UNIVERSITÉ DE LORRAINE

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

Université Henri Poincaré, Nancy 1
UFR ESSTIN
DFD Automatique et Production Automatisée
Ecole Doctorale IAE + M

CRAN
CNRS URA D821

THESE

S.C.D. - U.H.P.
NANCY 1
SCIENCES
BIBLIOTHÈQUE

Présentée par

Philippe Thomas

en vue de l'obtention du

DOCTORAT DE L'UNIVERSITE HENRI POINCARE, NANCY 1

Spécialité Automatique

Contribution à l'Identification de Systèmes Non Linéaires par Réseaux de Neurones.

Soutenue publiquement le 5 mars 1997 devant la commission d'examen :

Président :	A. RICHARD	
		Professeur à l'Université Henri Poincaré Nancy 1
Rapporteurs :	T. DENOEUX	
		Enseignant Contractuel Habilité à l'Université de Technologie de Compiègne
	C. JUTTEN	
		Professeur à l'Université Joseph Fourier de Grenoble
Examinateurs :	G. BLOCH	
		Maître de Conférences à l'Université Henri Poincaré Nancy 1
	C. HUMBERT	
		Professeur à l'Université Henri Poincaré Nancy 1
	J.C. TRIGEASSOU	
		Professeur à l'Université de Poitiers

Remerciements

Ce travail a été réalisé au Centre de Recherche en Automatique de Nancy dans l'équipe Automatique Symbolique et Neuromimétique implantée à l'Ecole Supérieur des Sciences et Technologies de l'Ingénieur de Nancy.

Je tiens à remercier profondément Monsieur A. RICHARD, Professeur à l'Université Henri Poincaré Nancy I de me faire l'honneur de présider le jury.

Je suis profondément reconnaissant à Messieurs T. DENOEUX, Enseignant Contractuel à l'Université de Technologie de Compiègne, et C. JUTTEN Professeur à l'Université Joseph Fourier de Grenoble pour l'honneur qu'ils me font en étant rapporteur de ce mémoire.

Je tiens à exprimer mes vifs remerciements à Monsieur le Professeur C. HUMBERT, directeur de l'ESSTIN, et Monsieur J.C. TRIGEASSOU, Professeur à l'Université de Poitiers, pour avoir bien voulu examiner et juger ce travail.

Je remercie de tout cœur Monsieur G. BLOCH, Maître de Conférences à l'Université Henri Poincaré Nancy I, pour ses discussions fructueuses qui m'ont été d'un précieux apport.

Je tiens également à remercier Monsieur D. THEILLIOL, Maître de Conférence à l'Université Henri Poincaré Nancy I, pour m'avoir initié aux réseaux de neurones.

Je souhaite rendre hommage à l'ensemble de mes collègues de l'ESSTIN pour le temps et l'aide qu'ils m'ont accordés pour réaliser ce mémoire. Que tous veuillent bien trouver ici l'expression de mes vifs et chaleureux remerciements pour leur contribution au maintien constant d'une ambiance cordiale dans laquelle il est agréable de travailler.

Je tiens également à associer à ces remerciements ma famille et mes amis qui m'ont été d'un soutien constant tout au long de mes études et tout particulièrement lors de la rédaction de ce mémoire.

SOMMAIRE

Chapitre III : Algorithmes d'initialisation du perceptron multicouches

Chapitre IV : Algorithmes de minimisation du critère

Chapitre V : Choix du critère à minimiser. Robustesse aux valeurs aberrantes

Introduction

L'établissement de modèles de comportement des systèmes est un domaine incontournable en automatique, que ce soit pour la conduite ou le diagnostic. Pour les systèmes linéaires, de nombreuses formes de modèles ont été proposées et de nombreux algorithmes ont été décrits et validés.

Cependant, les modèles linéaires ne représentent le plus souvent qu'une approximation sur un domaine réduit de systèmes intrinsèquement non-linéaires. Un certain nombre de modèles non-linéaires ont été proposés par les automaticiens tels que les modèles d'Hammerstein ou de Wiener.

Il existe d'autre part un modèle non-linéaire dont l'origine est clairement extérieure à l'automatique, mais qui présente des caractéristiques intéressantes : le perceptron multicouches. Ses capacités d'approximation universelle de fonctions continues permettent en effet de reproduire le comportement de systèmes non-linéaires et d'utiliser les modèles ainsi construits comme prédicteurs.

Les objectifs de ce travail sont donc, d'une part, de replacer ce modèle dans le cadre de la démarche classique d'identification de modèles de comportement de systèmes dynamiques, et, d'autre part, de mettre en place des techniques et une méthodologie efficace pour l'identification neuronale.

L'origine des modèles neuronaux date des premières modélisations du neurone formel, inspirées de travaux en neurobiologie, construites au début des années 40. Ces résultats ont abouti à l'élaboration du premier réseau de neurones : le perceptron. Les limites de ce réseau en tant que classificateur élémentaire ayant été mises en évidence, le domaine est tombé en sommeil pendant une vingtaine d'années. Mais, dès les années 80, la mise au point de l'algorithme d'apprentissage dénommé rétropropagation du gradient et l'accroissement des capacités de calcul informatique ont redonné un nouvel élan aux recherches sur les réseaux neuronaux.

Ce mémoire présente des travaux relatifs à l'identification par réseaux de neurones multicouches non-récurrents de systèmes SISO (simple-entrée, simple-sortie) et MISO (multi-entrées, simple-sortie) non-linéaires. Dans le premier chapitre, nous présenterons succinctement les divers types de modèles non-linéaires existants. Un rapide historique des réseaux de neurones et une présentation de leurs origines biologiques sont également effectués dans ce chapitre.

Dans le deuxième chapitre, nous décrirons plus précisément l'architecture du réseau de neurones que nous utiliserons tout en expliquant les choix qui nous ont conduits à cette structure. Nous appellerons cette structure : perceptron multicouches. Nous poursuivrons en présentant les liens étroits existants entre les modèles neuronaux et les modèles non-linéaires plus traditionnels. Nous finirons ce chapitre en présentant divers critères de qualité utilisables pour la validation des modèles neuronaux.

Dans le chapitre trois, nous nous intéresserons à l'initialisation des paramètres du perceptron multicouches. La question de l'initialisation des poids pour un tel modèle se pose quel que soit le contexte et tout utilisateur y est confronté même si cette difficulté est souvent éludée. Nous présenterons les problèmes que l'on peut rencontrer du fait d'une mauvaise initialisation ainsi que plusieurs algorithmes mis au point pour pallier ces problèmes. Parmi ces derniers, deux ont été mis au point par nos soins et sont une évolution d'une méthode plus ancienne. Nous finirons ce chapitre en comparant les résultats obtenus sur plusieurs exemples avec chacune des méthodes présentées.

Le problème de l'estimation des paramètres du réseau sera abordé au chapitre quatre. Nous y présenterons le principe de la rétropropagation du gradient. Les algorithmes basés sur ce principe seront présentés dans le cadre de la minimisation d'un critère très général. Divers algorithmes n'utilisant pas la rétropropagation du gradient seront également décrits dont un, récursif, que nous avons mis au point. Ce chapitre se terminera par une étude comparative de ce dernier algorithme avec l'algorithme traditionnel de l'erreur de prédiction récursif (Recursive Prediction Error).

Dans le dernier chapitre, nous nous intéresserons au problème de la robustesse aux valeurs aberrantes présentes dans les données d'identification. Ce problème prend toute son importance dans le cadre du traitement de données industrielles. Cette question est relativement récente dans le domaine des réseaux neuronaux. Pourtant ce problème est particulièrement important dès que l'on modélise un système industriel, car les données fournies par un tel système sont généralement polluées par la présence de valeurs aberrantes. Nous présenterons donc plusieurs critères à minimiser, robustes aux valeurs aberrantes, que nous comparerons ensuite sur un exemple de simulation. Nous finirons ce chapitre en présentant une application industrielle.

Chapitre I

Modélisation des systèmes non-linéaires

I.1. Introduction

Le problème central en identification, pour les modèles linéaires comme non-linéaires, est de définir une structure de modèle adaptée au système à étudier. La première règle à respecter en identification est de ne pas estimer ce qui est déjà connu. En effet, les systèmes étudiés suivent des lois physiques décrites par des relations mathématiques généralement connues. Ces connaissances a priori doivent être utilisées pour déterminer la structure du modèle la plus adaptée au problème considéré.

Il est courant de regrouper les modèles en trois classes suivant trois niveaux de connaissance a priori (Sjöberg *et al.*, 1995) :

- les modèles boîte blanche.
Le système est parfaitement connu et il est possible de construire entièrement la structure de son modèle à partir des connaissances a priori et des relations physiques.

- les modèles boîte grise.
Certaines relations physiques sont déterminables, mais plusieurs paramètres doivent être estimés à partir des observations. On peut considérer deux sous-classes :
  - les modèles physiques, où la structure du modèle peut être construite avec des relations physiques qui possèdent un certain nombre de paramètres à estimer à partir des données,
  - les modèles semi-physiques, où des relations physiques sont utilisées pour construire certaines combinaisons non-linéaires de données mesurées afin de créer des signaux théoriques (non mesurables). Ces nouveaux signaux sont alors utilisés dans un modèle de type boîte noire.

- les modèles boîte noire.
Aucune relation physique n'est connue ou utilisable, mais la structure du modèle appartient à une famille connue pour avoir une grande flexibilité ce qui permet de modéliser correctement le système.

Le choix des modèles boîte blanche et boîte grise étant fortement dépendant du système, du fait de l'utilisation de connaissances a priori dans la détermination de la structure, nous nous bornerons à étudier les modèles boîtes noires.

Le problème de l'identification de systèmes dynamiques linéaires est actuellement bien connu et un grand nombre de techniques ont été mises au point pour estimer les paramètres de modèles linéaires, discrets (Ljung, 1987; Söderström et Stoica, 1989) ou continus (Unbehauen et Rao, 1987; Sinha et Rao, 1991).

Ces modèles linéaires sont depuis longtemps utilisés pour la commande ou le diagnostic de systèmes réels. Cependant, ces systèmes sont généralement non-linéaires sur l'ensemble de leur domaine d'utilisation. Donc, un modèle linéaire construit pour un point de fonctionnement ne fournira pas une modélisation correcte pour d'autres points de fonctionnement ce qui va conduire à un échec du diagnostic ou de la commande du système,

ce dernier point pouvant même rendre le système instable. Deux stratégies peuvent être proposées pour contourner cette difficulté. La première est de construire un modèle qui assure au moins la stabilité du système sur un jeu donné de points de fonctionnement. Cependant, une telle stratégie fournit généralement un modèle peu précis. Ceci peut induire des difficultés dans l'implantation de ce modèle dans une stratégie de commande par exemple. Au mieux, un tel modèle peut être correct uniquement lorsque la nonlinéarité est faible. Parfois, la nonlinéarité est telle qu'elle implique un changement important du comportement du système durant son utilisation ce qui va conduire à l'échec d'une telle stratégie (Isidori, 1989; Nijmeier et Van der Schaft, 1990; Patra et Unbehauen, 1993). La seconde stratégie est de constamment superviser le système en estimant récursivement les paramètres d'un modèle linéaire de manière à ce que le modèle puisse évoluer d'un point de fonctionnement à l'autre. Cette stratégie ne permet cependant pas d'éviter toutes les difficultés. En effet, un tel modèle adaptatif peut devenir instable, en particulier parce que les systèmes fonctionnent souvent en boucle fermée. Ce point implique d'utiliser une procédure complexe pour superviser le système. Un autre inconvénient de cette stratégie est qu'elle ne garantit ni la stabilité, ni les performances durant les phases de transitions entre deux points de fonctionnement. Le contrôle d'un système à l'aide de cette stratégie durant la phase de transition peut conduire à des effets catastrophiques, particulièrement si le système est instable en boucle ouverte (Patra et Unbehauen, 1993).

La meilleure alternative à ces approches est de modéliser le système à l'aide d'un modèle non-linéaire et d'estimer les paramètres de ce modèle.

Cette approche a fait l'objet d'un intérêt croissant durant ces dernières années et tout particulièrement depuis que des progrès importants ont été faits dans le domaine de l'analyse et du contrôle de systèmes non-linéaires (Banks, 1988; Isidori, 1989; Nijmeier et Van der Schaft, 1990).

Dans le domaine de l'identification, plusieurs modèles ont été proposés en partant des travaux classiques de Wiener (1958) jusqu'aux récents développements dans le domaine des réseaux de neurones. Les différentes approches de modélisation non-linéaire ont été classées en quatre grandes classes par Patra et Unbehauen (1993).

- Les modèles non-paramétriques.
Cette classe regroupe les séries de Volterra, de Wiener et autres approximations en séries.

- Les modèles paramétriques discrets.
Les modèles de Hammerstein, de Wiener, bilinéaires et de Kolmogorov-Gabor font partie de cette catégorie.

- Les modèles paramétriques continus.
Les modèles de Hammerstein continus, bilinéaires et multilinéaires ainsi que les équations différentielles non-linéaires sont regroupés dans cette catégorie.

- Les modèles semi-paramétriques.

Ce terme est proposé par Patra et Unbehauen (1993) pour regrouper les nouvelles classes de modèles basées sur la logique floue, les réseaux d'ondelettes et les réseaux de neurones que nous allons plus particulièrement étudier au cours de ce mémoire. Ces modèles, et en particulier les réseaux neuronaux, présentent l'avantage de pouvoir approximer toute fonction non-linéaire avec une précision, et sur un espace des entrées, donnés (Cybenko, 1989; Funahashi, 1989).

Dans ce chapitre, nous présenterons les différents types de modèles non-linéaires, puis nous poursuivrons en rappelant les origines biologiques des réseaux de neurones, et nous finirons par la présentation des divers types de réseaux de neurones existants.

I.2. Modèles non-linéaires

Comme nous l'avons vu en introduction, les modèles linéaires du type boîte noire ont été les premiers à être étudiés et utilisés. Pour ces modèles, la tâche consiste en l'approximation de la réponse impulsionnelle du système étudié. Cette tâche correspond en fait à un problème traditionnel d'approximation qui a été souvent étudié avec succès en utilisant quelques structures bien connues de modèles linéaires de type boîte noire comme les modèles ARX, ARMAX...

Le problème de la modélisation non-linéaire du type boîte noire est plus difficile car rien n'est exclu a priori et un très grand éventail de familles de modèles doit être étudié.

Dans ce paragraphe, nous allons présenter les diverses familles de modèles non-linéaires. Les domaines abordés sont très divers en partant des mathématiques appliquées aux systèmes non-linéaires (Silverman, 1986; Banks, 1988) jusqu'aux algorithmes et concepts plus récents tels que les réseaux de neurones (Kung, 1993), les réseaux d'ondelettes (Meyer, 1990; Benveniste *et al.*, 1994) et les modèles flous (Wang, 1994). La structure de ce paragraphe est inspirée des articles de synthèse de Patra et Unbehauen (1993) et de Sjöberg *et al.* (1995).

I.2.1. Structure générale d'un modèle non-linéaire

Pour des commodités de présentation des équations, nous nous bornerons ici aux modèles SISO (simple-entrée, simple-sortie). Cette présentation peut cependant être étendue aux modèles MIMO (multi-entrées, multi-sorties). L'identification d'un système dynamique utilise l'observation de ses entrées u(t) et de ses sorties y(t) :

$$\begin{aligned} u_t &= [u(1) \quad u(2) \cdots u(t)] \\ y_t &= [y(1) \quad y(2) \cdots y(t)] \end{aligned} \tag{I.1}$$

Elle consiste dans la détermination d'une relation entre ces observations passées $[u_{t-1}; y_{t-1}]$ et la sortie réelle du système y(t) :

$$y(t) = f(u_{t-1}, y_{t-1}) + e(t) \tag{I.2}$$

Le terme additif e(t) traduit le fait que la sortie future y(t) ne sera pas une fonction exacte des observations passées. Pour pouvoir considérer $f(u_{t-1}, y_{t-1})$ comme une prédiction correcte de la sortie y(t), il est nécessaire que e(t) soit de faible amplitude.

La relation entrée-sortie recherchée va être définie par la fonction f dans l'équation (I.2). Quel que soit le type de modèle utilisé, nous allons chercher cette fonction f parmi une famille de fonctions que nous pouvons paramétrer :

$$f(u_{t-1}, y_{t-1}, \theta) \tag{I.3}$$

où θ représente le vecteur des paramètres de dimension finie et est une des caractéristiques de la famille de fonction considérée.

Il nous faut maintenant rechercher le vecteur θ des paramètres fournissant la meilleure prédiction de la sortie y(t). Il est possible de mesurer la qualité du vecteur θ sélectionné en évaluant l'ajustement entre les sorties du modèle et les sorties mesurées sur la base d'un critère :

$$\sum_{t=1}^{n} L(\varepsilon(t,\theta)) \tag{I.4}$$

où n représente le nombre de données mesurées, $\varepsilon(t, \theta) = y(t) - f(u_{t-1}, y_{t-1}, \theta)$ est l'erreur de prédiction et L est une fonction à définir, le plus souvent quadratique.

La structure de la famille de modèle décrite en (I.3) est trop générale pour pouvoir être utilisée directement. Il peut être utile de modifier ce modèle en utilisant un vecteur $\phi(t)$ de dimension finie, nommé vecteur de régression, permettant de sélectionner les observations passées utiles à la description du modèle :

$$f(u_{t-1}, y_{t-1}, \theta) = f(\phi(t), \theta) \tag{I.5}$$

où $\phi(t) = \phi(u_{t-1}, y_{t-1})$.

Le choix du modèle non-linéaire décrit à l'équation (I.3) se décompose alors en deux problèmes partiels pour les systèmes dynamiques :

- comment choisir le vecteur de régression $\phi(t)$ à partir des entrées et des sorties passées du système,
- comment choisir la transformation non-linéaire $f(\phi)$ de l'espace des régresseurs vers l'espace des sorties.

Nous allons nous intéresser dans un premier temps au choix de la fonction $f(\phi)$.

I.2.2. Modèles non-paramétriques continus

Cette classe de modèles a été proposée dès les années trentes par Volterra (1930; 1959) et Wiener (1958) afin de construire des modèles non-linéaires continus. Ces modèles peuvent être considérés comme la généralisation de la représentation par réponse impulsionnelle des systèmes linéaires. Leurs principes reposent sur le théorème de Stone-Weierstrass (Zbikowski et Dzielinski, 1995). Les séries de Volterra ont pour équation :

$$y(T) = H_1(u(T)) + H_2(u(T)) + \cdots + H_N(u(T)) = \sum_{k=1}^{N} H_k(u(T)) \qquad (I.6)$$

où l'opérateur de Volterra du $k^{ième}$ ordre $H_k(u(T))$ est déterminé par la relation :

$$H_k(u(T)) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_k(\tau_1, \cdots, \tau_k) u(T-\tau_1) \cdots u(T-\tau_k) d\tau_1 \cdots d\tau_k \qquad (I.7)$$

où $h_k(\tau_1, ..., \tau_k)$ est le noyau de Volterra du $k^{ième}$ ordre et T représente le temps.

Ces noyaux sont des fonctions bornées, continues en chaque point τ_j, symétrique et, pour les systèmes causals, $h_k(\tau_1, ..., \tau_k) = 0$ pour tout $\tau_j < 0$. L'utilisation d'une série de Volterra pour modéliser un système non-linéaire nécessite la détermination de ces noyau pour différentes valeurs de τ_j. Ceci correspond à la détermination de la réponse impulsionnelle dans le cas de modèle linéaire (Wigren, 1990).

Shetzen (1965) a montré que le noyau $h_1(\tau)$ de l'opérateur de Volterra H_1 du premier ordre est la réponse impulsionnelle d'un système. Deux difficultés sont associées à l'utilisation pratique des séries de Volterra. La première concerne la détermination des noyaux de Volterra $h_k(\tau_1, \tau_2, ..., \tau_k)$. La seconde est liée à la convergence des séries (Bissessur et Naguib, 1996).

La détermination des noyaux de Volterra d'un système est possible seulement si la contribution de chacun des opérateurs de Volterra du système peut être séparée à partir de la réponse totale du système (Bissessur et Naguib, 1996).

La seconde difficulté est liée au fait que la série de Volterra est en fait une série dérivée des séries de Taylor. L'ensemble des problèmes de convergence associés aux séries de Taylor est donc répercuté sur les séries de Volterra (Bissessur et Naguib, 1996).

Pour résoudre ces problèmes, Wiener (1958) a construit un nouveau jeu de fonctions à partir des opérateurs de Volterra. Le principe de la série de Wiener est de construire chaque nouveau noyau orthogonalement aux autres noyaux de manière à ce qu'ils puissent être mesurés indépendamment les uns des autres. Pour cela, il utilise l'orthogonalisation de Gram-Schmidt. La série de Wiener est représentée par :

$$y(T) = G_0(u(T)) + G_1(u(T)) + \cdots + G_N(u(T)) = \sum_{k=1}^{N} G_k(u(T)) \qquad (I.8)$$

où G_k est la fonction de Wiener du $k^{ième}$ ordre. Ces fonctions sont déterminées par les noyaux de Wiener et sont construites de telle sorte qu'elles soient orthogonale lorsque le signal d'entrée est constitué par un bruit blanc Gaussien.

