de neurones de type "narx" de 6 couches dans MATLAB et nous l'avons configuré sous la forme adéquate pour la construction du contrôleur NARMA-L2. Après nous avons fixé ses paramètres synaptiques comme désiré. Ce réseau de neurones est enfin utilisé pour générer le bloc Simulink nécessaire pour la simulation de la commande du système. Le schéma bloc en boucle fermée conçu sous Simulink est donné par la figure suivante :

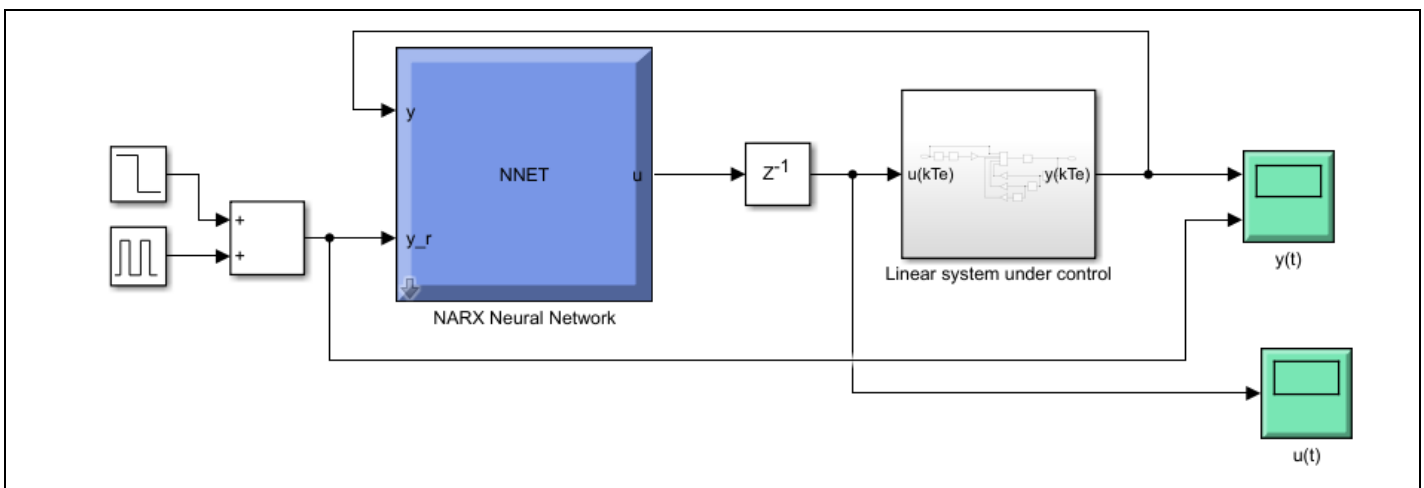


Figure (III-3) : Schéma bloc de la commande par réseau de neurones sous Simulink.

Nous avons fait la simulation en boucle fermée est nous avons obtenu les résultats des figures suivantes :

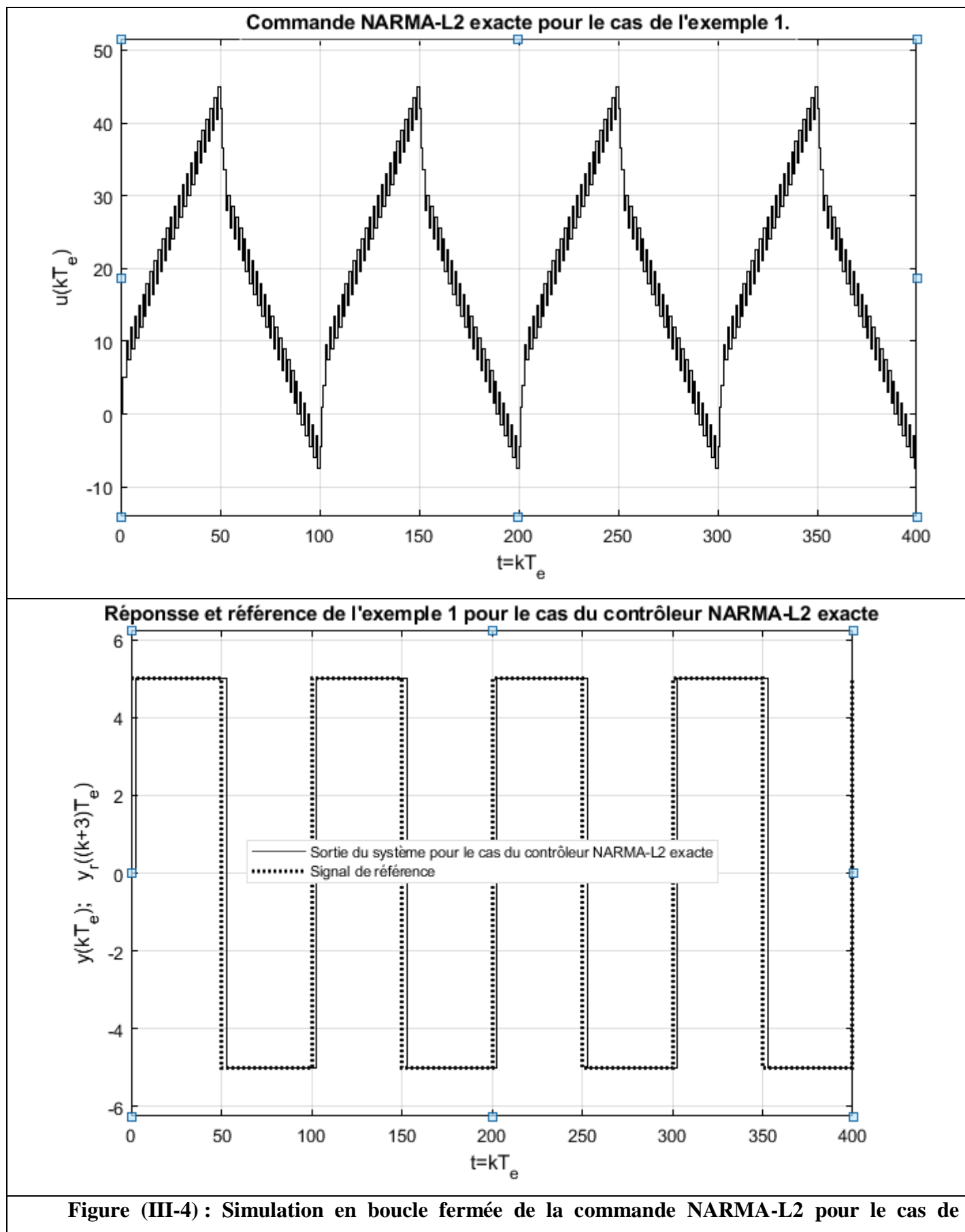
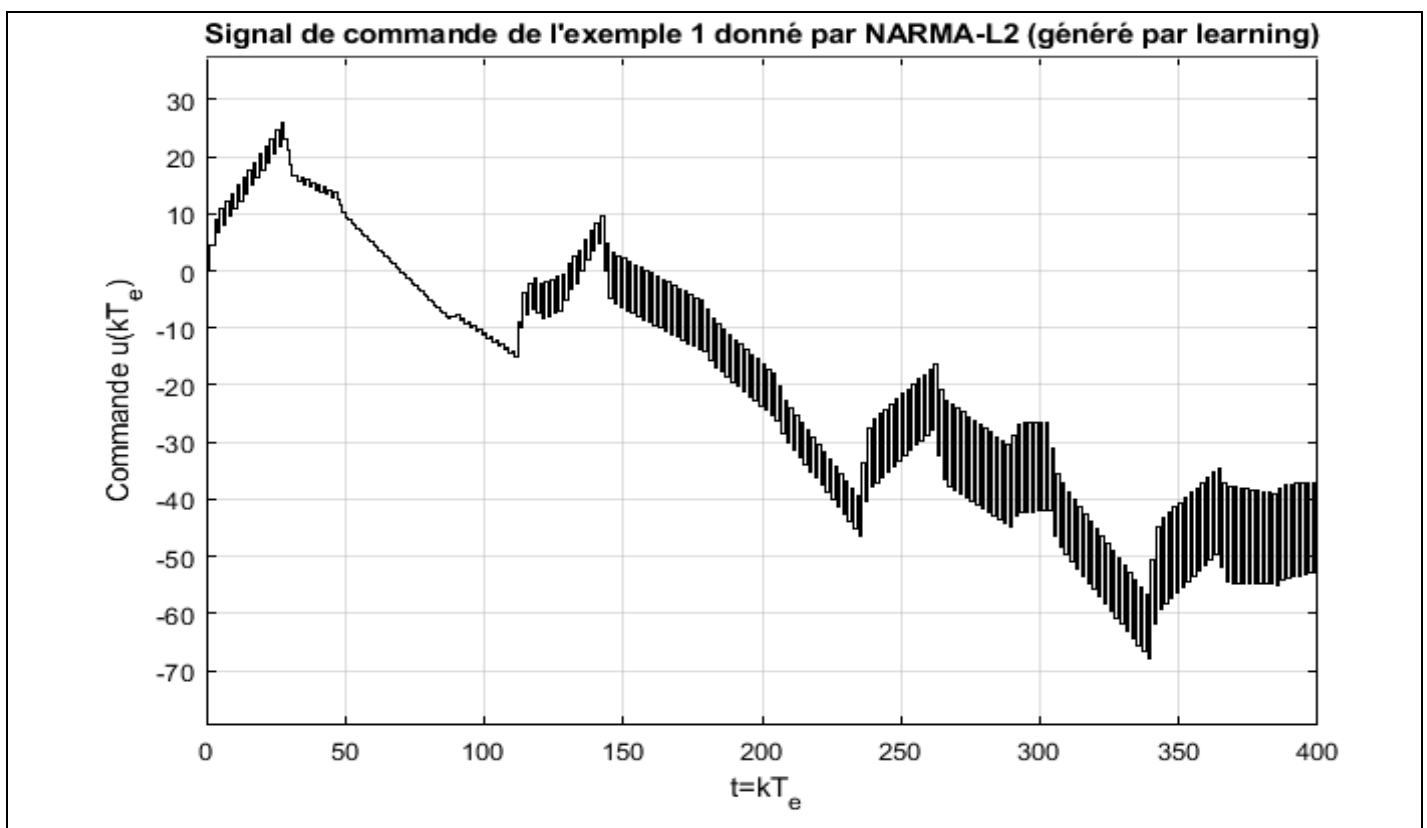


Figure (III-4) : Simulation en boucle fermée de la commande NARMA-L2 pour le cas de

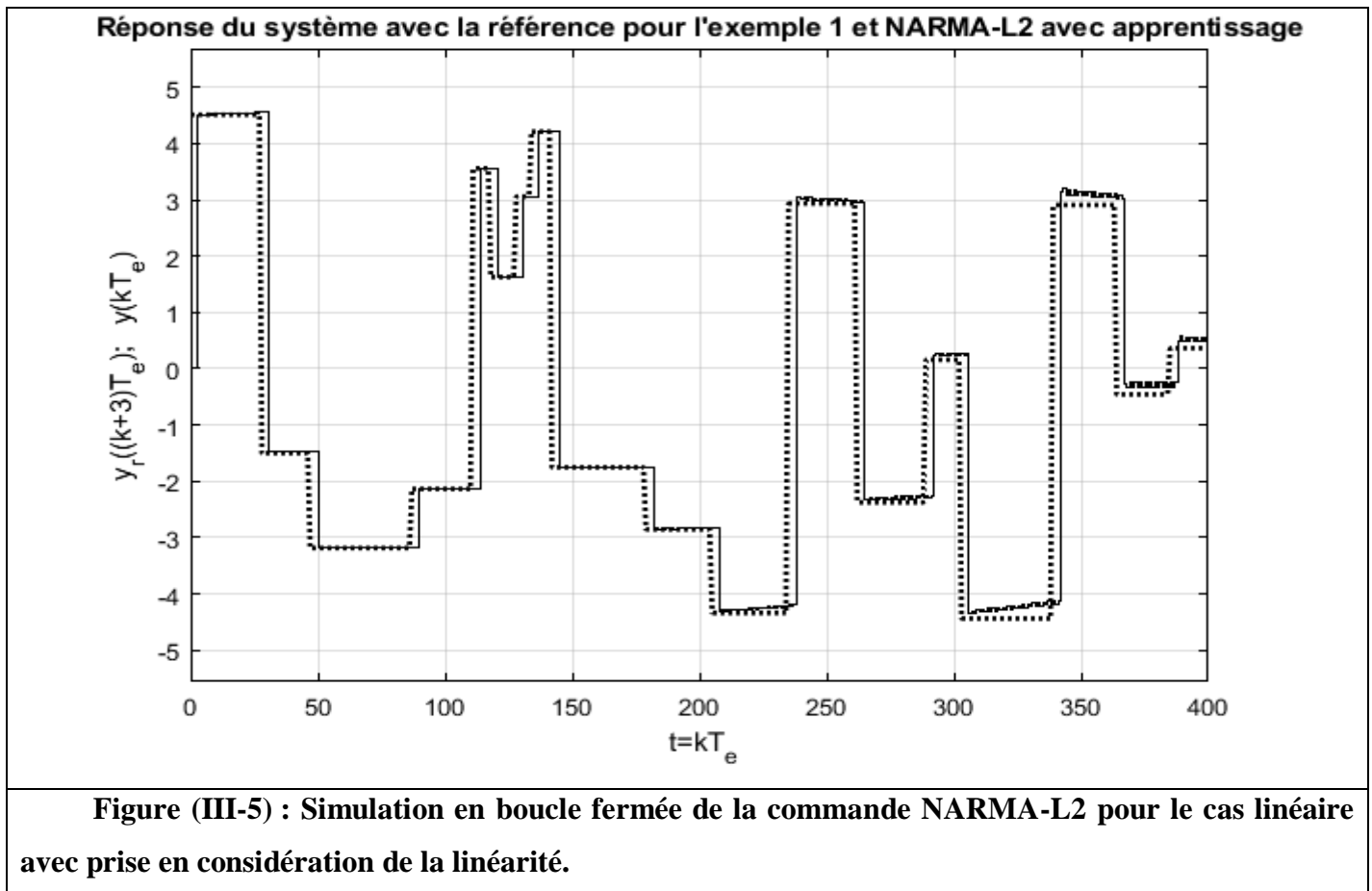
**contrôleur linéaire exacte**

Sur la figure nous avons représenté  $y(kT_e)$  et  $y_r((k+3)T_e)$  c'est ce qui explique le retard représenté sur la figure sinon la sortie  $y$  poursuit parfaitement la référence. La commande  $u(kT_e)$  est un peu oscillatoire ce qui est une caractéristique de la commande par NARMA-L2. Ceci peut être réduit en modifiant un peu la commande.

Nous avons ensuite fait la conception du contrôleur pour ce même système en passant par l'apprentissage mais en prenant la linéarité en considération. Les figures suivantes représentent les résultats obtenus.







