UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN ÉLECTRONIQUE INDUSTRIELLE

PAR
Bruno MBA MEGNER

ÉTUDE DE L'APPLICATION DES RÉSEAUX DE NEURONES À LA RECONSTITUTION DES SIGNAUX POUR UN SYSTÈME DYNAMIQUE NON LINÉAIRE

DÉCEMBRE 1998

“ À mes parents,

et enfants”

RÉSUMÉ

Dans le domaine du traitement des signaux, le schéma fonctionnel du système de mesure à étudier comporte en général deux blocs assurant des fonctions distinctes: la conversion et la reconstitution. Lorsque la grandeur physique mesurée est transformée en signal électrique par l'élément sensible du système de mesure, on parle alors de mesure électrique. L'utilisation courante de ce type de mesure se justifie par la facilité de manipulation qui caractérise les signaux électriques. Le processus de reconstitution du mesurande (grandeur physique mesurée) consiste à estimer ce dernier à partir de son équivalent électrique correspondant issu du bloc de conversion. On comprend par là que ce bloc établit une relation de causalité entre le mesurande et le signal électrique qu'il fournit. Ainsi, pour effectuer la reconstitution, une approche analytique nécessite d'abord de procéder à l'identification de la réponse impulsionnelle du bloc de conversion afin de connaître le modèle de la relation entre le mesurande et son équivalent électrique; puis d'estimer les paramètres de ce modèle.

Dans ce travail, nous proposons une méthode de reconstitution de signaux basée sur les réseaux de neurones artificiels (RNA) pour des systèmes dynamiques linéaires et non linéaires. Cette méthode se base sur une banque de données de références de qualité, composées de signaux d'entrées et de sortie du bloc de conversion. Cette banque de données est alors utilisée pour étalonner le modèle de reconstitution basé sur les RNA. Cette procédure nous permet de reconstituer le mesurande à la fois pour des systèmes de conversions linéaires et non linéaires sans connaissance analytique du système de conversion. Nous validons la

méthode proposée en l'appliquant sur un système de mesure de type spectrométrique selon des données synthétiques et expérimentales. Les résultats montrent que le modèle autorégressif d'un réseaux de neurones multicouches possèdent de bonnes capacités en matière d'apprentissage (étalonnage) et de généralisation (identification) des systèmes de conversion à la fois linéaires et non linéaires. Le cas spécifique de données expérimentales est issue d'un appareil d'analyse de spectre optique. Ces données, contrairement aux données synthétiques utilisés lors de l'entraînement de nos réseaux, sont fortement bruitées. Mais, l'aptitude des réseaux à interpoler n'est pas autant affectée. Nous démontrons alors que la méthode proposée offre une robustesse suffisante face aux signaux entachés de bruit sans avoir à ajuster de paramètres à l'algorithme de reconstitution, contrairement aux autres méthodes qui exigent l'ajustement d'un paramètre pour assurer une qualité acceptable.

Une étude comparative avec d'autres méthodes populaires utilisées en reconstitution de signaux spectrométriques révèle que la méthode proposée offre des résultats équivalents voire meilleurs dans le cas des systèmes linéaires, et largement supérieurs dans le cas des systèmes non linéaires. De plus, la complexité arithmétique et la forme régulière et récurrente de nos équations du modèle avantage notre méthode pour une implantation en technologie l'intégration à très grande échelle (ITGE) sous une architecture hautement parallèle (systolique). Dans cet objectif, l'étude de la quantification des variables et paramètres du réseau de neurones démontre qu'une architecture composée d'unités arithmétiques et logiques de 16×16 bits offre une précision de calcul suffisante dans un compromis entre la vitesse d'exécution et la surface d'intégration. Cela permet ainsi la proposition d'architectures systoliques pour la méthode de reconstitution proposée.

REMERCIEMENTS

Je remercie profondément Daniel MASSICOTTE, Professeur à l'Université du Québec à Trois-Rivières, qui a bien voulu m'apporter son soutien au cours de ces années de maîtrise. Il m'a fait l'honneur de m'accueillir au sein de son équipe au laboratoire de Microélectronique dans lequel les outils mis à ma disposition ont été très indispensables à la réalisation de cette étude.

Je tiens à exprimer ma reconnaissance à tous les autres professeurs qui m'ont accordé leur attention quant à mon adaptation à ce système scolaire qui m'était peu familier.

Aussi, je n'oublierai pas de témoigner ma gratitude aux étudiants de maîtrise m'ayant précédé et, de ce fait, apporté des conseils bénéfiques tirés de leur expérience et une assistance technique par les travaux qu'ils ont menés respectivement dans mon domaine.

Enfin, toute ma profonde reconnaissance au Programme Canadien de Bourses de la Francophonie qui m'a assuré un soutien financier et un encadrement m'ayant permis de mener mes études dans d'excellentes conditions.

A ma famille, amis et collègues, un grand merci pour votre soutien de quelque nature que ce soit.

TABLE DES MATIÈRES

Page

LISTE DES SYMBOLES ET ABRÉVIATIONS

A_i amplitude du pic i

b biais d'un neurone

$D^{étal}$ banque de données d'étalonnage ou d'apprentissage

D^{test} banque de données de test

D^{val} banque de données de validation

e_q erreur de quantification

ε erreur de reconstitution du réseau de neurones

$\varepsilon^{étal}$ erreur relative moyenne lors de l'étalonnage du réseau de neurones

ε^{test} erreur relative moyenne lors de l'optimisation du réseau de neurones

ε^{val} erreur relative moyenne lors de la généralisation

f(.) fonction de décision d'un neurone formel

g(.) fonction de transfert du bloc de conversion

G(.) opérateur de la fonction NNOE

k nombre d'itérations

M dimension de la réponse impulsionnelle g

m_x nombre de retards prélevés sur la sortie $\hat{x}$ du réseau

m_y nombre de retards prélevés sur l'entrée $\tilde{y}$ du réseau

n_c	nombre de neurones sur la couche cachée
N	nombre de points sur un signal
$N^{étal}$	nombre de points sur le signal d'étalonnage
N^{test}	nombre de points sur le signal de test
N^{val}	nombre de points sur le signal de validation
N_p	nombre de pics dans le signal
P_i	position du pic i
P_{opt}	point optimal d'obtention de la meilleure reconstitution
q	pas de quantification
R	opérateur de la fonction NNOE
s	somme pondérée des entrées d'un neurone
$\mathbf{W}1$	matrice des poids de connexion à l'entrée de la couche cachée du réseau
$\mathbf{W}2$	matrice des poids de connexion à la sortie de la couche cachée du réseau
$\mathbf{W}1^{opt}$	matrice des poids de connexion optimale à l'entrée de la couche cachée
$\mathbf{W}2^{opt}$	matrice des poids de connexion optimale à la sortie de la couche cachée
$\dot{x}$	signal idéal ou mesurande, entrée du système de mesure
$\hat{x}$	signal estimé du mesurande, sortie du réseau de neurones
$\dot{x}^{étal}$	signal idéal d'étalonnage
$\dot{x}^{étal}_{norm}$	signal idéal d'étalonnage normalisé
$\hat{x}_q$	signal estimé par le réseau quantifié
y	signal de sortie de conversion non bruité

$\tilde{y}$	signal brut de mesure ou signal de sortie de conversion bruité
β	paramètre de régularisation du filtre de Kalman
λ	intervalle de mesure (longueurs d'ondes)
η	perturbation externe ou bruit
σ_i	largeur du pic i
σ_n^2	quantité de bruit sur le signal
φ	matrice de régression de l'algorithme NNOE
φ_q	matrice de régression du réseau quantifié

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1

INTRODUCTION

1.1 Problématique

Dans le domaine du traitement du signal concernant les systèmes de mesure, le problème majeur le plus souvent rencontré se trouve comme étant celui de la reconstitution des signaux. En effet, lors de la mesure d'une grandeur physique *x(t)* (mesurande), deux processus importants se déroulent au sein du système de mesure : la *conversion* et la *reconstitution*. L'élément sensible, ou capteur, du système de mesure accomplit la première tâche qui est la conversion. La mesure étant électrique, l'ensemble formé par le capteur et son circuit de conditionnement fournit un signal électrique *y(t)* correspondant à la grandeur physique prélevée. Mais cet équivalent électrique, sous l'effet de certaines influences apparaissant lors des opérations effectuées dans la chaîne de mesure, est souvent obtenu sous forme dégradée. On le notera *y(t)*. La phase de reconstitution consiste alors à estimer le mesurande à partir de ce signal bruité. Les méthodes utilisées sont généralement basées sur des algorithmes analytiques plus ou moins complexes. Ces derniers nécessitent une parfaite connaissance du modèle mathématique de l'instrument de mesure et des différentes causes

de déformation du signal. Ces causes peuvent avoir des origines à la fois internes et externes aux systèmes.

Compte tenu des problèmes inhérents à la connaissance du système de conversion, ces algorithmes de reconstitution des signaux deviennent fort difficile à utiliser surtout dans le cas de système de conversion non linéaire. Ainsi, nous utiliserons une méthode basée sur l'identification du modèle inverse du système de conversion. Elle consiste à effectuer une estimation du mesurande, sans besoin direct de la connaissance du modèle de la réponse impulsionnelle du capteur. Cette méthode est basée sur les réseaux de neurones et leurs capacités d'identifier le modèle inverse du système de conversion dans le cas linéaire et non linéaire.

Le choix de cette méthode de reconstitution a aussi été basé sur le besoin de satisfaire les critères nécessaires à une implantation en technologie ITGE (Intégration à Très Large Échelle). En effet, les réseaux de neurones montrent un fort degré de parallélisme rendant possible l'application d'une approche systolique pour un développement architectural et algorithmique.

1.2 Objectifs

Les objectifs spécifiques poursuivis dans ce projet sont de démontrer la possibilité d'utiliser les réseaux de neurones dans le processus de reconstitution des signaux dans un système de mesure. Ayant remarqué que les signaux issus de ce genre de système sont souvent déformés, et ce par certaines imperfections et non linéarités des différents éléments

de la chaîne de mesure, cette étude sera orientée essentiellement vers les systèmes dynamiques de types linéaire et non linéaire. Aussi, la proposition d'une architecture sera faite sur la base d'une étude sur l'implantation de l'algorithme en technologie ITGE.

1.3- Méthodologie

Pour atteindre l'objectif visé par ce projet, il est nécessaire de pouvoir situer un réseau de neurones dans une chaîne de mesure conformément à la tâche à exécuter. Ensuite, après avoir compris le principe de fonctionnement du réseau artificiel, on procédera à sa configuration et à son étalonnage.

Ainsi, en considérant un réseau de neurones comme une *boîte noire* accessible uniquement par ses entrées et ses sorties, on applique à ses entrées un signal $y(t)$ qui est l'équivalent électrique de la grandeur physique $x(t)$ à mesurer. Après les étapes de configuration du réseau de neurones et de propagation du signal $y(t)$ dans ce réseau pour son entraînement, un résultat $\hat{x}(t)$, représentant l'estimation du signal désiré, est fourni par le réseau à sa sortie. Ces travaux seront facilités grâce à l'existence d'une banque d'algorithmes d'entraînement des réseaux de neurones programmées en Matlab et dédiées à ce genre d'application.

Étant donné que le systèmes de mesure étudiés dans ces travaux seront de types linéaire et non linéaire, la structure des réseaux de neurones multicouches sera adoptée et étudiée afin de répondre au problème posé par le système. Cette structure des réseaux pourra s'étalonner à l'aide de plusieurs algorithmes basés sur les modèles tels que le *Output Error* (Erreur de

sortie) et le *Time Delay Neural Network* (TDNN). Chaque modèle pourra être appliqué afin de comparer ses résultats et pour différentes conditions du système de conversion étudié.

La structure du présent mémoire est divisée en six chapitres bien distincts dont ce premier qui a situé la problématique, les objectifs et la méthodologie à suivre. Au chapitre 2, une introduction aux réseaux de neurones artificiels est faite en expliquant leurs principes de fonctionnement, leurs différentes caractéristiques et certaines des techniques d'apprentissage utilisées pour identifier le système de conversion. Aussi, on y effectue une étude de l'algorithme utilisé pour la reconstitution des signaux issus des systèmes linéaires et non linéaires. Dans le chapitre 3, la structure d'un système de mesure est étudiée et le bloc abritant le processus de reconstitution est identifié. C'est ce bloc qui sera remplacé par les réseaux de neurones artificiels. Au chapitre 4, on définit un cas d'application de la méthode proposée, qui sera dans ce travail du type spectrométrique, afin de valider la méthode servant à l'étalonnage des réseaux de neurones. Les résultats devront nous permettre de valider notre méthode d'entraînement des réseaux de neurones et ce, pour des données synthétiques et expérimentales. Pour des fins de comparaison, certaines méthodes populaires seront sollicitées avec les mêmes données. Enfin, une étude des possibilités d'implantation en technologie VLSI est effectuée dans le chapitre 5. Plus principalement, on effectuera une estimation de la quantité de mémoire requise pour assurer une reconstitution acceptable ainsi qu'une étude de l'effet de quantification. Une architecture sera alors proposée. Le travail se termine par le chapitre 6, conclusions, dans lequel on fait une synthèse de l'étude réalisée et des résultats obtenus.

CHAPITRE 2

SYSTÈME DE MESURE

2.1 Introduction

Dans ce travail, le but principal poursuivi est d'étudier les possibilités d'utiliser les réseaux de neurones artificiels dans la chaîne fonctionnelle d'un système de mesure dynamique non linéaire. Aussi, en regardant les réseaux de neurones comme des systèmes entiers, on devra s'assurer de comprendre la tâche qui leur est assignée dans un système de mesure, dans lequel ils deviennent des sous systèmes. La première partie de ce chapitre sera donc consacrée à une revue globale des définitions de base dans le domaine de la mesure ; suivie des principales caractéristiques d'un système de mesure. Puis les méthodes d'étalonnage seront présentées afin d'avoir une idée des opérations que devraient effectuer les réseaux de neurones. Le processus de la reconstitution de mesurandes sera ensuite évoqué avant de clore avec le chapitre avec un survol des différentes méthodes classiques couramment utilisées.

2.2 Généralités

2.2.1 Quelques notions de base

La première question que l'on devrait se poser lorsque l'on veut effectuer une

mesure est de savoir ce que ce terme signifie exactement. En effet, que signifie la *mesure* ? C'est à partir de cette définition que d'autres notions liées à ce domaine seront énumérées.

La mesure – Parmi de nombreuses définitions qui ont été faites sur ce terme [LEG94], il en ressort un point commun. Ce dernier définit la mesure comme étant un procédé opérationnel pendant lequel on applique des valeurs numériques (nombres) aux éléments appartenant à une certaines classes de caractéristiques. Ce procédé d'application se fait selon des règles bien définies. C'est ainsi qu'on est en mesure de définir, de façon quantitative, les propriétés des objets ; et de vérifier numériquement les lois ou d'en établir la forme empirique. Toutefois, certaines conditions devraient être respectées pour réussir une opération de mesure. En effet, en s'assurant que l'information à prélever est distant du point de lecture, il faudra que celle-ci soit démunie de toute impureté découlant de diverses sources de perturbations ou de bruit. Ces phénomènes parasites auraient pour conséquence de déformer le mesurande avant même d'être traitée. Le respect de ces conditions assurera une authenticité aux informations extraites des signaux électriques générés lors des différentes phases de la mesure.

Système de mesure – Un système de mesure peut être défini comme un ensemble de sous-systèmes techniques qui, travaillant ensemble, ont pour fonction d'acquérir une information sur une grandeur physique difficilement accessible [LEG94]. Le schéma général d'un système de mesure est représenté à la figure 2.1.

Ainsi, pour avoir le plus d'informations sur le mesurande, le système de mesure devra d'abord le transformer en un signal électrique. Puis, traiter ce dernier par divers processus afin de rencontrer les exigences pré-requises.

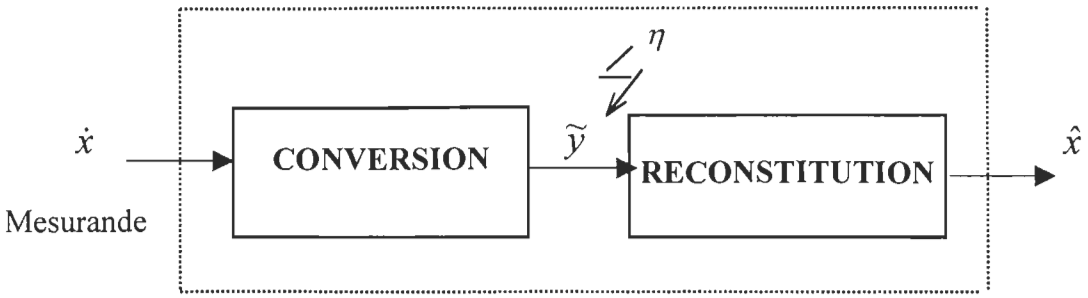


Figure 2.1: Schéma général d'un système de mesure

En somme, la fonction principale d'un système de mesure est donc de fournir un estimé de la grandeur à mesurer sous une forme facilement compréhensible. D'une manière générale, on le décompose en deux blocs distincts réalisant chacun une fonction bien spécifique: conversion et reconstitution. Le bloc de conversion est chargé de prélever le mesurande et de le transformer en un signal facile à interpréter. Le bloc de reconstitution permet quant à lui, de retrouver un estimé de la grandeur physique mesurée à partir du signal de sortie du bloc de conversion.

La conversion – Comme vu précédemment, ce processus consiste à prélever le mesurande et le transformer en un signal facilement compréhensible. De façon général, on obtient un signal électrique : on parle alors de mesure électrique. Dans un système de mesure, la conversion s'effectue en une série de transformations qui sont les suivantes :

- conversion A/A : c'est une conversion analogique - analogique qui a pour but de transformer le mesurande (grandeur analogique) en un signal électrique équivalent et de même nature. Cette opération est réalisée à l'aide d'un capteur ou d'un transducteur

spécifique dans lequel est implantée une loi, une propriété connue ou une condition physique reliée à la grandeur à mesurer.

- conversions A/N et N/A: La conversion A/N (analogique - numérique) établit une correspondance entre un vecteur d'entrée continu et une succession de messages numériques. L'utilisation d'un ordinateur ou d'un processeur justifie cette transformation. La conversion N/A (numérique - analogique) intervient lors de la tentative de restitution du signal d'entrée de la conversion A/N après son traitement par un processeur ; et pour des fin d'affichage et de lecture. Cependant, il faut noter que ces deux types de conversions se font sur deux paramètres principaux : l'amplitude et le temps.

- conversion N/N : désignant la conversion numérique - numérique, elle est effectuée par des blocs assurant des tâches comme le conditionnement des signaux, le filtrage, l'analyse statistique des données de mesures, etc.

La reconstitution – Cette étape est décrite plus en détail à la section 2.4. Mais d'une façon succinte, on la définit comme étant un processus par lequel on tente une estimation assez représentative du mesurande à partir du signal brut de mesure sorti du bloc de conversion.

2.2.2 Caractéristiques métrologiques

Les caractéristiques métrologiques des différents éléments ou instruments d'un système de mesure définissent les performances de ce dernier. Ces caractéristiques sont nombreuses et les principales sont [LEG94]:

- la fidélité : elle désigne l'aptitude d'un capteur à donner la même indication pour la même grandeur mesurée. Si cette condition n'est pas respectée, on a affaire à un indicateur.

- la résolution: c'est l'ensemble des valeurs couvertes par la sortie d'un instrument de mesure.

- la sensibilité : c'est le rapport entre la variation du signal indiqué et la variation correspondante du signal d'entrée.

- la mobilité : elle est définie comme étant l'aptitude d'un capteur à réagir aux moindres variations du signal mesuré.

- la rapidité : on dit qu'un système rapide s'il suit l'évolution du mesurande.

- la discrétion : elle caractérise l'aptitude à ne pas modifier la valeur de la grandeur mesurée.

- erreurs de mesure : ce sont les erreurs traduisant les écarts entre la valeur idéale mesurée et celle qui est indiquée par le capteur. Elles sont souvent reliées à la dégradation continue du signal lors de son traitement avec des éléments dotés d'imperfections dans la chaîne de mesure.

2.3 Étalonnage d'un système de mesure

L'étalonnage est l'opération qui permet d'identifier la relation existant entre le mesurande et le signal électrique délivré par le capteur ou le système de conversion. Ce procédé revient simplement à construire la fonction de transfert ou le modèle mathématique du bloc de conversion. On rencontre cependant deux types d'étalonnage : étalonnage statique et étalonnage dynamique. Lors d'un étalonnage statique, on ne tient pas compte du facteur temps dans le fonctionnement du système, ce qui est le cas durant un étalonnage dynamique.

En se basant sur la figure 2.1, un étalonnage statique consisterait à relever les performances du système de mesure dans des conditions comme : faire varier une grandeur parmi les grandeurs d'entrées, désirées, et les bruits pendant que les autres sont maintenues constantes. Cette variation se fait par palier et on prélève à chaque fois la grandeur de sortie après s'être assuré de la disparition de tout effet transitoire ou temporel.

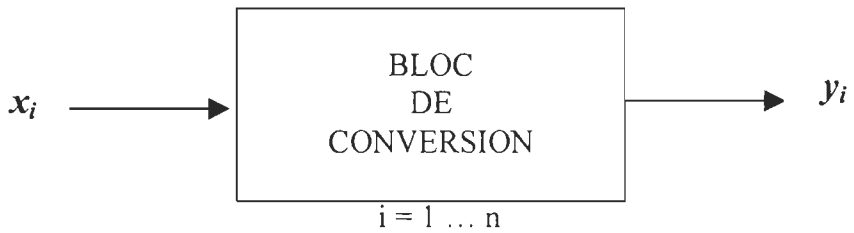


Figure 2.2 : Bloc de conversion à étalonner pendant *n* observations.

En représentant le bloc de conversion d'un système de mesure par l'illustration de la figure 2.2, le processus d'étalonnage statique se résumerait à l'opération qui consisterait à approximer la relation existant entre les sorties y_i et les entrées x_i. On classifie les différentes méthodes d'étalonnage statique utilisées en deux catégories distinctes : les méthodes *paramétrées* et les méthodes *non paramétrées* [LEG94]. En cherchant à lier chaque sortie y_i à une entrée x_i, le modèle d'étalonnage peut se présenter comme suit :

$$y_i = \varphi(x_i) + \varepsilon_i,\ i = 1 \ldots n \qquad (2.1)$$

avec ε_i l'erreur de moyenne zero et de variance σ^2 et $\varphi(.)$ le régresseur ou la fonction de

régression.

Dépendemment du niveau de complexité du problème à résoudre, on choisira le type de méthode, paramétré ou non paramétré, requis. En général, une méthode paramétrée suppose une connaissance de la fonction de régression et une détermination d'un nombre fini de paramètres. Cette difficulté est contournée dans une méthode non paramétrée. En effet dans celle-ci, la fonction de régression appartient à un espace de fonctions caractérisées par leurs différentes propriétés de lissage. La fonction choisie est celle dont les propriétés satisfont le mieux aux données d'étalonnage.

2.4 Reconstitution de mesurandes

En considérant que le modèle mathématique du bloc de conversion est linéaire et qu'il n'y a aucune influence perturbatrice sur le signal transmis, la sortie de conversion est liée au mesurande par l'équation:

$$y(t) = \int g(t-t')x(t')dt' = x(t)*g(t) \qquad (2.2)$$

où *g(t)* est la réponse impulsionnelle du bloc de conversion et * représente le produit de convolution.

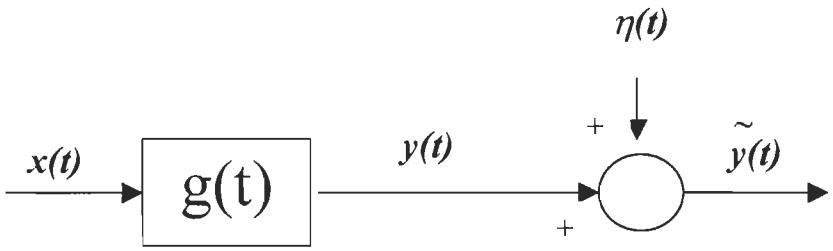


Figure 2.3 : Schéma fonctionnel du bloc de conversion

En pratique, la conversion est affectée par certaines perturbations $\eta(t)$ (figure 2.3) et l'équation 2.1 devient :

$$\tilde{y}(t) = g(t) * x(t) + \eta(t) \tag{2.3}$$

Le problème de la reconstitution consistera donc à estimer le signal $x(t)$ à partir de cette équation. Cela suppose donc que connaissant la réponse impulsionnelle de l'instrument de mesure et son signal de sortie, on essaye de retrouver le mesurande. Cela revient à effectuer une opération de déconvolution, une méthode consistant à passer par le domaine des fréquences afin de trouver l'inverse de la réponse impulsionnelle comme indiqué dans l'équation:

$$X(f) = Y(f)G^{-1}(f) \tag{2.4}$$

où X(f), Y(f) et G(f) sont les signaux x(t), y(t) et g(t) dans le domaine des fréquences respectivement.

Cette opération se complique si on fait apparaître le bruit additif $\eta(t)$ selon l'équation 2.3. La nouvelle équation sera alors :

$$X(f) = Y(f)\left[G(f) + \eta(f)\right]^{-1} \tag{2.5}$$

Ainsi, on voit clairement que cette opération aura pour conséquence une forte amplification du bruit sur l'estimation du mesurande après la transformée inverse.

Des résultats erronés peuvent alors être obtenus et ce, pour diverses raisons. D'abord, l'identification de la fonction de transfert du bloc de conversion se fait par des méthodes d'identification qui approximent cette fonction. Cela implique donc que l'on obtient un estimé de la fonction de transfert. Puis, pour peu que cet estimé ne rencontre pas tous les critères d'inversibilité, on est amené à chercher la pseudo-inverse en introduisant certaines contraintes [ALI96]. Cela a finalement pour conséquence l'obtention d'un signal reconstitué qui n'a peut-être rien à voir avec l'information recherchée sur le mesurande.

Les équations (2.2) et (2.3) sont présentées de façon très simplifiées. En effet, le signal de sortie de conversion étant toujours bruité, on exprimera alors le signal x(t) de la façon suivante :

$$x(t) = R\left(\tilde{y}(t), g(t)\right) \tag{2.6}$$

où R représente un opérateur effectuant l'inversion de la convolution afin d'obtenir le mesurande $\dot{x}(t)$.

Les méthodes de reconstitution des signaux peuvent être classées en six grandes catégories [LEG94]:

- les méthodes directes: elles consistent à numériser le bloc de conversion;

- les méthodes itératives: la plus connue étant celle de Jansson [CRI91], elle consiste à restaurer le mesurande par des approximations successives;

- les méthodes stochastiques: un exemple de ces méthodes est le filtrage de Kalman [MAS95]. Elles travaillent de façon récursive et tentent d'obtenir un estimé optimal en minimisant la variance entre le signal idéal et le signal reconstitué à chaque pas d'échantillonnage.

- les méthodes de transformations: celle de Tikhonov en est un exemple [LEG94]. Ces méthodes travaillent dans le domaine fréquentiel et représentent l'équation de convolution sous forme de transformée de Fourrier.

- les méthodes variationnelles: elles sont basées sur des approximations polynomiales de la solution. C'est le cas des fonctions *splines*.

2.5 Méthode de reconstitution basée sur les réseaux de neurones

Toutes les méthodes mentionnées ci-dessus nécessitent de connaître la réponse impulsionnelle de l'instrument de mesure. Elles ont donc souvent besoin de certaines opérations d'identification des modèles. Mais le succès de ces opérations, dépendant de la qualité des données utilisées, peut s'avérer cependant très délicat. Ce sont alors ces différents obstacles que nous allons éviter en utilisant les réseaux de neurones pour effectuer la tâche de reconstitution.

Contrairement aux méthodes citées à la page précédente, la réponse impulsionnelle de l'instrument de mesure n'est pas requise.

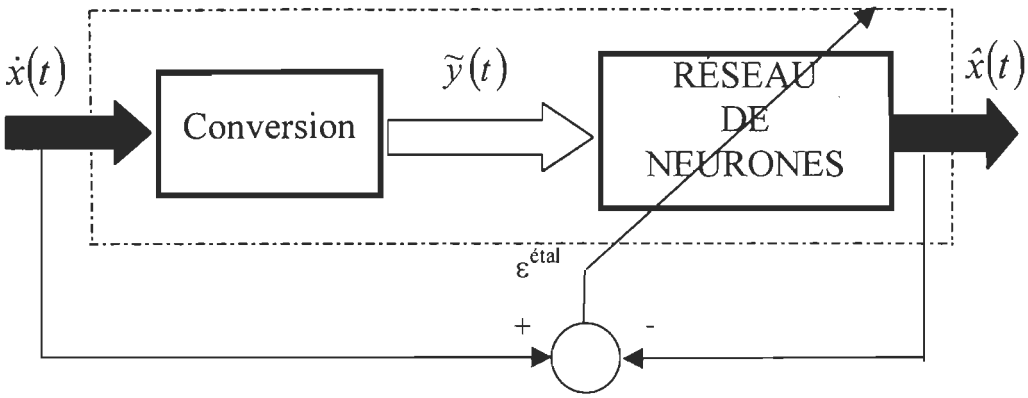


Figure 2.4 : Procédure d'apprentissage des réseaux de neurones

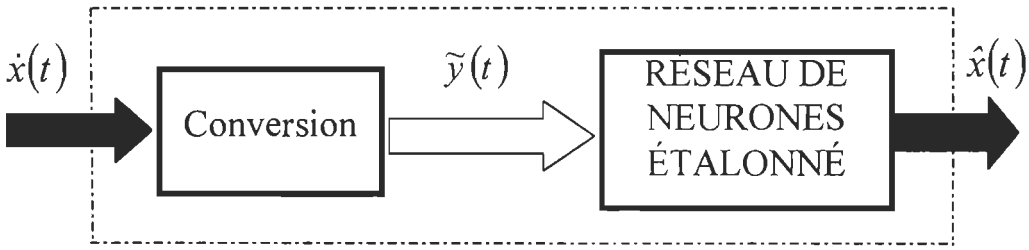


Figure 2.5 : Application des réseaux de neurones à la reconstitution des signaux

Seul un apprentissage (figure 2.4) avec des données assez représentatives sera requis pour obtenir les résultats escomptés. Cette approche est illustrée sur la figure 2.5.

Ainsi, les réseaux de neurones devraient subir un entraînement afin d'être capables de résoudre ce problème de reconstitution. Comparativement aux méthodes analytiques, la tâche d'identification du système de conversion est remplacée par l'ajustement des

paramètres des réseaux de neurones. Cette opération est faite en se servant des données d'entrée et de sortie du système de conversion. Les paramètres ajustés sont principalement la topologie du réseau, les valeurs des poids des connexions entre les neurones, le nombre d'entrées, et certains paramètres d'apprentissage dont le nombre d'itérations et le critère d'arrêt de l'entraînement [MAS99].

2.6 Conclusion

Pour conclure ce chapitre, on dira que le problème de la reconstitution des signaux dans un système de mesure est primordiale. En effet, pour atteindre le but recherché en mettant au point un tel système, il faudra s'assurer que toutes ses parties fonctionnelles accomplissent bien leurs tâches respectives. Bien que certaines perturbations aient lieu pendant la capture et la conversion du mesurande, l'équation majeure à résoudre réside dans la tentative d'extraction de l'information issue du bloc de conversion : c'est le processus de reconstitution. Les méthodes analytiques requérant une identification minutieuse du modèle mathématique de l'instrument de mesure, celle-ci est évitée en utilisant les réseaux de neurones artificiels.

Au chapitre suivant, nous allons présenter les réseaux de neurones et plus particulièrement ceux qui présentent les qualités requises pour résoudre le problème de reconstitution des signaux d'un système de mesure dynamique non linéaire.

CHAPITRE 3

RÉSEAUX DE NEURONES ARTIFICIELS

3.1 Introduction

La première partie de cette étude sera consacrée à la description des fondements biologiques des réseaux de neurones. En effet, on ne pourrait parler de ''neurones'' sans resituer l'outil mathématique dans le contexte qui l'a vu naître. Cependant, même si nous n'entreront pas en détail dans les équations biologiques qui seront rappelées, cela ne pourra nuire à la compréhension des modèles des réseaux de neurones artificiels.

Dans ce chapitre, on rappellera le principe de fonctionnement du neurone formel comparativement à son homologue biologique. Après avoir fait un aperçu des différents modèles de réseaux de neurones artificiels connus, nous parlerons des différentes caractéristiques des réseaux de neurones récurrents conçus pour les systèmes dynamiques. En dernier lieu, nous survoleront les différentes techniques d'entraînement ou d'apprentissage de ces réseaux.

3.2 Les fondements biologiques

L'étude des réseaux de neurones artificiels tient son origine à l'examen scrupuleux des processus biologiques qui s'activent dans la génèse, la propagation et la réception des

influx nerveux [AUS95]. On sait que le système nerveux tout entier repose sur le neurone biologique.

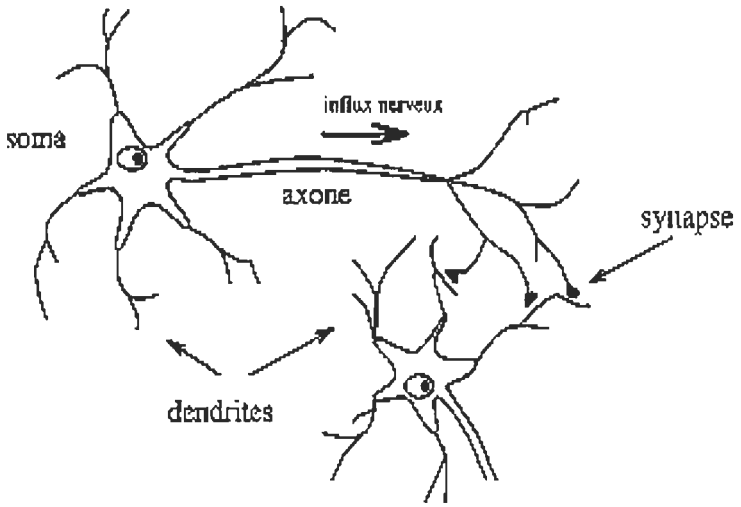


Figure 3.1 : Neurone biologique pyramidale

Il en existe des milliers de types distincts de différentes formes (pyramidale ou sphériques). Un neurone de forme pyramidale est illustré sur la figure 3.1. De façon qualitative, le neurone biologique accomplit une sorte d'intégration spatiale et temporelle des signaux qu'il reçoit à travers ses ramifications appelées *dentrites*. Lorsque le potentiel électrique à proximité de la membrane du corps cellulaire le permet, un potentiel d'action est généré puis propagé à travers l'*axone*. Ce dernier est la fibre nerveuse principale et relie les neurones entre eux. C'est une sorte de modulateur et d'amplificateur du signal transporté. Cependant, certains préceptes biologiques gouvernant la transmission électro-chimique des potentiels d'action doivent être connus lorsqu'on désire modéliser le traitement qu'effectue le neurone sur ces signaux. La perméabilité ionique de la jonction axone-corps cellulaire est

modulée par les signaux chimiques issus des neurones environnant pour donner naissance à un potentiel d'action.

