MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE SETIF 1
ALGERIE

MEMOIRE

Présenté à la Faculté de technologie
Département d'Electronique
Pour l'obtention du diplôme de

MAGISTER

Option : contrôle

Par

MERZOUKA NOURESSADAT

Thème:

Etude des performances des réseaux de neurones dynamiques à représenter des systèmes réels : une approche dans l'espace d'état

Soutenu le : 04 / 07 / 2009 devant la commission d'examen:

Prof: BEKKA Rais El'hadi, Professeur, Univ.de Sétif	**Président**
Prof: CHIKOUCHE Djamel, Professeur, Univ. de M'sila	**Rapporteur**
Prof: MERZOUKI Abdelaziz, Professeur, Univ.de Sétif	**Examinateur**
Prof: BOUZIT Nacereddine, Professeur, Univ.de Sétif	**Examinateur**
Dr: AMARDJIA Noureddine, MC, Univ.de Sétif	**Examinateur**

Remerciements

Derrière ce travail se cache la présence, le support et l'ouverture d'esprit de mon encadreur, le Professeur Djamel CHIKOUCHE, Professeur à l'université de M'sila, que je dois énormément. Je tiens à le remercier pour le suivi continu de mon travail. Je lui suis très reconnaissant pour m'avoir sensibilisé de l'importance de la communication scientifique.

Je remercie vivement monsieur Rais El'hadi. BEKKA, Professeur à l'université de Sétif, qui me fait l'honneur de présider mon jury de thèse et juger mon travail, ainsi que pour sa contribution à ma formation.

J'adresse mes sincères remerciements aux Professeurs: MERZOUKI Abdelaziz, et BOUZIT Nacereddine, Professeurs à l'université de Sétif, et Dr AMARDJIA Noureddine, maîtres de conférences, à l'université de Sétif: pour avoir acceptés de participer à ce jury, malgré leurs nombreuses obligations, et pour leurs aides précieuses et leurs grandes générosités.

Je remercie très sincèrement monsieur Toufik BEKKOUCHE, qui m'a aidé et m'a encouragé.

Pour tout le personnel de d'institut spécialisé de la formation professionnelle, je leur dit merci du soutien moral qu'ils m'ont réservé.

Dédicace

À toute ma famille et mes amis…

Table des matières

Chapitre III. La surveillance par les réseaux de neurones

Chapitre IV. Application des réseaux de neurones dynamiques à la modélisation des systèmes non linéaires

Liste des figures

Chapitre I.

Chapitre II.

Chapitre III.

Chapitre IV.

Introduction générale

Grâce aux résultats obtenus au cours de la dernière décennie, les réseaux de neurones connaissent un succès croissant et ont prouvé leur efficacité dans plusieurs domaines: comme le traitement de signal, l'identification des paramètres, la commande des procédés, l'estimation et la détection des défauts …. Ils demeurent toutefois un sujet d'un grand intérêt pour les chercheurs qui désirent améliorer les performances de ces réseaux et étendre leur champ d'applications [1][2][3]. Le développement des nouvelles techniques et méthodologies expérimentales donne la possibilité d'aller toujours plus loin dans l'étude de l'architecture neuronale. Ainsi, des modèles toujours plus élaborés sont proposés pour comprendre les mécanismes fondamentaux de la dynamique neuronale d'une part, et pour proposer de nouvelles expériences ou applications d'autre part. En parallèle avec cette évolution il y a un besoin de développer des méthodes mathématiques d'analyse ou d'améliorer les méthodes existantes, tel que les problèmes théoriques qui posent la dynamique des réseaux de neurones.

De plus, si l'identification des systèmes linéaires est relativement bien maîtrisée, l'identification des systèmes non-linéaires reste un enjeu majeur. Actuellement, de nombreuses méthodes ont été proposées afin d'utiliser les réseaux de neurones dans le cadre de l'identification des systèmes dynamiques. Toutefois, ces méthodes reposent essentiellement sur une modélisation de type "boîte noire", qui font l'hypothèse que le processus à modéliser peut être décrit par des modèles entrée-sortie ; ce qui leur vaut d'avoir une complexité de calcul et de mise en œuvre.

L'objectif de ce travail de magister consiste en un premier lieu à introduire les notions fondamentales des réseaux de neurones. Puis, nous allons examiner les avantages de la modélisation boite noir de systèmes non linéaires dynamiques en utilisant des réseaux de neurones décrits dans l'espace d'état. Des exemples de systèmes dynamiques non linéaires de test seront utilisés à cette fin ; alors que la seconde application concerne l'analyse des signaux vibratoires issus d'un système d'engrenage afin de détecter la

présence de défauts. Les problèmes de la mise en œuvre et la complexité de calcul de la méthode seront investis.

Une étude comparative de performances de cette technique avec les techniques conventionnelles est également envisagée, en faisant appel à des systèmes physiques.

Le mémoire est organisé autour de quatre chapitres. Les notions fondamentales des réseaux de neurones et leurs propriétés font l'objet du premier chapitre. Le deuxième chapitre qui constitue le cœur de ce mémoire, comprend deux parties. La première partie présente la modélisation dynamique des systèmes non linéaires, où nous avons déterminé les structures des prédicteurs neuronaux associés aux modèles-hypothèses donnés. La seconde partie présente l'identification des systèmes non linéaires par les réseaux de neurones dynamiques dans l'espace d'état. Le troisième chapitre est subdivisé à son tour en deux parties. La surveillance des systèmes dynamiques par les réseaux de neurones est présentée dans la première partie, alors que la deuxième partie aborde l'analyse vibratoire des engrenages où nous présenterons les différents outils de traitement du signal utilisé pour construire le vecteur forme des paramètres de la base des données effectuées sur le système. Deux applications de cette technique ont été présentées au quatrième chapitre, la première application concerne une étude d'un processus simulé et la seconde application une étude d'un processus réel, qui consiste à appliquer cette méthode pour l'analyse et le diagnostic de signaux de vibration d'engrenage.

Nous terminons notre mémoire par une conclusion générale, où nous rappellerons les principaux résultats obtenus dans cette étude et nous donnerons quelques perspectives.

Chapitre I

fondements des réseaux de neurones

Chapitre 1

Fondement des Réseaux de Neurones

I.1 Introduction

Les réseaux de neurones artificiels constituent l'une des approches d'intelligence artificielle dont le développement se fait à travers les méthodes par lesquelles l'homme essaye toujours d'imiter la nature et de reproduire des modes de raisonnement et de comportement qui lui sont propre. Nous présentons donc dans ce premier chapitre un état de l'art de ces réseaux de neurones. D'abord, nous rappellerons la définition et les propriétés des réseaux de neurones, avant de décrire les architectures neuronales les plus utilisées, à savoir les modèles statiques et dynamiques. Nous poursuivrons en exposant les différents types d'apprentissage, ainsi que les domaines d'applications.

I.2 Neurone et le réseau de neurones

La figure I.1 montre un schéma comportant la structure générale d'un neurone artificiel.

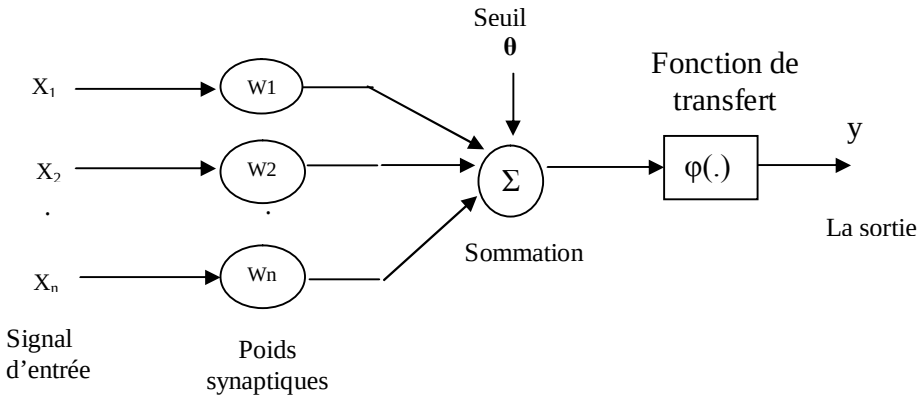


Figure I.1 : Neurone artificiel.

Un neurone artificiel est considéré comme un élément élémentaire de traitement de l'information, Il reçoit les entrées et produit un résultat à la sortie.

$$u = \sum_{j}^{n} w_j x_j + \theta = W' X + \theta \qquad \textbf{(1.1)}$$

$$y = \varphi(u) \qquad \textbf{(1.2)}$$

x1, x2, ..., xn ; sont les entrées externes. y est la sortie. w1, w2, ..., wn sont les poids associés à chaque connexion. X est le vecteur d'entrée, W' est le vecteur poids, θ est appelé le biais.

La fonction φ est appelée fonction d'activation, c'est une fonction non linéaire. Différentes fonctions d'activation peuvent être utilisées, parmi lesquelles on peut citer : *fonction signe, sigmoïde, tangente hyperbolique, Gaussienne [***4],[5]**.et le choix d'un type de fonction dépend de l'application.

Les réseaux de neurones artificiels sont des combinaisons de fonctions élémentaires appelées neurones formels, ou simplement neurones associés en couches et fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau.

I.3 Propriétés des réseaux de neurones

Les réseaux de neurones artificiels possèdent une propriété fondamentale qui justifient l'intérêt croissant qui leur est accordé et que sont capable d'intervenir dans des domaines très divers**,** et qui les distingue des techniques classiques de traitement des données.

- Les réseaux de neurones sont des approximateurs universels : Cette propriété peut être énoncée comme suit: Toute fonction bornée suffisamment régulière peut être approchée uniformément, avec bonne précision, dans un domaine fini de l'espace de ses variables, par un réseau de neurones qui comporte une couche de neurones cachée en nombre fini, possédant tous la même fonction d'activation et un neurone de sortie linéaire **[6] -[10].**

-Parcimonie : Lors de la modélisation d'un processus à partir de ses données, on cherche toujours à obtenir les résultats les plus satisfaisants possibles avec un nombre minimum de paramètres. On dit que l'on cherche l'approximation la plus parcimonieuse. Pour obtenir un modèle non linéaire de précision donnée, un RN a besoin de moins de paramètres ajustables que les méthodes de régression classiques (par exemple la régression polynomiale). Or le nombre de données nécessaires pour ajuster le modèle est directement lié au nombre de ses paramètres **[7]-[11]** .

I.4 Architecture de réseaux de neurones

Selon la topologie de connexion des neurones, on peut les classer en deux grandes catégories: réseaux non bouclés (statique ou feed forward) et réseaux bouclés (dynamique, feed back ou récurrent).

I.4.1 Réseaux statiques "feed-forward "

Un réseau de neurones non bouclé (appelé aussi statique) est représenté comme un graphe dont les noeuds sont les neurones. L'information circule des entrées vers les sorties sans retour en arrière (Figure I.2). Ce type de réseaux est utilisé pour effectuer des taches d'approximation de fonction non linéaire, de la classification ou de la modélisation de processus statiques non linéaires **[12].**

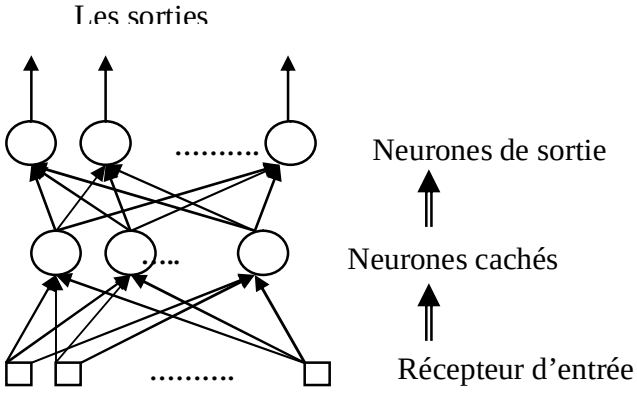


Figure: I.2 : Exemple d'un réseau de neurones non bouclé**.**

I.4.1.1 Réseaux multicouches (ou Perceptron Multi Couche PMC)

C'est le réseau de neurones statique le plus utilisé. Les neurones sont arrangés par couche. Les neurones de la première couche reçoivent le vecteur d'entrée, ils calculent leurs sorties qui sont transmises aux neurones de la seconde couche qui calculent eux même leurs sorties et ainsi de suite de couche en couche jusqu'à celle de sortie. Chaque neurone dans la couche cachée est connecté à tous les neurones de la couche précédente et de la couche suivante, et il n'y a pas de connexions entre les cellules d'une même couche.

Il peut résoudre des problèmes non linéairement séparables et il suit un apprentissage supervisé avec la règle de correction de l'erreur.

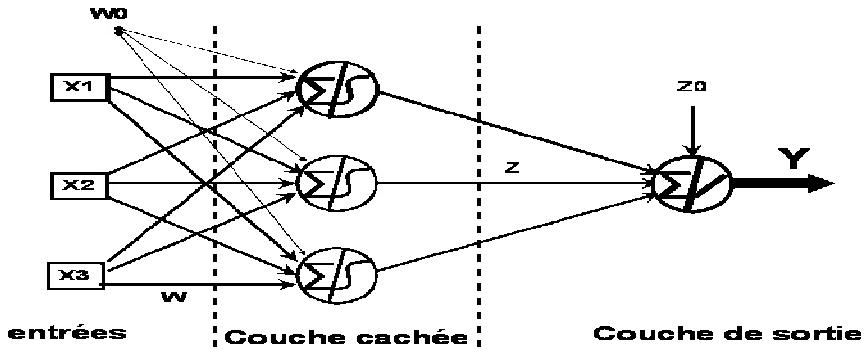


Figure I.3 : Perceptron Multi Couche PMC.

La sortie du réseau a pour expression :

$$Y = g\left[Z\left[f\left(WX + W_0\right)\right] + Z_0\right] \qquad \textbf{(1.3)}$$

Avec :

f et g les fonctions de transfert, des couches cachées et de sorties respectivement,

Y sortie du réseau, X vecteur des entrées. W : matrice des poids de connexions liant la couche d'entrée à la couche cachée. W_0 : vecteur des biais des cellules de la couche cachée. Z : matrice des poids des connexions liant la couche cachée à la couche de sortie. Z_0 : vecteur des biais des cellules de la couche de sortie.

I.4.2 Réseau récurrents "Feed-back"

Un réseau bouclé (récurrent), régi par une ou plusieurs équations différentielles, résulte de la composition des fonctions réalisées par chacun des neurones et des retards associés à chacune des connexions. Ces réseaux sont utilisés pour effectuer des taches de modélisation des systèmes dynamiques, de commande de processus ou de filtrage **[12]-[18].**

Le comportement dynamique d'un réseau de neurones bouclé peut être décrit par une représentation d'état paramétrée par les coefficients C, représentée sur la figure I.4.

$$\begin{cases} \mathbf{S(k+1) = \psi\ (\ S(k), I(k); C\ )} \\ \mathbf{Y(k) = \xi(\ S(k), I(k); C\ )} \end{cases} \qquad \mathbf{(1.4)}$$

Où I(k) est le vecteur des entrées externes, S(k) le vecteur des variables d'état, Y(k) le vecteur des sorties, ψ (., .; C) et ξ (., .; C) représentent les fonctions réalisées par le réseau de neurones statiques de la forme canonique interconnectés avec les coefficients C **[14]** .

Nerand a démontré dans ses articles **[12] [15]** que : tout réseau de neurones bouclé aussi complexe soit-il peut être mis sous une forme canonique comportant un réseau de neurones non bouclé dont certaines sorties (les variables d'état) sont ramenées aux entrées par des bouclages de retard unité .

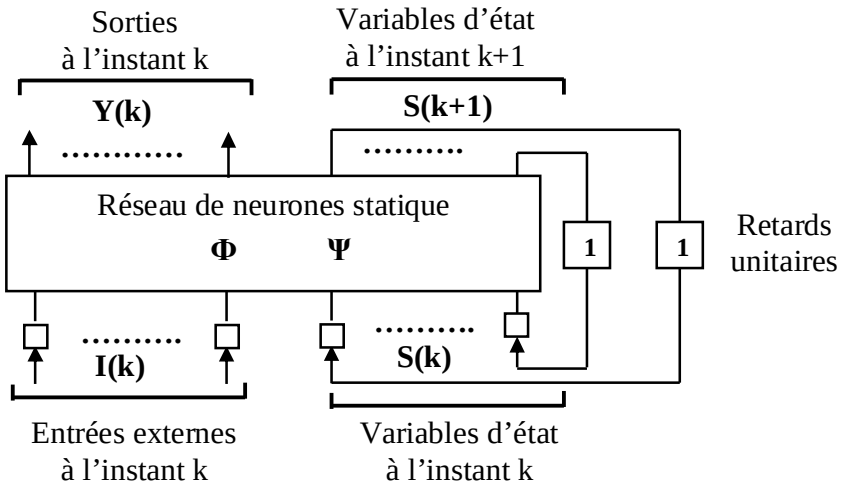


Figure I.4 : Forme canonique d'un réseau de neurones bouclé.

I.5 L'apprentissage de réseaux de neurones

L'apprentissage est la caractéristique principale des réseaux de neurones ; c'est le processus d'adaptation des paramètres d'un système pour remplir au mieux la tache pour la quelle le réseau est destiné. Le type d'apprentissage est déterminé par la manière dont les paramètres sont adaptés, et il existe plusieurs méthodes et algorithmes pour adapter ces paramètres **[3].**

- Apprentissage supervisé (supervised learning)

Pour les réseaux à apprentissage supervisé, on présente au réseau des entrées, et au même temps les sorties désirées pour cette entrée. Le réseau doit ajuster ses poids de façon à réduire l'écart entre la réponse désirée et la sortie du réseau. Cette procédure est répétée jusqu'à ce qu'un critère de performance soit satisfait. L'algorithme le plus utilisé est celui de la rétro propagation de l'erreur.

- Apprentissage non supervisé (unsupervised learning)

Pour les réseaux à apprentissage non supervisé, aucune information sur la réponse désirée n'est fournie au réseau **[1].** On présente une entrée au réseau et on le laisse évoluer librement jusqu'à ce qu'il se stabilise. Ce comportement est connu sous le nom "auto organisation."

I.5.1.Apprentissage des réseaux de neurones non bouclé

Dans le cas d'une application à base d'un réseau non bouclé, la première étape consiste à choisir le type de réseau ; si le réseau est un perceptron multicouche par exemple, il faudra aussi choisir le nombre de neurones d'entrée, de sortie et la couche cachée ; la fonction réalisée par le réseau à l'instant discret k s'écrit :

$$y_K = \sum_{i=1}^{Ncc} C_i \psi \left(\sum_{J=1}^{Ni} C_{i,j} x_K^J \right) + b \qquad \textbf{(1.5)}$$

Où N_{cc} représente le nombre de neurones de la couche cachée, C_i est le paramètre de connexion du i-ème neurone de la couche cachée vers la sortie, $C_{i,j}$ est le paramètre associé à la connexion de la j-ième entrée (parmi les Ni) vers le i-ème neurone de la couche cachée. ψ est la fonction réalisée par les neurones de la couche cachée, où b est le biais.

L'apprentissage du réseau consiste à trouver, parmi toutes ces fonctions, celle qui minimise une fonction de coût généralement quadratique :

$$J(\theta) = \frac{1}{2} \sum_{k=1}^{N} \left(y_p^k - y^k \right)^2 \qquad \textbf{(1.6)}$$

Où Y_p^K est la sortie mesurée du processus à l'instant k, Y^K est la sortie du réseau de neurones à l'instant k, le vecteur θ est les paramètres du réseau.

Comme la fonction réalisée par le réseau de neurones est non linéaire, il est nécessaire de mettre en œuvre des algorithmes basés sur le calcul itération du gradient de la fonction de coût. L'algorithme de rétropropagation est largement utilisé pour calculer le gradient de cette fonction de coût. (Voir Annexe).

Ce gradient s'écrit :

$$\frac{\partial J}{\partial \theta} = -\sum_{k=1}^{N} \left(y_p^k - y^k \right) \frac{\partial y_k}{\partial \theta} \qquad \textbf{(1.7)}$$

Pour calculer chacune des composantes du gradient de la fonction de coût, on est ramené au calcul de la dérivée de la sortie du réseau par rapport au paramètre considéré :

- le biais de sortie b :

$$\frac{\partial y_k}{\partial b} = 1$$

- les paramètres de connexions couche cachée / sortie C_i :

$$\frac{\partial y_k}{\partial C_i} = \psi \left(\sum_{j=1}^{Ni} C_{i,j} . X_k^j \right) \quad \text{pour i=1……Ncc} \qquad \textbf{(1.8)}$$

-les paramètres de connexions couche d'entrée / couche cachée $C_{i,j}$:

$$\frac{\partial y_k}{\partial C_{i,j}} = C_i X_k^i \psi' \left(\sum_{j=1}^{Ni} C_{i,j} . X_k^j \right) \qquad \textbf{(1.9)}$$

Après avoir calculé chacune des composantes du vecteur gradient, il est possible de mettre en œuvre des algorithmes d'optimisation itératifs, tels que celui de BFGS ou de Levenberg-Marquardt.

I.5.2 Apprentissage des réseaux bouclés

Nous avons vu que tout réseau de neurones bouclé peut être mis sous une forme canonique constituée d'un réseau non bouclé dont les entrées sont les entrées externes et les variables d'état à l'instant *k*, et dont les sorties sont les sorties du réseau et les variables d'état à l'instant *k*+1. Cette mise en forme d'un réseau bouclé facilite le calcul du gradient de la fonction de coût, et ramène l'apprentissage d'un réseau bouclé à celui d'un réseau non bouclé, comme nous le verrons **[07][09][12][15].**

La fonction réalisée par un réseau de ce type peut s'écrire :

$$Y_K = \sum_{i=1}^{Ncc} C_i \psi \left(\sum_{J=1}^{N_I + Ns} C_{i,j} x_K^J \right) + b \qquad \textbf{(1.10)}$$

Où *Ncc* est le nombre de neurones cachée, C_i est le paramètre associé à la connexion du *i*-ème neurone de la couche cachée vers la sortie, ψ est la fonction d'activation réalisée par les neurones de la couche cachée, *b* est le biais, *Ci,j* est le paramètre associé à la connexion de la *j*-ième entrée vers le $i^{\text{-ème}}$ neurone de la couche cachée. N_I est le nombre d'entrées exogènes, N_s est le nombre d'entrées d'états. Le caractère récurrent du réseau se traduit par le fait que le vecteur des entrées du modèle $X_{k=}$ [$X_k^{1,}$ X_k^2 …… X_k^{Ne}] où $Ne=N_I+Ns$ comprend les N_s sorties d'état à l'instant précédent.

Deux situations sont possibles :

• **les variables d'état sont mesurables** : Dans ce cas la, il s'agit d'un apprentissage d'un réseau à plusieurs sorties, et que chaque grandeur qu'il s'agisse d'une sortie du modèle ou d'une variable d'état, doit être prise en considération dans la fonction de coût, qui est la somme des fonctions de coût des moindres carrés associées à ces différentes grandeurs, et qui s'écrit :

$$J(C,i) = \frac{1}{2} \sum_{K=1}^{N} \left(y_K^p - y_k(\theta) \right)^2 + \frac{1}{2} \sum_{m=Ni+1}^{Ni+Ns} \sum_{K=1}^{N} \left(x_K^{m,p} - x_K^m(\theta) \right)^2 \qquad \textbf{(1.11)}$$

où $x_k^{m,p}$ est la mesure de la $m^{\text{-ième}}$ variable d'état à l'instant *k*, dont l'estimation est x_k^m. Le vecteur θ représente l'ensemble des paramètres du réseau.

• **les variables d'état sont non mesurables** : Dans ce cas on laisse les variables d'état évoluer librement sans leur imposer de valeurs désirées. Seule l'initialisation de ces variables d'état, en début d'apprentissage, a une influence sur les valeurs que prendront ces variables d'état aux instants suivants. On est alors ramené au problème de l'apprentissage d'un réseau récurrent à une seule sortie.

La méthode d'apprentissage d'un modèle neuronal récurrent dépend de l'hypothèse concernant la manière selon laquelle le bruit affecte le processus, et du type de prédicteur que l'on souhaite obtenir, (comme nous allons le voir au prochain chapitre).

La fonction de coût est minimisée par une méthode itérative utilisant la valeur de la fonction de coût et celle de son gradient, cette dernière étant calculée par l'algorithme de rétropropagation. Les méthodes quasi-Newtoniennes à pas asservi sont particulièrement efficaces dans ce cadre **[14][09]**. Nous présenterons en détail ce calcul dans le cas récurrent au prochain paragraphe.

On distingue deux types d'apprentissage:

***I.5.2.1 Apprentissage dirigé (teacher-forced)*:** L'algorithme est dit dirigé, si le réseau est entièrement dirigé par les entrées externes, Ce qui signifie que les variables d'états du réseau sont fixées à leurs valeurs mesurées pour toute la séquence d'apprentissage, quelles que soient leurs valeurs données par le modèle. C'est un réseau non bouclé. Il peut être utilisé pour un réseau récurrent décrit par une représentation entrée-sortie, pour lequel les variables d'état sont les sorties retardées **[7]**.

Le vecteur des entrées du modèle est constitué des Ni entrées imposées et de Ns entrées d'états $x_k = \left[x_k^1 .... x_k^{Ne}, x_k^{1,p} ..... x_k^{Ns,p}\right]^T$ **(1.12)**

Notons que, dans le cas d'une représentation entrée-sortie, le vecteur des variables du réseau fait intervenir les mesures retardées de la sortie ; il s'écrit alors :

$$x_k = \left[x_k^1 .... x_k^{Ne}, y_{k-1}^p ..... y_{k-m}^p\right]^T \quad \textbf{(1.13)}$$

Une fois la séquence d'apprentissage est construite, la mise en œuvre d'algorithmes d'optimisation itératifs se fait de la même manière que pour le réseau statique (bien sur en ce qui concerne le calcul du gradient de la fonction de coût).

I.5.2.2 Apprentissage semi-dirigé : Dans ce cas, l'état du réseau ne lui est imposé qu'au début de l'apprentissage. Le caractère récurrent du réseau de neurones est pris en considération durant la phase d'apprentissage. Au lieu de fixer les variables d'état du réseau à leurs valeurs mesurées, on leur donne les valeurs estimées par le modèle à l'instant précédent **[9]**. Du fait les algorithmes d'optimisation utilisés sont nécessairement itératifs. Pour simplifier l'écriture du calcul de la fonction de coût et de son gradient, Ils ont introduit la notion de copie.

I.5.2.2.1 Représentation modulaire d'un réseau d'apprentissage et notion de copie

Le réseau récurrent est remplacé par une série de N réseaux non bouclés de la forme canonique, appelés copies, et qui interviennent N fois en cascade avec les coefficients disponibles à l'itération i, ayant chacune les mêmes paramètres, et connectés entre eux via les entrées et sorties d'état **[09].**

Seules les valeurs des entrées et des sorties des neurones sont différentes d'une copie à l'autre. La copie numéro k est représentée sur la figure I.5.

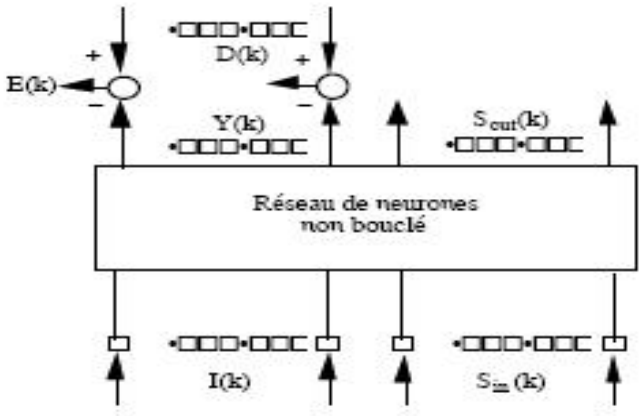


Figure I.5: Copie n°k d'un réseau d'apprentissage.

L'apprentissage s'effectue toujours sur un réseau non bouclé, que l'on appelle réseau d'apprentissage:

-Pour un prédicteur non récursif, il s'agit d'un réseau non bouclé constitué de N copies en cascade.

-Pour un prédicteur récursif, le réseau d'apprentissage est le réseau non bouclé constitué de N copies indépendantes, et l'apprentissage est identique à celui d'un modèle statique.

Dans le cas général de l'apprentissage:

Les entrées de la copie n° k (k =1 à N) sont **[09] :**

- les entées imposées I(k) (constitué des entrées externes u(k),….u(k-m+1) du prédicteur, et éventuellement des sorties mesurées $y_p(k)…y_p(k-n+1)$).

- Les entrées d'état S_{in}^{k}, ou les entrées bouclées du prédicteur: ce sont les sorties passées O (k)…O (k-n+1) et les composantes passées de l'état x (k),…x (n+1).

Les sorties de la copie n° k sont :

- les sorites O^k auxquelles sont associés les valeurs désirées $O_{dés}{}^{k}$;

  On note de plus $E^k = O_{dés}{}^{k} - O^{k.}$ **(1.14)**

  l'erreur de la copie n°k.

  - les sorties d'état $S_{out}{}^{k}$, ou sorties bouclées de prédicteur ; elles vérifient par conséquent : $S_{in}{}^{k+1} = S_{out}{}^{k}$ **(1.15)**

Remarque

Dans le cas d'un prédicteur non récursif il n''y a pas d'entrée d'état S^k_{in} ni de sortie S^k_{out}

Les réseaux d'apprentissages d'un prédicteur non récursif et un prédicteur récursif sont représentés sur la figure I.6.

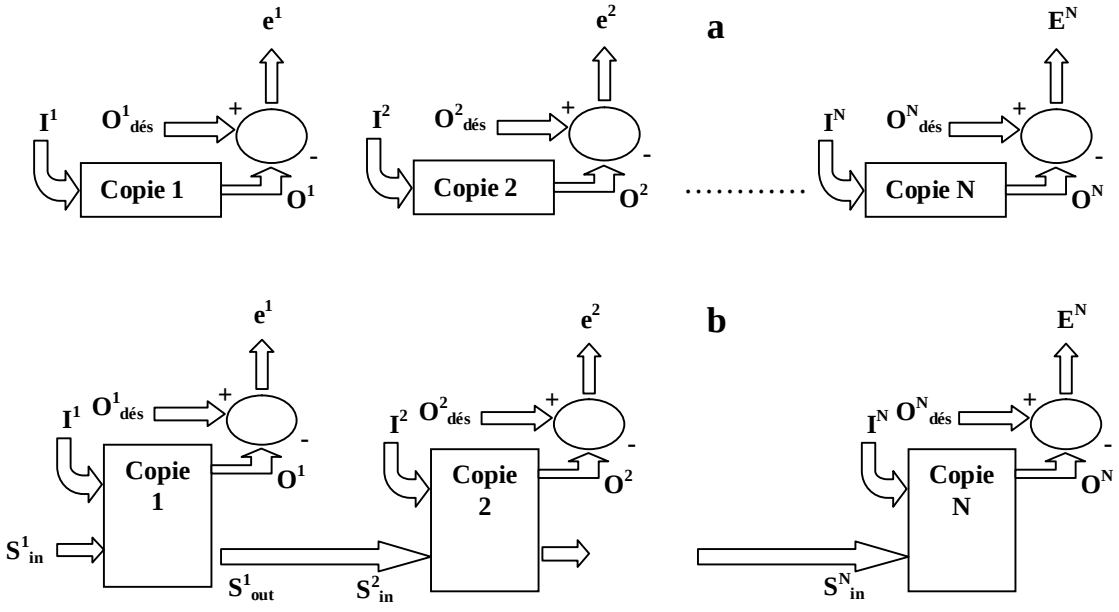


Figure I.6 : Les réseaux d'apprentissage : a) d'un prédicteur non récursif (algorithme dirigé) b) d'un prédicteur récursif (algorithme non dirigé) **[9]**.

I.5.2.2.2 Numérotation des nœuds d'une copie

Les entrées de la copie sont les entées imposées et les entrées d'état, les sorties de la copie sont les NS cellules de sorties connectées aux NO neurones de sorties et les sorties d'état qui sont les NS cellules connectées aux NS neurones d'état**.[09][12][19].**

Chaque copie est composée de :

- NI cellules d'entrées imposées (numérotées de 1 à NI)
- NS cellules d'entrées d'état (numérotées de NI+1 à NI+NS)
- NC neurones cachés (ordonnés de NI+NS +1 à NI+NS+ NC)
- NO neurones sortie (ordonnés de NI+NS +NC+1 à NI+NS +NC+ NO)
- NO cellules de sortie (numérotées de NI+NS +NC+ NO+1àNI+NS +NC+ NO+ NO)
- NS cellules de sorties d'état (numérotées de NI+NS +NC+ NO+NO+1 à NI+NS +NC+ NO+ NO+NS).

Nous désignerons aussi par x_k^n la sortie du neurone indexé par n (variant de 1 à Ni+2Ns+Nc), dans la copie k, c-à-d à l'instant discret k. De plus, l'activité des neurones de la couche cachée sera notée : $v_k^n = \sum_{n=1}^{Ni+Ns} C_{n,i} x_k^i$ (pour n variant de Ni + Ns + 1 à Ni + Ns + Nc), de telle sorte que $x_k^n = \psi\left(v_k^n\right)$.

Rappelons que la fonction de coût s'écrit :

$$J(\theta) = \frac{1}{2}\sum_{n=1}^{N}\left(y_k^p - y_k(\theta)\right)^2 = \frac{1}{2}\sum_{n=1}^{N}(e_k)^2$$
$$= \frac{1}{2}\sum_{n=1}^{N}\left(y_k^p - x_k^{Ni+Ns+Nc+1}(\theta)\right)^2 \qquad \textbf{(1.16)}$$

Le calcul du gradient de cette fonction de coût dans le cas récurrent peut se calculer par rétropropagation, en deux étapes : Dans un premier temps, on calcule la dérivée de la fonction de coût par rapport aux sorties et entrées d'état du réseau. Puis, connaissant ces quantités, on calcule sa dérivée par rapport aux paramètres du réseau. Bien que dans chacune des copies du dépliement temporel, le vecteur θ ait la même valeur, il est impératif de distinguer :

• le gradient de la fonction de coût par rapport à ce vecteur θ sur l'ensemble des copies,

• le gradient de la fonction de coût par rapport à θ(n) dans une copie n donnée. En effet, pour deux copies d'indice n et n' différents, les vecteurs $\frac{\partial J}{\partial \theta^{(n)}}$ et $\frac{\partial J}{\partial \theta^{(n')}}$ ne prennent pas la même valeur, bien que θ(n) et θ(n') soient égaux. Le gradient total peut ainsi s'écrire : $\frac{\partial J}{\partial \theta} = \sum_{n=1}^{N} \frac{\partial J}{\partial \theta^{(n)}}$ **(1.17)**

I.5.2.2.3 Calcul de la dérivée de la fonction de coût par rapport aux sorties et entrées d'état

Pour effectuer ce calcul, il convient de distinguer trois cas, suivant que l'on s'intéresse à la dernière copie du dépliement temporel, à la première, ou à une copie intermédiaire.

Pour la copie N :

Pour la sortie (k = Ni+Ns+Nc+1):

$$\frac{\partial J}{\partial x_N^k} = \frac{\partial J}{\partial y_N} = -\left(y_N^p - y_N(\theta)\right) = -e_N \qquad \textbf{(1.18)}$$

Pour les autres sorties d'état (k = Ni+Ns+Nc+2, …, Ni+2Ns+Nc):

$$\frac{\partial J}{\partial x_N^k} = 0 \qquad \textbf{(1.19)}$$

Pour les entrées d'état (k = Ni+1, …, Ni+Ns):

$$\frac{\partial J}{\partial x_N^k} = \frac{\partial J}{\partial y_N}\frac{\partial y_N}{\partial x_N^k} = -e_N \sum_{j=Ni+Ns+1}^{Ni+Ns+Nc} (C_{\alpha,j}\phi'(v_N^j)C_{j,k} \qquad \textbf{(1.20)}$$

Où α = Ni+Ns+Nc+1 et où Φ' est la dérivée de la fonction d'activation des neurones de la couche cachée.

Pour les copies intermédiaires (n = N-1,…,2) :

Pour la sortie (k = Ni+Ns+Nc+1):

$$\frac{\partial J}{\partial x_n^k} = \frac{\partial J}{\partial y_n} = -e_n + \frac{\partial J}{\partial x_{n+1}^{k-Ns-Nc}} \qquad \textbf{(1.21)}$$

Pour les autres sorties d'état (k = Ni+Ns+Nc+2, …, Ni+2Ns+Nc):

$$\frac{\partial J}{\partial x_N^k} = \frac{\partial J}{\partial x_{n+1}^{k-Ns-Nc}} \qquad \textbf{(1.22)}$$

Pour les entrées d'état (k = Ni+1, …, Ni+Ns):

$$\frac{\partial J}{\partial x_n^k} = \sum_{i=Ni+Ns+Nc+1}^{Ni+2Ns+Nc} \frac{\partial J}{\partial x_n^i}\frac{\partial x_n^i}{\partial x_n^k} \qquad \textbf{(1.23)}$$

$$= \sum_{j=Ni+Ns+1}^{Ni+2Ns+Nc} \phi'(v_n^j) C_{j,k} \left[\sum_{i=Ni+Ns+Nc+1}^{Ni+2Ns+Nc} \frac{\partial J}{\partial x_n^i} C_{i,j} \right]$$

Pour la copie N=1 :

Pour la sortie (k = Ni+Ns+Nc+1):

$$\frac{\partial J}{\partial x_1^k} = -e_1 + \frac{\partial J}{\partial x_2^{k-Ns-Nc}} \qquad \textbf{(1.24)}$$

Pour les autres sorties d'état (k = Ni+Ns+Nc+2, …, Ni+2Ns+Nc):

$$\frac{\partial J}{\partial x_N^k} = \frac{\partial J}{\partial x_2^{k-Ns-Nc}} \qquad \textbf{(1.25)}$$

Pour les entrées d'état (k = Ni+1, …, Ni+Ns):

Le calcul de $\frac{\partial J}{\partial x_1^k}$ est inutile

I.5.2.2.4 Calcul de la dérivée de la fonction de coût par rapport aux paramètres du réseau

La dérivée de la fonction de coût par rapport à chacun des paramètres du réseau s'exprime en distinguant le rôle de ces paramètres.