Les noyaux de Wiener sont notés $g_k(\tau_1, ..., \tau_k)$. Ces noyaux sont définis comme étant une série de fonctions de Laguerre dont les paramètres doivent être estimé (Billings, 1980). Les trois premières fonctions sont données par Shetzen (1980) :

$$\begin{aligned} G_0(u(T)) &= g_0 \\ G_1(u(T)) &= \int_0^\infty g_1(\tau)u(T-\tau)d\tau \\ G_2(u(T)) &= \int_0^\infty \int_0^\infty g_2(\tau_1, \tau_2)u(T-\tau_1)u(T-\tau_2)d\tau_1 d\tau_2 - P\int_0^\infty g_2(\tau_1, \tau_2)d\tau_1 \end{aligned} \qquad \text{(I.9)}$$

où P est la densité spectrale de puissance du signal d'entrée. Alpert (1965) a proposé un algorithme permettant de construire ces fonctions et d'estimer leurs noyaux.

Victor et Shapley (1980) ont proposé une autre méthode pour construire une série de fonctions. Le principe repose sur l'utilisation de noyaux fréquentiels. Ils ont montré que pour un système linéaire, le noyau fréquentiel du premier ordre est la transformée de Fourier du noyau de Volterra du premier ordre et que le noyau fréquentiel du second ordre est la transformée de Fourier en deux dimensions du noyau de Volterra du second ordre. Leurs travaux ont permis de déduire que, en fait, les séries de Wiener ne sont pas plus performantes que les séries de Volterra en ce qui concerne les problèmes de mesure des noyaux et de convergence de la série (Bissessur et Naguib, 1996).

Ces modèles sont bien adaptés à la modélisation de système comprenant des non-linéarités 'douces'. Cependant, l'identification de modèle incluant des noyaux d'ordre supérieur au deuxième est très difficile du fait du grand nombre de calcul à effectuer (Billings, 1980).

I.2.3. Modèles paramétriques discrets

Les modèles paramétriques discrets permettent une certaine compression de données et fournissent une représentation plus compacte des systèmes dynamiques. Ces modèles ont été utilisés intensivement pour, par exemple, le contrôle et le diagnostic des systèmes linéaires. Les modèles linéaires peuvent être décrits à l'aide de l'équation générale (Ljung, 1987) :

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t) \qquad \text{(I.10)}$$

En choisissant, dans ce modèle général, les polynômes :
- A = 1, on obtient le modèle de Box-Jenkins (BJ)
- F = D = 1, nous retrouvons le modèle ARMAX,

- A = C = D = 1, nous parvenons au modèle d'erreur de sortie (OE) et enfin,
- F = C = D = 1, nous obtenons le modèle ARX.

Les éléments du vecteur de régression $\phi(t)$ sont :
- u(t-k), les entrées retardées du système sont associées au polynôme B,
- y(t-k), les sorties retardées du système sont associées au polynôme A,
- $\hat{y}_u(t-k|\theta)$, les sorties simulées à partir uniquement des entrées retardées sont associées au polynôme F,
- $\varepsilon(t-k) = y(t-k) - \hat{y}(t-k|\theta)$, l'erreur de prédiction est associée au polynôme C,
- $\varepsilon_u(t-k) = y(t-k) - \hat{y}_u(t-k|\theta)$, l'erreur de simulation est associée au polynôme D.

Cependant, le cas des systèmes non-linéaires n'est pas aussi simple. En effet, il est nécessaire, lors de l'identification structurelle, de déterminer la forme de la non-linéarité du système.

Chen *et al.* (1990) et Chen et Billings (1992) ont proposé d'utiliser une classification des modèles non-linéaires suivant une nomenclature dérivée de celle des modèles linéaires et basée sur la constitution de leurs vecteurs de régression. Les modèles NARX qui regroupent dans leur vecteur de régression $u(t-k)$ et y(t - k). Les modèles NOE qui utilisent dans leur vecteur de régression u(t - k) et $\hat{y}_u(t-k|\theta)$. Les modèles NARMAX dont les vecteurs de régression sont constitués par u(t - k), $y(t-k)$ et $\varepsilon(t-k|\theta)$. Les modèles NBJ qui utilisent dans leur vecteur de régression u(t -k), $\hat{y}_u(t-k|\theta)$, $\varepsilon(t-k|\theta)$ et $\varepsilon_u(t-k|\theta)$.

Les structures de modèles du type NOE, NBJ, et NARMAX correspondent à des structures récurrentes, car une partie de leur vecteur de régression correspond à des sorties passées du modèle. Il est en général plus compliqué de travailler avec des structures récurrentes. De plus, il devient difficile de savoir pour quelles conditions le modèle obtenu est stable. De surcroît, l'utilisation de telles structures pose de plus grandes difficultés dans le calcul du gradient du résidu utilisé pour l'estimation des paramètres du modèle.

Plusieurs modèles NARX ont été proposés. Nous allons ici présenter succinctement les plus couramment utilisés.

Le modèle de Hammerstein est un modèle prenant en compte les non-linéarités uniquement sur les entrées des systèmes telles que des saturations d'actionneurs. Sa relation entrée-sortie est décrite par l'équation :

$$\hat{y}(k) = -\sum_{i=1}^{n_a} a_i\, y(k-i) + \sum_{i=1}^{n_b} b_i c_0 + \sum_{i=1}^{n_b}\sum_{j=1}^{n_c} b_i\, c_j\, u^j(k-i) \qquad (I.11)$$

où u^j est la $j^{ième}$ puissance de u.

Un exemple de modèle d'Hammerstein proposé avec les ordres $n_c = 3$, $n_a = n_b = 1$ est présenté à l'équation (I.12) :

$$\begin{aligned}\hat{y}(k) &= -a_1 y(k-1) + b_1 c_0 + b_1 c_1 u(k-1) + b_1 c_2 u^2(k-1) + b_1 c_3 u^3(k-1) \\ &= -d_1 y(k-1) + d_2 + d_3 u(k-1) + d_4 u^2(k-1) + d_5 u^3(k-1)\end{aligned} \qquad \text{(I.12)}$$

Il est clair que le modèle, donné par la deuxième relation, est un modèle linéaire en fonction des paramètres d_i. Ces paramètres d_i peuvent être estimés par une méthode traditionnelle du type moindres carrés. Cependant, le nombre des paramètres a_i, b_i et c_j à estimer est plus grand que celui fourni par la somme des ordres des trois polynômes. Il est donc nécessaire d'utiliser une autre stratégie pour retrouver ces paramètres. Ouladsine (1993) a présenté et comparé plusieurs méthodes permettant d'estimer l'ensemble des paramètres dont les plus efficaces sont l'algorithme de Narendra et Gallman (1966) et la méthode RSLS (Recursive Sequential Least Square) (Ouladsine, 1993; Ouladsine *et al.*, 1994). Dans ce but, le signal d'entrée doit exciter tous les modes dynamiques relatifs au systèmes et l'information qu'il apporte au système doit être uniformément répartie entre le minimum et le maximum de cette excitation (Ouladsine, 1993).

Le modèle de Wiener fonctionne dans le même esprit que la représentation des systèmes non-linéaires en série de Wiener (I.8). Ce modèle prend en compte uniquement les non-linéarités en sortie du système. Sa relation entrée-sortie est représentée par les équations :

$$\begin{aligned}\hat{y}(k) &= \sum_{i=0}^{n_c} c_i\, x^i(k) \\ x(k) &= \sum_{i=0}^{n_b} b_i\, u(k-i) - \sum_{i=1}^{n_a} a_i\, x(k-i)\end{aligned} \qquad \text{(I.13)}$$

où x^i est la $i^{ième}$ puissance de x.

La difficulté principale rencontrée avec ce modèle tient au fait que les régresseurs doivent utiliser l'estimation de la quantité x(k) qui n'est pas mesurable. Ce modèle n'est donc pas linéaire en fonction des paramètres. L'estimation des paramètres doit être résolu à l'aide d'algorithmes plus complexes comme, par exemple, l'algorithme de descente du gradient. Wigren (1990) a présenté plusieurs méthodes permettant d'effectuer cette estimation dont la Méthode de l'Erreur de Prédiction (PEM). Un exemple de modèle de Wiener avec $n_c = 3$, $n_a = n_b = 1$ est présenté ci dessous :

$$\begin{aligned}\hat{y}(k) &= c_0 + c_1 x(k) + c_2 x^2(k) + c_3 x^3(k) \\ x(k) &= b_1 u(k-1) + a_1 x(k-1)\end{aligned} \qquad \text{(I.14)}$$

Le modèle bilinéaire décrit une classe particulière de non-linéarité où seuls les produits entre les termes d'entrée et les termes de sortie du système sont considérés et où l'on ne considère pas leurs puissances d'ordre supérieur.

Sa relation entrée-sortie est représentée par l'équation :

$$y(k) = \sum_{i=1}^{n_b} b_i u(k-i) - \sum_{i=1}^{n_a} a_i y(k-i) + \sum_{i=1}^{n_a}\sum_{j=1}^{n_b} c_{ij} y(k-i)u(k-j) \qquad (I.15)$$

Ce modèle est bien évidemment linéaire en fonction des paramètres qui peuvent être aisément estimés par une méthode traditionnelle du type moindres carrés. Un exemple de modèle bilinéaire avec $n_a = n_b = 1$ est présenté ci dessous :

$$\hat{y}(k) = b_1 u(k-1) + a_1 y(k-1) + c_{11} y(k-1) u(k-1) \qquad (I.16)$$

Pour finir, le modèle de Kolmogorov-Gabor est le modèle paramétrique discret le plus général. Son principe repose également sur le théorème de Stone-Weierstrass. Sa relation entrée-sortie est décrite par l'équation :

$$\begin{aligned}\hat{y}(k) = \bar{y} + \sum_{j=1}^{q}\sum_{l=0}^{j}\sum_{i_1=0}^{n_a}\sum_{i_2=i_1}^{n_a}\cdots\sum_{i_l=i_{l-1}}^{n_a}\sum_{i_{l+1}=1}^{n_b}\sum_{i_{l+2}=i_{l+1}}^{n_b}\cdots \\ \sum_{i_j=i_{j-1}}^{n_b} a_{i_1 i_2 \cdots i_j} \prod_{m_1=i_1}^{i_l} u(k-m_1) \prod_{m_2=i_{l+1}}^{i_j} y(k-m_2)\end{aligned} \qquad (I.17)$$

où $\bar{y}$ représente la moyenne temporelle du signal y(k) et où l'ensemble des puissances et des produits des éléments du vecteurs de régression sont introduits.

Un exemple de modèle de Kolmogorov-Gabor avec les ordres $q = 3$, $n_a = n_b = 1$ est représenté par l'équation (I.18) :

$$\begin{aligned}\hat{y}(k) = \bar{y} &+ a_1 u(k-1) + a_2 y(k-1) \\ &+ a_3 u(k-1) y(k-1) + a_4 u^2(k-1) + a_5 y^2(k-1) \\ &+ a_6 u^2(k-1) y(k-1) + a_7 u(k-1) y^2(k-1) + a_8 u^3(k-1) + a_9 y^3(k-1)\end{aligned} \qquad (I.18)$$

Ce modèle présente le désavantage de conduire à une très forte croissance du nombre de paramètres pour un léger accroissement des ordres q, n_a et n_b. Il est donc nécessaire d'utiliser une détermination automatique de la structure afin d'éliminer les paramètres statistiquement insignifiants. Ivakhnenko (1968) a ainsi proposé la Méthode de Traitement des Données par Groupes (MTDG). Richard (1993) a cependant signalé que cette méthode échoue lorsque le nombre de variable d'entrée est important, qu'elle donne des résultats inconsistants lorsque les signaux sont bruités et qu'elle fournit des modèles instables en dehors du lot d'observation d'estimation.

I.2.4. Modèles semi-paramétriques

Cette dénomination a été proposée par Patra et Unbehauen (1993) pour regrouper les nouvelles structures de modèles pour l'identification introduites à partir d'approches diverses comme les réseaux de neurones, les ondelettes ou encore les modèles flous. L'ensemble de ces modèles a pour principe d'utiliser comme structure générale du modèle décrit par l'équation (I.5) un développement en série de fonction :

$$f(\phi(t), \theta) = \sum \alpha_k f_k(\phi(t)) \tag{I.19}$$

où α_k représente un paramètre et f_k une fonction.

La décomposition en ondelettes est un exemple typique de l'utilisation d'une base de fonctions f_k. Cette base de fonctions est constituée d'une fonction 'mère' que nous appellerons ψ qui est dilatée et translatée jusqu'à la formation d'une base d'ondelettes. Dans ce contexte, il est courant de réécrire l'équation (I.19) en utilisant une double indexation correspondant à l'échelle j et à la localisation k :

$$f_{jk}(\phi(t)) = 2^{j/2}\psi(2^j\phi(t) - k) \qquad \text{pour j et k entiers} \tag{I.20}$$

Il est particulièrement intéressant d'utiliser une famille d'ondelettes orthonormales introduites par Meyer (1990). Cependant, construire une base d'ondelettes orthonormales devient rapidement prohibitif en temps de calcul lorsque la dimension de la base devient importante. Les réseaux d'ondelettes et les réseaux de neurones à fonction de base radiale suivent le même principe en utilisant une base de fonctions dont seules la localisation et l'échelle varient (Zhang et Benveniste, 1992; Benveniste *et al.*, 1994).

Le modèle à hyperplans à charnières (hinging hyperplanes) (Breiman, 1993) correspond au choix de fonctions 'charnières' qui ont la forme d'un 'livre ouvert' dont un exemple est présenté à la figure (I.1). La fonction 'charnière' peut être exprimée par la relation:

$$h(\phi(t)) = \pm \max\left\{\beta^+\phi(t) + \gamma^+, \beta^-\phi(t) + \gamma^-\right\} \tag{I.21}$$

où β^+ et β^- sont des vecteurs colonnes et γ^+ et γ^- sont des scalaires.

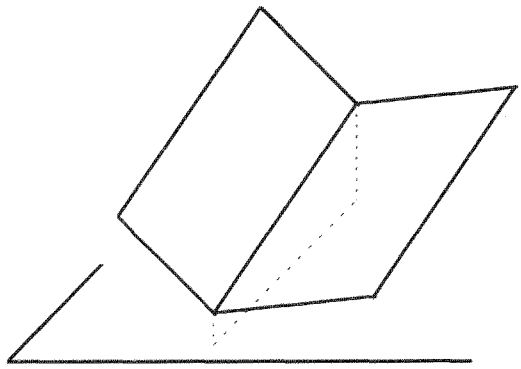


Fig. I.1. Fonction charnière.

Ce modèle est surparamétrisé dans sa forme originelle. Aussi, en introduisant la fonction de base suivante :

$$\psi(x) = \begin{cases} 0 & \text{pour } x < 0 \\ \pm x & \text{pour } x \geq 0 \end{cases} \tag{I.22}$$

le modèle à hyperplans à charnières peut s'écrire :

$$\sum \psi(\beta^T \phi(t) + \gamma) + \mu^T \phi(t) + \gamma_0 \tag{I.23}$$

où μ est un vecteur de paramètres de même dimension que $\phi(t)$.

Le principe de ce modèle est très proche des réseaux de neurones sigmoïdaux multicouches qui utilisent comme fonction de base des fonctions sigmoïdales.

Les modèles flous sont de deux types distincts, les modèles flous à conclusions symboliques (modèle de Mandani) et les modèles flous à conclusions procédurales (Takagi et Sugeno, 1985). Nous ne présenterons que ces derniers qui sont les seuls à être couramment utilisés pour identifier des systèmes non-linéaires. Ces modèles ont également une structure générale décrite par l'équation (I.19). Les fonctions de base sont alors construites à partir des opérations de fuzzyfication, des règles d'inférences, des opérations de defuzzyfication, et des fonctions d'appartenance. Le principe de la fuzzyfication consiste à rendre flou les mesures à étudier, qui sont des informations numériques, de manière à obtenir des informations symboliques. Le résultat de la fuzzyfication est utilisée par un moteur d'inférence qui détermine les déductions associées à chaque règle. Celles ci subissent alors une opération de défuzzyfication qui élabore une déduction unque sous la forme d'une valeur numérique. Le traitement flou de données peut se résumer par la figure (I.2).

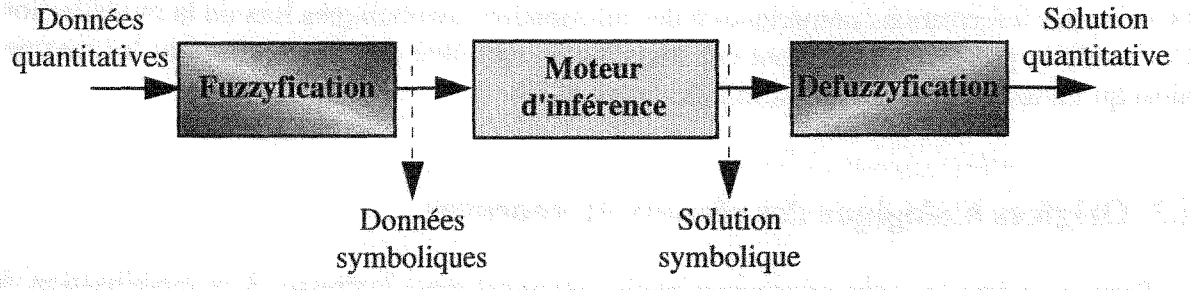


Fig. I.2. Raisonnement flou.

Si l'on considère une variable d'entrée scalaire ϕ, l'opération de fuzzification est définie par un label linguistique A (grand, petit, moyen ...), et la fonction d'appartenance correspondante est :

$$\mu_A : \phi \in \Re \mapsto \mu_A(\phi) \in [0; 1] \tag{I.24}$$

Les règles d'inférence sont élaborées en associant les termes linguistiques relatifs aux sous-ensembles flous d'entrée avec des opérateurs flous qui vérifient les propriétés de la logique binaire :

$$\begin{aligned} &A \text{ ET } B = \min(\mu_A(\phi), \mu_B(\phi)) \\ &A \text{ OU } B = \max(\mu_A(\phi), \mu_B(\phi)) \\ &\text{NON } (A) = 1 - \mu_A(\phi) \end{aligned} \tag{I.25}$$

Dans le modèle de Takagi-Sugeno (1985), les règles d'inférences sont de la forme :

$$\text{si } \phi \text{ est } A_i \text{ alors } y = f_i(\phi) \tag{I.26}$$

La fonction f_i est une fonction affine de la forme $f_i = a_i^T \phi + b_i$ où a_i est un vecteur de paramètres et b_i un biais.

Ces règles nous fournissent une conclusion symbolique que nous pouvons defuzzyfier. La defuzzyfication est obtenue en effectuant une somme pondérée des contributions des K règles constituant le moteur d'inférence :

$$y = \frac{\sum_{i=1}^{K} \mu_{A_i}(\phi) f_i(\phi)}{\sum_{i=1}^{K} \mu_{A_i}(\phi)} \tag{I.27}$$

Pour une description plus complète de la logique floue, se rapporter, par exemple, à Wang (1994). Ces modèles flous peuvent être utilisées pour modéliser un système non linéaire (Lee, 1990; Glorennec, 1993; Sjöberg *et al.*, 1995). Ils présentent l'avantage de pouvoir facilement associer des informations numériques à des informations symboliques lors de la modélisation. Ces modèles présentent cependant une assez forte sensibilité aux bruits affectant les mesures ainsi qu'un temps de calcul prohibitif (Richard, 1993).

I.3. Origines biologique des réseaux de neurones

Dans ce mémoire, nous allons tout particulièrement nous intéresser à la modélisation de systèmes non-linéaires à l'aide de réseaux de neurones artificiels. Ce paragraphe se propose donc de présenter une vue très simplifiée de la réalité biologique du cerveau tout en s'intéressant plus particulièrement aux aspects traitement de l'information et traitement du signal. Pour construire ce paragraphe, nous nous sommes inspirés des livres de Davalo et Naïm (1989), Hérault et Jutten (1994).

I.3.1. Le neurone

Le neurone est l'unité fonctionnelle de base du système nerveux. C'est une cellule spécialisée qui comprend un corps cellulaire ou soma dont les dimensions varient de 20 à 100 µm. Le soma possède, d'une part, des petits prolongements membraneux, les dendrites et, d'autre part, un axone ou cylindraxe qui peut atteindre 1m pour les plus longs (axone géant du calmar). Un exemple de neurone est présenté à la figure (I.3).