La propagation du signal – Supportée par l'axone, cette propagation est caractérisée par une série d'impulsions ondulatoires appelées potentiels d'action. Leur fréquence est de quelques centaines de Hertz et leur vitesse de propagation d'environ 80 m/s. En caractérisant un neurone par certains paramètres tels que sa conductance ionique, sa capacitance et les dimensions de son axone, l'équation suivante représente le modèle standard de propagation d'une onde électrique dans un câble coaxial. Elle permet de retranscrire fidèlement la propagation du signal dans l'axone.

$$\lambda^2 \frac{\partial^2 V}{\partial x^2} - \mathrm{V} - \tau \frac{\partial \mathrm{V}}{\partial \mathrm{t}} = 0 \qquad (3.1)$$

où x est la longueur de l'axe ; τ la constante temporelle de la membrane ; λ un paramètre adimensionnel et V le potentiel d'action.

Il est nécessaire de connaître les conditions aux frontières de l'axone pour résoudre cette équation. Toutefois, après plusieurs analyses, l'existence d'une non linéarité plus accentuée a été révélée.

La transmission synaptique – De façon fonctionnelle, l'action de la synapse est soit excitatrice, soit inhibitrice. Il agit donc comme un modulateur de conductance. La transmission synaptique a lieu lorsque l'influx nerveux atteint les terminaisons de l'axone. A la différence donc de ce dernier qui est un support passif, la synapse effectue une amplification du signal. L'intégration spatiale et temporelle des signaux – Les signaux issus

des synapses poursuivent leur propagation à travers les dentrites du neurones récepteur. L'intégration spatiale et temporaire des signaux d'entrée est effectuée au niveau des dentrites et du corps cellulaire. La double action des dentrites se traduit comme suit : ils atténuent l'amplitude des signaux d'entrée en les dispersant dans l'espace ; puis ils retardent la propagation de ceux-ci afin de suspendre leur effet dans le temps. Ces signaux convergent par la suite vers le corps cellulaire. Le processus de conduction étant linéaire, l'intégration des signaux s'avère aussi comme étant une sommation linéaire.

La génération du potentiel d'action – L'existence d'un potentiel d'action à la jonction entre le corps cellulaire et l'axone implique que ce dernier a franchi un certain seuil. Puis une courte inhibition du signal s'ensuit. La fréquence de génération des potentiels d'action est proportionnelle au stimulus d'entrée saturant à quelques centaines de Hertz. Elle prend naissance dans l'alternance de la perméabilité ionique qui a lieu au niveau de la membrane du corps cellulaire. Le modèle mathématique de ce processus de perméabilité est régi par un système d'équations fortement non linéaires.

La simulation – Une solution retenue pour mieux représenter le fonctionnement d'un neurone est de le diviser en plusieurs blocs. Ainsi, en représentant indépendamment mais fidèlement chacune des régions distinctes du neurone que nous venons d'évoquer, on évite la difficulté qui réside dans la résolution analytique des équations non linéaires couplées.

3.3 Le neurone formel

La première modélisation mathématique d'un neurone biologique date de 1943 [AUS95] et a été présentée par McCulloch et Pitts (figure 3.2). Ces derniers s'étaient

simplement inspirés de leurs travaux en neurologie. Le principe de fonctionnement du neurone formel (artificiel) devant ressembler à celui de son homologue biologique, ils ont exprimé sa sortie simplement par une fonction non-linéaire de la somme pondérée de ses entrées,

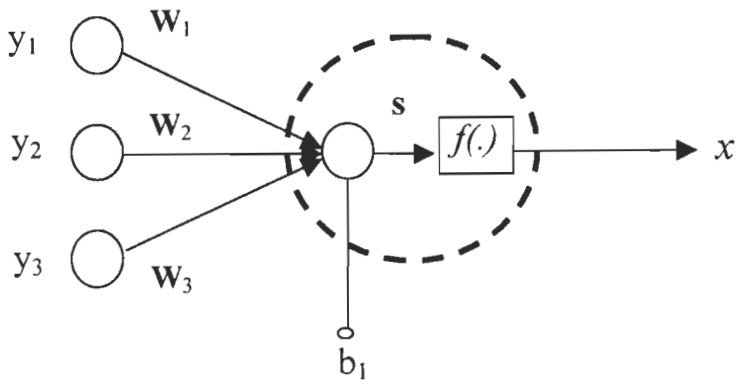


Figure 3.2 : Neurone formel

Le modèle mathématique est exprimé par les équations suivantes :

$$s = \sum_{i=1}^{N} W_i y_i + 1 \tag{3.2}$$

$$x = f(s) \tag{3.3}$$

où y_i sont les entrées du neurone ; W_i les poids synaptiques et b_1 le biais du neurone.

La non linéarité f est la fonction d'activation. Elle fut pendant longtemps représentée par une fonction de seuillage ou de Heaviside du type :

$$f(s) = \begin{cases} +1 \; si \; x > b \\ \\ 0 \quad si \; non \end{cases} \tag{3.4}$$

De cette manière, si la somme pondérée s dépasse un seuil b, sa sortie x est à +1 et le neurone est activé; sinon, la sortie est nulle et le neurone est inactivé. Ce modèle de base de la fonction d'activation a été légèrement modifié car il s'avère préférable d'utiliser une fonction non-linéaire dérivable et bornée sur l'intervalle]- ∞ ,∞[plutôt qu'une fonction linéaire par morceaux. Les fonctions les plus couramment employées sont les fonctions sigmoidale,

$$f(s) = \frac{1}{1+e^{\beta s}} \tag{3.5}$$

et tangente hyperbolique

$$f(s) = \frac{1-e^{-s}}{1+e^{-s}} \tag{3.6}$$

Pour le neurone formel conçu à partir du modèle décrit précédemment et utilisé dans ces travaux, la fonction de décision sera dérivée du type tangente hyperbolique.

On peut en définitive décrire un neurone formel comme étant un minuscule automate de décision [ARR98] (figure 3.3). Il comporte deux parties ayant des fonctions distinctes : une évaluation de la stimulation reçue et une évaluation de son état interne.

Un neurone formel est donc caractérisé par : son état, le niveau d'activation reçue, sa fonction de décision et sa fonction d'entrée (en général une somme).

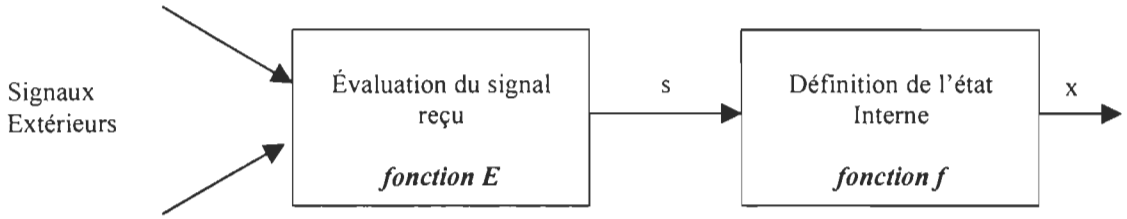


Figure 3.3: Schéma bloc d'un neurone formel

L'état que peut prendre un neurone est généralement représenté de façon binaire, mais peut aussi avoir une valeur discrète ou continue. Le type de cet état déterminera la fonction d'activation à choisir pour les neurones du réseau. Les plus courantes sont illustrées sur la figure 3.4.

Il existe d'autres fonctions pouvant permettre d'obtenir les résultats comparables, le tout dépendant des facilités d'implantation disponibles. En général, on choisit ce genre de fonction parce que leurs ensembles de valeurs permettent facilement de travailler dans l'intervalle compris entre 0 et 1. Ces deux valeurs étant celles qui représenteront clairement l'état d'activation ou d'inhibition d'un neurone.

Connaissant l'avantage d'utiliser ce mode de codage des informations, on conclut qu'une implantation matérielle d'un réseau de neurones devient facilement envisageable.

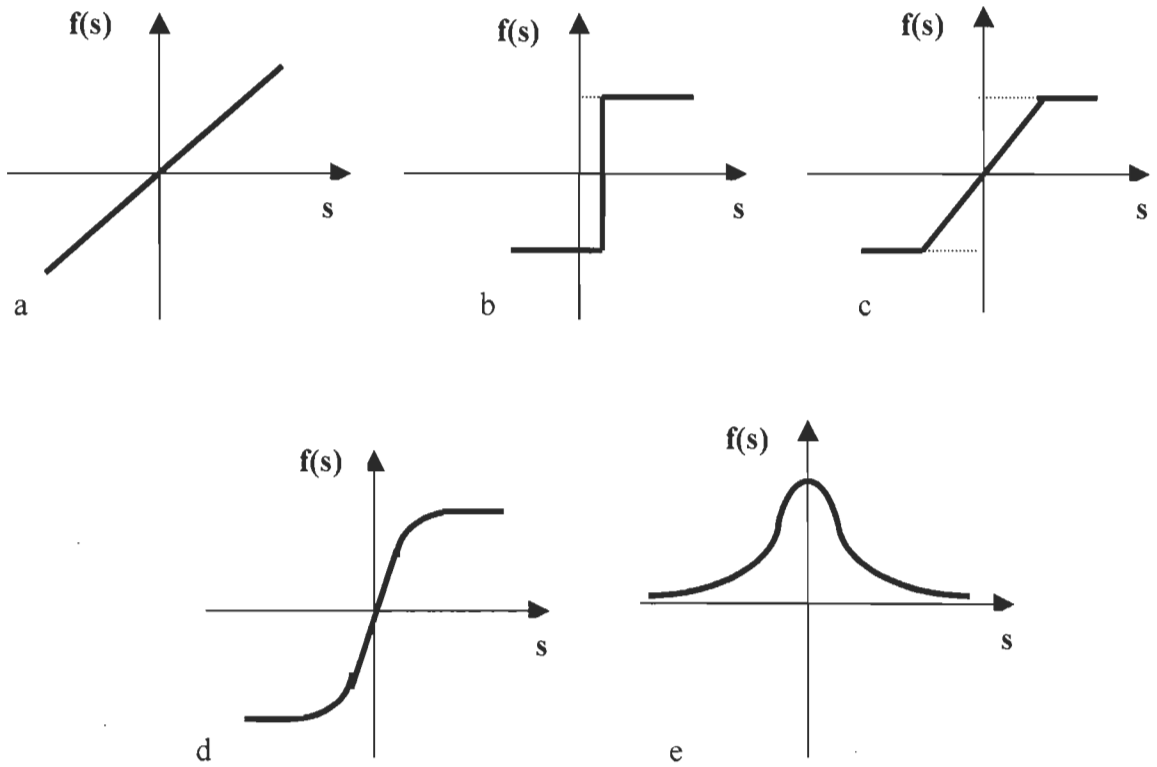


Figure 3.4: Fonctions d'activation courantes a) linéaire, b) heaviside, c) linéaire saturée, d) tangente hyperbolique et e) gaussoidale

3.4 Architectures des réseaux de neurones artificiels

À travers les origines multiples des réseaux de neurones, plusieurs architectures ont vu le jour depuis plusieurs décennies. Ainsi, après les résultats obtenus par McCulloch et Pitts en 1943, il faudra attendre 1958 pour voir la naissance de certaines méthodes analytiques d'adaptation des poids au sein d'un modèle multicouches : ce furent le

perceptron et les réseaux *adalines*. Sources d'inspiration pour bon nombre de chercheurs, on verra la naissance de plusieurs types de réseaux différenciés par leurs algorithmes d'apprentissage [ARR98]. Il s'agit de la *rétro-propagation du gradient*, le *réseau de Hopfield*, le *réseau de Anderson*, *la Machine de Boltzmann*, la *contre-propagation*, les *réseaux de Kohonen* et les *réseaux ART*. Parmi ces différents réseaux, il y en a qui sont mono-couche, multi-couches, et de type récurrent ou statique. Les réseaux mono-couche peuvent être entraînés avec des règles d'apprentissage relativement simple. Mais leur inconvénient majeur est leur limitation au calcul de fonctions assez simples. Cela justifie alors le besoin de créer d'autres réseaux plus élaborés, contenant une ou plusieurs couches cachées. Cependant, même si ces réseaux possèdent des capacités de calcul plus grandes, il demeure que leur entraînement devient difficile. Les réseaux multi-couches comme le perceptron multi-couches sont assez récents. D'ailleurs, ce dernier utilise l'algorithme de rétro-propagation du gradient pour pondérer le réseau.

La rétro-propagation du gradient

Au point de vue architectural (figure 3.5), ce type de réseau multi-couche est caractérisé par une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées. Chaque neurone est relié à tous les neurones de la couche suivante et leur fonction de transfert doit être dérivable en tout point tel que la fonction sigmoide ou tangente hyperbolique.

Utilisé dans le perceptron multi-couches, la plus importante phase du principe de l'algorithme de rétro-propagation réside dans l'entraînement du réseau. Cet apprentissage consiste à présenter au réseau des entrées, puis de modifier sa pondération de façon à

retrouver la sortie correspondante. Ainsi, à l'aide de l'algorithme, on propage d'abord les entrées vers l'avant jusqu'à l'obtention d'une sortie calculée par le réseau.

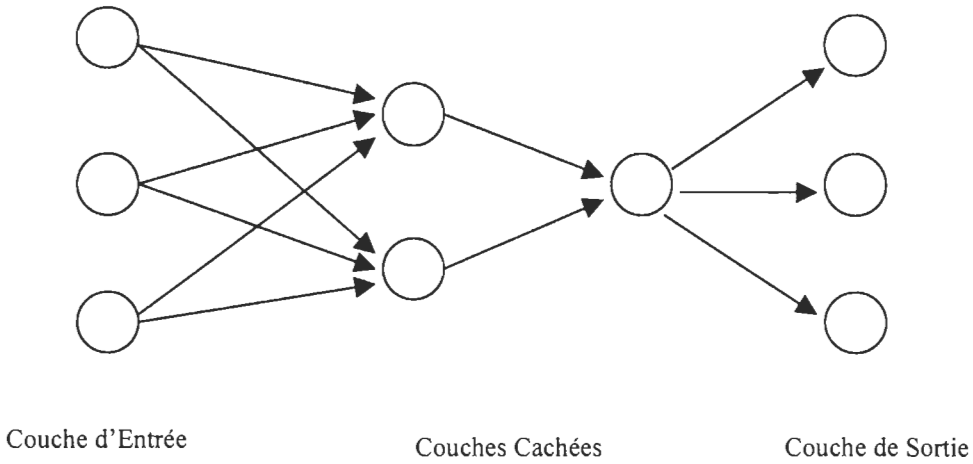


Figure 3.5 : Architecture de base du réseau à rétro-propagation.

En deuxième temps, on effectue une comparaison entre cette sortie et celle qui est réellement connue. On se sert alors de cette erreur pour modifier les poids de connexions de sorte à minimiser l'erreur lors de la prochaine itération. Ayant des couches cachées, on rétro-propage alors l'erreur vers l'arrière en modifiant les poids jusqu'à la couche d'entrée Ce processus est répété jusqu'à l'obtention d'une erreur de sortie jugée acceptable. L'étape de généralisation consiste simplement à une propagation des entrées vers l'avant. Aucune erreur n'est calculée car la sortie réelle reste inconnue.

Le réseau de Hopfield

Présenté en 1982, ce modèle est basé sur le principe des mémoires associatives dont la fonction principale est identique à similaire à celle d'un filtre conçu pour restituer une information en tenant compte de sa perturbation ou du bruit. Les réseaux de Hopfield sont à

rétroaction et à connexions symétriques (Figure 3.6). Les sorties dépendent des entrées et du dernier état pris par le réseau. Les neurones d'un tel réseau sont discrets et répondent à une fonction seuil telle que :

$$f(s) = \begin{cases} +1 & si\ s > 0 \\ \\ -1 & si\ s \leq 0 \end{cases} \tag{3.7}$$

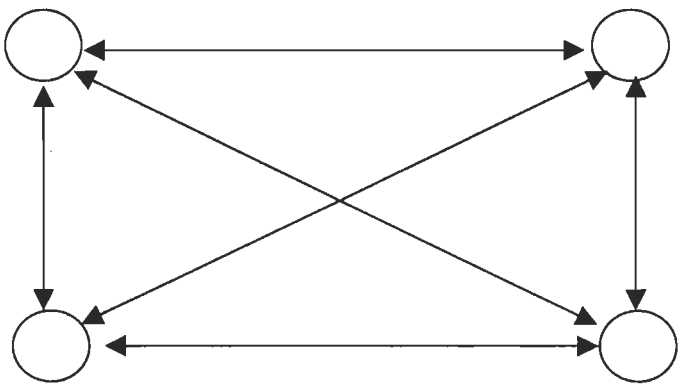


Figure 3.6: Exemple de réseau de Hopfield

Le réseau de Hopfield étant l'un des réseaux les plus simples, son principe démontre un apprentissage facile à mettre en œuvre. Les problèmes les plus importants apparaissent au cours de la phase d'évaluation. La règle d'apprentissage repose sur celle de Hebb qui consiste à forcer les poids de connexions entre les neurones actifs au même moment. Par contre, ces poids seront diminués si les neurones sont inactifs. Pour Hopfield, deux neurones dans l'état –1 sont actifs. La modification des poids se fait donc par un calcul analytique très simple. Lors du processus de généralisation, on présente alors un vecteur

d'entrée au réseau qui calcule la sortie correspondante et la ré-injecte à l'entrée. Le processus sera répété jusqu'à ce que la sortie (entrée de l'étape t+1) soit identique à l'entrée à l'étape t. C'est ce qui traduit la récurrence des réseaux de Hopfield.

Le réseau de Anderson

Ce réseau a été créé dans le but d'améliorer les résultats obtenus avec le réseau de Hopfield. Ayant la même architecture globale, ils sont différents au niveau des règles d'apprentissage.

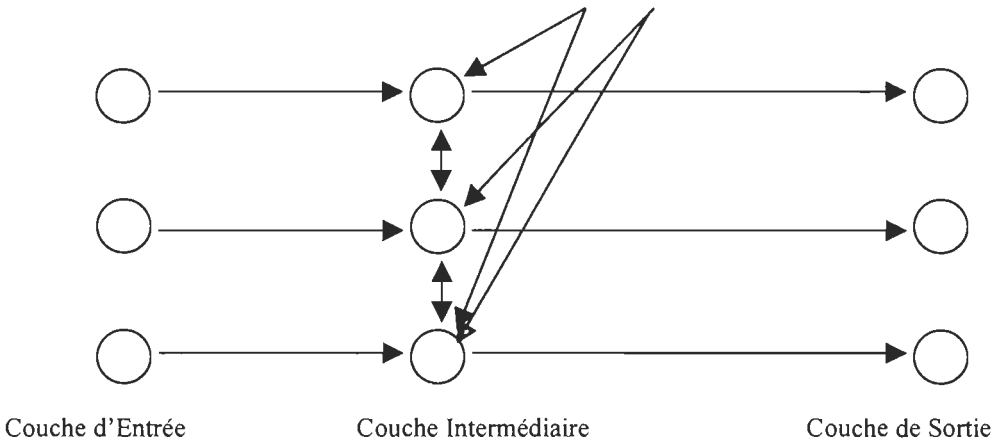


Figure 3.7 : Réseau de Anderson

Le réseau de Anderson (Figure 3.7) est identifié comme un réseau purement déductif. Il a l'avantage de définir directement ses couches d'entrée et de sortie pendant l'apprentissage et la validation. L'architecture du réseau de Anderson montre qu'elle est fortement récurrente au niveau de la couche cachée mais pas sur les autres. Son principe consiste dans une certaine mesure à la modélisation d'un comportement psychologique. Ainsi, après avoir effectuer un certain raisonnement logique, il suffira que l'on apprenne les règles de

base de ce raisonnement au réseau pour avoir les résultats escomptés ; le calcul de la sortie étant basé sur celui de Hopfield mais avec des termes plus complexes.

La machine de Boltzmann

Introduit en 1984 par Ackley, Seijnowski et Hinton, ce réseau est une extension originale de celui de Hopfield.

En effet, étant aussi récurrent, il a la particularité d'inclure une couche cachée. Il est composé de neurones fortement connectés et symétriques (figure 3.8). Mais contrairement au réseau de Hopfield, la fonction d'activation est de type stochastique et se met sous la forme :

$$P[s_i = 1] = \frac{1}{1 + e^{-I/T}} \tag{3.8}$$

C'est la probabilité que le neurone soit activé à 1 ; T étant la température.

Le principe général consiste à dicter au réseaux la probabilité des différents états désirés. La méthode d'apprentissage se résume à une phase contrainte où la probabilités des états désirés est augmentée en fixant les entrées et sorties ; et à une phase libre où cette probabilité est égalisée. C'est au cours de ces deux phases que les poids sont ajustés. La phase d'évaluation du réseau effectue une phase dite de ''recuit libre'' qui permet de fournir des valeurs de sortie. Le succès de cette opération repose sur la bonne définition des paliers de notre raisonnement ou de nos états désirés.

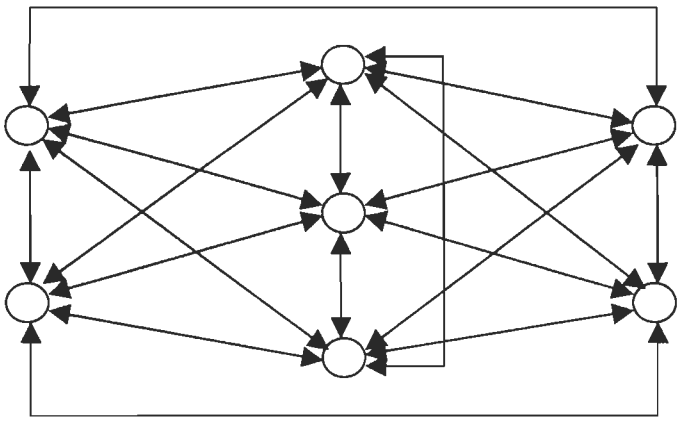


Figure 3.8: Architecture du réseau de Boltzmann

La contre propagation

Inventé par Hecht Nielsen, ce réseau est beaucoup plus destiné à des fins d'introduction aux réseaux de Kohonen. Étant un réseau hybride, il servira surtout à effectuer des opérations de classification et de reconnaissance.

L'architecture de l'algorithme de contre-propagation est proche de celle du perceptron (figure 3.9). Elle possède une couche d'entrée, une couche cachée et une couche de sortie. Ce réseau peut effectuer deux types d'apprentissage en fonction de la couche considérée. Cependant, la règle d'activation d'un neurone est de type compétitive. En effet, on désire déterminer un neurone gagnant qui sera le seul à être activé. L'apprentissage se faisant en mode supervisé, on entraîne en premier lieu la couche cachée dont les poids ont auparavant été choisis aléatoirement.

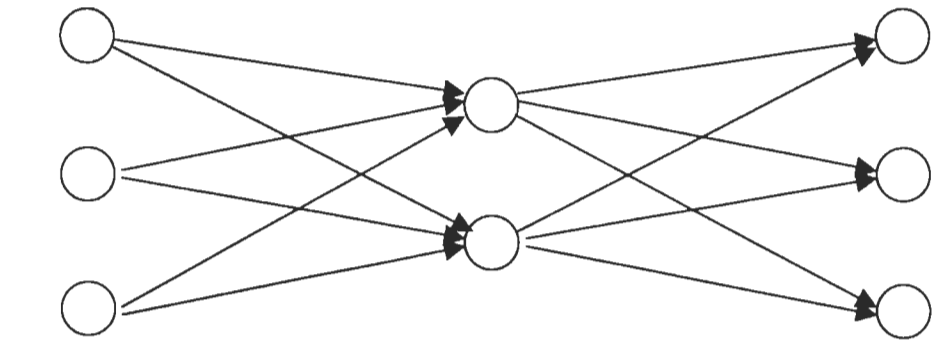


Figure 3.9: Réseau à contre-propagation

Une normalisation de l'entrée est nécessaire et l'activation de chaque neurone caché est calculée. Seuls les poids de connexion du neurone ayant la plus forte activation seront modifiés. La phase de généralisation consiste à une propagation de l'entrée vers la couche cachée afin de déterminer le neurone gagnant ; puis on propage vers la sortie.

Les réseaux de Kohonen

Ces réseaux sont de type auto-organisateurs car ils fonctionnent en grande majorité dans un mode non supervisé. Aussi, ce sont des réseaux dynamiques dans la mesure où ils ont la tendance à créer ou à éliminer des neurones. Une autre particularité de ce réseau est la compétition entre les neurones d'une même couche.

Rappelons que pour qu'un réseau soit compétitif (Figure 3.10), il faut que les neurones de la couche compétitive réagissent différemment aux entrées afin d'élire un vainqueur. On peut aussi établir un mécanisme de concurrence entre les neurones afin de choisir le plus activé et modifier ses poids de connexion en conséquence.

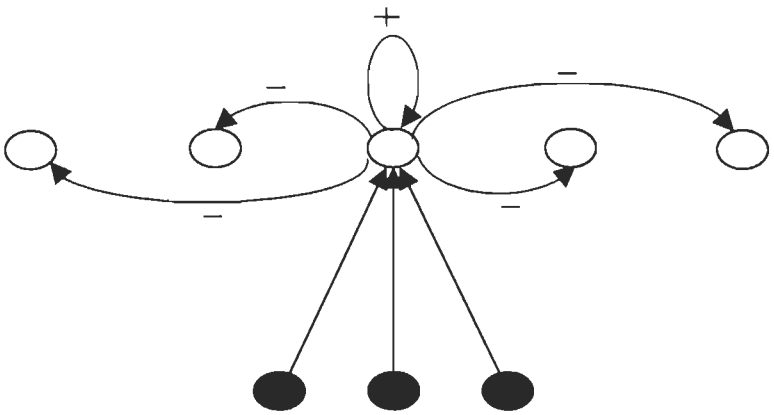


Figure 3.10: Architecture compétitive de Kohonen

L'apprentissage d'un tel réseau vise donc à favoriser le neurone vainqueur d'une couche en le rapprochant du vecteur d'entrée correspondant. Mais l'inconvénient est que pour une entrée de validation éloignée, ce neurone est pénalisé.

3.5 Réseaux de neurones dynamiques récurrents

3.5.1 Modèle général

Dans la recherche connexioniste, la quête d'un modèle neuronal qui soit à la fois versatile et dont l'entraînement requiert un effort de calcul raisonnable demeure un enjeu primordial. Dans cet esprit, il existe une grande classe de réseaux de neurones récurrents et dynamiques introduite pour le traitement temporel. Nous allons présenter ici les réseaux dédiés au traitement temporel et dont le modèle mathématique du neurone est basé sur une fonction de transfert sigmoidale.

L'architecture d'un modèle de réseau de neurones dynamique et récurrent est plus générale car les connexions sont arbitrairement retardées et bouclées.

Pour comprendre le fonctionnement des réseaux de neurones artificiels dynamiques, on devra d'abords se rappeler quelques notions importantes sur les systèmes dynamiques.

D'une manière générale, la description de tels systèmes se fait à partir d'un modèle mathématique et le plus courant est le *modèle d'état*. Dans ce modèle, on définit certains paramètres comme suit :

- *variables d'états* : ce sont les variables dont les valeurs à un instant donné contiennent une information suffisante pour effectuer la prédiction du comportement futur du système.

- *vecteur d'état* : en notant $x_1(t)$, $x_2(t)$, ..., $x_N(t)$ comme les variables d'état d'un système dynamique, t étant la variable indépendante et N l'ordre du système, ces variables sont contenues dans un vecteur $\mathbf{x}(t)$ appelé vecteur d'état du système.

- *dynamique du système* : soit F une fonction non linéaire, la dynamique d'un système dynamique non linéaire est souvent représenté par une équation différentielle de premier ordre telle que :

$$\frac{d}{dt}x(t) = F\left[x(t)\right] \tag{3.9}$$

Lorsque le vecteur F(x(t)) ne dépend pas du temps t, le système est dit autonome. Dans le cas contraire, on a un système non autonome. Si le vecteur x(t) est invariable dans le temps, le système n'est pas dynamique. On peut en définitive définir un système dynamique comme étant *un système dont l'état varie avec le temps*. En appliquant cette définition aux

réseaux de neurones récurrents dynamiques, on dira que ces derniers le seront si et seulement si leur état varie dans le temps.

Ainsi, dans les réseaux dits *bouclés*, le vecteur d'état du système est obtenu à l'issu d'une phase dite de *relaxation* vers un état d'équilibre stable [AUS94]. L'attraction majeure de ce type de réseau est sa mémoire adaptable lui permettant d'implémenter directement un système dynamique. Les interactions mutuelles retardées entre les différents neurones permettent de conserver une mémoire sous forme d'informations codées de la séquence présentée à l'entrée.

Compte tenu de la récursion temporelle et spatiale, le comportement du modèle est décrit par un système d'équations différentielles dont les variables internes sont les activations des neurones de la couche cachée. Une fois donc ce modèle défini, la prochaine étape est la mise au point d'une procédure d'adaptation des paramètres internes. C'est à cette fin que deux algorithmes récursifs de rétro-propagation ont été développés. Ils sont connus sous le nom de *propagation en chaînage avant* et *propagation en chaînage arrière* [AUS94]. La principale différence entre ces deux algorithmes est que le premier révèle une décroissance exponentielle du gradient d'erreur et une convergence. Par contre, la propagation en chaînage arrière montre une facilité de mise en oeuvre grâce à sa faible complexité d'implantation. Certains réseaux ont alors vu le jour pour ces algorithmes, et parmi eux on a les modèles *FIR*, *NARMA*, *Time Delay Neural Network*, et le *Autoregressive Network* [AUS94].

Dans les réseaux non bouclés, on sait que l'ajustement des poids se fait grâce à la propagation directe du gradient d'erreur dans le sens contraire des signaux, en remontant le réseau vers l'amont. Ce genre de rétro-propagation standard n'est pas applicable à un

réseau récurrent dynamique car il contient des connexions bouclées et retardées. D'où la nécessité d'une rétro-propagation dans le temps. En effet, le réseau disposera d'une mémoire lui permettant de remonter dans le passé afin de décéler l'origine de l'erreur courante. Cette aptitude est possible grâce à la formulation des équations différentielles décrivant le modèle du système [AUS95]. Ainsi, en supposant que l'entrée d'un tel réseau contient deux composantes : l'entrée externe y_n et m_y versions retardées de ce vecteur y_{n-1}, y_{n-2}, ... , y_{k-my} d'une part, et m_x vecteurs d'états retardés du système x_{n-1}, x_{n-2}, ..., x_{n-mx} d'autre part. En transposant la mémoire interne du réseau à l'extérieur comme une entrée supplémentaire, on peut exprimer la loi gouvernant l'évolution du système par :

$$x_n = f\left(y_n,\ldots,y_{n-m},x_{n-1},\ldots,x_{n-m},W_n^0,\ldots,W_n^m\right) \tag{3.10}$$

où la fonction f ne contient aucun délai et peut être une sigmoide et $m=m_y=m_x$. La représentation de ce système est visualisée sur la figure 3.11.

Les principales caractéristiques des réseaux de neurones artificiels dynamiques sont leurs performances en matières de stockage d'informations, de traitement des données et de reconnaissance. Ces performances tiennent leur origine d'abord dans l'aspect numérique (binaire) des signaux utilisés par les neurones pour communiquer entre eux ; puis, à leur travail en parallèle et à l'importante quantité d'informations qu'ils peuvent traiter pour exécuter une tâche précise.

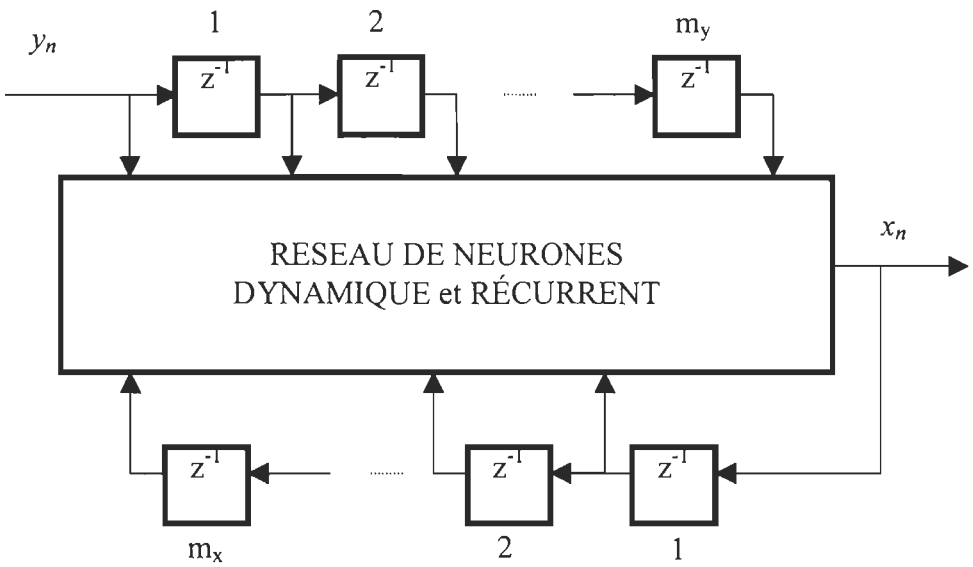


Figure 3.11: Représentation d'un réseau de neurones récurrent dynamique

Mais la caractéristique la plus importante des réseaux dynamiques récurrents qu'on ne manquerait d'indiquer est leur capacité de généralisation. En effet, après lui avoir appris adéquatement un ensemble de données représentant une forme quelconque d'un signal bien connu, le réseau de neurones montre des aptitudes à retrouver un estimé de n'importe quel autre signal de même type que celui présenté lors de l'apprentissage.

Cependant, il faut noter que la qualité de reconstitution du réseau de neurones repose en grande partie sur la pertinence des informations contenus dans les données utilisées lors de son étalonnage. En effet, pour un système de mesure quelconque, ces données doivent être de bonne qualité car elles devront traduire et contenir le maximum de renseignements sur le système de conversion contenant l'élément sensible qui permet de traduire le phénomène physique mesuré en signaux électriques. Ainsi, une fois ces données obtenues, leur

apprentissage au réseau de neurone permettra de donner la meilleure estimation de n'importe qu'elle autre donnée issu du bloc de conversion.

En résumé, les aptitudes des réseaux de neurones dynamiques à pouvoir interpoler viennent de leurs principales caractéristiques, à savoir : un travail en parallèle, une grande capacité de mémorisation et de généralisation.

3.5.2 Modèle NNOE

Le modèle NNOE (*Neural Network Output Error*) est conçu pour l'identification des systèmes dynamiques non linéaires. Il se base sur le calcul de l'erreur de sortie. Illustrée sur la figure 3.12, cette architecture [NOR95] montre qu'on a un réseau de neurones à trois couches dont une couche cachée.

Ayant un seul neurone de sortie, le nombre de neurones d'entrée est fonction du nombre d'informations prélevées sur l'entrée $\tilde{y}_n$ et la sortie estimée $\hat{x}_n$.

Les neurones de la couche cachée, dont le nombre est fixée durant la procédure détalonnage, ont une fonction de décision de type tangente hyperbolique (figure 3.13) tandis que ceux des autres couches sont en général linéaires. Cette architecture est semblable à celle du modèle NARX [HAY98] à la seule différence que cette dernière requière une connaissance à priori du signal recherché.