Paramètres qui pondèrent les sorties des neurones cachés dans le calcul des potentiels des neurones de sortie du modèle :

Il s'agit donc des paramètres $C_{k,j}$ pour des valeurs de

$k = Ni + Ns + Nc + 1 \ldots, Ni + 2Ns + Nc$ et $j = Ni + Ns + 1 \ldots Ni + Ns + Nc$ on peut écrire :

$$\frac{\partial J}{\partial C_{k,j}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k}\frac{\partial x_n^k}{\partial C_{k,j}^{(n)}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} x_n^j \qquad \textbf{(1.26)}$$

Paramètres qui pondèrent les variables (exogènes ou d'état) dans le calcul des potentiels des neurones cachés :

Il s'agit donc des paramètres Ck , j pour $k = Ni + Ns + 1 \ldots, Ni + 2Ns + Nc$ et $j = 1, \ldots, Ni + Ns$ on peut écrire :

$$\frac{\partial J}{\partial C_{k,j}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} \frac{\partial x_n^k}{\partial C_{k,j}^{(n)}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} \phi'(v_n^k) x_n^j$$

$$= \sum_{n=1}^{N} \phi'(v_n^k) x_n^j \left[\sum_{i=Ni+Ns+Nc+1}^{Ni+2Ns+Nc} \frac{\partial J}{\partial x_n^i} C_{i,k}^{(n)} \right] \qquad \textbf{(1.27)}$$

Pour les biais de la couche de sortie :

Il s'agit des paramètres Ck,1 pour $k = Ni + Ns + Nc + 1, \ldots, Ni + 2Ns + Nc$ on peut écrire :

$$\frac{\partial J}{\partial C_{k,1}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} \frac{\partial x_n^k}{\partial C_{k,1}^{(n)}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} x_n^1 = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} \qquad \textbf{(1.28)}$$

I.5.2.2.5 Algorithmes d'apprentissage

La fonction de coût peut être minimisée avec tout algorithme reposant sur le calcul du gradient, quasi-newtonien, ou autre. Le gradient est calculé avec l'algorithme de rétropropagation. On procède donc de la même manière que pour un réseau non bouclé, à ceci près que les paramètres des différentes copies doivent être identiques (poids partagés). L'apprentissage d'un réseau de neurones récurrent s'effectue alors de la façon suivante :

1. initialisation des paramètres du réseau
2. propagation à travers le réseau déplié temporellement
3. calcul du gradient de la fonction de coût (en tenant compte des poids partagés)
4. mise à jour des paramètres du réseau, puis retour à l'étape 2 pour une nouvelle itération.

I.6 Domaines d'application

Les grands domaines d'application des réseaux de neurones découlent naturellement des propriétés énoncées précédemment. Nous présentons dans les sections suivantes quelques exemples pour montrer le vaste étendu de leur applicabilité.

I.6.1 La régression non linéaire, ou modélisation de données statiques

Une immense variété de phénomènes statiques peut être caractérisée par une relation déterministe entre des causes et des effets ; les réseaux de neurones sont de bons candidats pour modéliser de telles relations à partir d'observations expérimentales, sous réserve que celles-ci soient suffisamment nombreuses et représentatives [08].

I.6.2 La modélisation de processus dynamiques non linéaires

Modéliser un processus consiste à déterminer un ensemble d'équations mathématiques qui décrivent le comportement dynamique du processus, c'est-à-dire l'évolution de ses sorties en fonction de l'évolution de ses entrées [09]. Ce problème peut être avantageusement résolu par un réseau de neurones, si le phénomène que l'on désire modéliser est non linéaire .

I.6.3 La commande de processus

La commande d'un processus consiste à concevoir un système comprenant un organe qui calcule la commande à appliquer au processus pour assurer un comportement dynamique spécifié par des cahiers de charges: régulation au voisinage d'un point de fonctionnement, poursuite d'une trajectoire de consigne, commande optimale…. L'ensemble commande / rocessus peut donc être considéré comme un système qui réalise une fonction (non linéaire) qu'un réseau de neurone peut approcher **[7] [9][14].**

I.6.4 La classification

Une autre grande catégorie de problème industriel consiste à attribuer de façon automatique un objet à une classe, parmi d'autres classes possibles. Et en raison de leur propriété d'approximateurs universels, les réseaux de neurones sont capables d'estimer de manière précise la probabilité d'appartenance d'un objet inconnu à une classe parmi plusieurs possibles **[10].**

Citons également les applications en diagnostic de panne, contrôle de qualité, analyse du signal, élimination du bruit, reconnaissance de formes (bruits, images, paroles etc.)
Dans le cadre de notre travail, nous nous somme intéressés à un champ d'application particulier: le diagnostic de pannes.

I.7 Conclusion

Dans ce chapitre, nous avons présenté le contexte de notre travail où nous avons vu un aperçu général sur les réseaux de neurones. Nous avons présenté les propriétés fondamentales des réseaux de neurones qui justifient l'intérêt croissant qui leur est accordé et que sont capable d'intervenir dans la résolution de nombreux problèmes de la modélisation ainsi que la classification.
Pour obtenir des bons résultats, la mise en œuvre des réseaux de neurones nécessite quelques précautions :

- Les réseaux de neurones sont des approximateurs parcimonieux des fonctions non linéaires, il faut donc s'assurer que le modèle est non linéaire. Pour un modèle linéaire, la mise en œuvre d'un modèle polynomial par exemple plus simple est moins couteuse en temps de calcul que les réseaux de neurones.
- Une séquence d'apprentissage, de test et de validation suffisamment riche (disposer des données suffisamment nombreuses et représentatives ;déterminer les paramètres qui ont une influence significative sur le problème à résoudre) ; et un algorithme d'apprentissage performant.

Dans le prochain chapitre, nous allons aborder la modélisation des systèmes non linéaires.

Chapitre II

Modélisation dynamique des systèmes non linéaires

Chapitre II

Modélisation dynamique des systèmes non linéaires

II.1Introduction

La modélisation d'un processus consiste à trouver une description mathématique de son fonctionnement, qui permet de rendre compte les relations existant entre ses entrées et ses sorties, et qui sont représentées par des équations. Si ces équations sont algébriques, le modèle est dit statique. Si ces équations sont des équations différentielles ou des équations aux différences récurrentes, le modèle est dit dynamique, respectivement à temps continu ou à temps discret **[19] [20].** Nous présentons dans ce chapitre une étude détaillée de cette approche, où nous avons donné une description des modèles-hypothèse considérés, et la définition du prédicteur associé à un modèle-hypothèse donné. Ainsi que le problème d'apprentissage de ces prédicteurs.

II.2 Modélisation statique et Modélisation dynamique

II.2.1 Modélisation statique: Il s'agit d'un modèle qui réalise une relation algébrique entre leur entrées et leur sorties. Ce modèle est utilisé pour prédire les états stationnaires de la sortie d'un processus industriel ou pour relier des grandeurs qui sont indépendantes du temps **[21].**

II.2.2 Modélisation dynamique*:* Les entrées et les sorties sont reliées entre elles ; soit par des équations différentielles (modèle à temps continu) soit par des équations récurrentes ou aux différentielles (modèle à temps discret) **[21].**

II.3 Classification des modèles

II.3.1 Classification selon le mode de conception

II.3.1.1 Les modèles de connaissance : Il s'agit d'une description qui peut être fondée sur une analyse physique, chimique, biologique etc. des phénomènes entrant en jeu dans le processus, en appliquant soit les lois générales, fondées sur des principes (lois de la mécanique, de l'électromagnétisme, de la thermodynamique, etc.), soit les lois empiriques (finance, économie) **[07] [13].**

II.3.1.2 Les modèles boîte-noire : Il s'agit d'une description qui peut être fondée sur une analyse statistique des mesures de ses entrées et sorties. La modélisation consiste alors à utiliser, pour représenter les relations entre les entrées et les sorties, des équations (algébriques, différentielles, ou récurrentes) paramétriques, et à estimer les paramètres, à partir des mesures disponibles, de manière à obtenir la meilleure précision possible avec le plus petit nombre possible de paramètres ajustables **[07] [13] [20]**.

II.3.1.3 Les modèles boîte-grise : Entre la boite noire et le modèle de connaissance se situe le modèle semi-physique ou le modèle boite-grise, qui contient à la fois les connaissances et les mesures **[07] [20]**.

II.3.2 Classification selon l'utilisation

II.3.2.1 Prédicteur : Un modèle de prédiction est utilisé en parallèle avec le processus, pour prédire la sortie du processus. Les prédicteurs sont utilisés pour effectuer la synthèse d'une loi de commande, ou pour être incorporé dans un dispositif de commande (commande avec modèle interne).

***II.3.2.2 Simulateur* :** Un simulateur est un système qui possède un comportement dynamique analogue à celui du processus et qui est destiné à fonctionner indépendamment de celui-ci. Les simulateurs sont dans de nombreuses applications **[22] [23]**:

- pour valider des hypothèses sur le processus que l'on étudie, pour extrapoler son comportement dans des domaines de fonctionnement où l'on ne dispose pas de résultats d'expériences.
- pour concevoir un système nouveau et appréhender à l'avance son comportement ou ses caractéristiques (c'est notamment le cas en microélectronique, mais également en construction automobile, etc.); on parle parfois de modèle de conception.
- pour tester de nouveaux dispositifs de commande ou de régulation qu'il serait trop coûteux, ou dangereux, de tester sur le processus lui-même (par exemple si le processus est une partie d'une centrale nucléaire, un avion, etc.).
- pour la synthèse du correcteur d'un système de commande du processus.

II.4 Conception d'un modèle

La première étape d'une modélisation d'un processus consiste à rassembler des connaissances sur son comportement (d'après des expériences et d'une analyse théorique des phénomènes physiques mis en jeu), ce qui conduit à faire plusieurs hypothèses de structures de modèles susceptibles de décrire ce comportement. Ces structures de modèles sont appelées **modèles hypothèse**, ils sont caractérisées par le nombre et la nature de leurs variables d'entrée (entrées de commandes ou perturbatrices), d'état et de sortie, et éventuellement par les relations entre ces variables. Par exemple, si on a des connaissances précises sur le processus, on choisit un modèle de connaissance. Si non on est conduit à choisir des modèles de type **''** boîte noire '' ; on peut également combiner les deux approches au sein d'un même modèle.

La seconde étape de la modélisation, aussi dite **identification**, consiste à estimer les paramètres du modèle. Pour cela, on met en œuvre un système d'apprentissage constitué d'un prédicteur du modèle-hypothèse, et un **algorithme d'apprentissage**. L'estimation des paramètres du modèle est effectuée en minimisant une fonction de coût définie à partir de l'écart entre les sorties mesurées du processus (séquences d'apprentissages) et les valeurs prédites. La qualité de cette estimation dépend du modèle-hypothèse choisi, de la richesse des séquences d'apprentissage et de l'efficacité de l'algorithme utilisé. Enfin, on choisit le modèle neuronal donnant les meilleures performances pour une utilisation prévue (prédiction, simulation, commande) **[13] [19] [20] [23]**.

II.5 Choix d'un modèle hypothèse

Pour élaborer un modèle hypothèse, il faut effectuer des hypothèses concernant la nature et les caractéristiques du processus : caractère statique ou dynamique du processus; présence ou non de perturbations (leur nature, leur mode d'action); caractère linéaire ou non linéaire du processus.

II.5.1 Modèle statique ou dynamique : Lors de la modélisation d'un processus, il est généralement facile de savoir si l'application envisagée nécessite de modéliser la dynamique du processus (où le temps joue un rôle déterminant dans la résolution du problème) ou si une modélisation statique suffit.

II.5.2 Modèle linéaire ou non linéaire : la plupart des processus que l'on peut rencontrer nécessiteraient des modèles non linéaires s'il fallait les décrire de manière précise dans la totalité de leur domaine de fonctionnement alors que les modèles linéaires constituent des approximations valables dans un domaine plus ou moins restreint. Il est donc important de pouvoir élaborer un modèle non linéaire pour rendre compte du comportement d'un processus, non seulement autour de ses points de fonctionnement habituels, mais également lors des passages d'un point de fonctionnement à un autre **[20]**.

II.5.3 Modèle entrée-sortie ou modèle d'état : Dans le cas d'une modélisation dynamique, deux représentations sont possibles: il s'agit d'une représentation d'état ou d'entrée- sortie.

Un modèle d'état est constitué d'un ensemble d'équations de la forme:

$$\begin{cases} \mathbf{x_p(k+1)=f(x_p(k),u(k),b_1(k))} & \text{équation d'état} \\ \mathbf{y_p(k)=g(x_p(k),u(k),b_2(k))} & \text{équation d'observation ou de sortie} \end{cases} \quad \mathbf{(2.1)}$$

Où $x_p(k)$ est le vecteur d'état (dont les composants sont les variables d'état), $u(k)$ est le vecteur des entrées de commande, $y_p(k)$ le vecteur de sortie du processus et les $b(k)$ sont des bruits. **f** et **g** sont des fonctions non linéaires. La dimension du vecteur d'état (le nombre de variables d'état) est appelé l'ordre du modèle.

L'objectif de la conception d'un modèle sous forme de représentation d'état est de trouver des approximations des deux fonctions **f** et **g** par apprentissage **[07] [09] [13] [21].**

Un modèle hypothèse entrée –sortie est défini par une relation de la forme:

$$y_p(k)=h(y_p(k-1)\ldots,.y_p(k-n),u(k)\ldots,u(k-m),b(k)\ldots\ldots b(k-p)) \quad \mathbf{(2.2)}$$

Où **h** est une fonction non linéaire, n est l'ordre du modèle, *m* et *p* sont deux nombres entiers constants positifs, $y_p(k)$ le vecteur de sortie du processus, **u(k)** est le vecteur des signaux de commande, **b(k)** est le vecteur des bruits.

II.5.4 Présence d'un bruit

Une fois on a effectué le choix entre la représentation entrée-sortie ou d'état**,** il convient de faire des hypothèses sur la façon dont le bruit intervient dans le processus **[07]**. On distingue notamment un bruit additif qui affecte la sortie du processus, un bruit additif qui affecte l'état du processus, et un bruit qui affecte les deux à la fois (l'état et la sortie). Comme, en général, on ne connaît pas avec précision la nature du bruit qui affecte le processus, on doit effectuer des hypothèses sur celle-ci ; on déduit de celles-ci la structure du modèle hypothèse, et l'algorithme utilisé pour l'ajustement des paramètres. Une hypothèse erronée peut dégrader considérablement les performances du modèle **[15] [24].**

II.5.5 Notation [19]

Dans le cas d'un modèle-hypothèse d'état, nous notons $\mathbf{x_p} \in R^n$ l'état du processus et $\mathbf{y_p} \in R$ sa sortie, $\mathbf{x} \in R^n$ l'état du prédicteur et $\mathbf{y} \in R$ sa sortie, et $\mathbf{u} \in R$ la commande. Dans le cas d'un modèle hypothèse entrée-sortie, nous notons $\mathbf{y_p} \in R$ la sortie du processus et $\mathbf{y} \in R$ celle du prédicteur, et $\mathbf{u} \in R$ la commande.

a- La notation :

$$y(k+1) = \psi_{RN}(y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1), C) \quad \textbf{(2.3)}$$

Signifie un réseau prédicteur non bouclé où $\boldsymbol{\psi_{RN}}$ est la fonction non linéaire réalisée par le réseau et *C* le vecteur des ses paramètres (voir figure II.1).

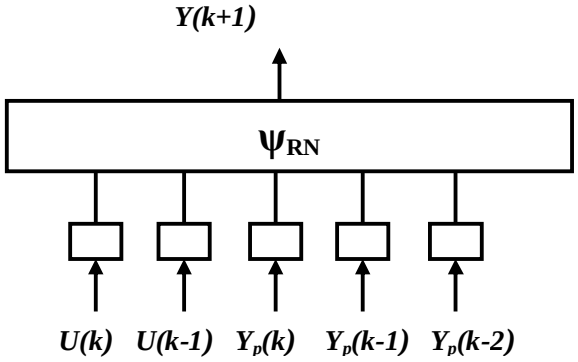


Figure II.1 : Exemple d'un réseau non bouclé avec : n=3 ; m=2 **[19]**.

b- La notation :

$$y(k+1) = \psi_{RN}\,(y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-m+1), C) \qquad \textbf{(2.4)}$$

Signifie un réseau prédicteur bouclé d'ordre n, dont les n variables d'état sont la sortie et n-1 valeurs retardées de celle-ci. $\boldsymbol{\psi_{RN}}$ est la fonction non linéaire réalisée par la partie non bouclée du réseau munie des coefficients C. Un tel réseau est appelé réseau bouclé entrée-sortie (voir figure II.2).

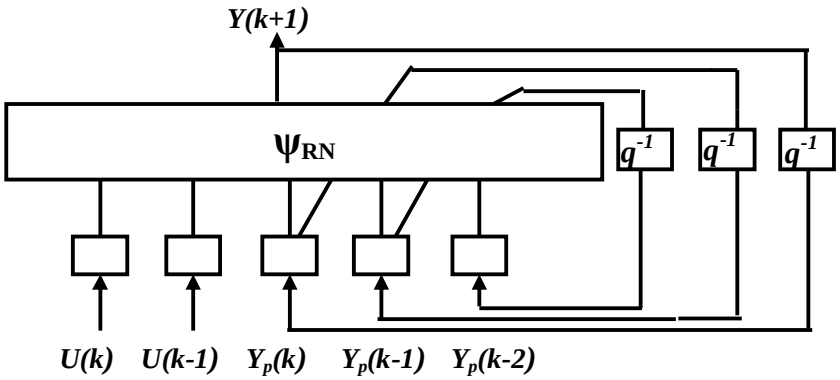


Figure II.2 : Exemple de réseau bouclé entrée-sortie avec : n=3 ; m=2 **[19]**.

c- La notation :

$$y(k+1) = \psi_{RN}\,(y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1), e(k), \ldots e(k-p+1)\,, C) \qquad \textbf{(2.5)}$$

Signifie un réseau prédicteur bouclé d'ordre p, dont les p variables d'état sont les p valeurs successives de l'erreur e(k) = yp(k) – y(k). $\boldsymbol{\psi_{RN}}$ est la fonction non linéaire réalisée par la partie non bouclée du réseau munie des coefficients C .Un tel réseau est appelé réseau bouclé entrée erreur (voir figure II.3).

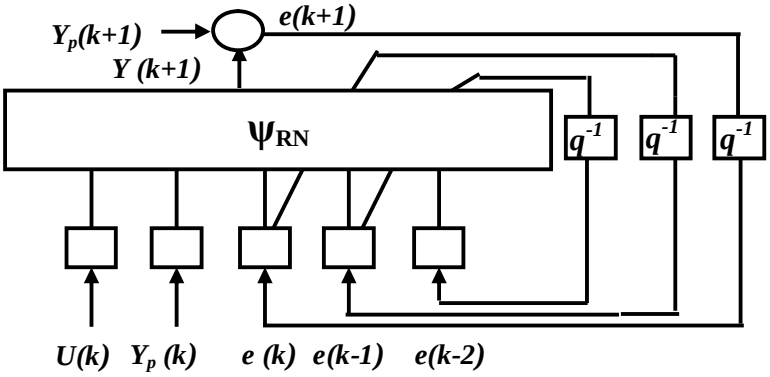


Figure II.3 : Exemple de réseau bouclé entrée-erreur avec : n=1 ; m=1 ; p=3 **[19]**.

d- La notation :

$$\begin{cases} x(k+1) = \varphi_{RN}(x(k), u(k); C) \\ y(k+1) = \Psi_{RN}(x(k), u(k); C) \end{cases} \quad \textbf{(2.6)}$$

Désigne un réseau prédicteur bouclé dont les variables d'état sont les activités de neurones d'état distincts des neurones de sortie, donc pour lesquels il n'y a pas de valeurs désirées.

φ_{RN} et Ψ_{RN} sont les fonctions réalisées par la partie non bouclée du réseau munie des coefficients C. Un tel réseau est appelé réseau d'état. (Voir figure II.4.a).

Lorsqu'on impose une dépendance exclusive de y(k+1) en les variables d'état x(k+1), on note le réseau (voir figure II.4.b).

$$\begin{cases} x(k+1) = \varphi_{RN}(x(k), u(k); C) \\ y(k+1) = \Psi_{RN}(x(k+1); C) \end{cases} \quad \textbf{(2.7)}$$

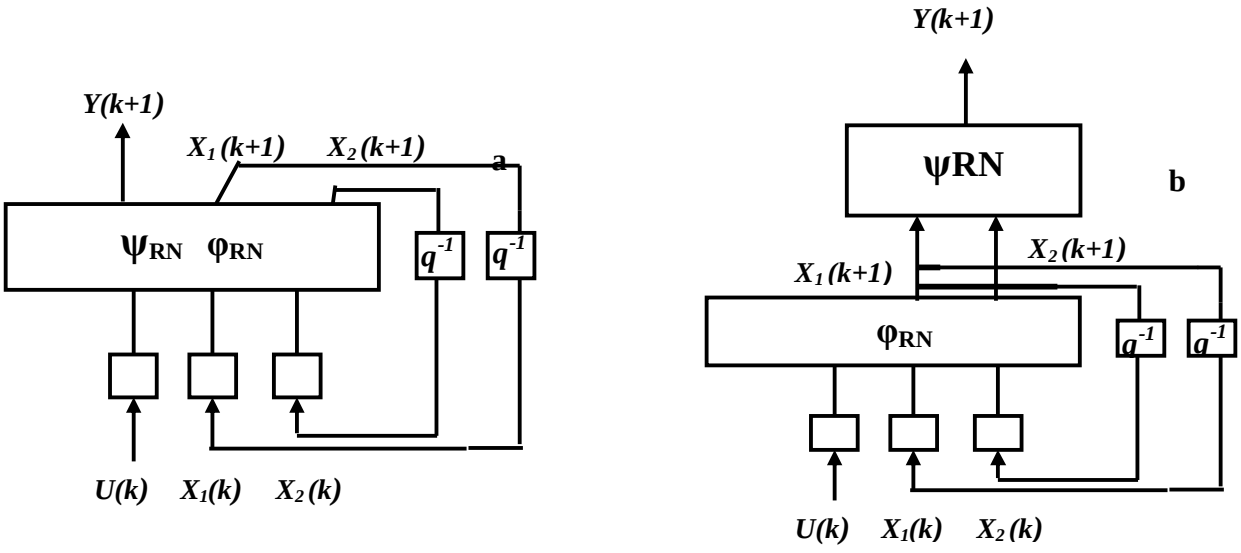


Figure II.4 : Exemple d'un réseau prédicteur d'état avec : n=2 **[19]**.

II.6 Modèles hypothèses dynamiques et prédicteurs associés

La notion de prédicteur associé est évidente : c'est un prédicteur qui, si l'hypothèse est vraie, permet de prédire sans erreur la sortie du processus, dans le cas où le modèle hypothèse est déterministe et qui minimise la variance de l'erreur de prédiction dans le cas d'un modèle hypothèse avec bruit. Nous commençons par présenter les

modèles entrée-sortie pour faciliter la compréhension, puis nous présentons les modèles hypothèses d'état. Ces considérations sont largement développées dans **[07] [09] [13] [14] [15] [19]**.

II.6.1 Représentation entrée –sortie

On peut citer plusieurs hypothèses suivant la manière dont le bruit agit sur le processus.

II.6.1.1 Modèle entrée-sortie avec bruit d'état NARX

Nous introduisons en premier lieu le modèle avec l'hypothèse d'un bruit d'état, NARX (Non linéaire Auto Régressif avec entrée eXogène ou extérieure). Il s'écrit :

$$y_p(k) = h(\, yp(k-1), \ldots, y_p(k-n), u(k-1), \ldots, u(k-m)) + b(k) \qquad \textbf{(2.8)}$$

Avec: $y_p(k)$ la sortie du processus. On suppose que le bruit b(k) est un bruit blanc et affecte la boucle du processus.

Le prédicteur théorique associé à ce modèle non bouclé est donné par l'expression :

$$y(k+1) = h(\, y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1)) \qquad \textbf{(2.9)}$$

Lors de la phase d'identification, il faut donc mettre en ouvre un prédicteur neuronal ayant la même structure que le préditeur théorique, donc un réseau de neurones non bouclé de la forme :

$$y(k+1) = \psi_{RN}\,(y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1), C) \qquad \textbf{(2.10)}$$

$\boldsymbol{\psi_{RN}}$ est la fonction non linéaire réalisée par le réseau et *C* le vecteur de ses paramètres.

Avec des conditions initiales correctes (modèle hypothèse correcte, séquences d'apprentissage suffisamment riches et un algorithme d'apprentissage performant) ,

On a alors: $y_p(k)-y(k)=b(k)$. **(2.11)**

Ce modèle est tel que l'erreur de modélisation soit égale au bruit ; c'est le modèle idéal. La fonction $\boldsymbol{\psi_{RN}}$ sera une bonne approximation de la fonction **h**. (figure II.5), et le prédicteur neuronal sera la bonne approximation du prédicteur théorique.

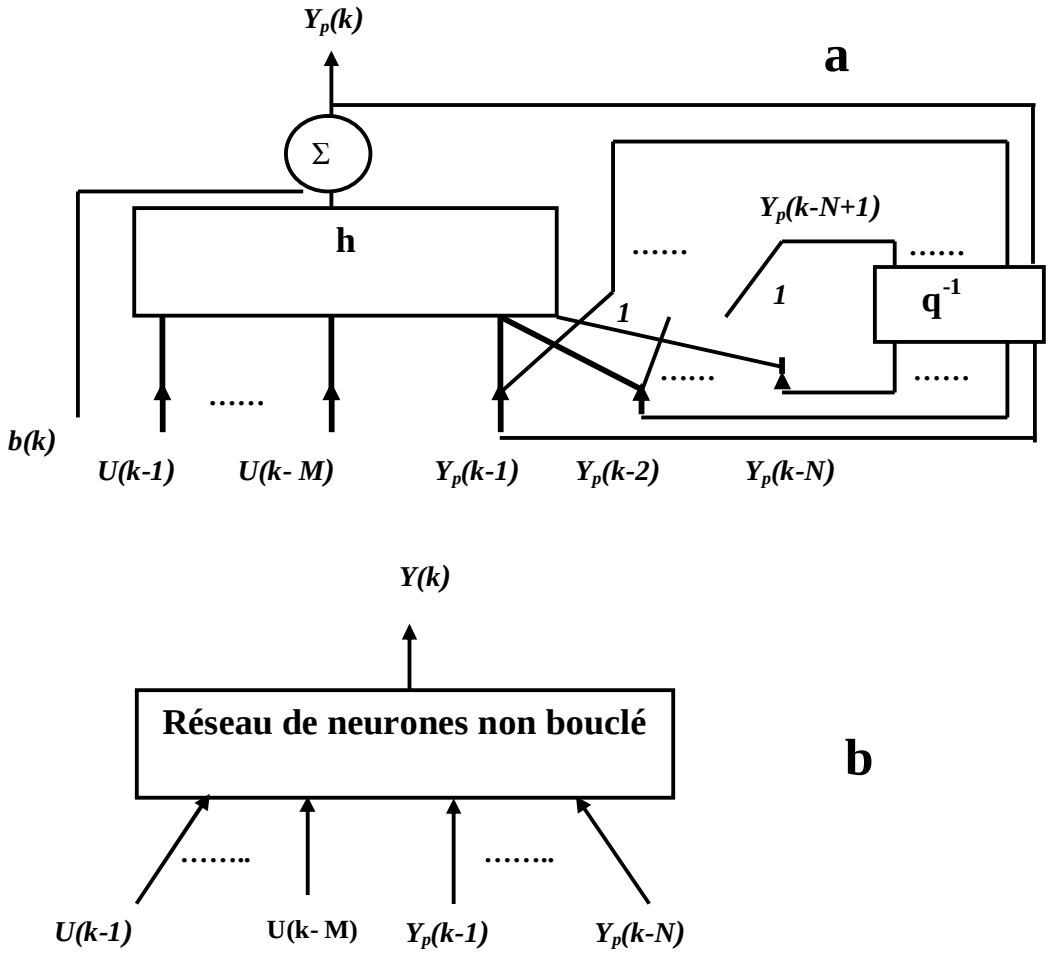


Figure : II.5

a : *Modèle* entrée-sortie avec bruit d'état NARX

b : Prédicteur neuronal non bouclé d'entrées définies à partir du prédicteur théorique associé **[15]**.

Donc les entées imposées et la sortie de la copie n° k pour l'apprentissage sont :

$$\begin{cases} I^K = \{y_p(k),....,y_p(k-n+1),u(k),....u(k-m+1)\} \\ O^K = y(k+1) \end{cases} \quad \textbf{(2.12)}$$

Ce prédicteur peut être réalisé à l'aide du réseau complètement connecté (figure II.6). C'est un réseau non bouclé, donc n'a pas d'état (ne possède pas d'entrées ni de sorties d'état S_{in} et S_{out},). Il définit la copie du réseau à l'instant k. où les entrées externes sont les composantes du vecteur I(k), de dimension m+n.

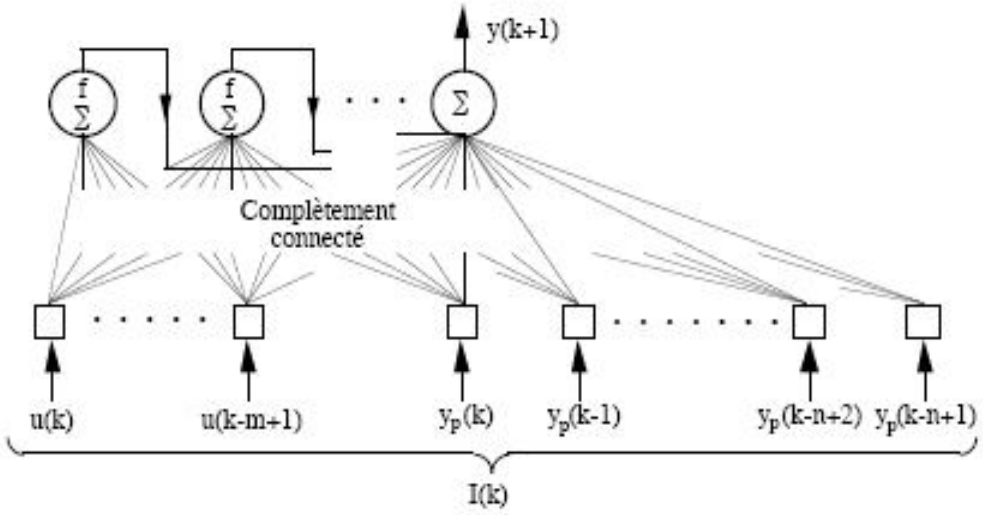


Figure II.6 : Copie n° k du réseau d'apprentissage du prédicteur non bouclé associé à un modèle hypothèse NARX **[14]**.

Le système d'apprentissage utilisé pour ce prédicteur est représenté sur la figureII.7. Pour le prédicteur non bouclé, on utilise un algorithme dirigé par le processus, donc le réseau n'a pas d'état, et qu'il est donc entièrement dirigé par les entrées externes à chaque instant de la fenêtre d'apprentissage. Le réseau est constitué de N copies indépendantes du réseau dont chacune fournit en sortie une erreur intervenant dans la fonction de coût. Chacune de ces erreurs est l'entrée d'un réseau de rétro-propagation, et le gradient est la somme des N gradients partiels calculés par rétropropagation.

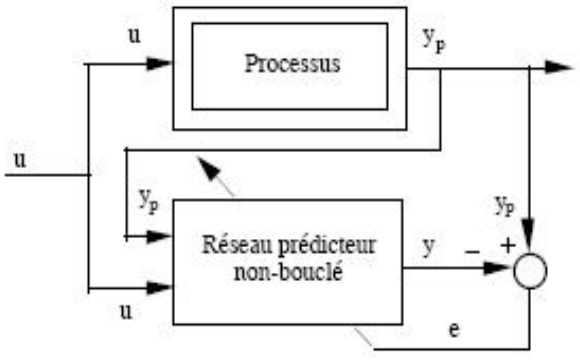


Figure II.7 : Modèle-hypothèse entrée-sortie NARX :
Le système d'apprentissage utilise un prédicteur non bouclé et un algorithme dirigé **[14]**.

II.6.1.2 Modèle entrée-sortie avec bruit de sortie additif NBSX

Un deuxième modèle hypothèse présente un bruit affectant la sortie. Ce modèle NBSX ((Non linéaire avec Bruit de Sortie additif et entrée eXogène) est de la forme:

$$\begin{cases} x_p(k) = h(x_p(k-1), \ldots, x_p(k-n), u(k-1), \ldots, u(k-m)) \\ y_p(k) = x_p(k) + b(k) \end{cases} \quad \textbf{(2.13)}$$

Le prédicteur théorique associé au modèle hypothèse avec un bruit de sortie est un prédicteur bouclé :

$$y(k+1) = h(y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-m+1)) \quad \textbf{(2.14)}$$

Supposons que les erreurs de prédiction soient égales au bruit, on a alors :

$$\begin{cases} y_p(k) - y(k) = b(k) \\ \ldots \\ y_p(k-n+1) = x_p(k-n+1) + b(k-n+1) \end{cases} \quad \text{avec} \quad \begin{cases} y_p(k) = x_p(k) + b(k) \\ \ldots \\ y_p(k-n+1) - y(k-n+1) = b(k-n+1)) \end{cases}$$

$$\begin{cases} y(k) = xp(k) \\ y(k-n+1) = x_p(k-n+1) \end{cases} \quad \ldots$$

alors

$y(k+1) = h(y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-m+1))$

$= h(x_p(k), \ldots, x_p(k-n+1), u(k), \ldots, u(k-m+1))$

$= x_p(k+1)$

et on a bien :

$$y_p(k+1) - y(k+1) = y_p(k+1) - x_p(k+1) = b(k+1) \quad \textbf{(2.15)}$$

Pour obtenir un modèle prédictif non biaisé, if faut mettre en ouvre un prédicteur neuronal bouclée comme le prédicteur théorique :

$$y(k+1) = \psi_{RN}(y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-m+1), C) \quad \textbf{(2.16)}$$

Ici encore, si l'hypothèse est vraie, et si toutes les conditions d'une bonne identification sont réunies, alors la fonction réalisée par la partie non bouclée du réseau de neurones sera une bonne approximation de h.(figure II.8).

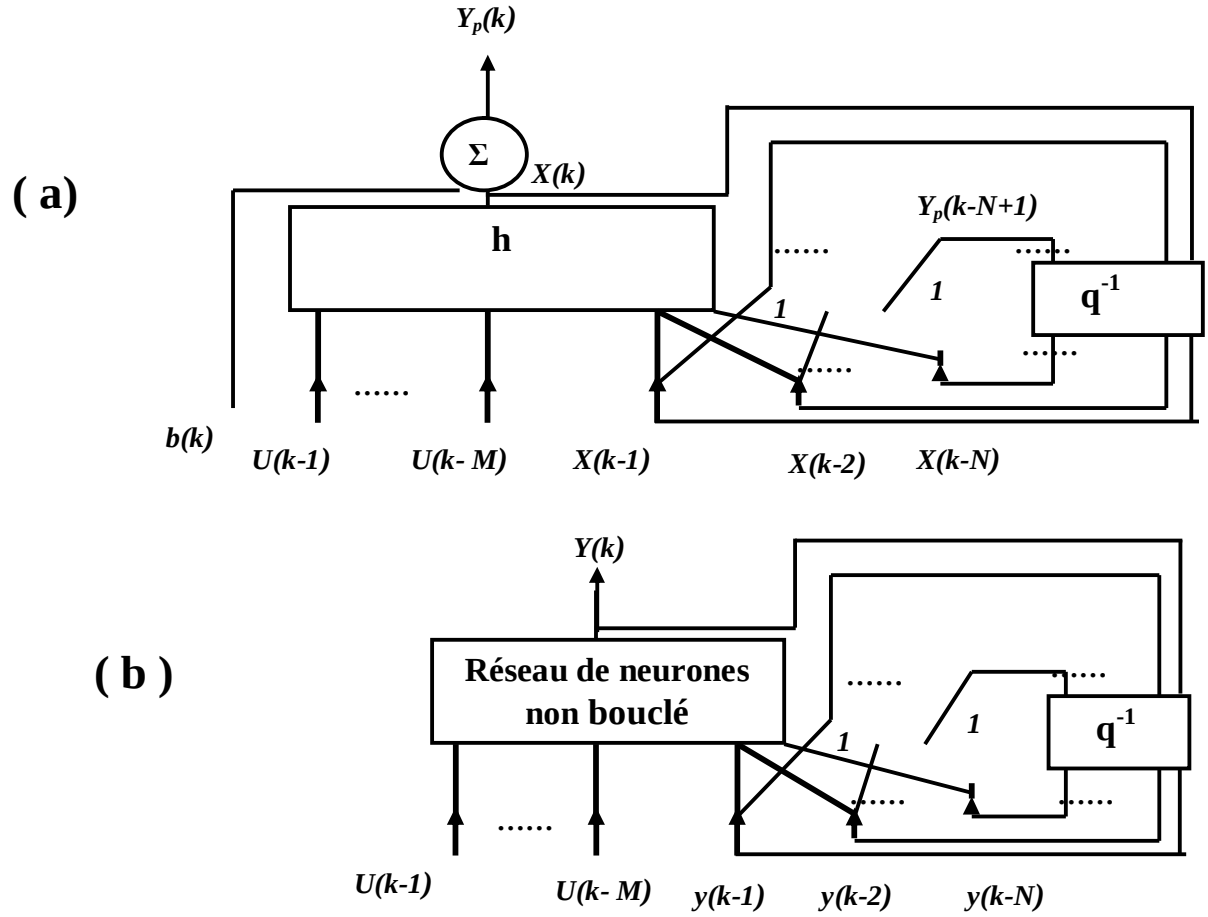


Figure :II.8
a : Modèle entrée-sortie Avec bruit de sortie NBSX
b : Prédicteur neuronal bouclé d'entrées définies à partir du prédicteur théorique associé **[15]**.