Le signal de référence utilisé pour générer les données d'apprentissage, est une suite d'impulsion rectangulaire de durée et d'amplitude choisies de façon aléatoire dans des intervalles prédéfinis. Nous avons constaté que les poids synaptiques convergent vers des valeurs qui vérifient l'expression donnée par l'équation récurrente du système. C'est à dire :

$$IW\{1,1\} * LW\{2,1\} = [0 \quad -1 \quad 0]; \quad IW\{1,2\} * LW\{2,1\} = [0.5 \quad 0.1 \quad 0.1]; \quad b\{1\} * LW\{2,1\} + b\{2\} = 0$$

$$IW\{3,1\} * LW\{4,3\} = [0 \quad 0 \quad 0]; \quad IW\{3,2\} * LW\{4,3\} = [0 \quad 0 \quad 0];$$

Ceci est dû au fait que la linéarité est prise en considération dans le réseau de neurone. Plus l'apprentissage du modèle est bon plus le contrôleur s'approche du cas idéal. Pour montrer la généralisation de cette commande, nous avons utilisé un signal de commande qui est différent de celui utilisé pendant l'apprentissage est nous avons obtenu les résultats des figures suivantes.

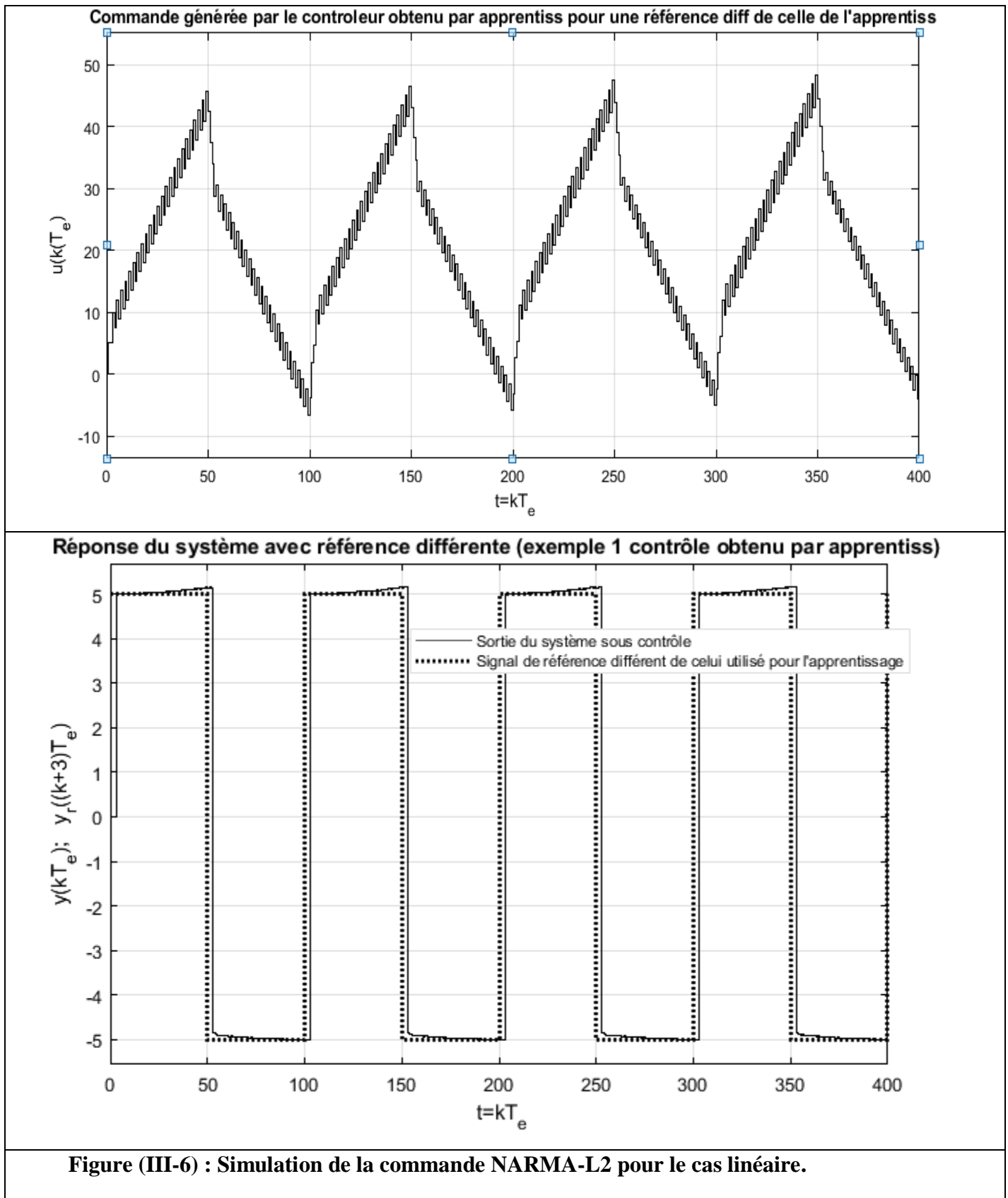


Figure (III-6) : Simulation de la commande NARMA-L2 pour le cas linéaire.

Enfin, nous avons aussi utilisé la commande NARMA-L2 pour ce même exemple sans prendre en considération la faite que le système est linéaire dans la conception du contrôleur. Au début nous avons utilisé un seul neurone avec comme fonction d'activation la fonction tangente hyperbolique "tansig" dans les

couches 1 et 3 pour identifier les fonctions  $f(\cdot)$  et  $g(\cdot)$ . Nous avons constaté que les paramètres convergent vers des valeurs qui vérifient les produits précédents avec des valeurs très petite pour  $IW\{1,1\}$  et  $IW\{1,2\}$  de l'ordre de  $(10^{-3})$  et une grande valeur de  $LW\{2,1\}$  (de l'ordre de 350) de telle sorte que l'ordre reste conservé. Ceci s'explique par le fait que la fonction "tansig" est linéaire au voisinage de 0 et ceci est l'une des causes de son très grande utilité. Ensuite nous avons utilisé des réseaux avec 10 neurones "tansig" dans les couches 1 et 3, nous donnons la réponse du système avec le contrôleur qui en résulte dans la figure (III-7).

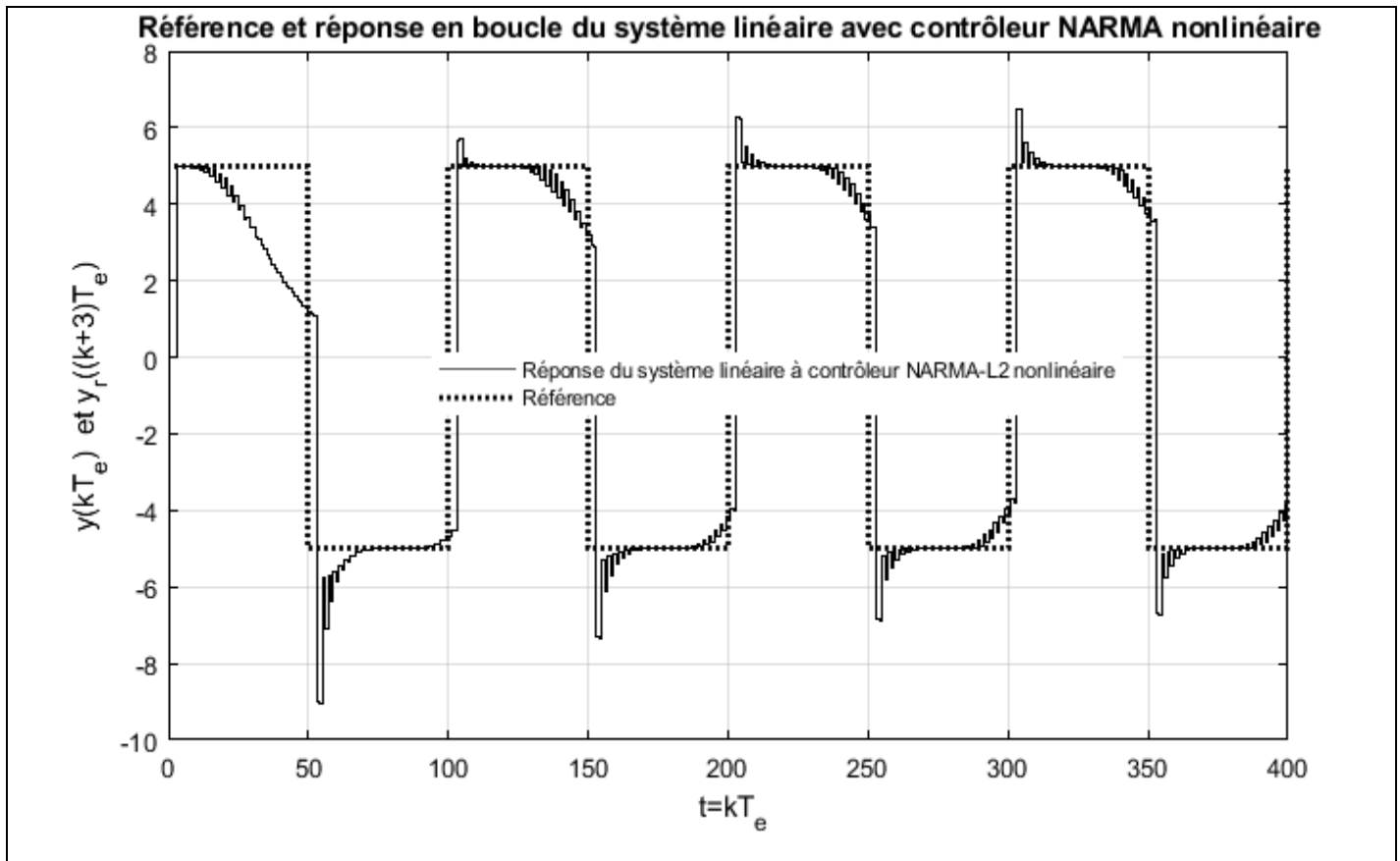


Figure (III-7) : Réponse du système linéaire utilisé avec le contrôleur NARMA-L2 non linéaire avec 10 neurones dans les couches 1 et 3.

**Exemple 2 :**

Le deuxième exemple que nous allons considérer est pris de [22]. L'équation aux différences qui régit le fonctionnement du système est donnée par :

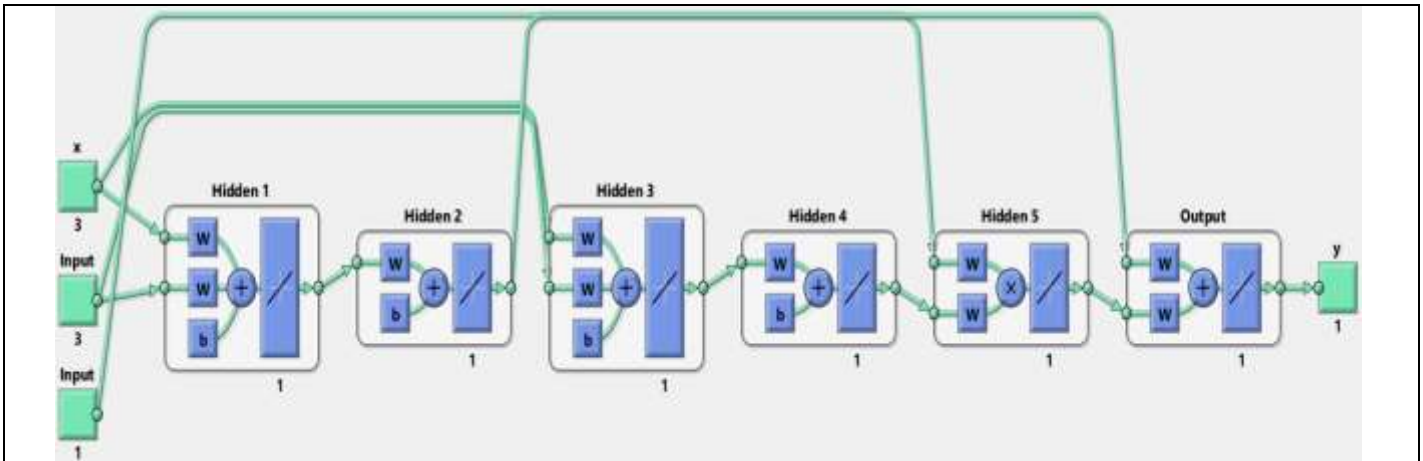
$$y(k + 1) = \frac{y(k)y(k - 1)(y(k) + 2.5)}{1 + y^2(k) + y^2(k - 1)} + u(k)$$

Le système est non linéaire et il n'est pas dans la forme adaptée pour la commande NARMA-L2. Pour concevoir la commande, nous avons supposé que le système soit quand même sous la forme canonique

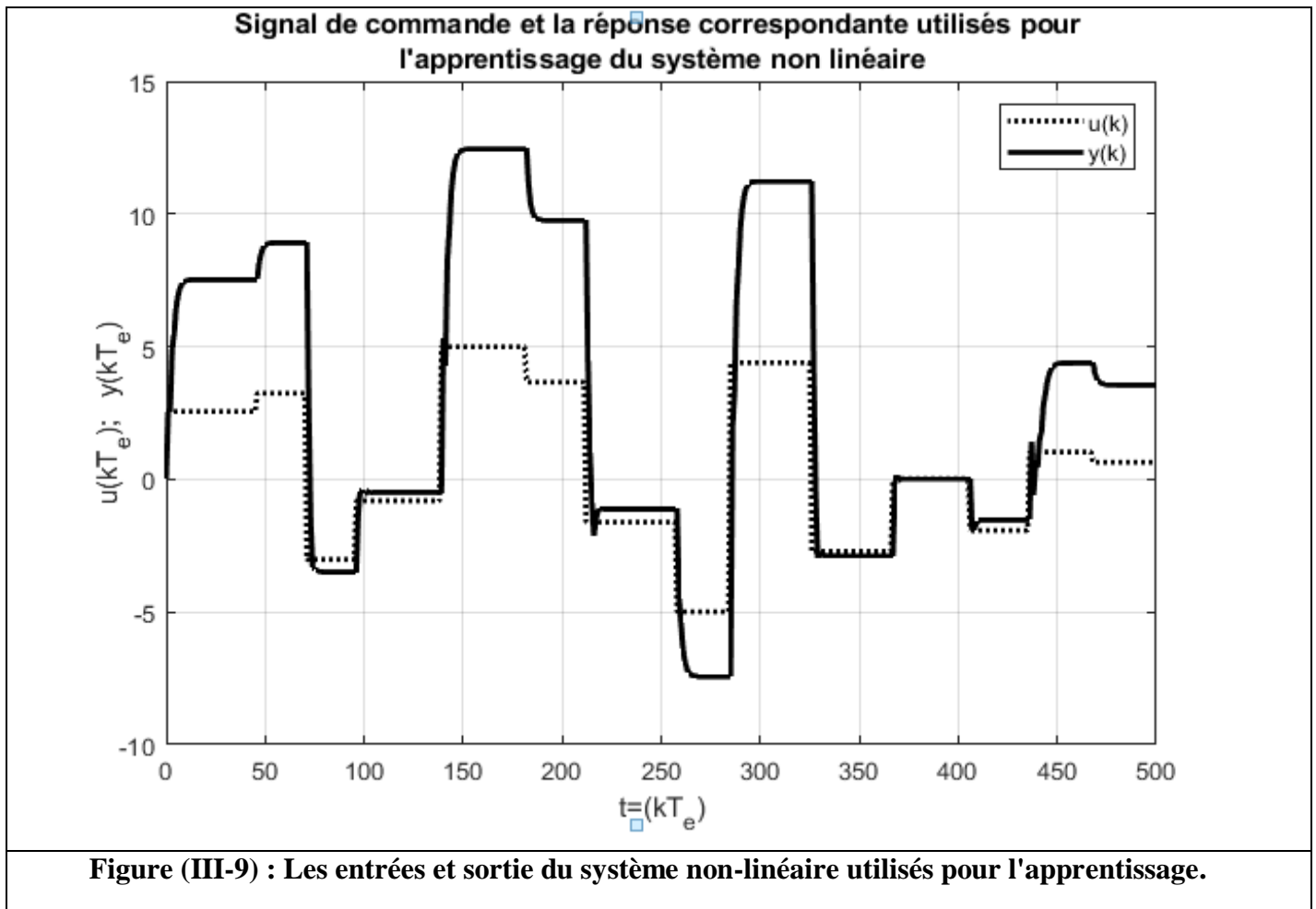
adaptée pour le contrôleur NARMA-L2 avec un degré relatif  $d = 2$  et un ordre  $n = 4$  c'est à dire qu'il peut être approximé par une équation aux différences de la forme :

$$y(k + 2) = f(y(k), y(k - 1), \dots, y(k - 3), u(k), \dots, u(k - 3)) + g(y(k), y(k - 1), \dots, y(k - 3), u(k), \dots, u(k - 3))u(k + 1)$$