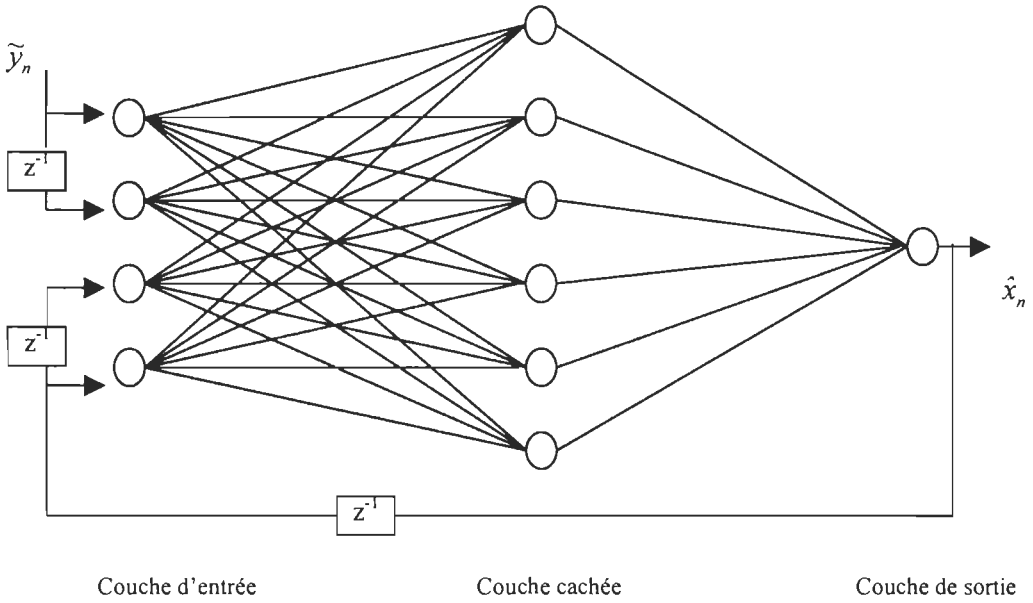


Figure 3.12 : Architecture du modèle NNOE (1 retard sur l'entrée et 2 sur la sortie)

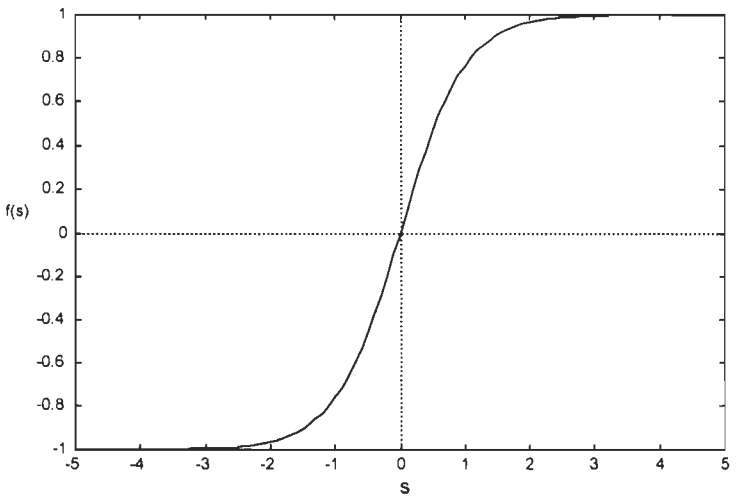


Figure 3.13 : Fonction d'activation selon léquation (3.6) des neurones cachés du modèle NNOE.

3.5.3 Modèle TDNN

Introduits pour la première fois par Alex Waibel [AUS94], les réseaux TDNN (*Time Delay Neural Network*) sont un groupe de réseaux ayant une topologie spéciale. Ils sont beaucoup utilisés pour la reconnaissance indépendante de composantes dans un espace assez large. De façon générale, l'activation d'un neurone se fait en passant la somme pondérée de ses entrées dans une fonction de décision. Pour les neurones d'un réseau TDNN, ce comportement est remplacée par l'introduction des délais. Ainsi, chaque entrée d'un neurone d'une couche donnée est multipliée par un nombre M de délais définis pour cette couche.

Le processus d'apprentissage de ce type de réseaux s'effectue à l'aide d'un algorithme semblable à celui de la rétro-propagation, mais en tenant compte des différentes connexions entre les couches. Pour que le réseau fournisse les résultats escomptés, il est nécessaire de lui présenter à l'entrée plusieurs cartes sur lesquelles l'élément à reconnaître est décalé différemment.

Dans l'architecture de ce réseau TDNN (figure 3.14) [CHE96], la couche d'entrée de huit neurones prélève huit échantillons successifs de données provenant d'un signal dynamique **y**; cet ensemble d'échantillons forme le vecteur d'entrée du réseau. Puis, à travers des poids fixes, ces données sont propagées vers une unité de cinq neurones de la première couche cachée. Les états de ces cinq neurones sont décalés dans cette couche cachée au fur et à mesure que les données se présentent à la couche d'entrée.

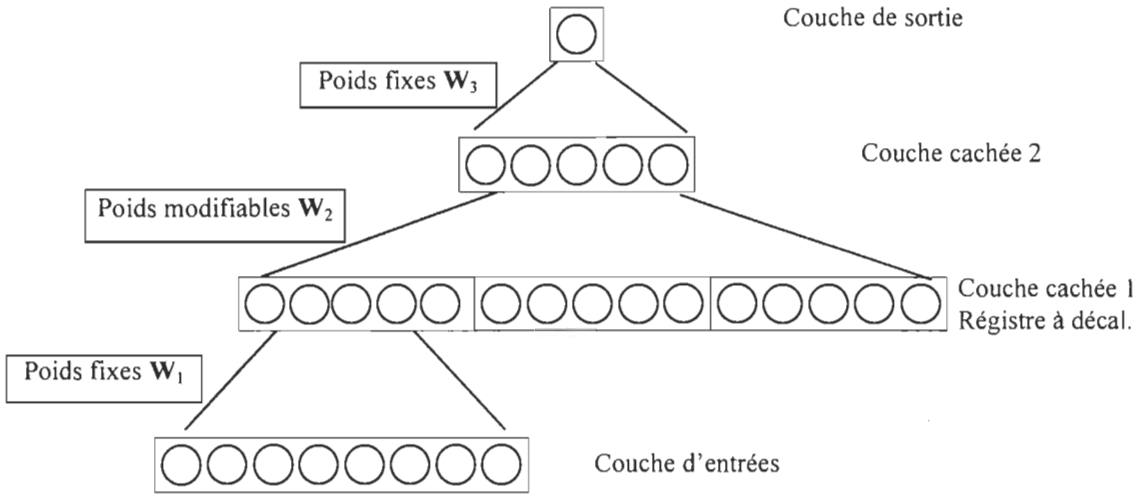


Figure 3.14 : Architecture d'un réseau TDNN

La première couche cachée agit donc comme un registre à décalage. Les trois ensembles de cinq neurones sur cette couche représentent une version compressée d'une série temporelle de 120 données. Ensuite, les sorties de tous les neurones de cette première couche cachée sont propagées à l'entrée d'une deuxième couche cachée de cinq neurones à travers des poids de pondération modifiables. Enfin, un neurone de sortie effectue la classification ou la reconnaissance à partir des données issues de cette deuxième couche de cachée.

Le principe de l'apprentissage consiste cependant à propager directement les entrées, puis de calculer l'erreur à la sortie du réseau. Ensuite, les dérivées de cette erreur sont calculées et rétro-propagées vers la couche d'entrée. Ceci a pour résultats d'obtenir différentes valeurs pour la correction des connexions impliquées. Une fois ces valeurs fixés, les poids de pondération sont réajustés ou mis à jour. La fonction d'activation des neurones d'un réseau TDNN est similaire à une fonction d'activation logistique régulière.

D'un autre coté, il est à noter que les réseaux TDNN appartiennent à la catégorie des réseaux ayant une architecture dédiée au traitement temporel. Un tel réseau étant multicouche récurrent, les neurones de la couche cachée et ceux de la couche de sortie sont répliquées en fonction du temps.

Un outil dédié à la simulation des réseaux TDNN a été élaboré dans le logiciel SNNS^{TM} (*Stuttgart Neural Network Simulator*). Ce type de réseau serait très intéressant dans l'application à la reconstitution de signaux. Mais, comme nous l'indiquerons à la section 3.7, le présent mémoire a pour but l'étude du réseau NNOE.

3.6 Méthodes d'apprentissage des réseaux de neurones

Comme annoncé dans le paragraphe précédent, la qualité des résultats de reconstitution obtenus d'un réseau de neurones repose en grande partie sur son apprentissage. Cette opération s'avère donc assez délicate et nécessite une connaissance approfondie de la tâche à exécuter par le réseau. Ainsi, en fonction du type de problème suggéré au réseau et de sa configuration, on trouve plusieurs techniques d'apprentissage Parmi ces techniques [LEG94], on retrouve celles dont l'apprentissage est supervisé ou non; avec prototypes; par cœur; et hors ligne ou en ligne.

Un apprentissage est dit supervisé lorsqu'un superviseur présente des ensembles entrée - sortie au réseau tout en définissant la règle d'apprentissage à partir de la nature du problème à résoudre. L'apprentissage avec prototypes concerne des réseaux de neurones dont l'algorithme est itératif et dont les valeurs des poids sont ajustées pendant ce processus. Dans l'apprentissage par cœur, on vise à obtenir une mémorisation définitive de

l'association des données. Enfin, lorsque le réseau de neurones subit d'abord un entraînement visant à l'étalonner avant de l'insérer dans un système de mesure, on parle d'apprentissage hors-ligne. Si ce dernier ajuste ses poids de connexion en même temps qu'il réalise sa fonction de reconstitution, il s'agit alors d'un apprentissage en ligne.

Dans le cadre de ces travaux, on choisira la technique d'apprentissage supervisé, avec prototypes et hors-ligne. Elle consistera donc à présenter au réseau de neurones des associations de données d'entrée et de sortie désirées. Il faut rappeler que les données de sortie du processus de conversion sont les entrées du réseau de neurones ; les données de sortie désirées étant simplement celles que devrait fournir le réseau.

Ce mode d'apprentissage sera effectué jusqu'à l'atteinte des performances recherchées. L'un des critères de performance est le calcul de l'erreur de reconstitution dont la valeur minimale déterminera la convergence du réseau.

3.7 Algorithmes de reconstitution par les réseaux de neurones

Les performances des réseaux de neurones artificiels dans le processus de reconstitution des signaux reposent en grande partie sur leur capacité d'identification de l'inverse du système de conversion. Étant donné que l'on veut appliquer cette méthode sur un système de mesure dynamique non linéaire, ce chapitre consiste à évoquer les types de réseaux dédiés à l'accomplissement de cette tâche. Ainsi, dans un premier temps, on décrira le fonctionnement du réseau de neurones que nous appliquerons à la reconstitution des signaux pour un système dynamique linéaire et non linéaire. Ensuite, après avoir

rappelé la procédure générale d'identification des systèmes et les différents modèles disponibles, nous examinerons en détail l'architecture et le fonctionnement de l'algorithme du modèle NNOE (*Neural Network Oputput Error*). Finalement, nous évaluerons l'influence de la qualité des données de référence sur les capacités de généralisation du modèle de reconstitution proposé, basé sur les réseaux de neurones.

3.7.1 Procédure d'étalonnage et de reconstitution

Les réseaux de neurones ayant été choisis afin de contourner certaines difficultés inhérentes aux méthodes analytiques (voir sections 2.5 et 4.7), ceux-ci effectuent une identification inverse du système délivrant les signaux à reconstituer.

Ainsi, la procédure générale à suivre pour identifier un système comporte quatre grandes étapes [NOR95]. Cette procédure est présenté à la figure 3.15. La première étape consiste à constituer une banque de données qui seront utilisées pour étalonner notre modèle de reconstitution. Ces données peuvent être cueillies en milieu expérimental ou générées de façon synthétique. Puis, en deuxième étape, on choisit le modèle ou la méthode de reconstitution en fonction des exigences du problème à résoudre. Parmi ces modèles, on peut citer ceux basés sur le filtre de Kalman, les méthodes de Jansson et de Van Cittert, et ceux basés sur les réseaux de neurones artificiels. Puis, une fois le modèle choisi, on procède à son étalonnage et à son optimisation. Cette troisième étape consiste à l'ajustement des paramètres régissant le modèle de reconstitution choisi à l'aide de la banque de données constituée dès le départ. Ainsi, les paramètres des équations mathématiques des modèles cités plus haut sont par exemple le gain, le nombre d'itérations, les poids de pondérations pour les réseaux de neurones, la dimension du modèle, etc.

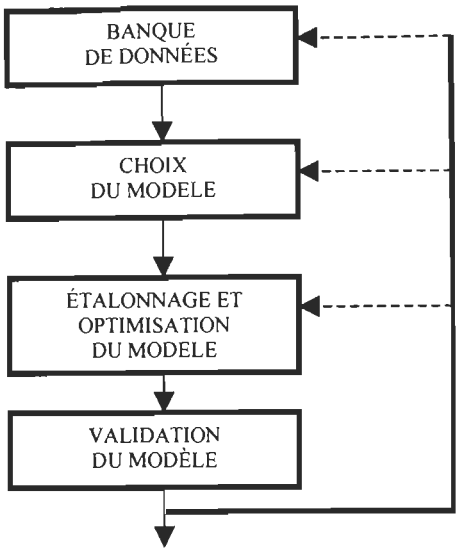


Figure 3.15: Procédure d'identification des systèmes

Enfin, une fois ces ajustements faits, on procède à la validation du modèle; cette étape consiste à utiliser des données différentes de celles de l'étalonnage pour vérifier les performances de la méthode choisie. Ces données de validation ont été établies à la première étape lors de la génération de la banque de données.

3.7.2 Génération de la banque de données de références

Les données de références qui constituent notre banque de données sont réparties en trois groupes: les données d'étalonnage $D^{étal}$, les données de test D^{test} et les données de validation D^{val}. Pour notre étude, les principales composantes de ces différentes données sont telles que:

$$D^{étal} = \left\{ \tilde{x}_n^{étal}, \tilde{y}_n^{étal} \right. \quad \text{pour n=1, 2, ..., } N^{étal} \tag{3.11}$$

$$D^{test} = \left\{ \dot{x}_n^{test}, \tilde{y}_n^{test} \right. \quad \text{pour n=1, 2, ..., } N^{test} \tag{3.12}$$

$$D^{val} = \left\{ \dot{x}_n^{val}, \tilde{y}_n^{val} \right. \quad \text{pour n=1, 2, ..., } N^{val} \tag{3.13}$$

Les données d'étalonnage seront utilisées pour ajuster les paramètres définissant notre modèle de reconstitution. Cet ajustement est fait pendant une procédure appelée entraînement ou apprentissage. Les données de test nous permettent de voir les limites de notre réseau et de pouvoir l'optimiser. Enfin, les données de validation seront utilisées pour vérifier les performances du réseau optimisé précédemment.

Toutes ces données sont générées en observant les sorties $\tilde{y}$ du système de conversion en fonction des entrées connues $\dot{x}$. Cette étape est essentielle car les performances d'un modèle sont reliées à la qualité de son étalonnage qui, à son tour, repose directement sur la qualité des données de référence.

En effet, on sait que l'un des avantages majeurs des réseaux de neurones est leur capacité de généralisation. Cette habilité n'est atteinte que si la phase d'apprentissage a été au préalable bien effectuée avec une base de données de références de bonne qualité. En effet, lors de la validation d'un réseau, des données non apprises lui sont présentées à l'entrée. Cependant, il faudra veiller à ce que ces données de test soient de même type que les données d'apprentissage. Par exemple, un réseau entraîné avec des données de type spectrométriques ne sauraient reconstituer les données de type binaires.

L'élaboration des ces banques de données est une opération assez ardue dans la mesure où certaines conditions expérimentales doivent être respectées. C'est ainsi qu'en milieu de laboratoire par exemple, les paramètres du milieu dans lequel l'appareil de mesure va

opérer doivent être établies. Parmi ces paramètres, on peut citer en exemple la température, la pression, les vibrations et l'acidité du milieu environnant. Aussi, il faudra tenir compte du facteur de vieillissement du capteur. En effet, on devra s'assurer que cet instrument associe une valeur à chaque étalon peu importe l'instant de la mesure, au sens d'une constante de temps beaucoup plus grande que la dynamique du système. Autrement dit, nous considérerons le système de conversion comme étant invariant mais pouvant être stationnaire ou non stationnaire. À partir du moment où deux étalons différents conduisent à une même valeur, on change alors la résolution du système pour reprendre les mesures. Les données de références seront alors les étalons (signaux idéaux) et les signaux fournis par le système de mesure (signaux bruités).

Les données d'apprentissage doivent donc être assez représentatives pour le type de signaux étudiés. De cette manière, l'essentiel de l'information est retenue par le réseau pour une meilleure pondération de ses poids de connexion.

3.7.3 Modèle du réseau de neurones

Cette étape consiste à choisir un ensemble de régresseurs dépendamment du système dynamique étudié. La grande difficulté est rencontrée en ce qui concerne les systèmes dynamiques non linéaires. Et de plus, il faudra disposer d'une architecture de réseau adéquate. Celle qui sera étudiée dans ces travaux est l'architecture NNOE (*Neural Network Output Error*). Cette architecture identifie un modèle de réseau basé sur l'erreur de sortie (Output Error). Son vecteur de régression $\varphi(n)$ et son prédicteur $\hat{x}(n|\theta)$ s'expriment par le système d'équation suivant :

$$\varphi(n) = \left[\hat{x}(n-1/\theta), \ldots \hat{x}(n-m_x/\theta), y(n-m_k), \ldots, y(n-m_y-m_k+1) \right]^T \quad (3.14)$$

$$\hat{x}(n/\theta) = G\left[\varphi(n), \theta \right] \quad (3.15)$$

avec θ la matrice contenant les poids de connexion, $\hat{x}(n)$ la sortie estimée du réseau de neurones et $\tilde{y}(n)$ l'entrée du réseau, m_x et m_y les nombres de retards prélevées sur $\hat{x}(n)$ et $\tilde{y}(n)$ respectivement ; m_k le nombre de retards global sur l'entrée. Dans l'étude effectuée dans la suite, nous considérons $m_k = 0$.

3.7.4 Étalonnage du modèle

L'algorithme d'étalonnage utilisé par le modèle NNOE est basé sur l'ajustement des poids de pondération à l'aide de l'erreur de sortie (Output Error). Dans son fonctionnement général (Figure 3.16), ce principe consiste à :

- propager les données de sortie de conversion du système dans le réseau ;
- calculer l'erreur de sortie entre la valeur désirée connue $x^{étal}$ et celle qui est estimée par le réseau de neurones $\hat{x}^{étal}$;
- rétropropager cette erreur pour réajuster les valeurs des poids de connexion entre les différentes couches du réseau ;
- injecter à l'entrée la ou les valeurs estimées et repartir pour une autre itération.

Cette procédure est répétée pendant un certain nombre d'itérations. Ce nombre étant fixé de façon arbitraire par l'opérateur, il devrait correspondre à celui qui permet d'obtenir une

validation acceptable sur la banque de données de test, selon différents critères dont certains seront énoncés à la section 4.3.

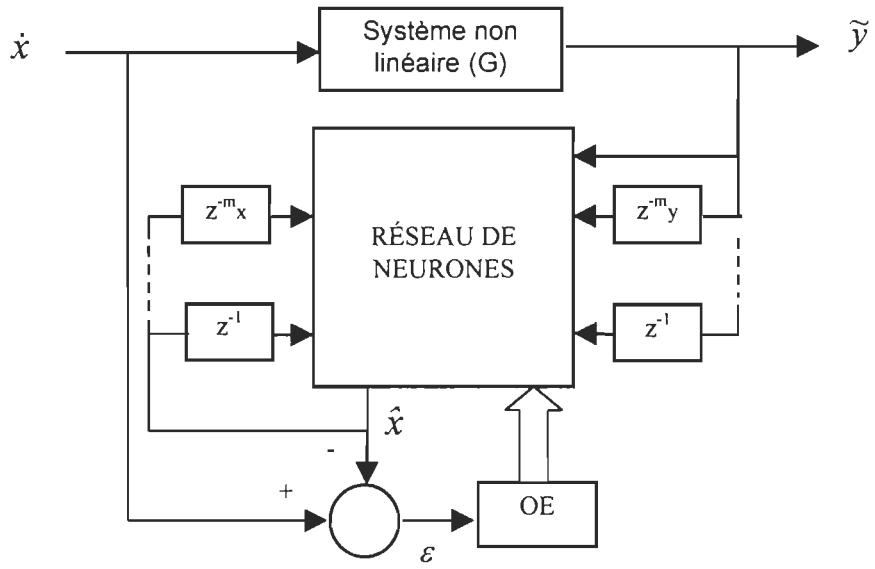


Figure 3.16: Principe d'étalonnage de l'algorithme NNOE

Le nombre de données d'entrée bruitées et de valeurs estimées injectées à l'entrée du réseau dépend du nombre de retards m_x et m_y fixés sur chaque signal par l'opérateur, et de ce fait, le nombre de neurones sur la couche cachée. Ceci a pour effet de modifier automatiquement le nombre de neurones sur la couche d'entrée du réseau. Ainsi, en se servant du système d'équations (3.14) et (3.15), l'équation de sortie du réseau est telle que :

$$\hat{x}(n/\theta) = R\left[\ \hat{x}(n-1/\theta), \ldots \hat{x}(n-m_x/\theta),\ y(n-m_k), \ldots,\ y(n-m_y-m_k+1),\ \theta\ \right] \quad (3.16)$$

Comme on le voit, la sortie estimée du réseau est directement liée aux sorties précédentes et aux entrées du réseau. La caractéristique importante de ce modèle est qu'il assure une bonne convergence du réseau lors de la phase d'apprentissage. Aussi, cet algorithme nous garantit de meilleurs résultats de reconstitution dans ce sens qu'il ne prend à son entrée que les valeurs du signal converti du système de mesure et celles de la sortie du réseau; les valeurs désirées n'étant utilisées que pour le calcul de l'erreur de sortie. Cette particularité facilite le processus de test de notre réseau car il nous est plus facile d'obtenir des signaux issus d'un système de conversion que ceux réellement cherchés.

3.7.5 Reconstitution et validation du modèle

En vue de valider notre modèle, la procédure générale de reconstitution est effectuée avec des données autres que celles utilisées lors de l'entraînement du réseau de neurones. Ainsi, la reconstitution d'un signal par le réseau consiste à ce qu'il nous fournisse un signal estimé à partir d'un signal bruité de conversion présenté à son entrée (figure 3.17). Il y a donc une simple propagation de cette entrée dans le réseau et seule la qualité du signal de reconstitution (sortie estimée $\hat{x}$) nous renseigne sur son étalonnage et ses capacités d'interpolation. L'équation régissant cette étape est l'équation (3.16) où les matrices des poids ont été obtenues lors de l'étalonnage du réseau.

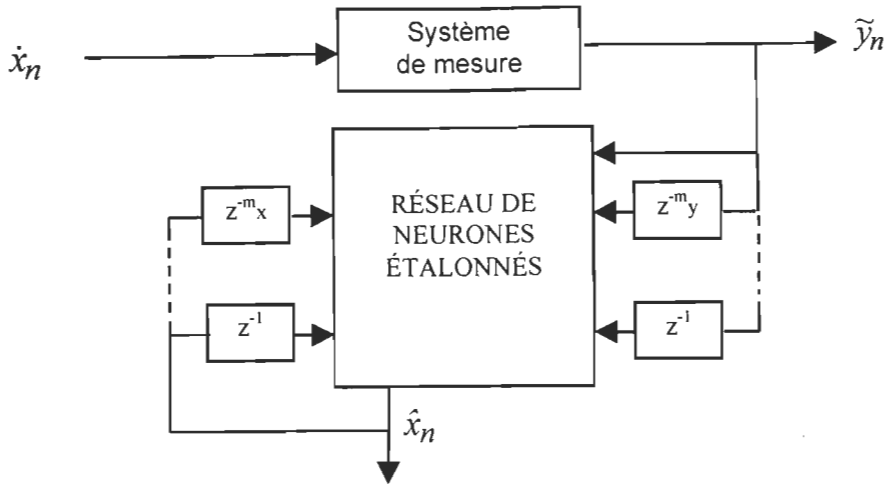


Figure 3.17 : Procédure de validation d'un réseau NNOE

Cependant, trois critères ont été élaborés pour pouvoir optimiser les réseaux de neurones satisfaisants. Le premier consistera à regarder, pour un nombre d'itérations fixes, l'erreur d'apprentissage du réseau. Celle-ci devrait être inférieure ou égale à 10%. Puis, le deuxième critère sera l'observation de la plus petite erreur de reconstitution du signal D^{test} non bruité ($\sigma_n^2 = 0$) pendant l'entraînement. Enfin, un troisième critère sera sur l'erreur de reconstitution minimale de D^{test} affecté d'un bruit de 45dB ($\sigma_n^2 = 10^{-4}$). Ces critères seront détaillés à la section **4. ?**.

Un modèle basé sur la structure NNSSIF (*Neural Network State Space Innovations Form*) a aussi été testé dans ces travaux mais il s'est avéré inadéquat au problème à résoudre.

En effet, sa procédure de validation nécessite une connaissance à priori des données désirées. Il convient alors aux opérations de contrôle et non de la reconstitution des signaux dont les références sont inconnues.

3.8 Conclusion

Dans ce chapitre, nous avons survolé les origines biologiques ayant servi de support quant à la modélisation mathématique des neurones artificiels. Après avoir énuméré les principales caractéristiques d'un neurone formel, quelques architectures de réseaux ont été expliquées. En se basant sur le fonctionnement du cerveau humain et conformément à la tache à exécuter, ces architectures possèdent chacune un algorithme spécifique. Ensuite, ayant évoqué les points forts des réseaux de neurones dynamiques, le mode d'entraînement supervisé a été choisi pour notre étude.

Il existe une diversité d'algorithmes utilisés dans la prédiction des systèmes dynamiques non linéaires. Mais compte tenu du fait que notre problématique concerne la mesure des grandeurs analogiques dont on ignore les valeurs réelles, le modèle NNOE a été choisi. En effet, cela se justifie par sa capacité d'apprentissage et son principe de généralisation. Comme on l'a vu au paragraphe 3.7.4, ce dernier consiste à ne fournir au réseau que les données issues du bloc de conversion d'un système de mesure ; données qui nous sont facilement accessibles. Pour prétendre à une bonne généralisation du réseau, il faudra avoir une bonne idée des signaux recherchés. D'où l'importance de la qualité des données d'apprentissage utilisées pour l'étalonnage du réseau de neurones. Compte tenu de la difficulté à obtenir des données expérimentales assez nombreuses, des données synthétiques seront générées afin d'amorcer notre étude.

CHAPITRE 4

APPLICATION À UN SYSTÈME DE MESURE SPECTROMÉTRIQUE

4.1 Introduction

Afin de démontrer les capacités des réseaux de neurones artificiels multicouches à effectuer le processus de reconstitution des signaux dans un système de mesure, nous avons décidé de les appliquer à un système de type spectrométrique. D'autres types pouvaient être choisis mais le choix de ce dernier se justifie par les domaines très variés d'applications des spectromètres. Ainsi, dans ce chapitre, nous allons en premier lieu définir le système de mesure en générant des banques de données synthétiques servant à l'étude de la méthode; et en second lieu, l'entraînement proprement dit des réseaux de neurones. L'effet de certains paramètres du modèle est étudié et nous permettra d'effectuer une optimisation selon certains critères et de sélectionner les réseaux présentant les meilleures performances. Puis, tenant compte de la robustesse des réseaux choisis, une étude comparative est effectuée avec les résultats issus d'autres méthodes populaires de reconstitution des signaux. Ces étapes seront respectivement suivies pour les systèmes dynamiques de mesure linéaires, puis non linéaires.

4.2 Génération des données synthétiques

Comme défini dans le chapitre 2, on sait que la précision des mesures issus du bloc de conversion d'un système de mesure (figure 2.1) est affectée par plusieurs facteurs d'origines internes et externes. En effet, le mesurande $\dot{x}$ capturé par ce bloc est transformé en un signal brut $\tilde{y}$ plus ou moins dégradé par des imperfections présentes dans l'instrument de mesure. Ce signal de sortie est en plus affecté par la présence des sources de perturbations externes (bruits additifs).

Pour effectuer l'étude de la méthode de reconstitution proposée, nous avons généré des données synthétiques de type spectrométrique. Ainsi, en supposant que la réponse impulsionnelle du bloc de conversion est g, la génération de ces données est faite de sorte que les signaux d'entrée, $\dot{x}$, et de sortie, $\tilde{y}$, comportent toutes les informations relatives à la connaissance de la réponse impulsionnelle de l'instrument et aux perturbations affectant les signaux. Ainsi, disposant d'une bibliothèque de fonctions permettant la réalisation de cette étape, les signaux synthétiques ont été générés de la façon suivante :

$$\dot{x}(\lambda_n) = \sum_{i=1}^{N_p} A_i \, gaussoid(\lambda_n - P_i, \sigma) \tag{4.1}$$

$$g(\lambda_n) = 0.0525 \, gaussoid(\lambda_n, 8) \tag{4.2}$$

avec

$$gaussoid(\lambda, \sigma) = exp\left[-0.5\left(\frac{\lambda_n}{\sigma}\right)\right] \tag{4.3}$$

où $\dot{x}$ est le vecteur de valeurs recherchées (mesurande); g est la réponse impulsionnelle du bloc de conversion; A_i et P_i sont l'amplitude et la position du pic i respectievement; N_p est le nombre de pics dans le signal; σ est la largeur du pic; η est le bruit additif présent dans le signal et λ_n est la longueur d'ondes numérisée selon n = 1, 2, ..., N_p. Un exemple de la forme de g est illustré sur la figure 4.1

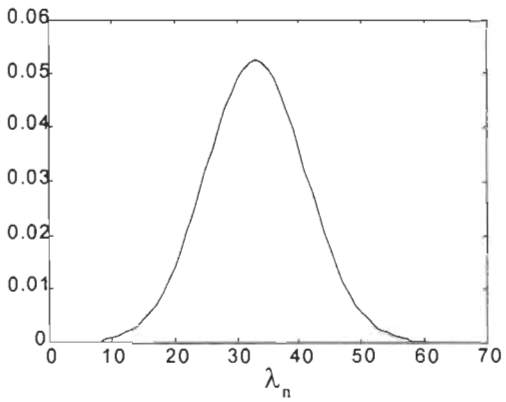


Figure 4.1 : Réponse impulsionnelle g

Le choix des données synthétiques s'est fait en se basant sur l'ensemble des données utilisées dans d'autres travaux présentant une problématique assez semblable à la nôtre [MAS95], [SCH98]. Ainsi, dans la forme de nos signaux, on essaye de représenter un certain nombre de cas rencontrés en spectrométrie. Parmi ces cas, on note principalement la variation de l'amplitude et de la position des pics. Cela nous permet alors de créer des effets de chevauchement des pics. On pouvait aussi, par la même procédure, générer des données

dont les pics ont des largeurs différentes. En essayant donc de représenter tous ces cas dans un signal de référence, on se retrouve facilement avec des signaux de 512 ou 1024 points.

Pour la méthode de reconstitution proposée, les signaux synthétiques générés sont réparties en trois banques de données selon la section 3.7.2. Nous avons:

- Banque de données d'étalonnage, $D^{étal}$, comportant 1024 points, elle est utilisée pour le réajustement des poids ;

- Banque de données de test, D^{test}, ayant 256 points et servant à optimiser la structure des réseaux;

- Banque de données de validation, D^{val}, utilisée pour observer les capacités de généralisation des réseaux. Ce signal comporte 512 points.

En utilisant les équations (4.1) et (4.2) pour générer ces signaux, les différentes valeurs des paramètres A_i et P_i figurent dans le tableau 4.1.

Ainsi, pour les systèmes linéaires, l'équation de sortie de l'instrument de mesure s'obtient de la manière suivante:

$$\tilde{y}(\lambda_n) = \dot{x}(\lambda_n) * g(\lambda_n) + \eta(\lambda_n) \tag{4.4}$$

Pour les signaux issus d'un système non linéaire, on peut choisir une fonction tangente hyperbolique (*tanh*) ou logarithmique (*Log*) pour simuler la non linéarité. Ainsi, en choisissant la fonction *tanh*, la sortie $\tilde{y}$ (équation 4.4) de l'instrument de mesure non

linéaire devient [SCH98]:

$$\tilde{y}(\lambda_n) = \tanh[\dot{x}(\lambda_n)] * g(\lambda_n) + \eta(\lambda_n) \tag{4.5}$$

Tableau 4.1 : Paramètres de génération des signaux de référence synthétiques

$\mathbf{D^{étal}}$						$\mathbf{D^{test}}$			$\mathbf{D^{val}}$		
i	A_i	P_i	i	A_i	P_i	i	A_i	P_i	i	A_i	P_i
1	2	30	15	6	540	1	6	50	1	4	30
2	5	50	16	5	610	2	3.5	70	2	8	50
3	3	80	17	2	640	3	2	90	3	5	70
4	6.5	130	18	6.5	680	4	5	110	4	5	90
5	6	180	19	7.5	720	5	6	150	5	7.5	110
6	7.5	200	20	6	760	6	6.5	165	6	7.5	130
7	5	280	21	5	830	7	7.5	195	7	2	160
8	2	300	22	3	850	8	5.5	225	8	6	180
9	6.5	330	23	5	860	9	3	240	9	3	200
10	3	380	24	4	880				10	6.5	230
11	7.5	430	25	8	900				11	5	265
12	6	470	26	6	920				12	6	300
13	3	470	27	2.5	935				13	3	320
14	7.5	510	28	5.5	965				14	2	340

Les signaux générés selon les données D^{app}, D^{test} et D^{val} sont illustés sur les figures 4.2 pour le cas linéaire, et 4.3 pour le cas non linéaire. Le niveau de bruit à injecter dans le signal y est caractérisé par le rapport signal sur bruit (SNR - *Signal Noise Ratio*) obtenu par l'équation suivante:

$$SNR = 20 \log \frac{\|\{\dot{y}_n\}\|_2}{\|\eta\|_2} \; ; \text{ avec } \eta = \tilde{y}_n - \dot{y}_n \tag{4.6}$$

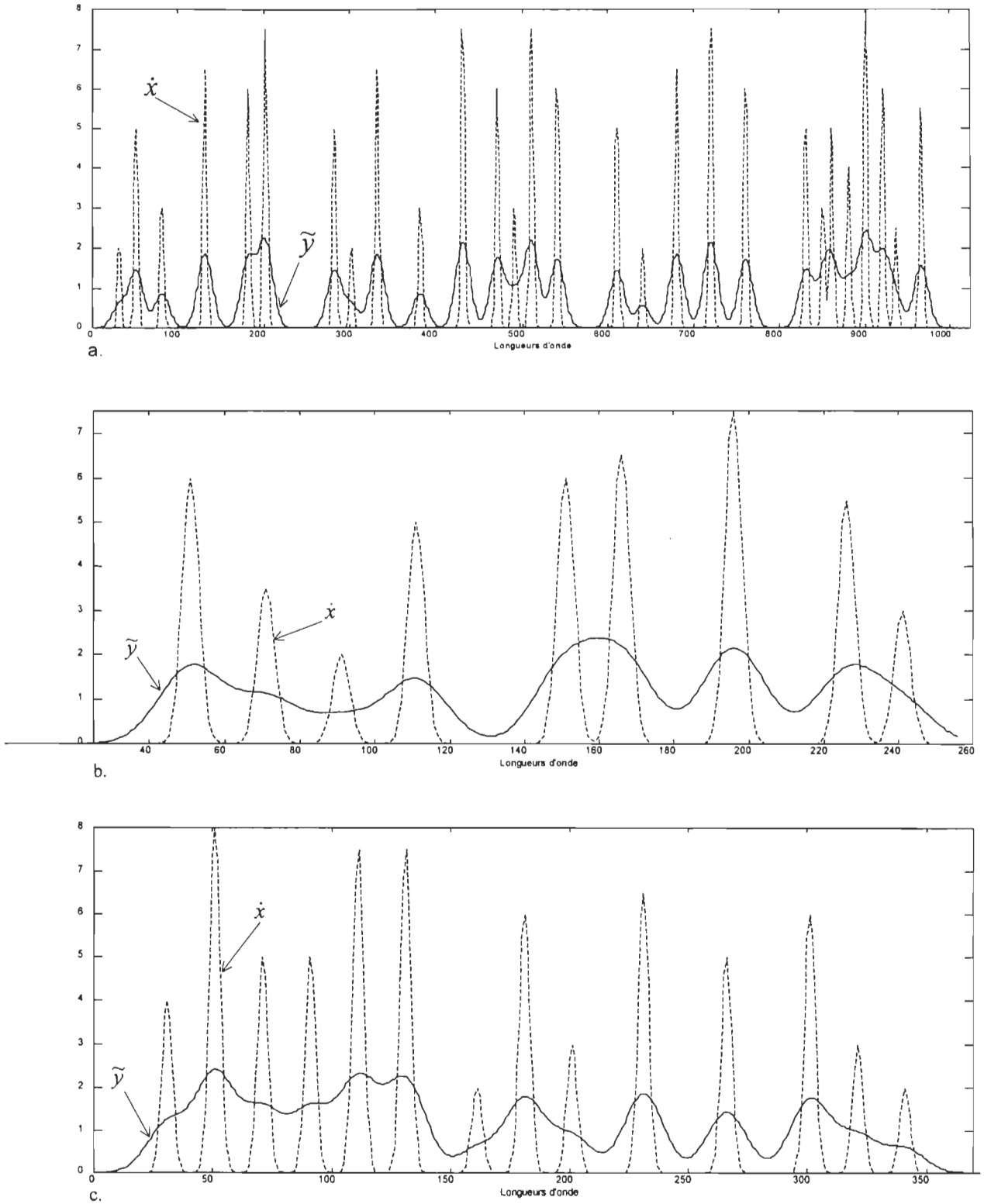


Figure 4.2 : Signaux de références synthétiques linéaires $D^{étal}$(a), D^{test}(b) et D^{val}(c).