Un tel prédicteur peut être réalisé, par un réseau complètement connecté de la figure II.9. Les entrées imposées, les entrées d'état, les sorties d'état et les sorties de la copie n°k pour l'apprentissage sont :

$$\begin{cases} I^K = \{u(k),...u(k-m+1)\} \\ S_{in}^K = S_{out}^{K-1} = \{y(k),.....y(k-p+1)\} \\ O^K = y(k+1) \end{cases} \quad \textbf{(2.17)}$$

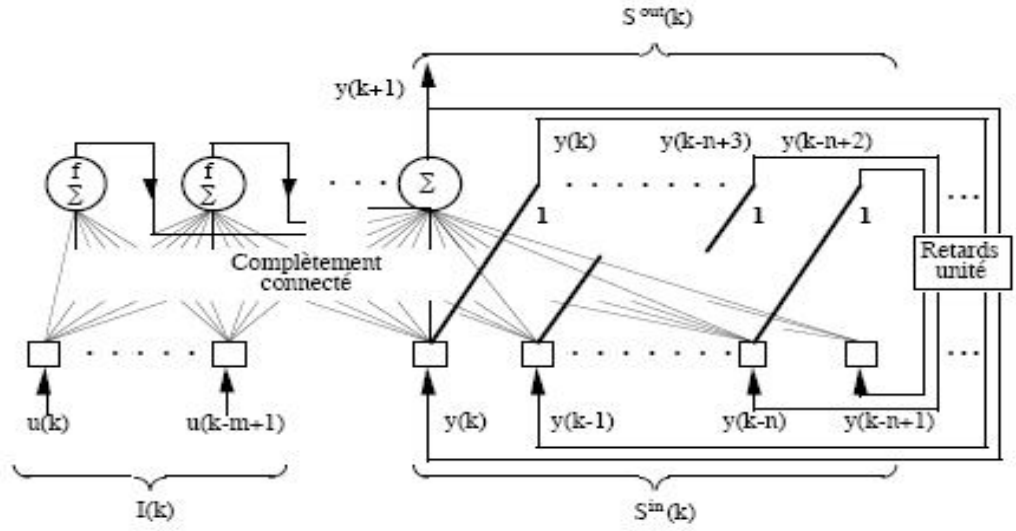


Figure II.9 : Copie n°k du réseau d'apprentissage du prédicteur bouclé associé à un modèle NBSX à l'aide d'un réseau bouclé complètement connecté **[14]**.

Le système d'apprentissage du prédicteur bouclé associé à ce modèle NBSX est représenté sur la figure II.10. Le prédicteur est semi-dirigé car les valeurs de ses entrées d'état qui ne sont imposées qu'au début de la fenêtre de la fonction de coût.

Nous remarquons que y(k) est la seule valeur calculée par le réseau, les autres variables d'état sont des sorties décalées dans le temps.

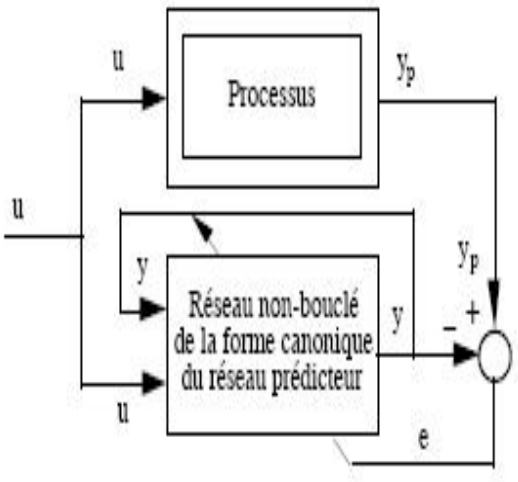


Figure II.10: Modèle-hypothèse entrée-sortie NBSX :

Le système d'apprentissage utilise un prédicteur bouclé sur sa sortie et un algorithme semi-dirigé **[14]**.

Dans le domaine du traitement du signal et de l'automatique, les modèles précédents sont souvent insuffisants pour décrire correctement les processus. Le modèle-hypothèse non linéaire le plus général est le modèle NARMAX (Non linéaire Auto-Régressif à Moyenne Ajustée avec entrée eXogène). Il est de la forme :

$y_p(k) = h(y_p(k-1), \ldots, y_p(k-n), u(k-1), \ldots, u(k-m), b(k-1), \ldots, b(k-p)) + b(k)$ **(2.18)**

Le prédicteur théorique associé à ce modèle est le suivant :

$y(k+1) = h(y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1), e(k), \ldots, e(k-p+1))$ **(2.19)**

Où $e(k) = y_p(k) - y(k)$ est l'erreur de prédiction et est égale au bruit. $e(k)=b(k)$

Le système d'apprentissage doit donc utiliser un réseau de neurones prédicteur bouclé d'ordre p dont les variables d'état sont les p erreurs de prédiction passées :

$y(k+1) = \psi_{RN}(y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1), e(k), \ldots e(k-p+1), C)$ **(2.20)**

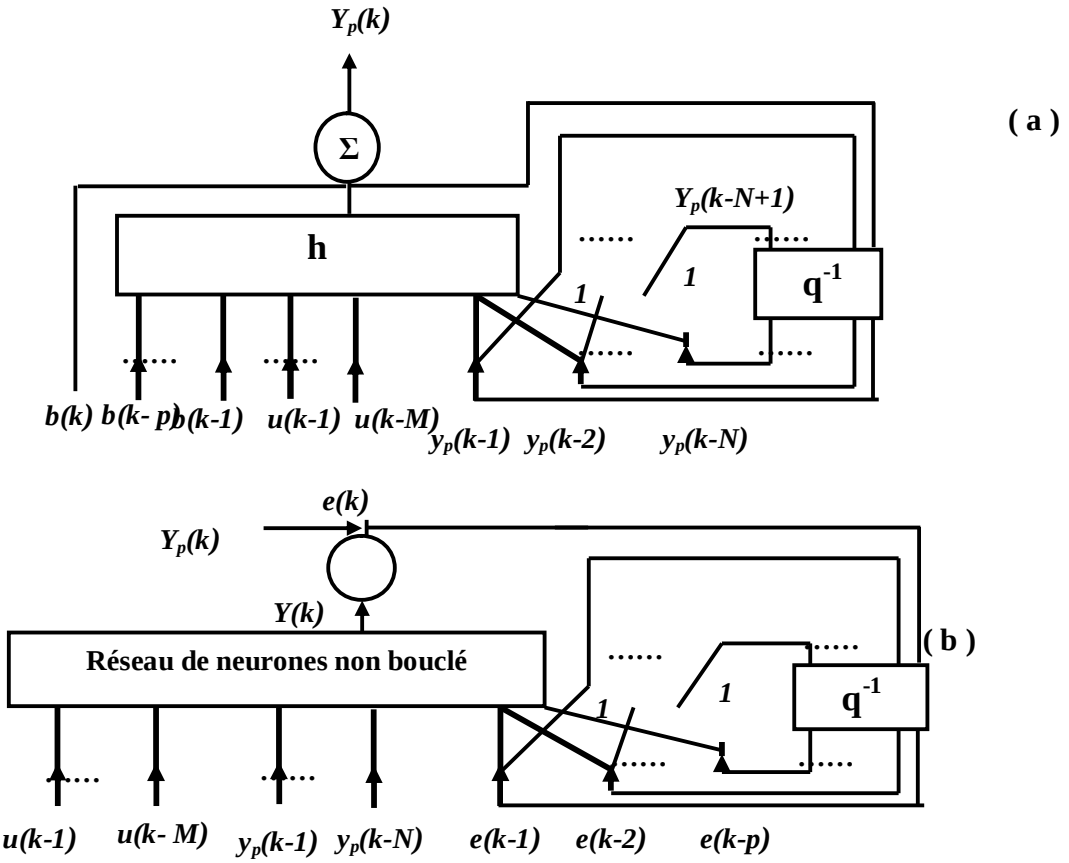


Figure II.11 :

a : Modèle entrée-sortie NARMAX

b : Prédicteur neuronal définie à partir du prédicteur théorique associé **[9]** .

De même, si l'hypothèse est vraie, et si toutes les conditions d'une bonne identification sont remplies, alors la fonction réalisée par la partie non bouclée du réseau de neurones est une bonne approximation de h. (figure II.11).

Les entrées imposées, les entrées d'état, les sorties d'état et les sortie de la copie n°k pour l'apprentissage sont :

$$\begin{cases} I^K = \{y_p(k)...., y_p(k-n+1), u(k),...u(k-m+1)\} \\ S_{in}^K = S_{out}^{K-1} = \{y(k),..... y(k-p+1)\} \\ O^K = y(k+1) \end{cases} \quad \textbf{(2.21)}$$

La figure II.12 représente le système d'apprentissage du prédicteur bouclé utilisé .

Pour l'identification d'un prédicteur NARMAX, la fonction ψ_{RN} est réalisé à l'aide d'un réseau de neurones complètement connecté.

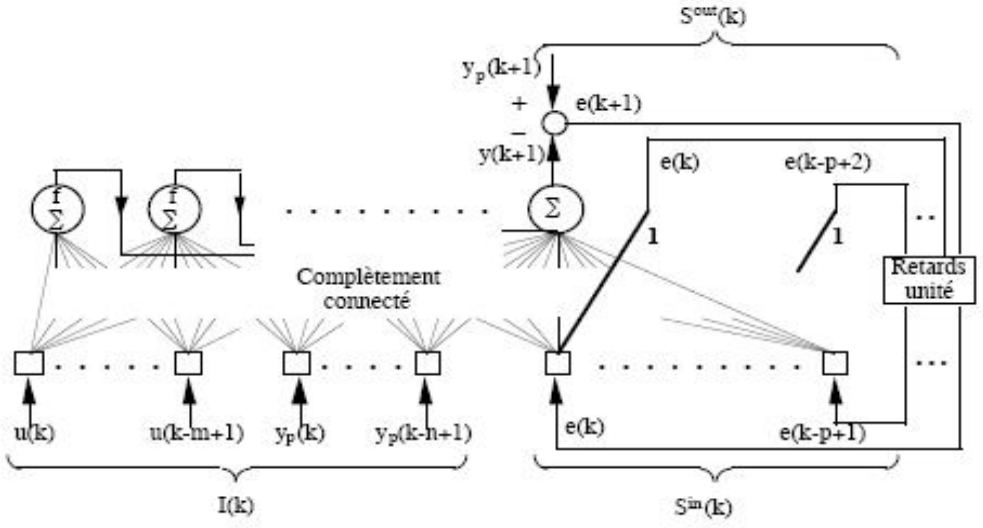


Figure II.12 :Prédicteur associé à un modèle-hypothèse NARMAX réalisé à l'aide d'un réseau bouclé complètement connecté **[9]**.

Le système d'apprentissage utilisé pour l'identification de ce prédicteur est représenté sur la figure II.13. Pour ce prédicteur bouclé, on utilise un algorithme semi-dirigé. Le réseau est constitué de N copies en cascade. Après l'initialisation des entrées d'état de la première copie, on laisse le réseau évoluer sans le diriger.

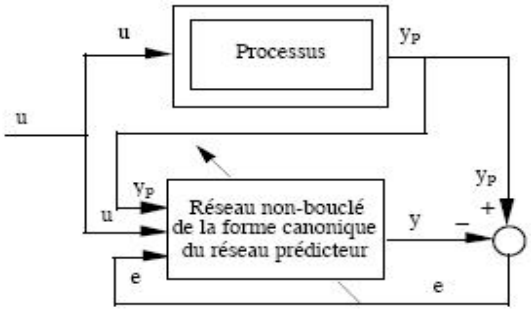


Figure II. 13 : Modèle-hypothèse entrée-sortie NARMAX :
Le système d'apprentissage utilise un prédicteur bouclé sur l'erreur de prédiction et un algorithme semi-dirigé **[9]**.

II.6.2 Représentation d'état

Nous reprenons les mêmes hypothèses précédentes avec des modèles d'état.

II.6.2.1 Modèle d'état Avec bruit de sortie

Considérons tout d'abord l'hypothèse selon laquelle le bruit serait un bruit de sortie. Le comportement du processus est décrit par des équations de la forme:

$$\begin{cases} x_p(k+1) = f\,(x_p(k), u(k)) \\ y_p(k) = g\,(x_p(k)) + b(k) \end{cases} \qquad \textbf{(2.22)}$$

Le bruit n'intervenant que dans l'équation de sortie; il n'a aucune influence sur la dynamique du modèle. Pour des raisons analogues à celui qu'on a développé pour le cas d'une représentation entrée sortie, le modèle idéal est un modèle bouclé.

Le prédicteur optimal associé est :

$$\begin{cases} x(k+1) = f\,(x(k), u(k)) \\ y(k+1) = g\,(x(k+1)) \end{cases} \qquad \textbf{(2.23)}$$

Si on suppose que $x(k) = x_p(k)$, on a $x(k+1) = x_p(k+1)$,

$$\text{et :}\ e(k+1) = y_p(k+1) - y(k+1) = b(k+1) \qquad \textbf{(2.24)}$$

Autrement dit que l'erreur de prédiction est égale au bruit.

Le système d'apprentissage doit utiliser un prédicteur neuronal bouclé sous la forme :

$$\begin{cases} x(k+1) = \Psi_{RN}\,(x(k), u(k); C) \\ y(k+1) = \xi_{RN}\,(x(k+1); C) \end{cases} \qquad \textbf{(2.25)}$$

ou bien de la forme:

$$\begin{cases} x(k+1) = \Psi_{RN}(x(k), u(k); C) \\ y(k+1) = \xi_{RN}(x(k), u(k); C) \end{cases} \quad \textbf{(2.26)}$$

Où les fonctions Ψ_{RN} *et* ξ_{RN} réalisent exactement les fonctions **f** et **g** respectivement.

Noter que le premier pédicteur (équation 2.24) nécessite au moins deux sous –réseaux alors que la seconde forme (équation2.25) peut être réalisée avec un seul réseau (figure II.14).

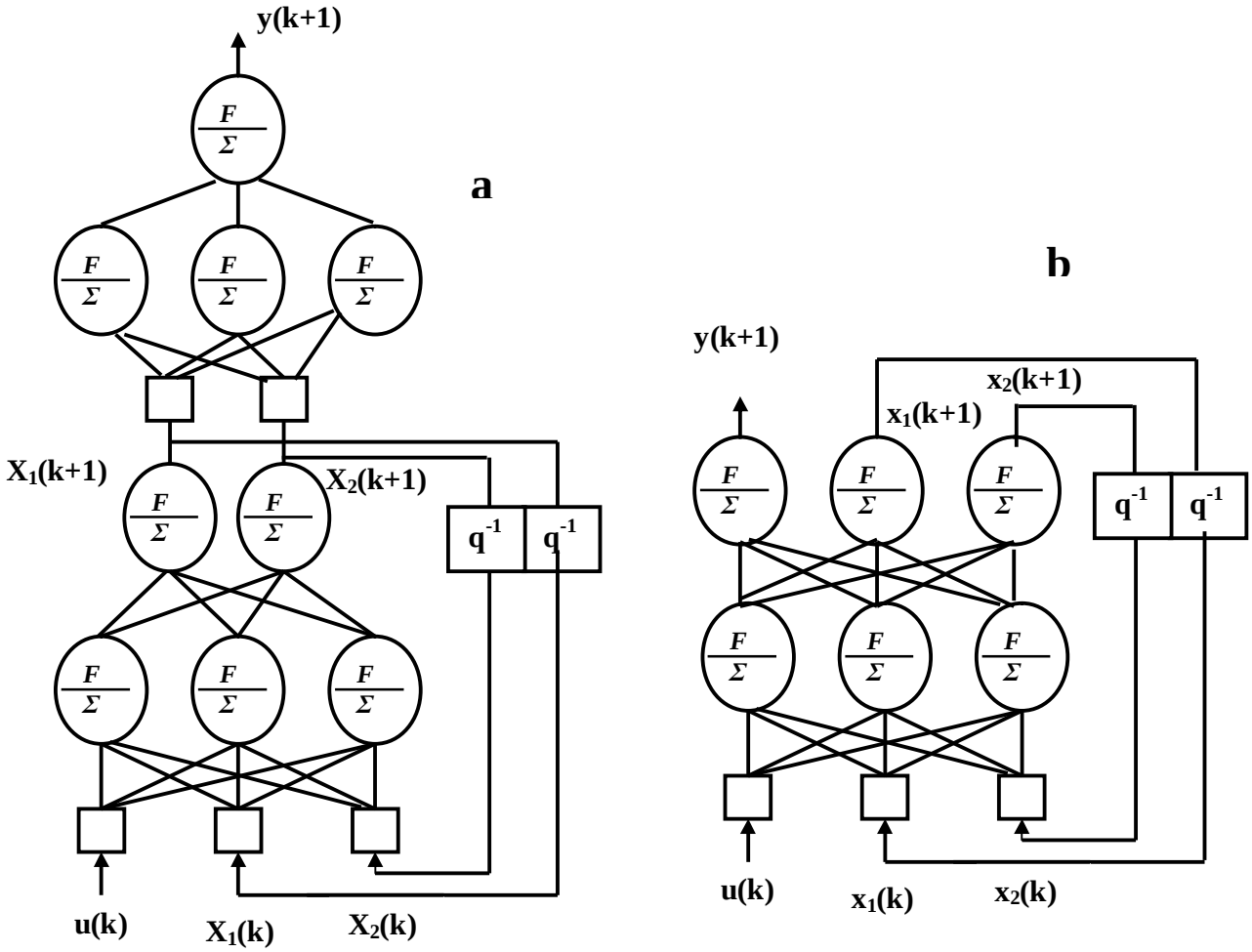


Figure II.14 : Prédicteur neuronal définie à partir du prédicteur théorique associe au modèle hypothèse d'état avec un bruit de sortie

a : Prédicteur réalisé avec deux sous-réseaux.

b : Prédicteur réalise avec un seul réseau **[9] .**

II.6.2.2 Modèle d'état avec bruit d'état

Ce modèle s'écrit:

$$\begin{cases} x_p(k+1) = f\,(x_p(k),\, u(k), w(k)) \\ y_p(k) = g\,(x_p(k)) \end{cases} \quad \textbf{(2.27)}$$

Alors par un argument analogue à celui qui a été développé pour les modèles entrée-sortie, le modèle idéal devrait avoir pour entrées, outre les entrées de commande u(k), les variables d'état du processus. Deux cas peuvent se présenter :

-1 Lorsque l'état du processus est mesuré

L'identification du prédicteur associé à un modèle-hypothèse d'état affecté d'un bruit d'état additif lorsque l'état du processus est mesuré impose aussi un prédicteur entrée-sortie non bouclé. Pour réaliser ce prédicteur, nous avons vu que l'on peut utiliser deux réseaux de neurones distincts, l'un réalisant la prédiction de l'état, et l'autre l'équation d'observation. Par exemple, le prédicteur associé au problème peut être réalisé à l'aide du réseau composé de n sous-réseaux complètement connectés, et du réseau complètement connecté de la figure II.15.

Ou : Pour réseau N°1 :

$$\begin{aligned} I^K &= \{X_p(k)....X_p(k-m+1), u(k)\} \\ O^K &= X(k+1) \end{aligned} \quad \textbf{(2.28)}$$

Pour réseau N°2 :

$$\begin{aligned} I^K &= \{X_p(k)....X_p(k-m+1)\} \\ O^K &= y(k+1) \end{aligned} \quad \textbf{(2.29)}$$

Le système d'apprentissage utilisant ces deux réseaux est représenté sur la figure II.16.
L'apprentissage des deux réseaux est dirigé par les variables d'état mesurées du processus. On est encore une fois en présence de N copies indépendantes de chacun des réseaux prédicteurs dont chacune fournit en sortie une erreur, e_x de dimension n et e_y de dimension 1, intervenant dans les 2 fonctions de coût.

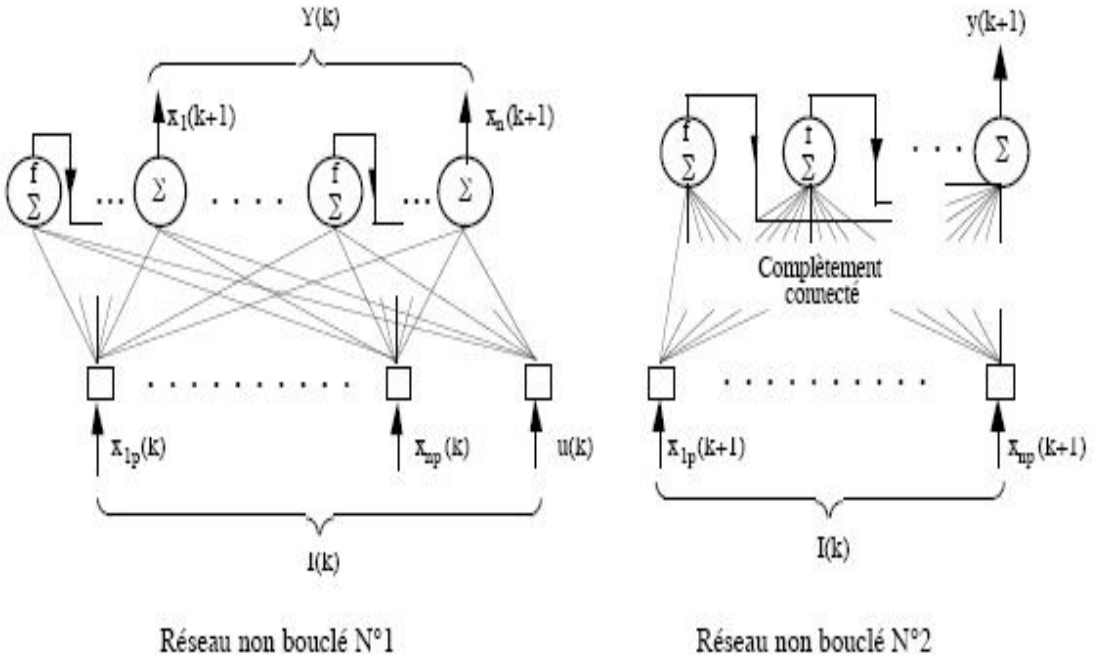


Figure II.15 : Le prédicteur associé à un modèle-hypothèse d'état affecté d'un bruit d'état, réalisé à l'aide d'un réseau composé de n sous-réseaux complètement connectés (réseau N°1), et d'un réseau complètement connecté (réseau N°2) **[20]**.

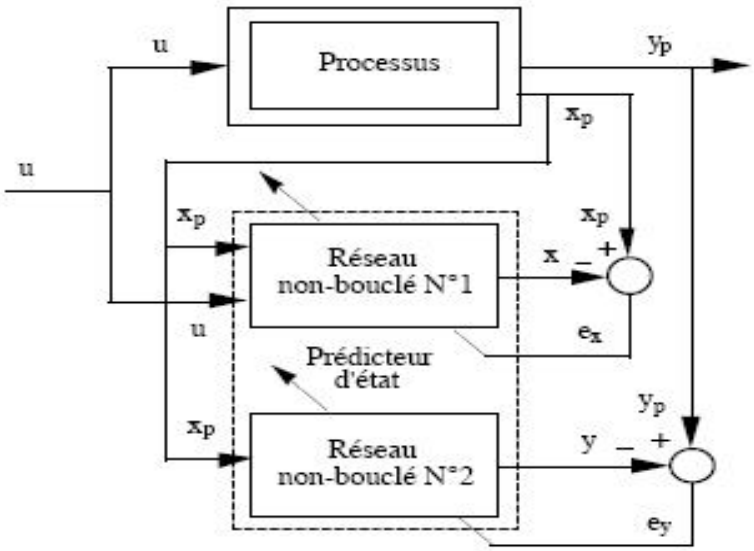


Figure II.16: Modèle-hypothèse d'état avec bruit d'état additif.
Le système d'apprentissage utilise un prédicteur composé de deux réseaux non bouclés et un algorithme dirigé. **[23]**.

- Lorsque l'état de processus n'est pas mesuré

L'identification du prédicteur associé à un modèle-hypothèse d'état lorsque l'état du processus n'est pas mesuré impose l'utilisation d'un réseau de neurones bouclé. Ce prédicteur peut, par exemple, être réalisé à l'aide du réseau composé de n+1 sous réseaux à une couche de neurones cachés de la figure II.17.

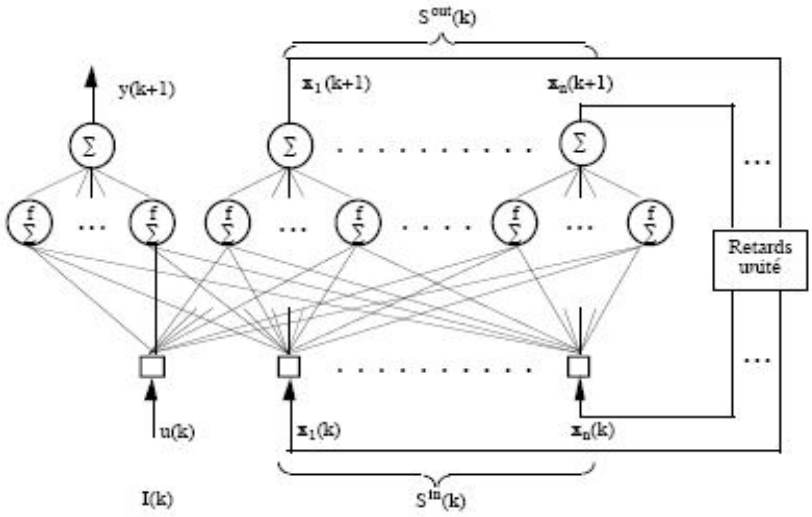


Figure II.17 : Prédicteur associé à à un modèle-hypothèse d'état, lorsqu'on ne mesure pas l'état du processus,réalisé à l'aide d'un réseau composé de n+1 sous-réseaux à une couche de neurones cachés **[23]**.

Le système d'apprentissage utilisant le prédicteur est représenté schématiquement sur la figure II.18.

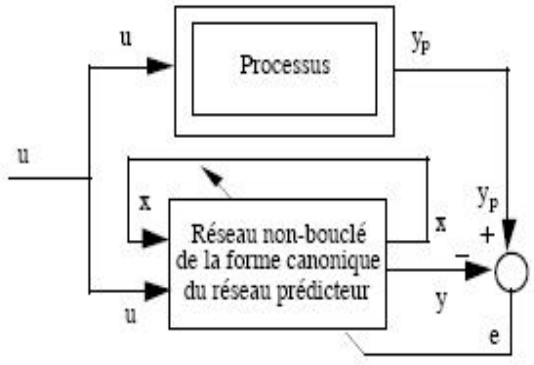


Figure II.18 : Modèle-hypothèse d'état, l'état du processus n'est pas mesuré : le système d'apprentissage utilise un prédicteur bouclé sur l'état (non imposé) et un algorithme semi-dirigé **[23]**.

Le prédicteur est encore dit semi-dirigé car les valeurs de ses entrées d'état ne sont imposées qu'au début de la séquence d'apprentissage. Comme pour un prédicteur entrée-sortie bouclé, le calcul du gradient doit être effectué sur le réseau composé des N copies en cascade.

L'algorithme de calcul de la fonction de coût et de son gradient est semi-dirigé.

II.6.2.3 Modèle d'état avec bruit d'état et de sortie NARMAX

Le processus est décrit par un modèle de la forme:

$$\begin{cases} x_p(k+1) = f\,(x_p(k),\, u(k), b_1(k)) \\ y_p(k) = g\,(x_p(k),\, b_2(k)) \end{cases} \quad \textbf{(2.30)}$$

On peut l'exprimer par un prédicteur de Kalman étendu **[09],** et on peut choisir la structure neuronale qui fait intervenir à la fois l'état et la sortie mesurée du processus.

$$\begin{cases} x_p(k+1) = f\,(x_p(k),\, u(k), y_p(k), C) \\ y_p(k+1) = g\,(x_p(k+1),\, C) \end{cases} \quad \textbf{(2.31)}$$

II.7 Le choix entre les deux modèles hypothèses

Plusieurs situations peuvent se présenter **[19] :**

A - Si on possède des connaissances physique, chimique,…etc, sur le comportement du processus qui peuvent conduire à un modèle de connaissance. Même très proche, une telle analyse aboutit à la conception d'un modèle d'état, dans lequel les variables d'état ont une signification (physique : pression, chimique : concentration….) :

-*Si les variables d'état sont mesurées*, les données utilisées pour la modélisation sont la séquence des entrées, et les séquences des variables d'état et des sorties mesurées (séquences d'apprentissage). On peut estimer la fonction f de l'équation d'état à l'aide d'un modèle entrée-sortie, dont les sorties sont les variables d'état du modèle d'état ; l'équation d'observation (fonction g) est modélisée séparément.

-*Si les variables d'état ne sont pas mesurées*, les séquences d'apprentissage sont alors la séquence des entrées et la séquence des sorties mesurées. Deux démarches sont possibles :

- on peut chercher à modéliser le comportement entrée-sortie du processus à l'aide d'un modèle d'état, mais dont l'état n'est pas imposé.
- on peut faire l'hypothèse plus restrictive d'un modèle entrée-sortie.

B - Si ces connaissances font défaut (Le processus est trop complexe pour être modélisé physiquement.), on a recours à la modélisation boite-noire. En pratique, cette situation est équivalente au cas où une modélisation physique est possible, mais l'état n'est pas mesuré. En effet, les mêmes démarches sont envisageables :

- tenter de modéliser le comportement global du processus à l'aide d'un modèle d'état, sans imposer l'état.
- faire l'hypothèse plus restrictive d'un modèle-hypothèse entrée-sortie.

La mise au point d'un modèle entrée-sortie ou d'un modèle d'état dans ces dernières conditions repose donc surtout sur les résultats expérimentaux.

Toujours dans le cadre d'une approche boîte noire, il peut être intéressant d'avoir recours à une représentation d'état. En effet, les modèles d'état sont plus généraux, ils constituent une classe plus vaste de systèmes dynamiques que les modèles entrée-sortie. On peut toujours trouver une représentation d'état équivalente pour un modèle entrée –sortie non linéaire, mais il n'existe pas toujours de modèle entrée-sortie équivalent à un modèle linéaire **[09][[14][21][19].** De plus, une représentation entrée-sortie équivalente à une représentation d'état donnée peut être d'ordre plus élevé. Donc, les modèles d'état peuvent conduire à des prédicteurs dont la fonction possède moins d'arguments et donc moins de paramètres que les modèles entrées-sorties **[09]**.

Remarque

D'une manière générale, si le bruit affecte la boucle du modèle hypothèse, la fonction du prédicteur ne doit avoir pour arguments que les sorties passées du processus, et il doit donc être non bouclé. Si le bruit affecte uniquement la sortie, la fonction du prédicteur ne doit avoir pour argument que les sorties passées du prédicteur, il doit donc être bouclé. Enfin, s'il affecte à la fois la sortie et la boucle, la fonction du prédicteur doit porter sur les sorties passées du processus comme du prédicteur**[09].** Et ces considérations restent valables dans le cas d'un modèle hypothèse d'état.

II.8 Conclusion

Dans ce chapitre, nous avons décrit la modélisation dynamique des systèmes non linéaires, où nous avons déterminé les structures des prédicteurs neuronaux associés aux modèles-hypothèses donnés, ceci aussi bien pour des modèles entrée-sortie que pour des modèles d'état. Nous avons également étudié l'estimation des paramètres de ces systèmes et les algorithmes qu'il convient de mettre en œuvre.

Chapitre III

La surveillance par les réseaux de neurones

Chapitre III

La surveillance par les réseaux de neurones

III.1 Introduction

La surveillance est un dispositif passif, informationnel qui analyse l'état du système et fournit des indicateurs. La surveillance consiste notamment à détecter et classer les défaillances en observant l'évolution du système puis à les diagnostiquer en localisant les éléments défaillants et en identifiant les causes premières **[04][25][26].** La surveillance d'un équipement industriel se fait à l'aide de deux fonctions de base: la détection et le diagnostic des défaillances. La détection des défaillances a pour rôle de signaler toute situation autre qu'une situation nominale. En d'autres termes, tout ce qui n'est pas normal doit être classé comme anormal.

Ce chapitre nous permet de donner un aperçu sur la technique de surveillance en utilisant des réseaux de neurones dynamiques, ainsi que les différentes techniques de traitement du signal appliquées pour construire la base d'apprentissage d'un système d'engrenage permettant de faire un diagnostic de défaillance .

III.2 Méthodes de surveillance

Les méthodes de surveillance industrielle sont illustrées sur la Figure III.1.

Un système de surveillance comporte deux fonctions de base : détection et diagnostic. Les méthodologies de surveillance sont généralement divisées en deux groupes : méthodologies de surveillance avec modèle et sans modèle. Les premières se basent sur l'existence d'un modèle formel de l'équipement et se compose essentiellement de deux techniques : méthodes de redondance physique et analytique et méthodes d'estimation paramétrique. La deuxième catégorie de méthodologies est plus intéressante dès lors qu'un modèle de l'équipement est inexistant ou difficile à obtenir. Dans ce cas, on utilise des outils statistiques et des méthodes de reconnaissance des formes.

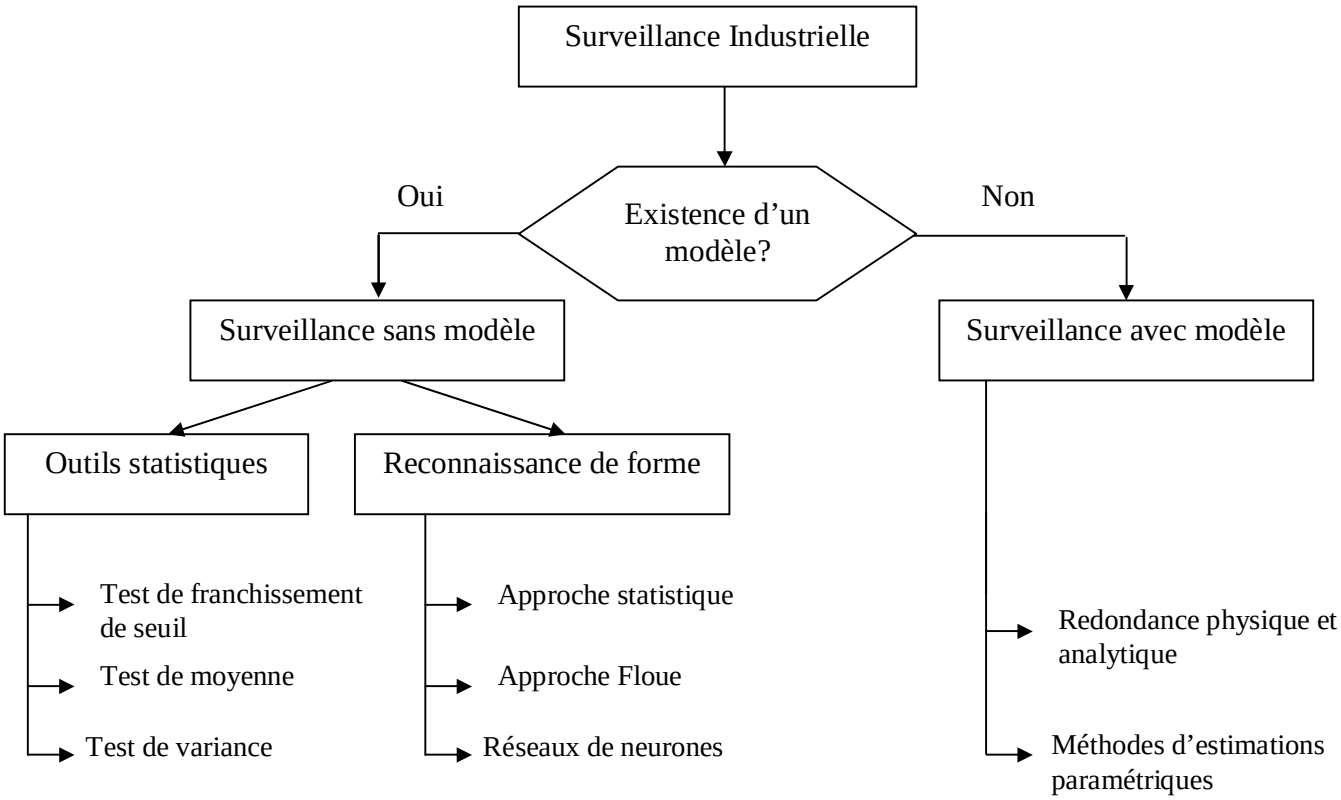


Figure III.1 : Classification des méthodes de surveillance industrielle **[4][43]**.

Les outils statistiques établissent des tests sur les signaux d'acquisition. Ces tests ne sont capables d'assurer que la fonction détection de défaillances. Par contre, les techniques de surveillance par reconnaissance des formes sont plus élaborées par rapport aux simples tests statistiques et sont capables de détecter et de diagnostiquer les défaillances **[04][25][26][43].** C'est dans ce deuxième contexte que se situe notre étude.

La fonction surveillance est alors vue comme une application de reconnaissance des formes. Les formes représentent le vecteur d'entrée composé par les différentes données de l'équipement (données mesurables et qualifiables) et les classes représentent les différents modes de fonctionnement.