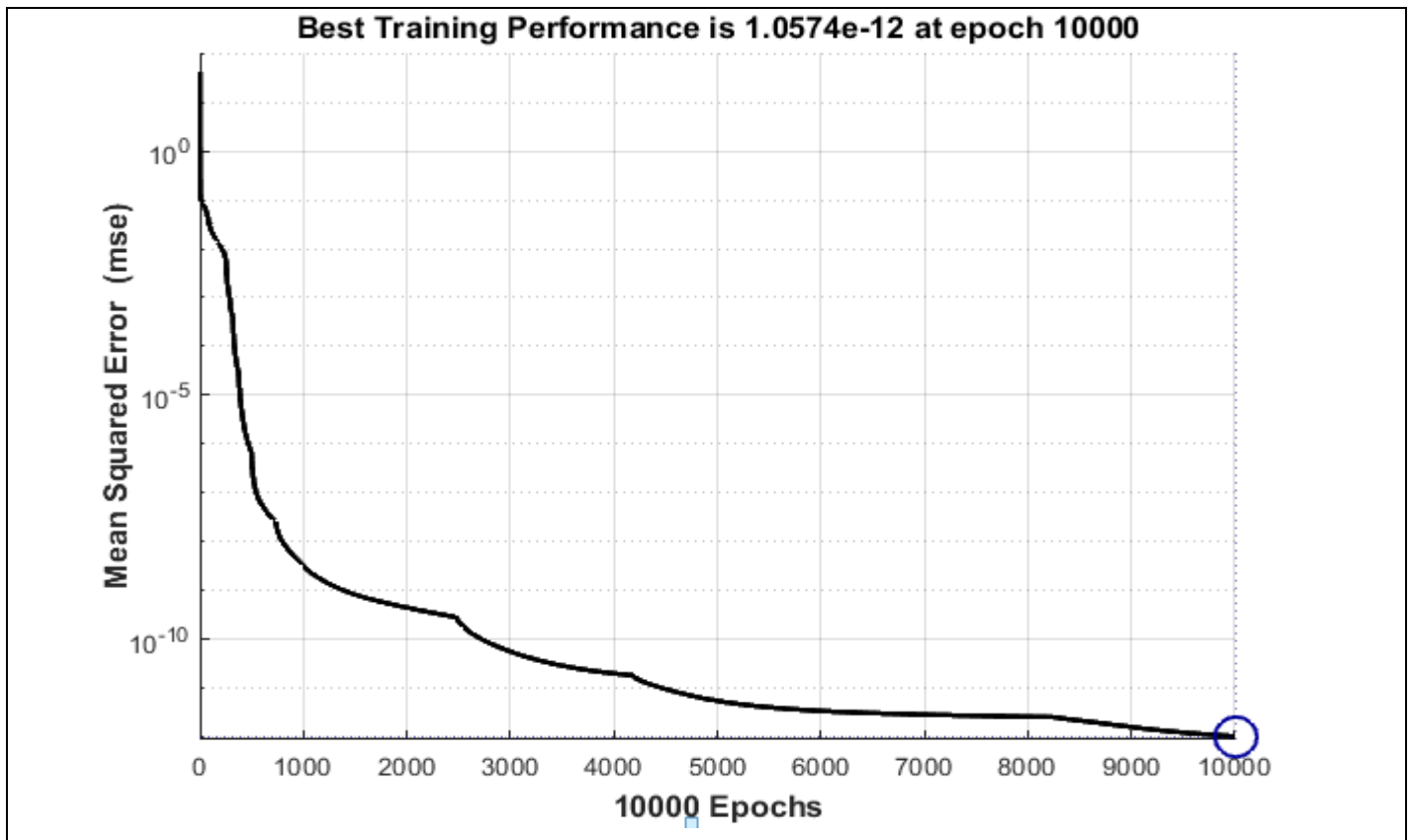
Nous avons choisi un modèle avec 20 neurones dans les couches 1 et 3 et une fonction d'activation de la forme tangente hyperbolique "tansig". Pour éviter de faire l'apprentissage en utilisant la back-propagation dynamique qui risque d'être très lente, nous avons transformé le modèle neuronal en un modèle statique pour lequel les données sont tout simplement décalées par l'utilisateur car le modèle statique ne contient pas de retards et c'est un modèle en boucle ouverte. Ce modèle est représenté sur la figure (III-8). A la fin de l'apprentissage, les poids synaptiques obtenus sont copiés dans le modèle dynamique de départ pour obtenir le contrôleur NARMA-L2. Les données utilisées pour l'apprentissage ont été générées en utilisant le modèle exact du système non linéaire donné par son équation aux différences. A ce modèle nous avons appliqué une entrée qui est formée d'une série d'impulsions rectangulaires de durées et d'amplitudes aléatoires. La figure suivante représente les données utilisées pour l'apprentissage du modèle.



Figure(III-8) : Modèle neuronal non cyclique (Feedforward) utilisé pour l'apprentissage.

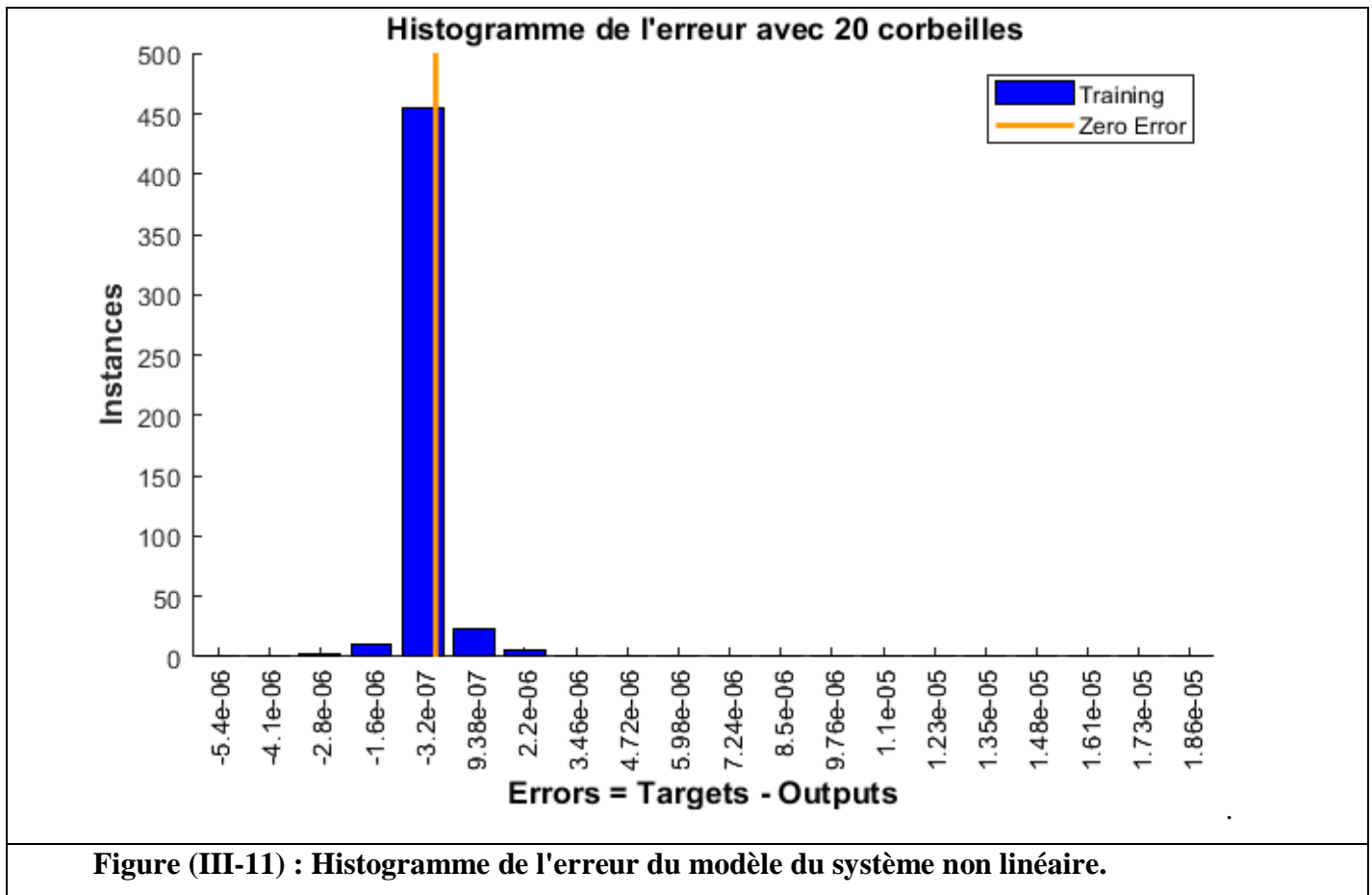


A titre indicatif, nous donnons dans ce qui suit certaines caractéristiques généralement utilisées pour évaluer le modèle neuronal obtenu. Nous commençons par l'erreur moyenne des moindres carrés. La figure suivante donne l'évolution du critère "mse" à travers les différentes époques d'apprentissage. La meilleure valeur pour cette somme ( $J = \frac{1}{N_f} \sum_0^{N_f} (y_{nnt}(k) - y(k))^2 = 1.0574 * 10^{-12}$ ) est obtenue à la 10000 époque. On doit noter qu'ici nous n'avons pas utilisé la régularisation des poids synaptique ni l'économisassions dans le critère de performance.



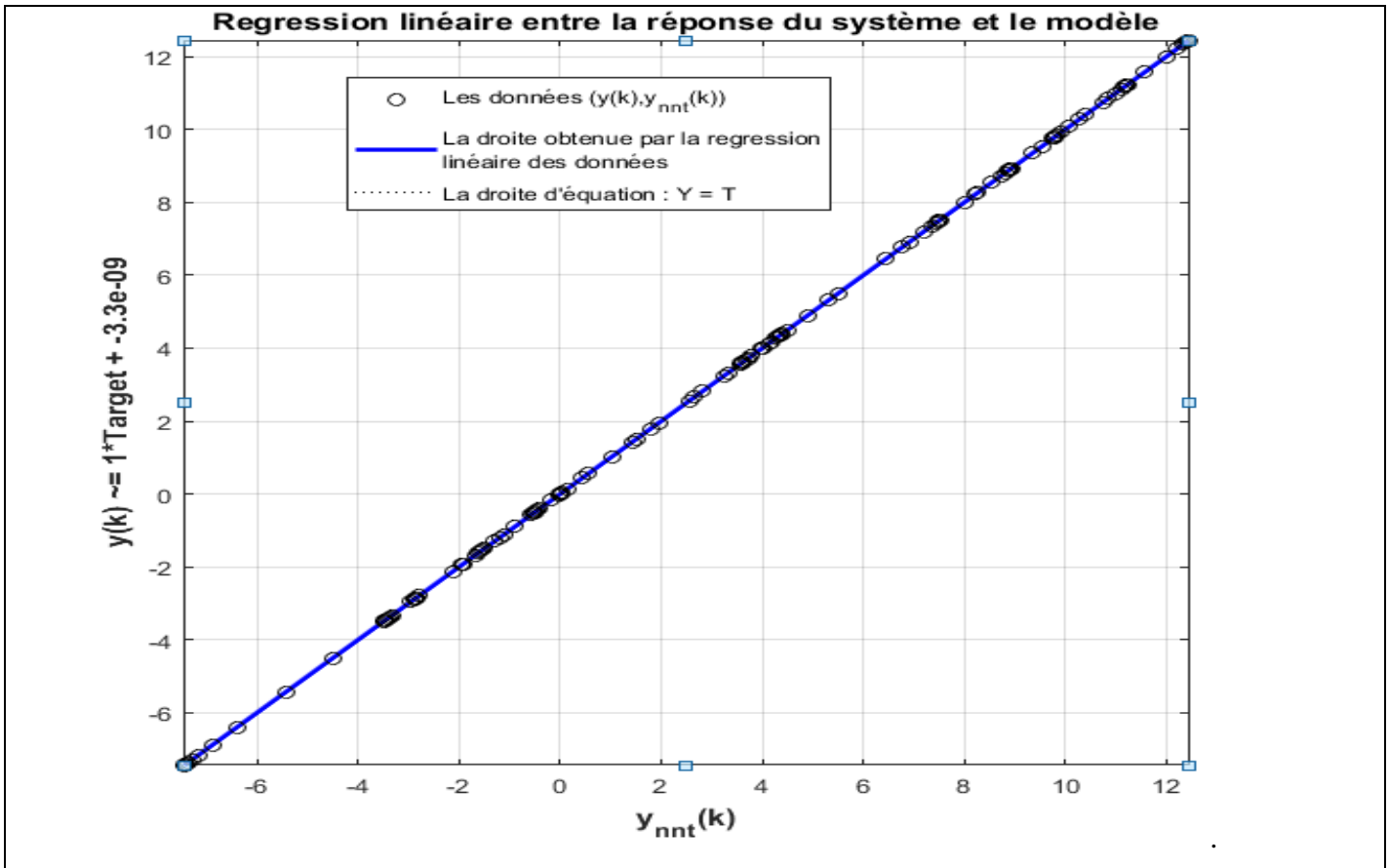
**Figure (III-10) : Evolution de la moyenne de la somme des carrés de l'erreur à travers les différentes époques d'apprentissage pour le modèle du système non linéaire.**

La deuxième caractéristique, qu'on peut aussi utiliser pour évaluer le modèle neuronal, est l'histogramme de l'erreur. Dans cet histogramme, les valeurs de l'erreur sont partagées en un ensemble d'intervalles appelés corbeilles et avec chaque intervalle on associe le nombre des erreurs  $e(k) = (y_{nn}(k) - y(k))$  qui appartiennent à cet intervalle. A partir de l'histogramme en question donné dans la figure (III-11), nous constatons que pratiquement toutes les instances de valeurs de l'erreur sont situées dans la corbeille voisinant 0 et que l'erreur ne dépasse jamais  $10^{-6}$ . Ainsi, l'histogramme des erreurs donne une idée sur les valeurs particulières de l'erreur et non seulement sa valeur moyenne.



On peut aussi utiliser la régression linéaire entre la réponse du système  $y(k)$  et la sortie du modèle neuronal  $y_{nnt}(k)$ . Dans le cas idéal où  $y(k) = y_{nnt}(k)$  cette régression est représentée par une droite (la première bissectrice du plan x-y). Plus cette régression est proche de cette droite plus le modèle est plus performant. Pour notre cas, on voit sur la figure (III-12) que la régression linéaire est complètement confondue avec la première bissectrice, ce qui implique que l'apprentissage est effectué avec succès.