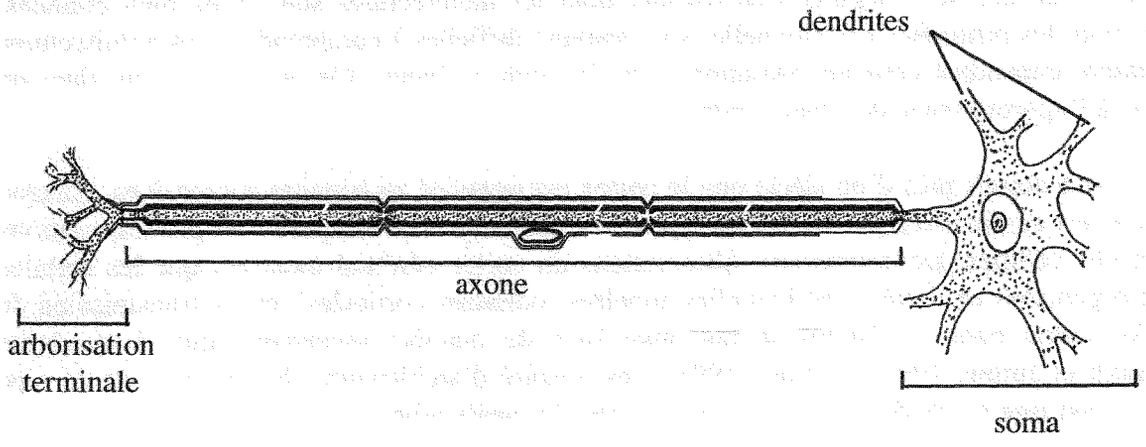


Fig. I.3. Schéma simplifié du neurone (Grignon, 1996).

L'arborisation dendritique va permettre au neurone de se connecter avec d'autres cellules. Les points de contact entre cellules sont appelés synapses et peuvent être de plusieurs types suivant les cellules concernées (motrice, sensorielle) ou suivant la partie du neurone impliquée (synapse axodendritique, dendiodendritique, axoaxonique, ...).

Le nombre de connexions synaptiques par neurone est variable. Par exemple, dans le cortex cérébral, on trouve des arbres dendritiques comportant une cinquantaine de branches alors que dans le cortex cérébelleux, certaines cellules comptent jusqu'à 100 000 synapses.

Le rôle des synapses est de faire passer un message d'un neurone à l'autre. Cette transmission d'information se fait par des mécanismes chimiques (neurohormones, neurotransmetteurs) et électriques (progression de potentiel d'action le long de l'axone). Dans cette présentation du neurone biologique, nous allons uniquement considérer les phénomènes électriques qui apparaissent dans un neurone. Dans notre vision simplifiée, nous pouvons donc considérer l'influx nerveux comme une impulsion électrique et les signaux dendritiques et somatiques comme des variations de potentiel électrique.

Le soma va avoir pour rôle de recueillir et de concentrer les informations reçues par les dendrites et d'en effectuer une sommation dite 'spatio-temporelle'. Ce terme s'explique par le fait que l'étendue de l'arbre dendritique fait parvenir au soma des informations venant d'un grand espace autour du neurone, et parce que la transmission des informations le long d'une dendrite est caractérisée par un retard, une atténuation et un effet de filtrage passe bas. Le terme 'd'intégration somatique' est donc à prendre au sens large (Hérault et Jutten, 1994).

I.3.2. L'organisation en réseau

Les biologistes estiment le nombre de neurones dans le système nerveux humain entre 100 et 1000 milliards et il faut compter pour chacun d'eux entre 1000 et 100 000 synapses. Ces chiffres donnent une idée de la complexité de notre système nerveux et expriment les nombreuses relations liant les cellules nerveuses entre elles.

Ces neurones sont organisés en réseaux dont les architectures sont assez bien connues, mais dont les propriétés fonctionnelles sont souvent difficiles à comprendre. Ces architectures subissent cependant certaines variations d'un individu à l'autre. Ces variations sont dues en partie à l'apprentissage de chaque individu.

On sait depuis près d'un siècle que le cortex est organisé en couches successives. Chaque couche est caractérisée par les divers types de cellules qui les composent et par les relations entre ces cellules. De nombreuses observations du cortex cérébral montrent que les cellules sont organisées en unités fonctionnelles appelées 'colonnes corticales', car la transmission de l'information dans ces unités se fait aussi bien de manière ascendante que descendante (Hérault et Jutten, 1994). Beaugé (1995) s'est inspiré d'architectures biologiques de ce type pour construire un modèle à base de cartes corticales artificielles.

Le cortex visuel est l'un des premiers dont on a pu déterminer les propriétés fonctionnelles. Les biologistes ont remarqué que si l'on se déplace dans une certaine direction dans ce cortex, les différentes 'colonnes corticales' présentent périodiquement une sensibilité préférentielle à l'oeil droit puis au gauche. Ces colonnes ont été appelées les colonnes de dominance oculaire. Lorsque l'on se déplace orthogonalement à notre premier mouvement, on remarque d'autres colonnes qui sont sensibles, elles, à l'orientation d'un stimulus dans le champ de vision. De plus, deux colonnes voisines sont sensibles à des angles d'orientation voisins. On dispose ainsi d'une continuité spatiale de la réponse du cortex visuel à des stimulations proches (Hérault et Jutten, 1994).

I.3.3. La plasticité synaptique

Dans les deux sections précédentes, certaines propriétés des neurones et de leur organisation ont été présentées. Cependant, il existe, pour les mêmes architectures de réseaux de neurones, des disparités suivant les individus. Ces disparités sont dues à une propriété importante du système nerveux, 'la plasticité synaptique'. Cette terminologie recouvre la faculté d'évolution des systèmes nerveux liée aux notions d'apprentissage et de mémoire.

Le système nerveux se caractérise à la naissance par une interconnexion complexe et non organisée des réseaux neuronaux due au précâblage inné redondant. Ces connexions vont évoluer en fonction des stimulations extérieures durant la phase de maturation par sélection, élimination ou renforcement, afin d'obtenir une spécialisation des circuits neuronaux jusqu'à obtenir la stabilisation de ces circuits par plasticité en phase de maturation. On sait notamment que si l'édification du cortex débute au cours de la vie embryonnaire, elle se poursuit longtemps après la naissance. Chez l'homme, par exemple, le cortex n'acquiert son

organisation définitive que vers l'âge de 10 ans (Kennedy et Dehay, 1993; Hérault et Jutten, 1994). L'environnement social a donc une grande influence sur le développement initial du cerveau (Bard, 1993).

D'autre part, lors d'observations suite à des lésions accidentelles de zones parfois importantes du système nerveux périphérique, on constate une récupération partielle ou totale de fonctions momentanément perdues par restauration des circuits. Cette caractéristique n'est cependant pas présente dans le système nerveux central (cerveau et moelle épinière) (Dusart *et al.*, 1993).

L'évolution des synapses, en cours de fonctionnement, constitue une caractéristique importante des systèmes nerveux qui intéresse particulièrement les informaticiens et automaticiens dans le contexte du traitement adaptatif de l'information.

Hebb (1949) a le premier modélisé l'évolution d'une connexion au sein d'un réseau de neurones et plus particulièrement le fonctionnement d'une cellule vis à vis de son environnement.

Considérons deux cellules, une émettrice, neurone cause et une réceptrice, neurone effet. Quand la cellule émettrice s'active et possède une connexion très forte avec le neurone effet, la cellule réceptrice s'active aussi. De surcroît, cette synapse doit être renforcée. Au contraire, dans le cas où un seul de ces neurones s'active, la connexion entre ces deux cellules n'est pas prépondérante dans le comportement de la cellule réceptrice. Dans la phase d'apprentissage, cette synapse reste inchangée.

En considérant le fait qu'un neurone peut se trouver dans deux états différents, neurone activé (A) et neurone inactivé (I), on peut schématiser la règle de Hebb par le tableau (I.1).

Neurone cause	Neurone effet	Règle de Hebb	Règle de R. et S.
A	A	renforcement fort	renforcement fort
I	A	pas d'évolution	affaiblissement fort
A	I	pas d'évolution	affaiblissement
I	I	pas d'évolution	affaiblissement

Tab. I.1. Règles de plasticité de Hebb et de Rauschecker et Singer.

L'application stricte de la règle de Hebb, qui ne prévoit que des renforcements des connexions synaptiques, conduit à la saturation du réseau en l'absence de phénomènes limitants.

La règle de Hebb a alors été modifiée par des travaux plus récents dont ceux de Rauschecker et Singer (1981) qui proposent les trois règles suivantes également synthétisées dans le tableau (I.1) :

- l'efficacité des synapses excitatrices augmente à chaque fois que les 'neurones causes' et effets sont actifs simultanément,
- l'efficacité d'une synapse excitatrice décroît si le 'neurone effet' est actif tandis que le 'neurone cause' est inactif,
- l'efficacité synaptique décroît lentement (oubli) indépendamment de l'activité du 'neurone cause' si le 'neurone effet' est inactif.

Ces relations sont valables pour les synapses excitatrices. Il existe également des synapses inhibitrices dont la réponse à un stimulus fait intervenir plusieurs synapses. L'étude du fonctionnement de ces synapses est donc beaucoup plus complexe. Cependant Beaugé (1995) a étudié l'intégration de ce type de synapses dans les réseaux de neurones artificiels.

Dans le cas des neurones formels, on dit que le fonctionnement des connexions suit une loi de Hebb si la variation temporelle d'une synapse w_{ij} entre un neurone i et un neurone j en fonction des activités s_j et s_i des neurones cause et effet est de la forme :

$$dw_{ij} / dt = \mu s_i s_j \qquad (I.28)$$

où μ est le paramètre de modification des poids pouvant varier au cours de l'apprentissage.

Cette relation a été très utilisée dans les réseaux de neurones artificiels. En effet, elle signifie que la moyenne de la variation d'une synapse est égale à l'intercorrélation entre les activités des neurones cause et effet (Hérault et Jutten, 1994).

I.4. Les principaux réseaux de neurones artificiels

Nous avons décrit succinctement dans le paragraphe précédent le fonctionnement des neurones biologiques et de leurs connexions. Ces enseignements obtenus par les biologistes sont utilisés dans le domaine neuromimétique depuis plus de cinquante ans et ont donné lieu à plusieurs types de réseaux de neurones formels pour des applications aussi diverses que la classification, la modélisation, la reconnaissance de forme... Dans ce paragraphe, nous allons présenter les principaux réseaux ainsi créés. Pour construire ce paragraphe, nous nous sommes inspirés des livres de Davalo et Naïm (1989) et Hérault et Jutten (1994) ainsi que de la thèse de Theilliol (1993).

I.4.1. Les premiers réseaux neuronaux

La première modélisation d'un neurone, appelée neurone formel, a été proposée par McCulloch et Pitts (1943). Ce neurone réalise une somme pondérée des potentiels d'action lui parvenant et s'active en fonction de la valeur de cette somme. Si cette somme dépasse un

certain seuil, le neurone s'active et sa sortie prend la valeur 1, sinon, le neurone reste inactivé et sa sortie garde la valeur 0.

A partir de ce neurone formel et de la règle d'activation des poids proposée par Hebb (1949), Rosenblatt (1959) a proposé le premier réseau neuronal appelé Perceptron. Le perceptron est organisé en trois couches successives :

- la première couche, servant à recueillir les informations, est appelée rétine,
- la seconde couche est appelée couche d'association,
- la dernière couche, appelée couche de décision, comprend un ou plusieurs neurones.

Les poids des connexions entre les neurones de la rétine et de la couche d'association sont fixés initialement et n'évoluent pas au cours de l'apprentissage. Les fonctions réalisées par les neurones d'association sont des fonctions booléennes comme la fonction 'ET'. Les neurones de la couche de décision comportent des fonctions d'activation à seuil. Un exemple de perceptron est présenté à la figure (I.4). Le perceptron est un classifieur linéaire et Minsky et Papert (1969) ont montré les limites d'une telle architecture, notamment par l'exemple classique du ou exclusif. L'algorithme d'apprentissage du perceptron permet donc de converger vers la solution du problème de classification posé si les diverses classes sont linéairement séparables.

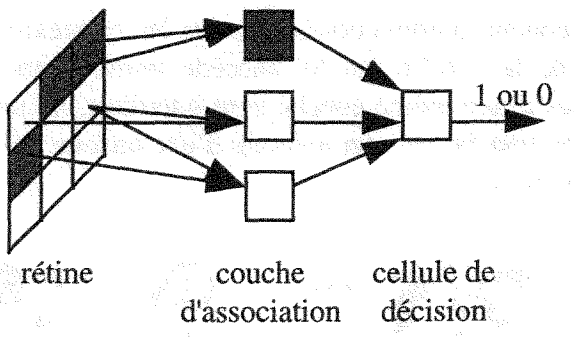


Fig. I.4. Exemple de perceptron.

Dans le perceptron de Rosenblatt, la modification des poids est calculée à partir d'un signal d'erreur établi par la différence entre la sortie des neurones de la couche de décision et le résultat attendu. Widrow et Hoff (1960) ont proposé une règle d'ajustement des poids w_i qui tient compte de la somme pondérée non seuillées $\sum_{i=1}^{n_a} w_i s_i$ des entrées s_i des neurones de la couche de décision et du résultat souhaité d. La règle dite de Widrow-Hoff s'écrit alors :

$$\Delta w_i = \mu(d - \sum_{i=1}^{n_a} w_i s_i) s_i \qquad (I.29)$$

A partir de cette règle, Widrow et Hoff ont réalisé en 1960 une machine nommée ADALINE pour ADAptive LInear NEuron, rebaptisée plus tard ADAptive LINear Element. Le réseau ADALINE, bien que n'étant plus utilisé, a eu des applications variées notamment en

traitement du signal (Widrow et Stearns, 1985). Plusieurs extensions du réseau ADALINE, comme le modèle MADALINE, ont également été proposées (Widrow et Lehr, 1990).

I.4.2. Les réseaux neuronaux multicouches

Au sein d'une architecture de réseau multicouches, les entrées, les sorties et les états des neurones sont à valeurs réelles. Les neurones utilisés effectuent la somme pondérée de l'ensemble de leurs entrées qu'ils transforment ensuite à l'aide d'une fonction mathématique continue et continûment dérivable. Ces neurones sont donc de type sommateur à fonction d'activation. Le plus fréquemment la fonction d'activation retenue est la même pour tous les neurones d'une même couche.

Les neurones de la couche d'entrée n'effectuent aucun traitement sur les données, mais distribuent uniquement les entrées aux neurones de la couche suivante, la première couche cachée.

I.4.2.1. Les réseaux neuronaux multicouches sigmoïdaux unidirectionnels

Dans un réseau multicouche unidirectionnel, seules les connexions entre neurones d'une couche et les neurones de la couche qui lui succède immédiatement sont autorisées. Les connexions entre neurones d'une même couche sont interdites, de même que les connexions d'un neurone situé sur une couche vers un neurone d'une couche précédente. Un exemple de réseau est présenté à la figure (I.5).

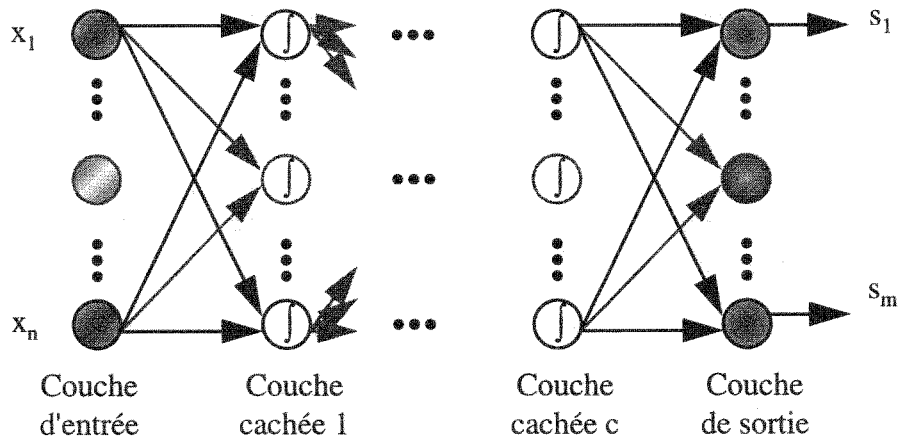


Fig. I.5. Réseau multicouche unidirectionnel à c couches cachées.

Les fonctions d'activation des neurones cachés sont des fonctions continues, dérivables et continûment croissantes comme la fonction sigmoïde dont un exemple est présenté à la figure (I.6).

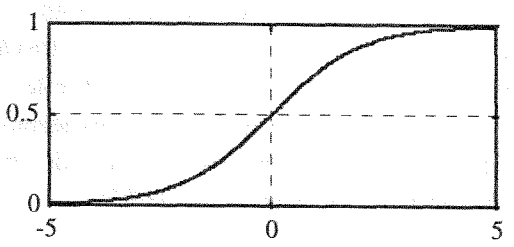


Fig. I.6. Fonction sigmoïde.

Ce choix satisfait aux exigences de la mise en oeuvre de la règle d'apprentissage des poids liée à ce type de réseau et appelée algorithme de rétropropagation du gradient. Cet algorithme également nommé 'backpropagation' est une généralisation de la règle de Widrow-Hoff pour les réseaux multicouches. Le principe de cet algorithme, élaboré par Rumelhart et McClelland (1986), est présenté en détail dans le chapitre IV.

Les fonctions d'activations des neurones de la couche de sortie peuvent être des fonctions sigmoïdes, comme pour les couches cachées, ou des fonctions identités.

I.4.2.2. Les réseaux neuronaux multicouches sigmoïdaux récurrent

Contrairement aux réseaux précédents, dans les réseaux récurrents, les connexions entre neurones d'une même couche (généralement bouclage sur le même neurone) et les connexions entre des neurones d'une couche et ceux de la couche précédente sont autorisées. Deux exemples de réseaux récurrents sont présentés à la figure (I.7). Plusieurs travaux ont porté sur ce type de réseaux (Elman, 1990; Funahashi et Nakamura, 1993; Pham et Liu, 1993; Delgado *et al.*,1995).

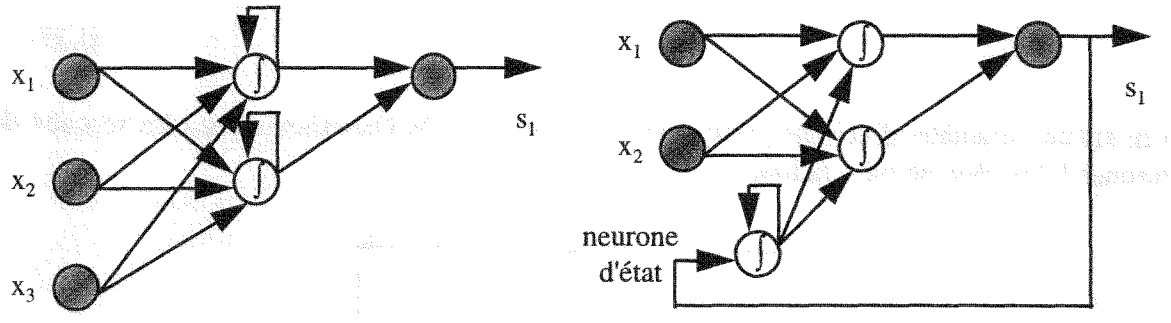


Fig. I.7. Exemples de réseaux de neurones multicouches récurrents.

Comme pour les réseaux non récurrents, les neurones de la couche d'entrée n'effectuent aucun traitement sur les données, mais se contentent de les distribuer aux neurones de la couche suivante. Les neurones des diverses couches cachées effectuent la somme pondérée de leurs entrées qu'ils transforment ensuite par une fonction d'activation de type sigmoïdale présentée à la figure (I.6). Les neurones de sorties utilisent à nouveau comme fonctions d'activation des fonctions sigmoïdes ou des fonctions identités.

Ces réseaux de neurones sont particulièrement utilisés pour l'identification de modèles d'états non-linéaires. Il est en général plus compliqué de travailler avec des structures récurrentes. En effet, il devient difficile de savoir pour quelles conditions le modèle de prédiction obtenu est stable. De surcroît l'utilisation de telles structures pose de plus grandes difficultés dans le calcul du gradient du résidu (erreur de modélisation) utilisé pour l'estimation des paramètres du modèle (Sjöberg *et al.*, 1995).

I.4.2.3. Les réseaux neuronaux multicouches à fonction de base radiale

Les réseaux de neurones à fonction de base radiale (radial basis function networks) ont été proposés par Broomhead et Lowe (1988), Moody et Darken (1989) et Girosi et Poggio (1990). Ce sont, avec les réseaux de neurones multicouches non récurrents, les plus utilisés actuellement. Ce type de réseau n'est constitué que de trois couches. La première, appelée couche d'entrée, ne fait, comme pour les autres réseaux multicouches, que présenter les entrées du réseau aux neurones de la couche cachée.

Les neurones de la couche de sortie effectuent la somme pondérée des sorties des neurones i de la couche cachée s_i.

$$S_1 = \sum_{i=1}^{n_c} w_i s_i \tag{I.30}$$

Les neurones i de la couche cachée vont commencer par calculer la distance Euclidienne entre le vecteur d'entrée ϕ et un vecteur de paramètre appelé centre c_i. Le résultat de cette distance est ensuite généralement transformé par une fonction d'activation Gaussienne dont un exemple est présenté à la figure (I.8). La sortie s_i du neurone i de la couche cachée s'écrit :

$$s_i = e^{(-(\|\phi - c_i\|^2)/\sigma_i^2)} \tag{I.31}$$

où σ_i est un paramètre d'échelle parfois appelé 'largeur' de la Gaussienne dans les réseaux de neurones à fonction de base radiale.