III.3 Technique de surveillance par des réseaux de neurones

Les techniques de surveillance par réseaux de neurones sont fondées sur l'existence d'une base de données d'apprentissage et non sur l'existence d'un modèle formel ou fonctionnel de l'équipement. Le principe d'une telle application est de trouver une relation entre des variables d'entrée et des variables de sortie. Les variables d'entrée peuvent être quantifiables (sorties capteurs) ou qualifiables (observations faites par l'opérateur). A partir de ces variables d'entrée, le réseau de neurones donne une réponse caractérisée par deux types de variables de sortie. Des variables de sortie réelles qui peuvent représenter une sortie estimée d'un paramètre de surveillance ou des variables de sortie catégorielles qui représentent l'état de fonctionnement de l'équipement. Selon la nature de ces données en sortie, il existe deux types d'applications. Le premier type est une application *d'approximation de fonctions*, qui consiste à estimer une sortie mesurée de l'équipement. Dans ce cas, les réseaux de neurones sont utilisés en tant qu'approximateur universel et fournissent un modèle sous la forme d'une boîte noire du système. Ceci n'est autre que de l'identification des processus industriels. Le deuxième type d'application considère le problème de la surveillance comme un problème de reconnaissance des formes. La forme à reconnaître est caractérisée par l'ensemble des données (quantifiables et/ou qualifiables) et les classes d'appartenance représentant les différents modes (de fonctionnement ou de dysfonctionnement). Le réseau de neurones doit nous fournir une réponse qui nous renseigne sur l'état de fonctionnement de l'équipement. Il assure la fonction de *détection* (fonctionnement normal ou pas) et la fonction de *diagnostic* (reconnaître un mode de défaillance) **[04].**

III.4 Principe de la reconnaissance des formes (RDF)

La reconnaissance des formes est la science de définition des algorithmes permettant de déterminer à quelle forme ou objet observé est similaire, ou autrement dit, à quelle classe d'objet connus il peut être associé.

Dans la littérature on distingue deux types de RDF :

- Rdf Structurelle : représente des formes à l'aide de grammaires.

- Rdf statistique : représentation numérique des formes ; c'est dans ce deuxième contexte que se situe notre étude car c'est le plus exploité dans le cadre du diagnostic.

En reconnaissance des formes, chaque observation est représentée par un vecteur X de D paramètres réels, appelé vecteur forme, tel que $X^i = [X_1 .....X_D^i]^T$; ce vecteur sera représenté par un point dans cet espace qui est connu sous le nom espace de représentation (figure III.2)**[25][27]** . Les paramètres de ce vecteur traduisent l'état du système étudié. Ils sont issus d'analyses effectuées sur les signaux mesurés par les capteurs implantés sur le système.

L'objectif de la reconnaissance des formes est de décider à laquelle des classes w_1 w_2 w_3 doit être associé une nouvelle forme ; cela revient à rechercher dans quelle zone de l'espace se situe la nouvelle observation (Figure III.3). On peut toujours faire le parallèle avec l'objectif en diagnostic qui est de définir à quel mode de fonctionnement correspond une nouvelle observation **[25]**[4**3]**.

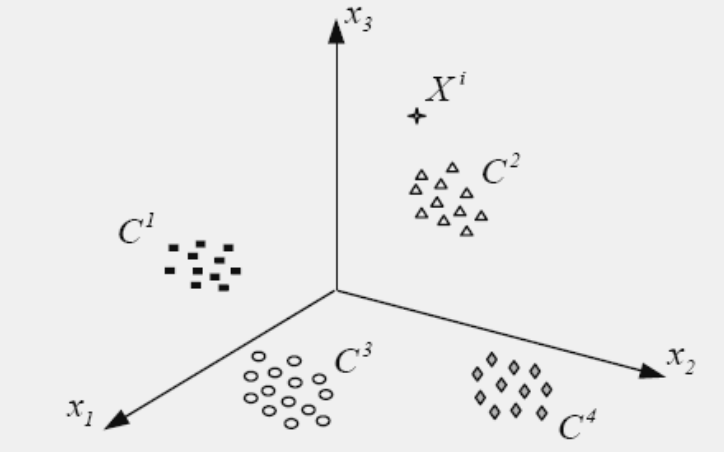


Figure III.2 : Caractéristique d'une observation par un vecteur forme représenté par un point dans R^D (D=3) **[25]**

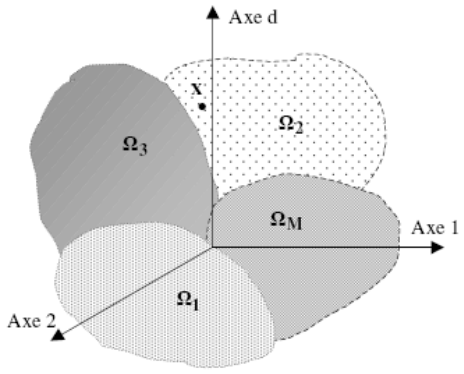


Figure III.3 : Objectif en reconnaissance des formes : associer une nouvelle observation X à l'une des classes **[25]**

III.5 Diagnostic par l'approche reconnaissance des formes

L'élaboration d'un système de diagnostic par reconnaissance des formes se déroule en deux phases : une phase d'analyse comprend l'analyse des données (le prétraitement), la définition d'un espace de représentation, la constitution d'un espace de décision au sein duquel sont définis les différents modes de fonctionnement, et une phase d'exploitation où l'objectif est d'associer aux nouvelles observations les modes de fonctionnement **[25][26][27][28] [43]**

III.5.1 Phase d'analyse

Au cours de laquelle il s'agit d'étudier les informations fournies par les capteurs implantés sur le système. Si ces informations sont sous formes des signaux, il est nécessaire d'en extraire des valeurs (ou paramètres) numériques. Ces paramètres, qui par ailleurs constituent le vecteur forme, doivent pouvoir décrire le comportement du système.

De cette phase d'analyse doit également sortir la définition précise des classes qui représenteront les différents modes de fonctionnement. On dispose alors d'un ensemble de N observations X_1 X_2 ….X_N regroupées en M classes ; c'est l'ensemble d'apprentissage **[25][41].**

La phase d'analyse permet de déterminer d'une part l'espace de représentation et d'autre part l'espace de décision.

III.5.1.1 Détermination de l'espace de représentation

Il s'agit dans cette phase de prétraitement, de construire le vecteur forme. Les D paramètres obtenus sont issus de différentes analyses effectuées sur les signaux recueillis à l'aide des capteurs. L'intervention de l'expert du processus à surveiller est souvent très utile pour aiguiller cette procédure. Ainsi chaque observation effectuée à un instant donné sera caractérisée par l'ensemble des paramètres. Les performances du système de diagnostic dépendront de la pertinence des paramètres calculés ; il est donc préférable de disposer de paramètres variant, de manière significative en fonction des différents états du système. Les N observations (ou vecteurs)(X_1 X_2….X_N) recueillies sur le système constituent l'ensemble d'apprentissage.

Le nombre souvent élevé de paramètres peut être pénalisant en terme de temps de calcul. De plus, tous les paramètres calculés ne seront pas forcément pertinents vis à vis des modes étudiés. Des méthodes doivent donc être mise au point afin de ne conserver que les paramètres les plus représentatifs.

III.5.1.2 Réduction de l'espace de représentation

La réduction de l'espace de représentation permettra de diminuer le temps de classification pour toute nouvelle observation et d'éviter la dégradation des performances de la règle de décision. Les paramètres sélectionnés vont jouer un rôle important dans la phase d'apprentissage; en effet, ceux-ci devront caractériser au mieux les différents modes de fonctionnement. En d'autres termes, les paramètres jugés comme étant les plus pertinents seront ceux qui permettent de distinguer les classes dans l'espace de représentation. La réduction de l'espace de représentation peut être réalisée soit par des méthodes d'extraction de paramètres (tel que la méthode d'analyse en composantes principales ACP), soit par des méthodes de sélection de paramètres (tel que le critère de Fisher) **[25][26][41]** .

III.5.1.3 Détermination de l'espace de décision

Il s'agit au cours de cette étape, de rechercher une structure en classe de l'ensemble des données par la mise en place d'une procédure de classification. On fait appel à des méthodes de classifications. Lorsqu'on connait la classe d'origine de chaque observation

de l'ensemble d'apprentissage, on utilise une classification supervisée. Dans le cas contraire, l'apprentissage des classes est fait en mode non supervisé.

III.5.1.4 Evaluation de la performance de la procédure

Quelle que soit la méthode à utiliser pour la mise en place du processus de diagnostic, il est primordial de justifier la robustesse du classifieur choisi. En effet, des erreurs de diagnostic pourraient découler de mauvaises performances de la procédure de décision. De plus, les performances de la classification sont dépendantes de la pertinence du vecteur forme, autrement dit de la signature de système à analyser.

Deux critères permettent d'évaluer les performances de la règle de décision :

- Le taux d'observations bien classées (T_B) :

$$T_B = \frac{N_B}{N} * 100 \qquad \textbf{(3.1)}$$

- Le taux d'observations mal classées (T_M) :

$$T_M = \frac{N_M}{N} * 100 \qquad \textbf{(3.2)}$$

Avec : N_B : Nombre de bien classées.

N_M : Nombre de mal classées.

N : Nombre total d'échantillons.

III.5.2 Phase d'exploitation

Il s'agit de reconnaitre à quel mode de fonctionnement correspond une nouvelle observation effectuée à un instant donnée, au sien de l'espace de décision. Cette phase consiste à implémenter le système décisionnel qui proposera une solution pour toute nouvelle observation recueillie sur le système.

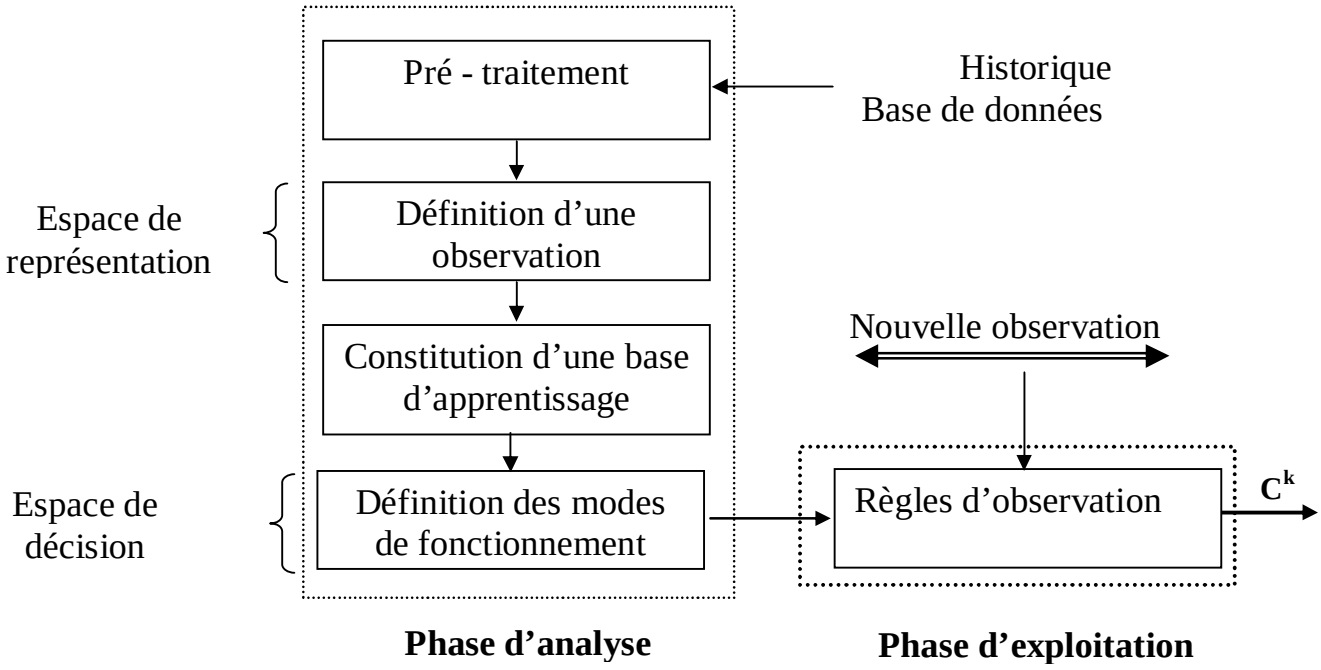


Figure III.4 : Phases d'élaboration d'un système de diagnostic par reconnaissance des formes **[27]**.

III.6 Reconnaissance des formes par les réseaux de neurones

Les réseaux de neurones sont des outils de l'intelligence artificielle, capables d'effectuer des opérations de classification. Leur principal avantage par rapport aux autres outils est leur capacité d'apprentissage et de généralisation de leurs connaissances à des entrées inconnues. Une des qualités de ce type d'outil est son adéquation pour la mise au point de systèmes de surveillance modernes, capables de s'adapter à un système complexe avec reconfigurations multiples. Deux architectures sont très utilisées en surveillance industrielle, à savoir le Perceptron Multi Couches (*PMC*) et les Réseaux à Fonctions de base Radiales (*RFR*). La Figure III.6 montre l'architecture générale d'une application de surveillance par reconnaissance des formes avec réseaux de neurones. L'expert humain joue un rôle très important dans ce type d'application. Toute la phase d'apprentissage supervisé du réseau de neurones dépend de son analyse des modes de fonctionnement du système. Chaque mode est caractérisé par un ensemble de données recueillies sur le système. A chaque mode, on associe une expertise faite par l'expert. Cette association

(ensemble de données - modes de fonctionnement) sera apprise par le réseau de neurones. Après cette phase d'apprentissage, le réseau de neurones associera les classes représentant les modes de fonctionnement aux formes d'entrée caractérisées par les données du système **[04][25][26][27]** .

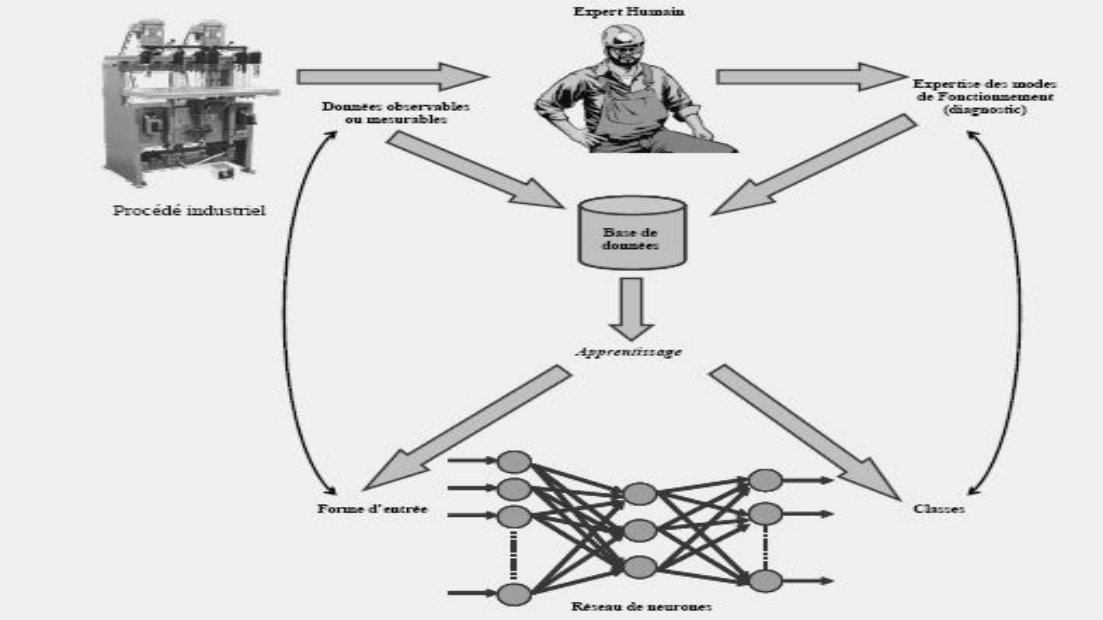


Figure III.6 : Reconnaissance des formes par réseau de neurones **[4]**.

III.7 Apprentissage des classifieurs neuronaux

Un classifieur est ainsi conçu pour attribuer un objet représenté par un vecteur d'entrée à une classe d'appartenance. Prenons l'exemple de la figure III.7, dans le cas d'un problème à deux classes, on peut voir les données en 2 dimensions de 2 différentes classes (classe "carré" et classe "rond") **[28]** .

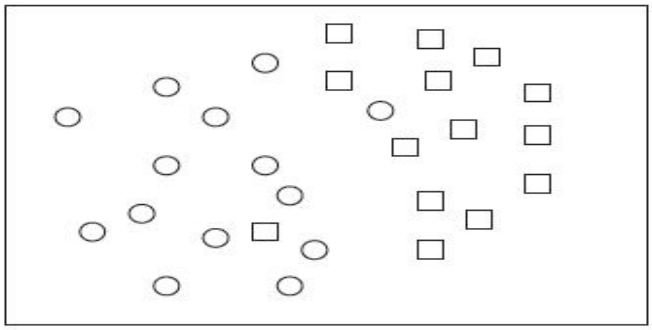


Figure. III.7 : Problème de classification à 2 classes **[28]**.

Il est toujours possible de trouver un classifieur notamment avec les réseaux de neurones fournissant une frontière de décision ou une surface de séparation. Grâce à cette frontière, nous pouvons affecter un individu de l'ensemble d'apprentissage à l'une des classes à partir de sa position par rapport à la surface comme sur la figure III.8.

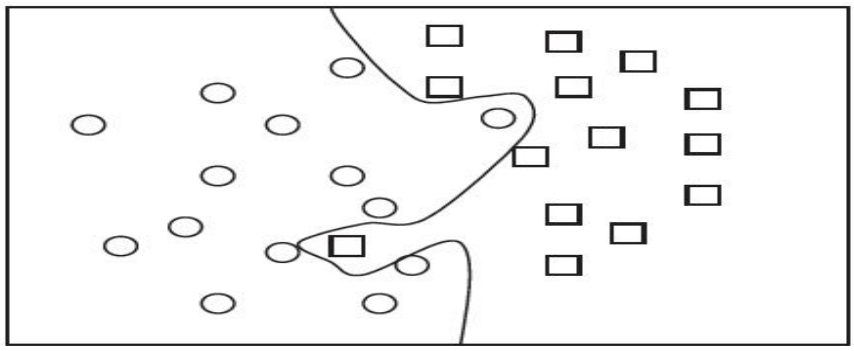


Figure. III.8 : Frontière de décision parfaite sur données **[28].**

La complexité des surfaces de séparation qu'un réseau multicouche est en mesure de réaliser dépond de deux paramètres : le nombre de couches dans les réseaux d'une part et le nombre de neurones par couche d'autre part.

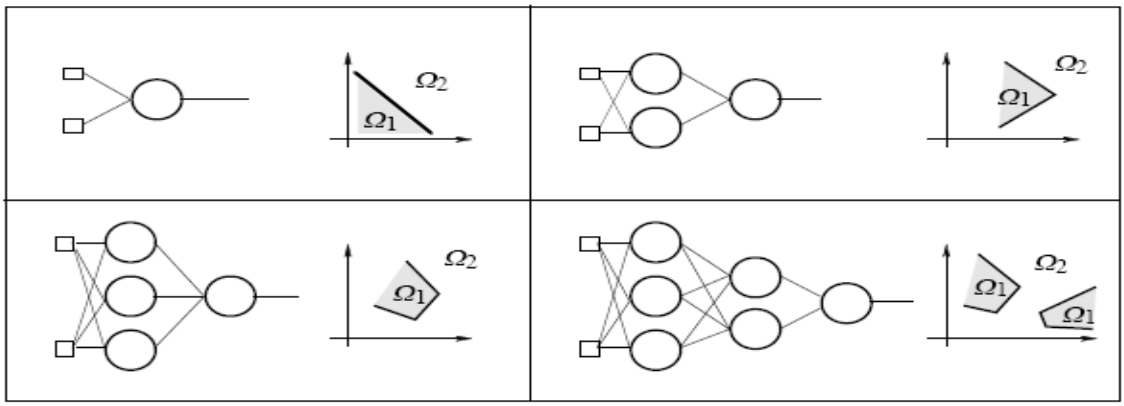


Figure III.9 :L'influence de l'architecture du réseau sur les frontières séparatrices **[28].**

La mise au point consiste à trouver un réseau dont la structure est juste adaptée à la complexité du problème à traiter. Un réseau insuffisamment complexe aura tendance à réaliser une modélisation insuffisante des frontières entre classes. Alors qu'à l' opposé, un

réseau complexe aura tendance à apprendre les données par cœur. Dans les deux cas, cela se traduit par une limitation des capacités de génération.

Maintenant, si nous soumettons à notre classifieur des individus qu'il n'a jamais vu, comment va-t-il réagir ? Sur la figure III.9, on peut voir deux nouveaux individus représentés en pointillé. Le classifieur classe alors le rond comme étant un carré, et le carré comme étant un rond **[28]** .

.

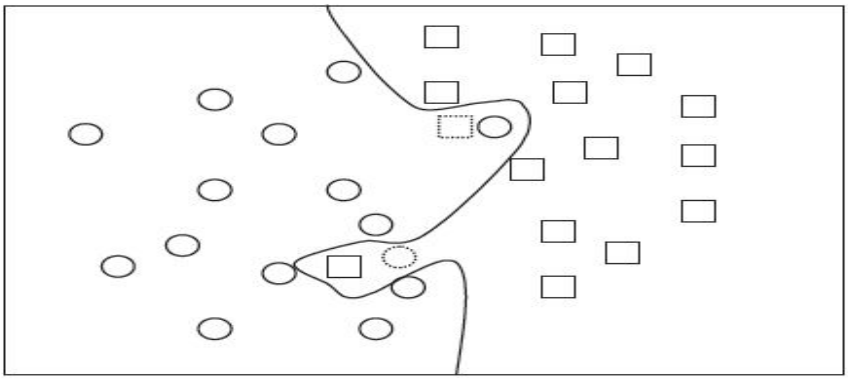


Figure.III.10 : Nouveaux individus mal classes **[28]**.

Dans le cas de cette figure III.10, le classifieur a perdu sa capacité de généralisation, il y a eu surapprentissage : il a trop appris l'ensemble d'apprentissage et ne sait reconnaître que celui-ci sans pouvoir le généraliser. Contrairement à la figure III.10, la figure III.11 présente un classifieur qui a généralisé son ensemble d'apprentissage. Bien entendu, il conduit lui aussi à quelques erreurs, mais à un taux plus faible que le même classifieur ayant surappris **[28].**

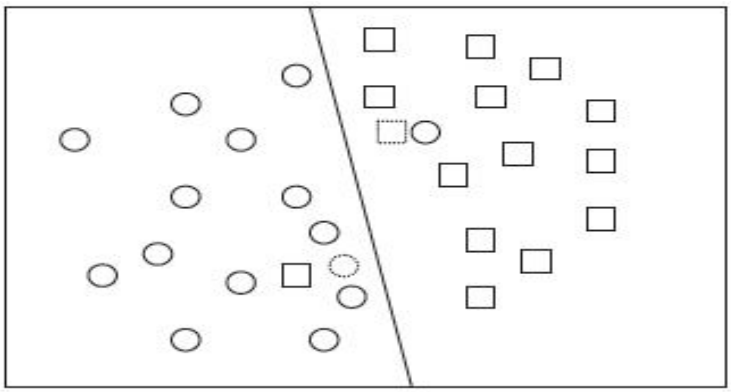


Figure III.11 : Nouveaux individus correctement classés **[28]**.

Les paramètres du classifieur sont estimés comme en modélisation à l'aide d'un ensemble d'apprentissage fourni par un professeur, ou superviseur.

- Pour une classification supervisée, l'ensemble d'apprentissage est constitué d'exemples (objet, classe d'appartenance). Comme en modélisation, on utilise souvent un algorithme minimisant une distance moyenne entre les sorties du classifieur et les sorties désirés de l'ensemble d'apprentissage. Dans ce cas, les frontières de séparation des classes sont déduites sur la base d'un ´échantillon d'individus pour lesquels on connait l'appartenance aux différentes classes.

- Pour une classification non-supervisé, où les objets seuls sont représentés, et ne sont pas étiquetés (il n'y a pas de classe d'appartenance spécifiée par le superviseur), les frontières de séparations sont obtenues suivant certains principes (tels que l'homogénéité des classes ou bien la distance aux différents voisins) à partir des données disponibles et sans aucune information concernant l'appartenance des individus aux différentes classes **[09]**.

III.8 Analyse vibratoire des signaux d'engrenage

Nous avons vu que l'élaboration d'un système de diagnostic nécessite une phase d'analyse pour construire le vecteur forme. Afin d'exploiter cette phase d'analyse, nous présentons une étude des signaux vibratoires d'un système d'engrenage. Nous présentons tout d'abord le banc d'essai utilisé pour la mesure de vibrations, ensuite nous citerons les différentes techniques de traitement du signal appliquées pour déterminer la signature des défauts d'engrenage. Il s'agit de décrire les outils statistiques du signal recueilli, les indicateurs temporels (valeurs efficace, facteur de crête, kurtosis, etc), l'analyse spectrale et cepstrale, l'analyse du signal enveloppe, l'analyse par ondelettes. etc.

III.8.1 Définition d'un système d'engrenages

Les engrenages sont les éléments de machines les plus utilisées dans la construction mécanique. Le rôle des engrenages est de transmettre un mouvement ou une puissance entre deux arbres, avec un rapport de vitesse constant. Les matériaux utilisés varient en fonction des utilisations, mais les plus couramment employés sont l'acier et la

fonte ; les matériaux plastiques sont toutefois de plus en plus employés pour transmettre de faible puissance. Les engrenages peuvent alors présenter des défaillances qui limitent leur durée de vie **[29]-[32]**.

III.8.2 Banc d'essai utilisé pour la mesure de vibrations

Le dispositif d'enregistrement est le suivant**:[30][32]**

Les signaux de vibrations d'engrenage sur lesquelles nous avons fait les tests ont été fournies par le C .E.T.I.M .(Centre des Techniques Industriel Mécanique –France). Le système fonctionne 24h sur 24h sous les conditions suivantes: (Figure III.12)

Le système est composé d'un moteur, d'un réducteur de bouclage de rapport 40/42 et du réducteur testé, de rapport 20/21.Le réducteur étudié est composé de deux roues comportant respectivement 20 et 21 dents (Figure III.13). L'arbre d'entraînement du réducteur sous test tourne à la vitesse de 1 000 trs/min, soit 16,67 Hz.

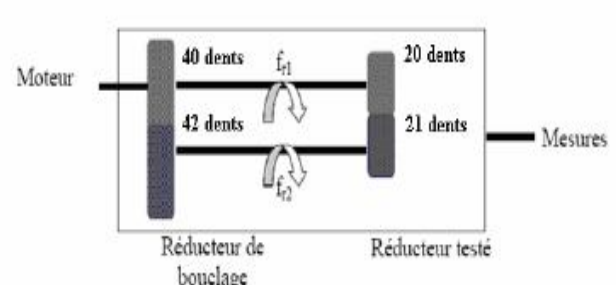


Figure III.12 : dispositif d'enregistrement **[33]**

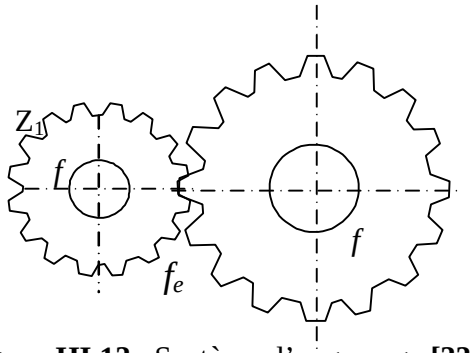


Figure III.13 : Système d'engrenage **[33]**

- Les fréquences de rotation des deux roues sont donc de l'ordre de 16,67 Hz et la fréquence d'engrènement de l'ordre de 330Hz.

-Au cours de l'expérimentation, l'engrenage de test passe de l'état de bon fonctionnement à celui d'engrenage détérioré.

- Un enregistrement a été effectué chaque jour pendant une duré de 13 jours, un signal vibratoire sur le banc d'essai comportant 60160 échantillons avec une fréquence d'échantillonnage de 20kHz. Vu le grand nombre de données (60160 échantillons), il est difficile de les traiter toutes, donc on doit choisir un nombre permettant de ne pas perdre beaucoup d'informations. Pour cela, on doit au moins couvrir une période.

Pour calculer le nombre d'échantillons couvrant une période, on divise la période de rotation sur la période d'échantillonnage, le nombre d'échantillons utile sera Nu=1200 échantillons. On choisit donc un nombre d'échantillons: N=2400 échantillons.

Les vibrations d'un engrenage sont produites principalement par le choc entre les dents des deux roues qui le composent. La vibration, appelée signal d'engrènement, est périodique, et sa fréquence est égale à la fréquence de rotation de l'une des deux roues, multipliée par le nombre de dents de cette roue **[34] [40].**

III.8.3 Les méthodes d'analyse des signaux d'engrenage

Afin de mieux situer notre travail, il a été nécessaire de regarder quelles sont les différentes méthodes de diagnostic utilisées pour la détection d'une anomalie dans les machines tournantes. Comme les chercheurs travaillent sur ce sujet depuis un certain nombre d'années, beaucoup de travaux ont vu le jour **[25][29]-[39]**. Dans cette partie, nous avons choisi de décrire les méthodes les plus couramment rencontrées pour le diagnostic des défauts d'engrenage, en précisant leurs points faibles et leurs points forts.

III.8.3.1 Les méthodes temporelles.

Les méthodes temporelles sont basées sur l'analyse statistique du signal recueilli, Ces méthodes utilisent des indicateurs scalaires qui permettent de suivre l'évolution d'une grandeur dérivant de la puissance ou de l'amplitude crête du signal. Sa valeur peut ne pas avoir de signification intrinsèque, mais c'est son évolution dans le temps qui est significative du défaut **[36].**

III.8.3.1.1 Forme générale du signal : Les représentations temporelles du signal émis par le système pour chaque jour sont données sur les figures III.14 à III.25 :

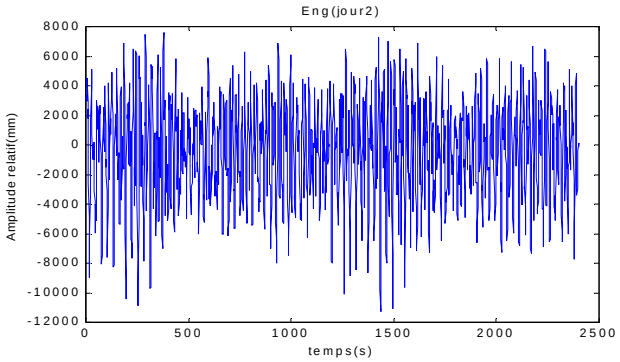


Figure III.14 : Signal du 2ème jour

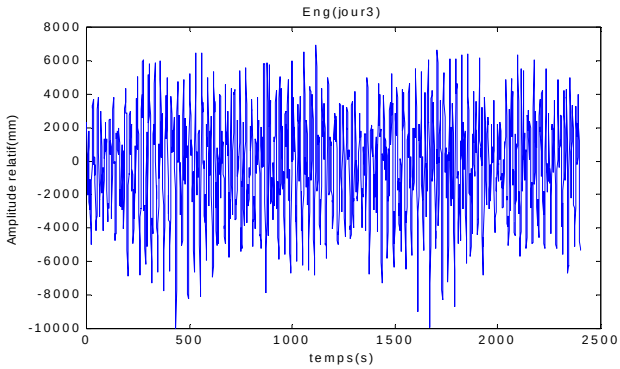


Figure III.15: Signal du 3ème jour

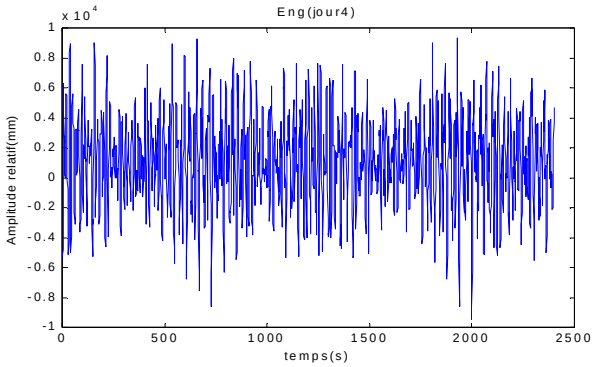


Figure III.16: Signal du 4ème jour

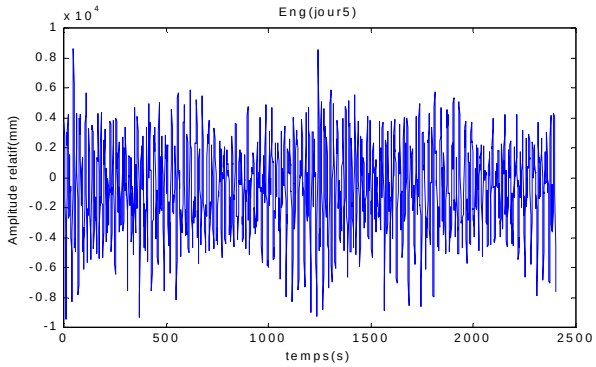


Figure III.17: Signal du 5ème jour

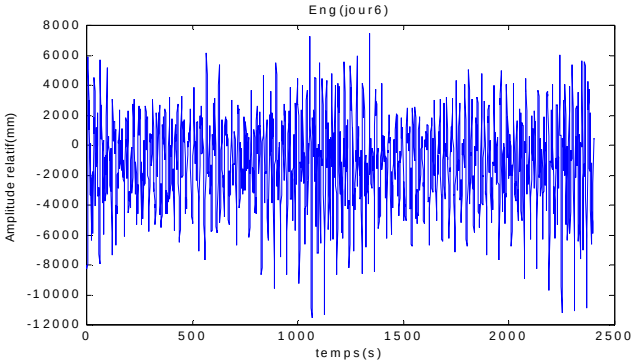


Figure III.18 : Signal du 6ème jour

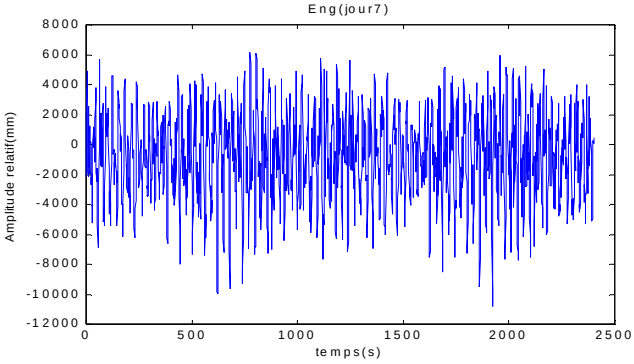


Figure III.19 : Signal du 7ème jour

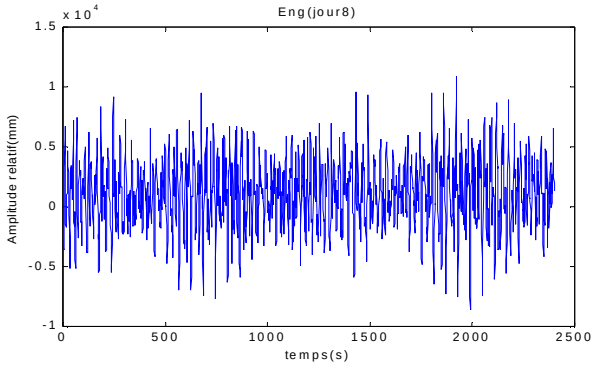


Figure III.20: Signal du 8ème jour

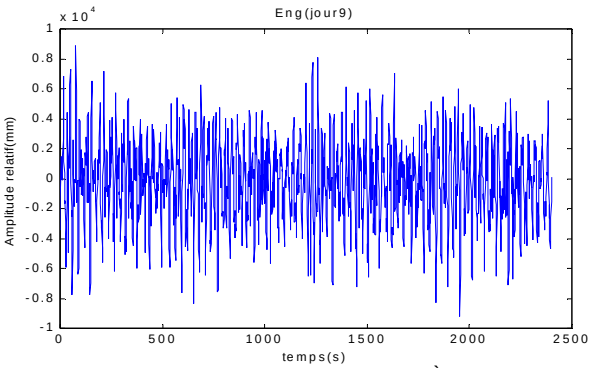


Figure III.21: Signal du 9ème jour

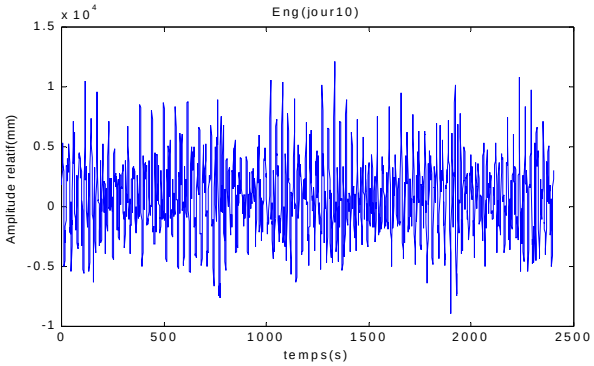


Figure III.22: Signal du 10ème jour

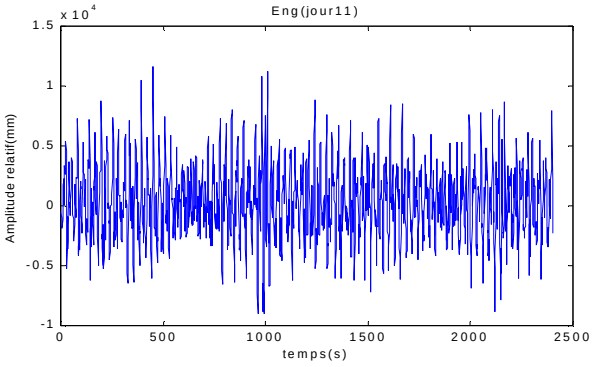


Figure III.23: Signal du 11ème jour

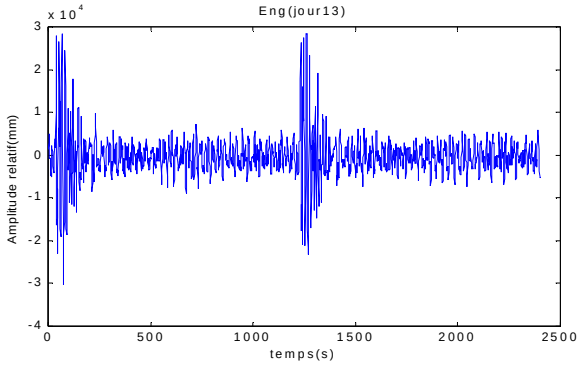


Figure III.24: Signal du 12ème jour

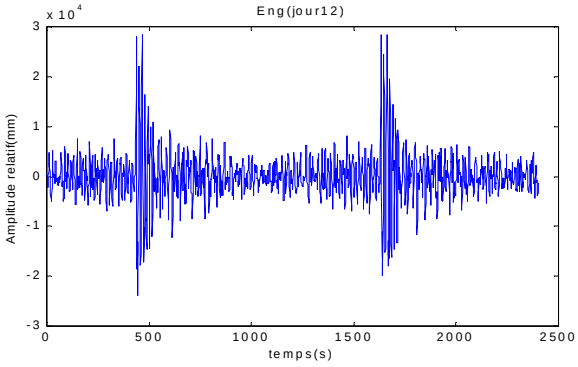


Figure III.25: Signal du 13ème jour

L'observation, dans le domaine temporel, des signaux d'engrenage (figures III.14 à, III.25) montre que l'aspect des signaux enregistrés les onze premiers jours ne donne aucune indication supplémentaire caractérisant l'apparition d'un défaut. Par contre, nous remarquons pour les signaux enregistrés le douzième et le dernier jour qu'un choc se produit à une période correspondant à la période de rotation du système d'engrenage et ayant une amplitude très élevée par rapport à celle du signal prélevé pendant les autres jours qui sont des indicateurs de la présence d'un défaut.