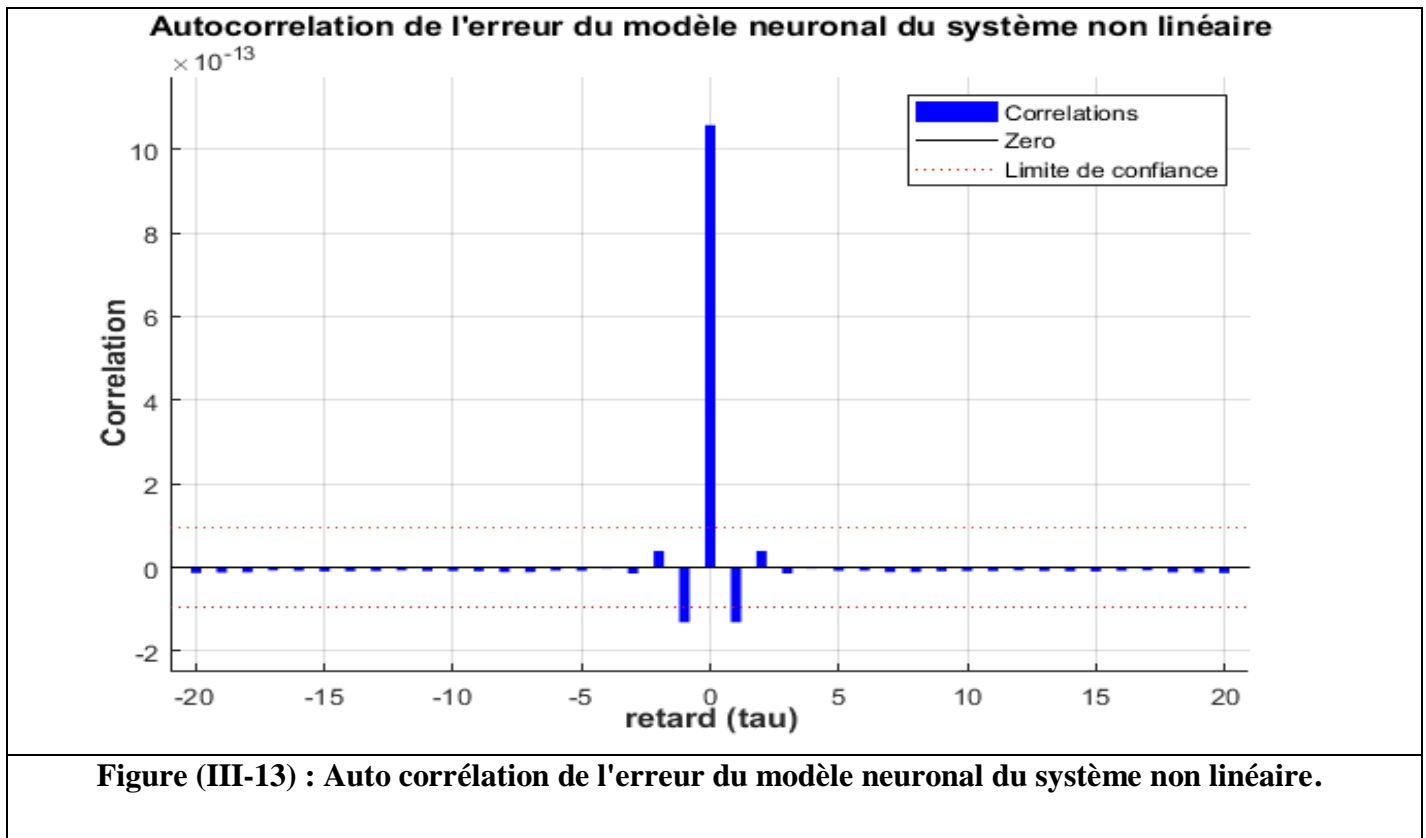
Une autre vérification valable consiste à tracer les courbes de  $y(k)$  et de  $y_{nnt}(k)$  pour voir à quel point elles sont confondues ou bien tracer directement la courbe de l'erreur pour voir son évolution.



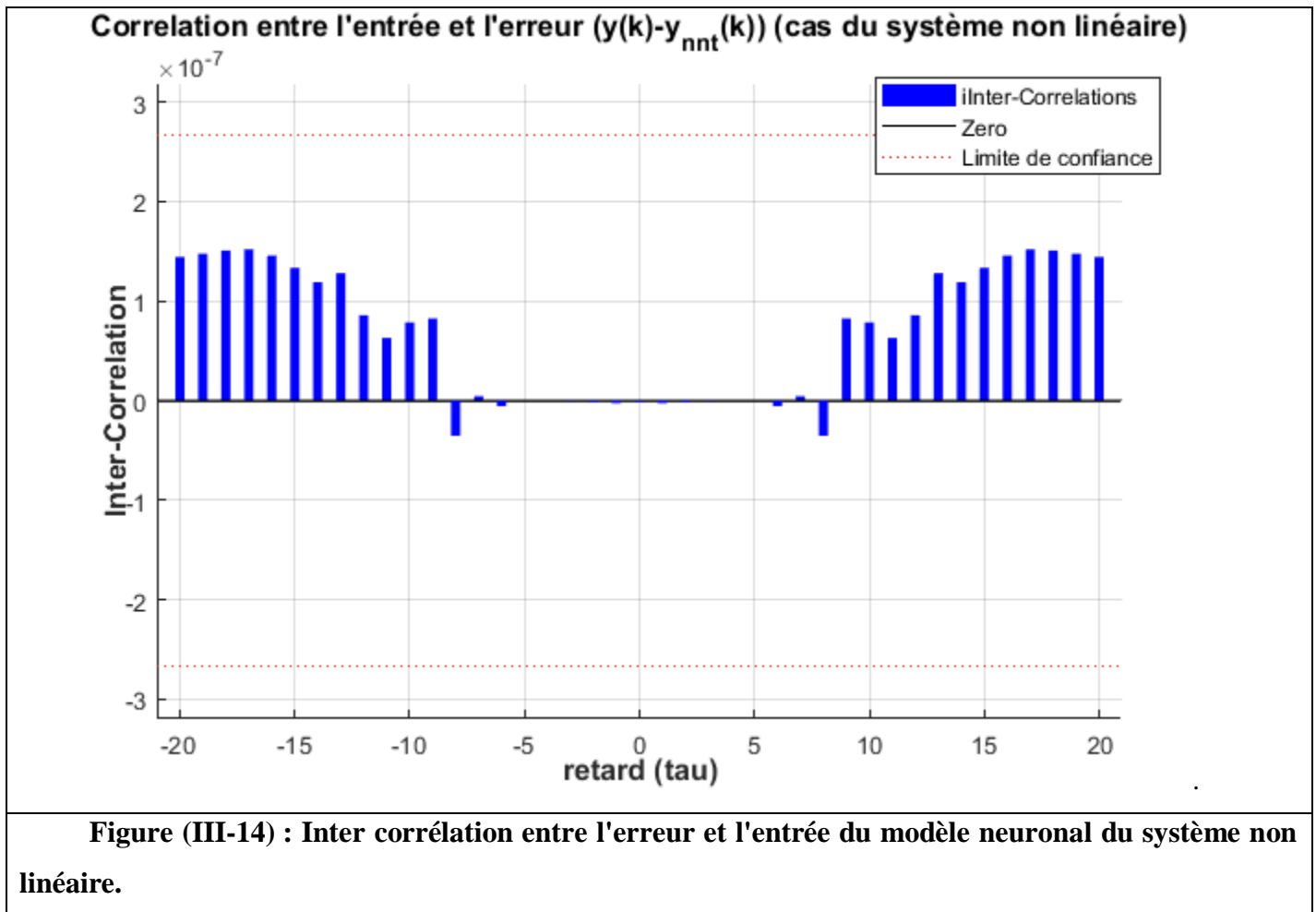
**Figure (III-12) : Régression linéaire entre les sorties du système non linéaire et son modèle neuronal.**

Pour connaître si le modèle neuronal a capturé toute l'information contenue dans les données, on utilise la corrélation. Dans le cas où le modèle est idéal, l'erreur doit être non corrélée (bruit blanc) et son auto corrélation doit être très proche de l'impulsion de Dirac avec des valeurs de part et d'autre de zéro incluses dans l'intervalle de confiance. La figure (III-13) donne la fonction d'auto corrélation de l'erreur du modèle neuronal.





On peut aussi examiner la fonction d'inter-corrélation entre l'erreur et l'entrée. Dans le cas idéal où le modèle a capturé toute l'information contenue dans les données, il ne doit y avoir aucune relation (corrélation) entre l'entrée et l'erreur, c'est à dire que leur inter-corrélation doit être partout proche de zéro (dans la limite de confiance). La figure (III-14) représente la fonction d'inter-corrélation de l'erreur entre la sortie du modèle neuronal et celle du système avec l'entrée correspondante.



Après l'étape d'identification du modèle neuronal, on passe à la commande du système en boucle fermée en utilisant le contrôleur NARMA-L2 obtenu à partir des paramètres du modèle neuronal. Les figures (III-15) donnent les résultats obtenus. Au premier cas, l'entrée est le signal utilisé pour l'apprentissage. La constatation qu'on peut faire est que le système n'arrive pas à suivre une référence qui varie trop souvent avant que le contrôleur n'arrive à suivre sa nouvelle valeur.

Pour éclaircir ce problème et avoir une idée sur la généralisation, nous allons dans le deuxième cas utiliser une référence différente qui ne varie pas trop souvent. On voit dans la figure (III-16) que la poursuite de la référence est beaucoup plus améliorée.

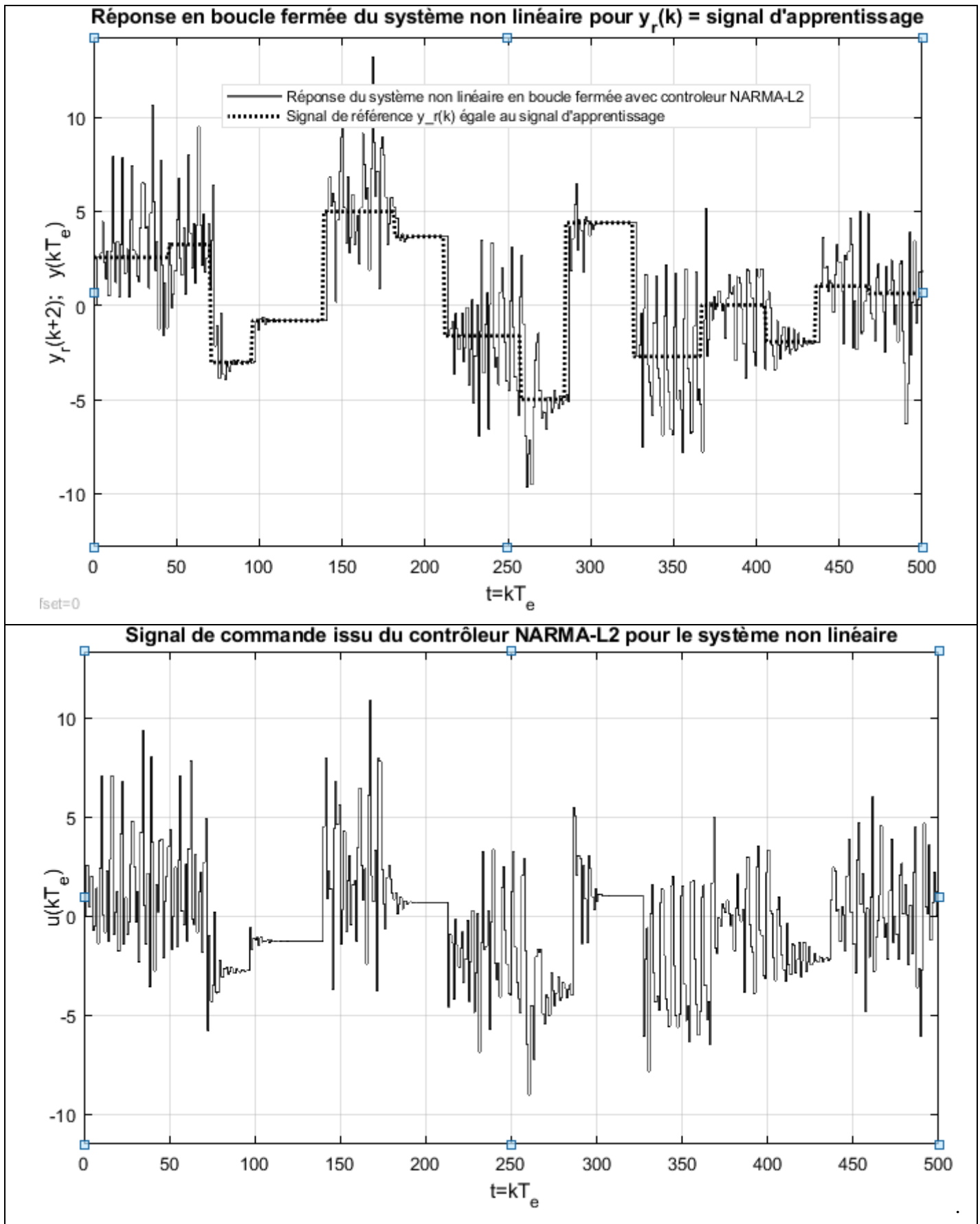
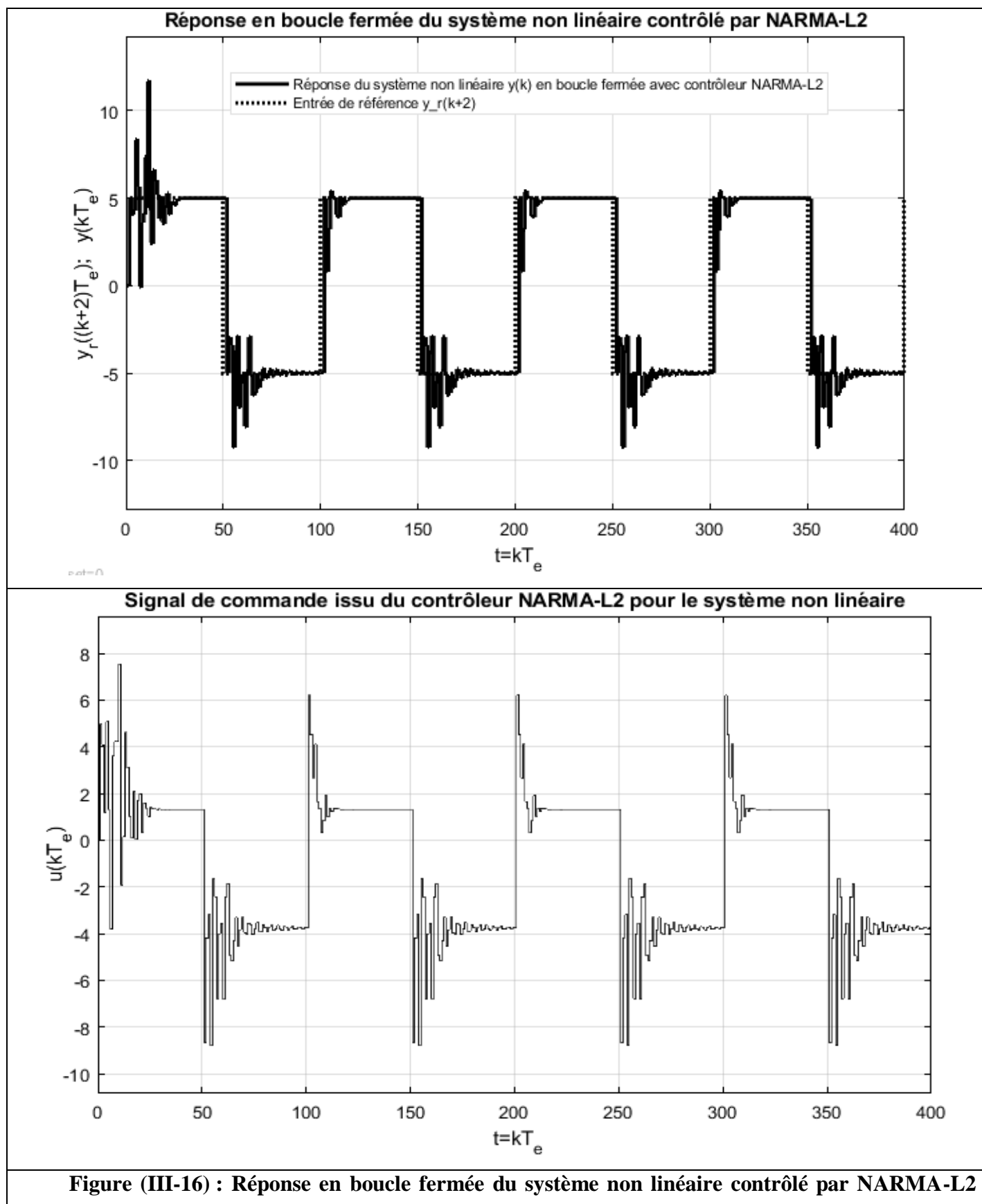


Figure (III-15) : Réponse en boucle fermée du système non linéaire avec contrôleur NARMA-L2 (cas de référence égale au signal d'apprentissage).