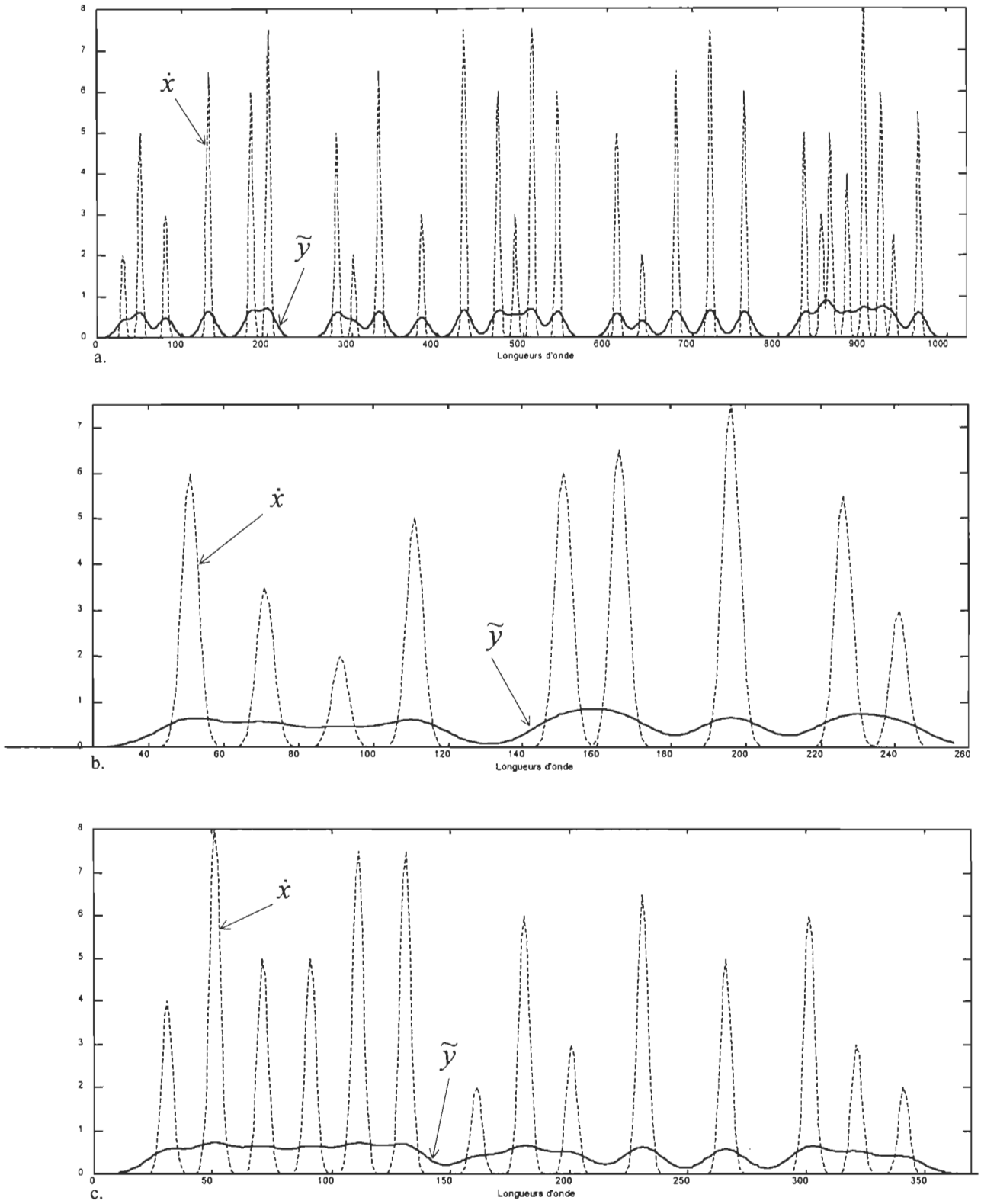


Figure 4.3 : Signaux de références synthétiques non linéaires $D^{étal}$(a), D^{test}(b) et D^{val}(c).

4.3 Étalonnage et optimisation des réseaux de neurones

Avant d'amorcer l'étalonnage proprement dit de nos modèles basés sur les réseaux de neurones, certaines étapes préliminaires devraient d'abord être suivies. D'abord, il faudra normaliser le signal d'entraînement $D^{étal}$. En effet, l'exploration du fonctionnement de l'algorithme montre qu'un bon étalonnage ou apprentissage est obtenu à l'aide d'un signal d'étalonnage normalisé. Cette étape étant possible grâce à une remise à l'échelle des données en utilisant la fonction *dscale.m*. On obtient ainsi un nouveau signal de moyenne zéro et de variance 1 comme le montre l'exemple de la figure 4.4.

On comprend aisément que les matrices des poids générées par l'algorithme sont aussi normalisées. Mais disposant du facteur de normalisation fourni par la fonction *dscale.m*, une dénormalisation de ces matrices est possible et nécessaire pour valider le signal $D^{étal}$ original. Cette dénormalisation est rendue possible grâce à la fonction *wrescale.m*.

Puis, après cette étape de mise au point, on ajuste certains paramètres d'entraînement dont les plus important sont le nombre d'itérations et le critère d'arrêt du processus. Ensuite, l'entraînement proprement dit est lancé en faisant appel à la fonction *nnoe.m*. Sachant que cet algorithme permet de former un réseau de neurones à trois couches, nous devrons nous mêmes fixer le nombre de neurones sur la couche d'entrée et sur la couche cachée; on a un seul neurone sur la couche de sortie.

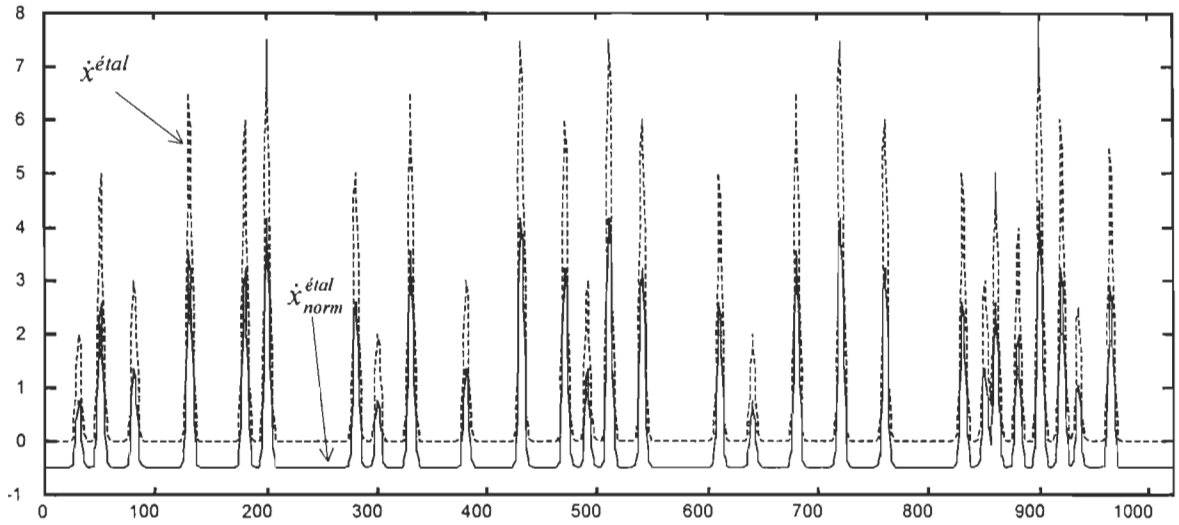


Figure 4.4: Signal d'apprentissage réel $\dot{x}^{étal}$ et normalisé $\dot{x}^{étal}_{norm}$ (moyenne 0 et variance 1)

Ainsi, m_x et m_y représentant le nombre de retards sur le signal estimé $\hat{x}$ et sur le signal d'entrée $\tilde{y}$ respectivement, on aura pour la couche d'entrée, $m_x + m_y + 1$ neurones; et pour la couche cachée, n_c neurones.

Le nombre n_c est fixé de façon empirique suivant des valeurs entières croissantes. Pour des fins d'analyses plus approfondies, on choisit de faire varier les trois nombres ci-dessus comme suit :

- m_x de 2 à 20 par pas de 2;

- m_y de 2 à 20 par pas de 2; et

- n_c = { 3, 4, 5, 10,15, 20 }.

De cette manière, la configuration des réseaux à entraîner se fera en faisant varier l'un des trois paramètres précédents pendant que les autres seront fixes. L'examen des résultats

obtenus après le processus d'entraînement des tous ces réseaux générés se fait selon une méthodologie précise. Le critère de comparaison des performances des différentes topologies de réseaux de neurones est le calcul de l'erreur relative moyenne. Elle est donnée par l'équation 4.7.

$$\varepsilon = \frac{\left\| \hat{x}(\lambda_n) - \dot{x}(\lambda_n) \right\|_2}{\left\| \dot{x}(\lambda_n) \right\|_2} = \sqrt{\frac{\sum_{n=1}^{N} \hat{x}(\lambda_n) - \dot{x}(\lambda_n)}{\sum_{n=1}^{N} \dot{x}(\lambda_n)}} \qquad (4.7)$$

D'abord, indépendemment du type du système (linéaire ou non linéaire), les réseaux de neurones présentant une erreur d'apprentissage $\varepsilon^{étal}$ au dessus de 0.1 ne peuvent offrir une bonne qualité de généralisation. Ainsi, notre premier critère de sélection des réseaux sera de recenser tous ceux dont cette erreur est plus petite ou égale à 0.1, ($\varepsilon^{étal} <= 0.1$).

Puis, parmi les réseaux de neurones satisfaisant à ce critère, on procédera au choix des réseaux optimaux. Dans notre étude, un réseau est dit optimal selon différents critères appliqués sur la reconstitution du signal D^{test} lorsqu'il est sans bruit ($\sigma^2_n = 0$) et bruité avec $\sigma^2_n = 10^{-4}$. Ainsi, en désignant respectivement par J_1 et J_2 les deux critères définis précédemment, leur satisfaction est obtenue lorsqu'un réseau de neurones offre une erreur de généralisation ε^{test} assez acceptable du signal de test D^{test}. Les matrices de poids $\mathbf{W}_1^{test}$ et $\mathbf{W}_2^{test}$ permettant d'obtenir ce résultat sont alors utilisées pour effectuer une généralisation du signal de validation D^{val}.

Ensuite, une fois que le réseau offrant les meilleures performances est trouvé, ce dernier subit un test de robustesse. En effet, étant donné que le processus d'entraînement des réseaux de neurones s'est fait avec un signal non bruité ($\sigma^2_n = 0$), ce test consiste à introduire des niveaux de bruits dans les signaux D^{test} et D^{val} pour ensuite observer les capacités de généralisation du réseau de neurones choisi. Les niveaux de bruits σ^2_n utilisés sont 10^{-8}, 10^{-6} et 10^{-4}.

4.4 Effets des variations du nombre de retards et de neurones cachés

Pour mieux comprendre les motifs de l'obtention de cette diversité de réseaux ayant de bonnes performances, on a pensé à étudier l'impact de la variation de certains paramètres fixés pour l'apprentissage sur les capacités des réseaux de neurones entraînés. Ainsi, l'étude sera effectuée sur la variation des erreurs d'apprentissage $\varepsilon^{étal}$, de test ε^{test} et de validation ε^{val} en fonction de l'initialisation aléatoire des poids de connexion, l'augmentation du nombre de retards sur le signal estimé, l'augmentation du nombre de retard sur le signal d'entrée et l'augmentation du nombre de neurones sur la couche cachée.

L'étude de l'effet des variations des trois derniers paramètres m_x, m_y, et n_c sera alors faite à partir de la meilleure topologie. Pour cela, on fixera deux paramètres pour en faire varier un seul. Mais avant de commencer chacune d'elle, il faut signaler que les poids de connexion seront initialisés à une valeur fixe choisie empiriquement dans chaque matrice. Ceci permet d'entraîner chaque topologie de réseau de neurones une seule fois et de pouvoir répéter les

apprentissages. La justification de cette procédure réside dans le fait que l'initialisation aléatoire des matrices des poids ne garantit pas une répétition des apprentissages et de ce fait, un réseau peut présenter des performances médiocres pour un nombre d'apprentissages insuffisant. Ainsi, aucune comparaison des performances n'est alors possible entre les différentes topologies. Pour 50 rétropropagations, l'étude est faite en observant les erreurs à la fin de chaque apprentissage.

Ainsi, de l'initialisation aléatoire des poids (figure 4.5a), on s'aperçoit alors que pour un réseau ayant m_x=6, m_y=10 et n_c=5, les meilleurs résultats sont obtenus au dixième apprentissage des mêmes données. En faisant varier m_x de 4 à 10, la plus petite erreur est obtenue pour m_x=6 (figure 4.5b). Enfin, pour la variation de m_y, la valeur 10 fournit les meilleurs résultats(figure 4.5c). L'augmentation du nombre de neurones n_c sur la couche cachée (figure 4.5d) n'améliore pas forcément ces erreurs mais pourrait assurer une certaine robustesse du réseau face au bruit.

À partir de ces graphiques, on constate à première vue que l'augmentation du nombre de retards sur chacun des signaux d'entrée du réseau de neurones contribue à l'amélioration de ses capacités d'apprentissage. La variation du nombre de neurones sur la couche cachée n'influence pas beaucoup les capacités d'apprentissage du réseau. Une garantie pour assurer un bon apprentissage serait d'avoir un nombre optimal de retards sur l'entrée du réseau.

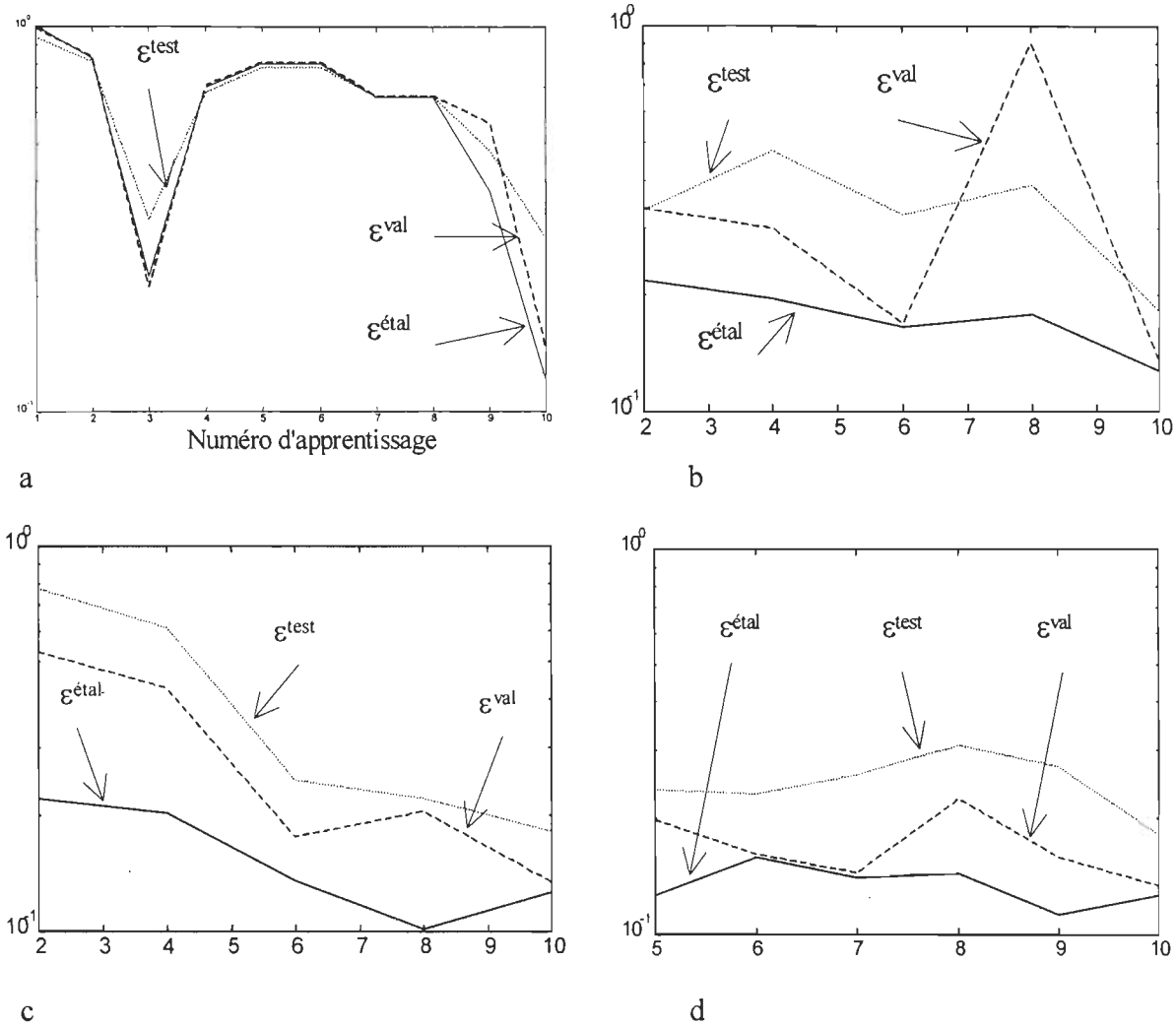


Figure 4.5 : a) Effet de l'initialisation aléatoire des matrices des poids $\mathbf{W}_1$ et $\mathbf{W}_2$ pour $m_x = 6$, $m_y = 10$ et $n_c = 5$.

b) Effet de la variation du nombre de retards m_x sur l'estimé du réseau pour $\mathbf{W}_1$, $\mathbf{W}_2 = 0.1$, $m_y = 10$ et $n_c = 5$.

c) Effet de la variation du nombre de retards m_y sur l'entrée du réseau.pour $\mathbf{W}_1$, $\mathbf{W}_2 = 0.1$, $m_x = 6$ et $n_c = 5$.

d) Effet de la variation du nombre de neurones n_c sur la couche cachée pour $\mathbf{W}_1$, $\mathbf{W}_2 = 0.1$, $m_x = 6$ et $m_y = 10$.

4.5 Résultats de reconstitution pour un système linéaire

Après avoir opté pour une initialisation non aléatoire des matrices des poids de pondération, nous avons entamé l'entraînement des réseaux de neurones dont la configuration des topologies est décrite dans la section 4.3. L'analyse des résultats est alors faite sur plusieurs observations. Ainsi, les premières caractéristiques observées sont les variations des erreurs d'étalonnage et de test en fonction des variations des paramètres m_x, m_y et n_c.

4.5.1 Résultats selon le critère J_1

De toutes les courbes de répartition des erreurs d'apprentissage et de test générées, les plus intéressantes se trouvent être celles fournies par l'ensemble des réseaux à trois (3) neurones sur la couche cachée (figure 4.6). Sur ces figures, il est établi que plus une zone est ombrée, plus l'erreur de reconstitution décroît. En procédant par la suite à la recherche du ou des réseaux présentant de bonnes capacités de généralisation, on trouve celui dont le nombre de retards sur chaque signal (m_x et m_y) est de 14. Comme nous le verrons dans le chapitre 5, l'égalité de m_x et m_y facilite l'implantation de l'algorithme en technologie VLSI.

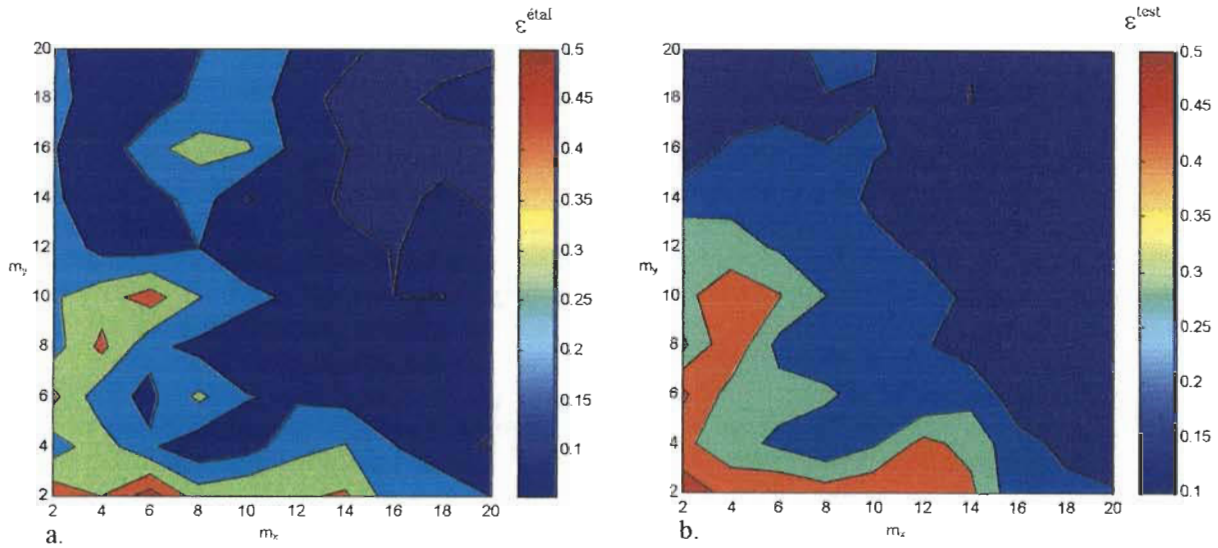


Figure 4.6: Répartition de l'erreur d'apprentissage $\varepsilon^{étal}$ (a) et de test ε^{test} (b) en fonction des paramètres m_x et m_y pour les signaux linéaires avec $n_c = 3$ et $\sigma^2_n = 0$.

Dans la suite de cette section, nous allons donc présenter les différents résultats extraits des performances de ce réseau selon les paramètres suivants: $m_x = 14$, $m_y = 14$, $n_c = 3$, $\mathbf{W}_1$ et $\mathbf{W}_2$ ayant tous les éléments initialisés à 0.1.

L'analyse des performances de ce réseau s'est faite en examinant l'évolution des différentes erreurs pendant l'entraînement du réseau de neurones. Cependant, une constatation générale a été ressortie lorsque l'on observait l'évolution des courbes d'erreurs de test et de validation. En effet, il se trouve que la courbe de l'erreur de test est toujours, à quelques exceptions près, au dessus de celle de l'erreur de validation. L'interprétation qui en découle est que le signal de test est beaucoup plus difficile à estimer que celui de la validation; et ce peu importe la nature des signaux. C'est ce qui justifie la classification faite sur les signaux utilisés dans ces travaux. Ainsi, on définit *le point de reconstitution optimale* comme étant l'itération de rétropropagation à laquelle le réseau de neurones

l'itération de rétropropagation à laquelle le réseau de neurones atteint l'erreur de généralisation minimale sur le signal de test. C'est le critère J_1 défini tel que:

$$J_1 = \min \{\varepsilon_i^{test} / i = 1,2,..., 60 \quad (4.8)$$

Pour revenir aux performances du réseau de neurones choisi plus haut, ses courbes d'erreurs (figure 4.7) indiquent que le point optimal où le signal D^{test} est mieux reconstitué se trouve à la $41^{ième}$ itération ($P^{opt} = 41$). À ce point, l'erreur d'apprentissage est de 0.0906 tandis que les erreurs de reconstitution des signaux D^{test} et D^{val} sont 0.1166 et 0.1246 respectivement.

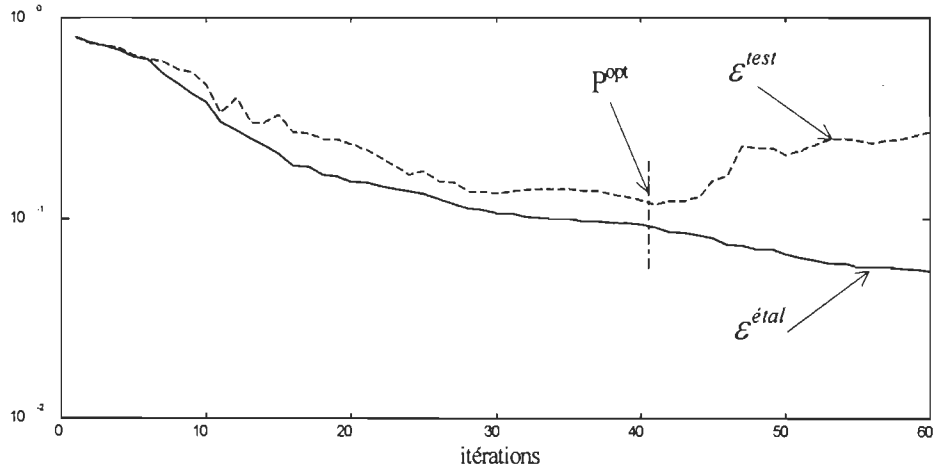


Figure 4.7: Courbes d'erreurs durant l'entraînement du réseau de neurones pour le système linéaire avec m_x=14, m_y=14, n_c=3 et $\mathbf{W}_1$ et $\mathbf{W}_2$ initialisés à 0.1.

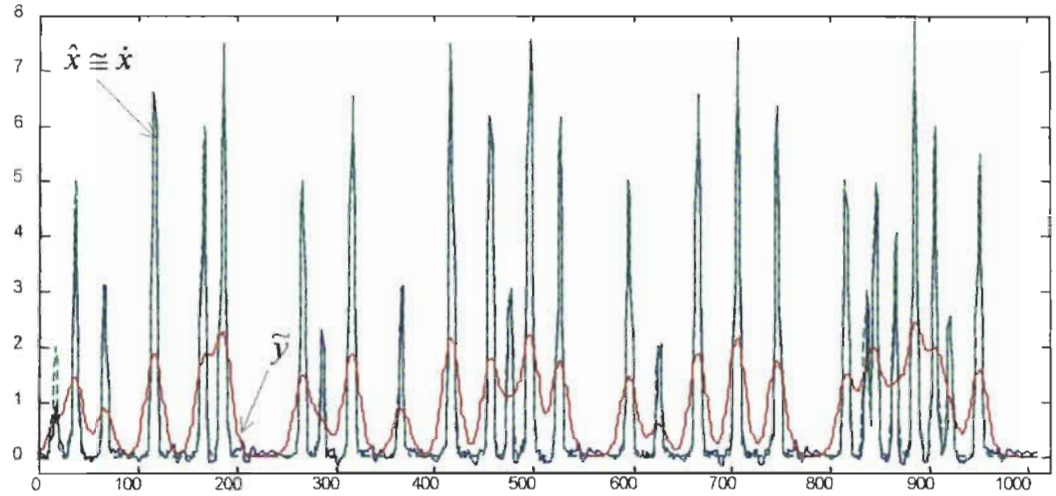


Figure 4.8 : Résultat d'apprentissage du réseau au point optimal P^{opt} ($\varepsilon^{étal}$=0.0906) selon J_1.

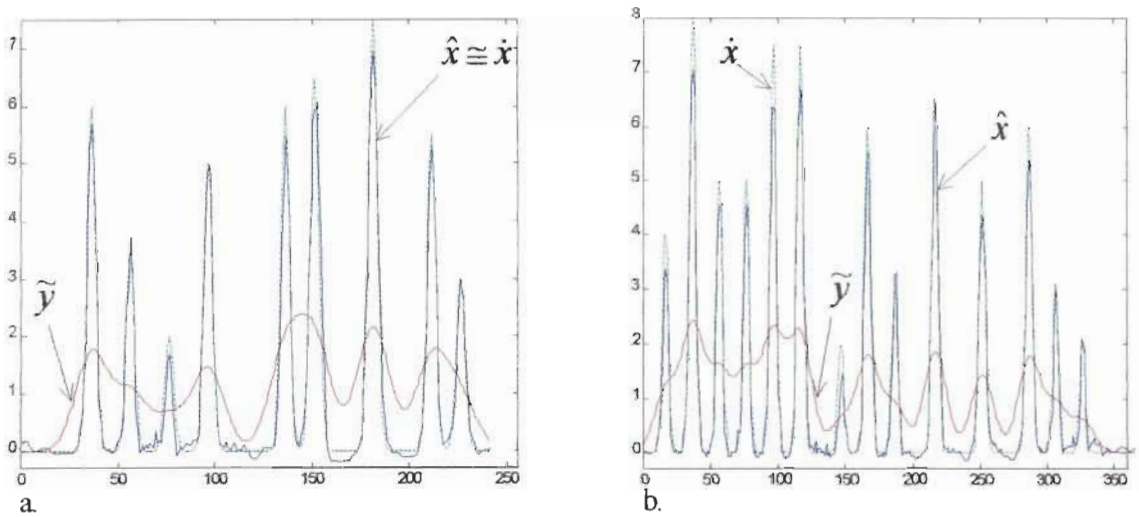


Figure 4.9 : Résultats de reconstitution optimale de D^{test} (ε^{test} = 0.1166) (a) et D^{val} (ε^{val} =0.1246) (b) pour le système linéaire avec σ_n^2=0.

Ces résultats ont été obtenus avec des signaux dépourvus de bruit (σ_n^2=0). Pour évaluer la robustesse de notre réseau, on va reconstituer ces mêmes signaux, mais affectés cette fois-ci

de bruits σ_n^2 dont les valeurs sont 10^{-8}, 10^{-6} et 10^{-4}. Pour les deux signaux D^{test} et D^{val}, on obtient le tableau 4.2 selon le critère J_1.

Tableau 4.2 : Erreurs relatives quadratiques moyennes (ε) de reconstitution des signaux linéaires D^{test} et D^{val} affectés de différents niveaux de bruit (σ_n^2) selon J_1.

σ_n^2	0	10^{-8}	10^{-6}	10^{-4}
SNR[dB], ε^{test}	Inf, 0.1166	81.84, 0.1172	61.84, 0.1236	41.84, 0.3302
SNR[dB], ε^{val}	Inf, 0.1246	80.92, 0.1243	60.92, 0.1238	40.92, 0.2489

Sur ces deux signaux, on voit que l'erreur de reconstitution pour un bruit de $\sigma_n^2=10^{-8}$ est sensiblement égale à celle de la reconstitution des signaux non bruités. Pour un bruit de $\sigma_n^2=10^{-6}$, les résultats de reconstitution des deux signaux D^{test} et D^{val} sont illustrés à la figure 4.10.

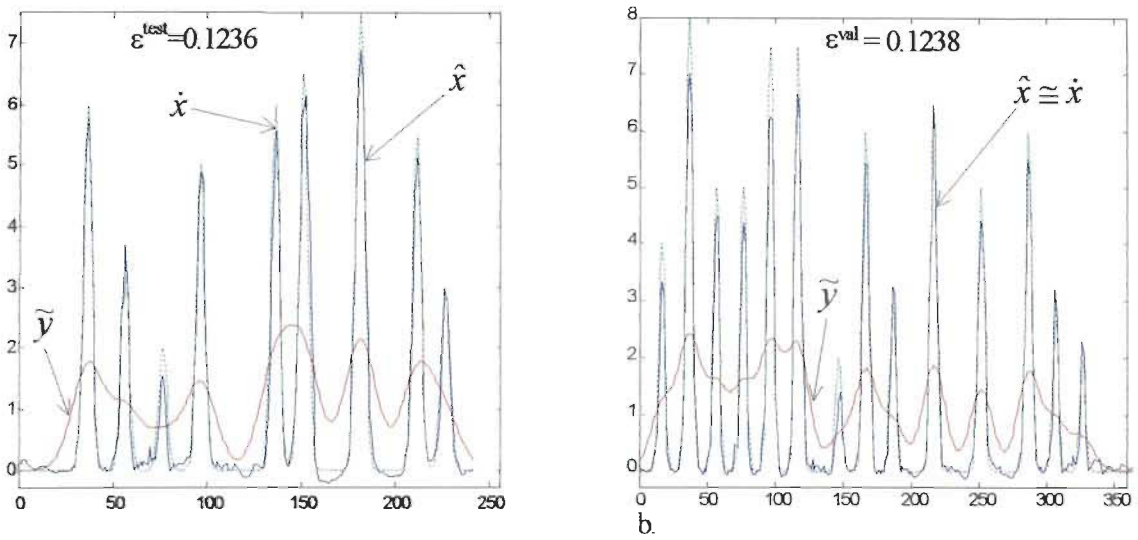


Figure 4.10 : Reconstitutions des signaux linéaires D^{test} (a) et D^{val} (b) affectés d'un bruit $\sigma_n^2=10^{-6}$ avec $m_x=14$, $m_y=14$ et $n_c=3$ selon J_1.