III.8.3.1.2 Calcul des indicateurs

a- La valeur efficace ou valeur RMS (Root Mean Square)

C'est un indicateur scalaire, très caractéristique du signal, vu qu'il a une relation directe avec l'énergie contenue dans celui-ci. Bien qu'il présente des inconvénients, car il ne détecte pas tous les défauts, son efficacité dépend de son bon paramétrage en terme de bandes de fréquences d'analyse liées à la structure même des machines. **[36][37][42].**

$$V_{effecace} = V_{RMS} = \sqrt{\frac{1}{Ne}\sum_{n=1}^{Ne}[x(n)]^2} \quad \textbf{(3.3)}$$

Où x(n) est le signal temporel mesuré, N_e représente le nombre d'échantillons prélevés du signal.

b-Facteur de crête : Il est défini comme étant le rapport entre la valeur crête et la valeur efficace,

$$Facteur\ \ crête = \frac{valeur\ \ crête}{valeur\ \ efficace} = \frac{\sup|x(n)|}{\sqrt{\frac{1}{Ne}\sum_{n=1}^{Ne}[x(n)]^2}} \quad \textbf{(3.4)}$$

Le facteur crête comporte l'avantage de détecter les défauts avant la valeur efficace. Ceci provient du fait que pour un signal d'engrenage sans défaut, le rapport reste sensiblement constant et augmente lorsqu'une dégradation apparaît alors que la valeur crête croit pendant que la valeur efficace reste à peu près constante. **[36][42].**

c- Le Kurtosis

Le Kurtosis est le moment d'ordre 4 normé de la distribution statistique du signal. C'est un indicateur permettant de caractériser la nature impulsive d'un signal et la détection précoce d'un défaut de roulement. Dans le cas d'un roulement sans écaillage, la distribution des amplitudes contenues dans le signal recueilli est gaussienne ;ce qui entraîne une valeur de Kurtosis proche de 3 (Moment d'ordre 4 d'un signal gaussien égal à 3). Lorsqu'un défaut est détecté, sa valeur devient supérieure à 3. L'analyse des défauts de roulement par le Kurtosis peut également être réalisée dans différentes bandes de fréquences liées aux résonances de la structure. **[36][37][42].**

$$Kurtosis = \frac{M_4}{M_2^2} = \frac{\frac{1}{N}\sum_{n=1}^{N}\left(x(n)-\bar{x}\right)^4}{\left[\frac{1}{N}\sum_{n=1}^{N}\left(x(n)-\bar{x}\right)^2\right]^2} \quad \textbf{(3.5)}$$

Où M_4 et M_2 sont les moments statistiques d'ordre 4 et d'ordre 2, *x(n)* est le signal temporel, *x* est la valeur moyenne des amplitudes, *N* est le nombre d'échantillons prélevés dans le signal.

Le Kurtosis doit cependant être utilisé avec beaucoup de précaution, car il est très sensible aux chocs. Ce qui oblige un emploi du Kurtosis dans un environnement peu complexe afin de ne pas commettre d'erreurs de diagnostic **[36][37].**

III.8.3.2 les méthodes fréquentielles

Les méthodes fréquentielles sont basées sur la transformée de Fourier. La connaissance des fréquences caractéristiques permet d'identifier et de localiser les défauts issus des composants mécaniques en analysant leur spectre **[36].**

III.8.3.2.1 L'analyse en fréquence.

L'analyse en fréquence est devenue l'outil fondamental pour le traitement des signaux vibratoires. Elle s'appuie sur la transformée de Fourier (équation 3.6) qui permet le passage du domaine temporel au domaine fréquentiel. Cette représentation permet de connaître le contenu spectral d'énergie ou de puissance, présent dans le signal à la fréquence f, et donc de détecter la présence d'un défaut générant un choc périodique à une fréquence de défaut. Dans la pratique, on utilise la transformée de Fourier discrète rapide (FFT) (équation 3.7) sur des signaux numérisés :

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi ft}\, dt \qquad \textbf{(3.6)}$$

Où X(f) est la transformée de Fourier, x(t) est le signal temporel, t est le temps, *f* est la fréquence.

$$X(k\Delta f) = \frac{1}{N}\sum_{n=0}^{N-1} x(nt_e) e^{-j2\pi k\frac{n}{N}} \qquad \textbf{(3.7)}$$

Où *X(kΔf)* est la transformée de Fourier discrète rapide, t_e est la période d'échantillonnage du signal temporel, *n* est le numéro de l'échantillon, Δ*f* est l'intervalle entre deux raies fréquentielles, N est le nombre d'échantillons prélevés.

La formule de Parseval est **[38]** :

$$\sum_{n=-\infty}^{+\infty} |x(n)|^2 = \int_{-1/2}^{1/2} |X(f)|^2 \, df \qquad \textbf{(3.8)}$$

Elle permet de rendre compte du fait que, le premier membre étant par définition l'énergie temporelle du signal, $|X(f)|^2$ s'interprète comme la distribution de l'énergie le long de l'axe des fréquences. On appelle densité spectrale de puissance (DSP) ou spectre de puissance, le carré du module de la transformée de Fourier, rapportée au temps d'observation (équation 3.9).Elle est la représentation fréquentielle (spectre de puissance) la plus utilisée dans le diagnostic vibratoire des éléments des machines tournantes et en particulier des engrenages **[36]**.

$$DSP(f) = \frac{|X(f)|^2}{d} \qquad \textbf{(3.9)}$$

Où *DSP(f)* est la densité spectrale de puissance, *X(f)* est la transformée de Fourier du signal, *d* est la durée d'observation.

Cette approche d'analyse spectrale est très efficace du point de vue complexité de calcul et produit des résultats raisonnables pour un grand nombre de signaux aléatoires. Toutefois l'approche FFT possède des limitations de performance. Parmi ces limitations, on cite l'inefficacité de la FFT due à la non stationnarité et la complexité des systèmes mécaniques qui donnent des défauts caractérisés par des variations complexes du spectre **[30][31].** L'autre limitation et la plus importante est la résolution fréquentielle, c'est-à-dire la possibilité de distinguer les réponses spectrales a deux ou plusieurs signaux. Une autre limitation est due au fenêtrage des données qui se manifeste lors du traitement par FFT. Ce fenêtrage a pour effet de cacher certains détails spectrales qui sont réellement présents **[30].**

III.8.3.3 L'analyse d'enveloppe.

L'analyse d'enveloppe appelée **H.F.R.T.** (High Frequency Resonance Technique) est une méthode qui permet de détecter des chocs périodiques à partir des résonances de structure. Cette méthode consiste à filtrer un signal autour d'une fréquence de résonance identifiable sur un spectre large bande (en général 0-20 KHz). Le signal ainsi obtenu est ensuite

redressé puis filtré ou démodulé. On obtient alors un signal enveloppe auquel on applique une transformée de Fourier.

Le principe de l'HFRT peut être décomposé en quatre étapes **[38]** :

- *Identification d'une fréquence de résonance*
- *Sélection de la zone à démoduler*

On réalise un filtrage passe bande autour de la résonance choisie.

_ *Calcul de l'enveloppe du signal temporel correspondant par la transformée de Hilbert*

Soit s(t) le signal temporel, la transformée de Hilbert de ce signal est définie par :

$$\tilde{s}(t) = H(s(t)) = \frac{1}{\pi}\int_{-\infty}^{+\infty}\frac{s(\tau)}{t-\tau}d\tau \qquad \textbf{(3.10)}$$

$$\tilde{s}(t) = s(t) * \frac{1}{\pi t} \qquad \textbf{(3.11)}$$

Soit z(t) le signal analytique associé :

$$\tilde{s}(t) = s(t) + i\tilde{s}(t) \qquad \textbf{(3.12)}$$

L'enveloppe du signal est définie par :

$$E(t) = |z(t)| = \sqrt{s(t)^2 . \ddot{s}(t)^2} \qquad \textbf{(3.13)}$$

_ *Calcul du spectre de cette enveloppe*

On effectue alors la transformée de Fourier sur le signal E(t).

III.8.3.4 Le Cepstre

Le cepstre se définit comme la transformée de Fourier inverse du logarithme décimal de la transformée de Fourier. Il a pour but d'identifier et de quantifier toutes les structures périodiques contenues dans le spectre. Il permet de définir des indicateurs adaptés à la détection précoce des défauts induisant, à des stades plus ou moins avancés, des énergies vibratoires que les indicateurs issus de techniques traditionnelles mettraient difficilement en évidence. Le cepstre et ses dérivées représentent les amplitudes des composantes dont les quéfrences correspondent aux périodes de répétition des chocs

induits par les défauts de la machine surveillée. Il est largement utilisé pour la détection de défauts d'engrenage et dans une moindre mesure pour les défauts de roulements **[36].**

$$C\left[s(t)\right] = C(\tau) = TF^{-1}\left[LogTF\ \left[s(t)\right]\right] \quad \textbf{(1.14)}$$

La variable ζ du cepstre a la dimension d'un temps. Elle est appelée « quéfrence » (anagramme du mot fréquence). Elle représente les périodes d'oscillations des réponses impulsionnelles de la structure et les périodes de répétition ou de modulation des forces d'excitation [**38][42].**

III.8.3.5 Analyse en Ondelettes

Une ondelette est une fonction élémentaire, à valeurs réelles ou complexes, très concentrée à la fois en temps et en fréquence.

Du point de vu mathématique, les ondelettes sont des fonctions élémentaires sur lesquelles on va décomposer le signal x(t). Ces fonctions vont permettre une analyse temps-fréquence. On peut également les introduire à partir d'une seule fonction ψ (t) appelée ondelette analysante (ou mère) ; ensuite, on construit les ondelettes $\psi_{a,b}$ (t) par dilatation et par translation:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right) \quad \textbf{(3.15)}$$

La transformée en ondelettes du signal $x(t)$ peut être écrite sous la forme:

$$TOC_x(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty} x(t)\psi^*\left(\frac{t-b}{a}\right)dt \quad \textbf{(3.16)}$$

III.8.3.5.1 Représentation temps- échelle du signal vibratoire

Les scalogrammes obtenus par la transformée en ondelettes, appliqués au signal vibratoire émis par le système d'engrenages pendant les 13 jours d'expérimentation du banc d'essai, sont représentés sur les figures III.26-III.33. L'algorithme de Morlet a été utilisé pour ses bonnes performances.

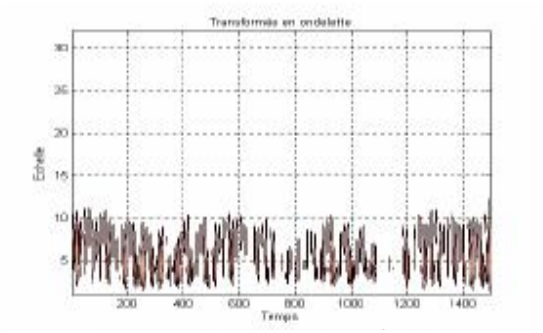


Figure III.26 :TO du signal du 2eme jour

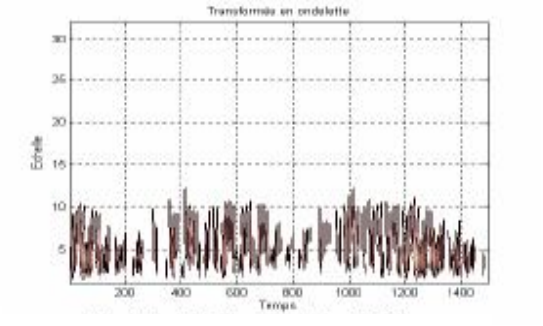


Figure III.27:TO du signal du 5eme jour

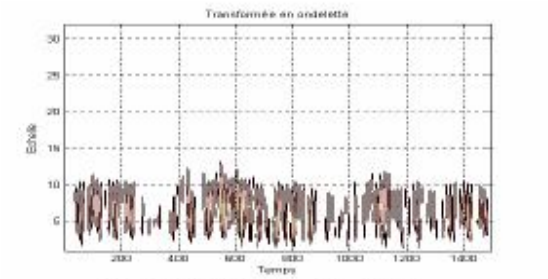


Figure III.28 :TO du signal du 7eme jour

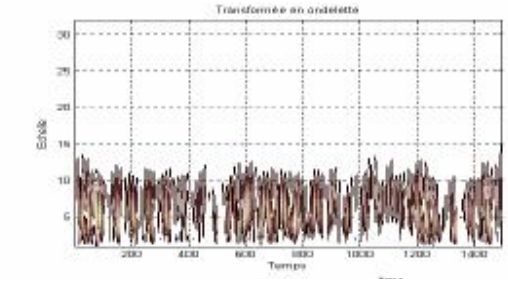


Figure III.29 :TO du signal du 9eme jour

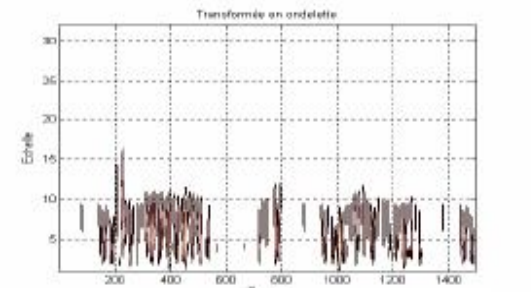


Figure III.30 :TO du signal du 10eme jour

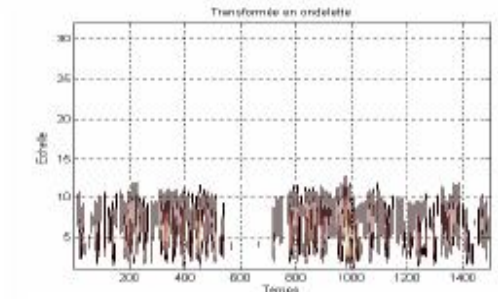


Figure III.31 :TO du signal du 11eme jour

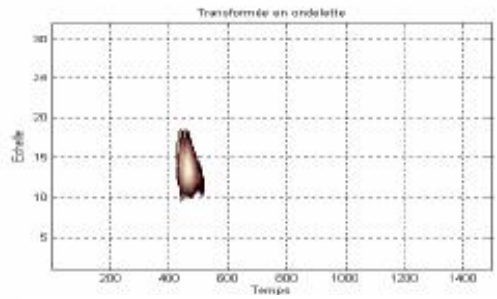


Figure III.32 : TO du signal du 12eme jour

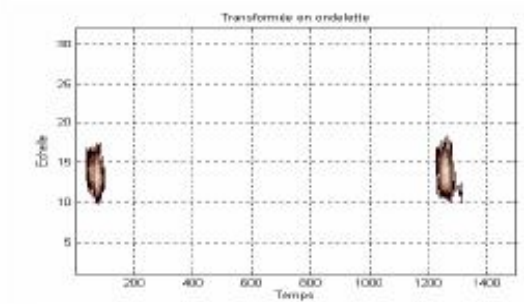


Figure III.33 : TO du signal du 13eme jour

Dans le domaine de la transformée en ondelettes, les coefficients sont stables et de même ordre de grandeur jusqu'au 9ème jour. Au 10ème jour, les coefficients commencent à changer leurs régimes (absence d'une partie de la bande sur le scalogramme). C'est un indice précoce qui indique que le système d'engrenage va subir des défauts. Le système d'engrenage présente un défaut au 12ème jour qui se traduit par un changement complet de la localisation des coefficients de la transformée en ondelettes **[33].**

III.9 Conclusion

Dans ce chapitre, nous avons décrit le principe de la surveillance d'équipements industriels par des réseaux de neurones, ou on a vu les différentes étapes de l'élaboration d'un système de diagnostic par reconnaissance des formes, en utilisant des réseaux de neurones. Ces derniers peuvent fournir une solution intéressante pour des problématiques de surveillance, et servent à reconnaître le mode de fonctionnement ou de dysfonctionnement à partir d'une analyse des données représentant le vecteur forme qui caractérise chaque mode d'un système d'engrenage.

Les résultats obtenus par les techniques présentées précédemment sont utilisés pour construire le vecteur forme afin d'entraîner des réseaux de neurones dynamiques pour le diagnostic des défauts d'engrenages.

Chapitre IV

Application des réseaux de neurones dynamiques à la modélisation des systèmes non linéaires

Chapitre IV

Application des réseaux de neurones dynamiques à la modélisation des systèmes non linéaires

IV. 1 Introduction

Afin de mettre en œuvre les algorithmes et les procédures présentés dans les chapitres précédents pour la modélisation dynamique à l'aide des réseaux de neurones décrits dans l'espace d'état, nous présentons dans ce chapitre deux applications, où nous avons choisi d'étudier un processus simulé, et un processus réel. L'approche d'état est adoptée dans cette étude pour décrire les réseaux dynamiques. Une étude comparative avec un modèle d'entrée-sortie sera considérée.

IV. 2 Étude d'un processus décrit dans l'espace d'état

IV.2.1. Présentation

Nous avons choisi d'étudier le processus décrit dans la référence **[13]** pour étudier les performances des réseaux de neurones dynamiques décrits dans l'espace d'état. Nous allons démontrer l'influence des hypothèses concernant l'origine des bruits sur la modélisation du processus et le prédicteur choisi soit pour la représentation d'état ou d'entrée sortie.

Les équations d'état et d'observation décrivant ce processus sont :

$$\begin{cases} x_{1p}(k+1) = 1.145\, x_{1p}(k) - 0.549 x_{2p}(k) + 0.584\, u(k) \\ x_{2p}(k+1) = x_{1p}(k) /(1 + 0{,}01\, x_2 p(k))^2 + 0.330 u(k) \\ y(k) = 4\, .tanh(x_{1p}(k)/4) \end{cases} \qquad \textbf{(4.1)}$$

Où $u(k)$ est le vecteur commande à l'instant k, $y_p(k)$ le vecteur sortie mesurée du processus et les $x_p(k)$ les vecteurs d'état.

Les séquences de commande utilisées pour l'apprentissage et l'estimation des performances comportent 400 échantillons, constitués de créneaux d'amplitudes aléatoires dans l'intervalle [-5, +5]

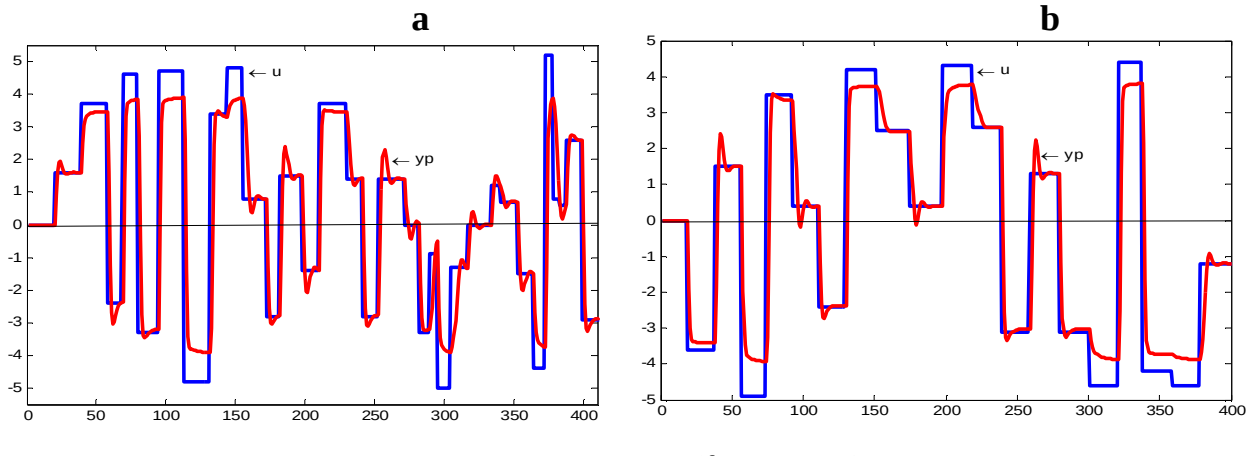


Figure IV.1 : Séquences de modélisation.
a) Séquences d'apprentissage. b) Séquences de test.

IV.2.2 Modélisation d'état

IV.2.2.1 Processus sans bruit (déterministe)

Le prédicteur neuronal associé au modèle d'état est un réseau bouclé constitué d'une couche d'entrée à 3 neurones (une entrée imposée et deux entrées d'état) , d'une couche cachée à 2 neurones dont la fonction d'activation est la tangente hyperbolique, et une couche de sortie à 3 neurones linéaires (une sortie du modèle et deux sorties d'état) (figure IV.2).

Les équations décrivant ce prédicteur sont :

$$\begin{cases} x_1(k+1) = \varphi_1(\zeta_1(k), \zeta_2(k), u(k); C) \\ x_2(k+1) = \varphi_2(\zeta_1(k), \zeta_2(k), u(k); C) \\ y(k+1) = \Psi(\zeta_1(k), \zeta_2(k), u(k); C) \end{cases} \quad \textbf{(4.2)}$$

Les fonctions φ_1, φ_2 et Ψ sont réalisées par un réseau de neurones complètement connecté de paramètres C **[20].**

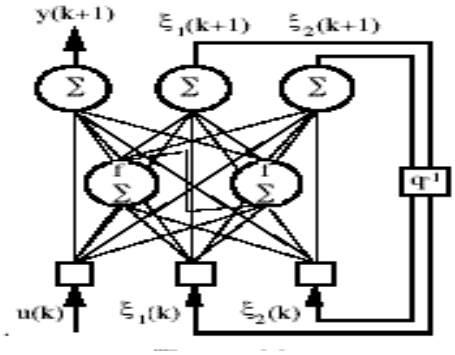


Figure IV.2:Prédicteur neuronal d'état pour un modèle déterministe.

L'algorithme d'apprentissage

La séquence d'apprentissage est construite de la manière suivante : le vecteur des variables du réseau à l'instant discret k est constitué des Ne entrées exogènes, ainsi que des mesures des Ns variables d'états : $x_k = \left[x_k^1 .... x_k^{Ne}, x_k^{1,p} ..... x_k^{Ns,p}\right]^T$

Le caractère récurrent se traduit par le fait que le vecteur des entrées comprend les Ns sorties d'état à l'instant précédent. Une fois construite cette séquence d'apprentissage, la mise en œuvre d'algorithmes d'optimisation itératifs se fait de la même manière que dans le cas d'un réseau statique, notamment en ce qui concerne le calcul du gradient de la fonction de coût. L'apprentissage est réalisé avec un algorithme de gradient basé sur le coût total, suivi de l'algorithme de quasi-Newton.

- Mise en œuvre de l'algorithme de quasi-Newton

On rappelle le principe de l'algorithme de quasi-Newton qui consiste à modifier les paramètres **θ** à chaque itération selon la direction du gradient, et dans le sens opposé selon :

$$\theta(i) = \theta(i-1) + \mu(i)D(i) \qquad \textbf{Avec :} \quad D(i) = -M(i)\frac{\partial J}{\partial \theta}$$

Où M(i) est une matrice définie positive donnant la direction de descente à partir du gradient $\frac{\partial J}{\partial \theta}$, cette matrice est une approximation de l'inverse du Hessien .

On met à jour la matrice M pour l'itération suivante à l'aide de la formule :

$$M(i+1) = M(i) + \left(\frac{1 + \gamma^T M(i) \gamma}{\delta^T \gamma} \right) . \frac{\delta \delta^T}{\delta^T \gamma} - \frac{\delta \gamma^T M(i) + M(i) \gamma \delta^T}{\delta^T \gamma}$$

- A l'itération i, on dispose de θ (i-1), M(i), $\frac{\partial J}{\partial \theta}\Big|_{\theta(i-1)}$, pour i =1, θ(0) est la valeur obtenue à la dernière itération de l'algorithme du gradient, et l'on impose M(1) = I_q .

Itération i

1) calcul d'une valeur μ(i) à l'aide d'un algorithme de minimisation unidimensionnelle du type décrit précédemment [voir annexe]. Ce calcul demande plusieurs évaluations du cout J.

2) Calcul de θ(i) selon :

$$\theta(i) = \theta(i-1) + \mu(i) D(i) \qquad \textbf{Avec :} \quad D(i) = -M(i) \frac{\partial J}{\partial \theta}$$

3) Calcul du nouveau gradient et des grandeurs qui en dépendent :

$$\frac{\partial J}{\partial \theta}\Big|_{\theta(i)} \qquad , \gamma = \frac{\partial J}{\partial \theta}\Big|_{\theta(i)} - \frac{\partial J}{\partial \theta}\Big|_{\theta(i-1)} \qquad et \quad \delta = \theta(i) - \theta(i-1)$$

4) tests d'arrêt de la minimisation portant, en particulier, sur le module du gradient et sur le module de sa variation γ ;

5) Pour améliorer la convergence, et indépendamment de la positivité de M, celle-ci doit être remise périodiquement à l'identité : M(i+1) =I_q ,par exemple toutes les 100 itérations .

6) Test de positivité de la matrice M(i+1) que l'on calcule pour l'itération suivante.

- **Si** $\delta^T \gamma > 0$, on calcule la matrice M (i+1) suivante, qui est définie positive :

$$M(i+1) = M(i) + \left(\frac{1 + \gamma^T M(i) \gamma}{\delta^T \gamma} \right) . \frac{\delta \delta^T}{\delta^T \gamma} - \frac{\delta \gamma^T M(i) + M(i) \gamma \delta^T}{\delta^T \gamma}$$

- **Si** $\delta^T \gamma < 0$: M(i+1) = I_q, la matrice identité.

Description de l'apprentissage et du test du réseau

L'apprentissage a été effectué avec le prédicteur neuronal bouclé de la figure IV.2. Pour étudier les performances des différents réseaux, nous avons défini au préalable certains critères des performances tel que l'erreur quadratique moyenne obtenu sur l'ensemble de la base de données d'apprentissage noté EQMA et l'erreur quadratique moyenne obtenue sur l'ensemble de la base de données de test notée EQMT. EQMA est donnée par l'expression :

$$EQMA = \frac{1}{2.N_A}\sum_{k=1}^{N_A}\left(y_p(k) - y(k)\right)^2$$

Où y(k) est la sortie réelle, $y_p(k)$ est la sortie désirée. N_A est le nombre d'échantillons d'apprentissage.

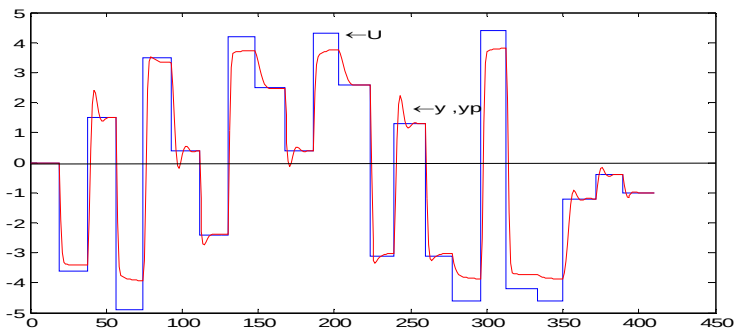
Et EQMT est donnée par l'expression :

$$EQMT = \frac{1}{2.N_T}\sum_{k=1}^{N_T}\left(y_p(k) - y(k)\right)^2$$

Où N_T est le nombre d'échantillons de l'ensemble de test.

Les variances des erreurs de prédictions obtenues pour la base de données d'apprentissage de la figure IV.1 – a sont: **EQMA =1.69.10^{-7}.** Et pour la base de données de test de la figure IV.1- b : **EQMT =3.9.10^{-7}.**

Les résultats obtenus sont illustrés sur la figure IV.3, (la sortie du réseau (y), l'erreur de prédiction ($e = y - y_p$), les variables d'états du processus (x_{p1}, x_{p2}) et du modèle(x_1,x_2). Le prédicteur neuronal basé sur un modèle d'état a réussi à bien estimer la sortie du processus. Toutefois les états obtenus par ce réseau ne sont pas identiques à ceux du processus ; ce qui est conformité avec la théorie (voir chapitre II).



(a)

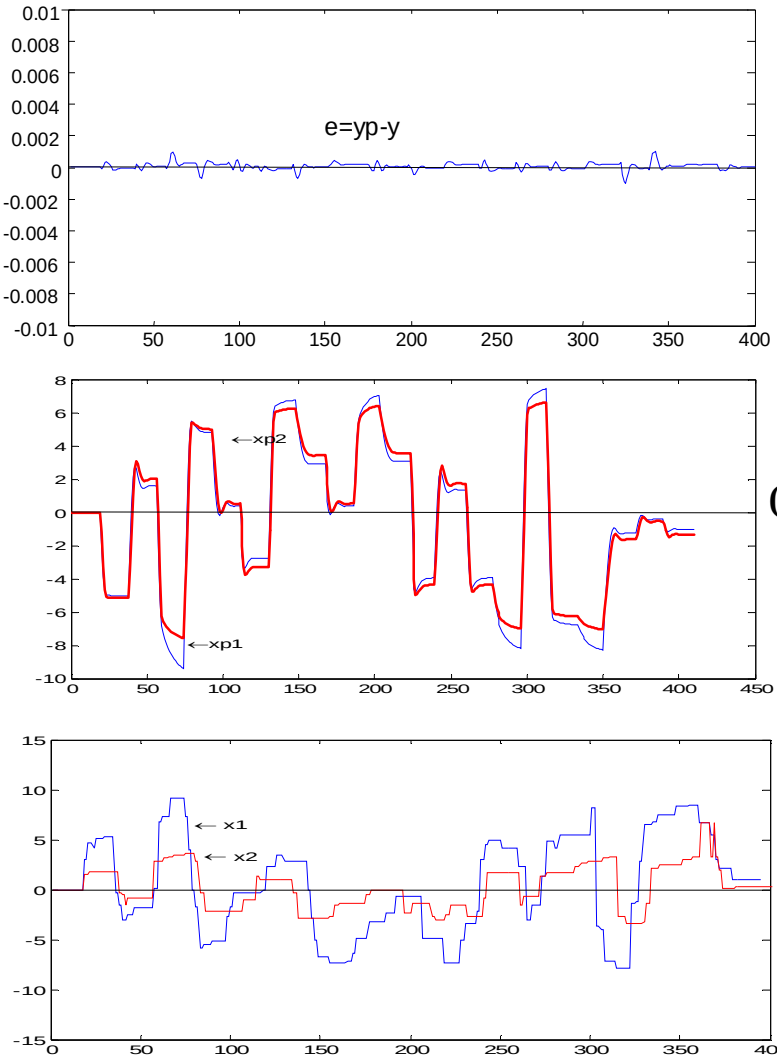


Figure IV.3:

Résultats obtenus pour le modèle déterministe, modélisation avec un pédicteur d'état :
a) - le signal de commende, la sortie du processus (*yp*), la sortie du réseau (*y*),
b) - l'erreur de prédiction ($e = y - y_p$),
c) - les variables d'états du processus (x_{p1}, x_{p2})
d) - les variables d'états du modèle(x_1,x_2).

Autres exemples de test

Pour vérifier la qualité de l'apprentissage de ce réseau, nous avons appliqué au réseau, dont les poids sont fixés aux valeurs finales, d'autres exemples de test.

Exemple 1 : *La* séquence d'entrée utilisée pour tester le réseau prédicteur est constituée de créneaux d'amplitudes aléatoires entre ±1, de longueur 1023. Elle est représentée sur la

figure VI.4-a. La variance des erreurs de prédictione obtenus pour cette base de données est: **EQMT =1.15.10^{-6} .** Les résultats obtenus sont illustrés sur la figure IV.4.

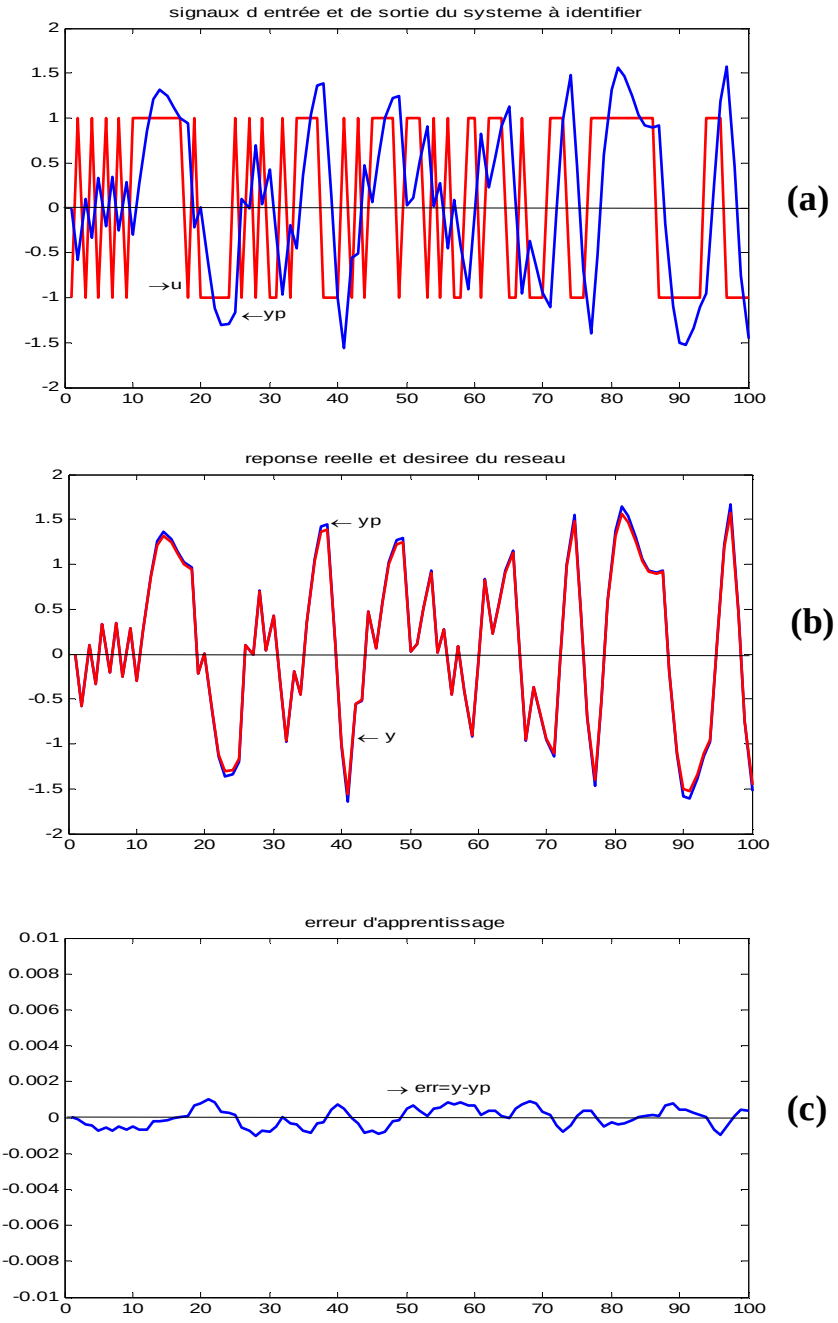


Figure IV.4:

Résultats obtenus avec autres données de test :

a) - le signal de commande, la sortie du processus (*yp*).

b) - la sortie du processus (*yp*), et du modèle (y)

c) - l'erreur de prédiction ($e = y - y_p$),

Exemple 2

Une autre séquence d'entrée, utilisée pour tester le réseau prédicteur, est représentée sur la figure IV.5-a.

La variance des erreurs de prédictions obtenue pour cette base de données est: **EQMT =2.34.10$^{-8.}$** Les résultats obtenus sont illustrés sur la figure IV.5.