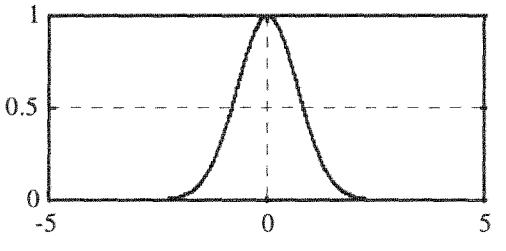


Fig. I.8. Fonction d'activation Gaussienne.

Ce type de réseau peut être utilisé, tout comme les réseaux multicouches sigmoïdaux non-récurrent, aussi bien pour effectuer une modélisation de systèmes non-linéaires que pour

résoudre des problèmes de classification ou de reconnaissance de formes. Cependant, les réseaux neuronaux à fonction de base radiale nécessitent généralement un grand nombre de paramètres lorsque l'espace des entrées est de grande dimension (Warwick, 1996).

I.4.3. D'autres réseaux

Les réseaux neuronaux multicouches de tous types confondus sont actuellement utilisés dans la quasi-totalité des applications des réseaux de neurones. Cependant, d'autres modèles ont été proposés. Ces modèles sont généralement dédiés à des problèmes de classification. Nous allons les présenter succinctement.

Hopfield (1982) a considéré que le système nerveux recherche des états stables, attracteurs, dans son espace d'état. Ainsi, une forme mémorisée est retrouvée par une stabilisation du réseau. Chaque neurone du réseau peut prendre deux valeurs de sorties 0 ou 1.

Les n neurones constituant le réseau sont tous interconnectés entre eux et la sortie s_i du neurone i est déterminée par :

$$\begin{cases} s_i = 1 & \text{si } \sum_{j=1}^{n} w_{ij}\, s_j > 0 \\ s_i = 0 & \text{sinon} \end{cases} \qquad \text{(I.32)}$$

où w_{ij} représente le poids de la connexion entre le neurone i et le neurone j et s_j est l'état du neurone j ($w_{ij} = w_{ji}$).

L'état du réseau est alors caractérisé par les sorties des n neurones constituant le réseau et peut être représenté par un mot de n éléments binaires. Un réseau de Hopfield constitue donc une mémoire adressable par son contenu.

Les valeurs des connexions vont constituer une mesure de la corrélation entre les états des neurones sur l'ensemble des exemples à mémoriser. Les réseaux de Hopfield permettent de réaliser des mémoires auto-associatives : chaque état du réseau correspond à une forme, un mot, ..., partiellement bruité ou incomplet que le réseau pourra ainsi reconnaître.

Kohonen (1988) a proposé une architecture de réseau très générale qui peut, suivant les applications, être plus précisément définie. Ce réseau, constitué de k neurones et n entrées, est présenté à la figure (I.9).

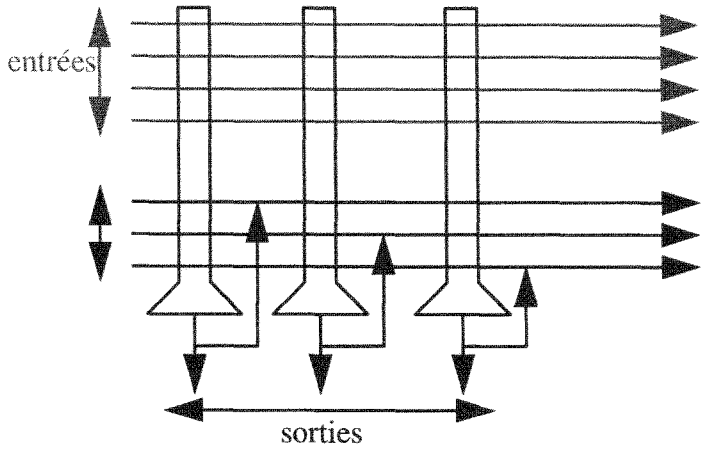


Fig. I.9. Architecture d'un réseau de Kohonen.

Toutes les entrées sont connectées à l'ensemble des neurones par nk connexions modifiables qui sont initialisées aléatoirement. Les neurones du réseau sont placés dans un espace (ici, 1 dimension) ce qui implique que chaque neurone possède des voisins. Enfin, la sortie de chaque neurone est connectée aux autres neurones du réseau par des connexions internes déterminées suivant la fonction dites du 'chapeau mexicain' présentée à la figure (I.10).

La détermination des valeurs des connexions internes s'effectue de telle sorte que :

- les neurones qui sont proche du neurone concerné ont une action excitatrice sur ce dernier;
- les neurones se trouvant dans un voisinage plus lointain possèdent une action inhibitrice sur le neurone concerné;
- l'action des neurones encore plus lointains est négligeable.

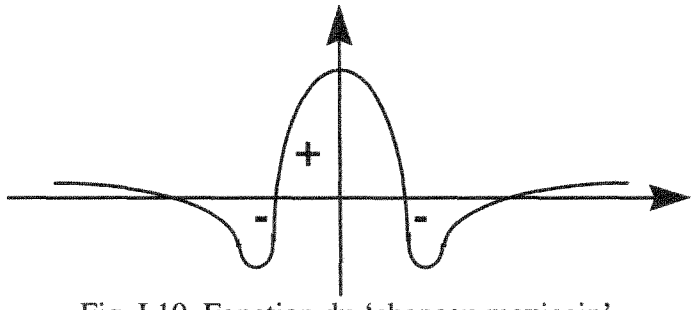


Fig. I.10. Fonction du 'chapeau mexicain'.

Initialement, les sorties des neurones sont uniquement fournies par la somme pondérée des entrées du réseau. Selon l'entrée x et les poids initiaux aléatoirement choisis, un neurone fournira la plus grande sortie et sera appelé neurone gagnant (Hérault et Jutten, 1994). Par la suite, l'action des connexions internes du réseau va faciliter l'activité des neurones voisins du neurone gagnant tandis que celle des neurones éloignés va être inhibée. Le réseau va donc

s'auto-organiser et va fournir une répartition des sorties telle qu'il se crée un amas de neurone d'activation élevé autour du neurone gagnant.

Le réseau de Kohonen, aussi appelé carte auto-organisatrice de Kohonen, va donc permettre d'effectuer un apprentissage non supervisé par compétition entre les différents neurones du réseau.

La construction de filtres à partir de réseaux de Kohonen permet de construire des modules de compression de données et d'améliorer la résistance au bruit pour des problèmes de mémoires auto-associatives.

Carpenter et Grossberg (1987) ont proposé un autre type de réseau appelé ART (Adaptive Resonance Theory). Ce réseau permet de reconnaître, sans superviseur, l'appartenance à différentes catégories des différents vecteurs d'entrées et est capable de s'adapter en créant de nouveaux neurones dans le cas ou un vecteur d'entrée n'appartient à aucune classe déjà définie.

I.5. Conclusion

Ce chapitre nous a permis d'effectuer une présentation succincte des différents types de modèles non-linéaires existants ainsi que les difficultés de mise en œuvre qui ont été relevées par leurs utilisateurs. Nous avons également rappelé les origines biologiques des réseaux de neurones et nous avons fini ce chapitre en présentant les principaux modèles neuronaux existant.

Dans ce mémoire, nous nous restreindrons à l'identification de systèmes SISO (simple entrée - simple sortie) et MISO (multi entrée - simple sortie) à l'aide de réseaux de neurones multicouches non-récurrent. Nous proposons de replacer ce modèle dans le cadre de la démarche traditionnelle d'identification de modèles de comportement de systèmes dynamiques non-linéaires. Nous présenterons également les quatre problèmes liés à l'utilisation de ce type de modèles qui sont, la détermination de la structure du réseaux de neurones, l'estimation et l'initialisation des paramètres ainsi que la robustesse aux valeurs aberrantes. Nous nous intéresserons tout particulièrement aux trois derniers problèmes et nous proposerons à chaque fois plusieurs méthodes permettant de les résoudre que nous comparerons.

Dans le chapitre suivant, nous présenterons plus précisément l'architecture du réseau que nous allons utiliser, puis les méthodes permettant de déterminer la structure du réseau (nombre de neurones cachés, choix des fonctions d'activation ...). Nous montrerons ensuite les liens étroits qui existent entre les modèles neuronaux multicouches et les modèles traditionels, et pour finir, les différents indicateurs de qualité du modèle utilisables pour les réseaux neuronaux seront énumérés.

I.6. Références

ALPERT P. (1965) 'A consideration of the discrete Volterra series', *IEEE Trans. On Automatic Control*, Vol.10, 322-327.

BANKS S.P. (1988) *Mathematical theories of nonlinear systems,* Prentice Hall, Englewood Cliffs, N.J., U.S.A.

BARD K.A. (1993) 'Nouveau-nés humains et chimpanzés: des comportements identiques', *La Recherche*, Vol.24, 250, 88-89.

BEAUGE L. (1995) *Définition de mécanismes de mémorisation pour les systèmes neuromimétiques - inspiration pluridisciplinaire*, Thèse de doctorat de l'université Henri Poincaré Nancy I, spécialité informaltique.

BENVENISTE A., A. JUDITSKY, B. DELYON, Q. ZHANG, P-Y. GLORENNEC (1994) 'Wavelets in identification', *preprints of 10th IFAC Symp. on System Identification SYSID '94,* Copenhagen, Denmark, 4-6 july, Vol.1, 27-48.

BILLINGS S.A. (1980) 'Identification of non-linear systems - a survey', *IEE Proc. Pt. D,* Vol.127, 6, 272-285.

BISSESSUR Y., R.N.G. NAGUIB (1996) 'Buried plant detection : A volterra series modelling approach using artificial neural networks', *Neural Networks*, Vol.9, 6, 1045-1060.

BREIMAN L. (1993) 'Hinging hyperplanes for regression, classification and function approximation, *IEEE. Trans. Inf. Theory*, Vol.39, 999-1013.

BROOMHEAD D.S., D. LOWE (1988) 'Multivariable functional interpolation and adaptive networks', *Complex Systems*, Vol.2, 321-355.

CARPENTER G.A., S. GROSSBERG (1987) 'ART2 : self-organization of stable category recognition codes of analog input patern', *Applied Optics*, Vol.26, 23, 4919-4930.

CHEN S., S. BILLINGS, P. GRANT (1990) 'Non-linear systems identification using neural networks', *International Journal of Control*, Vol.51, 1191-1214.

CHEN S., S. BILLINGS (1992) 'Neural networks for nonlinear dynamic systems modelling and identification', *International Journal of Control*, Vol.56, 319-346.

CYBENKO G. (1989) 'Approximation by superpositions of sigmoidale function', *Mathematics of Control Signal and Systems*, Vol.2, 303-314.

DAVALO E., P. NAÏM (1989) *Des réseaux de neurones,* Eyrolles, Paris, France.

DELGADO A., C. KAMBHAMPATI, K. WARWICK (1995) 'Dynamic recurrent neural network for system identification and control', *IEE Proc. Control Theory Appl.*, Vol.142, 4, 307-314.

DUSART I., B.C. RUBIN, M.E. SCHWAB (1993) 'La régénération des fibres nerveuses', *La Recherche.*, Vol.24, 258, 1068-1074.

ELMAN J.L. (1990) 'Finding structure in time', *Cognitive Sci.*, Vol.14, 179-211.

FUNAHASHI K. (1989) 'On the approximate realization of continuous mappings by neural networks', *Neural Networks*, Vol.2, 183-192.

FUNAHASHI K., Y. NAKAMURA (1993) 'Approximation of dynamical systems by continuous time recurrent neural networks', *Neural Networks*, Vol.6, 801-806.

GIROSI F., T. POGGIO (1990) 'Neural networks and the best approximation property', *Biol. Cybernetics*, Vol.63, 169-176.

GLORENNEC P.Y. (1993) 'a general class of fuzzy inference systems', *Proc. of* CES2, Prague, Tchécoslovaquie, Vol.3, 1039-1048.

GRIGNON G. (1996) *cours d'histologie,* Ellypse, Paris, France.

HEBB D.O. (1949) *The organization of the behaviour,* John Wiley, New-York, U.S.A.

HERAULT J., C. JUTTEN (1994) *Réseaux neuronaux et traitement du signal,* Hermès, Paris, France.

HOPFIELD J.J. (1982) 'Neural networks and physical systems with emergent collective computational abilities', *Proc. of the National Academy of SciencesUSA*, Vol.79, 2554-2558.

IVAKHNENKO A.G. (1968) 'Group method of data handling - a rival of the method of stochastic approximation', *Soviet Automatic Control*, Vol.13, 3, 43-71.

ISIDORI A. (1989) *Nonlinear control systems,* Springer Verlag, Berlin, Allemagne.

KENEDY H., C. DEHAY (1993) 'Le développement du cortex cérébral', *La Recherche*, Vol.24, 251, 132-141.

KOHONEN T. (1988) 'An introduction to neural computing', *Neural Networks*, Vol.1, 3-16.

KUNG S. (1993) *Digital neural networks*, Prentice Hall, Englewood Cliffs, N.J., U.S.A.

LEE C. (1990) 'Fuzzy logic in control systems, part i and ii', *IEEE trans. on Systems Man. and Cybernetics*, Vol.20.

LJUNG L. (1987) *System Identification - Theory for the users*, Prentice Hall, Englewood Cliffs, N.J., U.S.A.

MCCULLOCH W.S., W.H. PITTS (1943) 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematics and Biophysics*, Vol.5, 115-133.

MEYER Y. (1990) *Ondelettes et opérateurs*, Hermann, Paris, France.

MINSKY M.I., S. PAPERT (1969) *Perceptrons : an introduction to computational geometry*, MIT press, Cambridge, Massachusetts.

MOODY J., C. DARKEN (1989) 'Fast learning in networks of locally-tuned processing units', *Neural Computation*, Vol.1, 281-294.

NARENDRA K.S., P.G. GALLMAN (1966) 'An iterative method for the identification of the nonlinear systems using Hammerstein model', *IEEE Trans. Auto. Control*, Vol.12, 546-550.

NIJMEIER H., A.J. VAN DER SCHAFT (1990) *Nonlinear dynamical control systems,* Springer Verlag, Berlin, Alllemagne.

OULADSINE M. (1993) *Identification des systèmes dynamiques multi-variables*, Thèse de doctorat de l'Université de Nancy I, spécialité Automatique, France.

OULADSINE M., A. KOBI, J. RAGOT (1994) 'Identification using Hammerstein model', *Journal A*, Vol.35, 2, 9-16.

PATRA A., H. UNBEHAUEN (1993) 'Nonlinear modelling and identification', *Proc Conf. on Systems Man. and Cybernetics*, Le Touquet, France, 17-20 Oct., Vol.3, 441-446.

PHAM D.T., X. LIU (1993) 'Identification of linear and nonlinear dynamic systems using recurrent neural networks', *Artificial Intelligence in Engineering*, Vol.8, 1, 67-75.

RAUSCHEKER J.P., W. SINGER (1965) 'The effects of early visual experience on the cat's visual cortex and their possible explanation by Hebb synapses', *Journal of Physiol.*, Vol.310, 215-239.

RICHARD A. (1993) *Modélisation et Identification de Signaux et de Systèmes Industriels*, habilitation à Diriger des Recherches, Nancy, France.

ROSENBLATT F. (1959) *Principles of neurodynamics,* Spartan Press, Washington, U.S.A.

RUMELHART D.E., J.L. MCCLELLAND (1986) *Parallel distributed processing*, MIT press, Cambridge, Massachusetts.

SHETZEN M. (1965) 'Measurement of kernels of a nonlinear systems of finite order', *International Journal of Control*, Vol.1, 251-263.

SHETZEN M.(1980) *The Volterra and Wiener theories of nonlinear systems,* John Wiley, New-York, U.S.A.

SILVERMAN B. (1986) *Density estimation for statistics and data analysis,* Chapman and Hall, London, R.U.

SINHA N.K., G.P. RAO (1991) *Identification of continuous time systmes,* Kluwer Academic, Dordrecht.

SJÖBERG J., Q. ZHANG, L. LJUNG, A. BENVENISTE, B. DELYON, P.Y. GLORENNEC, H. HJALMARSSON, A. JUDITSKY (1995) 'Nonlinear black-box modeling in system identification : A unified overview', *Automatica,* Vol.31, 12, 1691-1724.

SÖDERSTRÖM T., P. STOICA (1989) *System Identification,* Prentice Hall, Englewood Cliffs, N.J., U.S.A.

TAKAGI T., M. SUGENO (1985) 'Fuzzy identification of systems and its application to modelling and control', *IEEE Trans. on Systems Man and Cybernetics,* Vol.12, 6, 903-907.

THEILLIOL D. (1993) *Identification de systèmes SISO linéaires et non linéaies par réseaux de neurones multicouches,* Thèse de doctorat de l'université Nancy I, spécialité automatique.

UNBEHAUEN H., G.P. RAO (1987) *Identification of continuous systems,* North Holland, Amsterdam, Pays-Bas.

VICTOR J., R. SHAPLEY (1980) 'A method of nonlinear analysis in the frequency domain', *Biophysical Journal,* Vol.29, 459-484.

VOLTERRA V. (1930) *Theory of Functions,* Blackie.

VOLTERRA V. (1959) *Theory of functionnals and of integral and integro-differential equations,* Dover Publications, New-York, U.S.A.

WANG L. (1994) *Adaptive fuzzy systems and control : design and stability analysis,* Prentice Hall, Englewood Cliffs, N.J., U.S.A.

WARWICK K. (1996) 'System identification using neural networks', *in Identification in Engineering Systems,* The Cromwell Press Ltd, Broughton Gifford, G.B., 689-701.

WIENER N. (1958) *Nonlinear problems in random theory,* MIT press, Cambridge, Massachusetts.

WIDROW B., M.A. LEHR (1990) '30 years of adaptive neural networks : perceptron, madaline, and backpropagation', *IEEE Trans. on Neural Networks,* Vol.78, 9, 1415-1442.

WIDROW B., S.D. STEARNS (1985) *Adaptive signal processing,* Prentice-Hall, Englewood Cliffs, N.J.

WIDROW G., M.E. HOFF (1960) 'Adaptive switching circuits', *Institute of Radio Engineers Western Electric Show and Convention Record,* 23 Aout, Vol.4, 96-104.

WIGREN T. (1990) *Recursive identification based on the nonlinear Wiener model,* Doctoral thesis at Uppsala University, Suède.

ZBIKOWSKI R., A. DZIELINSKI (1995) 'Neural approximation: a control perspective', *Neural Network Engineering in Dynamic Control Systems,* K.J. Hunt, G.R. Irwin et K. Warwick eds., Springer-Verlag, London, G.B., 1-25.

ZHANG G., A. BENVENISTE (1992) 'Wavelet networks', *IEEE Trans. on Neural Networks,* Vol.3, 889-898.

Chapitre II

Choix de l'architecture du réseau

II.1. Introduction

Nous avons vu dans le premier chapitre qu'il existe différents types de modèles pour les systèmes non-linéaires, dont les réseaux de neurones. Cependant, la modélisation neuronale, d'origine biologique, regroupe sous une même appelation des familles de modèles différentes.

Parmi les modèles neuronaux existants, un certain nombre sont uniquement dédiés à des problèmes de classification ou de reconnaissance de formes qui sortent du cadre de cette étude. Seuls les réseaux multicouches (récurrents, non-récurrents ou à fonction de base radiale) peuvent être utilisés aussi bien pour des problèmes de classification ou de reconnaissance de formes que des problèmes d'identification de systèmes non-linéaires qui nous intéressent ici.

Les réseaux de neurones récurrents sont particulièrement adaptés à l'identification de modèles sous forme d'état. Ils permettent de surcroît la construction de modèles de simulation. Cependant, il est en général plus difficile de travailler avec des structures récurrentes car il devient difficile de savoir pour quelles conditions le modèle de prédiction obtenu converge. Enfin, l'utilisation de telles structures pose de plus grandes difficultés dans le calcul du gradient du résidu utilisé pour l'estimation des paramètres du modèle. De surcroît, les deux types de réseaux principalement étudiés actuellement sont les réseaux de neurones multicouches non-récurrents et les réseaux à fonction de base radiale. Nous n'étudierons donc pas précisément les réseaux les réseaux multicouches récurrents dans ce mémoire.

Dans le cas de ces réseaux, le choix des centres affecte directement l'efficacité du réseau tant en ce qui concerne le choix de leur nombre que de leurs positions. Il est donc essentiel de placer les centres dans l'espace des entrées aux endroits correspondant à une forte activité (généralement aux points de fonctionnement) (Warwick, 1996). D'autre part, ce type de réseau nécessite un grand nombre de paramètres lorsque l'espace du domaine d'entrée est de grande dimension (Narendra, 1994; Warwick, 1996). De plus, Sanchez *et al.* (1996) ont indiqué que les réseaux de neurones multicouches semblent plus adaptés à l'identification de systèmes non linéaires.