La conclusion qu'on peut faire ici, est que le signal utilisé pour la génération des données d'apprentissage doit être suffisamment riche pour que le modèle soit assez fidèle sans se soucier de la faite que le contrôleur pour le poursuivre ou non.



(cas d'une entrée différente du signal d'apprentissage.)

### III.3. Commande neuronale à base de modèle de référence :

Maintenant, nous allons appliquer la commande neuronale à base de modèle de référence au système non-linéaire de l'exemple 3 de [22] :

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

Cette stratégie de commande s'effectue en deux étapes. La première étape consiste à faire l'identification du système en utilisant un modèle NARX neuronal. Cette étape est similaire à l'étape d'identification que nous avons effectuée pendant la conception de la commande NARMA-L2 seulement pour ce cas le modèle n'a pas à avoir une structure canonique. La deuxième étape consiste en la conception du contrôleur. Durant cette étape on augmente le réseau neurone obtenu pendant l'identification du système avec deux couches en amont qui représentent le contrôleur. Ensuite, on configure les poids synaptiques et les biais des couches du modèles (couches 3 et 4) comme fixes (ne changent pas durant l'apprentissage). Enfin, le réseau de neurone global est entraîné avec une séquence de données issue d'un modèle de référence. Cet apprentissage permet de calculer les poids synaptiques du contrôleur. La figure suivante représente le réseau de neurone global avec la partie modèle et la partie contrôleur.

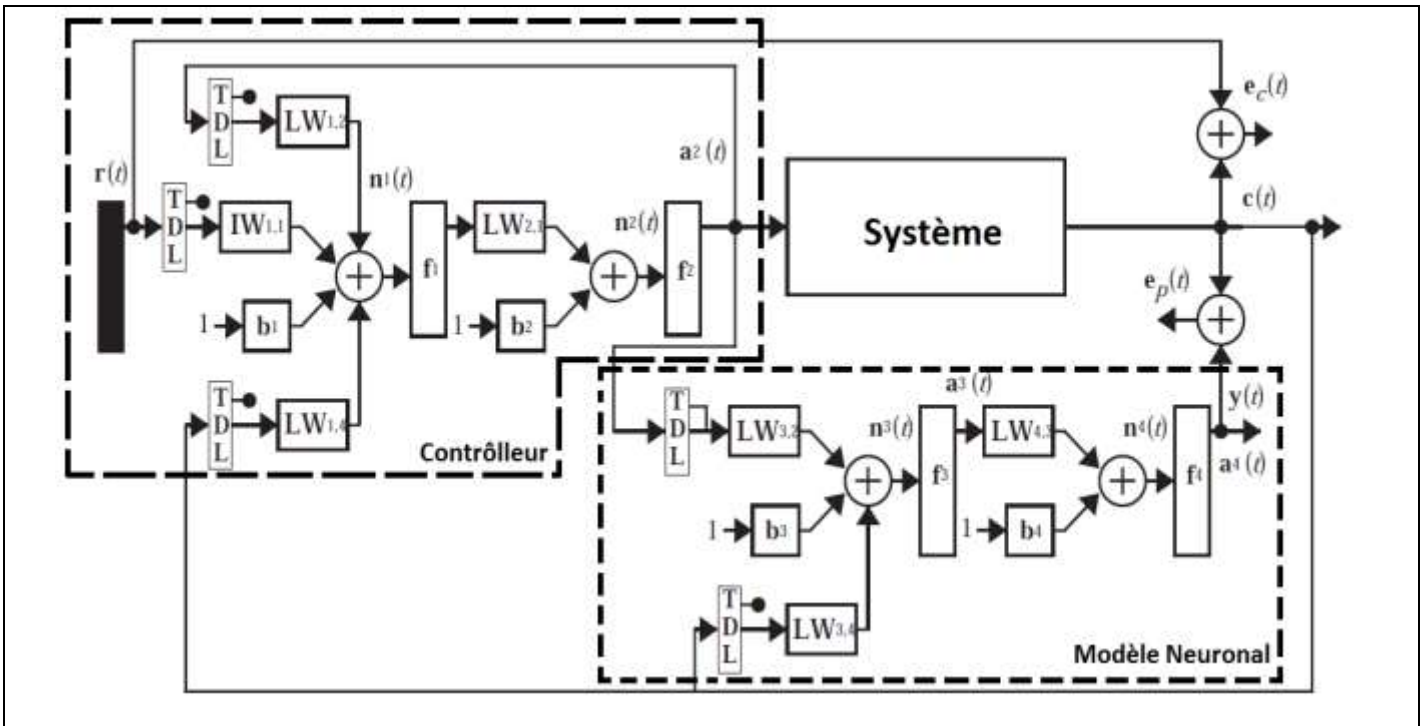


Figure (III-17) : Structure de la Commande Neuronale à modèle de référence de systèmes dynamiques.

Pour le modèle du système, nous avons utilisé un réseau de neurone de deux couches avec 5 neurones dans la couche cachée et 2 retards à l'entrée et 2 retards dans la contre réaction. Ce réseau est représenté par la figure suivante :

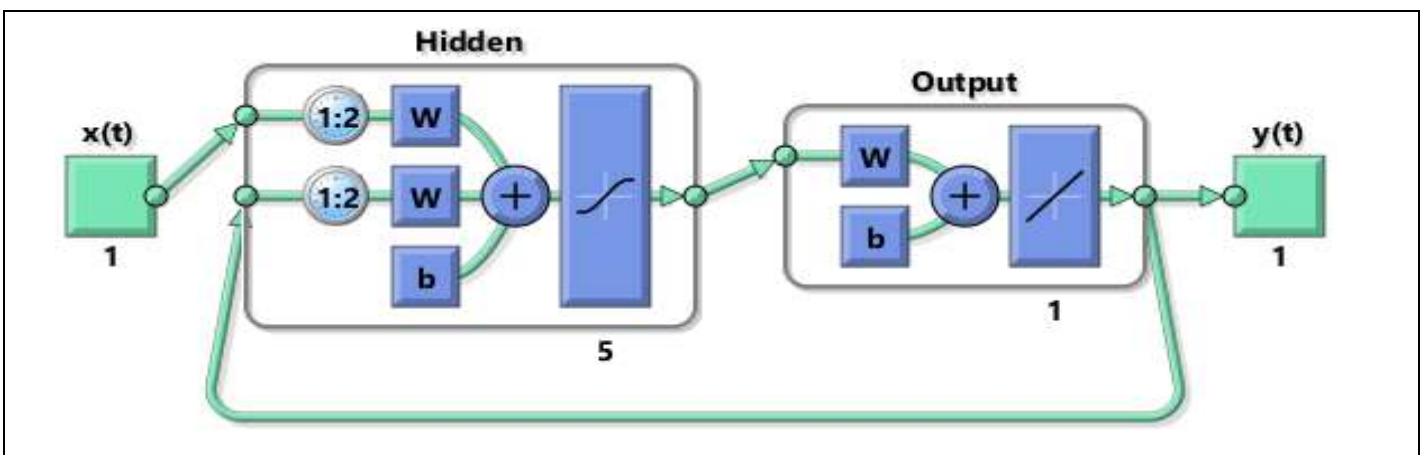


Figure (III-18) : Structure du réseau de neurone utilisé comme modèle pour le système

Pour la génération des données d'apprentissage, nous avons utilisé Simulink où nous avons commandé le système en boucle fermée avec une commande linéarisant simple le bloc diagramme du système et de la commande en boucle fermée est donnée sur la figure suivante :

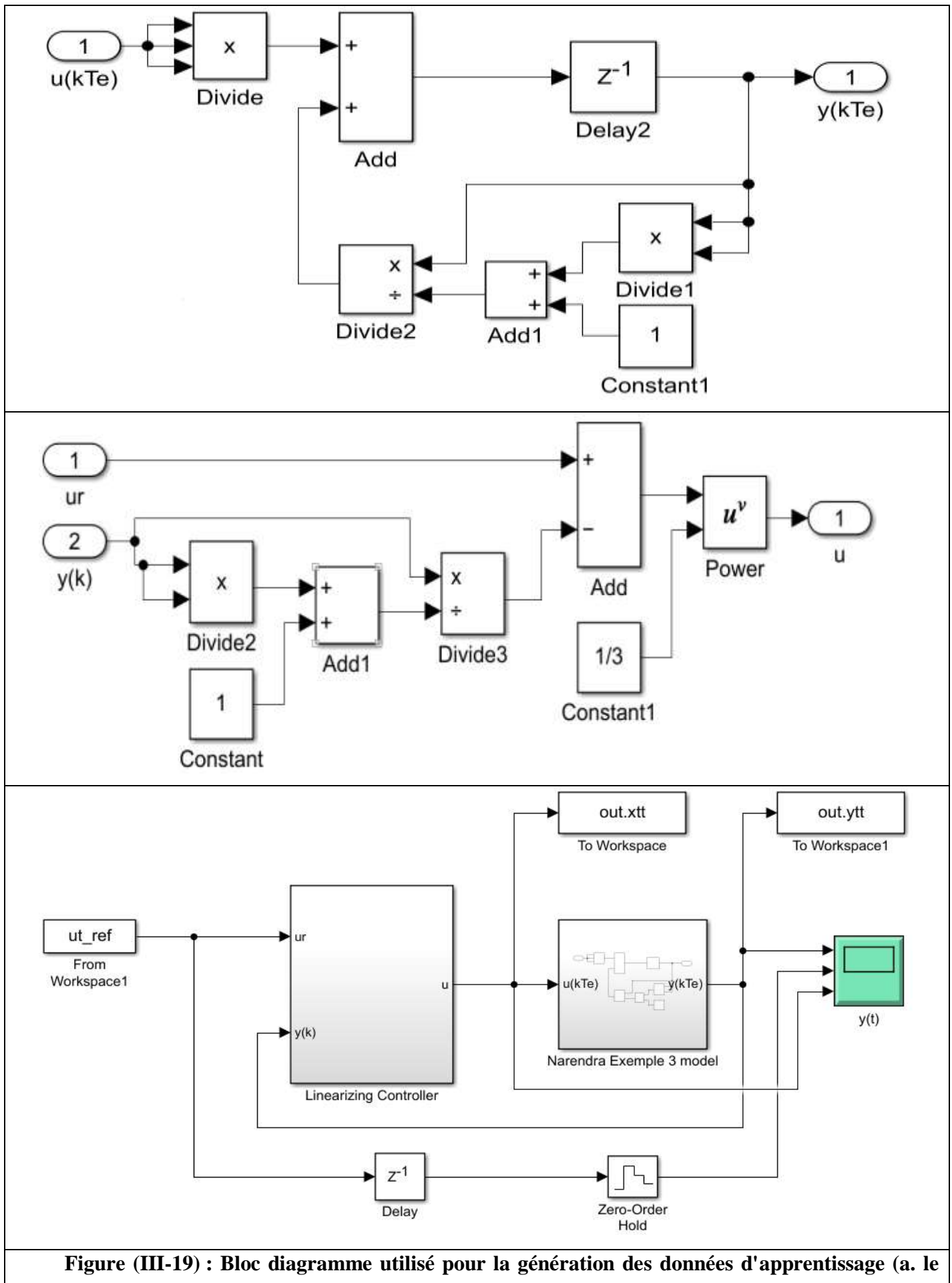
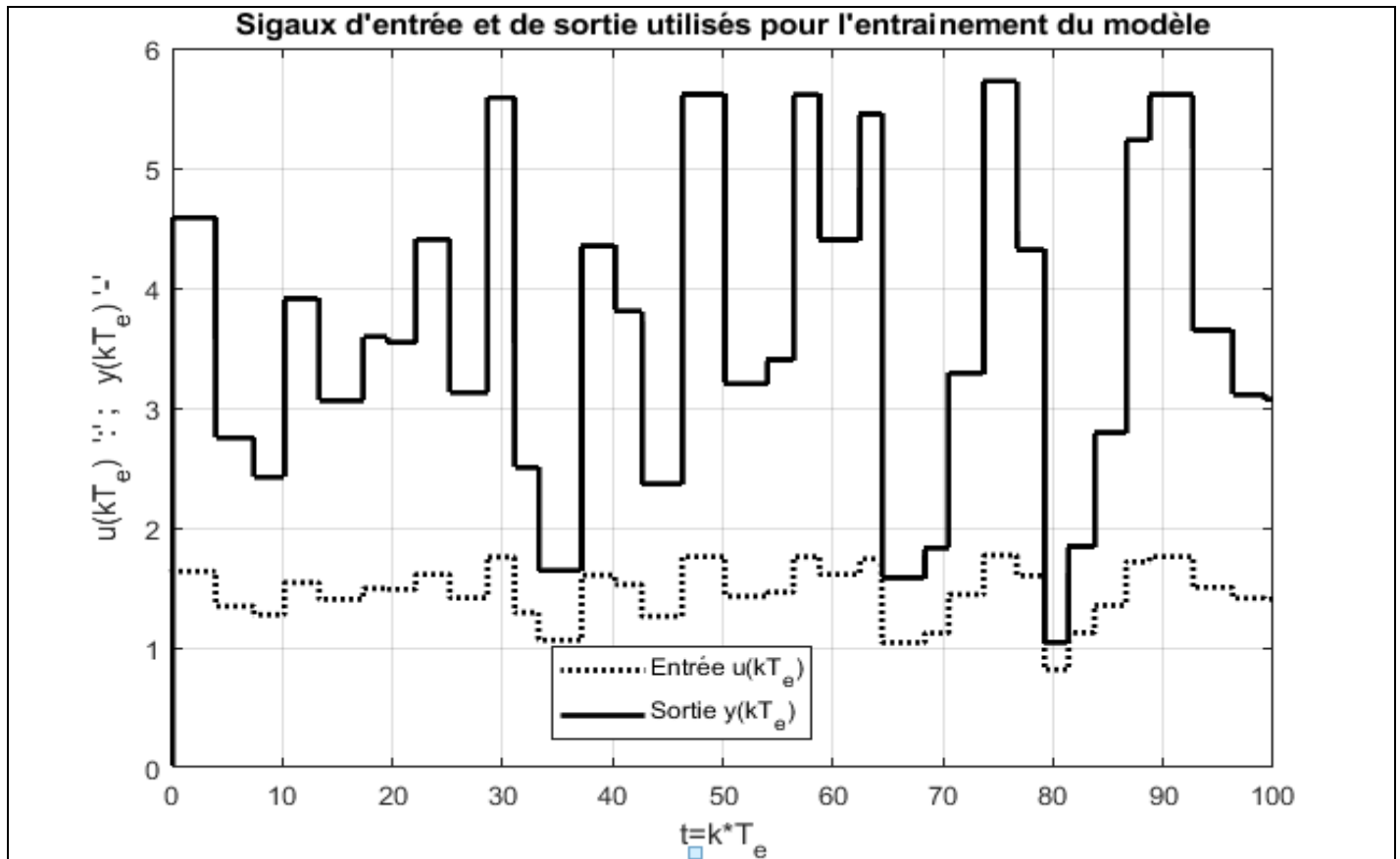


Figure (III-19) : Bloc diagramme utilisé pour la génération des données d'apprentissage (a. le

système, b. Le contrôleur, c. Structure en boucle fermée).