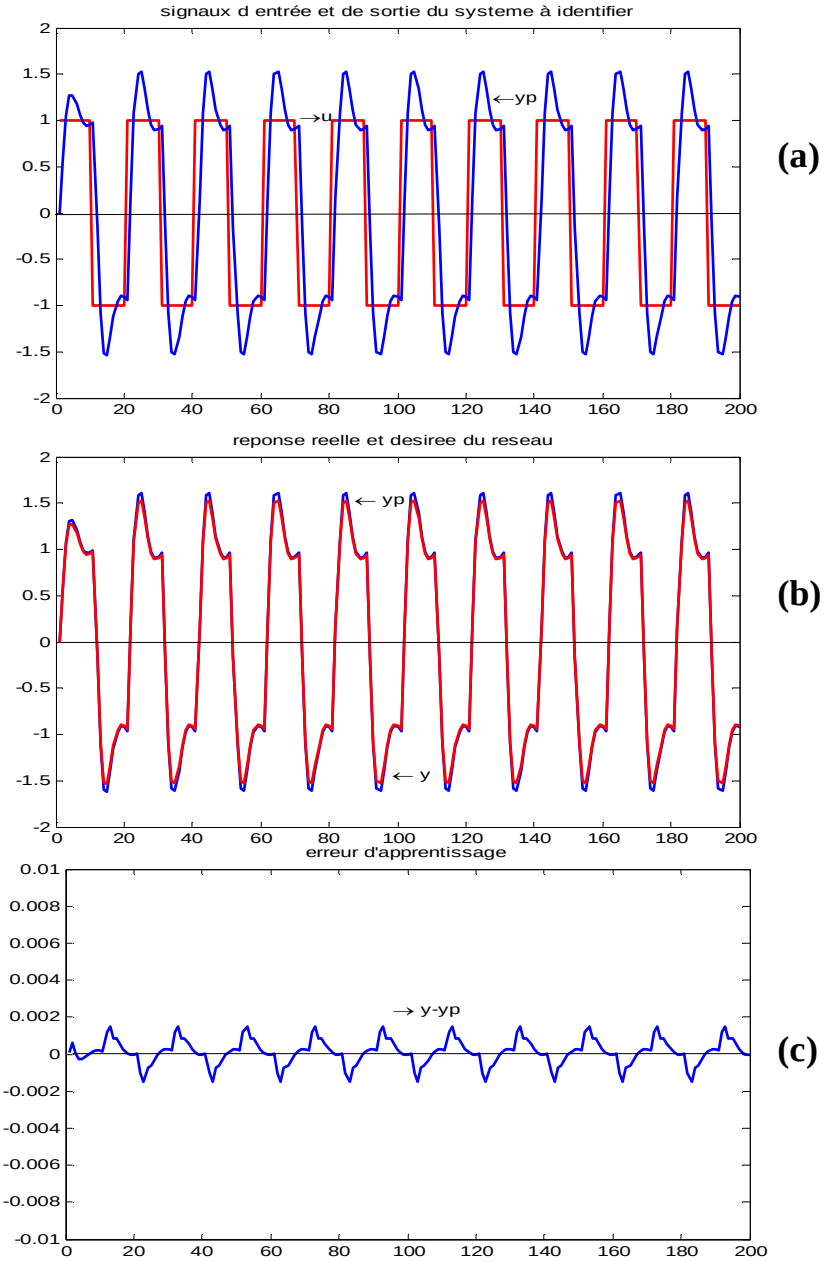


Figure IV.5:

Résultats obtenus avec autres données de test :

a) - le signal de commande, la sortie du processus (*yp*).

b) - la sortie du processus (*yp*), et du modèle (y)

c) - l'erreur de prédiction ($e = y - y_p$).

VI.2.2.2 Processus avec bruit (stochastique)

Lors de la modélisation d'un processus, la manière dont le bruit intervient dans le processus n'est pas connue. On fait donc successivement des hypothèses sur ce type de bruit, on effectue l'apprentissage en fonction d'une hypothèse retenue, et l'on compare les résultats obtenus avec d'autres hypothèses. C'est donc ce que nous allons faire dans ce travail.

IV.2.2.2.1 Processus avec bruit blanc affectant la sortie (NBSX)

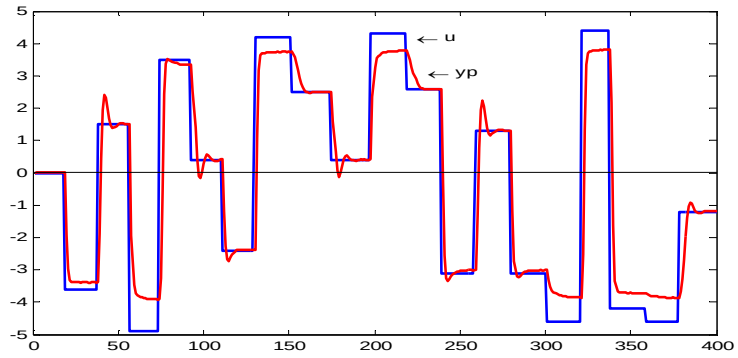
Les équations d'état et d'observation décrivant ce processus sont :

$$\begin{cases} x_{1p}(k+1) = 1.145\, x_{1p}(k) - 0.549 x_{2p}(k) + 0.584\, u(k) \\ x_{2p}(k+1) = x_{1p}(k) / (1 + 0{,}01\, x2p(k))^2 + 0.330 u(k) \\ y(k) = 4\, .tanh(x_{1p}(k)/4) + w(k) \end{cases} \qquad \textbf{(4.3)}$$

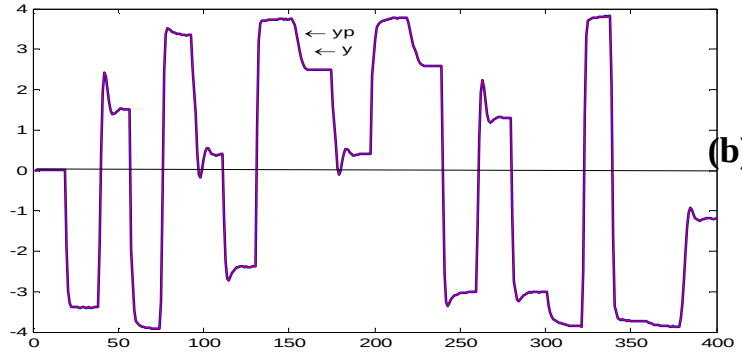
Où *w(k)* est un bruit blanc avec une valeur moyenne nulle et une variance de 3.10^{-2}.

Nous utilisons les mêmes séquences de commande utilisées pour l'apprentissage et test du processus déterministe (figure.IV.1), ainsi que le même predicteur neuronal (figure IV.2). Nous obtenons des erreurs d'estimation de variances: **EQMA =2.8.10^{-2}** et **EQMV = 3.2.10^{-2}** qui sont sensiblement égales à la variance de bruit **3.10^{-2}**. La puissance de l'erreur de modélisation est bien égale à celle du bruit .Ce qui constitue le meilleur résultat de modélisation que l'on puisse obtenir d'après les références **[07] [09] [13] et [14].** Donc ce prédicteur neuronal réalise une excellente approximation du prédicteur théorique (donc c'est un bon prédicteur).

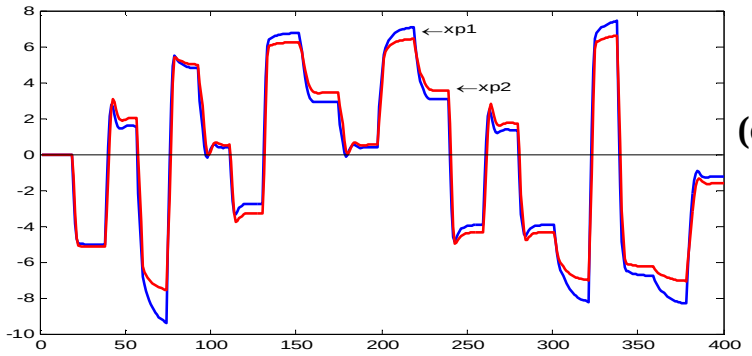
Les résultats obtenus avec ce processus sont illustrés sur la figure IV.6.



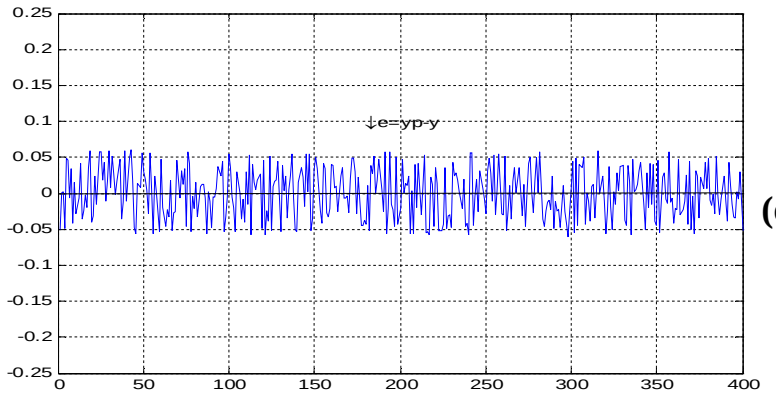
(a)



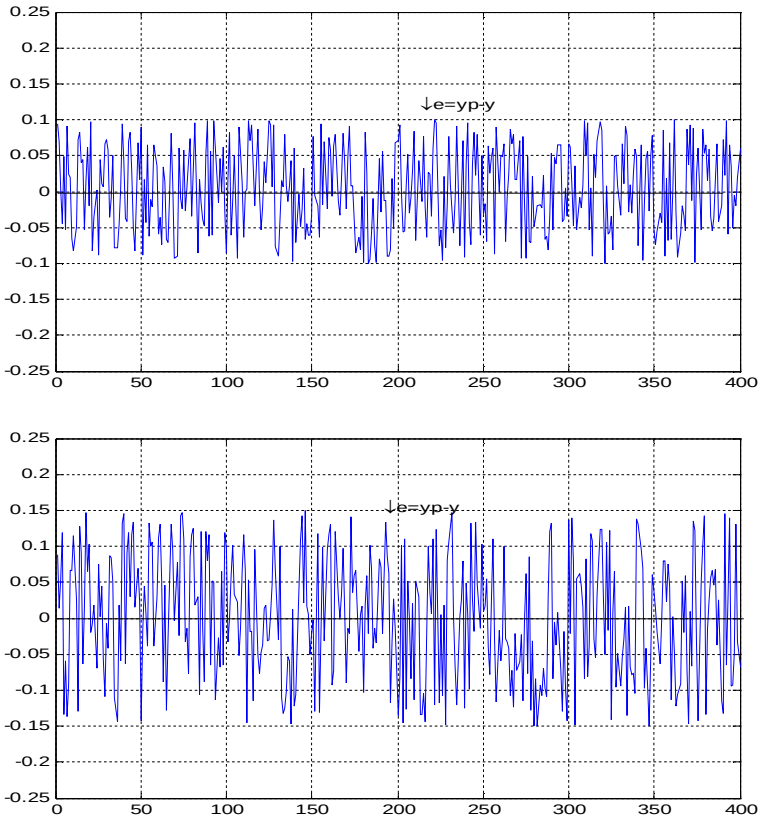
(b)



(c)



(d1)



(d2)

(d3)

Figure IV.6:
Résultats obtenus pour le modèle NBSX, modélisation avec un pédicteur d'état
a) - le signal de commande, la sortie du processus (*yp*).
b) - la sortie du processus (*yp*), et du modèle (y)
c) - les variables d'états du processus (x_{p1}, x_{p2})
d)- l'erreur de prédiction ($e = y - y_p$), pour 3 niveaux de bruit (0.25, 0.35, 0.45 respectivement)

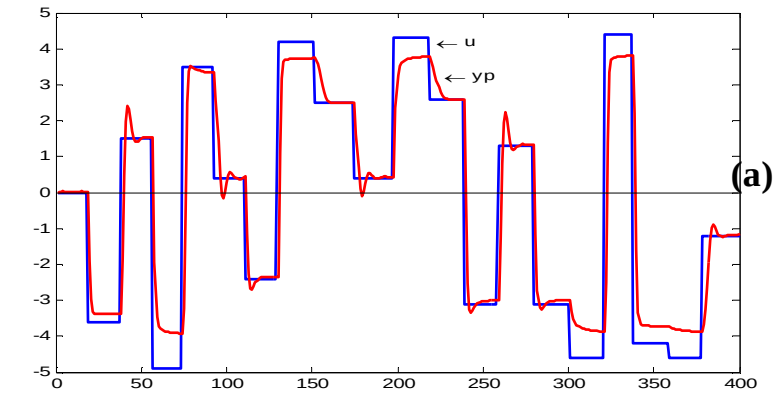
IV.2.2.2.2 Processus avec un bruit blanc affectant l'état (BE)

Les équations d'état et d'observation décrivant ce processus sont :

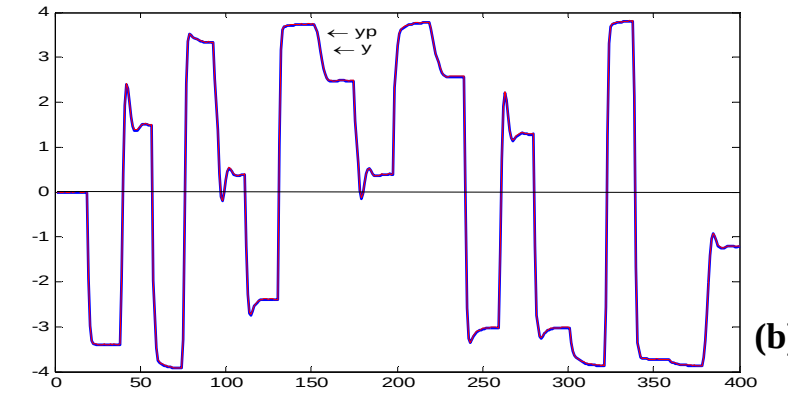
$$\begin{cases} x_{1p}(k+1) = 1.145\, x_{1p}(k) - 0.549 x_{2p}(k) + 0.584\, u(k) \\ x_{2p}(k+1) = x_{1p}(k) / (1 + 0{,}01\, x2p(k))^2 + 0.330 u(k) + w(k) \\ y(k) = 4\, .tanh(x_{1p}(k)/4) \end{cases} \quad \textbf{(4.4)}$$

Ou *w(k)* est un bruit blanc avec une valeur moyenne nulle et une variance de 3.10^{-2} .

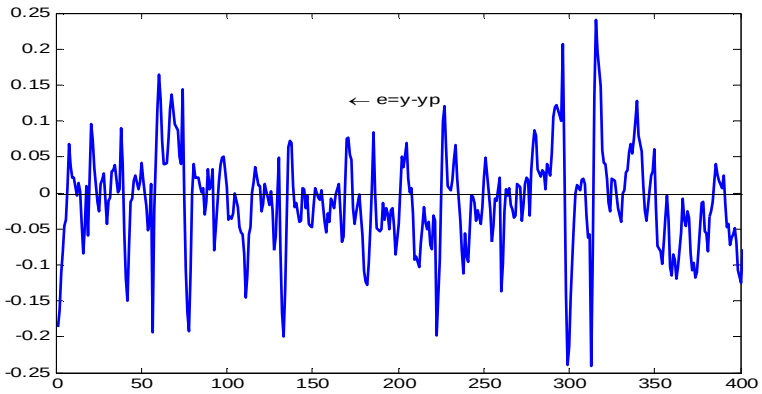
Sachant que le bruit affecte l'état du processus en utilisant toujours le même prédicteur neuronal du processus déterministe. Les résultats obtenus avec ce processus sont illustrés sur la figure IV.7.



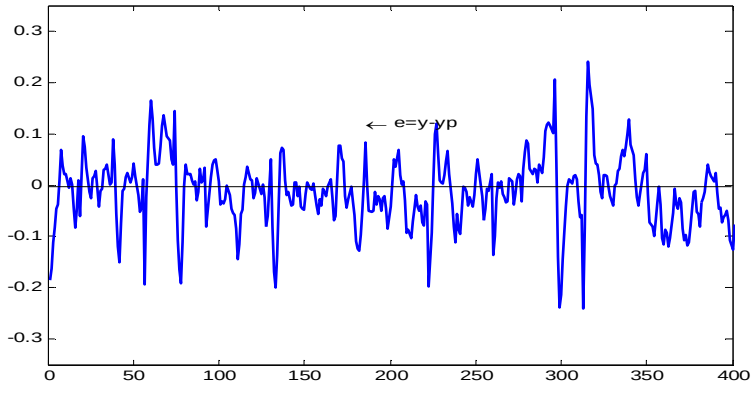
(a)



(b)



(c1)



(c2)

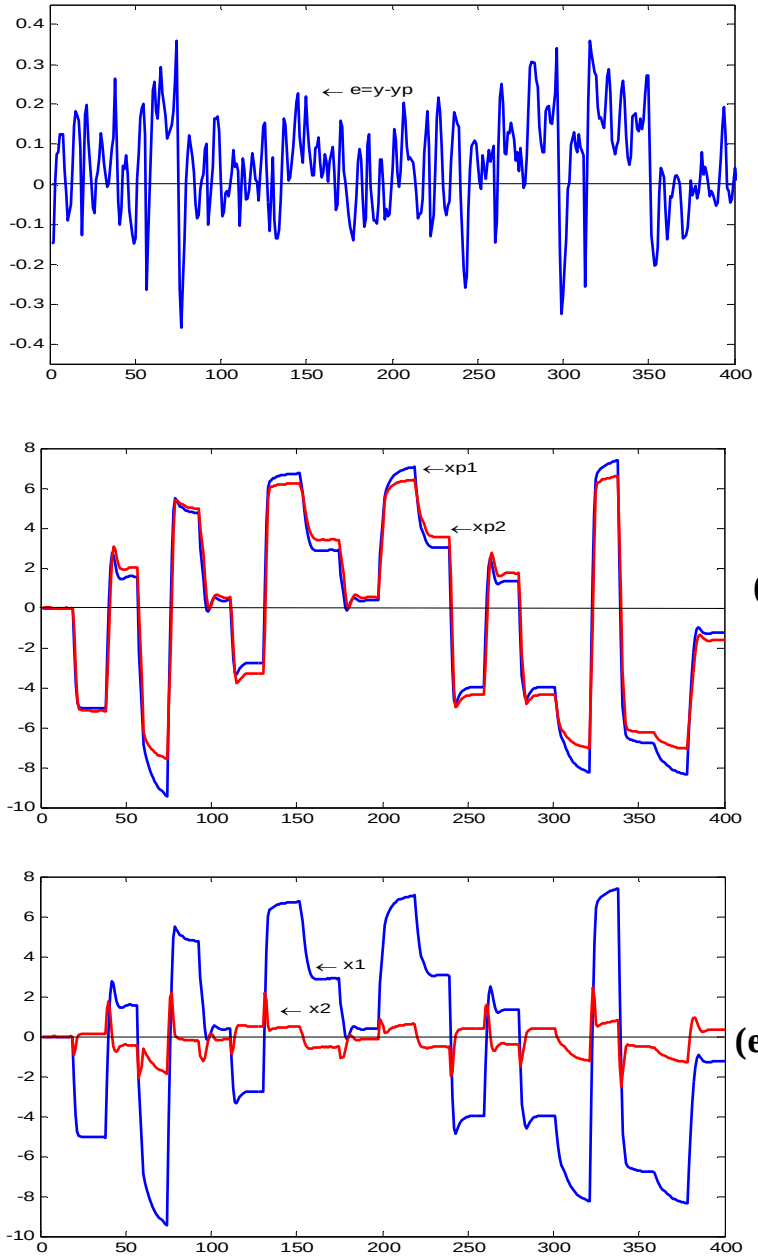


Figure IV.7:

Résultats obtenus pour le modèle BE, modélisation avec un pédicteur d'état

a) - le signal de commande, la sortie du processus (*yp*),

b) - la sortie du processus (*yp*), et du réseau (*y*),

c) - l'erreur de prédiction ($e = y - y_p$), pour 3 niveaux de bruit (0.25, 0.35, 0.45 respectivement)

d) - les variables d'états du processus (x_{p1}, x_{p2})

e) - les variables d'états du modèle(x_1,x_2).

Rappelons que le prédicteur neuronal du processus déterministe est un prédicteur bouclé, et que le bruit affecte l'état du processus ; et nous avons vu au chapitre II que si le bruit affect l'état du processus, le prédicteur doit être non bouclé. D'après les résultats obtenus: **EQMA=8.4.10^{-2} et EQMT=2.6.10^{-2}**, Nous observons bien que l'erreur ne vérifie pas les caractéristiques d'un bruit blanc. Donc cette hypothèse ne permet pas d'obtenir des résultats optimaux (donc c'est un mauvais prédicteur) d'après les références **[07] [09] [13] et [14].**

IV.2.3 Modélisation entrée- sortie

Nous reprenons la même procédure précédente avec des modèles entrées- sorties.

IV.2.3.1 Processus sans bruit (déterministe)

Le système d'apprentissage utilise un prédicteur bouclé entrée- sortie. Ce prédicteur décrit par l'équation (4.6) est représenté sur la figure IV.8, l'algorithme d'apprentissage est semi-dirigé car les valeurs de ses entrées ne sont imposées qu'au début de la fenêtre de la fonction de coût.

$$y(k+1)=h(y(k),..y(k-1),u(k),C). \quad \textbf{(4.6)}$$

Pour identifier ce prédicteur, nous utilisons donc un réseau de neurones bouclé de la forme : $y(k+1) = \psi_{RN}(y(k),y(k-1), u(k), C)$ **(4.7)**

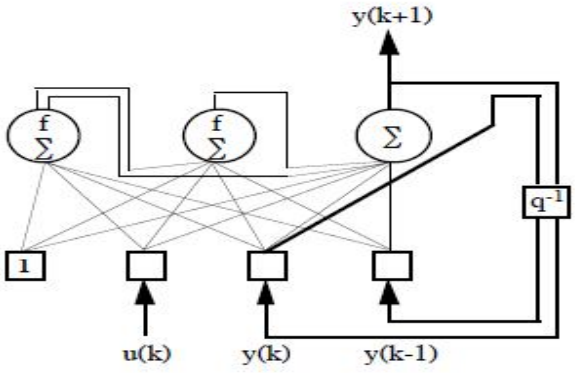


Figure IV.8:Prédicteur neuronal d'entrée-sortie utilisé.

Description de l'apprentissage du réseau

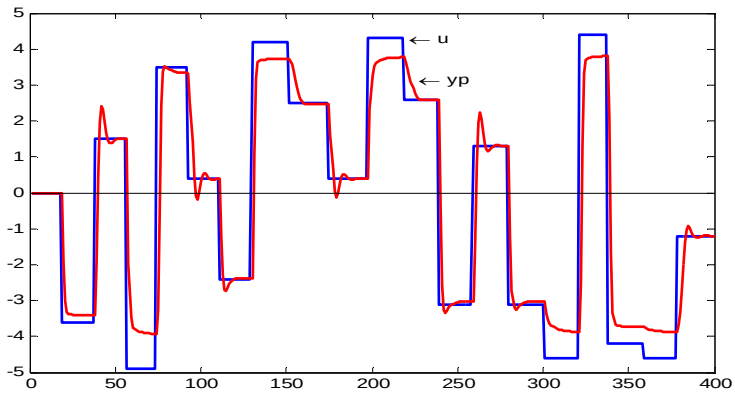
Nous avons vu que ce type de prédicteur impose un algorithme d'apprentissage semi-dirigé ; où le caractère récurrent du réseau de neurones est pris en considération durant la phase d'apprentissage. C'est-à-dire au lieu de fixer les variables d'état du réseau à leurs valeurs mesurées, on leur attribue les valeurs estimées par le modèle à l'instant précédent.

Nous utilisons les mêmes séquences de commande utilisées pour l'apprentissage et le test du processus déterministe du modèle d'état (figure.IV.1).

Nous effectuons l'apprentissage de telle manière qu'il réalise les meilleurs résultats. On prend le prédicteur bouclé, et on augmente progressivement le nombre de neurones jusqu'à ce qu'on obtient les performances désirées. Les résultats obtenus avec ce processus sont illustrés sur Le tableau IV.1 et sur la figure IV.9.

Nombre de neurones cachés	EQMA	EQMT
3	$4.5.10^{-1}$	$9.28.10^{-1}$
4	$3.2.10^{-3}$	$3.6.10^{-3}$
5	$9.3.10^{-2}$	$1.47.10^{-3}$

Tableau IV.1 Résultats obtenus pour le modèle déterministe.



a

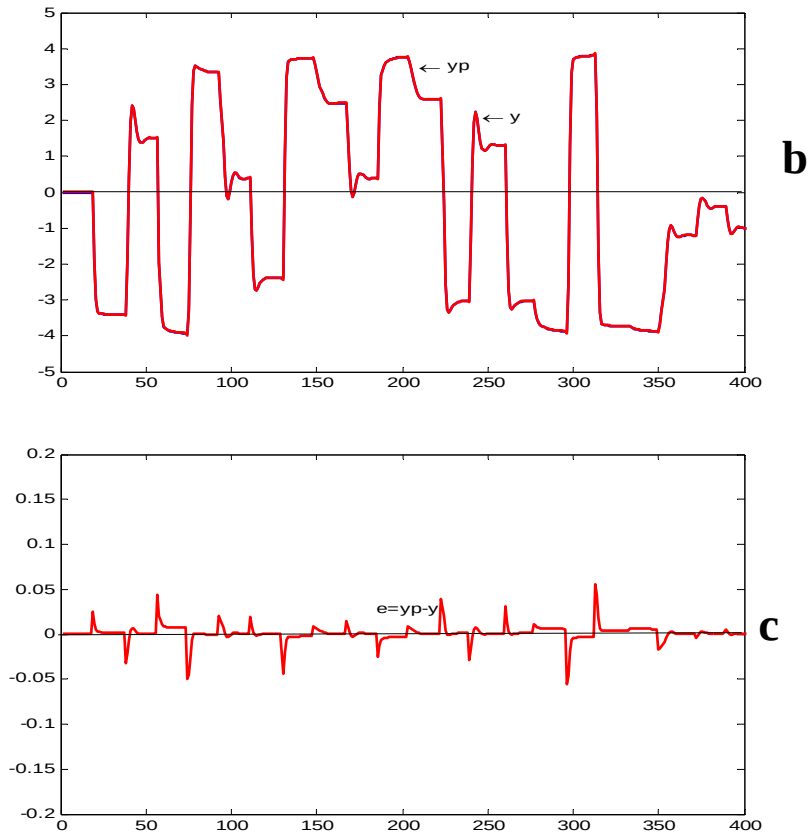


Figure IV.9:
Résultats obtenus pour le modèle déterministe, modélisation avec un pédicteur entré-sortie :
a) - le signal de commande, la sortie du processus (*yp*).
b) - la sortie du processus (*yp*), la sortie du réseau (*y*).
c) - l'erreur de prédiction ($e = y - y_p$),

Autres exemples de test

Afin de vérifier la qualité de l'apprentissage de ce réseau, nous avons appliqué au réseau, dont les poids sont fixés aux valeurs finales, d'autres exemples de test.

Exemple 1

La séquence d'entrée utilisée pour tester le réseau prédicteur est constituée de créneaux d'amplitudes aléatoires entre ±1, de longueur 1023. Elle est représentée sur la figure IV.10-a.

La variance des erreurs de prédictions obtenue pour cette base de données est: **EQMT =1.52.10^{-3}**

Les résultats obtenus sont illustrés sur la figure IV.10.

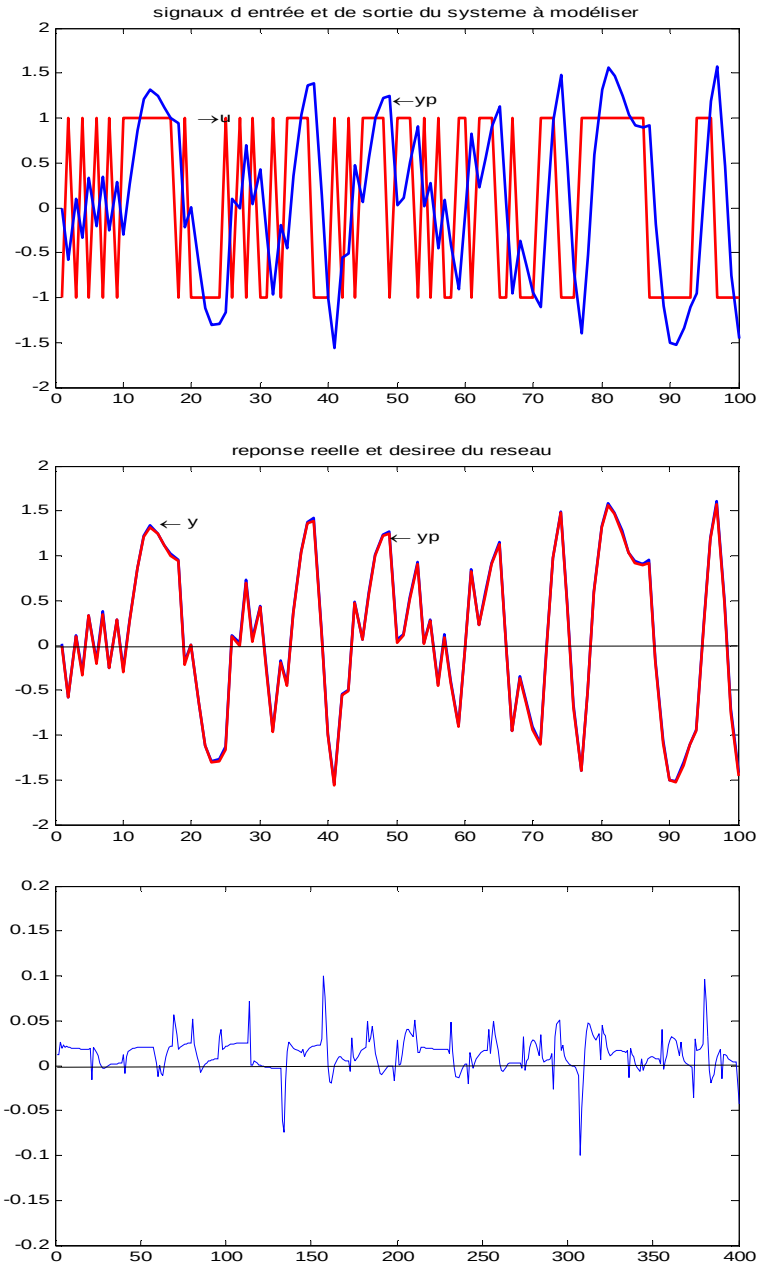


Figure IV.10:

Résultats obtenus avec autres données de test :

a) - le signal de commande, la sortie du processus (*yp*).

b) - la sortie du processus (*yp*), et du modèle (y).

c) - l'erreur de prédiction ($e = y - y_p$).

Exemple 2 : Une autre séquence d'entrée utilisée pour tester le réseau prédicteur est représentée sur la figure VI.11-a. La variance des erreurs de prédictions obtenue pour

cette base de données est: **EQMT =1.84.10^{-3}.** Les résultats obtenus sont illustrés sur la figure IV.11.

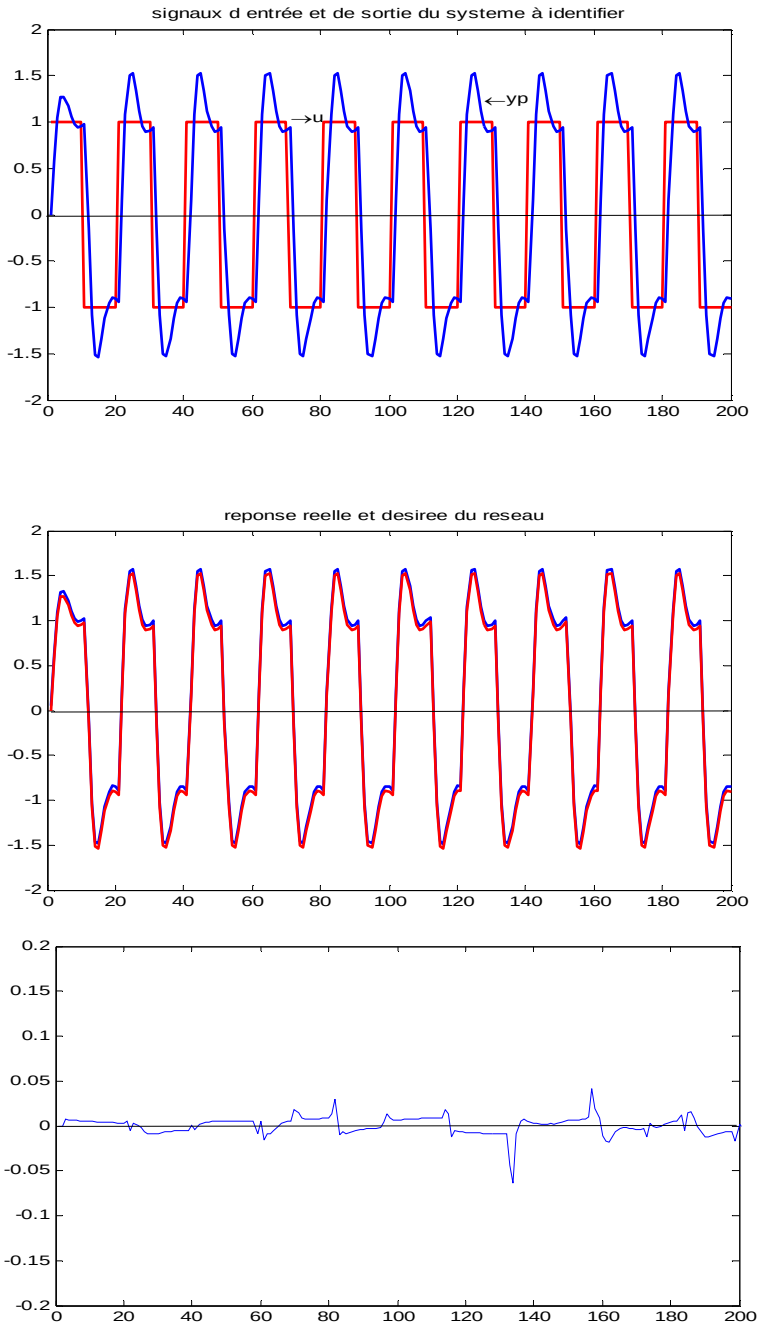


Figure IV.11:

Résultats obtenus avec autres données de test :

a) - le signal de commande, la sortie du processus (*yp*).

b) - la sortie du processus (*yp*), et du modèle (y)

c) - l'erreur de prédiction ($e = y - y_p$).

IV.2.3.2 Processus avec un bruit blanc affectant la sortie (NBSX)

Faisons enfin l'hypothèse d'un modèle entrée sortie avec un bruit de sortie.

Le processus décrit par :

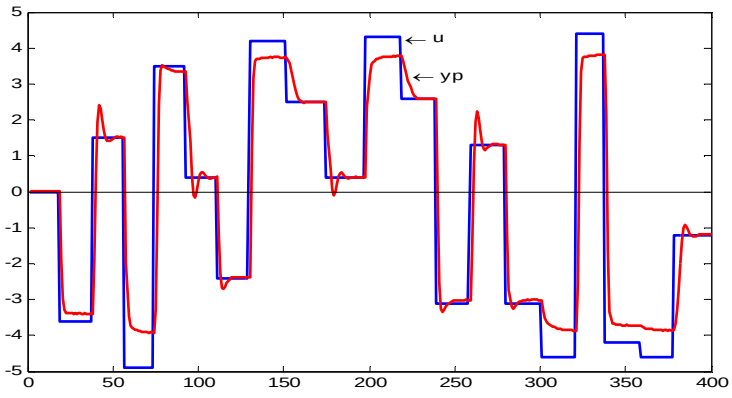
$$\begin{cases} x_{1p}(k+1) = 1.145\, x_{1p}(k) - 0.549 x_{2p}(k) + 0.584\, u(k) \\ x_{2p}(k+1) = x_{1p}(k) /(1 + 0{,}01\, x2p(k))^2 + 0.330 u(k) \\ y(k) = 4\, .tanh(x_{1p}(k)/4) + w(k) \end{cases} \quad \textbf{(4.5)}$$

Ou *w(k)* est un bruit blanc avec une valeur moyenne nulle et une variance de 3.10^{-2}.

En utilisant les mêmes séquences de commande utilisées pour l'apprentissage et test du processus déterministe (figure.IV.1), ainsi que le même predicteur neuronal entrée-sortie (figure IV.8).

De la même procédure que le modèle déterministe, On effectue l'apprentissage de telle manière qu'il réalise les meilleurs résultats. On prend le prédicteur bouclé, et on augmente progressivement le nombre de neurones jusqu'à ce qu'on obtienne les performances désirées. Avec n = m =2 et 5 neurones cachés. On obtient des erreurs d'estimation avec les variances : **EQMA=2.6.10^{-2}** et un **EQMV=3.8.10^{-2}** qui sont voisine de celle du prédicteur d'état.

Les résultats obtenus avec ce processus sont illustrés sur la figure IV.12.



a

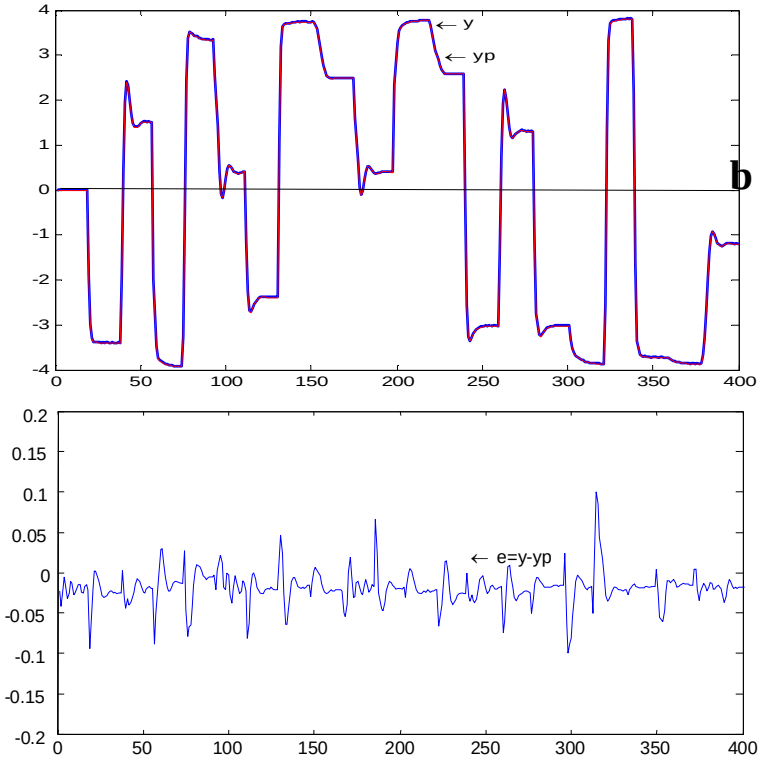


Figure IV.12:
Résultats obtenus pour le modèle NBSX, modélisation avec un pédicteur entrée-sortie
a) - le signal de commande, la sortie du processus (*yp*),
b) - la sortie du processus (*yp*), et du réseau (*y*),
c) - l'erreur de prédiction ($e = y - y_p$),

IV.2.4 Modélisation par un réseau statique

Pour comparer les performances des réseaux de neurones dynamiques, nous avons également étudié ce modèle avec des réseaux statiques (perceptron multicouches) avec des fonctions d'activations tangentes hyperbolique pour les neurones de la couche cachée et des fonctions d'activation linéaires pour la couche de sortie. Les tests ont été effectués à l'aide du logiciel « MATLAB 6.5 ».