Nous avons donc restreint notre étude au problème de l'identification de systèmes non-linéaires à l'aide de réseaux de neurones multicouches non-récurrents. Cependant, une fois la structure générale ainsi définie, il nous reste à déterminer l'architecture précise que nous allons utiliser.

En effet, nous ne connaissons toujours pas le nombre de couches cachées nécessaires pour identifier un système non-linéaire. Nous ne savons pas comment déterminer simplement le nombre de neurones dans chacune des couches cachées. Nous ne savons pas quelle fonction d'activation inclure dans les neurones des couches cachées et de sortie. Enfin, dans le cas d'un système non-linéaire MIMO (multi-entrées, multi-sorties), nous ne savons pas s'il est préférable de construire un seul réseau modélisant l'ensemble des sorties en même temps ou, au contraire, s'il est préférable de construire un modèle neuronal pour chacune des sorties du système.

Nous allons, dans ce chapitre, nous efforcer de répondre à ces questions. Nous commencerons par nous intéresser aux problèmes du choix de l'architecture proprement dite et des choix des fonctions d'activation pour les neurones des couches cachées et de sortie. Une fois fixée l'architecture du réseau étudié, nous présenterons les liens importants qui existent entre une telle modélisation neuronale et les modèles paramétriques traditionnels. Finalement, nous présenterons les divers indicateurs de qualité, utilisables dans le cadre neuronal, permettant de tester les résultats des divers exemples d'identification traités aux chapitres suivants.

L'identification par réseaux de neurones utilise aussi bien une approche statistique qu'une approche neuromimétique. Aussi, diverses notions communes à ces deux domaines peuvent posséder plusieurs dénominations. Suivant Sjöberg *et al.* (1995), nous utiliserons indifféremment, au cours de ce mémoire, ces diverses dénominations dont nous présentons les équivalences :

estimation = apprentissage,
validation = généralisation,
structure du modèle = réseau,
données d'estimation = données d'apprentissage,
données de validation = données de généralisation,
sur-ajustement = sur-apprentissage,
paramètres = poids et biais.

II.2. Détermination de l'architecture du réseau

II.2.1. Cas des systèmes non-linéaires MIMO

Nous avons dit, en introduction, que dans le cas d'un système non-linéaire MIMO (multi-entrées, multi-sorties), nous ne savons pas, a priori, s'il est préférable de construire un seul réseau modélisant l'ensemble des sorties en même temps ou, au contraire, s'il est préférable de construire un modèle neuronal pour chacune des sorties du système.

Face à ce problème, une question que l'on doit se poser est : est-il possible de construire un modèle neuronal, dans le cas d'un système MIMO, pour chacune des sorties séparément ? Si on utilise un réseau de neurones multicouches récurrents, la réponse est négative, car l'estimation d'une sortie à l'instant k va nécessiter, par les connexions entre les neurones d'une couche et ceux de la couche précédente, l'estimation des autres sorties à l'instant précédent. Dans notre cas, où le réseau utilisé est un réseau multicouches non-récurrent, l'ensemble de l'information nécessaire pour estimer une sortie se retrouve dans les vecteurs d'entrée du réseau. Ces vecteurs doivent comprendre l'ensemble des entrées et des sorties retardées du système. A cette condition, il est tout à fait possible de construire un modèle neuronal par sortie du système.

Dans ce cadre, est-il préférable de construire un modèle neuronal pour chacune des sorties du système plutôt que de construire un modèle MIMO ? Pour répondre à cette question, nous

allons nous intéresser à certaines considérations pratiques en s'appuyant sur un exemple de simulation.

Pour construire cet exemple, nous allons identifier un système MIMO non-linéaire constitué de deux entrées et deux sorties décrit par les relations suivantes :

$$\begin{aligned} y_1(k) = {} & [0{,}8 - 0{,}5e^{-y_1^2(k-1)-y_2^4(k-2)}]y_1(k-1) \\ & -[0{,}3 + 0{,}9e^{-y_1^4(k-1)-y_2^2(k-2)}]y_1(k-2) \\ & +u_1(k-1) + 0{,}2u_2(k-2) + 0{,}1u_2(k-1)u_1(k-2) + e_1(k) \end{aligned} \qquad \text{(II.1a)}$$

$$\begin{aligned} y_2(k) = {} & \frac{0{,}2y_1(k-1) + 0{,}7y_2(k-2)}{1 + y_2^2(k-1)} \\ & +u_1^3(k-1) + u_2^2(k-1) + u_1^2(k-2)u_2^2(k-2) + e_2(k) \end{aligned} \qquad \text{(II.1b)}$$

où $u_1(k)$, $u_2(k)$ sont les entrées du système dont les sorties sont $y_1(k)$ et $y_2(k)$. Les bruits $e_1(k)$ et $e_2(k)$ sont des bruits normaux centrés de variances respectives 0,01 et 0,64.

Deux jeux de données sont créés pour l'identification et la validation du modèle. Pour chacun des jeux de données, les entrées u_1 sont constituées par des valeurs aléatoires uniformément distribuées entre -1 et 1, u_2 étant une séquence d'échelons d'amplitude aléatoire comprise entre -2 et 3 et de durée aléatoire comprise entre 5 et 10.

Nous allons construire trois modèles différents :
- NN1 : modèle MIMO décrivant en un seul réseau l'ensemble du système,
- NN21 : modèle MISO de la sortie 1,
- NN22 : modèle MISO de la sortie 2.

Les entrées de chacun de ces réseaux sont les entrées retardées du système $u_1(k-1)$, $u_2(k-1)$, $u_1(k-2)$, $u_2(k-2)$ et les sorties retardées $y_1(k-2)$, $y_2(k-2)$, $y_1(k-2)$ et $y_2(k-2)$.

Chacun des trois réseaux de neurones possède donc huit neurones sur sa couche d'entrée. Nous avons dans un premier temps déterminé le nombre optimal de neurones à introduire dans la couche cachée pour chacun des modèles en effectuant une validation croisée sur un jeu de données de validation. NN1, NN21 et NN22 nécessitent respectivement 12, 5 et 7 neurones dans leurs couches cachées.

Dans un second temps, nous allons nous intéresser au temps nécessaire pour effectuer l'identification de chacun de ces modèles. Le tableau II.1 présente ces temps calculés pour 1 itération et obtenus en utilisant le logiciel MATLAB sur un Pentium 90.

	NN1	NN12	NN22
temps	13,65"	1,37"	2,37"

Tab. II.1. Temps nécessaire pour effectuer 1 itération.

Ces premiers résultats montrent clairement qu'il est beaucoup plus intéressant du point de vue temps de calcul de construire un modèle neuronal pour chacune des sorties plutôt que de construire un seul modèle global.

Nous allons maintenant nous intéresser aux résultats de l'identification pour chacun des modèles. Le tableau II.2 présente les valeurs des critères résiduels V_{y1} et V_{y2} en identification et en validation obtenues pour les deux sorties pour chacun des modèles. Ces modèles ont été obtenues en effectuant plusieurs initialisations et en sélectionnant ceux fournissant les meilleurs résultats. Le critère V utilisé est le critère quadratique :

$$V = \frac{1}{n} \sum_{k=1}^{n} (y(k) - \hat{y}(k))^2 \qquad \text{(II.2)}$$

où $\hat{y}(k)$ est la sortie estimée par le réseau. Les valeurs théoriques vers lesquelles les critères V_{y1} et V_{y2} devraient tendre si la structure du système était parfaitement connue sont les valeurs de la variance des bruits e_1 et e_2, c'est à dire 0,01 et 0,64 respectivement.

		NN1	NN21	NN22
Identification	V_{y1}	0,0438	0,0138	
	V_{y2}	0,5916		0,745
validation	V_{y1}	0,0535	0,0454	
	V_{y2}	1,7792		1,1812

Tab. II.2. Valeurs, pour les 2 sorties, du critère V obtenues pour chaque modèle.

Ces résultats nous montrent que le modèle NN1 a eu tendance à effectuer un sur-apprentissage de la sortie y2 tout en ne permettant pas au critère V_{y1} de s'approcher de sa valeur théorique. Au contraire, l'utilisation de deux modèles MISO pour modéliser chacune des sorties permet d'obtenir des résultats sensiblement meilleurs. Norgaard (1996) explique ce type de résultat notamment par le fait que les variances des bruits de chacune des sorties peuvent être très différentes. Pour pallier ce problème, il propose d'introduire dans le critère à minimiser une pondération par la matrice de variance covariance des bruits (Norgaard, 1996). Cette matrice étant inconnue, il est nécessaire de l'estimer au cours de l'apprentissage.

Ces considérations nous ont conduit à ne présenter que des réseaux neuronaux multicouches non-récurrents fonctionnant avec une seule sortie. Cependant, si on le désire, il est très simple de modifier les algorithmes que nous présentons dans ce mémoire pour les adapter aux cas multi-sorties.

II.2.2. Choix du nombre de couches cachées

Une première limite des réseaux de neurones pour identifier des systèmes non-linéaires est représentée par l'indétermination sur le nombre de couches introduit dans le réseau. En effet, le nombre de connexions d'un réseau est fonction du nombre de couches cachées. Par

conséquent, en augmentant le nombre de couches cachées, nous devons accroître le nombre d'exemples à présenter au réseau afin de déterminer correctement les paramètres de ce dernier. Intuitivement, un nombre de couches cachées trop grand risque de réaliser un apprentissage 'par coeur', c'est-à-dire, d'apprendre le bruit et de faire ainsi disparaître la capacité de généralisation du modèle (Hérault et Jutten, 1994), et risque également d'introduire des problèmes numériques lors de l'apprentissage.

Dans le domaine de l'identification, un modèle de prédiction doit être capable d'identifier un système sans redondance paramétrique. En optimisant le nombre de couches cachées, il est possible de minimiser le nombre de paramètres d'un modèle neuronal décrivant un système.

Depuis que Hilbert (1900) a proposé une liste de 23 problèmes non résolus, de nombreux chercheurs se sont penchés sur le treizième qui nous intéresse plus particulièrement. Ce treizième problème concerne la possibilité de représenter toute fonction de plusieurs variables comme une superposition de fonctions utilisant moins de variables. Hilbert présuppose notamment qu'il existe des fonctions à trois variables qui ne peuvent pas être approchées par une superposition de fonctions à deux variables.

Kolmogorov (1957) a été l'un des premiers à infirmer cette supposition en démontrant que toute fonction continue de plusieurs variables (sur un espace des entrées borné) peut être représentée comme une superposition d'un petit nombre de fonctions à une seule variable.

Par la suite, Funahashi (1989), en se basant sur les travaux de Kolmogorov, a prouvé que toute fonction continue peut être approchée par un réseau de neurones sigmoïdal à k couches ($k \geq 3$) au sens de la norme L_2.

Simultanément, Cybenko (1989) a montré que toute fonction continue de plusieurs variables, toujours sur un espace des entrées borné, peut être approximée par une superposition de fonctions discriminantes à une seule variable dont la définition est :

Une fonction σ est discriminante sur l'espace I_n des entrées x de la fonction à plusieurs variables si la relation :

$$\int_{I_n} \sigma(y^T x + \theta) d\mu(x) = 0 \qquad \text{(II.3)}$$

implique que, pour tout $y \in \Re^n$ et $\theta \in \Re$, $\mu = 0$. μ est une mesure signée régulière de Borel.

Cybenko (1989) a remarqué d'autre part que la fonction sigmoïde est discriminante. La combinaison de ces deux résultats permet de montrer que toute fonction continue peut être approximée par un réseau de neurones à trois couches utilisant une fonction d'activation sigmoïdale pour les neurones de la couche cachée et une fonction d'activation linéaire pour les neurones de la couche de sortie.

Si ces résultats nous indiquent qu'il existe toujours un réseau permettant d'approximer une fonction non-linéaire donnée, ils ne nous fournissent pas d'indications quant au nombre de

neurones à introduire dans la couche cachée. De plus, il existe toujours le risque d'obtenir un minimum local.

Cependant, l'utilisation de réseaux de neurones incluant plus d'une couche cachée ne permet pas de résoudre plus simplement ces difficultés et elle complique, de surcroît, la détermination des poids initiaux comme nous le verrons dans le chapitre suivant.

Nous avons donc utilisé des réseaux multicouches possédant une seule couche cachée. Ces réseaux sont donc constitués de trois couches : une couche d'entrée, une couche cachée et une couche de sortie.

II.2.3. Choix du nombre de neurones

Après avoir fixé le nombre de couches, il reste à déterminer le nombre de neurones par couche afin de définir la structure du modèle neuronal. Nous n'avons pas effectué une étude approfondie de cette question. Cependant nous allons présenter succinctement quelques méthodes permettant d'y répondre.

Comme nous avons restreint notre étude aux systèmes SISO (simple-entrée, simple-sortie) et aux systèmes MISO (multi-entrées, simple-sortie), la couche de sortie est uniquement constituée d'un seul neurone.

Le nombre de neurones composant la couche d'entrée est fonction des valeurs passées des entrées et de la sortie du système à prendre en compte dans le modèle. Ce nombre peut être déterminé à partir d'une procédure classique inspirée des méthodes d'identification de systèmes linéaires. Certaines méthodes, moins classiques, proposées par Duong et Landau (1993a, b, c) peuvent également être employées afin de déterminer ce nombre avant l'apprentissage.

Tout d'abord, il faut déterminer le retard pur n_k du système. En effet, l'action d'une entrée u(k) sur la sortie y(k) d'un système n'est souvent pas immédiate. Si l'on ne dispose pas de connaissance a priori, ce retard doit être identifié en tant que tel. Borne *et al.* (1992) ont proposé plusieurs méthodes pour identifier ce retard pur. L'une de ces méthodes consiste dans le choix d'un nombre identique $n_a = n_b$ d'entrées et de sorties retardées du système utilisé dans le vecteur de régression. On va alors modéliser le système en choisissant $n_k = 0, 1, \ldots$ et la valeur estimée de n_k correspondra au modèle fournissant le plus petit critère résiduel. Cette procédure a posteriori est longue mais nécessaire lorsque n_k est inconnu (Theilliol, 1993).

A ce stade, il faut construire le vecteur de régression en déterminant les ordres n_a et n_b. Si les connaissances a priori du système ne nous fournissent pas d'indication sur ces valeurs, il est nécessaire de les estimer également. De nombreuses méthodes ont été proposées pour résoudre ce problème dans le cas linéaire. Ces méthodes peuvent généralement être adaptées à l'identification de systèmes non-linéaire par perceptron multicouches. L'une d'elle consiste à modéliser le système avec des ordres n_a et n_b croissant puis à utiliser un critère permettant de choisir l'ordre minimal (Theilliol, 1993). L'un de ces critères est le critère de l'erreur de

prédiction finale (FPE) développé par Akaïke (Ljung, 1987) et que nous présentons à la fin de ce chapitre. Une telle procédure est également longue et assez lourde d'emploi (Theilliol, 1993).

Comme nous l'avons vu au paragraphe précédent, l'introduction dans le modèle d'un trop grand nombre de paramètres risque de conduire à un sur apprentissage, et donc, de détruire les capacités de généralisation du réseau. Le nombre de paramètres est directement lié au nombre de neurones de la couche cachée.

Le choix du nombre de neurones dans l'unique couche cachée est généralement effectué de manière heuristique. Comme précédemment, il est nécessaire d'effectuer une approche par essai-erreur. Le nombre optimum de neurones est généralement déterminé en effectuant une validation croisée sur un jeu de données de validation. Il existe toutefois des algorithmes permettant de construire itérativement la couche cachée (Fahlman et Lebière, 1990; Bornholdt et Graudenz, 1992; Bartlett, 1994; Chentouf et Jutten, 1996; Lengellé et Denoeux, 1996; Mohraz et Protzel, 1996). Il est cependant nécessaire d'utiliser dans ce type d'algorithme une validation croisée afin de contrôler la croissance du réseau.

Afin de déterminer la taille optimale du réseau, il est possible de suivre une autre approche. Cette dernière consiste à partir d'un réseau utilisant un grand nombre de neurones cachés, puis, d'éliminer progressivement les connexions inutiles. Pour cela, diverses méthodes ont été mises au point. Une de ces méthodes est présentée à la section suivante.

II.2.4. Algorithme d'élimination des poids

Une fois choisie l'architecture du réseau multicouches non-récurrent, il faut estimer les paramètres du réseau. Cette phase peut s'effectuer à l'aide de plusieurs algorithmes de minimisation que nous décrirons au chapitre IV. Cependant, il est connu que les méthodes permettant de déterminer, avant cette phase, la structure minimale du réseau, peuvent inclure trop de paramètres.

Or il est particulièrement intéressant de déterminer cette structure minimale, et ce pour deux raisons. La première est que le modèle va être utilisé dans une loi de commande ou dans un algorithme de diagnostic d'un système industriel. Aussi est-il nécessaire que le calcul de l'estimation de la sortie se fasse le plus rapidement possible et que l'algorithme lui-même utilise la plus petite place mémoire possible. Ces deux conditions sont clairement antinomiques avec un modèle utilisant un nombre de paramètres trop élevé.

D'autre part, la modélisation d'un système avec un réseau de neurones possédant beaucoup trop de paramètres risque de conduire à un sur-apprentissage, c'est-à-dire que le réseau risque, dans ces conditions, d'apprendre le bruit. Une telle situation empêche de généraliser le modèle ainsi trouvé sur d'autres données que celles utilisées lors de l'apprentissage (Sjöberg *et al.*, 1995).

Cette structure minimale n'étant pas facilement déterminable a priori, de nombreux auteurs ont proposé de la rechercher après la phase d'apprentissage (Mozer et Smolensky, 1989; Abe *et al.*, 1990a, b; Le Cun *et al.*, 1990; Hergert *et al.*, 1992; Hassibi et Stork, 1992; Hansen et Pedersen, 1994; Cottrell *et al.*, 1995; Norgaard, 1995; Cibas *et al.*, 1996).

Le fonctionnement traditionnel d'un algorithme d'élimination des poids suit l'évolution suivante. Tout d'abord, on fait évoluer les poids du réseau jusqu'à l'obtention d'un minimum local ou jusqu'à ce que les performances de généralisation du réseau commencent à se détériorer sur le jeu de données de validation. On retire alors certaines connexions (les moins significatives) et on recommence l'apprentissage des poids en partant des poids restants précédemment trouvés.

Nous allons plus particulièrement présenter l'algorithme OBS (Optimal Brain Surgeon) de Hassibi et Stork (1992). Considérons un vecteur W constitué de l'ensemble des poids du réseau et calculons la sensibilité δE de l'erreur quadratique E au second ordre à une petite modification des poids δW :

$$\delta E = \delta W^T \nabla_w E + \frac{1}{2} \delta W^T H \delta W + o(\|W\|^2) \qquad \text{(II.4)}$$

où $\nabla_w E$ est le gradient de E et H est le Hessien de l'erreur quadratique par rapport aux paramètres W.

Après convergence, le gradient est nul, et l'équation (II.4) se réduit à :

$$\delta E = \frac{1}{2} \delta W^T H \delta W \qquad \text{(II.5)}$$

On cherche à supprimer le poids w_q qui possède la plus petite influence sur le modèle. En construisant le vecteur $e_q^T = (0, \cdots, 1, 0, \cdots, 0)$ dont la $q^{ème}$ composante vaut 1 et toutes les autres 0, on va chercher à obtenir la relation :

$$e_q^T (\delta W + W) = 0 \qquad \text{(II.6)}$$

en posant $\delta w_q = -w_q$. On peut alors effectuer la minimisation de la sensibilité δE de l'erreur quadratique E à une petite modification des poids δW sous la contrainte précédente :

$$L = \frac{1}{2} \delta W^T H \delta W + \lambda (e_q^T (\delta W + W)) \qquad \text{(II.7)}$$

où λ est un scalaire.

Les conditions de stationarité s'écrivent alors :

$$\frac{\partial L}{\partial \delta W} = H\delta W + \lambda e_q^T = 0 \qquad \text{(II.8a)}$$

et :

$$\frac{\partial L}{\partial \lambda} = e_q^T(\delta W + W) = 0 \qquad \text{(II.8b)}$$

L'équation (II.8a) conduit à une expression simple de δW que l'on peut introduire dans l'équation (II.8b) ce qui nous fourni la relation donnant λ :

$$\lambda = \frac{e_q^T W}{e_q^T H^{-1} e_q} = \frac{w_q}{H_{qq}^{-1}} \qquad \text{(II.9)}$$

En utilisant cette relation dans l'équation (II.8a), nous obtenons :

$$\delta W = -\frac{w_q}{H_{qq}^{-1}} H^{-1} e_q \qquad \text{(II.10)}$$

En pratique, le poids w_q éliminé est celui qui conduit à la plus petite valeur de l'équation (II.4). Après suppression de ce poids, on ajuste tous les poids restant à l'aide de l'équation (II.10) ce qui permet d'éviter un nouvel apprentissage après chaque suppression de poids.