4.5.2 Résultats selon le critère J_2

Selon le critère J_2, les courbes d'erreurs les plus intéressantes sont fournies par l'ensemble des réseaux à dix (10) neurones sur la couche cachée (figure 4.11). Puis, c'est le réseau avec $m_x=14$, $m_y=14$ et $n_c=10$ qui offre les meilleurs résultats. Les figures 4.12 et 4.13(a et b) illustrent les performances d'apprentissage et de reconstitution de ce réseau, tandis que les erreurs relatives moyennes de reconstitution des signaux bruités sont dans le tableau 4.3

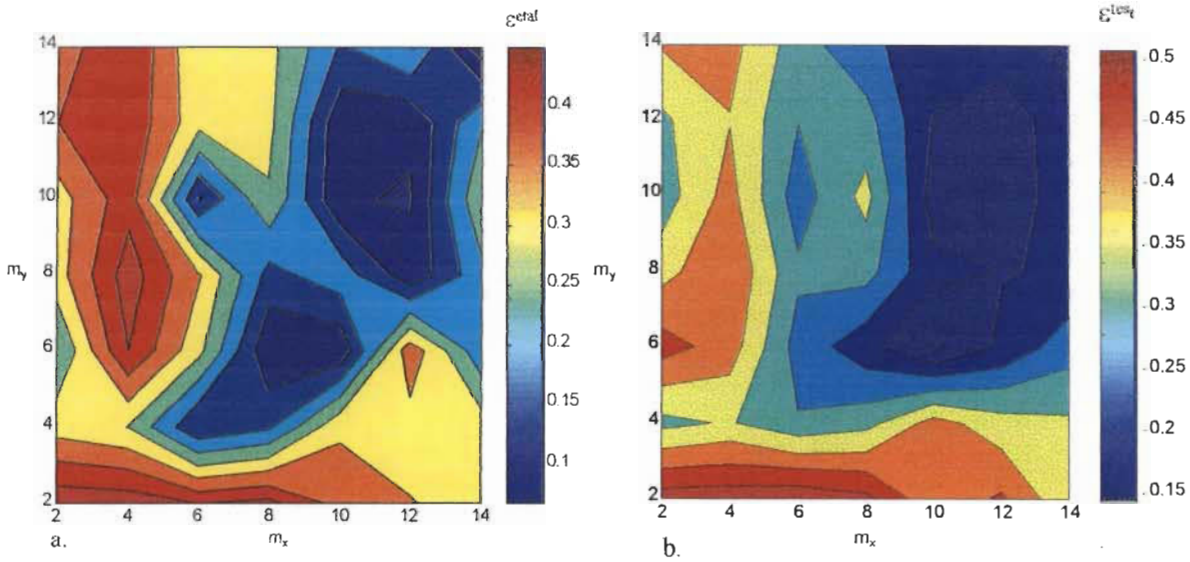


Figure 4.11: Erreur relative moyenne pour l'apprentissage $\varepsilon^{étal}$ avec $\sigma^2_n = 0$ (a) et pour le test ε^{test} avec $\sigma^2_n = 10^{-4}$ (b) en fonction des paramètres m_x et m_y pour les signaux linéaires avec $n_c = 10$.

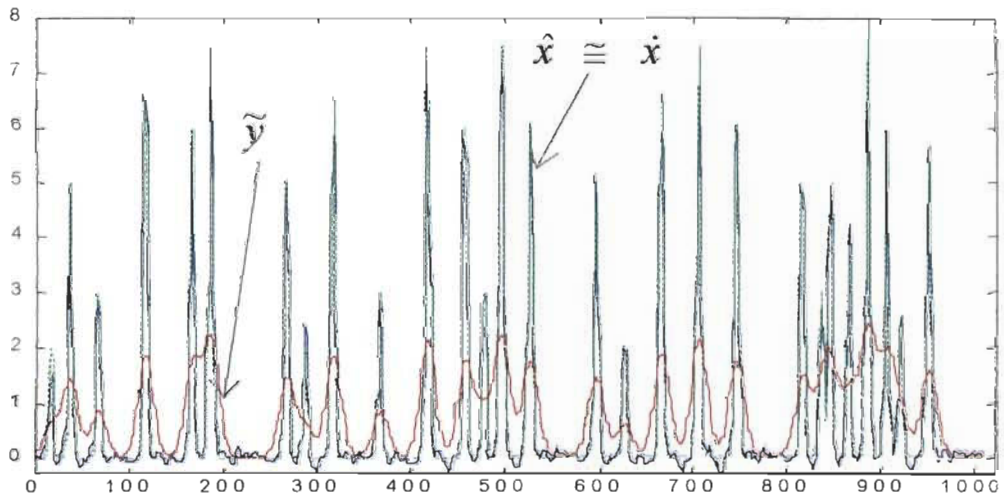


Figure 4.12 : Résultat d'apprentissage du réseau au point optimal P^{opt} ($\varepsilon^{étal}$=0.0675) selon J_2.

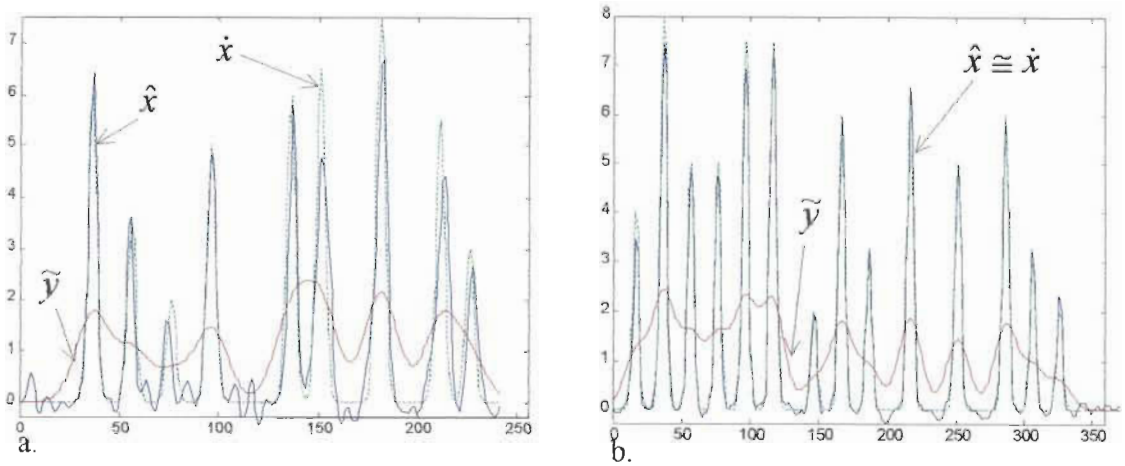


Figure 4.13 : Résultats de reconstitution optimale de D^{test} (ε^{test} = 0.2774) (a) et D^{val} (ε^{val} =0.0906) (b) pour le système linéaire avec $\sigma_n^2=10^{-4}$.

Tableau 4.3 : Erreurs relatives quadratiques moyennes (ε) de reconstitution des signaux linéaires D^{test} et D^{val} affectés de différents niveaux de bruit (σ_n^2) selon J_2.

σ_n^2	0	10^{-8}	10^{-6}	10^{-4}
SNR[dB],	Inf,	81.84,	61.84,	41.84,
ε^{test}	0.1655	0.1655	0.1665	0.2775
SNR[dB],	Inf,	80.92,	60.92,	40.92,
ε^{val}	0.0906	0.0908	0.0938	0.2955

On remarque que pour des niveaux de bruits plus importants que 62dB ($\sigma_n^2=10^{-6}$), l'erreur de généralisation du réseau augmente considérablement.

4.6 Résultats de reconstitution pour un système non linéaire

Tout comme pour le système linéaire, les résultats des entraînements effectués avec les signaux d'un système non linéaire montrent l'existence d'une multitude de réseaux satisfaisant au critère J_0 établi pour s'assurer d'une bonne généralisation ($\varepsilon^{étal} <= 0.1$).

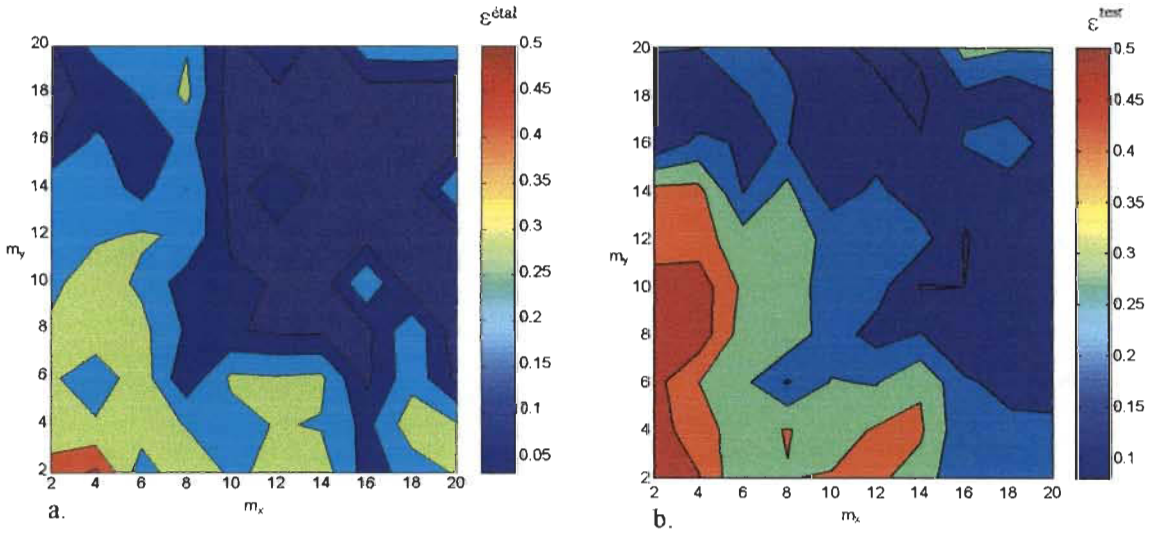


Figure 4.14: Répartition de l'erreur d'apprentissage(a) et de test (b) en fonction des nombres de retards m_x et m_y pour les signaux non linéaires avec $n_c = 5$ et selon J_1.

4.6.1 Résultats selon J_1

Suivant le critère J_1, les réseaux satisfaisants sont ceux comportant cinq neurones (n_c=5) sur la couche cachée. Les répartitions des erreurs d'étalonnage et de test sont illustrés sur la figure 4.14. De cette figure, on recherche le réseau qui présente les meilleures performances d'apprentissage et d'estimation du signal D^{test}. Sa topologie est faite à partir

retards (m_x et m_y) égaux à 18. Comme pour le cas linéaire, les figures 4.15 à 4.18 sont obtenues de ce réseau de neurones.

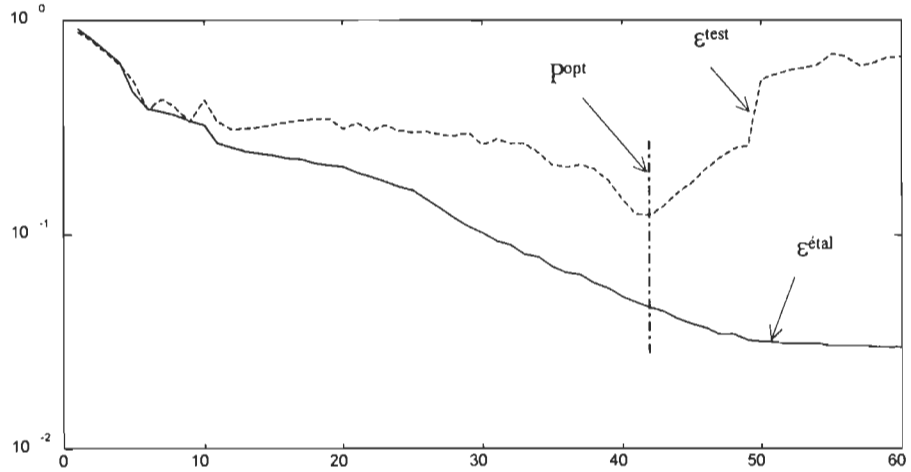


Figure 4.15: Courbes d'erreurs après entraînement du réseau non linéaire avec m_x=18, m_y=18, n_c = 5 et $\mathbf{W}_1$,$\mathbf{W}_2$ initialisés à 0.1.

Au point optimal de reconstitution (42^{ième} itération), l'erreur d'apprentissage $\varepsilon^{étal}$ est de 0.0459 tandis que celles de test et de validation sont respectivement 0.1233 et 0.1124. Les signaux correspondant à ces résultats de sortie du réseau ayant m_x=18, m_y=18 et n_c=5 sont illustrés sur les figures 4.16 et 4.17.

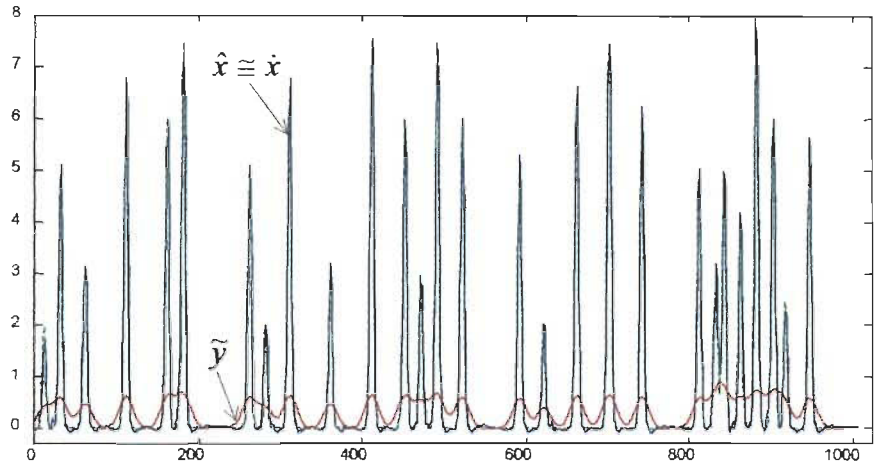


Figure 4.16 : Résultat d'apprentissage du réseau ayant m_x=18, m_y=18 et n_c=5 au point optimal ($\varepsilon^{étal}$ = 0.0459) selon J_1.

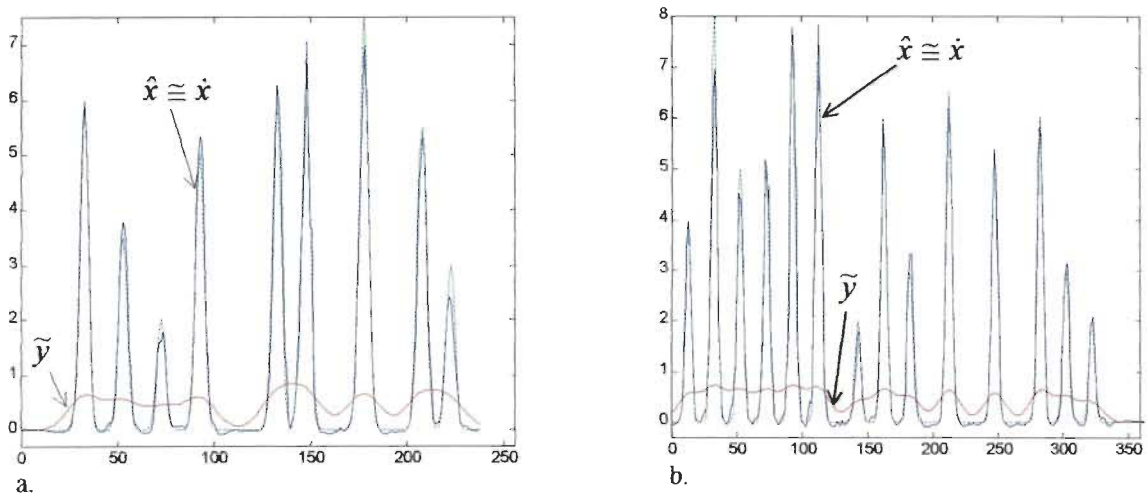


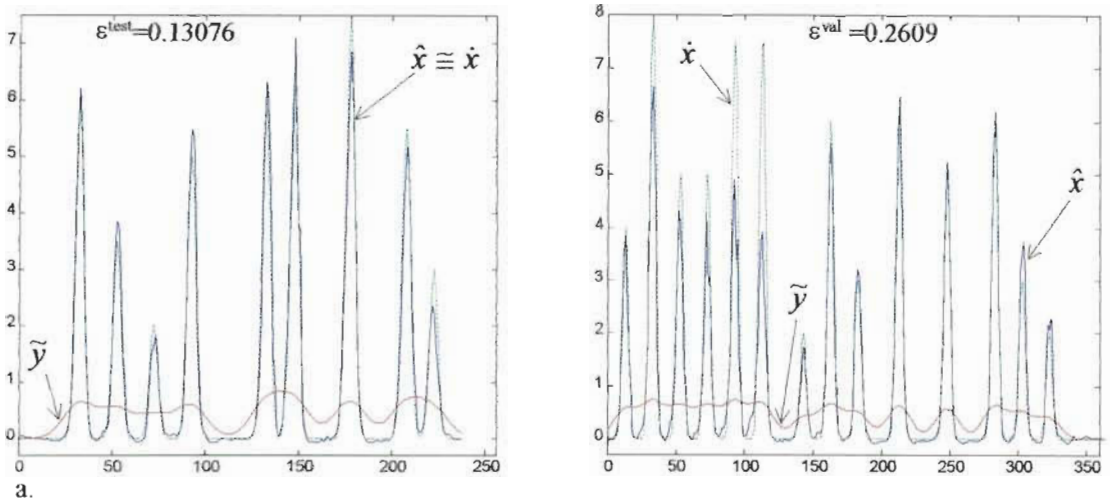
Figure 4.17 : Résultats de reconstitution optimale avec σ_n^2=0 de D^{test} (a) (ε^{test} = 0.1233) et D^{val} (b) (ε^{val} = 0.1124) d'un système non linéaire avec m_x=18, m_y=18 et n_c=5.

En ce qui concerne le comportement de notre réseau face au bruit, on va suivre la même procédure que celle effectuée pour les signaux linéaires, on obtient alors le tableau 4.4 suivant :

Tableau 4.4 : Erreurs relatives quadratiques moyennes (ε) de reconstitution des signaux non linéaires D^{test} et D^{val} affectés de différents niveaux de bruit (σ_n^2) selon J_1

σ_n^2	0	10^{-8}	10^{-6}	10^{-4}
SNR[dB], ε^{test}	Inf, 0.1233	81.84, 0.1238	61.84, 0.1308	41.84, 0.3020
SNR[dB], ε^{val}	Inf, 0.1124	80.92, 0.1112	60.92, 0.2609	40.92, 0.6046

Des remarques identiques aux signaux linéaires sont faites quant à la non sensibilité du réseau aux bruits inférieurs à $\sigma_n^2=10^{-8}$. Aussi, le réseau est capable d'effectuer une estimation assez acceptable jusqu'à $\sigma^2=10^{-6}$ (figure 4.18).



a.

Figure 4.18 : Estimations des signaux non linéaires D^{test} (a) et D^{val} (b) affectés d'un bruit $\sigma_n^2=10^{-6}$ avec $m_x=18$, $m_y=18$ et $n_c=5$.

4.6.2 Résultats selon le critère J_2

En appliquant le critère J_2 comme pour le cas linéaire, l'ensemble formé par les réseaux à 10 neurones sur la couche cachée s'avère plus apte. La répartition de différentes erreurs est illustrée par la figure 4.19. Comme dans le cas linéaire, c'est le réseau avec $m_x=14$, $m_y=14$ et $n_c=10$ qui offre les meilleurs résultats. Leurs illustrations sont faites sur les figures 4.20 et 4.21(a et b) pour les résultats d'apprentissage et de reconstitution respectivement, et par le tableau 4.5 en ce concerne les erreurs relatives moyennes de reconstitution des signaux bruités.

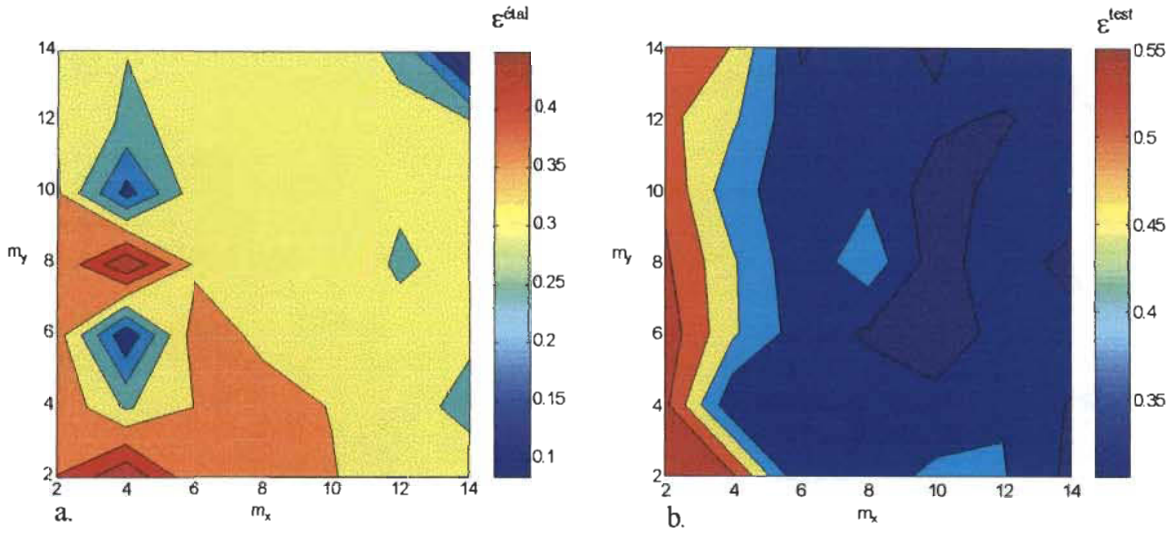


Figure 4.19: Erreur relative moyenne pour l'apprentissage $\varepsilon^{étal}$ avec $\sigma^2_n = 0$ (a) et pour le test ε^{test} avec $\sigma^2_n = 10^{-4}$ (b) en fonction des paramètres m_x et m_y pour les signaux non linéaires avec $n_c = 10$.

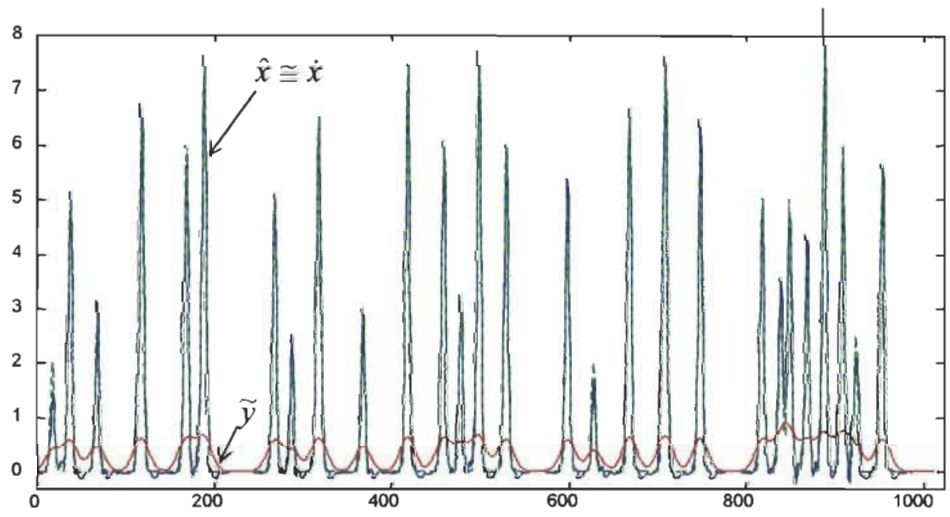


Figure 4.20: Résultat d'apprentissage du réseau au point optimal P^{opt} ($\varepsilon^{étal}$=0.0849) selon J_2.

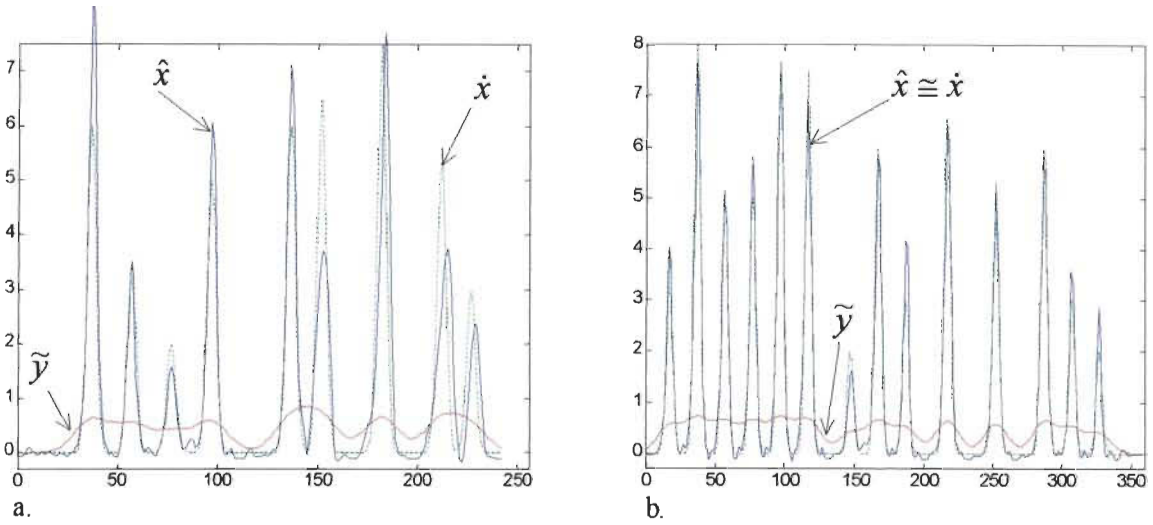


Figure 4.21: Résultats de reconstitution optimale de D^{test} (ε^{test} = 0.388) (a) et D^{val} (ε^{val} =0.1285) (b) pour le système non linéaire avec $\sigma_n^2=10^{-4}$.

Tableau 4.5 : Erreurs relatives quadratiques moyennes (ε) de reconstitution des signaux non linéaires D^{test} et D^{val} affectés de différents niveaux de bruit (σ_n^2) selon J_2.

σ_n^2	0	10^{-8}	10^{-6}	10^{-4}
SNR[dB], ε^{test}	Inf, 0.2819	81.84, 0.2808	61.84, 0.2731	41.84, 0.3878
SNR[dB], ε^{val}	Inf, 0.1285	80.92, 0.1433	60.92, 0.3881	40.92, 0.6225

Pour des niveaux de bruits plus importants que 60dB ($\sigma_n^2=10^{-6}$), un constat identique à celui de la section 4.5 est fait.

4.7 Comparaison avec d'autres méthodes de reconstitution

L'étude effectuée dans ce paragraphe consiste à comparer les résultats de reconstitution [MAS99] entre la méthode proposée basée sur les réseaux de neurones multicouches avec d<autres méthodes couramment utilisées (Jansson[CRI91], Van Cittert [CRI91] et Kalman[MAS95]). Pour cela, nous allons utiliser les réseaux de neurones - pour les cas linéaire et non linéaire - sélectionnés dans les sections 4.5 et 4.6 afin d'étudier leur robustesse face aux perturbations. Ainsi, en introduisant dans les signaux D^{val} et D^{test} des bruits avec σ_n^2 égal á 10^{-8}, 10^{-6} et 10^{-4}, les résultats obtenus des réseaux de neurones seront alors comparés à ceux fournis par les autres méthodes. Mais auparavant, une brève présentation des modèles de ces méthodes peut se faire comme suit:

Jansson -

$$\{\hat{x}_n\} = JANSSON(\{\tilde{y}_n\};\{h\},a,b,c,k) \tag{4.9}$$

Van Cittert -

$$\{\hat{x}_n\} = VanCittert(\{\tilde{y}_n\};\{h_n\},b,k) \tag{4.10}$$

Kalman -

$$\{\hat{x}_n\} = KALMAN(\{\tilde{y}_n\};\{h_n\},k_\infty,\beta,\alpha) \tag{4.11}$$

Pour la méthode proposée dans cette étude, elle peut se modéliser par l'équation suivante:

$$\{\hat{x}_n\} = RNA(\{\tilde{y}_n\},W_1,W_2,n_c,m_x,m_y) \tag{4.12}$$

4.7.1 Résultats pour un système dynamique linéaire

Pour le système linéaire, on compare dans un premier temps les résultats de reconstitution du réseau de neurones selon les paramètres suivants: m_x=14, m_y=14 et n_c=3, avec ceux obtenus à l'aide des méthodes de Jansson, de Van Cittert et du filtre de Kalman (figure 4.22). À cette fin, on utilisera les signaux de validation D^{val}.

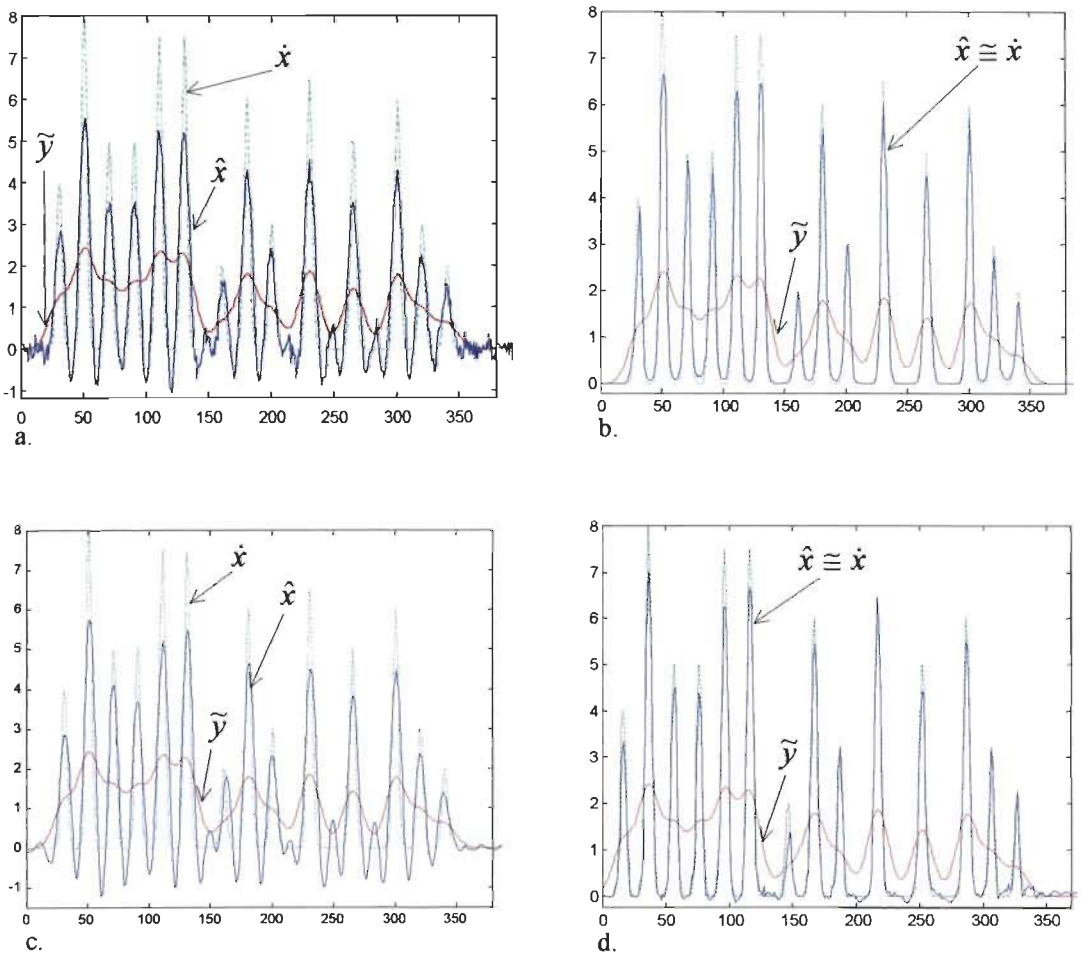


Figure 4.22: Résultats de reconstitution pour un système linéaire avec $\sigma_n^2=10^{-6}$:

a) Jansson, b) Van Cittert, c) Kalman et d) Réseau de neurones.

Le critère permettant de faire cette comparaison est une fois de plus le calcul de l'erreur quadratique relative moyenne entre le signal reconstitué et le signal idéal. Les résultats sont montrés au tableau 4.6 pour différents niveaux de bruits appliqués aux données D^{test} et D^{val}.

Tableau 4.6: Erreurs relatives quadratiques moyennes (ε) pour les méthodes de reconstitution des signaux linéaires bruités.

Méthodes		$\{\tilde{y}(n), \hat{x}(n)\}^{test}$			$\{\tilde{y}(n), \hat{x}(n)\}^{val}$		
	σ_η^2	10^{-8}	10^{-6}	10^{-4}	10^{-8}	10^{-6}	10^{-4}
	SNR	75 dB	55 dB	35 dB	75 dB	55 dB	35 dB
RNA	ε	0.1172	0.1236	0.3302	0.1243	0.1238	0.2489
Kalman	ε, β^{opt}	0.5441, 10^4	0.5188, 10^3	0.5142, 10^3	0.3714, 10^4	0.3717, 10^4	0.4059, 10^3
Jansson	ε, k^{opt}	0.1355, 219	0.1499, 195	0.3165, 69	0.1207, 192	0.1282, 181	0.2862, 66
Van Cittert	ε, k^{opt}	0.3286, 237	0.3401, 219	0.5036, 43	0.3552, 178	0.3601, 138	0.4535, 36

4.7.2 Résultats pour un système dynamique non linéaire

En ce qui concerne le cas non linéaire, une procédure identique à celle des signaux linéaires est suivie. On obtient alors les résultats de la figure 4.23 et du tableau 4.7.