L'apprentissage du réseau

L'entraînement du réseau a été réalisé par l'algorithme de rétropropagation [Annexe]. Nous avous utilisé les mêmes bases de données que celle de la modélisation dynamique, représentées sur la figure IV.1.

On lance la phase d'apprentissage avec l'algorithme de rétropropagation (fonction newff du matlab). Lorsqu'on constate que le réseau converge vers une erreur quadratique acceptable, on sauvegarde les poids de pondérations (connexions). On lance ensuite la phase de test. Si on trouve que le réseau généralise mal, on reprend le processus d'apprentissage avec une nouvelle architecture.

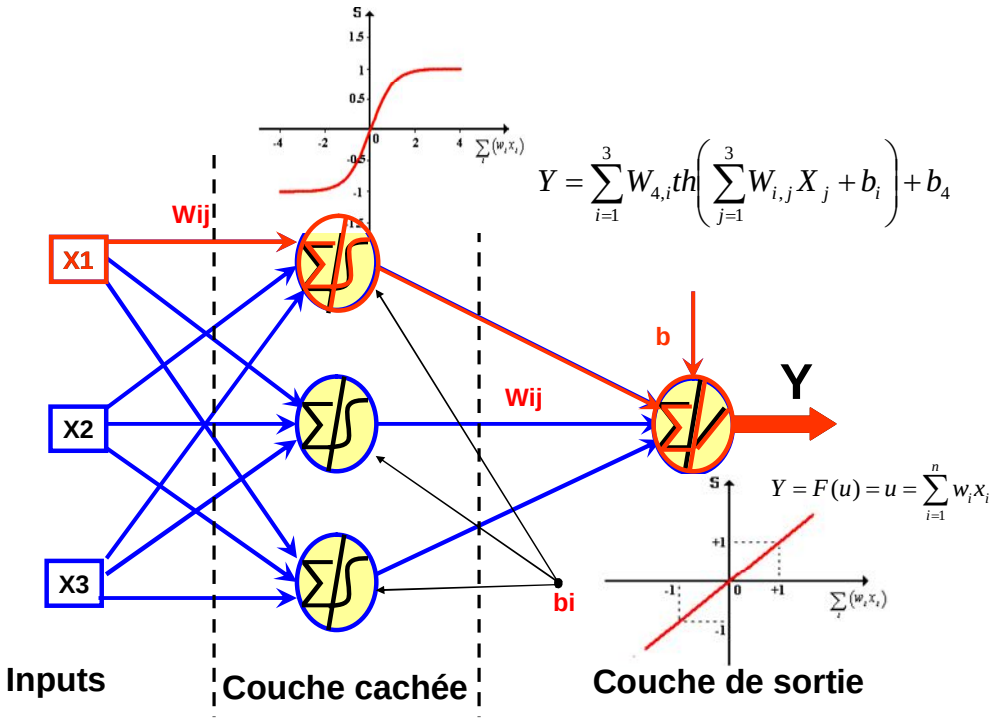


Figure IV.13 : Structure du MLP utilisé pour la modélisation statique.

Nous avons utilisé un réseau de neurones constitué d'une couche d'entrée à 3 neurones dont la fonction d'activation est la tangente hyperbolique, et d'une couche cachée ayant un nombre de neurones croissant afin de rechercher celle qui permet d'obtenir la meilleure performance par rapport aux réseaux dynamiques, avec une fonction d'activation linéaire (figure IV.13).

Le tableau montre les performances de chaque architecture :

Nbr de neurones cachés	EQMA	EQMT	Nombre d'itérations
8	$1.31.10^{-1}$	$2.28.10^{-1}$	157027
11	$3.015.10^{-2}$	$3.13.10^{-2}$	43210
15	$7.6.10^{-2}$	$1.5.10^{-1}$	43505

Tableau IV.2 Meilleurs résultats obtenus pour le modèle statique.

IV.2.5 Analyse des performances

Rappelons que l'objectif de ce modeste travail consiste en un premier lieu à étudier les avantages de la modélisation boite-noir des systèmes non linéaires dynamiques en utilisant des réseaux de neurones décrits dans l'espace d'état par rapport aux modèles entrés- sortie. Puis, d'examiner les performances des réseaux dynamiques par rapport aux réseaux statiques.

IV.2.5.1 Comparaison des performances du réseau d'état avec le réseau entré-sortie

Le tableau IV.3 représente les meilleurs résultats obtenus avec un prédicteur associé à un modèle hypothèse avec un bruit qui affecte la sortie du processus :

Type de réseau	EQMA	EQMT
Modèle d'état bouclé avec 2 neurones cachés	$2.8.10^{-2}$	$3.2.10^{-2}$
Modèle entrée- sortie avec n=m=2 et 5 neurones cachés	$2.6.10^{-2}$	$3.6.10^{-2}$

Tableau IV.3 Meilleurs résultats obtenus avec les deux modèles.

Nous constatons qu'un prédicteur d'état ayant 2 neurones cachés présente le meilleur EQMT et qu'aucun prédicteur entré-sortie n'approche cette performance même avec l'augmentation de neurones cachés. Or si on augmente le nombre d'entrées, on augmente aussi le nombre des paramètres du réseau et donc le risque de surajustement.

Ces résultats confirment la supériorité des prédicteurs d'état avec moins de neurones.

IV.2.5.2 Comparaison des performances du réseau dynamique avec le réseau statique (percepton)

Le tableau IV.4 représente les meilleurs résultats obtenus avec des modèles dynamiques d'état, d'entrées-sorties, et des modèles statiques.

Type de réseau	EQMA	EQMT
Modèle d'état bouclé avec 2 neurones cachés	$2.8.10^{-2}$	$3.2.10^{-2}$
Modèle entrée- sortie avec n=m=2 et 5 neurones cachés	$2.6.10^{-2}$	$3.6.10^{-2}$
Perceptron avec 11 neurones dans la couche cachée	$3.015.10^{-2}$	$3.13.10^{-2}$

En analysant les résultats de la modélisation dynamique et la modélisation statique, nous constatons que :

- Le temps d'apprentissage est bien meilleur pour les réseaux dynamiques que les réseaux statiques (en regardent le nombre d'itérations pour chaque modèle).
- Du point de vue complexité et structure du réseau, les réseaux dynamiques sont bien meilleurs que les réseaux statiques (en regardent le nombre de neurones par couche).
- Du point de vue qualité des résultats, les réseaux dynamiques sont bien meilleurs que les réseaux statiques (valeurs des erreurs EQMA et EQMT).

IV.2.5.3 Comparaison de nos résultats avec ceux de la référence [13]

Dans la référence [13], I. RIVALS, L. PERSONNAZ réalise l'apprentissage du même prédicteur neuronal d'état bouclé de la figure VI.2 avec la base de données de la figure IV.1, et fournit pour le modèle déterministe un EQMA = $1.6.10^{-7}$ et EQMT =$3.8.10^{-7}$ avec des fonctions d'activations sigmoïdes pour la couche cachée et la couche de sortie, et avec des initialisations à zéro des poids de connexions.

Dans notre étude, nous avons utilisé le même type de prédicteur d'état, avec des fonctions d'activation tangente hyperbolique pour la couche cachée, et des fonctions linéaires pour la couche de sortie, ainsi que des initialisations aléatoires des poids de connexions.

Nos résultats sont presque identiques à ceux de la référence [13].

IV.2.6 Problème des minimums locaux

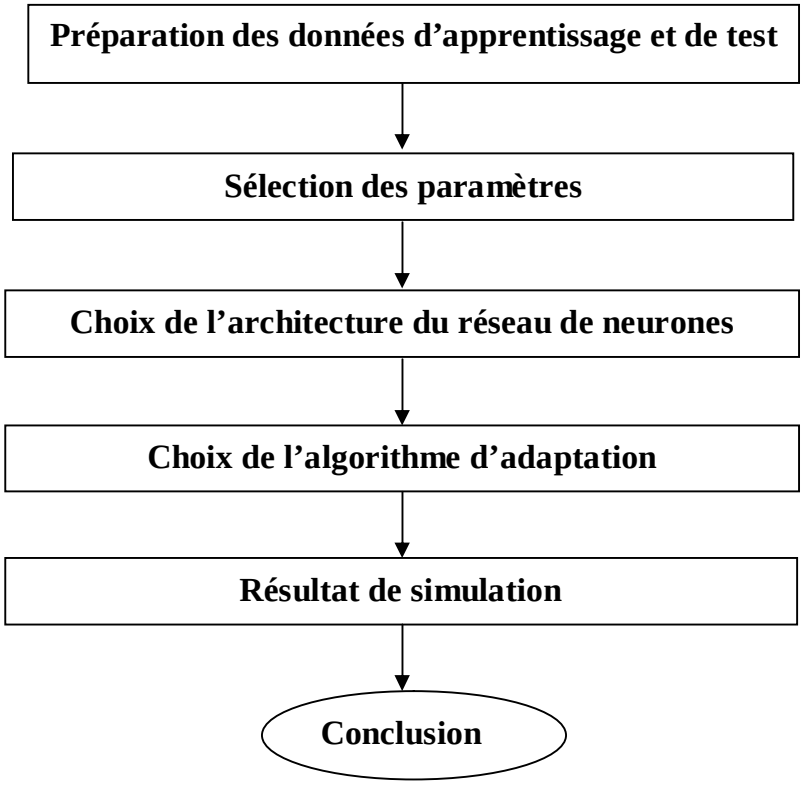
En raison du choix des fonctions d'activation de la couche cachée, qui sont généralement non linéaires (fonctions sigmoïdes, tangente hyperbolique), et éventuellement en raison du caractère récurrent des réseaux employés, la fonction de coût ne présente plus un minimum local, mais plusieurs minima locaux, vers lesquels peuvent converger les algorithmes d'optimisation. En effet, des initialisations différentes peuvent conduire à trouver, dans l'espace des paramètres, des minima différents. Donc les résultats de l'apprentissage peuvent être très différents **[20][25]**.

Dans notre application, nous avons procédé à des initialisations différentes des paramètres ; de telle sorte que, en les valeurs des sorties des neurones cachés se situent dans les parties linéaires des sigmoïdes pour l'ensemble des séquences d'apprentissage. Pour chaque architecture de réseau, on effectue plusieurs apprentissages avec des initialisations différentes, et l'on retient le réseau qui correspond à la plus petite valeur de la fonction de coût.

IV.3 Etude d'un processus réel (système d'engrenage)

IV.3.1 Méthodologie d'entraînement et de validation du réseau de neurone

L'approche que nous avons retenu vise à faire l'entraînement du réseau par une banque de données riche et assez représentative dans le but d'assurer un apprentissage rapide et une généralisation correcte. Cette approche se compose des quatre étapes suivantes :



IV.3.1.1 Préparation des données d'apprentissage et de test

La recherche de signatures consiste à construire le vecteur forme pour définir d'une manière pertinente les observations effectuées sur le système.

Rappelons que notre objectif consiste à identifier un mode de fonctionnement parmi les modes disponibles dans la base de données. C'est aussi et surtout la possibilité de détecter les nouveaux modes de fonctionnement lorsque ceux-ci apparaissent et de pouvoir les intégrer dans sa base de données après expertise. Et cela est difficilement réalisable si la recherche de signature est basée sur un seul type d'analyse **[25].** C'est pour cela que les paramètres du vecteur forme sont issus de divers types d'analyses. Nous

allons exposer au cours des paragraphes suivants les différents paramètres utilisés pour construire le vecteur forme. Ces paramètres sont calculés en utilisant les méthodes classiques de traitement des signaux. Une fois le vecteur forme est préparé et normalisé, 75% des données sont utilisées pour l'apprentissage du réseau et 25% restantes sont retenues pour tester la validité du réseau.

IV.3.1.1.1 Données temporelles

Les analyses temporelles ont permis d'extraire les différents indicateurs pour pouvoir les exploiter comme base de données d'apprentissage. Puisque le signal délivré par un accéléromètre est constitué de la réponse de nombreuses résonances, la mesure ou le calcul des indicateurs dans une bande fréquentielle étendue (large bande) réduit considérablement voir annule leur capacité de détecter l'existence de défauts induisant des chocs périodiques. La solution consiste à calculer ces indicateurs dans un certain nombre de bandes fréquentielles ni trop étroites ni trop larges, définis en fonction des caractéristiques cinématiques de chaque machine **[26].** Cette opération peut être facilement réalisée grâce au calcul des indicateurs temporels du signal filtré passe bande autour des résonances excitées préalablement identifiées dans le spectre du signal à analyser , ou plus simplement dans 5 bandes fréquentielles issues de la décomposition de la bande d'analyse choisie en 4 bandes adjacentes.

Nous avons choisis après une analyse préliminaire de calculer les indicateurs temporels dans différentes bandes de largeur 167 HZ, soit dix bandes latérales autour de la fréquence d'engrènement et ses deux harmoniques (fe=*333,33 Hz*. *2fe=666,66Hz 3.fe=999,99Hz).*

Le tableau IV.4 présente le calcul des indicateurs temporels, dans les différentes bandes fréquentielles, effectué pour chaque jour d'expérimentation.

	Bande 250-417 Hz			Bande 583-750 Hz			Bande 916-1083 Hz		
	V.Eff.1	F.Crt.1	Krts.1	V.Eff.2	F.Crt.2	Krts.2	V.Eff.3	F.Crt.3	Krts.3
Jour 2	972.25	2.91	1.97	1956.4	2.54	2.05	1950.9	2.97	2.48
Jour 3	1048.7	2.56	1.97	2079.3	2.5	2.01	1860.8	2.84	2.45
Jour 4	1130.1	3.33	2.63	1618.6	3.15	2.73	1872.2	3.6	2.79
Jour 5	877.93	3.39	2.31	1704.8	3.08	2.31	1851.4	3.22	2.54
Jour 6	1112.3	2.65	2.34	1894.7	2.46	2.08	1774.8	2.98	2.65
Jour 7	1001.8	2.74	2.12	1916.7	2.56	2.07	1781.3	2.94	2.62
Jour 8	1432.5	2.81	2.24	1859.5	2.73	2.15	1750.6	3.04	2.66
Jour 9	1006.5	2.72	2.28	1815.1	2.72	2.17	1676.4	2.92	2.66
Jour10	1176.1	3.45	2.50	2005.7	2.45	2.01	1669	3.3	2.66
Jour11	1083.8	3.09	2.26	1965	2.39	1.96	1635.7	3.39	2.61
Jour12	1561.9	8.25	21.8	1914.5	5.79	4.6	2595.9	6.22	9.94
Jour13	1942.1	15.06	71.89	2.45	9.71	10.58	2646.8	6.33	10.65

Tableau IV.4: Calcul des indicateurs temporels.

Où :V.Eff : valeur efficace. F.Crt : facteur de crête. Krts : Kurtosis

L'observation de l'évolution des indicateurs temporels dans les différentes bandes nous permet de remarquer le changement de ces valeurs dans les différentes bandes et une augmentation de ces valeurs à partir du 12ème jour, caractérisant une dégradation de l'état de l'engrenage et l'apparition d'un mode de fonctionnement avec défaut.

IV.3.1.1.2 Données fréquentielles

Nous avons vu que la densité spectrale de puissance (DSP) est la plus utilisée dans le diagnostic vibratoire des machines tournantes et en particulier les engrenages.

$$DSP(f) = \frac{|X(f)|^2}{d}$$

Où *DSP(f)* est la densité spectrale de puissance du signal vibratoire, *X(f)* sa transformée de Fourier, *d* sa durée d'observation.

IV.3.1.1.3 Analyse cepstrale

Nous avons vu que l'analyse par le cepstre présente des propriétés qui la rendent adaptée à l'étude des vibrations d'engrenages. Le cepstre est utile pour l'étude des signaux présentant des modulations. C'est le cas du signal d'engrenage qui est produit par un signal d'engrènement et des modulations liées à la rotation des roues. L'observation de ces modulations peut donner des renseignements sur l'état de l'usure de l'engrenage. Le tableau IV.5 présente pour chaque jour d'expérimentation les valeurs des amplitudes des deux raies de modulation, leurs positions et le rapport entre les deux roues. La $12^{ème}$ raie correspondant à la 21 dent a entièrement disparu [**26][32]**

Les renseignements que donne dans ce cas l'analyse cepstrale sont les suivantes **[32]** :

- Elle met en évidence la présence de deux modulations alors que dans le signal temporel et sur le spectre, il semble qu'il n'en existe qu'une.
- La détérioration affectera la vibration enregistrée à chaque passage de la dent abîmée au niveau de la prise de charge ; c'est à dire avec une fréquence égale à la fréquence de rotation de la roue concernée. L'amplitude de la raie associée à cette roue sera affectée par l'apparition du défaut.

	Premier pic (roue à 20 dents)		Deuxième pic (roue à 21 dents)		Rapport entre les positions
	position	amplitude	position	amplitude	
Jour 2	0.0591	0.1986	0.0621	0.1361	1.050
Jour 3	0.0588	0.2154	0.0618	0.136	1.051
Jour 4	0.0606	0.1675	0.0636	0.146	1.049
Jour 5	0.0597	0.19.36	0.0627	0.153	1.050
Jour 6	0.0591	0.1905	0.0621	0.1589	1.050
Jour 7	0.0591	0.2154	0.0621	0.1419	1.050
Jour 8	0.0591	0.129	0.0621	0.0891	1.050
Jour 9	0.0591	0.25	0.0621	0.01373	1.050
Jour10	0.0579	0.2782	0.0608	0.0695	1.050
Jour11	0.0579	0.2446	0.0609	0.0803	1.051
Jour12	0.0597	0.1969	-	-	-
Jour 13	0.0597	0.1753	-	-	-

IV.3.1.2 Sélection des paramètres

La première partie de cette étude nous a permis de mettre en œuvre plusieurs techniques pour construire la base de données du vecteur forme. Le nombre souvent élevé de paramètres peut être pénalisant en terme de temps de calcul. De plus tous les paramètres calculés ne seront pas forcément pertinents vis à vis des modes étudiés.

La sélection d'un nombre réduit de paramètres pertinents pour les signatures des défauts présents plusieurs intérêts : elle permet de réduire le temps de calcul et la complexité des algorithmes de classification mis en jeu.

Dans l'espace de représentation où s'effectuera la classification, les défauts appartenant à la même classe sont les plus proches regroupés possible et à l'inverse, les défauts issus de classes différentes sont situés dans une des régions distinctes de l'espace.

Nous présentons sur le tableau IV.6 l'ensemble des paramètres qui définissent le vecteur forme. Les variables d'entrée sont les différents paramètres de ce vecteur. On cherche à associer un mode de fonctionnement à ces vecteurs d'entrée. Les variables de sortie sont les classes représentant le mode de fonctionnement.

La signature des signaux d'engrenage est définie par seize paramètres.

Domaine	Dénomination		symbole
temporel	Bande 250-417	Valeur efficace 1	V.Eff 1
		Facteur de crête 1	F.crt 1
		Kurtosis1	Krts1
	Bande 583-750	Valeur efficace 2	V.Eff 2
		Facteur de crête 2	F.crt 2
		Kurtosis2	Krts 2
	Bande 916-1083	Valeur efficace 3	V.Eff 3
		Facteur de crête 3	F.crt 3
		Kurtosis3	Krts 3
Fréquentielle	Bande 0-400Hz	Energie 1	B1
	Bande 400-740Hz	Energie 2	B2
	Bande 740-1080Hz	Energie 3	B3
	Bande 1080-1400Hz	Energie 4	B4
	Bande totale 6Hz	Energie 5	B5
Cepstre		Amplitude pic roue20	S1
		Amplitude pic roue21	S2

Tableau IV.6 : L'ensemble des paramètres qui définissent le vecteur forme.

IV.3.1.3 Choix d'une hypothèse

Le travail que nous avons mené consiste à identifier un mode de fonctionnement parmi les modes disponibles dans la base de données. Cette identification peut être décrite par un modèle dynamique non linéaire, possédant 16 variables d'entrées. Sur la base de ces informations, on peut donc d'élaborer le modèle non linéaire dynamique correspondant à l'hypothèse suivante :

$$\begin{cases} x(k+1) = \varphi\,(x\,(k),\, u(k)) \\ y(k+1) = \Psi(x\,(k),\, u(k)) + br\,(k) \end{cases}$$

Où x(k) est le vecteur d'état (dont les composants sont les variables d'état), u(k) est le vecteur des entrées de commande, représenté par l'ensemble des paramètres qui définissent le vecteur forme (les 16 paramètres), y (k) la sortie du processus et br(k) est un bruit additif qui affecte la sortie du processus. **φ** et **Ψ** sont des fonctions non linéaires.

IV.3.1.4 Choix de l'architecture du réseau de neurones

La deuxième étape concerne la sélection de l'architecture de réseau. Cette tache est délicate, il n'y a pas de règles systématiques pour le choix du nombre de neurones par couche cachée. Rappelons que ce nombre est particulièrement important car il détermine la capacité de calcul du réseau .Un nombre insuffisant de neurones cachés peut compromettre la capacité du réseau à résoudre le problème. Inversement, un nombre élevé de neurones force le réseau à apprendre par cœur. Alors que le nombre de neurones de la couche d'entrée et de sortie soit imposé par le nombre d'entrées du système (le nombre de neurones à d'entrée est égal au nombre de composantes des vecteurs formes), ainsi que par la codification des différentes classes (pour la couche de sortie). Pour cela à chaque fois qu'on présente des exemples d'apprentissage, on évalue les résultats obtenues, s'ils ne sont pas acceptables, on augmente progressivement le nombre de neurones jusqu'à ce qu'on obtient les performances désirées.

On a essayé avec plusieurs architectures ; on a utilisé un réseau de neurones bouclé avec un retour de la sortie vers la couche d'entrée. Le caractère récurrent du réseau se traduit par le fait que le vecteur des variables des entrées est constitué de l'ensemble des paramètres qui définissent le vecteur forme, ainsi que la sortie à l'instant précédent. Il est constitué donc d'une couche d'entrée à 17 neurones (les 16 paramètres plus la sortie à l'instant précédent), d'une couche cachée à 10 neurones dont la fonction d'activation est la sigmoïde, et une de sortie à 1 neurone de fonction d'activation Heaviside (représente la classe d'appartenance) (figure IV.14). Le réseau de neurones doit nous fournir une réponse qui nous renseigne sur l'état de l'équipement.

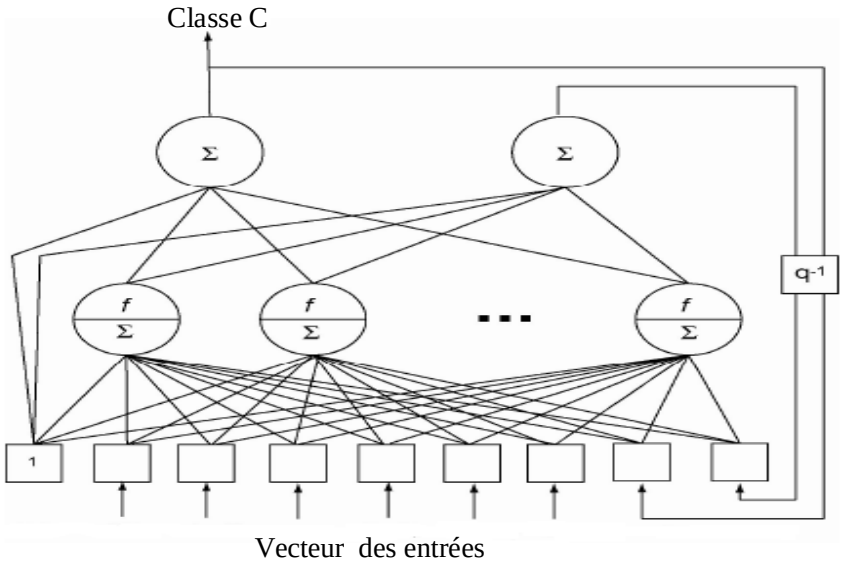


Figure IV.14 : Structure adoptée pour le modèle neuronal récurrent.

IV.3.1.5 Choix de l'algorithme d'adaptation

La dernière étape consiste à choisir l'algorithme d'apprentissage parmi les nombreux algorithmes d'adaptation cités en Annexe, Il est alors possible d'utiliser l'algorithme le plus connu pour ce type de tâche : l'algorithme de rétropropagation du gradient celui de rétropropagation du gradient de l'erreur. Il faut préciser que l'algorithme de rétropropagation du gradient demande un nombre d'exemples d'apprentissage conséquent afin de trouver des frontières de classification correctes. De plus, le calcul est rapide pour obtenir une réponse du réseau ayant appris les données d'apprentissage.

On rappelle son principe :

1- Initialiser les poids à des valeurs aléatoires.

2- Présenter un exemple d'entraînement.

3- Calculer :

- Les sorties des neurones de la couche cachée.
- Les sorties du réseau.

4- Calculer :

- Les erreurs pour les neurones de la couche de sortie.
- Les erreurs pour les neurones de la couche cachée.

5- Ajuster :

- Les poids de la couche de sortie.
- Les poids de la couche cachée.

6- Présenter un autre exemple d'entraînement et aller à l'étatpe3.

Les exemples sont présentés d'une manière récursive jusqu'a ce que l'erreur de sortie se stabilise à un niveau acceptable, et on dit que le réseau a convergé.

IV.3.1.6 Description de la base de données d'apprentissage et de test

Nous allons exposer au cours des sections précédentes les différents paramètres utilisés pour construire le vecteur forme. Ces paramètres sont calculés en utilisant les méthodes classiques de traitement des signaux. Pour que notre entraînement soit correct, on doit calculer ces paramètres sur plusieurs périodes des signaux vibratoires; ils sont calculés sur les premières 25 périodes. Une fois le vecteur forme est préparé et normalisé ,75% des données sont utilisées pour l'apprentissage du réseau, et 25% restantes sont retenues pour tester la validité du réseau.

IV.3.1.7 Résultat de simulation

L'entraînement des réseaux a été réalisé en se basent principalement sur l'algorithme de rétropropagation du gradient. Les critères de classement que nous avons choisi d'utiliser sont fondés sur la séparabilité entre classes. Il faut ensuite affecter une classe à chaque observation à l'aide d'une règle de décision. Plusieurs règles sont

possibles **[25]-[27]**. Nous avons choisi la règle simple qui consiste à affecter l'observation à la classe correspondant à la sortie maximale. La figure IV.15 présente le résultat de cette affectation après plusieurs entraînements.

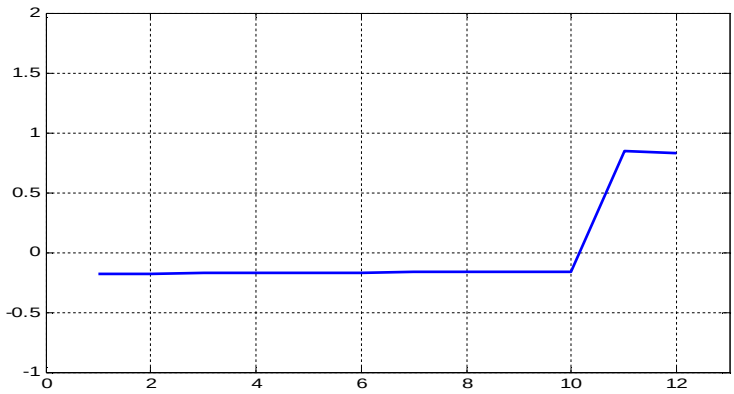


Figure IV.15 : Affectation des observations par classe

A la fin de l'apprentissage, les poids sont sauvegardés dans un fichier sous forme de tableaux multidimensionnels où chaque page représente le résultat obtenu à chaque itération. L'évolution des poids liant la couche cachée à celle de la couche de sortie est donnée par la figure IV.16. On constate qu'au bout des 150 itérations, on obtient une bonne convergence des poids et ne varient plus au delà de cette itération.

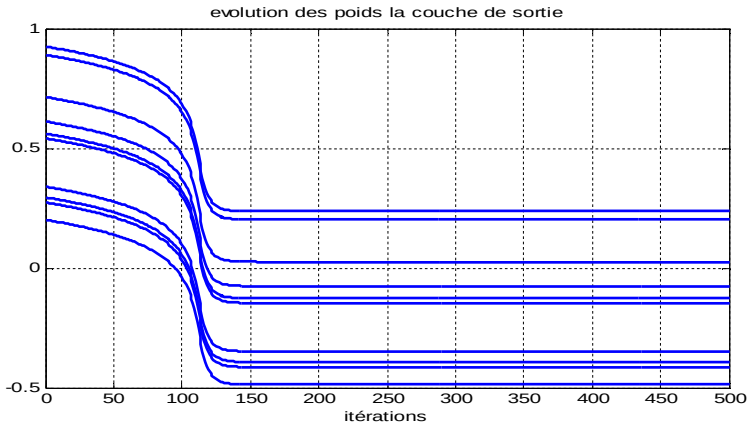


Figure VI.16: L'évolution des poids de la couche de sortie

Afin de faciliter l'interprétation des résultats, on les représente graphiquement. La figure IV.17 présente la répartition des observations comme nous pouvons le constater ; dans l'espace de représentation où s'effectuera la classification, les défauts appartenant à

la même classe sont les plus proches regroupés possible et à l'inverse, les défauts issus de classes différentes sont situés dans une des régions distinctes de l'espace des classes.

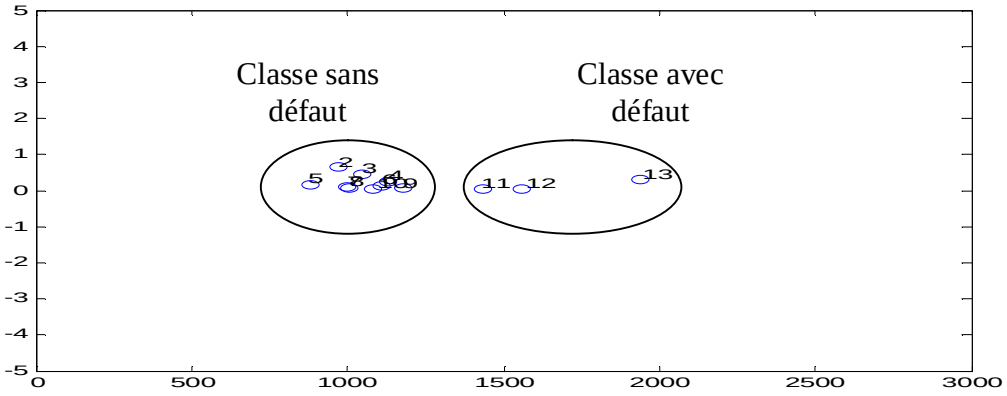


Figure IV.17 : La répartition des observations par classe.

On voit que la structure en classes a bien été trouvée, et que les signaux enregistrés le 11^{éme} 12^{éme} et 13^{éme} jour sont assez particuliers. On aperçoit qu'il forme la classe de fonctionnement avec défaut ; alors que les autres forment la classes de fonctionnement sans défaut.

L'évaluation des performances du réseau a été effectuée à l'aide de la base des données qui a été préparée, 75% des données est utilisée pour l'apprentissage du réseau et 25% restante sont retenues pour tester la validité du réseau. On mesure les performances selon le critère de pourcentage des individus bien classés.

- Le taux d'observations bien classées (T_B) :

$$T_B = \frac{N_B}{N} * 100$$

- Le taux d'observations mal classées (T_M) :

$$T_M = \frac{N_M}{N} * 100$$

Avec : N_B : Nombre de bien classées. N_M : Nombre de mal classées.

N : Nombre total d'échantillons.

Les résultats sont présentés dans le tableau IV.7.

	Performance
Base d'apprentissage	100%
Base de test	96,8%

Tableau IV .7 : Pourcentage de bien classés pour le réseau de neurone.

La figure IV.18 présente la représentation graphique de la répartition des observations de test du réseau. Nous constatons que le réseau a bien classé les données d'apprentissage.

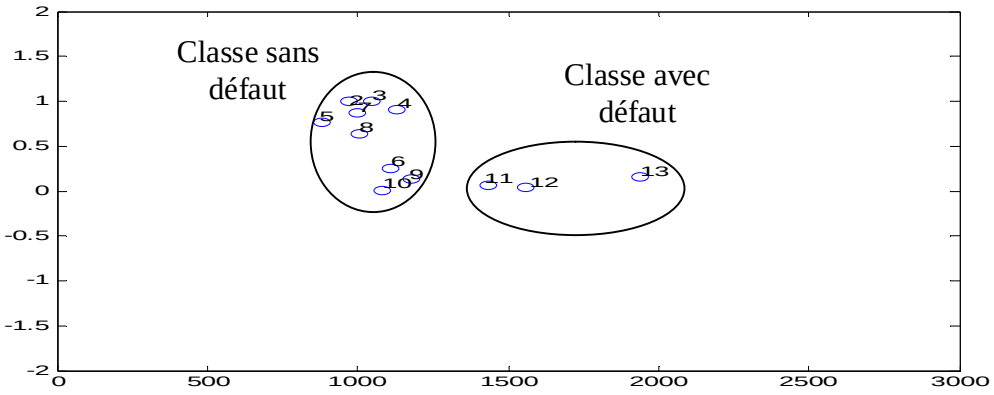


Figure IV.18 : La répartition des observations de test.

IV.4. Test avec un réseau statique (perceptron)

Pour étudier les performances du réseau dynamique par rapport au réseau statique, nous avons également testé ce modèle avec des réseaux statiques ou non bouclés (perceptron multicouches) où nous avons utilisé un réseau de neurones constitué d'une couche d'entrée de 16 neurones, et d'une couche cachée avec une fonction d'activation sigmoïde et ayant un nombre de neurones croissant afin de rechercher celle qui permet d'obtenir la meilleure performance par rapport aux réseaux dynamiques où nous avons arrivé a une couche caché de 15 neurones, et une couche de sortie à 1 neurone de fonction d'activation Heaviside (figure IV.19).

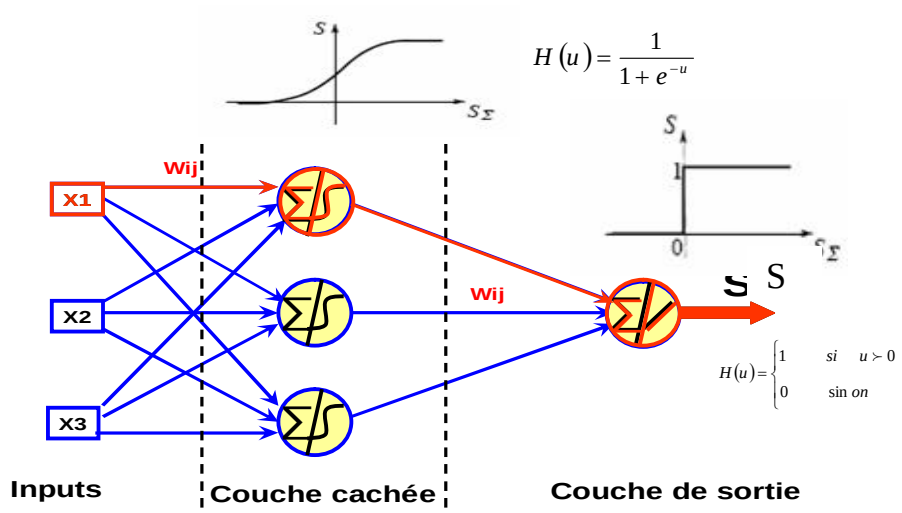


Figure IV.19 : Structure du MLP utilisé pour la modélisation statique.

La représentation graphique de la répartition des observations est donnée sur la figure IV.20. On constate aussi que le réseau est arrivé également à bien classer les données d'apprentissage.

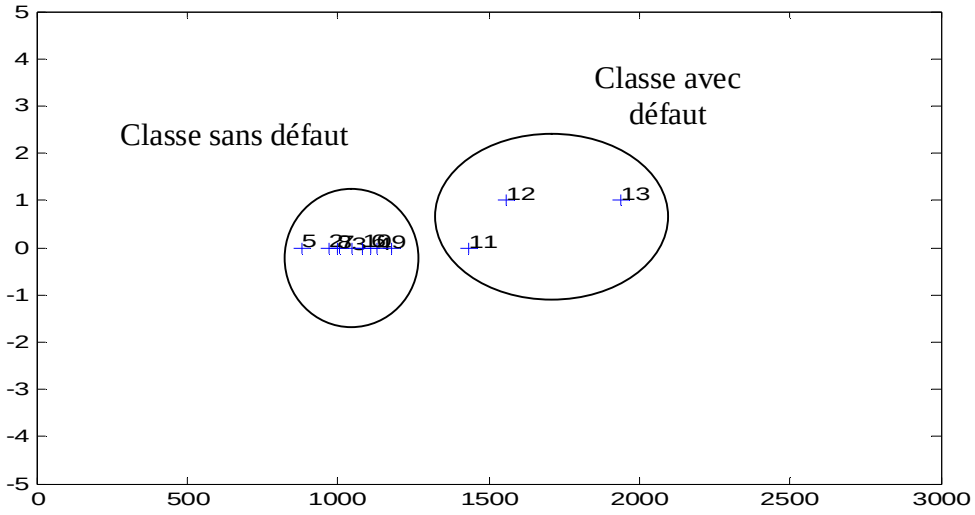


Figure IV.20 : La répartition des observations par un réseau statique.

IV.5 Comparaison entre le réseau dynamique et le réseau statique

Dans notre étude, nous avons appliqué les réseaux de neurones pour l'analyse et le diagnostic précoce de signaux de vibration d'engrenage. Pour cela, nous avons utilisé un réseau de neurones bouclé avec un retour de la sortie vers la couche d'entrée. Ils est constitué d'une couche d'entrée à 17 neurones, d'une couche cachée à 10 neurones dont la fonction d'activation est la sigmoïde, et une couche de sortie à 1 neurone de fonction d'activation Heaviside pour le réseau dynamique ,et un réseau non bouclé (perceptron multicouche) où nous avons utilisé un réseau de neurones constitué d'une couche d'entrée de 16 neurones ,et d'une couche cachée ayant 15 neurones avec une fonction d'activation sigmoïde, et une couche de sortie à 1 neurone de fonction d'activation Heaviside.