Hansen et Pedersen (1994) ont présenté une évolution de cet algorithme qui a été implantée dans un critère régularisé par Norgaard (1995) et que nous utiliserons au chapitre V. Quel que soit l'algorithme utilisé, la condition d'arrêt d'une telle procédure n'est pas simple à déterminer. Pour cela, il est possible d'utiliser les critères de qualité présentés à la fin de ce chapitre ou encore d'effectuer une validation croisée qui permet de la déterminer.

Cotrell *et al.* (1993) propose d'éliminer les poids statistiquement nuls uniquement qi le réseau résultant produit de 'meilleurs' résultats que le précédent. La notion du qualité du modèle neuronal est alors définie comme le critère BIC (B Information Criterion) (Akaïke, 1974).

Pour cela, Cotrell *et al.* (1993) remarquent que le vecteur des paramètres estimés, obtenu après l'apprentissage, est un vecteur $\hat{W}$ de variables aléatoires connu comme l'estimateur de la solution théorique W*. Dans ce cadre, $\hat{W}$ suit asymptotiquement une distribution Gaussienne :

$$\sqrt{n}(\hat{W} - W^*) \xrightarrow[n\to\infty]{} N(0, \sigma_r^2 H^{-1}) \qquad \text{(II.11)}$$

où n est la taille du jeu de donnée d'apprentissage.

Ceci posé, il est possible de tester (statistiquement) la nullité de chacun des composants de $\hat{W}$ en utilisant le test traditionnel :

$$t_i = \left| \frac{\hat{w}_i}{\hat{\sigma}(\hat{w}_i)} \right| \qquad \text{(II.12)}$$

où $\hat{\sigma}_r^2$ est l'estimateur de σ_r^2, et la variance de $\hat{w}_i$ est obtenue en multipliant $\hat{\sigma}_r^2$ par le $i^{ième}$ terme de la diagonale de l'inverse de la matrice Hessienne.

$\hat{w}_i$ est alors considéré comme insignifiant lorsque $t_i < 1{,}96$ et il est éliminer uniquement si le critère BIC est améliorer :

$$BIC = \log(\frac{\sigma_r^2}{n}) + n_p \frac{\log(n)}{n} \qquad \text{(II.13)}$$

où n_p est le nombre de paramètres du modèle.

Afin de s'affranchir partiellement de la détermination du nombre de neurones dans la couche cachée, il est possible d'utiliser le principe de la régularisation pour préserver les capacités de généralisation du réseau. Nous allons présenter ce principe dans le paragraphe suivant.

II.2.5. Régularisation et généralisation

Pour modéliser un système, on utilise une structure de modèle dont on va déterminer les paramètres en minimisant les carrés des erreurs. Une fois l'apprentissage des paramètres effectué, la sortie du modèle ne reproduit toujours pas à l'identique la sortie du système et il existe toujours une erreur de modélisation.

Deux raisons sont à l'origine de cette erreur de modélisation (Norgaard, 1996).

Premièrement, la structure du modèle ne correspond généralement pas à la structure exacte du système. Cette partie de l'erreur est généralement appelée erreur de biais. Dans le domaine neuronal, nous savons notamment qu'un réseau de neurones est capable d'approcher toute fonction non-linéaire avec une précision arbitraire. Cette précision dépend du nombre de paramètres introduit dans le modèle. Plus nombreux sont les paramètres introduits dans le modèle, plus cette erreur de biais sera faible. Il est cependant évident que cette partie de l'erreur est inévitable.

La seconde origine de l'erreur de modélisation est due au fait que le jeu de données d'apprentissage est bruité et de taille limitée. Ceci implique que les paramètres obtenus lors de l'apprentissage seront différents des paramètres optimaux. Cette erreur est appelée erreur de variance.

Norgaard (1996) a montré que, si l'erreur de bais diminue avec l'accroissement de la taille du réseau, il n'en est pas de même pour l'erreur de variance qui, elle, a plutôt tendance à s'accroître. Il est donc nécessaire d'effectuer un compromis entre ces deux sources d'erreur lors du choix de la structure du modèle.

Une approche classique pour améliorer le compromis erreur de biais/erreur de variance consiste à additionner au critère à minimiser un terme de 'régularisation'. Ce terme de pénalité va favoriser le 'lissage' du modèle (Bishop, 1995).

Le principe de la 'régularisation' repose sur deux hypothèses, la première concernant la loi de distribution du bruit, la seconde concernant la loi de distribution des paramètres. Partant de ces deux hypothèses, il est possible d'écrire une fonction de vraissemblance à maximiser (Williams, 1995) :

$$P(\theta|D) = P(D|\theta)\, P(\theta) \qquad \text{(II.14)}$$

où $P(\theta|D)$ est la densité de probabilité a posteriori dans l'espace des paramètres, $P(D|\theta)$ est la vraissemblance des données D, et $P(\theta)$ est la densité de probabilité a priori des paramètres.

Maximiser cette fonction de vraissemblance revient à minimiser l'opposé de son logarithme. L'hypothèse classique d'une distribution du bruit Gaussienne associée à l'hypothèse d'une distribution des paramètres également Gaussienne nous conduit à la méthode de régularisation la plus couramment utilisée qui repose sur l'adjonction au critère (II.2) d'un terme d'affaiblissement des poids (weight decay term) :

$$W = V + \frac{1}{n}\theta^T D\theta = \frac{1}{n}\sum_{k=1}^{n}(y(k) - \hat{y}(k))^2 + \frac{1}{n}\theta^T D\theta \qquad \text{(II.15)}$$

où D est une matrice généralement choisie diagonale, avec $D = \alpha I$, où I est la matrice identité et α le paramètre d'affaiblissement des poids qui est difficile à régler. Jutten et Fambon (1995) proposent d'initialiser ce paramètre à 0 et de le faire évoluer au cour de l'apprentissage.

Parfois, on utilise un paramètre différent pour les poids connectant les neurones de la couche d'entrée aux neurones cachés, d'une part, et pour les poids connectant les neurones cachés aux neurones de sortie, d'autre part, ou encore un paramètre différent pour chacun des poids du réseau. La matrice D doit être déterminée par une stratégie d'essai-erreur.

D'autres hypothèses de distribution des paramètres peuvent être utilisées et sont présentées par Williams (1995).

Bishop (1995) rappelle également qu'il est possible d'effectuer une régularisation du réseau en arrêtant prématurément l'apprentissage ou encore en ajoutant du bruit sur les données d'entrées du réseau lors de l'apprentissage.

II.2.6. Choix des fonctions d'activation

Nous avons vu que dans les premiers réseaux de neurones, les fonctions d'activation utilisées étaient des fonctions 'seuils'. L'utilisation de telles fonctions non continûment dérivables dans les réseaux de neurones multicouches n'est plus possible, car les algorithmes d'estimation des paramètres de ces réseaux utilisent la dérivée des fonctions d'activation. Aussi, Rumelhart et McClelland (1986) ont proposé d'utiliser une fonction sigmoïde. Certains chercheurs ont proposé d'utiliser des fonctions d'activation différentes (Fombellida *et al.*, 1990; Elliott, 1993). D'autres encore ont étudié l'influence du choix de ces fonctions d'activation (Alippi, 1991).

Cependant, le choix de la fonction tangente hyperbolique, dont un exemple est présenté à la figure (II.1), s'est progressivement imposé pour les neurones de la couche cachée. Son équation s'écrit :

$$g(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1 \qquad \text{(II.16)}$$

Cette fonction, de type sigmoïdale, est continûment dérivable, croissante et possède un espace de sortie s'étendant de -1 à 1.

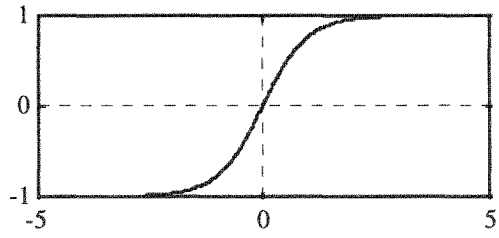


Fig. II.1. Fonction tangente hyperbolique.

Elle présente l'avantage d'avoir une dérivée pouvant s'écrire sous une forme très simple :

$$g'(x) = \frac{4e^{-2x}}{(1+e^{-2x})^2} = 1 - g^2(x) \qquad \text{(II.17)}$$

Dans le cadre de la modélisation, l'utilisation dans les neurones de sortie d'une fonction sigmoïdale imposera d'effectuer une normalisation des sorties désirées, ainsi qu'une dénormalisation de la sortie estimée par le réseau. Une telle normalisation impose d'introduire des paramètres supplémentaires ainsi qu'un traitement supplémentaire ce qui va augmenter le temps de traitement et la place mémoire utilisée par l'algorithme pouvant être préjudiciable lors de l'installation d'un tel modèle sur un système industriel.

Puisque Cybenko (1989) ont montré que l'utilisation de la fonction identité comme fonction d'activation du neurone de sortie préserve les capacités d'approximateur universel

des réseaux de neurones, nous effectuerons ce choix dans le modèle neuronal que nous étudierons.

II.2.7. Présentation de la structure sélectionnée

Au cours des sections précédentes, nous avons effectué un certain nombre de choix quant à la structure du réseau multicouches que nous proposons d'utiliser pour modéliser un système non linéaire.

Ces choix concernent aussi bien l'architecture du réseau utilisé que le nombre de couches cachées, le nombre de neurones dans la couche de sortie, ou le choix des fonctions d'activation utilisées. La figure (II.2) présente la structure que nous allons utiliser.

Un neurone dit, de biais, est généralement ajouté sur la couche d'entrée et la couche cachée afin d'améliorer la convergence lors de la phase d'apprentissage. L'entrée et la sortie de ce neurone valent 1. Au sein du réseau de neurones, chaque neurone de biais est uniquement connecté à la couche en aval de sa couche d'appartenance comme le montre la figure (II.2). Nous rappelons que la première couche est constituée de neurones "transparents" qui distribuent simplement les entrées du réseau aux neurones de la couche suivante, appelée couche cachée.

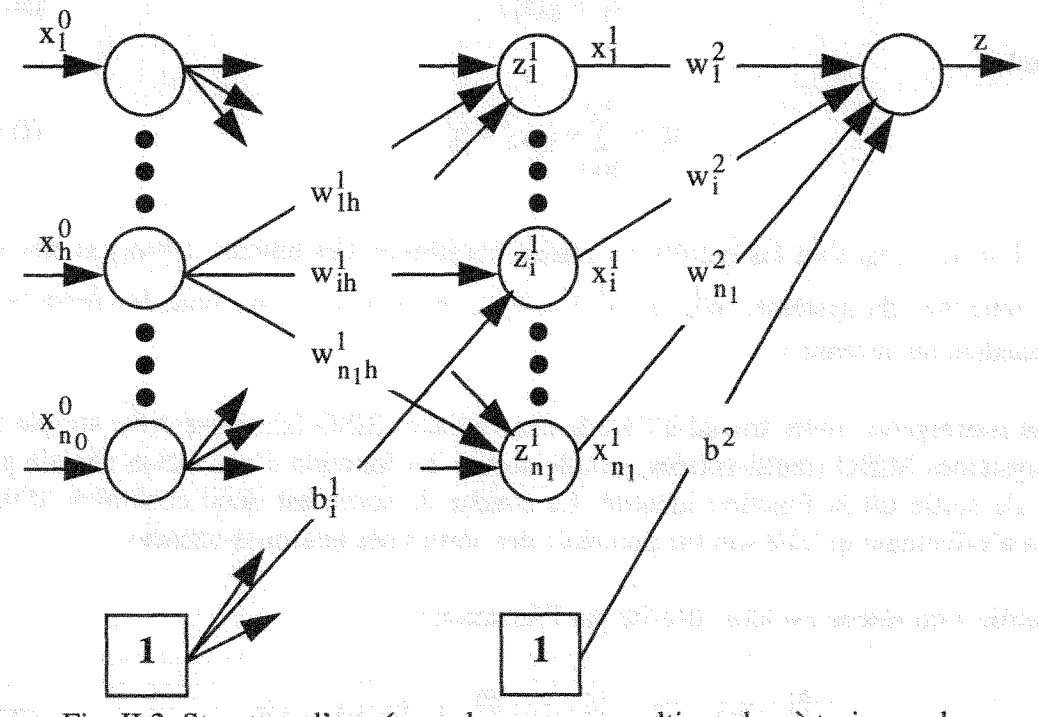


Fig. II.2. Structure d'un réseau de neurones multicouches à trois couches.

x_i^j étant la sortie du $i^{ième}$ neurone de la $j^{ième}$ couche. Les neurones de la couche cachée calculent dans un premier temps la somme pondérée des entrées du réseau, puis, transforment cette somme à l'aide d'une fonction d'activation g. Les neurones de la couche cachée sont représentés à la figure (II.3).

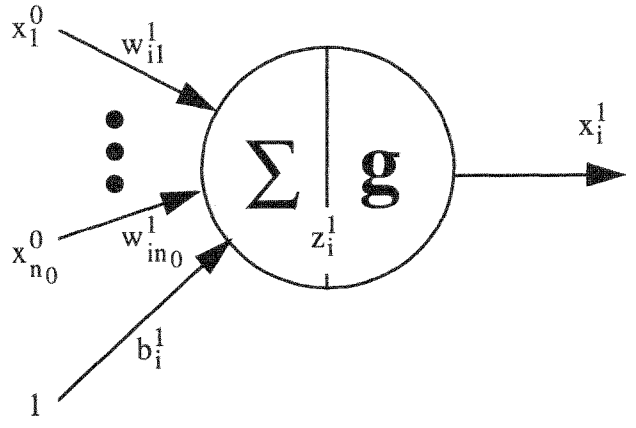


Fig. II.3. Neurone i de la couche cachée.

La fonction d'activation g choisie ici pour les neurones de la couche cachée est la fonction tangente hyperbolique : $g(x) = 2/(1+e^{-2x}) - 1$.

La sortie x_i^1 du neurone i de la couche cachée peut alors être décrite par l'équation :

$$x_i^1 = g(z_i^1) \tag{II.18}$$

où z_i^1 est :

$$z_i^1 = \sum_{h=1}^{n_0} w_{ih}^1 x_h^0 + b_i^1 \tag{II.19}$$

où x_h^0, h = 1, ..., n_0 sont les entrées du réseau constituées des entrées $x(k-n_b)$ et des sorties $y(k-n_a)$ retardées du système, w_{ih}^1 et b_i^1 i = 1, ..., n_1, h = 1, ..., n_0, sont les poids et biais correspondant au neurone i.

Nous restreignons notre travail à l'étude des systèmes SISO (simple-entrée, simple-sortie) et des systèmes MISO (multi-entrées, simple-sortie). La fonction d'activation choisie pour la couche de sortie est la fonction identité. La couche de sortie est donc constituée d'un seul neurone n'effectuant qu'une somme pondérée des sorties des neurones cachées.

La sortie z du réseau est alors décrite par l'équation :

$$z = \sum_{i=1}^{n_1} w_i^2 x_i^1 + b^2 = \sum_{i=1}^{n_1} w_i^2 g\left(\sum_{h=1}^{n_0} w_{ih}^1 x_h^0 + b_i^1\right) + b^2 \tag{II.20}$$

où x_i^1, i = 1, ..., n_1 sont les sorties des neurones cachés, w_i^2 et b^2, i = 1, ..., n_1, sont les poids et biais correspondant au neurone de sortie.

L'ensemble des poids et biais du réseau peuvent être regroupés dans le vecteur des paramètres θ.

Ce modèle neuronal que nous allons utiliser tout au long de notre étude sera appelé par la suite perceptron multicouches afin de pouvoir le différencier des autres modèles neuronaux.

II.3. Liens entre le perceptron multicouches et les modèles traditionnels

Nous avons vu, dans le précédent chapitre, que les réseaux neuronaux tirent leurs origines des études biologiques effectuées sur le système nerveux dans les années quarante. Cependant, nous utilisons les réseaux de neurones dans le cadre de l'identification de systèmes non-linéaires. Pour valider l'utilisation de telles stratégies, nous allons présenter les liens existant entre le perceptron multicouches utilisé et divers modèles traditionnels.

II.3.1. Liens avec le modèle ARX

Une première approche succincte peut être effectuée sur un modèle linéaire. Cette étude (Thomas, 1993) a été réalisée sur un exemple. Nous avons fait apprendre à un perceptron multicouches possédant un seul neurone sur sa couche cachée un système dynamique linéaire de type ARX dont on connaît parfaitement les paramètres.

Après convergence, nous constatons que la fonction d'activation du neurone de la couche cachée n'est excitée que dans sa partie linéaire. Il nous est alors possible d'établir une correspondance simple et directe entre modèle paramétrique et modèle neuronal.

En effet, puisque la fonction d'activation n'est excitée que dans sa partie linéaire, il nous est possible d'approcher cette fonction par sa pente à l'origine. Ceci revient en fait à approcher la tangente hyperbolique à la fonction identité.

En remplaçant la fonction d'activation par la fonction identité dans l'équation (II.20), on constate que la sortie estimée devient directement une somme pondérée des éléments du vecteur de régression.

Un exemple est présenté pour un système à deux entrées en n'utilisant qu'un seul neurone sur la couche cachée :

$$\begin{aligned} z &= w_1^2 w_{11}^1 x_1^0 + w_1^2 w_{12}^1 x_2^0 + w_1^2 b_1^1 + b^2 \\ &= a_1 x_1^0 + a_2 x_2^0 + a_3 \end{aligned} \qquad \text{(II.21)}$$

Il est notable que le réseau le plus simple que l'on puisse utiliser correspond déjà à une sur-paramétrisation par rapport au modèle ARX.

En effectuant le calcul, on a constaté que nous retrouvions bien les paramètres du modèle ARX de départ, et ce, avec des résultats tout à fait comparables à ceux obtenus par estimation des paramètres d'un modèle linéaire à l'aide de l'algorithme des moindres carrés.

Ces premiers résultats nous ont poussés à étudier plus précisément les relations existant entre le perceptron multicouches et les modèles paramétriques non-linéaires.

II.3.2. Liens avec les modèles paramétriques non-linéaires

Nous avons vu que les fonctions d'activation des neurones de la couche cachée sont excitées uniquement dans leurs parties linéaires lorsque le perceptron multicouches sert à modéliser un système linéaire. Ceci n'est pas le cas lors de la modélisation d'un système non-linéaire.

Le problème de la détermination des liens entre le perceptron multicouches et un modèle paramétrique devient donc plus complexe.

La difficulté du passage d'un modèle neuronal à un modèle paramétrique réside dans l'utilisation par le réseau de fonctions d'activation tangente hyperbolique. Nous pouvons cependant supposer qu'après apprentissage, les sommes pondérées z_i^1 en entrée de chaque neurone i sont comprises dans un espace restreint autour de 0.

En effet, si ce n'était pas le cas, nous nous trouverions face à un problème de saturation des fonctions d'activation, et donc, dans le cadre de l'identification, à un échec de l'apprentissage. Cette hypothèse nous autorise à utiliser un développement de Taylor autour de 0 de la fonction tangente hyperbolique :

$$g(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1 \approx g(0) + xg'(0) + \cdots + \frac{x^N}{N!} g^{(N)}(0) \qquad \text{(II.22)}$$

Ceci nous permet de réécrire la sortie du réseau sous la forme d'une somme pondérée de puissances et de produits termes à termes des éléments du vecteur de régression. Il est remarquable que La fonction tangente hyperbolique g est une fonction impaire, et donc, toutes les dérivées paires s'annulent dans ce développement de Taylor. Dans le cadre de la modélisation, à l'aide d'un perceptron multicouches, d'un système dynamique dont le vecteur de régression $\phi(k)$ est :

$$\begin{aligned} \phi(k) &= [y(k-1), ..., y(k-n_a), u(k-1), ..., u(k-n_b)] \\ &= [x_1^0, ..., x_m^0] \end{aligned} \qquad \text{(II.23)}$$

où $m = n_a + n_b$.