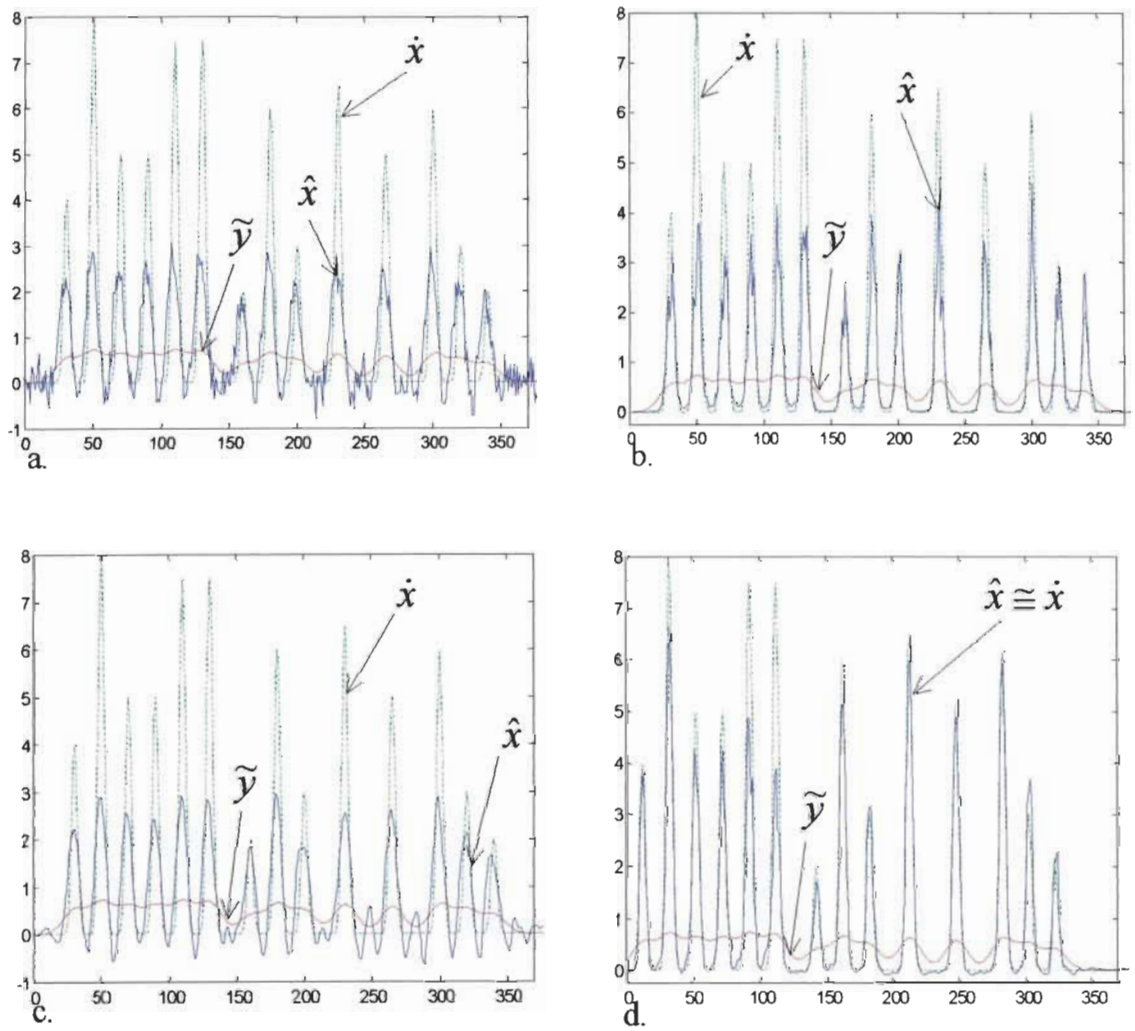


Figure 4.23: Résultats de reconstitution pour un système non linéaire avec $\sigma_\eta^2=10^{-6}$:

a) Jansson , b) Van Cittert, c) Kalman et d) Réseau de neurones.

Tableau 4.7: Erreur relative quadratique moyenne (ε) pour les méthodes de reconstitution des signaux non linéaires bruités.

Méthodes		$\{\tilde{y}(n), \hat{x}(n)\}^{test}$			$\{\tilde{y}(n), \hat{x}(n)\}^{val}$		
	σ_η^2	10^{-8}	10^{-6}	10^{-4}	10^{-8}	10^{-6}	10^{-4}
	SNR	75 dB	55 dB	35 dB	75 dB	55 dB	35 dB
RNA	ε	0.2934	0.2918	0.2781	0.1983	0.1890	0.1717
Kalman	ε, β^{opt}	0.6076, 10^4	0.6060, 10^4	0.6135, 10^4	0.5257, 10^5	0.5278, 10^5	0.5597, 10^4
Jansson	ε, k^{opt}	0.3992, 876	0.4333, 521	0.5754, 100	0.4337, 662	0.4511, 471	0.5809, 99
Van Cittert	ε, k^{opt}	0.4574, 861	0.5335, 451	0.6478, 40	0.5206, 725	0.5432, 233	0.6493, 41

Dans ces tableaux et dans les deux cas, l'indication 'opt' désigne la valeur optimale du paramètre. β est le paramètre de régularisation du filtre de Kalman tandis que *k* est le nombre d'itérations effectuées pour les méthodes de Jansson et Van Cittert.

Comme on le remarque, les réseaux de neurones offrent des performances comparables, et parfois meilleures, à celles de certaines autres méthodes.

De plus, les performances des réseaux de neurones varient peu avec le niveau de bruit. Tandis que avec les autres méthodes, il faudra réajuster certains paramètres comme β pour Kalman et le nombre d'itérations pour Jansson et Van Cittert, pour obtenir les performances optimales.

4.7.3 Analyse de la complexité algorithmique

La complexité des algorithmes utilisés pour cette comparaison est évaluée en fonction du nombre de multiplications et d'additions effectuées durant le processus de reconstitution des signaux [MAS98].

Ainsi, soient n_c le nombre de neurones sur la couche cachée, M la dimension de la réponse impulsionelle définie dans l'équation (4.2) et N le nombre de points contenus dans le signal $\widetilde{y}$, et k le nombre d'itérations, on obtient les résultats du tableau 4.6. De ce dernier, on voit bien que pour le signal D^{test} de 256 (N=256) points et dont la réponse impulsionnelle de génération a 64 points (M=64), la complexité des algorithmes basés sur notre méthode et sur le filtrage de Kalman est pratiquement au même niveau.

En effet, pour $m_x=m_y=n_c=10$, les réseaux de neurones effectueront 220N additions et 209N multiplications tandis que le filtre de Kalman en requiert 128N additions et 128N multiplications.

Le nombre d'opérations effectuées par les autres méthodes a un facteur de multiplication équivalent au nombre d'itérations. On a 67Nk multiplications pour la méthode de Jansson, soient 33500N multiplications pour 500 itérations par exemple. Cet algorithme et celui de Van Cittert deviennent alors très lents pour une réalisation en technologie ITGE.

Tableau 4.8: Complexité des algorithmes de reconstitution des signaux.

Algorithmes	Multiplications	Additions
RNA	$\left[\left(m_x + m_y + 2\right)n_c\right]N$	$\left[\left(m_x + m_y + 1\right)n_c - 1\right]N$
Kalman	2MN	2MN
Jansson	N(M+3)k	(NM+3N-M+1)k
Van Cittert	(NM+1)k	(NM+N-M+1)k

4.8 Résultats de reconstitution pour les données expérimentales

Les données réelles sur lesquelles notre méthode est appliquée proviennent d'un analyseur de spectre optique (OSA : Optical Spectrum Analyzer).

Avant d'entreprendre la phase d'étalonnage du réseau de neurones, une forme approximative du signal recherché a été estimée. Cette estimation a été rendu possible grâce à la connaissance de l'information à extraire : la largeur de la base des pics. Quant au reste du signal (partie bruitée), une approximation polynomiale y est appliquée afin d'obtenir une courbe lisse. C'est de cette manière que nous avons élaboré nos banques de données pour l'étalonnage et la validation du réseau.

Après avoir entraîné une série de topologies de réseaux, les meilleurs résultats sont obtenus avec un réseau avec m_x=4, m_y=4 et n_c=5. Ils sont illustrés sur les figures 4.24 à 4.26.

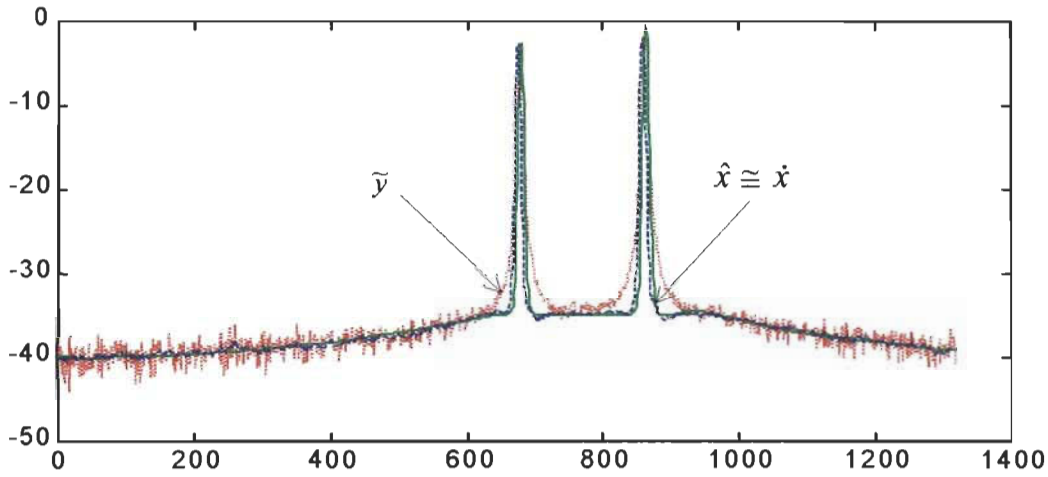


Figure 4.24: Apprentissage des données réelles par le réseau ayant m_x=4, m_y=4 et n_c=5

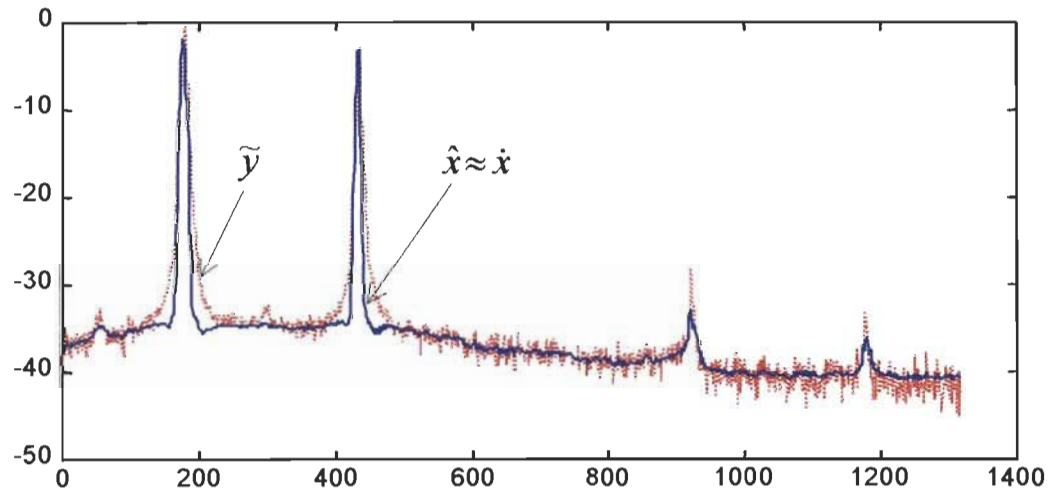


Figure 4.25: Reconstitution du signal de validation réel du réseau avec m_x=m_y=4, n_c=5.

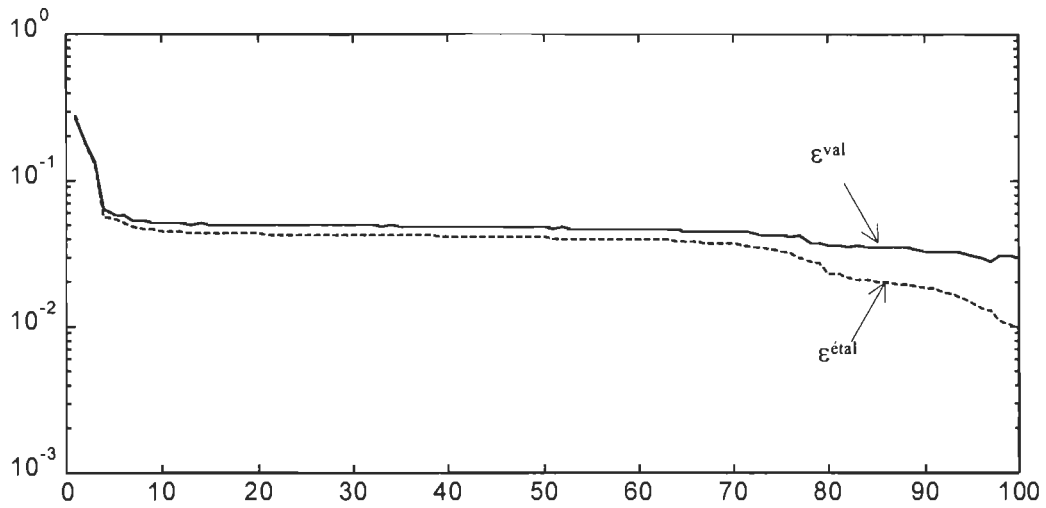


Figure 4.26: Erreurs d'apprentissage et de validation des signaux réels

Cependant, il faut noter que les réseaux ont été entraînés avec une initialisation aléatoire des poids synaptiques et que ces résultats ont été obtenus à la dixième séance d'apprentissage du réseau configuré ci-dessus. De plus, il faut noter que l'étalonnage a été réalisé sur un signal $\widetilde{y}$ fortement bruité ainsi que la généralisation.

Ces résultats montrent à suffisance que notre méthode est bien applicable dans un système de mesure réel et non linéaire. Nous n'avons pas eu besoin de chercher à identifier la réponse impulsionnelle du système nous ayant fourni ces données. La tâche la plus difficile demeure dans la détermination de la forme du signal recherchée. Mais heureusement, on en a toujours une estimation approximative car certaines marges d'erreurs sont tolérées.

D'autres résultats, aussi satisfaisants que ceux présentés précédemment, ont été obtenus avec une autre topologie de réseau. Cette dernière a m_x=5, m_y=6 et n_c=5. Les résultats d'apprentissage et de validation se présentent comme indiqués sur les figures 4.27 et 4.28.

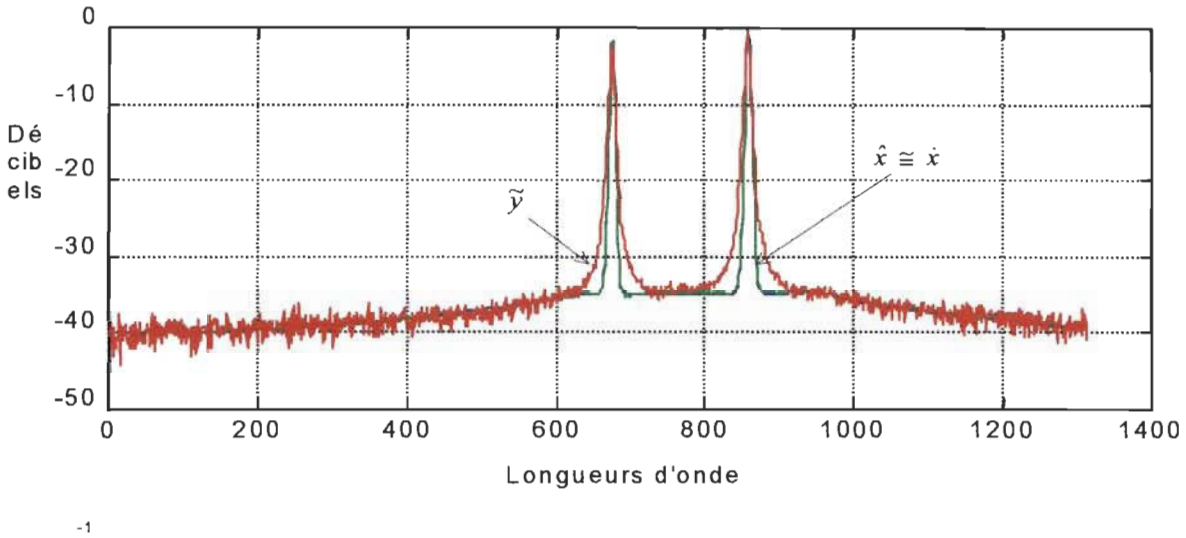


Figure 4.27 : Résultats d'apprentissage des données réelles par le réseau ayant m_x=5, m_y=6 et n_c=5.

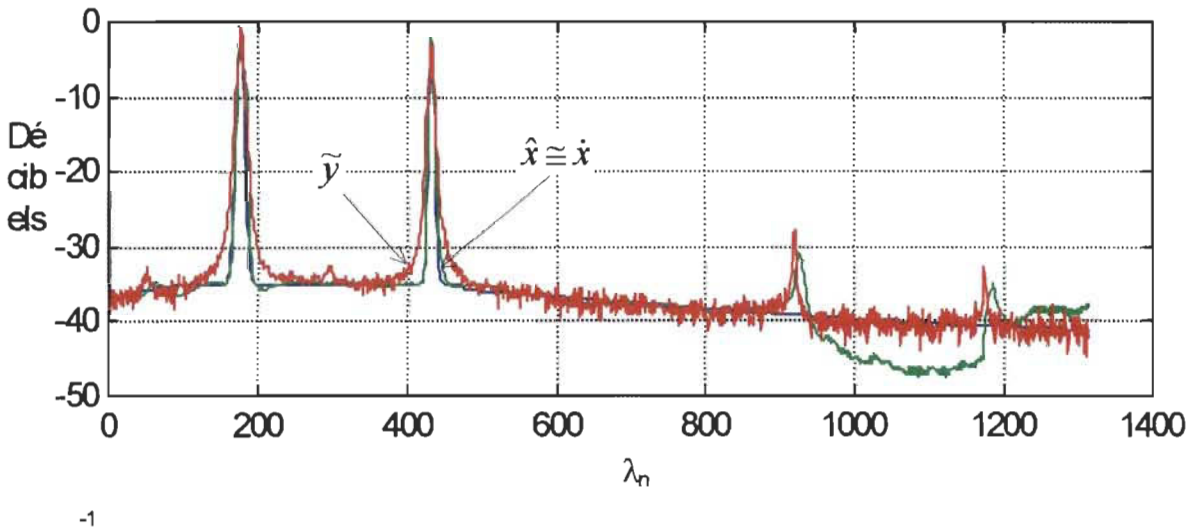


Figure 4.28 : Résultats de reconstitution du réseau m_x=5, m_y=6, n_c=5 d'un signal réel.

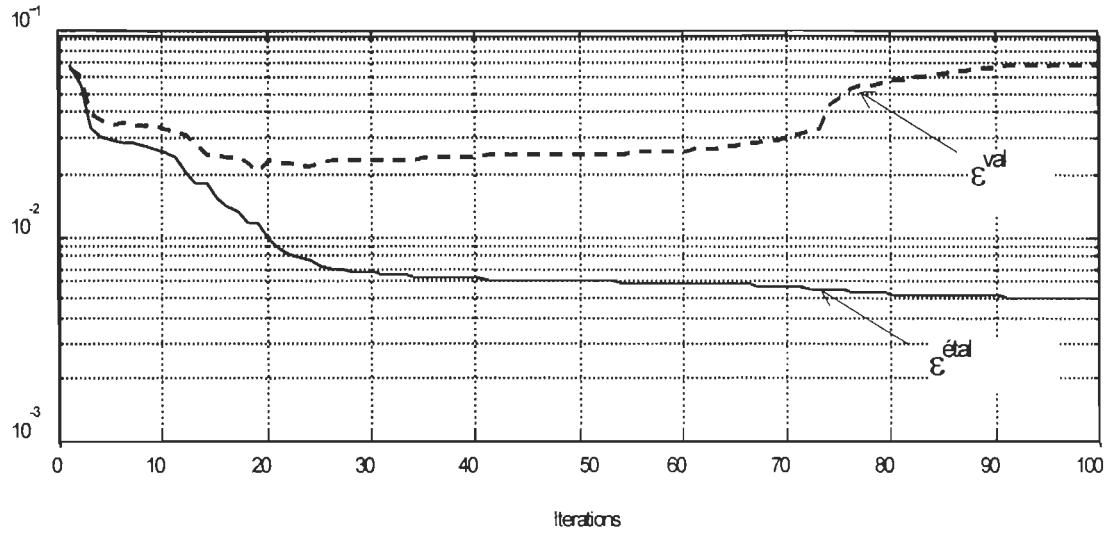


Figure 4.29: Erreurs d'apprentissage et de validation des signaux réels du réseau m_x=5, m_y=6 et n_c=5.

4.9 Conclusion

En conclusion, on voit bien que les performances des réseaux de neurones artificiels sont dépendantes de plusieurs paramètres, dont ceux de l'étape d'apprentissage. En effet, ayant remarqué que l'initialisation aléatoire des poids a beaucoup d'influence sur les capacités d'apprentissage des réseaux à un instant précis, ce facteur temporel est éliminé par leur initialisation à des valeurs fixes et identiques, facilitant ainsi l'implantation en technologie ITGE. Par la suite, il est constaté de l'apprentissage que plus on fournit d'informations sur le signal d'entrée du réseau, plus il est capable d'apprendre et surtout de

généraliser. Mais pour assurer une robustesse ou une insensibilité aux bruits d'ordre assez élevés, un certain nombre minimum de neurones sur la couche cachée doit être atteint.

CHAPITRE 5

ÉTUDE DE L'IMPLANTATION EN TECHNOLOGIE ITGE

5.1. Introduction

Dans bon nombre des systèmes de mesure, les blocs fonctionnels impliqués dans le processus de reconstitution manipulent des informations constituées par un grand nombre de données. Ces données étant sous forme de nombre, on fait souvent appel à des systèmes rapides tels que les ordinateurs ou les processeurs numériques. Ainsi, les signaux d'entrée étant analogiques, ils subissent successivement deux types de conversions : analogique-numérique (A/N) et numérique - analogique (N/A). La conversion A/N permet d'obtenir des données numériques indispensables à la compréhension du processeur, tandis que la conversion N/A se sert de la sortie du système de traitement numérique pour représenter le signal sous forme analogique.

À la sortie de la conversion A/N, le données sont représentées par un nombre fini de valeurs binaires, c'est-à-dire composées uniquement de 0 ou 1. Mais cette représentation peut prendre deux formes : à virgule fixe ou à virgule flottante. La représentation à virgule flottante est plus précise que celle à virgule fixe, mais plus lente et plus coûteuse en surface d'intégration dans un semi-conducteur. Ce processus de transformation de données analogiques en données numériques s'appelle la quantification. Le résultats de

reconstitution dépend alors du nombre de bits utilisés lors de cette opération. En effet, il a été montré que la numérisation d'un signal est affectée par une erreur de quantification liée au nombre de bits utilisés. Ainsi, plus il y a de bits , plus on réduit cette erreur et plus le résultat est précis; mais plus la surface d'intégration est importante, plus le temps de calcul augmente. Aussi, rappelons que le réseau de neurones multicouche étudié ici a été étalonné avec l'algorithme NNOE (*Neural Network Output Error*) et validé par le code *nnvalid.m* Avec des données synthétiques, il présente des capacités d'apprentissage et de généralisation satisfaisantes selon les résultats du chapitre 4.

Ce chapitre consiste donc à le numériser afin de déterminer le nombre de bits nécessaires à la représentation minimale de ses matrices des poids de connexion, de chacune de ses opérations et du signal d'entrée afin d'obtenir un signal reconstitué numérique pratiquement identique au signal estimé en virgule flottante obtenu lors de l'étalonnage du réseau.

5.2 Principes de la quantification des signaux

La quantification est en général la dernière étape, après l'échantillonnage, du processus de numérisation d'un signal. Elle peut être définie comme étant la règle de correspondance entre un signal d'entrée analogique x(t) et le signal numérique $x_q(t)$ associé. Cela permet alors de représenter un nombre quelconque en un mot binaire de b bits. Cependant, deux possibilités sont offertes pour cette représentation : soit elle est à virgule fixe, soit elle est à virgule flottante [MEF97].

5.2.1 La quantification à virgule fixe

En laissant un bit du mot pour le signe de la valeur, la représentation à virgule fixe consiste à avoir une suite de valeurs représentatives, séparés par un même intervalle, sur l'axe réel. De manière général, un ensemble de valeurs symétrique par rapport à l'origine des axes de coordonnées. Le nombre de valeurs représentables détermine le nombre de bits nécessaires à leur représentation numérique. Dans ce cas, la dynamique des signaux se définit dans l'intervalle [-1 , +1]. Les nombres quantifiés en virgule fixe peuvent être représentés par *signe et module*, ou par *complément à 2*.

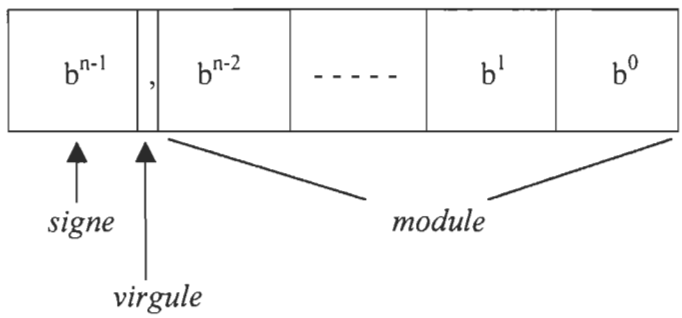


Figure 5.1 : Représentation par signe et par module.

La représentation par signe et par module est couramment utilisée. Elle consiste à écrire le nombre sur un certain nombre de bits dont un sera réservé pour le signe. Ainsi, si on a n bits dans le mot, (n-1) bits seront utilisés pour le module. Cette représentation est illustrée sur la figure 5.1.

Soit x un nombre réel, sa représentation binaire par signe et par module sera :

$$x_q = (-1)^{C_0} \sum C_i 2^{-i} \tag{5.1}$$

Dans ce genre de représentation, la dynamique est définie par $[-1+2^{-(n-1)}, 1-2^{-(n-1)}]$. Le pas de quantification q définissant l'espace entre deux valeurs quantifiées voisines, dépend du nombre de bits n et des extrémums du signal x à quantifier. Il peut s'exprimer par la formule:

$$q = \frac{x_{max} - x_{min}}{2^n - 2} \tag{5.2}$$

La représentation par complément à 2 est beaucoup plus utilisée pour une numérisation des signaux de type bipolaires; car les opérations arithmétiques impliquant des nombres négatifs sont beaucoup plus faciles à exécuter. Si X est un nombre positif, son opposé en complément à 2 est définie par $X' = 2^n - X$.

Différemment de la représentation par signe et module, la dynamique pour la représentation par complément à 2 est $[-1,1-2^{-(n-1)}]$ et le pas de quantification q sera :

$$q = \frac{x_{max} - x_{min}}{2^n - 1} \tag{5.3}$$

La principale différence entre ces deux représentation est que la valeur 0 est représentée deux fois dans la première, tandis qu'elle ne l'est que d'une façon unique dans la deuxième. Pour avoir une meilleure quantification, l'erreur entre la valeur représentable et la valeur donnée soit la plus petite possible. En faisant cette étude à l'intérieur de la dynamique du signal, on a affaire à une quantification par arrondi.

Pour une quantification par troncature, on cherche à ce que le module de la valeur la plus proche de la valeur représentable soit inférieur à celui de la donnée. C'est une troncature par module.

Lors de la quantification par troncature et par complément à 2 diffère de la précédente seulement dans le cas des nombres négatifs. Une valeur négative est représentée par la valeur la plus proche dont le module est supérieur à celui de la donnée.

Sur les figures 5.2, on illustre les caractéristiques et les erreurs des différentes sortes de quantifications décrites précédemment.

En résumé , on peut dire que pour effectuer une quantification , il faut d'abord définir sa loi. Ainsi, on devra spécifier [KUN91]:

- le type de représentation (virgule fixe ou virgule flottante);
- la dynamique et le pas de quantification;
- le nombre de bits de représentation;
- le type de représentation des nombre négatifs (signe – module ou complément à 2);
- la loi de quantification à l'intérieur de la dynamique (arrondi ou troncature);
- la loi de dépassement (saturation ou modulaire).

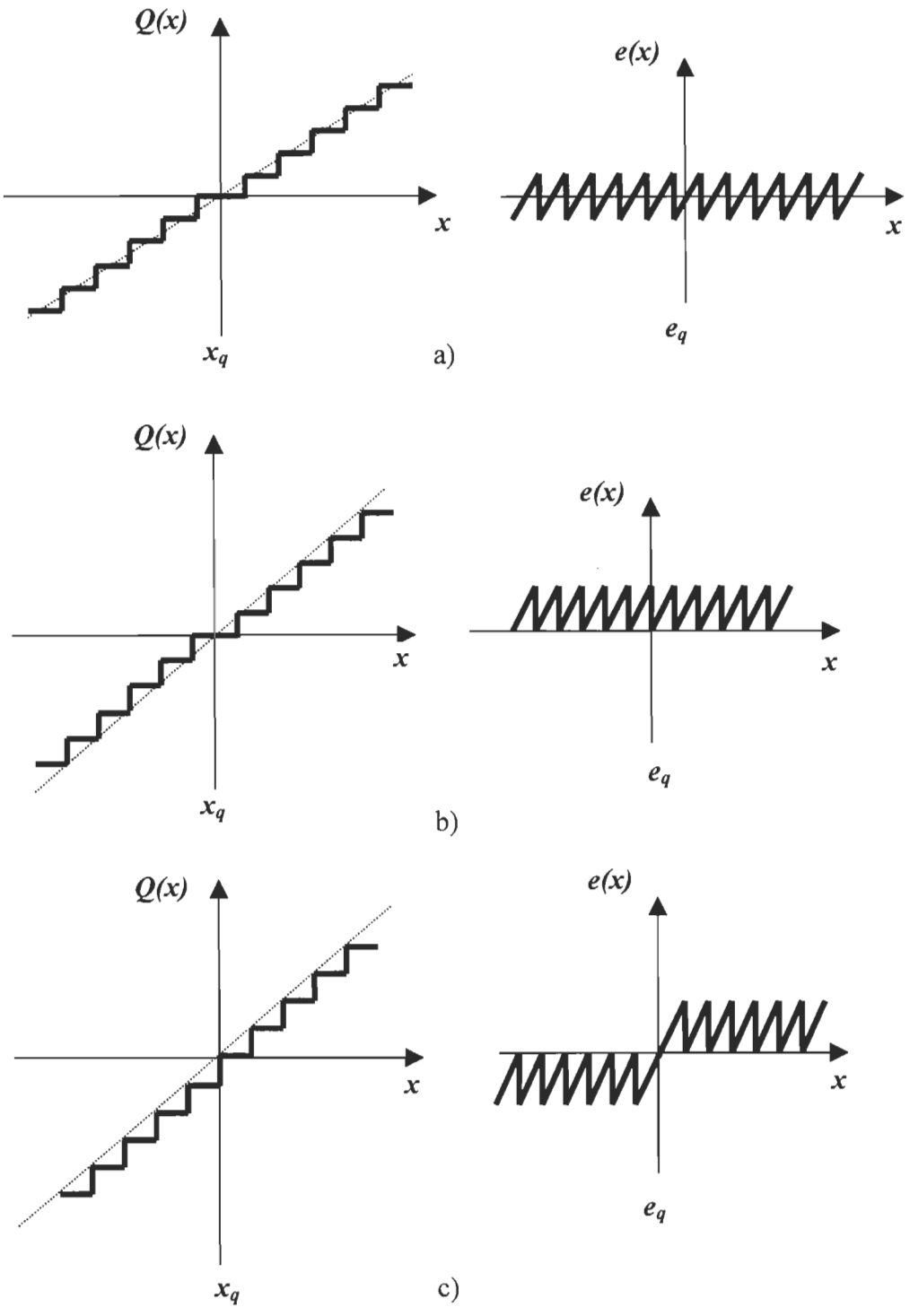


Figure 5.2 : Caractéristiques de quantification par arrondi (a), par troncature de module(b) et par troncature par complément à 2 (c)

5.2.2 La quantification à virgule flottante

La quantification à virgule flottante est la moins utilisée car très coûteuse à intégrer. En effet, elle est comparable à une quantification à virgule fixe mais pour un nombre infini de bits de représentation. On devra représenter à la fois la mantisse et l'exposant en spécifiant un nombre de bits pour chaque partie. Pour ces travaux, nous allons utiliser la quantification à virgule fixe.

5.3 Quantification du réseau de neurones

Le procédé de quantification du réseau de neurones consiste simplement à se baser sur la procédure *nnvalid.m*, et à numériser les différents paramètres et toutes les opérations qui y sont effectuées pour obtenir un estimé numérique du signal idéal $\dot{x}$. Des quatre types de quantifications basés sur la quantification à virgule fixe, à savoir : la quantification arrondi par complément à 2, la quantification par troncature par complément à 2, la quantification arrondi par signe et par module, et la quantification par troncature par signe et par module, on va utiliser la première pour bâtir notre étude sachant que celle-ci nous semble préférable pour une intégration en technologie ITGE.

Le code généré en Matlab^{TM} réalisant cette numérisation figure à l'annexe C du présent mémoire. L'appel de cette fonction nécessite cependant la connaissance des types d'arguments à lui fournir. Il est donc recommandé de :

- normaliser le signal à numériser;
- choisir le nombre de bits de quantification;
- leur type (signé ou non).

Inspiré du code *nnvalid.m*, le code *nnvalidq.m* correspondant au procédé de quantification a été élaboré. Ainsi, après les étapes d'initialisation des paramètres et de la construction de la matrice de régression, les différentes opérations suivantes sont effectuées :

a- Numérisation des signaux et des matrices des poids

b- Construction de la matrice de régression φ_q

c- Calcul de la sortie estimée $\hat{x}_q$ du réseau de neurones quantifié

d- Calcul de l'erreur e_q entre les signaux estimé et l'idéal

Les codes en Matlab^{TM} des fonctions exécutées dans ce processus figurent à l'annexe C de ce mémoire.

5.4 Résultats sur la quantification du réseau de neurones

Les résultats numériques obtenus après la quantification du réseau de neurones sont réalisés avec l'estimation du signal d'apprentissage fourni par le réseau en virgule flottante, que nous noterons $\hat{x}_\infty$. Les performances de reconstitution du réseau quantifié sont vérifiées en comparant le signal issu de ce réseau à celui du réseau en virgule flottante. Cette comparaison est faite en calculant l'erreur relative moyenne entre ces deux signaux. Ainsi, sachant que le signal d'entrée du réseau et les matrices $\mathbf{W}_1$ et $\mathbf{W}_2$ des poids de connexion sont les principales données à quantifier, on étudie l'effet de quantification en faisant varier le nombre de bits de représentation de 8 à 32 bits sur chacun de ces paramètres. Le tableau 5.1 montre les différentes erreurs relatives moyennes entre les signaux issus des deux réseaux en fonction de la variation du nombre de bits de représentation du signal et des matrices des poids.

Tableau 5.1: Erreurs relatives moyennes entre $\hat{x}_\infty$ et $\hat{x}_q$ en fonction du nombre de bits de quantification.

Bits sur les signaux	Nombre de bits sur les matrices $\mathbf{W}_1$ et $\mathbf{W}_2$						
	8	12	16	20	24	28	32
8	0.69994	0.5933	0.58175	0.6582	0.69649	0.69649	0.69649
12	0.40545	0.1265	0.07624	0.06649	0.07553	0.07233	0.07776
16	0.4086	0.05021	0.00767	0.0044	0.00432	0.00566	0.00456
20	0.4084	0.05134	0.004	0.00035	0.00028	0.00019	0.0002
24	0.40841	0.05124	0.00382	0.00015	3.77E-5	3.84E-5	2.98E-5
28	0.4084	0.05126	0.00385	0.00016	9.55E-6	1.05E-6	8.94E-7
32	0.4084	0.05126	0.00385	1.62E-4	8.67E-6	5.76E-7	1.62E-7

Les courbes obtenues à partir de ces valeurs (figure 5.3) permettent de mieux voir les performances des différentes quantifications. On y voit aisément que les performances de généralisations du réseau quantifié s'approchent de celles du réseau en virgule flottante au fur et à mesure que l'on augmente le nombre de bits de quantification du signal d'entrée et des matrices des poids. En effet, l'erreur relative entre les signaux issus de ces deux réseaux diminuent considérablement avec la croissance du nombre de bits.