Les signaux d'entrées et de sorties obtenus sont représentés sur la figure suivante.

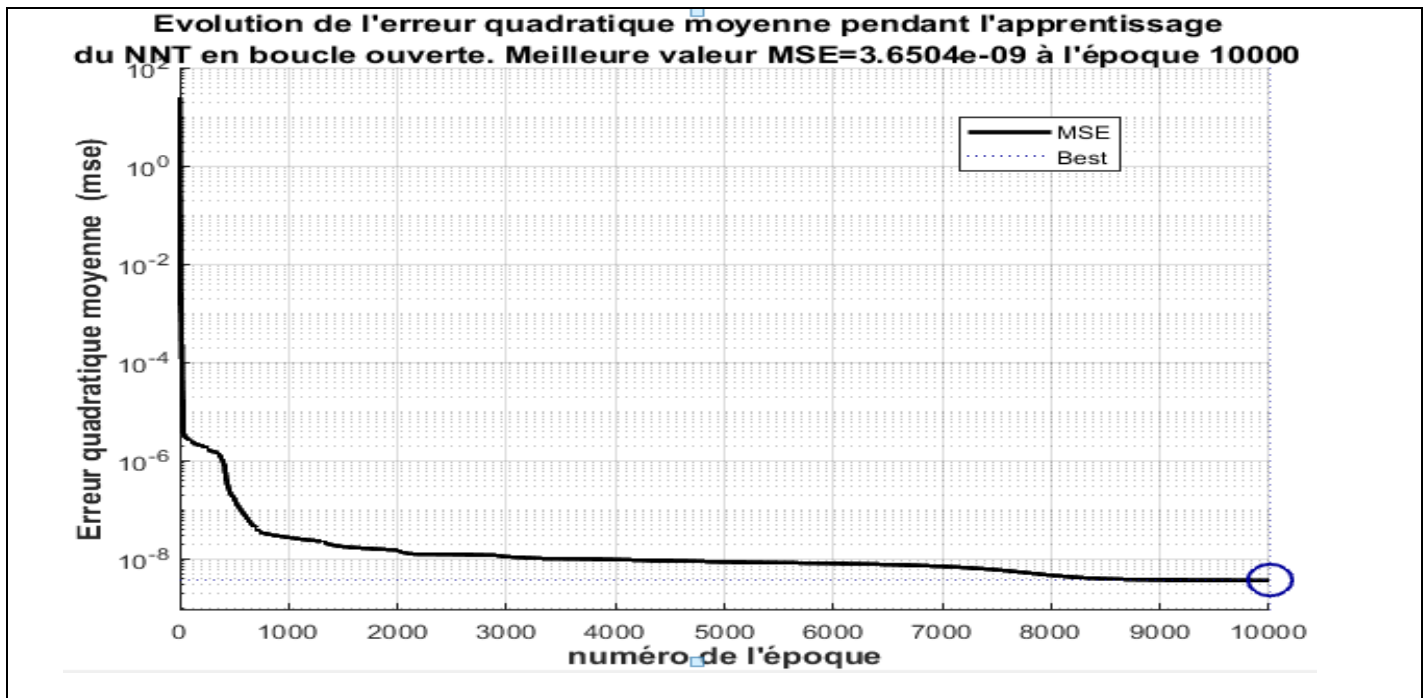


**Figure (III-20) : Signaux utilisés pour l'apprentissage du modèle.**

Les résultats de l'identification du modèle du système sont comme suit :

Pour l'apprentissage nous avons procédé comme suit : Au début nous avons fait l'apprentissage du réseau de neurones en boucle ouverte (la contre réaction est remplacée par les données de sorties du système et leurs décalages) pendant 10000 épochès. L'évolution de l'erreur quadratique moyenne "mse" est donnée par la figure suivante.

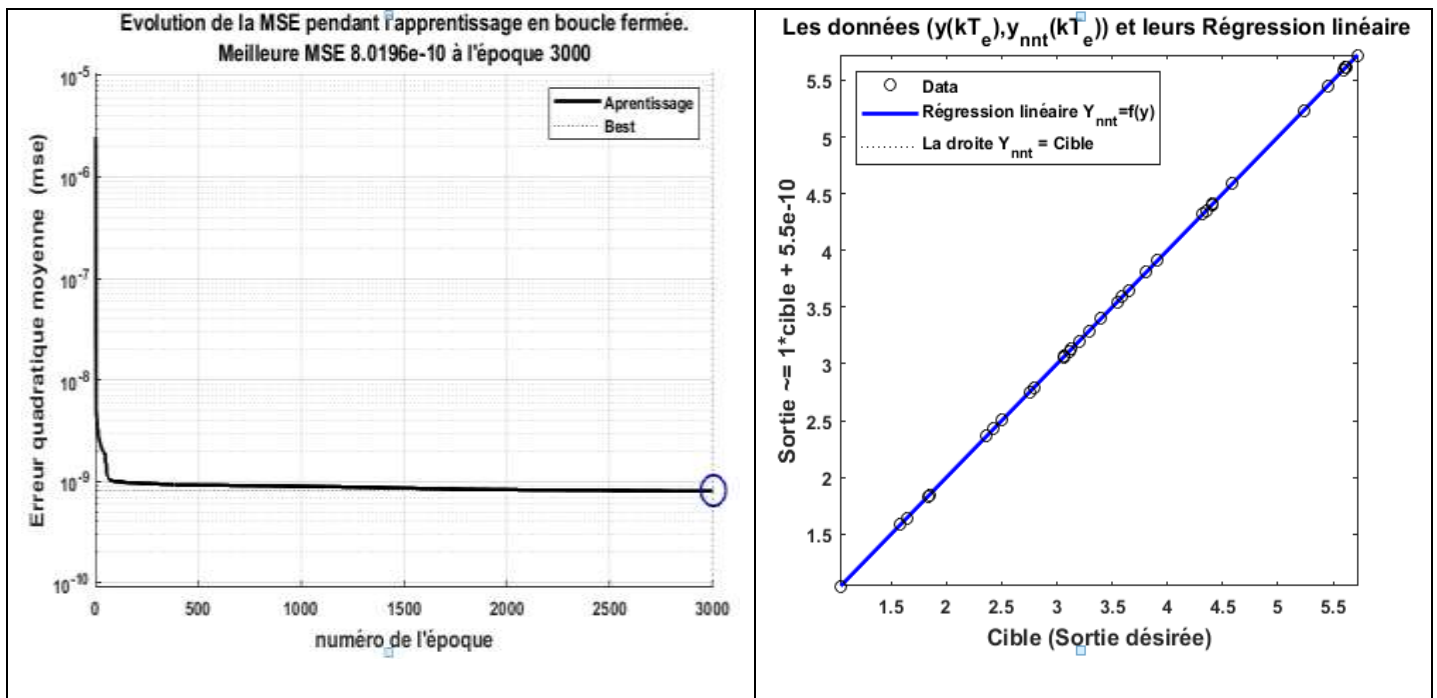




**Figure (III-20.b) Evolution du MSE pendant l'entrainement du modèle neuronal en boucle ouverte.**

Ainsi la valeur de l'erreur quadratique à la dernière itération est  $MSE = 3.65 * 10^{-9}$ . Nous avons ensuite initialisé le réseau de neurone au voisinage du précédent et refait l'apprentissage en boucle ouverte toujours pendant 3000 époques supplémentaires et ceci a amélioré la MSE à une valeur de  $MSE = 9.9525 * 10^{-10}$ .

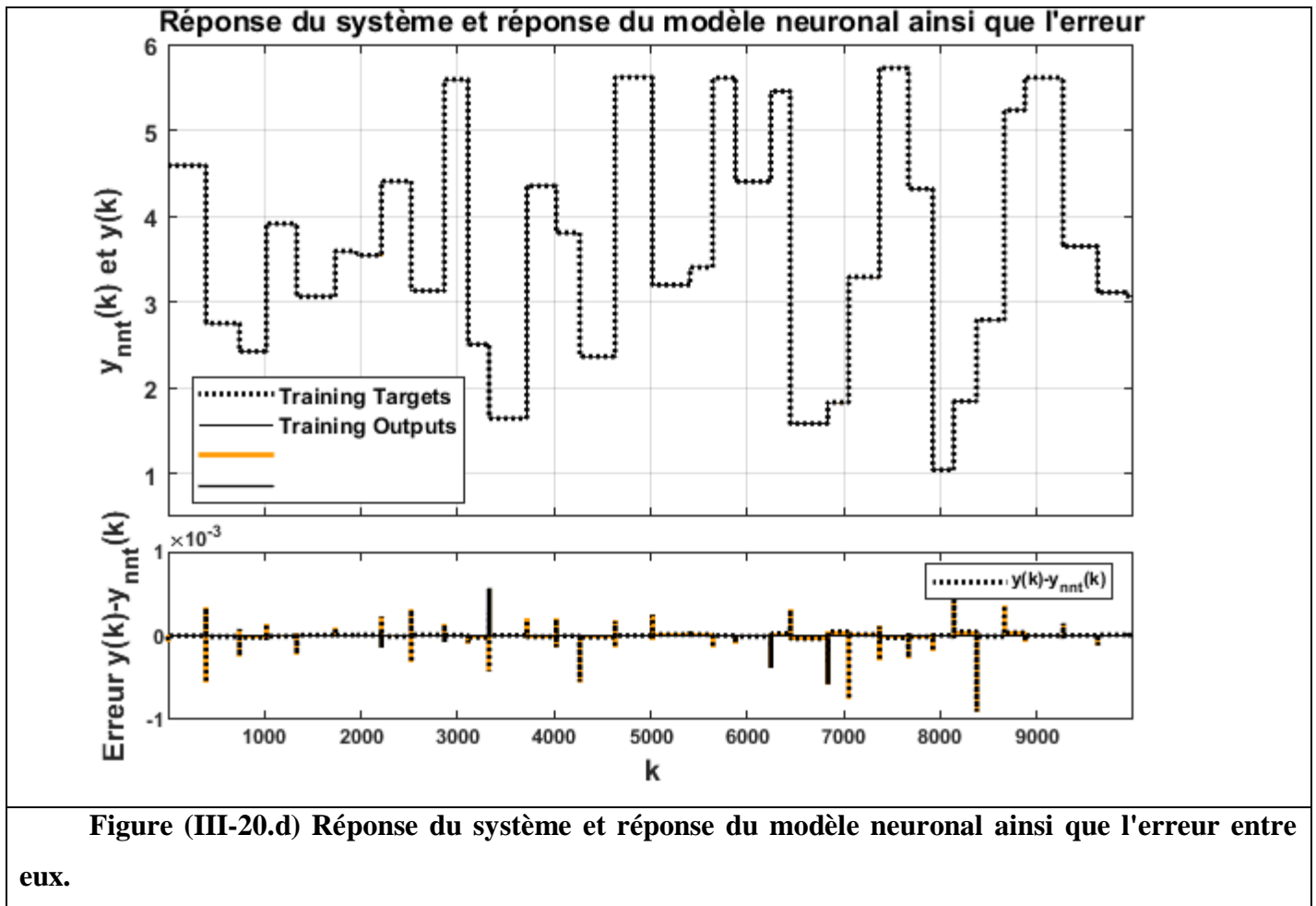
Après, nous sommes passé à l'apprentissage en boucle fermée (réseau de la figure (III-18)) de façon cumulative c'est à dire en partant par le dernier réseau de neurone obtenu pendant l'apprentissage en boucle ouverte et ceci pendant 3000 époques. Nous avons constaté que l'apprentissage en boucle fermée est beaucoup plus lent que l'apprentissage en boucle ouverte. Les résultats que nous avons obtenus sont données dans la figure suivante :



**Figure (III-20.c) Evolution de l'erreur quadratique moyenne pendant l'apprentissage en boucle fermée ainsi que la régression linéaire entre la sortie du réseau de neurone et la sortie du système.**

On remarque que cette façon de faire est bénéfique et elle permet d'accomplir une erreur quadratique moyenne de  $8.0196 \times 10^{-10}$ . La régression linéaire est parfaite et tous les points de données sont sur cette droite. La réponse du système et celle du modèle neuronal obtenu sont représentées par la figure suivante, accompagnées par l'erreur entre eux.

On remarque que quoique l'apprentissage en boucle ouverte a accomplie une  $MSE = 3.65 \times 10^{-9}$ , l'apprentissage en boucle ouverte cumulatif a commencé par une  $MSE \approx 10^{-6}$ . Ceci est du au faite que l'apprentissage en boucle ouverte n'est qu'approximatif car dans ce genre d'apprentissage, on remplace la contre réaction de la sortie  $y_{nnt}(k)$  du réseau lui-même par la cible  $y(k)$ .