En remplaçant la fonction d'activation par son développement de Taylor dans l'équation (II.20), nous obtenons la relation :

$$
\begin{aligned}
z &= w_1^2[g(0) + g'(0)(\sum_{i=1}^{m} w_{1i}^1 x_i^0 + b_1^1) + \ldots + \frac{g^{(N)}(0)}{N!}(\sum_{i=1}^{m} w_{1i}^1 x_i^0 + b_1^1)^N] + b^2 \\
&= b^2 + w_1^2[g(0) + g'(0)b_1^1 + \ldots + \frac{g^{(N)}(0)}{N!}(b_1^1)^N] \\
&\quad + w_1^2[g'(0) + g''(0)b_1^1 + \ldots + \frac{g^{(N)}(0)}{(N-1)!}(b_1^1)^{N-1}]\sum_{i=1}^{m} w_{1i}^1 x_i^0 \\
&\quad + \ldots \\
&\quad + w_1^2[g^{(k)}(0) + \ldots + \frac{g^{(N)}(0)}{(N-k)!}(b_1^1)^{N-k}] \sum_{i_1=1}^{m} \sum_{i_2=1}^{i_1} \ldots \sum_{i_k=1}^{i_{k-1}} w_{1i_1}^1 \ldots w_{1i_k}^1 x_{i_1}^0 \ldots x_{i_k}^0 \\
&\quad + \ldots \\
&\quad + w_1^2[g^{(N)}(0)] \sum_{i_1=1}^{m} \sum_{i_2=1}^{i_1} \ldots \sum_{i_N=1}^{i_{N-1}} w_{1i_1}^1 \ldots w_{1i_N}^1 x_{i_1}^0 \ldots x_{i_N}^0
\end{aligned}
\qquad \text{(II.24)}
$$

Cette équation est strictement équivalente à un modèle de Kolmogorov-Gabor utilisant les ordre n_a, n_b et $q = N$ (ordre du développement de Taylor) que nous avons vu au chapitre I et que nous rappelons ici :

$$
\begin{aligned}
z = a_0 + \sum_{j=1}^{q} \sum_{l=0}^{j} \sum_{i_1=0}^{n_a} \sum_{i_2=i_1}^{n_a} \cdots \sum_{i_l=i_{l-1}}^{n_a} \sum_{i_{l+1}=1}^{n_b} \sum_{i_{l+2}=i_{l+1}}^{n_b} \cdots \\
\sum_{i_j=i_{j-1}}^{n_b} a_{i_1 i_2 \cdots i_j} \prod_{m_1=i_1}^{i_l} u(k-m_1) \prod_{m_2=i_{l+1}}^{i_j} y(k-m_2)
\end{aligned}
\qquad \text{(II.25)}
$$

Nous constatons ici qu'un perceptron multicouches est bien équivalent à un modèle de Kolmogorov-Gabor dont l'ordre q est déterminé par l'ordre du développement de Taylor de la fonction tangente hyperbolique. Le perceptron multicouches correspond donc à un modèle de Kolmogorov-Gabor d'ordre q infini. De plus nous avons également vu, au chapitre I, que ce modèle est le plus complet des modèles paramétriques, mais aussi, le plus complexe à mettre en oeuvre, notamment du point de vue de la détermination des ordres des différents polynômes.

II.3.3. Liens avec les modèles non-paramétriques non-linéaires

Bissessur et Naguib (1996) ont démontré la relation existant entre les perceptrons multicouches et les séries de Volterra. La difficulté réside, comme dans la section précédente,

dans le fait que les perceptrons multicouches utilisent une fonction d'activation tangente hyperbolique dans les neurones de la couche cachée.

Les auteurs proposent d'utiliser une décomposition en série de polynômes orthogonaux tels que les polynômes de Legendre. Afin que ces polynômes soient orthonormés dans l'espace $[-1; 1]$, la somme pondérée z_i^1 en entrée de chaque neurone caché i est normalisée en la divisant par un facteur r :

$$g(z_i^{1*}) = \frac{2}{1+e^{-2rz_i^{1*}}} - 1 \qquad \text{avec } z_i^{1*} = \frac{z_i^1}{r} \tag{II.26}$$

avec $r = \max\left(\left|z_i^1\right|\right)$.

En utilisant les polynômes de Legendre L_j, définis par la relation :

$$L_j(x) = \frac{1*3*...*(2j-1)}{j!}\left[x^j - \frac{j(j-1)}{2(2j-1)}x^{j-2} + \frac{j(j-1)(j-2)(j-3)}{2*4*(2j-1)(2j-3)}x^{j-4}+...\right] \tag{II.27}$$

L'approximation $\hat{g}$ de la fonction d'activation est donnée par :

$$\hat{g}(z_i^{1*}) = \sum_{j=0}^{M} \alpha_j L_j(z_i^{1*}) \tag{II.28}$$

où M est l'ordre du polynôme. Les coefficients α_i sont fournis par la relation suivante (O'Neil, 1991) et en utilisant la méthode des trapèzes pour approximer les intégrales :

$$\alpha_j = \frac{\int_a^b g(x)L_j(x)dx}{\int_a^b (L_j(x))^2 dx} \tag{II.29}$$

En reportant l'approximation de la fonction d'activation (II.28) dans l'expression (II.19), nous obtenons la relation :

$$\begin{aligned} x_i^1 = \hat{g}(z_i^{1*}) &= \sum_{j=0}^{M} \alpha_j L_j\left(\left(\sum_{h=1}^{n_0} w_{ih}^1 x_h^0 + b_i^1\right) r^{-1}\right) \\ &= d_{i0} + d_{i1}\left(\sum_{h=1}^{n_0} w_{ih}^1 x_h^0\right) + d_{i2}\left(\sum_{h=1}^{n_0} w_{ih}^1 x_h^0\right)^2 + \cdots + d_{i(M-1)}\left(\sum_{h=1}^{n_0} w_{ih}^1 x_h^0\right)^{M-1} \end{aligned} \tag{II.30}$$

La deuxième relation étant une écriture simplifiée où les d_{i0}, d_{i1}, ..., $d_{i(M-1)}$ sont les paramètres construits à partir des paramètres α_j, des paramètres des polynômes de Legendre, des biais de la couche d'entrée et de r.

Bissessur et Naguib (1995) montrent qu'en utilisant la relation (II.30) dans l'équation donnant la sortie du réseau (II.20), nous obtenons, en simplifiant l'écriture et en omettant les termes d'ordre supérieur à trois, une relation équivalente à une série de Volterra présentée au chapitre I :

$$z = b^2 + \sum_{h=1}^{n_0} \{\Gamma_{1h}\} x_h^0 + \sum_{k=1}^{n_0} \sum_{h=1}^{n_0} \{\Gamma_{2hk}\} x_h^0 x_k^0 + \sum_{m=1}^{n_0} \sum_{k=1}^{n_0} \sum_{h=1}^{n_0} \{\Gamma_{3hkm}\} x_h^0 x_k^0 x_m^0 \qquad \text{(II.31)}$$

Les expressions complètes des paramètres Γ_{1h}, Γ_{2hk}, Γ_{3hkm} sont données dans Bissessur et Naguib (1996). Ces paramètres correspondent aux noyaux d'une série de Volterra que l'on peut déterminer à l'aide des paramètres internes du réseau. Ce dernier point est particulièrement intéressant car le calcul de ces noyaux est généralement très difficile.

II.4. Indicateurs de qualité du modèle

L'obtention d'un modèle du système non-linéaire fournissant des résultats corrects sur le jeu de données d'identification n'est pas suffisant. En effet, le modèle mathématique obtenu à partir de l'estimation des paramètres d'un perceptron multicouches n'est valable, en toute rigueur, que pour les données d'apprentissage. Il faut donc vérifier que ce modèle est compatible avec d'autres formes d'entrées afin de représenter correctement le fonctionnement du processus générateur.

La plupart des indicateurs de qualité du modèle ont été mis au point pour la validation de modèles paramétriques et en particulier, la vérification des hypothèses posées pour construire le modèle. Cependant, leur utilisation pour des modèles neuronaux ne pose aucun problème particulier. Nous allons présenter les indicateurs de qualité les plus couramment utilisés. Pour bâtir ce paragraphe, nous nous sommes inspirés des travaux de Theilliol (1993) et Norgaard (1995).

II.4.1. Fonction d'autocorrélation des résidus

Les méthodes les plus communes de validation de modèle consistent dans l'étude des résidus (erreurs de prédiction) obtenus par une validation croisée sur un jeu de données de validation.

Ces résidus $\varepsilon(k)$ constituent l'estimation des perturbations non mesurables présentes au sein du système. Une modélisation correcte du système doit fournir, d'une part, des résidus indépendant entre eux, et d'autre part, des résidus indépendants de l'entrée du système. Les résidus sont supposés suivre une loi normale de distribution.

Le test d'indépendance des résidus entre eux peut être obtenu en utilisant la fonction d'autocorrélation des résidus $E\{\varepsilon(\tau)\varepsilon(k+\tau)\}$ où E représente l'espérance mathématique. Un estimateur de la fonction d'autocorrélation est fourni par la relation :

$$R^n_{\varepsilon\varepsilon}(\tau) = \frac{1}{n}\sum_{k=1}^{n-\tau}\varepsilon(\tau)\varepsilon(k+\tau) \qquad \text{(II.32)}$$

où n représente le nombre de mesures à tester et τ définit le décalage ou retard.

Le modèle est validé si la fonction d'autocorrélation de ses résidus, dans le cas de l'hypothèse classique d'une distribution Gaussienne de ces derniers, appartient à l'intervalle de confiance de moyenne nulle au niveau de probabilité de 95% (Cetama, 1986). Le test de validation est effectué sur la fonction d'autocorrélation des résidus normalisée par la relation :

$$\frac{\left|R^n_{\varepsilon\varepsilon}(\tau)\right|}{\left|R^n_{\varepsilon\varepsilon}(0)\right|} \le \frac{1{,}96}{\sqrt{n}} \qquad \forall\tau > 0 \qquad \text{(II.33)}$$

où $R^n_{\varepsilon\varepsilon}(0)$ représente la variance des résidus, $R^n_{\varepsilon\varepsilon}(0) = \sigma$.

II.4.2. Fonction d'intercorrélation entre les résidus et les entrées précédentes

Nous avons vu à la section précédente qu'il est nécessaire également que les résidus $\varepsilon(k)$ soient indépendants des entrées u(k) afin de pouvoir valider le modèle de prédiction. Ce test est réalisé en utilisant la fonction d'intercorrélation entre les résidus et l'entrée du système. Un estimateur de cette fonction est fourni par la relation :

$$R^n_{\varepsilon u}(\tau) = \frac{1}{n}\sum_{k=1}^{n-\tau}\varepsilon(\tau)u(k+\tau) \qquad \text{(II.34)}$$

Cette fonction d'intercorrélation $R^n_{\varepsilon u}(\tau)$ doit appartenir à une loi normale de moyenne nulle. Pour que le modèle soit validé, la fonction d'intercorrélation doit être comprise dans l'intervalle de confiance de moyenne nulle au niveau de probabilité de 99% décrit par l'inégalité :

$$\left|R^n_{\varepsilon u}(\tau)\right| \le \frac{1{,}96\sqrt{P(\tau)}}{\sqrt{n}} \qquad \forall\tau \qquad \text{(II.35)}$$

où $P(\tau)$ représente la variance de la fonction d'intercorrélation que l'on peut calculer par la relation:

$$P(\tau) = R^n_{\varepsilon}(0)R^n_{u}(0) + 2R^n_{\varepsilon}(\tau)R^n_{u}(\tau) \qquad \text{(II.36)}$$

Pour une meilleure interprétation graphique, la fonction d'intercorrélation est normalisée dans l'intervalle :

$$\frac{\left|R_{\varepsilon u}^{n}(\tau)\right|}{\sqrt{\left|R_{\varepsilon}^{n}(0)R_{u}^{n}(0)\right|}} \leq \frac{1{,}96\sqrt{P(\tau)}}{\sqrt{n}\sqrt{\left|R_{\varepsilon}^{n}(0)R_{u}^{n}(0)\right|}} \qquad \text{(II.37)}$$

Le test construit à partir de la fonction d'intercorrélation peut être utilisé également lors de l'estimation du retard pur n_k. Une mauvaise estimation de ce retard engendre une dépendance entre les entrées passées et les résidus. Cette erreur est caractérisée par une valeur de $R_{\varepsilon u}^{n}(n_k)$ n'appartenant pas à l'intervalle de confiance.

II.4.3. Autres tests de corrélation

Dans les deux sections précédentes, nous avons présenté deux tests de corrélation utilisés initialement pour les modèles linéaires.

Cependant, l'utilisation de ces tests dans le cadre de la modélisation non-linéaire peut conduire à l'échec du diagnostic lorsque des termes non-linéaires du modèle sont oubliés (Billings et Voon, 1983). Aussi, d'autres tests doivent être considérés.

Pour régler le problème précédent, il est possible d'utiliser une extension des tests précédemment décrits en prenant en compte les puissances du signal d'entrée ou du résidu (Billings et Voon, 1986). Il est également possible d'étendre ces tests à l'exploitation de l'information comprise dans la sortie du système (Billings et Zhu, 1994 et 1995).

Nous allons présenter ces tests qui prennent en compte la corrélation entre les vecteurs d'entrée, de sortie ou de résidus. Ces tests sont définis par :

$$\begin{aligned} R_{\xi\eta}(\tau) &= E\{\xi(k)\,\eta(k+\tau)\} \\ R_{\upsilon\eta}(\tau) &= E\{\upsilon(k)\,\eta(k+\tau)\} \end{aligned} \qquad \text{(II.38)}$$

où $\xi(k)$, $\eta(k)$ et $\upsilon(k)$ sont donnés par :

$$\begin{aligned} \xi(k) &= \varepsilon^2(k) \\ \eta(k) &= y(k)\varepsilon(k) \\ \upsilon(k) &= u^2(k) \end{aligned} \qquad \text{(II.39)}$$

En pratique, la relation (II.38) est remplacée par une fonction de corrélation normalisée définie par :

$$R_{\xi\eta}(\tau) = \frac{\sum_{k=1}^{n-\tau} \xi^o(k)\eta^o(k+\tau)}{(\sum_{k=1}^{n-\tau} (\upsilon^o(k))^2 \sum_{k=1}^{n-\tau} (\eta^o(k))^2)^{1/2}} \qquad \text{(II.40a)}$$

$$R_{\upsilon\eta}(\tau) = \frac{\sum_{k=1}^{n-\tau} \upsilon^o(k)\eta^o(k+\tau)}{(\sum_{k=1}^{n-\tau} (\upsilon^o(k))^2 \sum_{k=1}^{n-\tau} (\eta^o(k))^2)^{1/2}} \qquad \text{(II.40b)}$$

où $\xi^o(k)$, $\eta^o(k)$ et $\upsilon^o(k)$ sont donnés par :

$$\xi^o(k) = \frac{\varepsilon^{2^o}(k)}{\sqrt{\frac{1}{n}\sum_{k=1}^{n}(\varepsilon^{2^o}(k))^2}} \quad \eta^o(k) = \frac{(y(k)\varepsilon(k))^o}{\sqrt{\frac{1}{n}\sum_{k=1}^{n}(\varepsilon^{2^o}(k))^2}} \quad \upsilon^o(k) = \frac{u^{2^o}(k)}{\sqrt{\frac{1}{n}\sum_{k=1}^{n}(u^{2^o}(k))^2}} \qquad \text{(II.41)}$$

avec :

$$\varepsilon^{2^o}(k) = \frac{1}{n}\sum_{k=1}^{n}\varepsilon^2(k) \quad (y(k)\varepsilon(k))^o = \frac{1}{n}\sum_{k=1}^{n}y(k)\varepsilon(k) \quad u^{2^o}(k) = \frac{1}{n}\sum_{k=1}^{n}u^2(k) \qquad \text{(II.42)}$$

Idéalement, si le modèle est validé, les résultats de ces tests de corrélation normalisés conduisent aux résultats suivants :

$$R_{\xi\eta}(\tau) = \begin{cases} \kappa & \tau = 0 \\ 0 & \tau \neq 0 \end{cases} \qquad \text{(II.43)}$$
$$R_{\upsilon\eta}(\tau) = 0 \quad \forall\tau$$

où κ est une constante définie par (Billings et Zhu, 1994) :

$$\kappa = \frac{\sqrt{\sum_{k=1}^{n}(\xi^o(k))^2)}}{\sqrt{\sum_{k=1}^{n}(\eta^o(k))^2)}} \qquad \text{(II.44)}$$

L'utilisation des différents tests présentés dans ce paragraphe est nécessaire pour valider la modélisation du système étudié. Cependant, d'autres indicateurs existent et fournissent une information non négligeable sur le modèle.

II.4.4. Autres indicateurs de qualité

Le critère le plus simple et le plus répandu dans le domaine de la validation de modèle reste la somme des carrés des erreurs appelé également critère résiduel :

$$V = \frac{1}{n} \sum_{k=1}^{n} (\varepsilon(k))^2 \qquad \text{(II.45)}$$

Ce critère ne prend en compte que le résultat brut de la modélisation sans tenir compte d'autres considérations comme le nombre de paramètres du modèle. Au contraire, Akaike propose d'utiliser le critère d'erreur de prédiction finale (FPE) (Ljung, 1987) :

$$V_{FPE} = \frac{n+d}{n-d} \sum_{k=1}^{n} (\varepsilon(k))^2 \qquad \text{(II.46)}$$

où d représente le nombre de paramètres du modèle, n le nombre de données du jeu d'identification et ε le résidu.

Ce critère est calculé en fin d'apprentissage pour chacun des modèles testés. Le minimum de ce critère correspond au modèle fournissant le meilleur compromis entre résultat de la modélisation et nombre de paramètres. Ce critère a été mis au point pour valider les modèles linéaires. Cependant Norgaard (1996) l'a utilisé lors de la recherche de la structure du perceptron multicouches.

Il existe beaucoup d'autres indicateurs de la qualité du modèle.

II.5. Conclusion

Au cours de ce chapitre, nous avons défini dans un premier temps le type de réseau de neurones que nous allons utiliser. Nous avons effectué un certain nombre de choix quant à la structure de ce réseau de neurones.

Ces choix nous ont permis de définir une structure précise de réseau de neurones appelée perceptron multicouches dans ce mémoire. Cependant, nous avons montré qu'un certain nombre de paramètres, comme le nombre de neurones sur la couche cachée, ne peuvent être définis qu'en fonction du système étudié. Nous avons donc proposé des méthodes pour déterminer la structure optimale du perceptron multicouches en fonction du problème considéré.

Par la suite, nous avons démontré les relations importantes qui lient la modélisation à l'aide de perceptron multicouches et les méthodes classiques de modélisation. Ces considérations nous permettrons d'adapter des stratégies classiques à notre problème de modélisation neuronale.

Nous avons conclu ce chapitre en présentant divers indicateurs permettant de déterminer la qualité (ou la non-qualité) des modèles que nous allons construire au cours de ce travail.

Si nous avons bien défini le domaine dans lequel nous allons travailler, nous n'avons pas encore présenté les principaux problèmes qu'il sera nécessaire de régler pour effectuer une modélisation correcte en utilisant les perceptrons multicouches. L'un de ces principaux problèmes se pose à nous dès l'initialisation du perceptron multicouches. Ce problème, ainsi que les diverses méthodes que nous proposons pour le résoudre, sera présenté au chapitre suivant.

II.6. Références

ABE S., M. KAYAMA, Y. MOROOKA, H. TAKENAGA (1990) 'Constructiong optimal neural networks by linear regression analysis', *Neuro-Nimes'90*, Nimes, France, 12-16 Nov., 363.

ABE S., M. KAYAMA, T. KITAMURA, Y. OKUYAMA, K. TADAAKI, H. TAKENAGA (1990) 'Optimal input selection of neural networks by sensitivity analysis and its application to image recognition', *IAPR Workshop on Machine Vision Applications*, Tokyo, Japon, 117.

AKAÏKE H. (1974) 'A new look at the statistical model identification', *IEEE trans. On Automatic Control*, Vol.19, 6, 716-723.

ALIPPI C. (1991) 'Weight update in back-propagation neural networks : the role of activation functions', *Int. Joint Conf. on Neural Networks IJCNN'91*, Seattle, U.S.A., 8-12 juil., 560-565.

BARTLETT E.B. (1994) 'Dynamic node architecture learning: an information theoretic approach', *Neural Networks*, Vol.7, 1, 129-140.

BILLINGS S.A., W.S.F. VOON (1983) 'Structure detection and model validity tests in the identification of nonlinear systems', *Proc. of the Institution of Electrical Engineers*, Vol.130, 193-199.

BILLINGS S.A., W.S.F. VOON (1986) 'Correlation based model validity tests for nonlinear models', *Int. J. of Control*, Vol.44, 235-244.

BILLINGS S.A., Q.M. ZHU (1994) 'Nonlinear model validation using correlation tests', *Int. J. of Control*, Vol.49, 1107-1120.

BILLINGS S.A., Q.M. ZHU (1995) 'Model validation tests for multivariable nonlinear models including neural networks', *Int. J. of Control*, Vol.62, 4, 749-766.

BISHOP C. M. (1995) *Neural networks for pattern recognition*, Oxford University Press, Oxford, G.B.

BISSESSUR Y., R.N.G. NAGUIB (1996) 'Buried plant detection : a Volterra series modelling approach using artificial neural networks', *Neural Networks*, Vol.9, 6, 1045-1060.

BORNE P., G. DAUPHIN-TANGUY, J.P. RICHARD, F. ROTELLA, I. ZAMBETTAKIS (1992) *Modélisation et identification des processus*, Technip, Paris, France.

BORNHOLDT S., D. GRAUDENZ (1992) 'General asymetric neural networks ans structure design by genetic algorithms', *Neural Networks*, Vol.5, 327-334.

CETAMA (1986) *Statistiques appliquée à l'exploitation des mesures*, 2nde édition, Masson, Paris, France.

CHENTOUF R., C. JUTTEN (1996) 'Combining sigmoids and radial basis functions in evolutive neural architecture', *European Symposium on Artificial Neural networks ESANN'96*, Bruges, Belgique, 24-26 Avril, 129-134.