Les tests du réseau neuronal ont mis en évidence ses bonnes performances à détecter le défaut d'engrenages, aussi bien pour le réseau dynamique que statique, mais toujours avec moins des performances (La convergence de l'algorithme de rétropropagation peut devenir lent, de plus la complexité et la structure du réseau, et la qualité des résultats).

IV.6 Conclusion

Le système d'engrenages utilisé dans les essais est composé de deux roues comportant respectivement 20 et 21 dents. Un enregistrement du signal vibratoire a été effectué chaque jour pendant une durée de 13 jours correspondant au fonctionnement permanent du système d'engrenages sous test. L'engrenage se détériore au $11^{ème}$ jour par l'écaillage d'une dent.

La première phase de cette application consiste à construire la base de données du vecteur de forme et sélectionner un nombre réduit de paramètres pertinents pour la signature du défaut. Ces paramètres sont issus des données temporelles, fréquentielles et cepstrales. Ainsi, 16 paramètres sont retenus pour la définition du vecteur de forme. La seconde phase concerne la sélection de l'architecture du réseau de neurones utilisé pour le diagnostic du défaut d'engrenages. L'architecture retenue comporte une couche cachée

formée des neurones avec une fonction d'activation sigmoïde et une couche de sortie formée des neurones avec une fonction d'activation Heaviside. Les tests du réseau neuronal ont mis en évidence ses bonnes performances à détecter le défaut d'engrenages.

Conclusion générale

L'objectif de ce travail était l'étude des performances de la modélisation des systèmes non linéaires dynamiques au moyen des réseaux de neurones décrits dans l'espace d'état.

Pour atteindre cet objectif, nous avons d'abord présenté le principe de la modélisation dynamique des systèmes et processus non linéaires. Nous avons développé les structures des prédicteurs neuronaux associés aux modèles – hypothèses (NARX, NARMAX, NBSX) donnés aussi bien pour des modèles entrée – sortie que pour des modèles d'état. Nous avons également décrit les techniques d'estimation des paramètres du système non linéaire associé au modèle – hypothèse donné. Les algorithmes d'apprentissage sont nécessaires à mettre en œuvre pour l'ajustement de paramètres où on a introduit la notion de copie utile pour simplifier l'écriture du calcul de la fonction coût et son gradient.

En vue d'analyser les performances des réseaux de neurones dynamiques à représenter des systèmes réels , nous avons étudié le principe de la surveillance d'équipements industriels en utilisant des réseaux de neurones dont les capacités de mémorisation , d'apprentissage et d'adaptation sont des fonctions très utiles au système de surveillance

Deux applications des réseaux de neurones dynamiques sont considérées dans cette étude : La modélisation d'un processus décrit dans l'espace d'état et le diagnostic des défauts d'un système d'engrenages.

Pour la première application, le processus étudié est écrit par deux équations d'état et une équation d'observation non linéaires. Le prédicteur neuronal d'état utilisé pour modéliser ce processus est un réseau bouclé constitué d'une couche d'entrée à trois neurones, d'une couche cachée à deux neurones non linéaires et une couche de sortie à trois neurones linéaires. Les séquences de test comportent 400 échantillons constitués de créneaux d'amplitudes aléatoires dans [-5, +5]. L'apprentissage est effectué à l'aide d'un algorithme du gradient suivi d'un algorithme quasi- Newton. Nous avons remarqué que le prédicteur a bien réussi à estimer la sortie du processus.

Toutefois, les résultats obtenus par ce réseau ne sont pas identiques à ceux du processus confirmant ainsi les prédictions théoriques.

Le test des performances du prédicteur neuronal en présence de bruit blanc affectant la sortie ou l'état du processus montre que le prédicteur neuronal constitue une excellente approximation du processus. La modélisation de ce même processus par un prédicteur entrée – sortie et le test dans les mêmes conditions donne des résultats similaires avec un grand nombre de neurones. Ce qui confirme la supériorité des prédicteurs d'état sur les prédicteurs entrée – sortie.

Pour la seconde application, le système d'engrenages utilisé dans les essais est composé de deux roues comportant respectivement 20 et 21 dents. Un enregistrement du signal vibratoire a été effectué chaque jour pendant une durée de 13 jours correspondant au fonctionnement permanent du système d'engrenages sous test. L'engrenage se détériore au $12^{éme}$ jour par l'écaillage d'une dent.

La première phase de cette application consiste à construire la base de données du vecteur de forme et sélectionner un nombre réduit de paramètres pertinents pour la signature du défaut . Ces paramètres sont issus des données temporelles, fréquentielles et cepstrales. Ainsi, 17 paramètres sont retenus pour la définition du vecteur de forme.

La seconde phase concerne la sélection de l'architecture du réseau de neurones utilisé pour le diagnostic du défaut d'engrenages. L'architecture retenue comporte une couche cachée formée des neurones avec une fonction d'activation sigmoïde et une couche de sortie formée des neurones avec une fonction d'activation heaviside. Les tests du réseau neuronal ont mis en évidence ses bonnes performances à détecter le défaut d'engrenages.

Références bibliographiques

[01] N. RIZKALLA **« Nanoparticules et réseaux de neurones artificiels : de la préparation à la modélisation »**, thèse de doctorat de l'université de Montréal Canada, février 2005.

[02] P. QUANG DUNG **« Réseaux de neurones pour la reconnaissance des formes »** Travail d'Intérêt Personnel Encadré par Prof. HO Tuong Vinh, Hanoi, 15 juillet 2005

[03] C.TOUZET **« Les réseaux de neurones artificiels introduction au connexionnisme cours, exercices et travaux pratiques »,** Humaine, Marseille, Juillet 1992.

[04] R ZEMOURI **« Contribution à la surveillance des systèmes de production à l'aide des réseaux de neurones dynamiques : Application à la e-maintenance »** Thèse de Doctorat, Université de Franche Comté, Besançon. 2003 **.**

[05] B. VIROLE **« Réseaux de neurones et psychométrie »**
Editions du Centre de Psychologie Appliquée – ECPA, Juin 2001

[06] J.GHOUILI **« Commande sans capteur d'une machine asynchrone avec estimation de la vitesse par les réseaux de neurones »** Thèse de Doctorat, Université de Québec. Avril 2005.

[07] G. DREYFUS, J-M. MARTINEZ, M. SAMUELIDES, M.B. GORDON, F. BADRAN, S. THIRIA ET L.HERAULT **« Réseaux de neurone, Méthodologies et Application » ,** Paris, Edition Eyrolles.2002.

[08] G. DREYFUS **« les réseaux de neurones, une technique opérationnelle pour le traitement des données industrielles, économiques et financières »**
École Supérieure de Physique et de Chimie Industrielles de la Ville de Paris (ESPCI), Novembre 1997.

[09] I.RIVALS, L PERSONNAZ **« Les réseaux de neurones formels pour la modélisation, la commande et la classification »** CNRS Editions, Paris 2003

[10] B.GOSSELIN **« Application de réseaux de neurones artificiels a la connaissance automatique de caractères manuscrits »** Thèse de Doctorat, université Mons 1996.

[11] L.BAGHLI **« Contribution à la commande de la machine asynchrone, l'utilisation de la logique floue, des réseaux de neurones et des algorithmes génétiques »** Thèse de Doctorat, université Nancy I , 1999.

[12] O. NERRAND, P. ROUSSEL-RAGOT, L. PERSONNAZ & G. DREYFUS **« Neural Networks and Non-linear Adaptive Filtering: Unifying Concepts and New Algorithms. »** Neural Computation, Vol. 5, pp 165-199, 1993.

[13] I. RIVALS , L. PERSONNAZ **« Black Box Modeling With State Neural Networks »**In Neural Adaptive Control Technology I, R. Zbikowski and K. J. Hunt eds.,World Scientific, 1996.

[14] I. RIVALS, L. PERSONNAZ, G. DREYFUS & J.L. PLOIX **« Modélisation, classification et commande par réseaux de neurones : principes fondamentaux, méthodologie de conception et illustrations industrielles »** Publié dans : Les réseaux de neurones pour la modélisation et la commande des procédés, J.-P. Corriou coordonnateur, Lavoisier Technique et Documentation, Paris.1995.

[15] **O**. NERRAND, P. ROUSSEL-RAGOT, D. URBANI, L. PERSONNAZ ,G.DREYFUS **« Training Recurrent Neural Networks : Why and How? An Illustration in Process Modeling »** *IEEE* Trans. on Neural Networks, Vol. 5, No. 2, pp. 178-184, 1994.

[16] S. KUMPATI, F.NARENDRA, K.PARTHASARATHY **« adaptive Identification and Control of Dynamical Systems using Neural Networks »** Proceeding of the 28th Conference on Decision *and* Control Tampa, Florida December 1989.

[17] S. KUMPATI, F.NARENDRA ,K.PARTHASARATHY **« Identification and Control of Dynamical Systems using Neural Networks »** IEEE Trans. on Neural Networks, Vol.1 , No. 1, pp. 04-27, 1990.

[18] V.SRINIVASA .CHAKAVATHY AND JOYDEEP GHOSH **» Scale-based clustering using the radial basis function network »** IEEE Trans. on Neural Networks, Vol.7 , No. 5, pp. 1250-12617, 1996.

[19] I. RIVALS **« Modélisation et Commande de Processus par Réseaux de Neurones; Application au Pilotage d'un Véhicule Autonome »** Thèse de Doctorat de l'Université Paris 6 ,1995.

[20] D. URBANI **« Méthodes statistiques de sélection d'architectures neuronales : Application à la conception de modèles de processus dynamiques »** Thèse de Doctorat de l'Université Paris 6 ,1995

[21] T. KOHONEN **«The self –organizing Map »** Proceedings of the IEEE, VOL. 78, pp1464-1480 NO. 9, SEPTEMBER 1990 .

[22] Y.OUSSAR, **« Réseaux d'ondelette et réseaux de neurones pour la modélisation statique et dynamique de processus »**, thèse de doctorat de l'université Pierre et Marie Curie, Juillet 1998.

[23] I. RIVALS AND L. PERSONNAZ," **Neural-Network Construction and Selection in Nonlinear Modeling »** IEEE Trans. on Neural Networks Vol.14 No 4 pp. 804-819, 2003.

[24] LIJUN LUO, YANG LU, CAIRONG ZOU, ZHENYA HE **« Image sequence macroblock classification using neural networks"** Signal Processing 69 pp191-198, 1998.

[25] R.CASIMIR **« Diagnostic des défauts des machines asynchrones par des reconnaissances des formes »,** Thèse de Doctorat, université Lyon , 2003.

[26] S.FEDALA **« Le diagnostic vibratoire automatisé ; comparaison des méthodes d'extraction et de sélection du vecteur forme »** Thèse de magistère, Département mécanique, université de Sétif, 2004

[27] C.LURETTE **« Développement d'une technique neuronale auto –adaptative pour classification dynamique de données évolutives : application a la supervision d'une presse hydraulique »** Thèse de Doctorat, université Lille, 2003.

[28] S. VERRON « **Diagnostic et surveillance des processus complexes par réseaux bayésiens »** Thèse de Doctorat, université d'Angers, 2007.

[29] M.AYAD **« Etude comparative d'algorithmes de la transformée en ondelettes: Application a l'analyse des signaux empiriques »,** Thèse de magistère, Département d'électronique, université de Sétif, 2005.

[30] N. HALOUI, D. CHIKOUCHE, M. BENIDIR **« Application des méthodes d'analyse spectrale paramétriques à la détection des défauts d'engrenage dans les machines tournantes »,** IEEE Canadian Conférence on Electrical & Computer Engineering, 2002.

[31] [3] K. DROUICHE, M. SIDAHMED, Y. GRENIER, **« Détection de défauts d'engrenages par analyse vibratoire »,** Traitement du signal, vol. 8, **n°** 5, pp.331-343, 1994.

[32] C.CAPDESSUS, M.SIDAHMED **«Analyse vibratoire d'un engrenage : Cepstre, corrélation, spectre. »** Traitement du signal, vol. 8, n° 5, pp. 365-372,1994.

[33] M.AYAD , D. CHIKOUCHE **« Application de la transformée en ondelettes à l'analyse des signaux vibratoires d'un système d'engrenage en vue d'un diagnostic précoce »** Premier Congrès d'Electronique PCE'04, Sétif, 14-15 Avril 2004.

[34] M.BASSEVILLE, A BENVENISTE **« Surveillance et diagnostic à l'aide de modèles : études en mécanique des vibrations »** Traitement du signal, vol. 8, n° 5, pp. 291-300.1994.

[35] G.AMOURS **«Détection précoce de la propagation de fissures dans les engrenages droits par analyse vibratoire » ,** Mémoire maître de sciences (M.Sc.) université Laval, Canada **,**2000.

[36] P. ESTOCQ **« Une approche méthodologique numérique et expérimentale d'aide à la détection et au suivi vibratoire de défauts d'écaillage de roulements à billes »,** Thèse de Doctorat, université de Reims champagne Ardenne, 2004.

[37] A.BOULENGER, C.PACHAUD **« Analyse vibratoire en maintenance, Surveillance et diagnostic des machines. »,** DUNOD, Paris 2003**.**

[38] O.COUSINARD, P. MARCONNET **«Détection de l'endommagement d'un engrenage par l'emploi de l'analyse cepstrale et de la détection d'enveloppe : Application et validation industrielles pour le diagnostic d'un réducteur fonctionnant à faible vitesse »** 16ème Congrès Français de Mécanique Nice, 1-5 septembre 2003.

[39] D.THUILLIER **" Principe et applications des réseaux de neurones »** revue région et développement, n°5 Québec 1997.

[40] F.BONNARDOT « **Comparaison entre les analyses angulaire et temporelle des signaux vibratoires de machines tournantes. Etude du concept de cyclostationnarité floue. »** Thèse de Doctorat, université Grenoble, *2004*

[41] O. ONDEL « **diagnostic par reconnaissance des formes : application a un ensemble convertisseur – machine asynchrone »** Thèse de Doctorat, l'école doctorale électronique de Lyon, 2006.

[42] M. EL BADAOUI « **Contribution au Diagnostic Vibratoire des Réducteurs Complexes à Engrenages par l'Analyse Cepstrale »** Thèse de Doctorat, université Jean-Monnet, juillet 1999

[43] R. MAHADOUI « **Diagnostic industriel par neuro-flou -application a un système de production »** Thèse de magistère, Département Génie industriel, université de BATNA, 2008

[44] J. CHAPPELIER **« RST : une architecture connexionniste pour la prise en compte de relations spatiales et temporelles », thèse** de doctorat de l'Ecole Nationale Supérieure des Télécommunications, Paris 1996.

[45] P. DAY and R. DAVENPORT **"Continuous-Time Temporal BackPropagation with Adaptable Time Delays",** IEEE transactions on neural networks, vol. 4, no2, pp.348 - 354, march 1993.

[46] S.VAKULENKO **« Complexité dynamique des réseaux de Hopfield »** C. R. Acad. Sci. Paris, Ser. I 335 639–642, 2002.

[47] YONG LI, ZHENG TANG, GUANGPU XIA, AND RONGLONG WANG "**A Positively Self-Feedbacked Hopfield Neural Network Architecture for Crossbar Switching",** IEEE transactions on circuits and systems -i: regular papers, vol. 52, no. 1, January 2005.

[48] I RIVALS. PERSONNAZ L., DREYFUS G., CANAS D. **"Real-time control of an autonomous vehicle: a neural network approach to the path following problem."**5th International Conference on Neural Networks and their Applications, 1993.

[50] L. BUNIET **« Traitement automatique de la parole en milieu bruité : étude de modèles connexionnistes statiques et dynamiques »,** Thèse de Doctorat, université Nancy I , 1999.

[51] I .RIVALS **« les réseaux de neurones formels pour le pilotage de robots mobiles »** FLUX, revue de l'Association amicale Les Ingénieurs SUPÉLEC : La robotique mobile ; la fonction achats-logistique, No 178, ISSN 0766-3536 , septembre-octobre 1996.

1- Présentation

Soit un ensemble d'apprentissage $\{x^k, y_p^k\}_{k=1\ldots N}$ de *N* mesures des entrées et sorties obtenues sur un processus pour sa modélisation, où x^k , $y^k{}_p$ représentent respectivement le vecteur regroupant toutes les variables du modèle et la sortie du processus à l'instant discret *k*. Le problème d'apprentissage revient alors à déterminer, parmi un ensemble de fonctions candidates $\psi(\mathbf{x},\theta)$, paramétrées par le vecteur , celle qui minimise une fonction de coût prédéfinie. On utilise fréquemment la fonction de coût des moindres carrés, définie par :

$$J(\theta)=\frac{1}{2}\sum_{k=1}^{N}\left(y_p^k - y^k \right)^2 \qquad \textbf{(I.01)}$$

L'élaboration du modèle nécessite d'estimer les valeurs de ses paramètres, cette approche consiste à déterminer les paramètres optimaux par minimisation directe d'une fonction de coût vis-à-vis des paramètres du modèle. Ce type d'optimisation fait appel au calcul du gradient et/ou de la matrice Hessienne de la fonction de coût. C'est le type d'optimisation qui est mise en œuvre lors de l'apprentissage de réseaux de neurones.

Dans le cas où la fonction $\psi(x,\theta)$ est linéaire par rapport aux paramètres ; la minimisation de la fonction de coût est obtenue par solution des moindres carrés .

Dans le cas où la fonction $\psi(x,\theta)$ est non linéaire par rapport aux paramètres , des algorithmes itératifs doivent être adoptés pour rechercher le minimum de la fonction de coût.

2. Méthodes du premier ordre

Ces méthodes sont fondées sur le calcul du gradient de la fonction de coût J par rapport au vecteur des paramètres $\theta(\theta_1, \theta_2 \ldots \theta_m)^T$. Celui-ci s'écrit :

$$\nabla j = \left(\frac{\partial j}{\partial \theta_1}, \frac{\partial j}{\partial \theta_2}, \ldots, \frac{\partial j}{\partial \theta_m}\right)^T \qquad \textbf{(I.02)}$$

Le principe de cette méthode est de mettre à jour le vecteur des paramètres **θ** selon la formule suivante :

$$\theta^{k+1} = \theta^k - u_k \nabla j\left(\theta^k\right) \qquad \textbf{(I.03)}$$

Où le scalaire μ_k est appelé pas du gradient.

Suivant la valeur du pas du gradient μ_k, on distingue deux variantes :

2.1 Méthode du gradient à pas constant *:* dans cette variante, le pas du gradient est fixé à une valeur constante $\mu_k = \mu$. La direction de descente est donc simplement l'opposé de celle du gradient, ce qui garantit une décroissance de la fonction de coût, à condition que le pas μ soit bien choisi. Cette méthode présente l'inconvénient d'avoir une vitesse de convergence très lente lorsque l'on s'approche du minimum.

2.2 Méthode du gradient à pas asservi *:* pour tenir compte de la décroissance de la norme du gradient au voisinage du minimum, de nombreuses heuristiques ont été proposées ; par exemple, il est possible d'adapter le pas du gradient par la formule suivante :

$$u_k = u\left(1 + \left|\nabla j\right|\right) \qquad \textbf{(I.04)}$$

Typiquement, $\mu = 10^{-3}$. Cette méthode conduit à une valeur importante du pas du gradient, et donc à une décroissance rapide, loin du minimum et garantit d'autre part la convergence au voisinage du minimum, compte tenu de la faible valeur de μ.

3. Méthodes du second ordre

Dans la méthode du premier ordre, les techniques exposées sont fondées sur la seule estimation du gradient de la fonction de coût. Alors qu'on peut également établir une procédure de minimisation itérative fondée sur un développement limité au second ordre au voisinage du vecteur θ_k obtenu à l'itération *k* :

$$j\left(\theta^{k+1}\right) = j\left(\theta^k\right) + \nabla j^T\left(\theta^k\right)\left(\theta^{k+1} - \theta^k\right) + \frac{1}{2}\left(\theta^{k+1} - \theta^k\right)^T H\left(\theta^k\right)\left(\theta^{k+1} - \theta^k\right) \qquad \textbf{(I.05)}$$

Où $\mathbf{H}(\boldsymbol{\theta}^k)$ est la matrice Hessienne de la fonction de coût *J*, définie par :

$$\mathbf{II}\left(\theta^k\right) - \begin{bmatrix} \frac{\partial^2 J}{\left(\partial \theta_1^k\right)^2} & \frac{\partial^2 J}{\partial \theta_1^k \partial \theta_2^k} & \cdots & \frac{\partial^2 J}{\partial \theta_1^k \partial \theta_m^k} \\ \vdots & & & \vdots \\ \frac{\partial^2 J}{\partial \theta_m^k \partial \theta_1^k} & \frac{\partial^2 J}{\partial \theta_m^k \partial \theta_2^k} & \cdots & \frac{\partial^2 J}{\left(\partial \theta_m^k\right)^2} \end{bmatrix}$$

(I.06)

Supposons que $\boldsymbol{\theta^{k+1}}$ soit un minimum de la fonction de coût. Le gradient de celle-ci est donc nul, ce qui s'écrit :

$$0 = \nabla j^T\left(\theta^k\right) + H(\theta)\left(\theta^{k+1} - \theta^k\right)$$

(I.07)

On obtient donc l'estimation itérative recherchée :

$$\theta^{k+1} = \theta^k - H\left(\theta^k\right)\nabla j^T\left(\theta^k\right)$$

(I.08)

Pour que la direction de mise à jour des paramètres corresponde effectivement à une direction de descente, il est nécessaire que la matrice Hessienne soit définie positive (ce qui assure par ailleurs son inversibilité, et le fait que l'inverse est également définie positive).

Cette méthode de recherche itérative du minimum, appelée méthode de Newton, présente de très bonnes propriétés de convergence au voisinage du minimum, mais nécessite l'inversion d'une matrice, ce qui peut s'avérer lourd en termes de temps de calcul. Cependant, cette augmentation du temps de calcul pour une itération est compensée par la diminution du nombre d'itérations nécessaires.

D'un point de vue pratique, il est donc préférable de mettre en œuvre dans un premier temps des méthodes du premier ordre, de manière à s'approcher du minimum, avant d'utiliser cette approximation du deuxième ordre.

Sur la base de cette approximation dite de Newton, plusieurs algorithmes ont été développés.

• l'algorithme de BFGS • l'algorithme de Levenberg-Marquardt

4. L'algorithme de BFGS

Cet algorithme (du nom de ses inventeurs Broyden, Fletcher, Glodfarb et Shanno) est fondé sur une approximation de la méthode de Newton exposée précédemment. La règle de mise à jour des paramètres est définie comme suit :

$$\theta^{k+1} = \theta^k - u_{k+1}.M\left(\theta^k\right)\nabla j^T\left(\theta^k\right) \qquad \textbf{(I.09)}$$

Où le scalaire μ_{k+1} est appelé pas de descente, et où $\mathbf{M}_{k+1}$ est une approximation, calculée itérativement, de l'inverse de la matrice Hessienne, selon la formule suivante :

$$M(k+1) = M(k) + \left(\frac{1+\gamma_k{}^T M(k)\gamma_k}{\delta_k{}^T\gamma_k}\right).\frac{\delta_k\delta_k{}^T}{\delta_k{}^T\gamma_k} - \frac{\delta_k\gamma_k{}^T M(k) + M(k)\gamma_k\delta_k{}^T}{\delta_k{}^T\gamma} \qquad \textbf{(I.10)}$$

$\gamma_k = \nabla j\left(\theta^k\right) - \nabla j\left(\theta^{k-1}\right)$ *et* $\delta_k = \theta^k - \theta^{k-1}$ La valeur initiale de la matrice **M** est généralement la matrice identité (valeur à laquelle $\mathbf{M}_{k+1}$ sera également réinitialisée au cours de l'algorithme si elle s'avère ne plus être définie positive).

La valeur du pas μ_k peut être choisie comme étant la valeur permettant la diminution la plus importante de la fonction de coût. On procède alors de la façon suivante :

- initialiser μ_k à une petite valeur (typiquement 10^{-3}) ;
- mettre à jour les paramètres conformément à (I.10), puis vérifier que la condition de descente est bien respectée (soit $J(\theta_{k+1}) < J(\theta_k)$) ;
- si tel est le cas, rechercher une valeur du pas plus importante (par multiplication par un facteur supérieur à 1), et vérifier que cette valeur conduit à une fonction de coût moindre que précédemment ;
- si la condition de descente n'est pas vérifiée, multiplier le pas par un facteur inférieur à 1, et tester à nouveau la condition de descente. Cette procédure d'ajustement du pas est arrêtée soit si la décroissance de la fonction de coût est inférieure à un seuil prédéfini, soit si le nombre de tentatives atteint une limite prédéfinie.

L'intérêt de cet algorithme de BFGS réside en ce qu'il permet de s'affranchir du calcul de l'inverse de la matrice Hessienne (qui peut lui-même s'avérer délicat dans certains cas), n'estimant itérativement une approximation de cette matrice inverse suivant la formule (I.10).

Cette méthode quasi-newtonienne n'est efficace qu'à proximité du minimum de la fonction de coût : il convient donc de combiner l'utilisation de cet algorithme avec, dans un premier temps, une approche du minimum par une méthode du premier ordre.

5. L'algorithme de Levenberg-Marquardt

Cet algorithme, qui appartient également à la classe des méthodes quasi-newtoniennes, obéit à la formule suivante de mise à jour des paramètres

$$\theta^{k+1} = \theta^k - \left[H\left(\theta^k\right) + u_{k+1}.I\right]^{-1} \nabla j\left(\theta^k\right) \qquad \textbf{(I.11)}$$

Où **H**(θ*k* est la matrice Hessienne de la fonction de coût *J*, **I** est la matrice identité, et où μ_{k+1} est un scalaire appelé pas. Pour de petites valeurs du pas μ_{k+1}, cette méthode s'approche de celle de Newton, tandis que pour de grandes valeurs du pas, la méthode tend vers celle du gradient simple. En choisissant judicieusement la valeur du pas au cours de l'algorithme, il est donc possible de s'affranchir de la mise en œuvre préalable d'une méthode de gradient simple pour s'approcher du minimum.

Le calcul de l'inverse de la matrice $\left[H\left(\theta^k\right) + u_{k+1}.I\right]$ peut s'effectuer par des méthodes d'inversion directe. Néanmoins, compte tenu de la fonction de coût envisagée (I.01), il est préférable de mettre en œuvre une méthode d'inversion itérative, fondée sur la propriété suivante : étant données quatre matrice **A**, **B**, **C** et **D**

$$\mathbf{(A + BCD)^{-1} = A^{-1}. A^{-1} B (C^{-1} + DA^{-1}B)^{-1} DA^{-1}} \qquad \textbf{(I.12)}$$

Or, l'expression de la matrice Hessienne est la suivante :

$$H\left(\theta^k\right)=\sum_{n=1}^{K}\left(\frac{\partial e_n}{\partial \theta^k}\right)\cdot\left(\frac{\partial e_n}{\partial \theta^k}\right)+\sum_{n=1}^{N}\frac{\partial^2 e_n}{\partial \theta^k\left(\partial \theta^k\right)^T}e_n \qquad \textbf{(I.13)}$$

où $e_n = y_p - y$ est l'erreur de prédiction

En négligeant dans la relation (I.13) le second terme, qui est proportionnel à l'erreur, on aboutit à l'approximation suivante de la matrice Hessienne :

$$H\left(\theta^k\right)=\tilde{H}\left(\theta^k\right)=\sum_{n=1}^{K}\left(\frac{\partial e_n}{\partial \theta^k}\right)\cdot\left(\frac{\partial e_n}{\partial \theta^k}\right)^T \qquad \textbf{(I.14)}$$

Cette matrice Hessienne approchée obéit à la formule de récurrence suivante :

$$\tilde{H}^n=\tilde{H}^{n-1}+X^n\left(X^n\right)^T ou \quad X^n=\frac{\partial e_n}{\partial \theta^k}=\frac{\partial y_n}{\partial \theta^k}$$ pour n=1….N

Par définition, et en fixant comme valeur initiale : $\tilde{\mathbf{H}}^0=\mu_k\mathbf{I}$ on a donc :

$\tilde{\mathbf{H}}^N=\tilde{\mathbf{H}}+\mu_k\mathbf{I}$.

L'utilisation du lemme d'inversion énoncé précédemment avec :

$A=\left(\tilde{H}^n\right)^{-1}, B=X^n, C=I, D=\left(X^n\right)^T$ permet alors d'écrire :

$$\left(\tilde{H}^n\right)^{-1}=\left(\tilde{H}^{n-1}\right)^{-1}-\frac{\left(\tilde{H}^{n-1}\right)^{-1}X^n\left(X^n\right)^T\left(\tilde{H}^{n-1}\right)^{-1}}{I+\left(X^n\right)^T\left(\tilde{H}^{n-1}\right)X^n} \qquad \textbf{(I.15)}$$

Il est alors possible de calculer itérativement l'inverse de la matrice : $\tilde{\mathbf{H}}^N=\tilde{\mathbf{H}}+\mu_k\mathbf{I}$

Notons que cette méthode de calcul approché de l'inverse de la matrice découle de l'approximation (I.14), qui n'est valable que pour de faibles valeurs de l'erreur de prédiction $\boldsymbol{e_n}$, et donc pour des valeurs de $\boldsymbol{\theta}$ proches de la valeur optimale. Le domaine de validité de l'approximation Newtonienne, a priori étendu par l'ajout du terme $\mu_k\mathbf{I}$ dans la formule (I.11), est finalement restreint pour pouvoir calculer efficacement l'inverse de cette matrice Hessienne augmentée.

6. Algorithme de rétropropagation

Dans cet algorithme, on dispose d'un ensemble d'exemples qui sont des couples entré-sortie. A chaque étape, un exemple est présenté à l'entrée du réseau. On réalise deux étapes successives :

Une propagation de l'entrée vers la sortie du réseau, où chaque cellule de la couche cachée somme toutes ses entrées et fait passer le signal obtenu à travers sa fonction de transfert, nous avons vu que le vecteur de sortie n° k a pour expression :

$$Y_k = g\left[Z\left[f\left(WX_k + W_0\right)\right] + Z_0\right] \quad \textbf{(I.16)}$$

Le vecteur de sortie est comparé à celui que l'on désire obtenir, le vecteur D_K.

On déduit l'erreur : $e_K = D_K - Y_K$ **(I.17)**

Une Rétropropagation de l'erreur

L'apprentissage du réseau consiste à modifier, à chaque pas d'apprentissage, les poids et les biais afin de minimiser la somme des carrés des erreurs en sortie. La méthode de rétropropagation est basée sur la technique du gradient.

La quantité à minimiser, à chaque pas d'apprentissage k est la variance de l'erreur en sortie du réseau.

$$E_K = e_K^T e_K = \frac{1}{2}\left(D_K^T.D_K + Y_K^T Y_K - 2Y_K^T D_K\right) \quad \textbf{(I.18)}$$

$$\nabla E_{K/Z} = \frac{1}{2}\nabla\left[Y_K^T Y_K - 2Y_K^T D_K\right]_Z \quad \textbf{(I.19)}$$

Si les fonctions de transfert sont identiques pour les deux couches ($F=g$), la sortie du réseau est donnée par :

$$Y_K = f\left[Z.h_K + Z0\right] \quad \textbf{(I.20)}$$

Ou h_K est le vecteur de sortie de la couche cachée.

Le gradient de E_K par rapport à la matrice Z peut se calculer comme suit :

$$\nabla E_{K/Z} = \frac{\partial E_K}{\partial Z} = \frac{\partial E_K}{\partial Y_K} \cdot \frac{\partial Y_K}{\partial (Z.h_K + Z_0)} \cdot \frac{\partial (Z.h_K + Z_0)}{\partial Z} \quad \textbf{(I.21)}$$

D'après l'expression de E, la première dérivée partielle est :

$$\frac{\partial E_K}{\partial Y_K} = Y_K - D_K \quad \textbf{(I.22)}$$

La deuxième dérivée partielle dépend du type de fonction de transfert utilisée. Dans le cas de la sigmoïde :

$$f(x) = \frac{1}{1+e^{-x}} \quad \textbf{(I.23)}$$

et $f'(x) = f(x).[1 - f(x)]$ **(I.24)**

La deuxième dérivée partielle a pour expression :

$$\frac{\partial Y_K}{\partial (Z.h_K + Z_0)} = f(Z.h_K + Z_0).*[1 - f(Z.h_K - Z_0)] = Y_K.*(1 - Y_K) \quad \textbf{(I.25)}$$

(Pour la programmation le 1 représente un vecteur unitaire de même taille que le vecteur de sortie Y_K).

La troisième dérivée partielle est simple à calculer :

$$\frac{\partial (Z.h_K + Z_0)}{\partial Z} = h_K^T \quad \textbf{(I.26)}$$

La mise à jour se faisant dans le sens inverse du gradient, la matrice des poids Z de l'étape suivante (k+1) est :

$$Z(k+1) = Z(k) - \eta \nabla E_{K/Z} = Z(k) + \eta (D_K - Y_K).*Y_K.*(1 - Y_K).h_K^T = Z(k) + \eta \partial S.h_K^T \quad \textbf{(I.27)}$$

Avec η est le pas d'apprentissage.

La même méthode sera utilisée pour la mise à jour du vecteur des biais Z_0 ; on obtient :

$$Z_0(k+1) = Z_0(k) - \eta \nabla E_{K/Z_0} = Z_0(k) + \eta (D_K - Y_K).*Y_K.*(1 - Y_K) = Z_0(k) + \eta \partial S \quad \textbf{(I.28)}$$

Par analogie à l'expression précédente de la mise a jour de la matrice Z, avec h_k le vecteur de sortie de la couche cachée ; on obtient :

$$W(k+1) = W(k) - \eta \nabla E_{K/W} = W(k) + \eta.Z(k)^T \partial S.*h_K.*(1 - h_K).X_K^T = W(k) + \eta \partial h.X^T \quad \textbf{(I.29)}$$

Et

$$W_0(k+1) = W_0(k) - \eta \nabla E_{K/W0} = W_0(k) + \eta . Z(k)^T \partial S . * h_K . * (1 - h_K) = W_0(k) + \eta \partial h. \quad \textbf{(I.30)}$$

Le processus de calcul du gradient de l'erreur et d'adaptation des poids est répété jusqu'au moment où un minimum de l'erreur, ou un point qui en est suffisamment proche, est trouvé. La fin de la phase d'apprentissage peut être décidée lorsque l'erreur moyenne observée à la sortie du réseau devient inférieure à un seuil prédéfini, ou encore lorsque l'amplitude moyenne du gradient de cette erreur devient très faible, puisque, par définition, le gradient est nul au minimum de la fonction de coût.

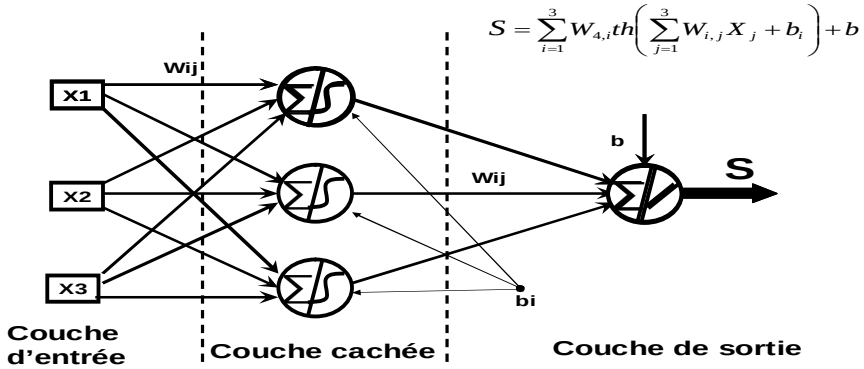


Figure I.1: Principe de la rétropropagation.

ملخص

من خلال دراستنا هده قمنا باستخدام الألياف العصبة الاصطناعية الديناميكية على نماذج فيزيائية غير خطية من اجل إعداد نموذج رياضي علبة سوداء، حيث استعملنا منبئات عصبية ممثلة في نماذج إدخال وإخراج ونماذج الحالة ، كما أن التشويش العشوائي المضاف لحق بالنموذج على مختلف المستويات (معادلة الخروج، الحالة، الاثنين معا)، النتائج المحصل عليها باستعمال المنبئات العصبية بنماذج الحالة كانت أكثر نجاعة من منبئات عصبية ممثلة في نماذج إدخال وإخراج

الكلمات المفتاحية : الألياف العصبة الاصطناعية الديناميكية. نموذج رياضي علبة سوداء. نماذج فيزيائية غير خطية . المنبئات العصبية.المسننات.تمثيل إدخال- إخراج وتمثيل الحالة .

Abstract

Through our study, we have presented some applications of nonlinear dynamical neural networks for dynamical black box modelling of the nonlinear process. and we have used the corresponding input-output and state-space neural predictors, the random variables (noise) can be introduced in different levels (state equation, output equation, and both) .The results of simulation show that since state-space models constitute a larger class of nonlinear dynamical models, there is an advantage in making the assumption of a state-space description, and in using the corresponding state-space neural predictors.

Keywords: dynamical neural networks .black box modelling . nonlinear process. neural predictors. Gears. Input-output and state-space representation.

Résumé

Dans ce mémoire, nous présentons des applications des réseaux de neurones dynamiques pour la modélisation boite - noir d'un processus non linéaire .Les prédicteurs neuronaux utilisés sont représentés par des modèles aussi bien d'entrée- sortie que des modèles d'état. Les perturbations aléatoires qui affectent le processus sont introduites à différents niveaux (sortie, l'état, et les deux a la fois) .Les résultats de simulation obtenus avec des prédicteurs neuronaux d'état sont largement supérieurs à ceux obtenus par des prédicteurs entrée- sortie.

Mots clés : réseaux de neurones dynamiques, modélisation boite-noir, processus non linéaire. prédicteurs neuronaux. Engrenage. Représentation entrée-sortie représentation d'état