Après avoir obtenu le modèle neuronal, nous avons créé un nouveau réseau de neurone de 4 couches qui va contenir le modèle et le contrôleur. Evidemment, la troisième couche et la quatrième doivent avoir la même structure que celle du modèle et leurs poids synaptiques et leurs biais sont copiés directement du modèle obtenu dans l'étape précédente. Pour le contrôleur nous avons utilisé 10 neurones dans la couche cachée, un retard dans l'entrée, deux retards dans la contre réaction de la commande et un retard dans la contre réaction de la sortie. Le réseau de neurone global est représenté par la figure suivante :

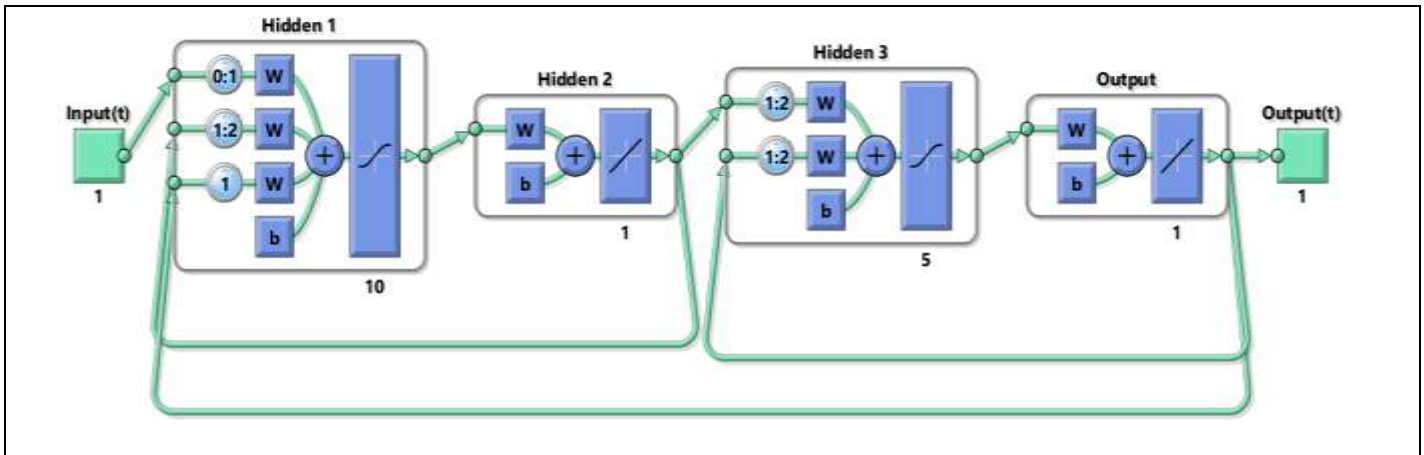
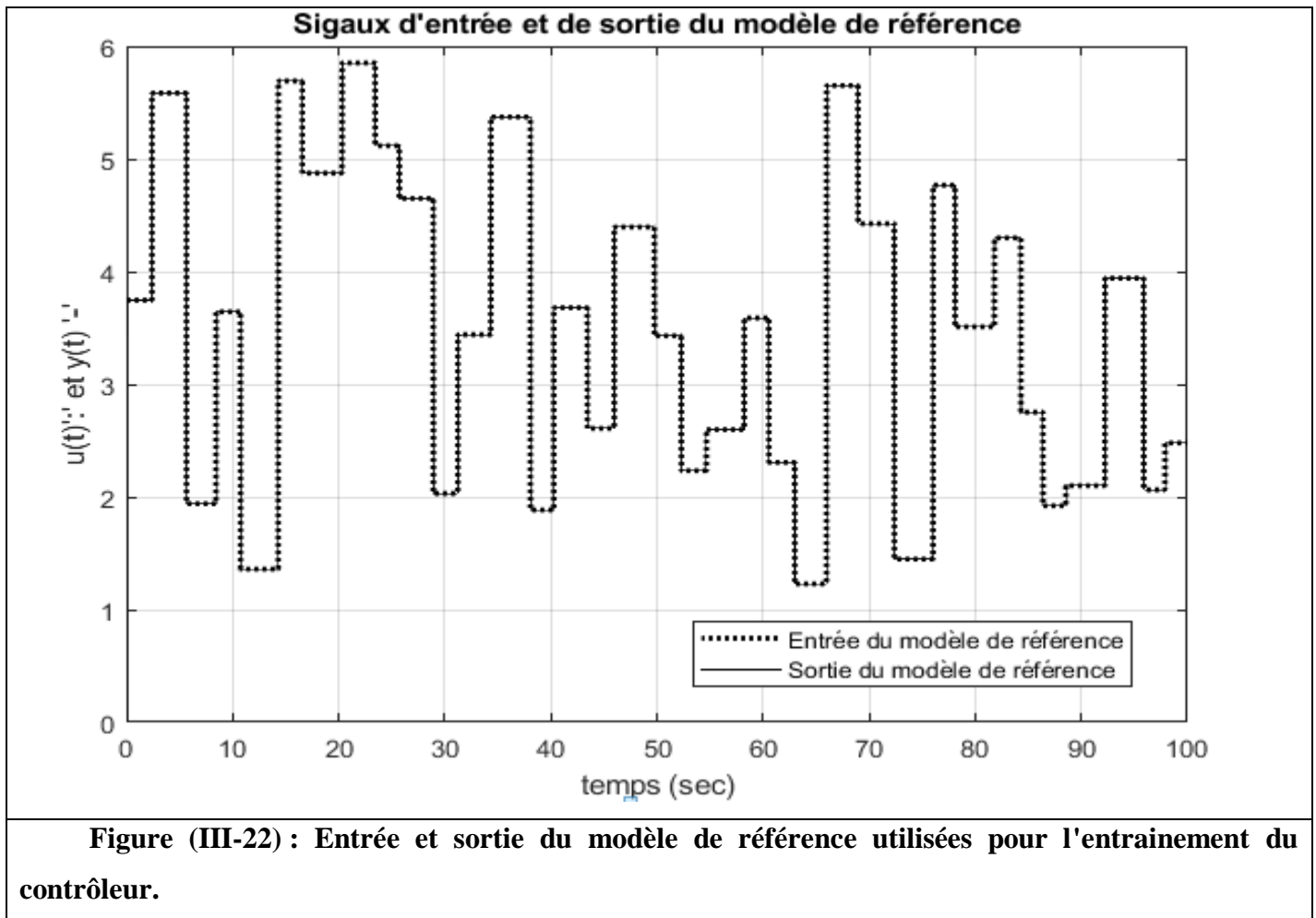


Figure (III-21) : Réseau de neurone contenant le modèle et le contrôleur.

Pour le modèle de référence, nous avons utilisé la fonction de transfert suivante :

$$G(z) = \frac{0.9}{z + 0.1} \text{ avec } T_e = 0.01s$$

Le signal de référence est construit à partir de la réponse de ce système à une séquence d'impulsions rectangulaires de durées et d'amplitudes choisies aléatoirement. Les signaux d'entrée et de sortie du modèle de référence sont donnés par la figure suivante.



L'identification du réseau de neurone global a été partiellement difficile par rapport aux cas précédents. Ceci est dû essentiellement au fait que le réseau de neurone est dynamique et possède des retards non seulement à l'entrée mais aussi en sortie et dans les couches intermédiaires. La convergence n'est pas garantie pour chaque essai. Pour minimiser ces difficultés nous avons commencé par faire l'apprentissage sans contre réaction de la sortie qui sera remplacée par le décalage de la cible qui est dans ce cas  $y_r(k)$  pris comme entrées supplémentaires. Le réseau de neurone ainsi modifié est représenté par la figure suivante (III-22.b). Les poids synaptiques et les biais des couches 3 et 4 sont bien entendu fixes durant cet apprentissage. L'apprentissage a été fait pendant 200 époques et la meilleure performance obtenue après quelques tentatives est  $MSE = 0.00347$ .

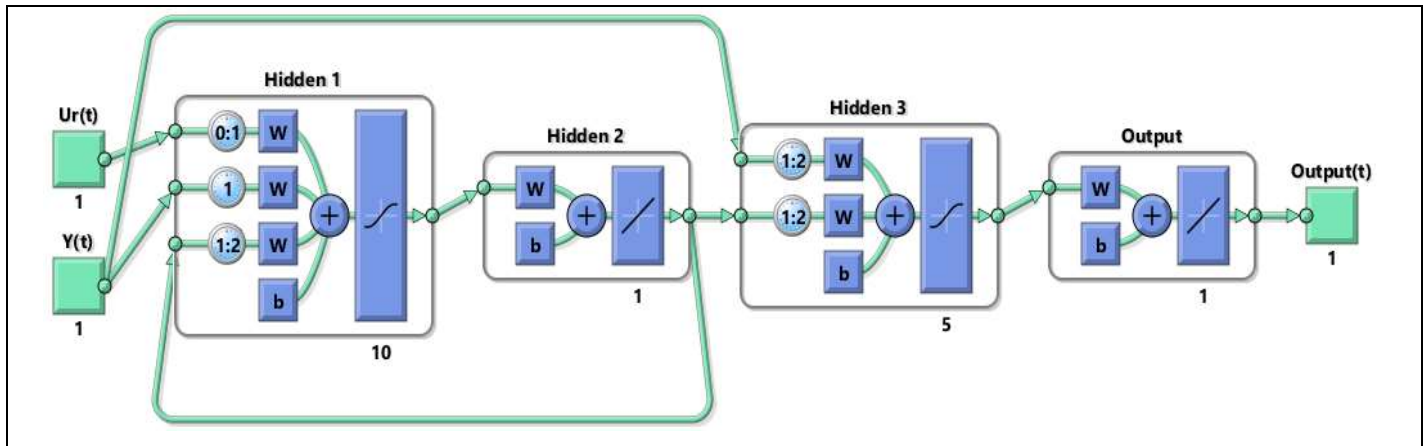
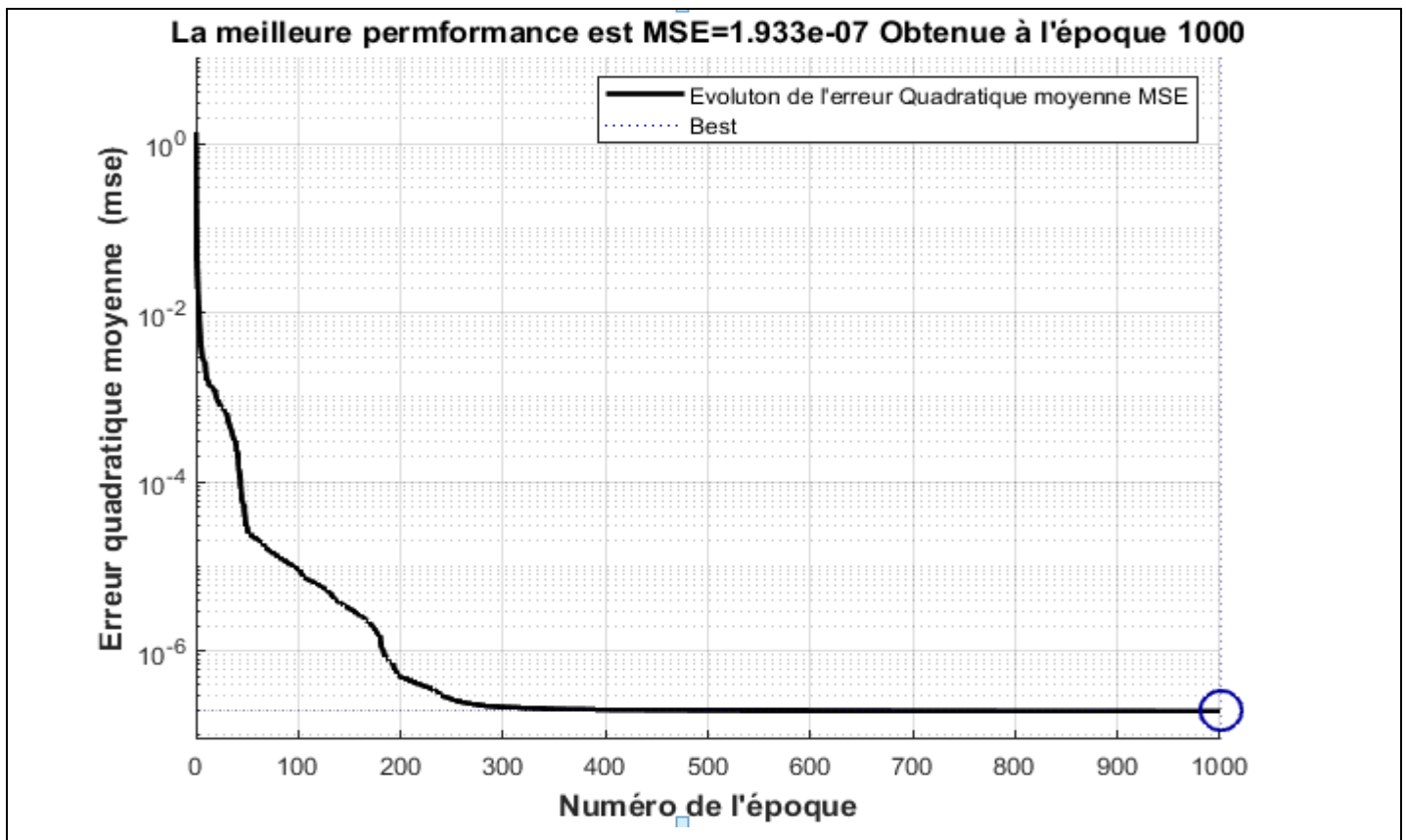
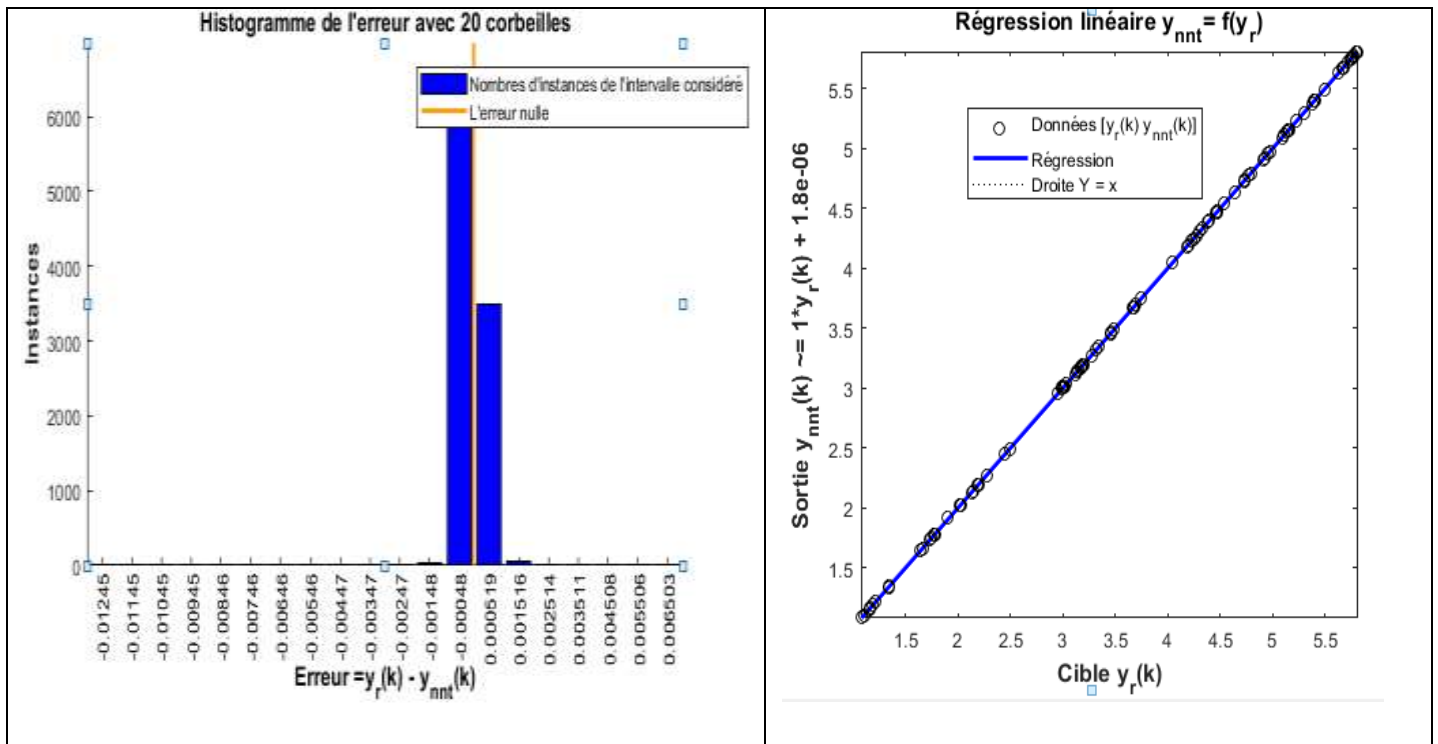


Figure (III-22.b) Structure en boucle ouverte utilisée pour le premier apprentissage.