Pour des raisons d'implantation réalistes, le réseau quantifié à 16 bits sur le signal et sur les matrices des poids convient parfaitement aux spécifications technologiques actuelles pour une intégration en technologie VLSI.

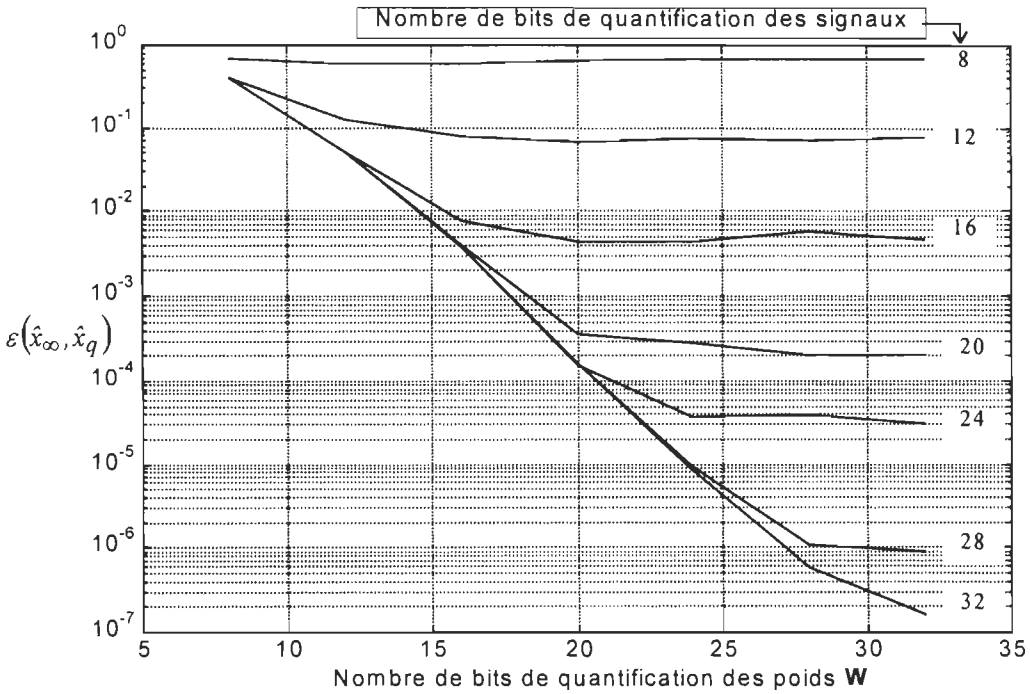


Figure 5.3 : Erreurs de quantification en fonction du nombre de bits sur les signaux et sur les poids de connexion

La comparaison de sa sortie et celle du réseau non quantifié est illustré sur la figure 5.4. tandis que la figure 5.5 montre la progression de l'erreur de quantification.

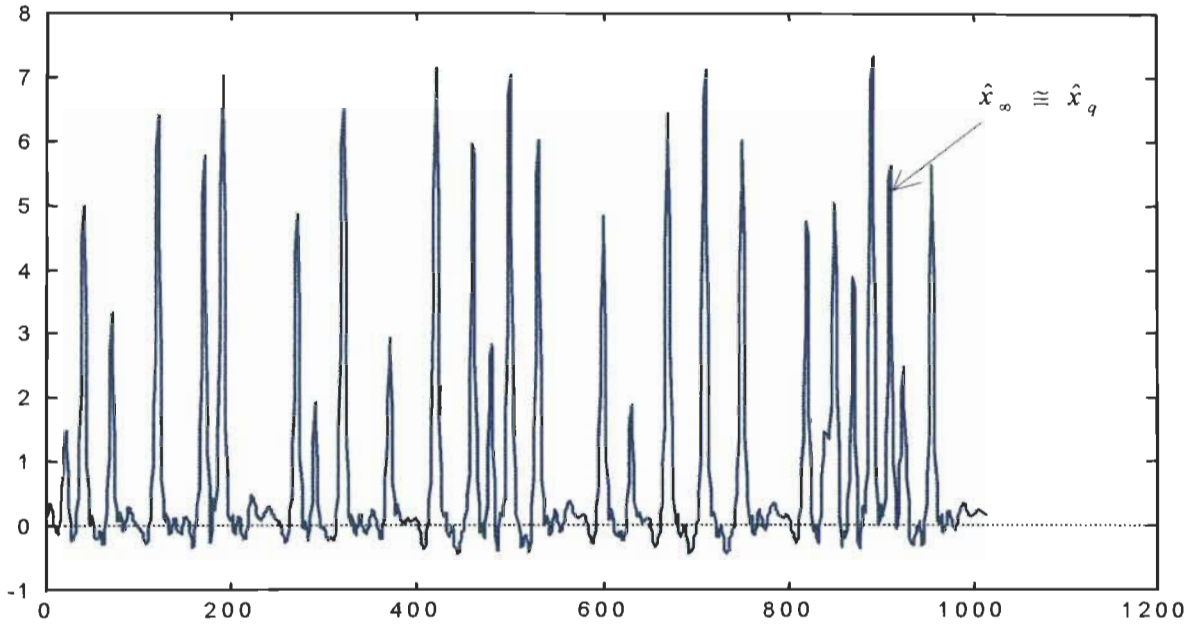


Figure 5.4 : Signaux estimés $\hat{x}_\infty$ et $\hat{x}_q$ avec 16 bits pour m_x=10, m_y=10, n_c=5, σ^2_n=0.

$$\varepsilon(\hat{x}_\infty,\hat{x}_q)= 0.0077,\ \varepsilon(\dot{x},\hat{x}_q)= 0.093, \varepsilon(\dot{x},\hat{x}_\infty)= 0.085.$$

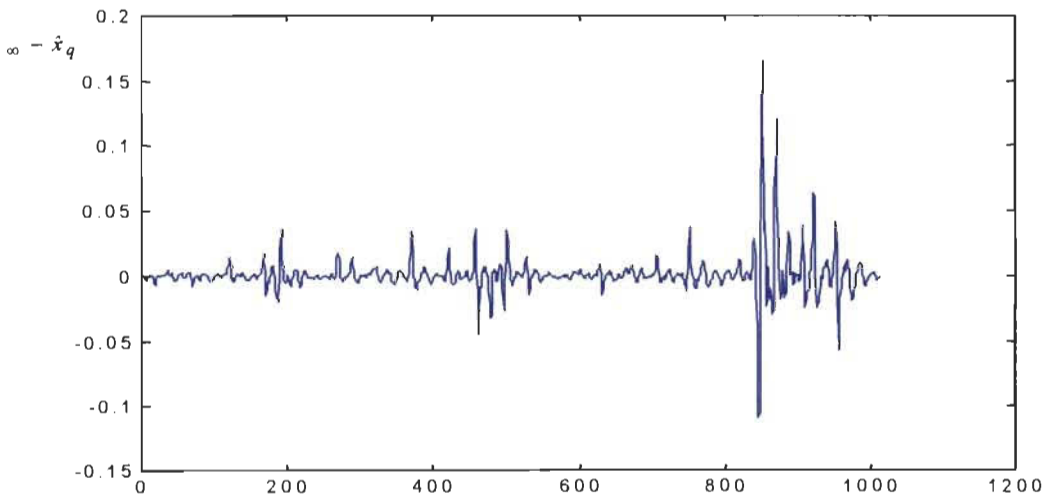


Figure 5.5 : Erreur de quantification sur les signaux $\hat{x}_\infty$ et $\hat{x}_q$ pour 16 bits.

Cependant, il faudra noter que l'obtention de ces résultats ne s'est pas faite sans obstacles. En effet, après s'être assuré de la normalisation des paramètres à numériser, il fallait détecter à quel moment la dénormalisation était requise afin d'obtenir des résultats de reconstitution non erronés. Pour contourner ce problème, on a donc décidé en premier lieu de choisir un même facteur de normalisation des variables : ce facteur est pris comme un multiple de deux. Puis, en deuxième lieu, il a fallu exécuter l'algorithme pas à pas pour déceler les entrées et les sorties de chaque couche du réseau afin de déterminer à quel niveau du procédé la phase de dénormalisation devait intervenir.

5.5- Spécifications architecturales

L'implantation en technologie ITGE d'un réseau de neurones requiert plusieurs opérations préliminaires. Ainsi, en se basant sur le modèle mathématique représentant les neurones utilisés dans un réseau quelconque, on peut réaliser la mise en œuvre dans un processeur [VID98a]. Pour les neurones utilisés dans ces travaux, ceux appartenant aux couches d'entrée et de sortie ont des fonctions de décision linéaires, tandis que ceux de la couche cachée ont une fonction de décision non linéaire de type tangente hyperbolique. En tenant compte des matrices des poids de connexion, la réalisation du modèle mathématique d'un neurone caché se résume donc à un produit matrice vecteur en passant par sa fonction de décision. Le traitement global à travers un processeur est illustré par la figure 5.6

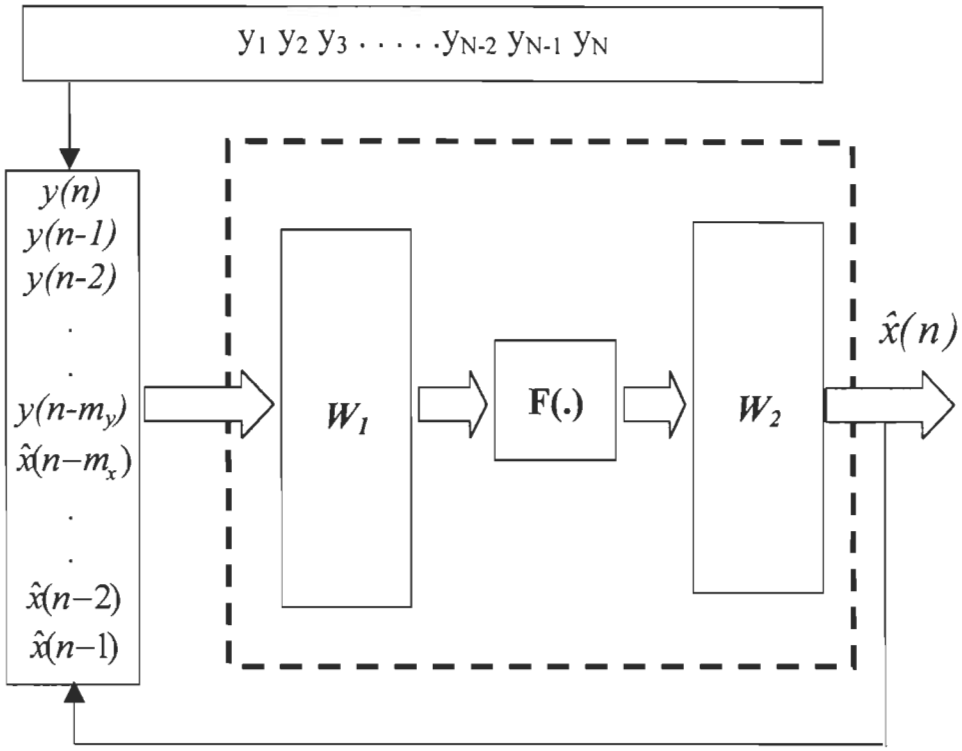


Figure 5.6 : Schéma général du traitement des données à travers un processeur

Dans ce processeur [LAS97], le réseau de neurones sera représenté par :

- un registre d'entrée qui recevra les valeurs numériques du signal de sortie de conversion et celles du signal estimé par le réseau. La dimension de ces valeurs est connue d'avance - grâce aux nombres m_x et m_y correspondant au nombre de retards sur l'estimé et le signal d'entrée obtenus lors de l'étalonnage du réseau.

- des registres dans lesquelles les éléments des matrices des poids de pondération ($\mathbf{W}_1$ et $\mathbf{W}_2$) seront stockées.

- une cellule représentant le modèle mathématique de la fonction de décision non linéaire. Cette cellule peut pourra jouer le rôle d'un ALU.

- un registre de sortie qui contiendra les valeurs successivement estimée par le réseau de neurones

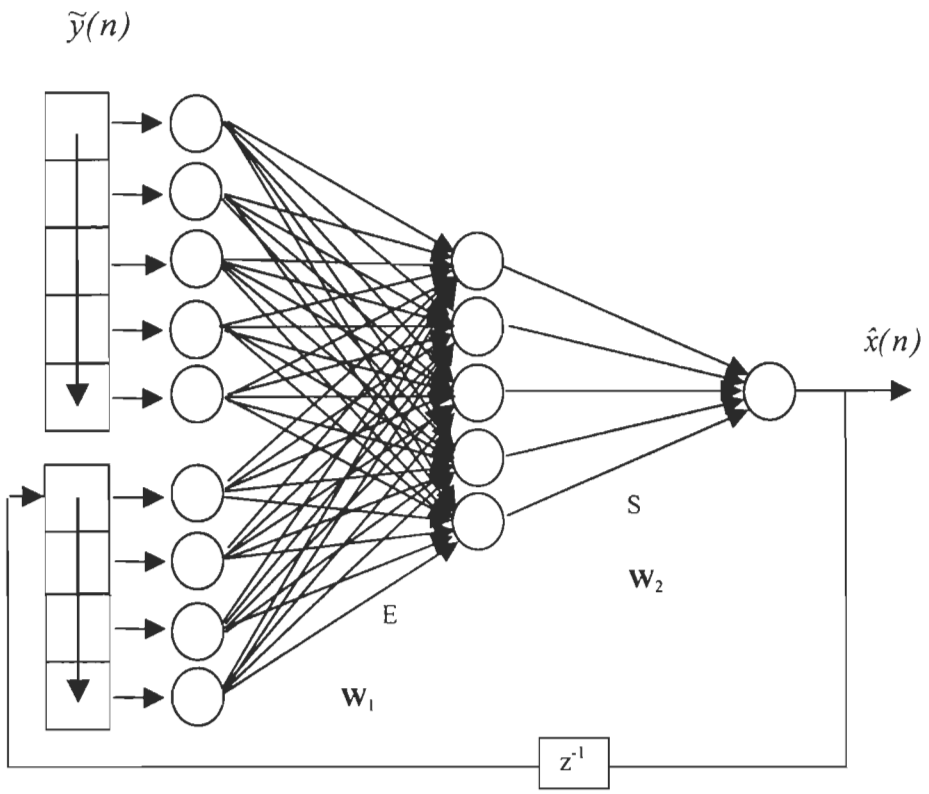


Figure 5.7: Schéma du RNA à modéliser avec $m_x=m_y=4$ et $n_c=5$.

La sortie du réseau de neurones peut être dirigée vers un organe de prise de décision.

Le choix de l'architecture dans laquelle notre réseau va être implanté ne peut être fait si l'on ne comprend pas le déroulement des opérations à effectuer. Ainsi, on va d'abord essayer de modéliser l'un des réseaux de neurones utilisées pour l'application aux données expérimentales (section 4.5). On prendra celui dont $m_x=m_y= 4$ et $n_c=5$ et sa modélisation est illustrée sur la figure 5.7.

Les principales opérations qui ont lieu dans le fonctionnement du réseau de neurones sont les produits, les sommes et la fonction de décision tangente hyperbolique. Ainsi, à chaque neurone d'entrée linéaire, la sortie est multipliée par le poids correspondant dans la matrice **W**1 pour ensuite entrer dans la couche cachée. Soient U (dim(U) = 9x1) le vecteur des neuf entrées du réseau et **W**1 (dim(**W**1) = 5x9) la matrice des poids d'entrée à la couche cachée, les entrées E_i de chacun des neurones de cette couche seront des sommes de produits telles que:

$$
\begin{aligned}
E_1 &= U_1 W1_{11} + U_2 W1_{2i} + \ldots + U_9 W1_{91} \\
E_2 &= U_1 W1_{12} + U_2 W1_{22} + \ldots + U_9 W1_{92} \\
E_3 &= U_1 W1_{13} + U_2 W1_{23} + \ldots + U_9 W1_{93} \\
E_4 &= U_1 W1_{14} + U_2 W1_{24} + \ldots + U_9 W1_{94} \\
E_5 &= U_1 W1_{15} + U_2 W1_{25} + \ldots + U_9 W1_{95}
\end{aligned}
\tag{5.4}
$$

D'une façon générale, ces entrées sont traduites par l'équation

$$
E_i = \sum U_j W1_{ji} \text{ avec i=1,...,5 et j=1,...,9} \tag{5.5}
$$

Le produit matrice vecteur représentant alors ce processus est comme suit.

$$\begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \\ E_5 \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ . \\ . \\ U_9 \end{bmatrix} \text{ X } \begin{bmatrix} W1_{11} & W1_{21} & & W1_{91} \\ W1_{12} & W1_{22} & & W1_{92} \\ W1_{13} & W1_{23} & \ldots & W1_{93} \\ W1_{14} & W1_{24} & & W1_{94} \\ W1_{15} & W1_{25} & & W1_{95} \end{bmatrix} \tag{5.6}$$

[E] [U] [W1]

Puis, une fois le vecteur d'entrée **E** de la couche cachée obtenu, ses éléments sont chacun passés à travers de la fonction de décision *pmntanh* caractérisant ces neurones. Ce qui nous donne la sortie **S** de cette couche. Cette sortie devient l'entrée du neurone de sortie en se voyant ses éléments être multipliés par les poids contenus dans la matrice **W**2. On obtiendra un scalaire comme résultat de sortie du réseau. Le neurone de sortie ayant une fonction de décision linéaire, le scalaire $\hat{x}$ obtenu sera simplement une somme de produits **S W**2 telle que :

$$\hat{x} = \begin{bmatrix} S_1 & S_2 & S_3 & S_4 & S_5 \end{bmatrix} \begin{bmatrix} W2_1 \\ W2_2 \\ W2_3 \\ W2_4 \\ W2_5 \end{bmatrix} \tag{5.7}$$

ou encore

$$\hat{x} = S_1 W2_1 + S_2 W2_2 + \ldots + S_5 W2_5 \tag{5.8}$$

Comme on peut le remarquer dans le fonctionnement de cet algorithme, la principale difficulté se trouve lors du passage de l'entrée E dans la fonction de décision pmntanh de chaque neurone caché pour obtenir la sortie **S**.

Mais, auparavant, il faut remarquer que les résultats présentés dans la section 5.4 ont été obtenus à l'aide du logiciel Matlab^{TM} car ce dernier est capable de générer la fonction *pmntanh* (tangente hyperbolique). Ainsi, dans notre algorithme de quantification, seules les données d'entrée et de sortie de cette fonction sont numérisées. Mais, pour une implantation en VLSI, on aura besoin de placer cette fonction de décision dans le processeur afin que les calculs soient faits en virgule fixe.

De toutes les stratégies envisageables pour représenter cette fonction de décision, celle de l'élaboration d'une table de correspondance, ou 'look-up table', à partir de la caractéristique de la fonction de décision a été choisie [VID98b]. Cette table consiste à mémoriser un nombre fini de points de la tangente hyperbolique. Dans sa lecture, les bits les plus significatifs du point à évaluer sont assignés à l'adresse de la mémoire où se trouve la valeur de la fonction. Pour des raisons d'optimisation, cette table sera codée avec un nombre de bits minimale afin de réduire la taille de la mémoire et donc, la surface d'intégration. Bien que la précision en soit affectée, la surface d'intégration ne sera pas assez coûteuse. Une façon de calculer toutes les valeurs de la fonction est de procéder par interpolations entre les points de la table. Ainsi, entre deux points P1 et P2, la pente permettant l'interpolation se calcule par :

$$a = \frac{P2 - P1}{\Delta X} \tag{5.9}$$

où ΔX est la distance séparant le deux points sur l'axe des abscisses.

En considérant P1 comme la valeur de la case mémoire correspondant au point à évaluer et P2 la valeur suivante immédiate, on se propose donc de précalculer les pentes et de les garder dans une mémoire. Le tableau suivant nous montre un exemple de structure d'une look-up table.

Tableau 5.2 : Structure d'une table de correspondance.

Adresse	**Valeur du point**	**Pente**
0…	0…0	1…
…	…	…
F…	11…	00…

Après donc avoir généré cette table, une requantification du réseau de neurones est requise afin de pouvoir utiliser la fonction de décision numérisée.

Pour l'implantation de notre réseau, une architecture linéaire a été choisie [VID98b]. Comme indiqué sur la figure 5.8, cette architecture fait un produit matrice – vecteur. Elle a pour entrées deux piles contenant respectivement les entrées bruitées $\tilde{Y}$ et les sorties estimées $\hat{X}$. Les piles formant les matrices **W**1 et **W**2 contiennent les poids et les seuils.

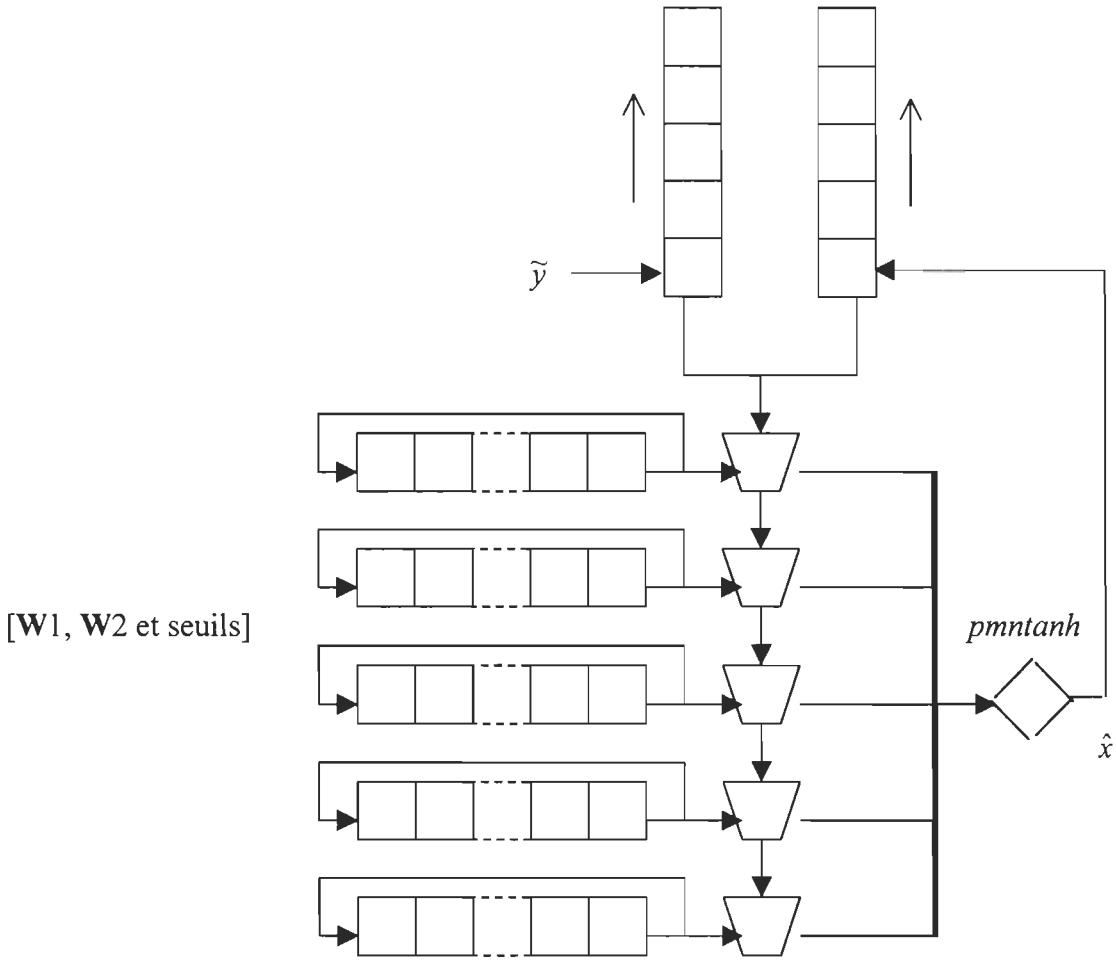


Figure 5.8 : Architecture linéaire pour l'implantation du réseau de neurones NNOE

Dans son fonctionnement, la première étape consiste à insérer les échantillons du signal bruité dans la pile correspondante en fonction du nombre de retards exigé ; la pile destinée aux valeurs de l'estimé étant d'abord remplie des 0 (zéros). Puis, les processeurs élémentaires effectuent un à la fois les produits matrice – vecteur représentés par le système d'équations (5.6). Les éléments d'une matrice de poids se présentent un à la fois à l'entrée du processeur élémentaire correspondant puis, ils sont ensuite retourné au fond de cette

pile. Les cinq processeurs élémentaires représentent les cinq neurones de la couche cachée et effectuent toutes les opérations permettant l'obtention des vecteurs E, S et du scalaire de sortie. Étant donné qu'on utilise des retards sur les entrées, certaines valeurs sont donc gardées en mémoire pendant le processus. Cela se manifeste par un décalage dans la pile pour permettre l'entrée d'une nouvelle valeur calculée ou bruitée. Autrement dit, chaque nouvelle valeur introduite dans chacune des piles d'entrée entraîne l'écrasement de celle qui se trouve à l'extrême haut de la pile.

Pour estimer les performances de cette architecture, nous avons besoin d'évaluer la latence et le débit induits au cours de son fonctionnement. La latence traduit le nombre de cycles nécessaires pour obtenir un résultat, tandis que le débit est la fréquence de sorite des résultats. Ainsi, pour notre réseau de neurones, soient M le nombre d'entrées et n_c le nombre de neurones sur la couche cachée, il faudra $M + n_c + 2$ cycles pour avoir un résultat à la sortie. En effet, la première somme de produits est obtenue après $M + 1$ cycles. Puis, il faudra $n_c - 1$ cycles pour avoir tous les résultats de sortie de la couche cachée. Enfin, la fonction de décision est exécutée en un cycle et le chargement des nouvelles données est effectué en un cycle aussi. La structure de notre architecture étant linéaire, le débit est égal à la latence.

Sur le marché actuel, il existe plusieurs types de processeurs dans lesquels on peut réaliser l'implantation du réseau de neurones. Ainsi, selon les spécifications du cahier de charges, on peut se servir : d'un micro-contrôleur, d'un DSP (Digital Signal Processor), d'un circuit à applications spécifiques (ASIC : Application Specific Integrated Circuit) ou d'un FPGA (Field Programmable Gate Array). Chacune de ces composantes peut réaliser des

opérations en virgule fixe ou flottante. Mais un compromis devra être trouvé car l'usage de la virgule flottante permet l'obtention d'une grande précision. Cela implique un temps d'exécution assez long dû au nombre d'opérations considérables. D'un autre coté, les calculs en virgule fixe sont plus courts mais la précision est quelque peu affectée.

Dans le cas des calculs à effectuer par notre processeur, ils sont de trois types. D'abords, on a les multiplications de chaque entrée par les poids de connexion correspondants. Puis, pour chaque neurone de la couche cachée, on effectue la sommation de tous les produits précédents et la réalisation de la fonction de décision non linéaire. Pour plus de détails, différentes architectures hautement parallèles (systoliques) ont été proposées dans [VID98a] et [VID98b] pour l'algorithme étudié dans le présent travail.

5.6 Conclusion

En conclusion, on peut dire que le réseau de neurones quantifié aura besoin d'au maximum 16 bits sur chaque paramètre de quantification pour avoir une bonne reconstitution. Aussi, pour optimiser le réseau quantifié, on pourra chercher le nombre de bits optimal pour représenter les matrices de poids de connexion. En effet, c'est surtout ces matrices qui exigent de la mémoire par le fait qu'elles sont stockées. Une solution acceptable serait d'avoir un nombre identique de bits de quantification sur les signaux et les matrices des poids synaptiques.

CHAPITRE 6

CONCLUSION

Dans ces travaux, l'objectif principal était de démontrer l'applicabilité des réseaux de neurones dans le processus de reconstitution des signaux pour les systèmes dynamiques linéaires et non linéaires. Étant dans le domaine du traitement du signal, on entre dans un autre champ qui est la *mesure*. C'est en effet dans ce dernier que la problématique reliée à la reconstitution des signaux est généralement rencontrée; et nous avons cherché à démontrer que les méthodes neuronales étaient une alternative aux méthodes analytiques encore utilisées pour résoudre ce problème.

Au chapitre 2, on fait un rappel de certaines notions reliées aux systèmes de mesure. On décrit ainsi un système de mesure comme ayant deux fonctions principales : la conversion et la reconstitution d'une grandeur mesurée. Dans le cadre de ce travail, c'est la deuxième fonction qui nous intéresse. Étant souvent effectuée à l'aide des méthodes analytiques, nous avons montré que cette tâche peut être effectuée par les réseaux de neurones. Cette alternative permet donc d'éviter le processus d'identification de la réponse impulsionnelle du capteur, par opposition aux méthodes analytiques qui en ont besoin. Aussi, ayant remarqué que le mesurande est le plus souvent dégradé par la non linéarité et

le bruit lors du processus de conversion en signal électrique, les réseaux de neurones demeurent encore adéquats pour ce genre de signaux car ils présentent une grande robustesse face à la non linéarité et au bruit. Alors, ils peuvent travailler en milieu perturbé. Ainsi, pour n'importe quelle forme de signal issue de la conversion ou de la reconstitution analytique, il nous suffit de connaître la forme recherchée afin de bâtir une banque de données de référence pour construire le réseau de neurones.

Ainsi, avant de commencer l'étude proprement dite, le troisième chapitre de ce travail consiste à faire un aperçu des réseaux de neurones artificiels dynamiques et leurs caractéristiques. Partant du fait que le modèle mathématique d'un neurone formel est basé sur les principes de fonctionnement de son homologue biologique, la sortie du neurone artificiel peut se résumer comme étant la somme pondérée de ses entrées passées à travers une fonction d'activation non linéaire. Dépendemment du type de problème à résoudre, il existe un vaste choix de fonctions d'activation. La référence essentielle des recherches en intelligence artificielle étant le fonctionnement du cerveau humain, plusieurs architectures de réseaux de neurones artificiels ont été élaborées. Leurs principales différences sont situées au niveau du type de connexion entre les neurones, le nombre de couches du réseau, la fonction d'activation. Leurs principes de fonctionnement étant alors différents, les algorithmes d'entraînement spécifiques sont développées pour chaque modèle de réseau. Cet entraînement consiste à ajuster les poids synaptiques du réseau à partir d'un mode d'apprentissage bien choisi. Dans ces travaux, le nôtre est de type supervisé.

Aussi, après avoir présenté, les algorithmes dédiés à la reconstitution des signaux pour les systèmes dynamiques non linéaires, on choisira celui basé sur le modèle NNOE (Neural Network Output Error) pour effectuer notre étude, soit un cas particulier du modèle NNARX. Ce choix s'explique par l'application des données d'entrée en tenant compte du temps, le principe d'ajustement des matrices des poids de connexion et par la récurrence du réseau. L'algorithme calcule l'erreur entre la sortie du réseau et celle désirée pour modifier les poids. Mais lors de l'évaluation, seul le signal de sortie de conversion est requis; le signal idéal étant utilisé pour des fins de comparaison. En somme, on dira que la force de cet algorithme est de pouvoir faire une estimation à partir uniquement du signal brut, et pas accompagné du signal recherché. Mais le succès de cette opération, comme pour tout algorithme, réside dans la qualité et la pertinence des données utilisées lors de l'étalonnage.

Au chapitre 4, une application de cet algorithme est alors faite sur un système de mesure de type spectrométrique. Après avoir généré nos banques de données synthétiques, on étalonne différentes topologies de réseaux. Nous avons montré que l'augmentation du nombre de neurones sur la couche cachée, ou du nombre de prélèvements sur les signaux, n'implique pas une amélioration des performances des réseaux de neurones. Cependant, on peut établir que qu'il existe un nombre d'informations minimales à extraire du signal d'entrée du réseau pour que ses capacités de généralisation ou d'interpolation soient acceptables. Aussi, la robustesse du réseau est atteinte avec un nombre optimal de neurones sur la couche cachée. En comparant les résultats fournis par nos réseaux de neurones à ceux obtenus avec d'autres méthodes populaires, on voit qu'ils sont semblables, sinon meilleurs

dans certains cas. De plus, la complexité algorithmique d'un réseau de neurone est de moindre importance que celle des autres méthodes; ce qui favorise une implantation en technologie ITGE. Notre satisfaction a été encore plus grande lorsqu'avec certaines données expérimentales, les réseaux ont présenté d'excellentes aptitudes de généralisation.

La dernière étape de notre étude, chapitre 5, a consisté à voir les possibilités d'implantation de nos réseaux de neurones en technologie ITGE. Étant donnée que cette technologie utilise des données numériques, ce étude revient essentiellement à estimer de façon précise la quantité de mémoire qu'occupera notre réseau. Ceci est rendu possible grâce à une quantification du réseau et à l'étude de son effet sur les résultats de sortie du réseau. Ainsi, après avoir survolé les différents types de quantifications, on choisit la quantification en virgule fixe par arrondi. Cette dernière, effectuée sur les signaux, la fonction d'activation et les matrices de poids, révèle que plus on a de bits de quantification, plus l'erreur de quantification diminue. Mais conformément aux cahiers de charges de bon nombre de processeurs et vu l'erreur acceptable entre le signal estimé en virgule flottante et quantifié, nous choisissons une représentation 16 bits par 16 bits pour notre réseau. Le choix d'un nombre plus grand améliorera certes la précision mais accroîtra le temps de calcul.

Pour conclure ce travail, une remarque importante devrait être faite quant à l'obtention des résultats présentés dans ce rapport. En effet, lors de l'entraînement des réseaux, certains paramètres d'apprentissage on été fixés ou délimités de façon arbitraire.

Nous citerons principalement l'initialisation des matrices des poids à des valeurs fixes. Une initialisation aléatoire est recommandé mais nous avons fait ce choix pour pouvoir répéter les entraînements. Cependant, les résultats obtenus avec les données expérimentales proviennent des réseaux de neurones entraînés avec une initialisation aléatoire de ces matrices.

Comme recommandations finales, des études futures des aptitudes de l'algorithme peuvent être faites et on pourra chercher à établir :

- le nombre de données nécessaires pour garantir un bon entraînement des réseaux de neurones;
- l'effet de l'apprentissage avec des signaux bruitées sur les performances des réseaux;
- une contrainte de positivité sur la sortie estimée du réseau à l'apprentissage et/ou à la validation.

RÉFÉRENCES

[ALI96] C. Alippi et V. Piuri, '' Experimental Neural Networks for Prediction and Identification'', IEEE Trans. on Instr. and Meas., Vol. 45, No. 2, Avril 1996, pp 670-676.

[ARR98] W. Arrouy, Les Réseaux de Neurones, http://www.worldnet.net/~willy/ neural/neutm.html.

[AUS94] A. Aussem, Dynamical Reccurent Neural Networks – Towards Environment Times Series Prediction, Article à publier dans International Journal on Neural Systems, Université René Descartes, Paris, 1995

[AUS95] A. Aussem, Théorie et Application des Réseaux de Neurones Récurrents et Dynamiques à la Prédiction, à la Modélisation et au Contrôle Adaptatif des Processus Dynamiques., Thèse de Doctorat, Université Réné Descartes – Paris V, Juin 1995.