Après ce premier apprentissage nous sommes passés à l'apprentissage du réseau de neurone en boucle fermée que nous avons effectué durant 1000 époques. Les résultats obtenus sont donnés par les figures qui suivent.

Nous donnons dans ce qui suit, les résultats des 10 dernières itérations (nous avons fait un apprentissage cumulatif plusieurs fois).





**Figure (III-23) Résultats de l'apprentissage en boucle fermée : a. Evolution de moyenne des carrés de l'erreur "MSE", b. Histogramme de l'erreur, c. Régression linéaire entre la réponse du modèle de référence et celle du réseau de neurone.**

Enfin, nous donnons la réponse en boucle fermée du réseau de neurone ainsi que la sortie du modèle de référence avec l'erreur entre les deux signaux.

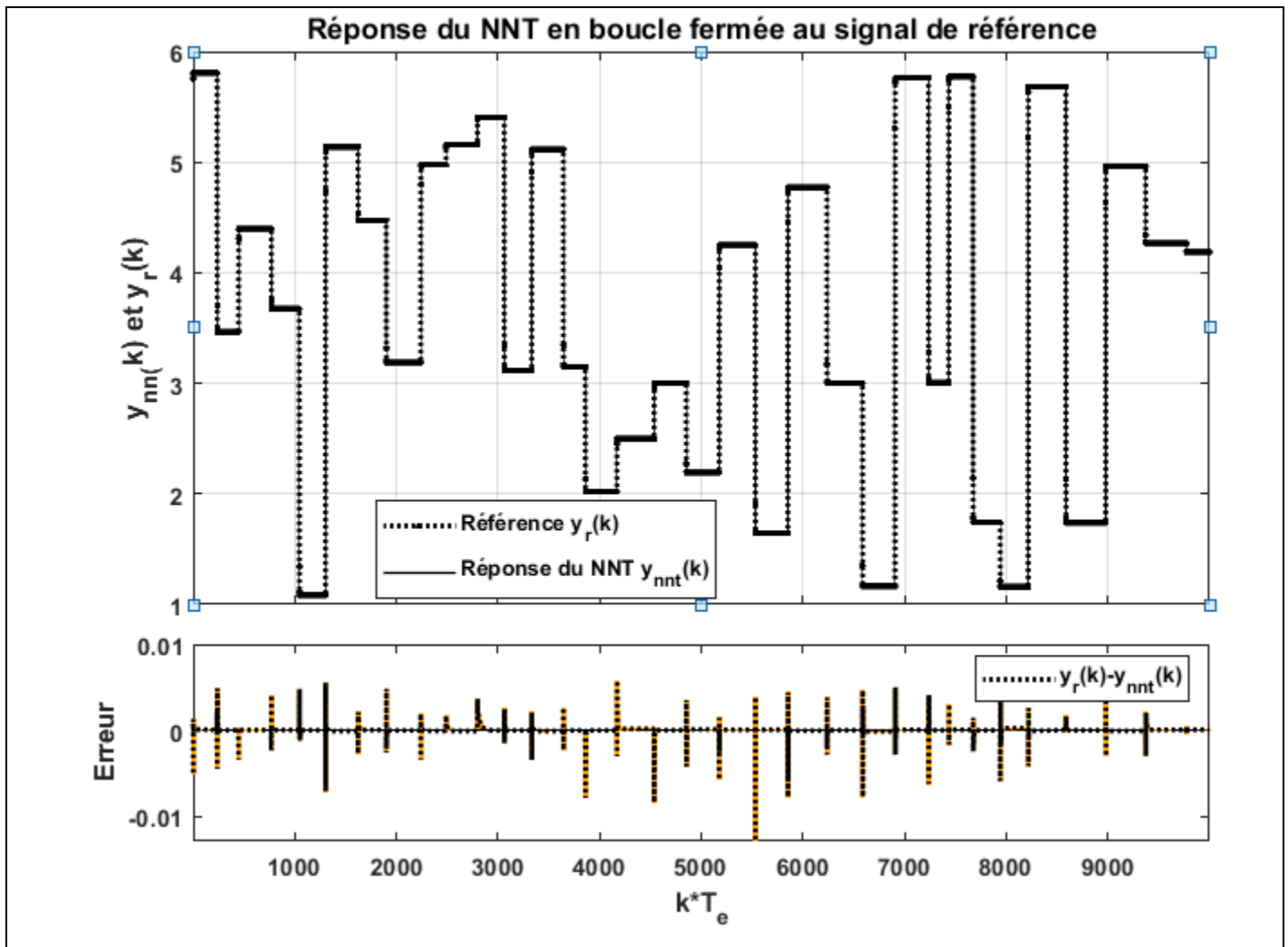


Figure (III-24) Réponse du réseau de neurone en boucle fermée avec celle du modèle de référence ainsi que l'erreur entre les deux.

On remarque que l'erreur devient importante seulement aux instants de discontinuité du signal de référence ce qui est typique pour toute autre commande.

### III.4. Conclusion :

Dans ce chapitre, nous avons appliqué la commande neuronale NARMA-L2 et la commande neuronale à modèle de référence des systèmes non-linéaires. Les réseaux de neurones représentent une approche très adaptée pour la commande de ces systèmes. La connaissance du modèle exacte du système n'est pas exigée, seule un ensemble de données est nécessaire pour l'apprentissage du modèle par le réseau de neurone. Parfois l'obtention de ces données est difficile et nécessite en moins un contrôleur stabilisant pour faire fonctionner le système et obtenir l'enregistrement de données.



**Conclusion**

**Générale**

# Conclusion générale

---

Les propriétés d'apprentissage et de généralisation des réseaux de neurones artificiels nous amènent à étudier l'apport des techniques neuronales dans le problème d'identification et de contrôle des systèmes dynamiques non linéaires. Les techniques classiques ont rencontré des difficultés à la fois en pratique et en théorie, et en raison d'une méthodologie rigoureuse comme la conception de modèles, les réseaux de neurones sont devenus un puissant outil de modélisation avec un large éventail d'applications. Ils permettent de créer des modèles statiques ou dynamiques précis de manière simple et efficace.

Arrivé au terme de ce mémoire et avant d'en évoquer les perspectives, nous résumons le contenu de notre travail comme suit :

Dans le premier chapitre nous avons donné une description générale sur l'approximateur universel RNA, leur origine biologique et les différents types existant dans la commande des processus, en particulier le perceptron multicouche. Ce dernier constitue des modèles de réseaux très efficaces grâce à leurs propriétés d'apprentissage tel que l'algorithme de retro-propagation

Le deuxième chapitre présente un aperçu général sur quelques techniques de commande des systèmes non linéaires, particulièrement la commande adaptative neuronale qui est utilisée dans ce travail. L'identification neuronale est utilisée pour simuler le comportement dynamique du système.

Dans le troisième chapitre, nous avons étudié et appliqué deux commandes neuronales, la première est la commande NARMA-L2 et la seconde est la commande à modèle de référence MRAC. Dans les deux cas, une étape d'identification est nécessaire, qui consiste à régler le réseau de neurones par l'apprentissage d'un modèle représentatif du système à contrôler.

La conception du contrôleur pour ces deux cas est différente. Le contrôleur dans le cas de commande NARMA-L2 est conçu directement à partir du modèle neuronal par contre dans le cas du MRAC, une deuxième étape d'identification est utilisée. La modélisation et la description de ce système a été donnée en détails. Les résultats de simulation sont illustrés pour confirmer l'efficacité de la méthode implémentée.

Les propriétés d'apprentissage et de généralisation des réseaux de neurones artificiels, nous conduisent à étudier l'apport des techniques neuronales dans les problèmes d'identification et de commande des systèmes dynamiques non linéaires pour lesquels les techniques classiques se heurtent à des difficultés d'ordre pratique et théorique, et grâce au développement des méthodologies rigoureuses pour la conception de modèles, les réseaux de neurones sont devenus des outils de modélisation puissants dont les domaines d'applications sont multiples.

## Bibliographie

- [1] SAIKI H, OULTAF K, « Reconnaissance de caractères à base des réseaux de neurones », Mémoire de Master, département d'électronique, UMMTO, 2013.
- [2] FENTAZI S, FERHATI S, « Modélisation des systèmes chaotiques à base des Réseaux de Neurones Artificiels : système de Lorenz, l'équation logistique », Université Mouloud Mammeri De Tizi-Ouzou, 2020.
- [3] Djeriri Y, « Les réseaux de neurones artificiels », Université Sidi Bel Abbas, 2017.
- [4] [https://fr.wikipedia.org/wiki/Mod%C3%A8les\\_du\\_neurone\\_biolgique#/media/Fichier:Neurone.svg](https://fr.wikipedia.org/wiki/Mod%C3%A8les_du_neurone_biolgique#/media/Fichier:Neurone.svg)
- [5] [https://www.researchgate.net/figure/Modele-de-base-dun-neurone-formel\\_fig3\\_328981633](https://www.researchgate.net/figure/Modele-de-base-dun-neurone-formel_fig3_328981633)
- [6] Oukacine N, « Utilisation des réseaux de neurones pour la reconstitution de défauts en évaluation non destructive », Mémoire magister, Option Entraînements Electriques, Université UMMTO, 2012.
- [7] HAROUZ A, « Reconstitution de défauts complexes avec la méthode des réseaux de neurones .Application pour l'évaluation Non Destructive », Mémoire de master, Spécialité Entraînements électriques, Université UMMTO, 2015.
- [8] TOUFIK T, « Analyse et Mémoire du Master Académique commandes d'un pendule inversé », Mémoire du Master Académique, Université Badji Mokhtar Annaba, 2020.
- [9] Krose B, Van Der Smagt P, «An introduction to neuronal networks », Eighth edition, The University of Amsterdam, November1996.
- [10] TIOURA A, ZEROUAL W, « Commande du système pendule –chariot par les techniques de l'espace d'état », Université Larbi Ben M'hidi Oum El Bouaghi, Master Informatique Industrielle, 2010.
- [11] GACEM S E, « Indentification des systèmes non linéaires par réseaux de neurones », Mémoire de Master, Université Mohamed Khider Biskra, 2015.

- [12] BOUTANA W, YKHELFOUNE N, « Etude comparative en simulation entre un régulateur PID et un régulateur flou », Mémoire Master, Spécialité Electronique, Université Mohammed Seddik BENYAHIA de Jijel, 2019.
- [13] JEAN-PIERRE, BABARY « Commande optimale des systèmes continus déterministes », Masson., 1985,279p
- [14] ISSAADI T, SAAD M, « Conception de la commande linéaire quadratique régulateur LQR 'Robustifiée' en utilisant l'optimisation H2/H $\infty$  (Appliquée à la commande d'un drone) », Master en commande des systèmes, Université UMMTO, 2014.
- [15] CHETTIBI I, ZENIFECHE B, « Commande adaptative par réseaux de neurones artificiels d'un système non linéaire incertain », Master en automatique, Université de Jijel, 2018.
- [16] Mendil B, « Cours de module Commande intelligente », Chapitre 05, Université Béjaia, 2023.
- [17] Lemita A, « Commande adaptative par structure neuronale récurrente pour la conduite d'un procédé de traitement d'eau usée », Mémoire de Magister en option contrôle, 2020.
- [18] Barkaoui M, « Etude et développement d'une commande neuronale adaptative », Université LAVAL, 1998.
- [19] MELAKHESSOU L, « Contrôle et identification des systèmes non linéaires par les techniques neuronales », Thèse Magister. Université El-Hadj Lakhdar- Batna.
- [20] Rebouli S, Hamoudi S, « Identification et contrôle de la machine asynchrone par réseaux de neurones flous », mémoire d'ingénieur, Institut d'électronique université Mohamed Boudiaf De M'Sila.
- [21] Kumpati S. Narendra, Snehasis Mukhopadhyay, "Adaptive Control Using Neural Networks and Approximate Models", IEEE Transactions on Neural Networks vol. 8, n° 1, May 1997
- [22] Kumpati S. Narendra, Kannan Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Transactions on Neural Networks, Vol. 1, n° 1, March 1990.
- [23] CHABOUR O, « Stabilisation des systèmes non linéaires », Thèse doctorat, Spécialité mathématique appliquées, Université de Metz, 31 Mars 2000.

**Résumé :**

Les méthodes de commande non linéaire se basent sur l'existence d'un modèle analytique du système. Cependant, pour des systèmes non linéaires complexes, le modèle peut être inexploitable, imprécis ou tout simplement inexistant. Les réseaux de neurones sont récemment apparus comme des approches capables de reproduire le comportement complexe de systèmes non linéaires. Dans cette étude, nous allons combiner la commande adaptative et les réseaux neuronaux pour profiter de leurs caractéristiques dans la résolution de problèmes de commande de systèmes non linéaires. Le premier chapitre introduit la généralité des réseaux de neurones. Le deuxième chapitre consiste à éclaircir les notions de base sur les commandes stabilisant les systèmes non linéaires. Ensuite, nous présentons un aperçu de la commande choisie dans ce travail, la commande neuronale adaptative. L'application de la commande par réseau de neurones à un système dynamique qui fait l'objet du chapitre 3. Les résultats de simulation montrent la robustesse et la fiabilité de la commande choisie sur la stabilité de système. Les résultats sont obtenus via Simulink/Matlab.

**Abstract:**

Non-linear control methods are based on the existence of an analytical model of the system. However, for complex nonlinear systems, the model may be unworkable, imprecise or simply non-existent. Neural networks have recently emerged as approaches capable of reproducing the complex behavior of nonlinear systems. In this study, we then combine adaptive control and neural networks to take advantage of their characteristics in solving nonlinear system control problems. The first chapter introduces the generality of neural networks. The second chapter clarifies the basic notions of control systems stabilizing nonlinear systems. Next, we present an overview of the control chosen in this work, adaptive neural control. The application of neural network control to a dynamic system which is the subject of Chapter 3. Simulation results show the robustness and reliability of the chosen control on system stability. The results are obtained via Simulink/Matlab.

**ملخص**

تعتمد طرق التحكم غير الخطية على وجود نموذج تحليلي للنظام. ومع ذلك، بالنسبة للأنظمة غير الخطية المعقدة، قد يكون النموذج غير قابل للتشغيل أو غير دقيق أو ببساطة غير موجود. ظهرت الشبكات العصبية مؤخرًا كمقاربات قادرة على إعادة إنتاج السلوك المعقد للأنظمة غير الخطية. في هذه الدراسة، نجمع بعد ذلك بين التحكم التكيفي والشبكات العصبية للاستفادة من خصائصها في حل مشاكل التحكم في الأنظمة غير الخطية. يقدم الفصل الأول عمومية الشبكات العصبية. يتألف الفصل الثاني من توضيح المفاهيم الأساسية للأوامر التي تعمل على تثبيت الأنظمة غير الخطية. بعد ذلك، نقدم نظرة عامة على عنصر التحكم المختار في هذا العمل، وهو التحكم العصبي التكيفي. تطبيق التحكم بالشبكة العصبية على نظام ديناميكي وهو موضوع الفصل 3. تظهر نتائج المحاكاة متانة وموثوقية التحكم المختار على استقرار النظام. يتم الحصول على النتائج عبر Simulink / Matlab.