[CRI91] P. B. Crilly, '' A Quantitative Evaluation of Various Iterative Deconvolution Algorithms '', IEEE Trans. on Inst. and Meas., Vol. 40, No. 3, juin 1991, pp 558-562.

[CHE96] C. H. Chen, Fuzzy Logic And Neural Network Handbook, IEEE Press, Mc Graw Hill, 1996.

[GLA87] F. H. Glanz, and W. T. Miller, '' Deconvolution using a CMAC Neural Network '', First Annual Conference on International Neural Network Society, 1987, pp. 294-298.

[HAY98] S. Haykin, Neural Networks, A Comprehensive Foundation, 2nd Edition, Prentice Hall, 1998.

[JAN98] P. A. Jansson, '' Deconvolution with Application in Spectroscopy '', 2nd Edition, Academic Press, 1998.

[KUN91] M. Kunt, Techniques modernes de traitement numérique des signaux, Presses polytechniques et universitaires romandes, traitement de l'information, Vol. 1, 1991.

[LAS96] A. Lassonde, Réseau de neurones pour la commande d'un convertisseur CC, Rapport de génie électrique, UQTR, 1996.

[LEG94] S. Legendre, Étude de l'application des réseaux de neurones multicouches pour l'étalonnage et la reconstitution de mesurandes dans un système de mesure, Mémoire de maîtrise, Université du Québec à Trois-Rivières, 1994.

[MAS95a] D. Massicotte, Une approche à l'implantation en technologie VLSI d'une classe d'algorithmes de reconstitution de signaux, Thèse de Doctorat, École Polytechnique de Montréal, 1995.

[MAS95b] D. Massicotte, R. Z. Morawski et A. Barwicz, '' Incorporation of a Positivity Constraint Into a Kalman-Filter-Based Algorithms for Correction of Spectrometric '', IEEE Trans. on Inst. and Meas., Vol. 44, No.1, février 1995, pp 2-7.

[MAS97] D. Massicotte, R. Z. Morawski et A. Barwicz, '' Kalman-Filter-Based Algorithms of Spectrometric Data Correction – Part I : An Iterative Algorithm of Deconvolution '', IEEE trans. on inst. & meas., Vol. 46, No.3, juin 1997, pp 678-684.

[MAS98a] D. Massicotte et B. MBA MEGNER, '' Spectrometric Data Correction Using Neural Network in a Dynamic Measuring System '', Int. Conf. On Engineering Applications of Neural Networks (EANN'98), Gibraltar, Espagne, 10-12 Juin 1998, pp. 338-341.

[MAS98b] D. Massicotte, S. Legendre and A. Barwicz, '' Neural-Network-Based Method of Calibration and Measurand Reconstruction For a High-Pressure Measuring System '', IEEE Trans. on Inst. and Meas., 1998.

[MAS99] D. Massicotte et B. MBA MEGNER, '' Neural-Network-Based Method of Correction in a Nonlinear Dynamic Measuring System '', Article soumis à Instrumentation and Measurement Technology Conference (IMTC '99), Venise, Italie, mai 1999.

[MEF97] A. Meftah, Application des réseaux de neurones dans le domaine de l'électronique de puissance, Mémoire de maîtrise, Université du Québec à Trois-Rivières, 1997.

[MOR94] R. Z. Morawski, " Unified Approach to Measurand Reconstruction, IEEE Trans. on Inst. and Meas., Vol. 43, No 2, Avril 1994, pp. 226-231.

[NOR95] M. Norgaard, Neural Network Based System Identification Toolbox, Université Technique du Danmark, 28 septembre, 1995.

[SIM92] P. K. Simpson, " Foundations of Neural Networks ", Artificial Neural Networks, édité par E. Sanchez-Sinencio and C. Lau, IEEE Press, 1992, pp. 3-24.

[SJO92] J. Sjoberg et L. Ljung, " Overtraining, Regularization, and Seraching for Minimum in Neural Networks", IFAC Adaptive Systems in Control and Signal Processing, Grenoble, France, 1992, pp 73-78.

[SZC95] L. Szczecinski, R. Z. Morawski and A. Barwicz, " A Cubic FIR-Type Filter for Numerical Correction of Spectrometric Data ", IEEE Trans. on Inst. and Meas., Vol. 46, No. 4, 1997, pp. 922-928.

[VID98a] M. Vidal, Modélisation en VHDL d'un réseau de neurones multicouches, Rapport interne, Groupe de Recherche en Électronique Industrielle, UQTR. 1998.

[VID98b] M. Vidal, L'égalisation des canaux par un réseau de neurones, Rapport interne, Groupe de recherche en électronique industrielle, UQTR, 1998

[WIL89] R. J. Williams et D. Zipser, ''A Learning Algorithm for Continually Running Fully Reccurent Neural Networks'', Neural Computation 1, 1989, pp 270-280.

ANNEXES

ANNEXE A

Programmes de génération des signaux synthétiques, de configuration et entraînement, et de validation des réseaux de neurones.

```
%============================================================
%
%     Programme de génération des signaux synthétiques
%
%============================================================

% Signaux d'Apprentissage

[Xc,Yc] = xycal1(256,0,1);
[xs,yscale]=dscale(Xc);
[ys,uscale]=dscale(Yc);

% Signaux de Validation

[xv1,yv1] = xyval(512,0,1);

[xv2,yv2] = xytest(256,0,1);



%============================================================
%
% Programme de configuration et d'entraînement des réseaux
%
%============================================================

signaux;

cachee='HHHH';sortie='L---';
trparms=[100 0 1 0];
skip=0;
while length(cachee)<=20
  NetDef=[cachee;sortie];
  a=size(NetDef);
  for mx=10
    for my=10
      w1=0.1*ones(length(cachee),na+nb+1);
      w2=0.1*ones(1,length(cachee)+1);
      NN=[na nb 1];
      [Yh,W12op,W22op,Er_vals]=nnoe(NetDef,NN,w1,w2,trparms,
                                    skip,ys,us,Yc,Uc,yv2,uv2,
                                    yscale,uscale);
    end
  end
  cachee=[cachee 'H'];sortie=[sortie '-'];
end
```

```
%============================================================
%
%       Programme de validation des réseaux de neurones
%
%============================================================

function [Yhat,Eps]=nnvalidd(method,NetDef,NN,W1,W2,
                             par1,par2,par3)


% >>>>>>>>>>>>>>>>      GET PARAMETERS      <<<<<<<<<<<<<<<<<<
skip = 1;
if strcmp(method,'nnoe'),
  mflag=4;
  Y=par1;
  U=par2;
else
  disp('Unknown method!!!!!!!!');
  break
end


% >>>>>>>>>>>>>>>>     INITIALIZATIONS      <<<<<<<<<<<<<<<<<<
Ndat     = length(Y);                    % # of data
na = NN(1);

% ---------- NNARX model ----------
if mflag==1 | mflag==4,
  if length(NN)==1                        % nnar model
    nb = 0;
    nk = 0;
    nu = 0;
  else                                    % nnarx or nnoe mode
    [nu,N] = size(U);
    nb = NN(2:1+nu);
    nk = NN(2+nu:1+2*nu);
  end
  nc = 0;
end


% --------- Common initializations --------
nmax     = max([na,nb+nk-1,nc]);          % 'Oldest' signal
used as input to the model
N        = Ndat - nmax;                   % Size of training
set
nab      = na+sum(nb);                    % na+nb
```

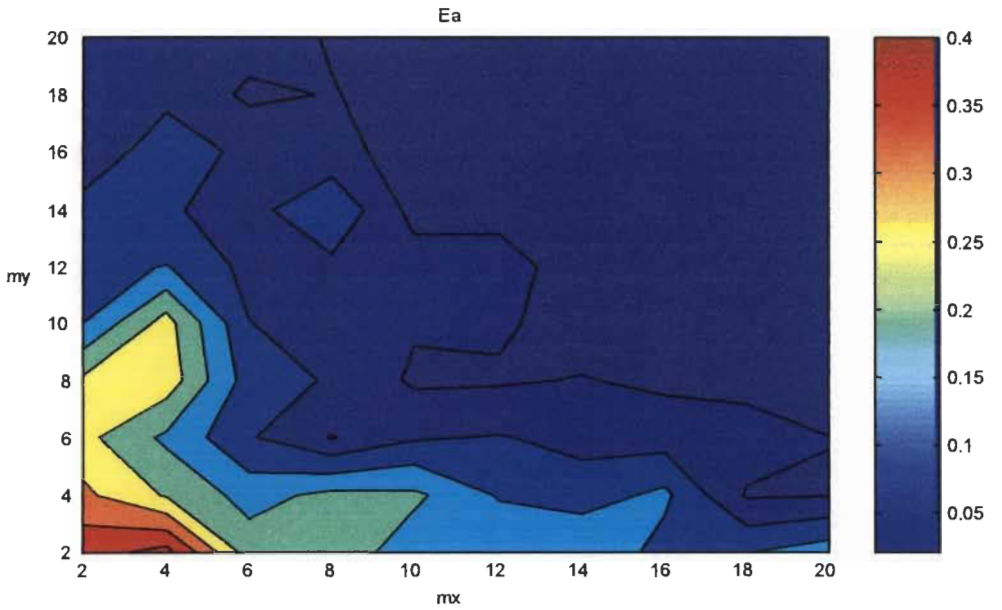
```
nabc     = nab+nc;                          % na+nb+nc
outputs     = 1;                            % Only MISO models
considered
L_hidden = find(NetDef(1,:)=='L')';         % Location of linear
hidden neurons
H_hidden = find(NetDef(1,:)=='H')';         % Location of tanh
hidden neurons
L_output = find(NetDef(2,:)=='L')';         % Location of linear
output neurons
H_output = find(NetDef(2,:)=='H')';         % Location of tanh
output neurons
[hidden,inputs] = size(W1);
inputs          = inputs-1;
E        = zeros(outputs,N);
y1       = zeros(hidden,N);
Yhat     = zeros(outputs,N);


% >>>>>>>>  CONSTRUCT THE REGRESSION MATRIX PHI   <<<<<<<<<<<
PHI = zeros(nab,N);
jj  = nmax+1:Ndat;
for k = 1:na, PHI(k,:)     = Y(jj-k); end
index = na;
for kk = 1:nu,
  for k = 1:nb(kk), PHI(k+index,:) = U(kk,jj-k-nk(kk)+1); end
  index = index + nb(kk);
end


% >>>>>>>>>>>>>>   COMPUTE NETWORK OUTPUT   <<<<<<<<<<<<<<<<<<<
%  ---------- NNOE model ----------
Eps=[];

if mflag==4,
  Y  = Y(nmax+1:Ndat);
  PHI_aug=[PHI;ones(1,N)];
  y1 = [y1;ones(1,N)];
  N2=N+1-skip;
  for t=1:N,
    h1 = W1*PHI_aug(:,t);;
    y1(H_hidden,t) = pmntanh(h1(H_hidden));
    y1(L_hidden,t) = h1(L_hidden);

    h2 = W2*y1(:,t);
    Yhat(H_output,t) = pmntanh(h2(H_output,:));
    Yhat(L_output,t) = h2(L_output,:);

    for d=1:min(na,N-t),
```

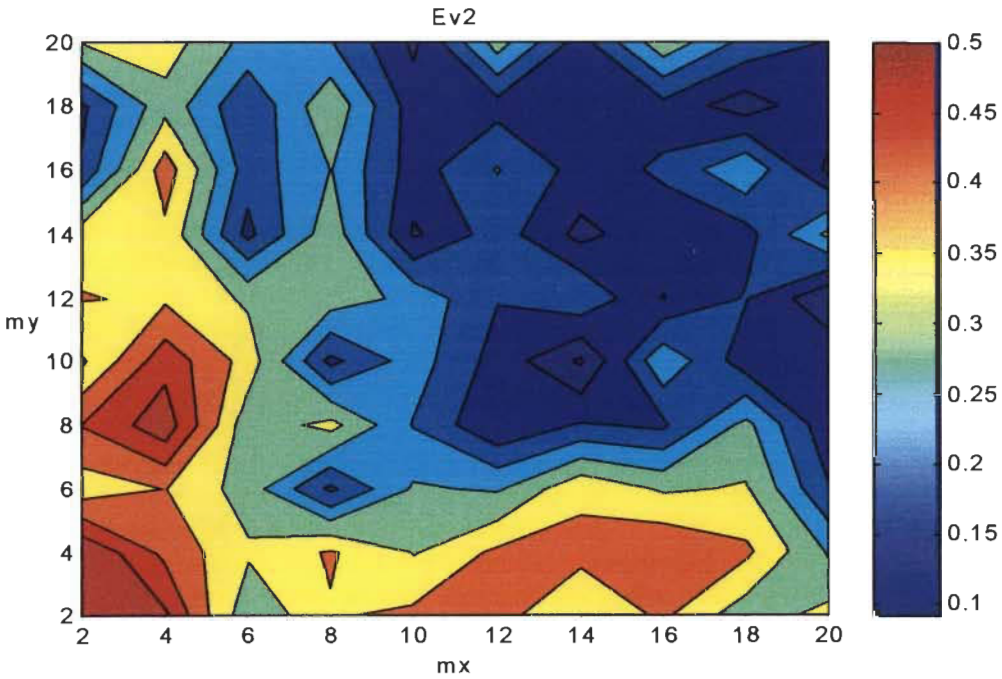
```
      PHI_aug(d,t+d) = Yhat(:,t);
    end
  end
  E     = Y - Yhat;  % Error between Y and deterministic part
  e=(norm(Y-Yhat))/norm(Y);

  Eps = [Eps e];

  SSE       = E(skip:N)*E(skip:N)';          % Sum of squared
errors (SSE)
  PI        = SSE/(2*N2);                    % Performance index
end
```

ANNEXE B

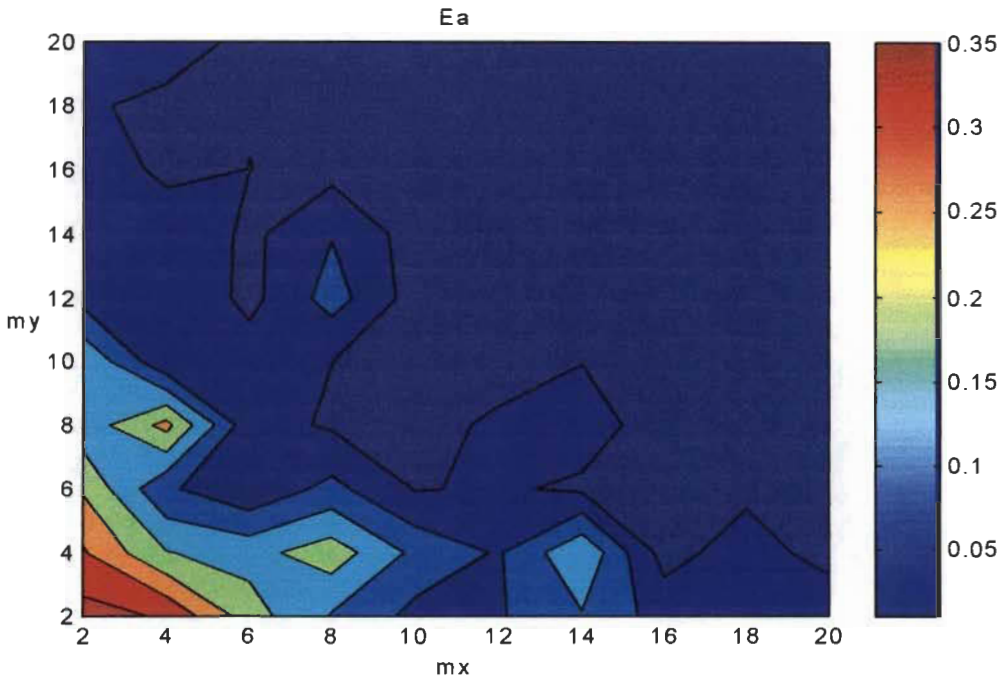
Courbes de répartition des erreurs d'apprentissage et de validation en fonction du nombre de neurones sur la couche cachée.

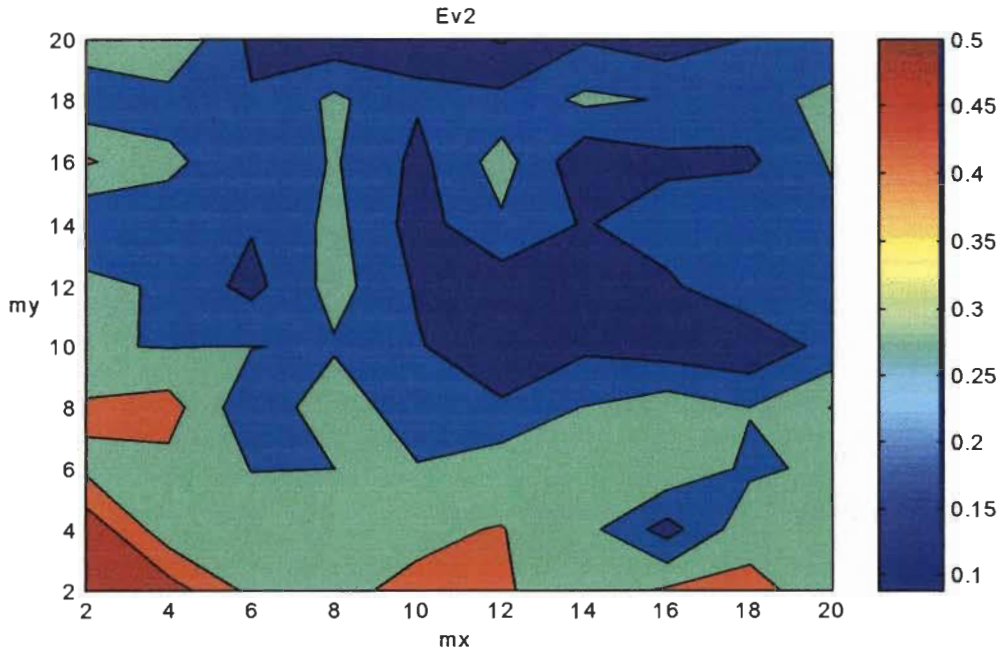
Nc = 5


Ea
20
18
16
14
12
10
8
6
4
2
my
2
4
6
8
10
12
14
16
18
20
mx
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05

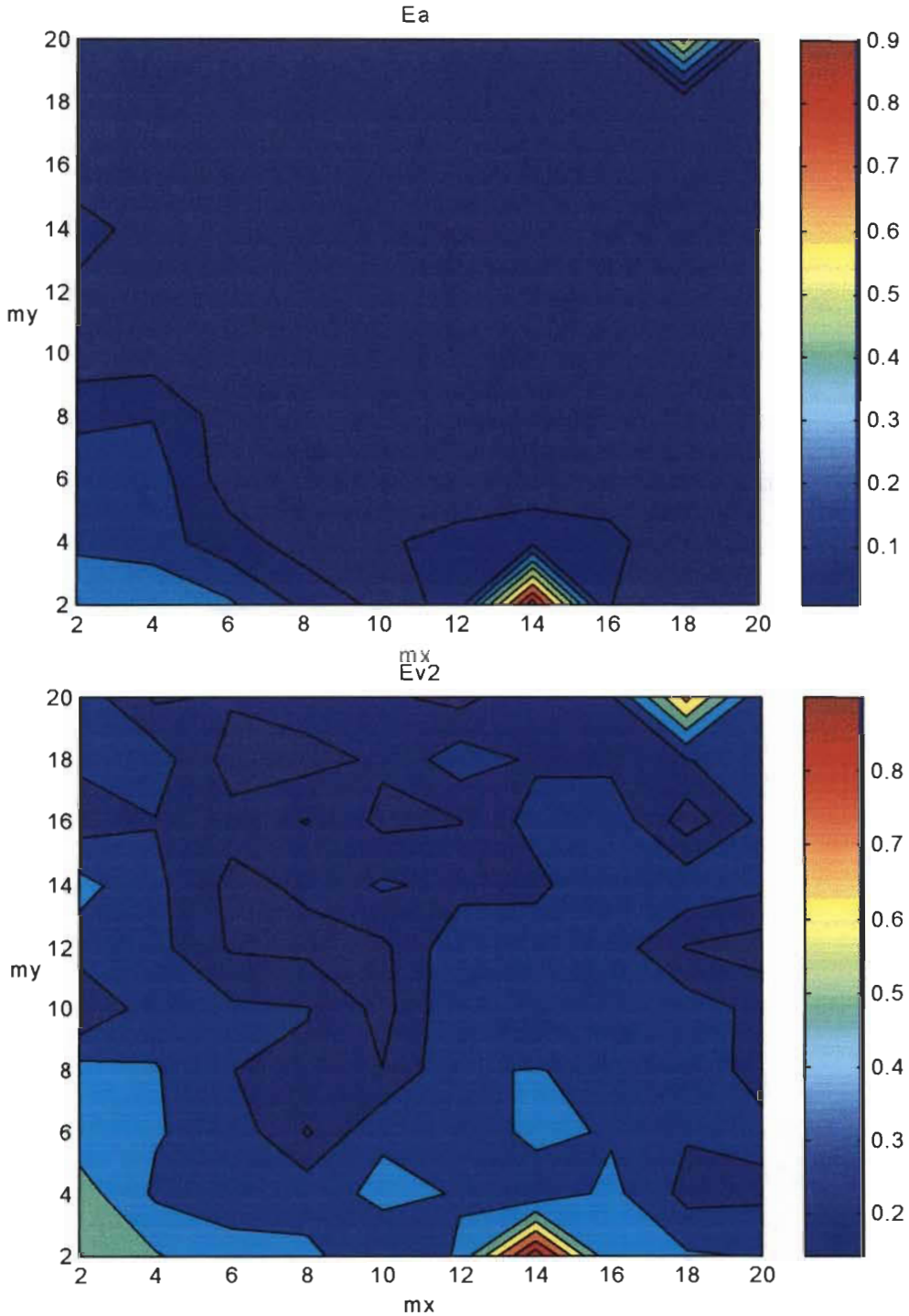

Ev2
20
18
16
14
12
10
8
6
4
2
my
2
4
6
8
10
12
14
16
18
20
mx
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1

Nc=10

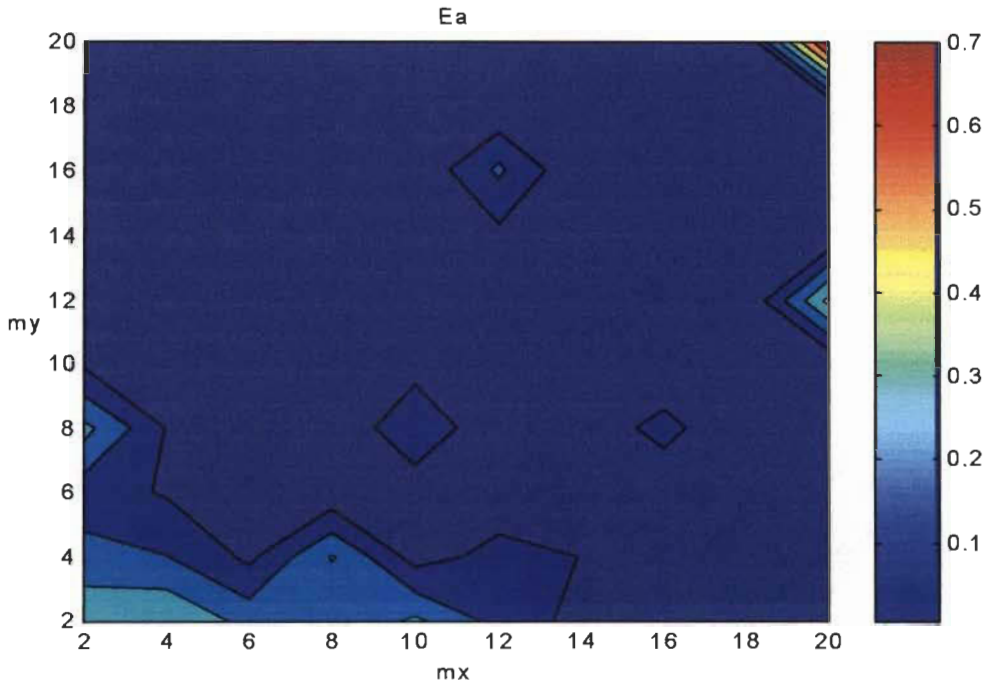

Ea
20
18
16
14
12
10
8
6
4
2
my
2
4
6
8
10
12
14
16
18
20
mx
0.35
0.3
0.25
0.2
0.15
0.1
0.05

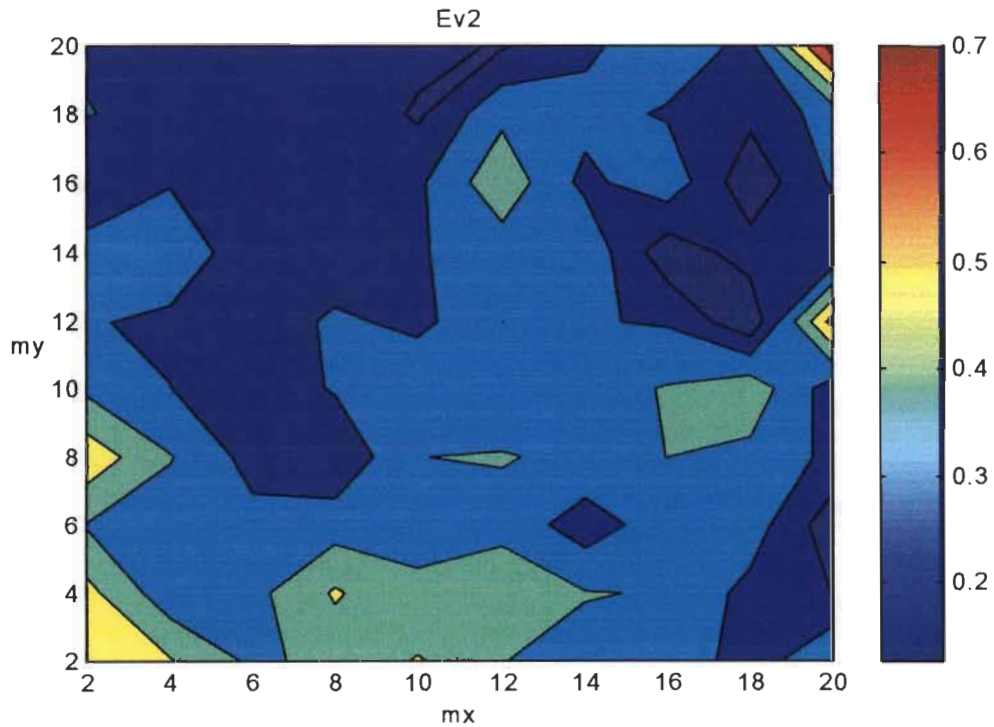

Ev2
20
18
16
14
12
10
8
6
4
2
my
2
4
6
8
10
12
14
16
18
20
mx
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1

Nc=15



Nc=20


Ea
20
18
16
14
12
10
8
6
4
2
my
2
4
6
8
10
12
14
16
18
20
mx
0.7
0.6
0.5
0.4
0.3
0.2
0.1


Ev2
20
18
16
14
12
10
8
6
4
2
my
2
4
6
8
10
12
14
16
18
20
mx
0.7
0.6
0.5
0.4
0.3
0.2

ANNEXE C

Programmes de numérisation des signaux et de quantification des réseaux de neurones.

```
function resultat = digitale(x,Nb_BIT,signe);
%  Fonction numerisation de la valeur analogique "x"
%  Daniel Massicotte, le 9 fevrier 1994
%  Modifie par Bruno MBA MEGNER ( Avril 98 )
%  signe = 1 : un bit de signe est considere dans
l'arondissement
%        = 0 : arondissement non-signe
%  Nb_BIT >= 3 pour signe = 0
%  Nb_BIT >= 2 pour signe = 1

%%%%%%% QUANTIFICATION PAR ARONDI EN COMPLEMENT À 2
%%%%%%%%%%%%%%%


MAX = 1;
for i = 1:length(x)
if x(i) >= 0
   bit_signe = 1;   % x(i) est positif
else
   if signe == 1
      bit_signe = -1;    % x(i) est negatif
   else
      bit_signe = 0;
      resultat(i) = 0;
   end
end

x(i) = abs(x(i));
Nb_bit = Nb_BIT - signe;
if (x(i) >= MAX - 2^(-Nb_bit)) & (bit_signe == 1)
      resultat(i) = MAX - 2^(-Nb_bit);
elseif (x(i) >= MAX) & (bit_signe == -1)
      resultat(i) = MAX;
elseif abs(Nb_BIT) == inf
   resultat(i) = x(i);
else

      if modulo(x(i),2^(-Nb_bit)) <= 2^(-(Nb_bit+1)) | Nb_BIT
== 0

     resultat(i) = x(i) - modulo(x(i),2^(-Nb_bit));

      else
         resultat(i) =  x(i) + 2^(-Nb_bit) - modulo(x(i),2^(-
Nb_bit));
      end
end
```

```
resultat(i) = bit_signe*resultat(i);
end
resultat = resultat';
return
```

```
function [Yhat,Yhatq,E]=nnvalidq(method,NetDef,NN,W1,W2,
                          par1,par2,Nb_bits,Nb_bitp,signe)

%  NNVALIDq
%  -------
%   Quantify & Validate a neural network input-output model
%of a dynamic system.

% >>>>>>>>>>>>      GET PARAMETERS     <<<<<<<<<<<<<<<<<<<<
skip = 1;
if strcmp(method,'nnoe'),
  mflag=4;
  Y=par1;
  U=par2;
end

% >>>>>>>>>>>>     INITIALIZATIONS     <<<<<<<<<<<<<<<<<<<<<<
Ndat     = length(Y);                    % # of data
na = NN(1);

% ---------- NNARX model ----------
if mflag==1 | mflag==4,
  if length(NN)==1                       % nnar model
    nb = 0;
    nk = 0;
    nu = 0;
  else                                   % nnarx or nnoe model
    [nu,N] = size(U);
    nb = NN(2:1+nu);
    nk = NN(2+nu:1+2*nu);
  end
  nc = 0;
end

% --------- Common initializations --------
```

```
nmax     = max([na,nb+nk-1,nc]);        % 'Oldest' signal
used as input to the model
N        = Ndat - nmax;                 % Size of training
set
nab      = na+sum(nb);                  % na+nb
nabc     = nab+nc;                      % na+nb+nc
outputs     = 1;                        % Only MISO models
considered
L_hidden = find(NetDef(1,:)=='L')';     % Location of linear
hidden neurons
H_hidden = find(NetDef(1,:)=='H')';     % Location of tanh
hidden neurons
L_output = find(NetDef(2,:)=='L')';     % Location of linear
output neurons
H_output = find(NetDef(2,:)=='H')';     % Location of tanh
output neurons
[hidden,inputs] = size(W1);
inputs          = inputs-1;
E        = zeros(outputs,N);
y1       = zeros(hidden,N);
y1q = zeros(hidden,N);
Yhat     = zeros(outputs,N);
Yhatq = zeros(outputs,N);

% ----Digitalisation des signaux et des matrices des poids---

Fnorm=16;
a=length(Y);

Yn=Y/Fnorm;                             %Normalisation de Y
Un=U/Fnorm;
Ynq=digitale(Yn,Nb_bits,signe);         %Quantification
Unq=digitale(Un,Nb_bits,signe);
Yq=Ynq'*Fnorm;                          %Denormalisation
Uq=Unq'*Fnorm;

wn1=W1/Fnorm;                           %Normalisation des poids
W1 et W2
wn2=W2/Fnorm;
[a,b]=size(wn1);                        %Quantification des poids
for i=1:a
  for j=1:b
    wnq1(i,j)=digitale(wn1(i,j),Nb_bitp,signe);

  end
end
[a,b]=size(wn2);
for i=1:a
```

```
  for j=1:b
    wnq2(i,j)=digitale(wn2(i,j),Nb_bitp,signe);
  end
end
Wq1=wnq1*Fnorm;                          %Denormalisation
Wq2=wnq2*Fnorm;


% >>>>>>>>>  CONSTRUCT THE REGRESSION MATRIX PHI   <<<<<<<<<
PHI = zeros(nab,N);PHIq = zeros(nab,N);
jj  = nmax+1:Ndat;
for k = 1:na,PHI(k,:)=Y(jj-k); PHIq(k,:)=Yq(jj-k); end
index = na;
for kk = 1:nu,
  for k = 1:nb(kk), PHI(k+index,:)=U(kk,jj-k-nk(kk)+1);
PHIq(k+index,:) = Uq(kk,jj-k-nk(kk)+1); end
  index = index + nb(kk);
end

% >>>>>>>>>>   COMPUTE NETWORK OUTPUT   <<<<<<<<<<<<<<<<
%  ---------- NNOE model ----------

if mflag==4,
  Y  = Y(nmax+1:Ndat); Ynq  = Ynq(nmax+1:Ndat);
  PHI_aug=[PHI;ones(1,N)]; PHIq_aug=[PHIq;ones(1,N)];
  y1 = [y1;ones(1,N)]; y1q = [y1q;ones(1,N)];
  N2=N+1-skip;
  [a1,b1]=size(W1);
  [a2,b2]=size(W2);

  for t=1:N,

    h1=[];h1q=[];
    phi=PHI_aug(:,t);phiq=PHIq_aug(:,t);
    for i1=1:a1
      h1(i1)=0;h1q(i1)=0;
      for j1=1:b1
        h11 = W1(i1,j1)*phi(j1);h1(i1)=h1(i1)+h11;     %An

h11q=digitale(wnq1(i1,j1)*phiq(j1),max(Nb_bits,Nb_bitp),signe
);   %Num

h1q(i1)=digitale(h1q(i1)+h11q,max(Nb_bits,Nb_bits),signe);
      end
end
    h1=h1';                                              %An
    h1q=h1q'*Fnorm;                                      %Num
```

```
    for i1=1:length(H_hidden)
      y1h(i1) = pmntanh(h1(i1));                          %An.
      y1hq(i1) =
digitale(pmntanh(h1q(i1)),max(Nb_bits,Nb_bitp),signe);%Num.
    end
    y1(H_hidden,t) = y1h(H_hidden)';                      %An
    y1q(H_hidden,t) = y1hq(H_hidden)';                    %Num
    %y1q(H_hidden,t), pause

    h2=0; y1t=y1(:,t);
    h2q=0; y1tq=y1q(:,t);
    for i1=1:b2
      h22=W2(i1)*y1t(i1);h2=h2+h22;                       %An

h22q=digitale(wnq2(i1)*y1tq(i1),max(Nb_bits,Nb_bitp),signe);
     %Num
      h2q=digitale(h2q+h22q,max(Nb_bits,Nb_bitp),signe);
    end
    h2q=h2q*Fnorm;

    Yhat(L_output,t) = h2(L_output,:);       %Sortie an
    Yhatq(L_output,t) = h2q(L_output,:);     %Sortie num

    for d=1:min(na,N-t),           %Actualisation des retards
      PHI_aug(d,t+d) = Yhat(:,t);
      PHIq_aug(d,t+d) = Yhatq(:,t);
    end
end
  E     = norm(Yhat - Yhatq)/norm(Yhat);     %Erreur entre
                                              Yhat & Yhatq
end
```