CIBAS T., F. FOGELMAN-SOULIE, P. GALLINARI, S. RAUDYS (1996) 'Variable selection with neural networks', *Neurocomputing*, Vol.12, 223-248.

COTTRELL M., B. GIRARD, Y. GIRARD, M. MANGEAS (1993) 'Times series and neural networks: a statistical method for weight elimination', *European Symposium on Artificial Neural Network ESANN'93*, Bruges, Belgique, 157-164.

COTTRELL M., B. GIRARD, Y. GIRARD, M. MANGEAS, C. MULLER (1995) 'Neural modelling for time series: A statistical stepwise method for weight elimination', *IEEE Trans. on Neural Networks*, Vol.6, 6, 1355-1364.

LECUN Y., J. DENKER, S. SOLLA (1990) 'Optimal brain damage', *Proc. of the Neural Information Processing Systems*, Denver, U.S.A., 598-605.

CYBENKO G. (1989) 'Approximation by superpositions of sigmoidale function', *Mathematics of Control Signal and Systems*, Vol.2, 303-314.

DUONG D.H., I.D. LANDAU (1993) 'On an EIV based structural estimation method', *IFAC World Congress*, Sydney, Australie.

DUONG D.H., I.D. LANDAU (1993) 'On a criterion for estimating the structure of linear MIMO systems', *Second European Control Conference ECC'93*, Groningen, Hollande.

DUONG D.H., I.D. LANDAU (1993) 'On statistical properties of a test for model structure selection using the extended instrumental variable approach', *IEEE Trans. Auto. Control.*, Nov.

ELLIOTT D.L. (1993) *A better activation function for artificial neural networks*, Tech. Repport. TR-93-8, Institute for Systems Research, University of Maryland.

FAHLMAN S.E., C. LEBIERE (1990) 'The cascade correlation learning architecture', *Neural Information Processing Systems*, Vol.2, 524-532.

FOMBELLIDA M.R.J., M.J.M. MINSOUL, J.L.O. DESTINE (1990) 'Perceptrons multicouches et fonctions d'activation non-monotones', *Neuro-Nimes'90*, Nimes, France, 12-16 Nov., 321-339.

FUNAHASHI K. (1989) 'On the approximate realization of continuous mappings by neural networks', *Neural Networks*, Vol.2, 183-192.

HANSEN L.K., M.W. PEDERSEN (1994) 'Controlled growth of cascade correlation nets', *Int. Conf. on Artificial Neural Networks ICANN'94*, Sorento, Italie, 797-800.

HASSIBI B., D.G. STORK (1992) 'Second order derivatives for neural pruning : Obtimal Brain Surgeon', *Neural Information Processing Systems*.

HERGERT F., W. FINNOFF, H.G. ZIMMERMANN (1992) 'A comparison of weight elimination methods for reducing complexity in neural networks', *Int. Joint Conf. on Neural Networks IJCNN'92*, Baltimore, U.S.A., 7-11 juin, Vol.3, 980-987.

HILBERT D. (1900) 'Mathematishe probleme', *Nachrichten der Akademie der Wissenschaften Göttingen*, 290-329.

JUTTEN C., FAMBON O. (1995) 'Pruning methods: a review', *European Symposium on Artificial Neural Network ESANN'95*, Bruges, Belgique, 19-21 Avril, 129-140.

KOLMOGOROV A.N. (1957) 'on the représentation of continuous functions of many variables by superposition of continuous functions of one variable and addition', *American Mathematical Society Translations*, Vol.28.

LENGELLE R., T. DENOEUX (1996) 'Training MLPs layer by layer using an objective function for internal representation', *Neural Networks*, Vol.9, 1, 83-97.

LJUNG L. (1987) *System identification : theory for the user*, Prentice-Hall, Englewood Cliffs, N.J.

MOHRAZ K., P. PROTZEL (1996) 'FlexNet. A flexible neural network construction algorithm', *European Symposium on Artificial Neural networks ESANN'96*, Bruges, Belgique, 24-26 Avril, 111-116.

MOZER M., P. SMOLENSKY (1989) 'Skeletonization : a technique for trimming the fat from a network via a relevance assessment, *Advance in Neural Information Processing*, 107-115.

NARENDRA K.S. (1994) 'Adaptive control of dynamical systems using neural networks', *in Handbook of Intelligent Control*, Van Nostrand.

NORGAARD M. (1995) *Neural network based system identification toolbox*, Tech. Repport. 95-E-773, Institute of Automation, Technical University of Denmark.

NORGAARD M. (1996) *System identification and control with neural networks*, Thèse, Institute of Automation, Technical University of Denmark.

O'NEIL P.V. (1991) *Advanced engineering mathematics*, Wadsworth Pub. Co., California.

RUMELHART D.E., J.L. MCCLELLAND (1986) *Parallel distributed processing,* The MIT Press, Cambridge, England.

SANCHEZ M.S., H. SWIERENGA, L.A. SARABIA, E. DERKS, L. BUYDENS (1996) 'Performance of multi layer feedforward and radial base function neural networks in classification and modelling', *Chemometrics and Intelligent Laboratory Systems*, Vol.33, 101-119.

SJÖBERG J., Q. ZHANG, L. LJUNG, A. BENVENISTE, B. DELYON, P.Y. GLORENNEC, H. HJALMARSSON, A. JUDITSKY (1995) 'Nonlinear black-box modeling in system identification: A unified overview', *Automatica*, Vol.31, 12, 1691-1724.

THEILLIOL D. (1993) *Identification de systèmes SISO linéaires et non linéaires par réseaux de neurones multicouches*, Thèse de doctorat de l'université Nancy I, spécialité automatique.

THOMAS P. (1993) *Identification robuste de systèmes SISO linéaires et non-linéaires. Approche paramétrique et neuronale*, Rapport de D.E.A. Automatique Traitement du Signal de l'université Nancy I.

WARWICK K. (1996) 'System identification using neural networks', *in Identification in Engineering Systems*, The Cromwell Press Ltd, Broughton Gifford, G.B., 689-701.

WILLIAMS P.M. (1995) 'Bayesion regularization and pruning using a Laplace prior', *Neural Computation*, Vol.7, 1, 117-143.

Chapitre III

Algorithmes d'initialisation du perceptron multicouches

III.1. Introduction

III.1.1. Problématique

Depuis la mise au point du principe de la rétropropagation du gradient par Le Cun (1985), d'une part, et Rumelhart et McClelland (1986) d'autre part, les perceptrons multicouches sont connus comme étant des outils puissants pour apprendre ou identifier des relations entrées-sorties dans des domaines aussi divers que, la classification, la reconnaissance de formes, la détection de défauts, le contrôle de systèmes non linéaires et, bien entendu, la modélisation et l'identification de systèmes non linéaires (Chen *et al.*, 1990; Narendra et Parthasarathy, 1990; Sjöberg *et al.*, 1994). Malgré cela, l'utilisation de perceptrons multicouches se heurte encore à divers problèmes.

Les problèmes principaux rencontrés durant l'apprentissage par perceptrons multicouches à l'aide de l'algorithme de rétropropagation du gradient sont sa lenteur de convergence et également le risque de tomber dans un minimum local éloigné du minimum global recherché. Les performances limitées de ce type d'algorithme peuvent avoir deux causes principales : la faible efficacité de l'algorithme d'apprentissage lui-même, l'initialisation des poids du réseau très éloignés du résultat à obtenir. Il est bien évident que la résolution de l'un ou l'autre de ces problèmes permet d'améliorer fortement la rapidité de l'apprentissage ainsi que la qualité de la solution.

Beaucoup de travaux se sont concentrés sur l'optimisation de l'algorithme d'apprentissage. Ces travaux ont conduit à plusieurs variations de l'algorithme de la rétropropagation du gradient (Rumelhart *et al.*, 1986; Chen *et al.*, 1990; Billings et Jamaluddin, 1991; Van der Smagt et Kröse, 1991, Hagan *et al.*, 1995), l'initialisation des poids étant généralement effectuée de manière aléatoire (Lari-Najafi *et al.*, 1989). Cependant, si l'initialisation des poids n'est pas bonne, la convergence va être fortement ralentie, même si un algorithme amélioré est utilisé. Dans le pire des cas, le perceptron multicouches peut même converger vers un minimum local très éloigné du minimum global recherché. Dans ce cas, la stratégie habituelle consiste à recommencer l'ensemble de la phase d'apprentissage avec une nouvelle initialisation aléatoire des poids si la précédente n'a pas conduit à une convergence acceptable. Cependant, rien n'assure que l'initialisation des poids à l'aide de valeurs aléatoires permette au réseau de converger correctement vers le minimum global. C'est pourquoi, plusieurs autres formes d'initialisation ont été étudiées pour la classification (Wessels et Barnard, 1992; Kim, 1993; Denoeux et Lengellé, 1993; Karouia *et al.*, 1994; 1995a; b) ou pour l'identification de systèmes non linéaires (Nguyen et Widrow, 1990; Burel, 1991; Chen et Nutter, 1991; Denoeux et Lengellé, 1993; Lehtokangas *et al.*, 1995). Certains algorithmes d'initialisation sont dédiés à des formes de réseaux particulières que nous ne traitons pas ici (Chen et Bastani, 1992).

Dans ce chapitre, nous cherchons à recenser les méthodes d'initialisation adaptées au réseau que nous avons sélectionné au chapitre précédent dans le cadre de l'identification. Ces méthodes seront présentées en deux classes différentes : les méthodes nécessitant un jeu de données et celles ne nécessitant que des informations succinctes sur les données. Nous avons choisi de créer ces deux classes car, s'il est généralement possible de connaître les valeurs

maximales et minimales que peut prendre une variable à l'aide de connaissances succinctes sur le système qui la génère, il n'est pas toujours évident, pour l'identification en ligne, de disposer d'un jeu de données complet. Une telle situation exclut donc l'utilisation d'un algorithme d'initialisation nécessitant un jeu complet de données. Au cours de ce chapitre, nous présenterons donc, successivement, les méthodes utilisant uniquement des informations succinctes sur les données puis, celles nécessitant un jeu complet de données. Dans chacune de ces catégories, nous présenterons une évolution de l'algorithme de Nguyen et Widrow (1990) que nous avons mise au point. Nous finirons ce chapitre par une étude de simulation comparant l'ensemble des méthodes présentées et illustrant l'intérêt des algorithmes que nous proposons.

III.1.2. Rappel de l'architecture du réseau

Nous avons vu dans le chapitre II que nous restreignons notre étude aux perceptrons multicouches. Nous rappelons que la première couche est constituée de neurones "transparents" qui distribuent juste le vecteur des entrées du réseau $X = [x_1^0 \cdots x_{n_0}^0]^T$, correspondant au vecteur d'information à l'instant k, $\phi(k) = [\phi_1(k) \cdots \phi_{n_0}(k)]^T$, aux neurones de la couche suivante, appelée couche cachée. Ces neurones calculent dans un premier temps la somme pondérée des entrées du réseau :

$$z_i^1 = \sum_{h=1}^{n_0} w_{ih}^1 x_h^0 + b_i^1 \qquad \text{(III.1)}$$

où w_{ih}^1 et b_i^1 sont les poids et biais correspondant au neurone i. Les sorties des n_1 neurones cachés sont obtenues par la relation :

$$x_i^1 = g(z_i^1) \qquad \text{(III.2)}$$

où g est la fonction d'activation choisie tangente hyperbolique. La sortie désirée à l'instant k d(k) est alors estimée par la sortie z du réseau qui est décrite par l'équation :

$$z = \sum_{i=1}^{n_1} w_i^2 x_i^1 + b^2 = \sum_{i=1}^{n_1} w_i^2 g\left(\sum_{h=1}^{n_0} w_{ih}^1 x_h^0 + b_i^1\right) + b^2 \qquad \text{(III.3)}$$

où w_i^2 et b^2 sont les poids et biais correspondant au neurone de sortie.

Au cours de ce chapitre, nous allons utiliser un certain nombre de notations. Tout d'abord X0 est une matrice de dimension $(n.n_0)$ comprenant les n vecteurs d'entrée du jeu d'identification. D est le vecteur de dimension (n.1) regroupant l'ensemble des sorties désirées du jeu d'identification. X1 est la matrice de dimension $(n.n_1)$ comprenant toutes les sorties des neurones cachés pour l'ensemble du jeu d'identification. B est un vecteur de dimension (n.1)

composé de 1 représentant une entrée supplémentaire correspondant aux biais. W1 est la matrice de dimension (n_0. n_1) comprenant les poids connectant la couche d'entrée à la couche cachée. On appelle W_i^1 le vecteur de dimension (n_0.1) des poids w_{ih}^1 du neurone i. B1 est le vecteur de dimension (1.n_1) des biais de la couche cachée. W2 est le vecteur de dimension (n_1.1) comprenant les poids connectant la couche cachée à la couche de sortie. WB1 est la matrice de dimension (n_0+1.n_1) comprenant l'ensemble des poids et biais connectant la couche d'entrée à la couche cachée que l'on peut écrire :

$$WB1 = [B1 \ W1] \tag{III.4}$$

WB2 est le vecteur comprenant tous les poids et biais connectant la couche cachée à la couche de sortie que l'on peut écrire :

$$WB2 = [b^2 \ W2] \tag{III.5}$$

III.2. Algorithmes d'initialisation ne nécessitant pas de jeu de données

III.2.1. Initialisation par linéarisation sur le domaine des entrées

Cette méthode d'initialisation est due à Nguyen et Widrow (1990). Son principe se rapproche de la linéarisation par morceaux. L'algorithme initial est prévu pour travailler avec des entrées du réseau comprises entre -1 et 1.

Afin d'en rappeler le principe, nous allons dans un premier temps nous situer dans le cas simple de l'identification d'une fonction à entrée unique g(x).

Nous allons alors considérer que chaque neurone de la couche cachée va simplement correspondre à une fonction linéaire de x sur un petit intervalle de l'espace des entrées. La taille de l'intervalle est déterminée par w_{i1}^1. Plus w_{i1}^1 est grand, plus l'intervalle est petit. La localisation de l'intervalle dans l'espace des x est fournie par b_i^1. En effet, le centre de l'intervalle se trouve pour $x = -b_i^1 / w_i^1$.

Dans ce cas simple, nous pouvons déterminer les poids initiaux connectant la couche d'entrée à la couche cachée en prenant :

$$w_{i1}^1 = 0.7 * n_1 \tag{III.6}$$

Le choix de multiplier n_1 (le nombre de neurone dans la couche cachée) par 0.7 permet d'obtenir un léger chevauchement des intervalles. b_i^1 est une valeur aléatoire choisie uniformément entre $\left[-\left|w_{i1}^1\right|; \left|w_{i1}^1\right|\right]$.

On obtient alors :

$$-1 < w_{i1}^1 x + b_i^1 < 1 \qquad \text{(III.7)}$$

et donc :

$$\frac{-1-b_i^1}{w_{i1}^1} < x < \frac{1-b_i^1}{w_{i1}^1} \qquad \text{(III.8)}$$

Il faut maintenant chercher à étendre ce principe au cas plus complexe ou le vecteur d'entrée X comporte n_0 éléments. Dans un premier temps, nous allons donner des valeurs aléatoires uniformément distribuées entre -1 et 1 aux poids w_{ih}^1. Ce choix aléatoire des poids a pour but de rompre la symétrie du réseau.

L'amplitude des vecteurs de poids W_i^1 doit être ajustée de telle sorte que chaque sortie de neurone caché soit linéaire sur un petit intervalle du domaine des entrées. Par analogie avec le cas simple ou le réseau ne possède qu'une seule entrée, Nguyen et Widrow, considérant des entrées normalisées entre -1 et 1, estiment la longueur L de chaque intervalle par la relation :

$$L = \frac{2}{n_1^{1/n_0}} \qquad \text{(III.9)}$$

Les amplitudes des vecteurs des poids W_i^1 peuvent alors être ajustées à :

$$\left|W_i^1\right| = 0.7 * n_1^{1/n_0} \qquad \text{(III.10)}$$

Les b_i^1 sont tirés aléatoirement de manière uniforme sur l'intervalle $\left[-\left|W_i^1\right|;\left|W_i^1\right|\right]$ pour fournir les centres des intervalles. Les poids de la couche de sortie w_i^2 et le biais b^2 sont choisis aléatoirement de manière uniforme sur l'intervalle $[-1;1]$.

Demuth et Beale (1994) ont proposé dans leur boîte à outils Matlab une variante de cet algorithme permettant de travailler avec des entrées pouvant avoir une amplitude quelconque. Pour cela, ils divisent les poids w_{ih}^1 obtenus à l'équation (III.10) par la valeur :

$$\alpha_{ih} = \frac{2}{\max(x_h^0) - \min(x_h^0)} \qquad \text{(III.11)}$$

De plus, ils recentrent les intervalles dans l'espace des entrées en ajoutant aux b_i^1 précédemment trouvés la valeur :

$$\beta_i = \sum_{h=1}^{n_0} w_{ih}^1 \frac{\min(x_h^0) + \max(x_h^0)}{2} \qquad \text{(III.12)}$$

Cette méthode d'initialisation sera appelée NW (Nguyen Widrow) dans les simulations ultérieures.

III.2.2. Initialisation par utilisation de connaissances a priori sur la sortie

L'algorithme de Nguyen et Widrow (1990) repose sur le principe d'une linéarisation par morceau des sorties des neurones cachés. Cependant, nous pouvons également utiliser pour l'initialisation des paramètres du réseau certaines connaissances que l'on a sur les variables.

Or, comme nous l'avons dit en introduction, il est généralement possible de connaître a priori les valeurs maximales (Ymax) et minimales (Ymin) que peut prendre une variable avant même d'en avoir des échantillons. Aussi, il peut être intéressant d'initialiser le réseau en utilisant ces caractéristiques, connues a priori, de l'objet de l'apprentissage, c'est-à-dire de la sortie désirée du réseau.

La méthode développée par Nguyen et Widrow (1990) et modifiée par Demuth et Beale (1994) associe les données d'entrées d'amplitude quelconque à l'espace [-1; 1] en sortie des neurones de la couche cachée par une linéarisation par morceaux. Cependant, il est rare que les amplitudes des sorties soient comprises entre -1 et 1. Aussi, l'utilisation de valeurs aléatoires pour initialiser les poids et biais de la couche de sortie est insuffisante. Nous allons donc chercher à étendre cette linéarisation par morceau entre -1 et 1 en sortie des neurones cachés à l'espace couvrant l'ensemble des amplitudes que peut prendre la sortie désirée. Pour cela, nous allons modifier le choix des poids et biais connectant la couche cachée à la couche de sortie préconisé par Nguyen et Widrow dans leur algorithme.

Tout d'abord, rien ne nous assure que la moyenne du signal de la sortie désirée soit nulle, ni même proche de 0. Il est cependant aisé de ramener la linéarisation par morceau autour de cette moyenne. Pour cela, il suffit d'initialiser le biais b^2 à la valeur :

$$b^2 = \frac{1}{2}(Y\max + Y\min) \qquad \text{(III.13)}$$

Ce simple choix améliore déjà fortement la convergence et la rapidité d'apprentissage d'un réseau dans le cas où la sortie désirée est de moyenne non nulle. Au contraire, si la sortie désirée est de moyenne nulle, cette modification ne fournit ni amélioration, ni dégradation de la convergence.

D'autre part, la plage de variation de la sortie désirée n'est pas forcément proche de 2. De même que précédemment, il est possible d'approcher la valeur de la plage de variation de la sortie estimée à proximité de la plage de variation réelle de la variable considérée à l'aide d'un choix particulier des poids connectant la couche cachée à la couche de sortie w_i^2 :

$$w_i^2 = \alpha_i * (Y\max - Y\min) \quad \text{pour } i = 1, \cdots, n_1 \qquad \text{(III.14)}$$

α_i étant une valeur aléatoire choisie uniformément dans l'intervalle [0; 1].

Le fait d'utiliser des paramètres partiellement aléatoires permet de toujours pouvoir recommencer l'apprentissage avec une autre initialisation si l'on tombe dans un minimum local. Cette méthode d'initialisation sera appelée NWM (Nguyen Widrow Modifié) lors des tests de simulation.

III.3. Algorithmes d'initialisation nécessitant un jeu de données

III.3.1. Initialisation aléatoire des poids dans un espace particulier

Cette méthode d'initialisation est due à Burel (1991) et a été reprise par Hérault et Jutten (1994). Elle a initialement été adaptée au problème du traitement d'images. Elle repose sur l'idée que les perceptrons multicouches nécessitent une initialisation aléatoire des poids de façon à briser la symétrie du réseau. Généralement, les poids sont tirés uniformément entre des bornes $-M_1$ et $+M_2$. Ces bornes ne sont pas forcément les mêmes pour l'ensemble des neurones. Cependant, le choix de ces bornes n'est pas aisé. C'est pourquoi, Burel (1991) propose une méthode de calcul automatique de ces valeurs.

Dans cet algorithme, Burel considère les biais comme des poids connectant une entrée unitaire aux neurones de la couche cachée.

Tout d'abord, il faut chercher à obtenir les conditions suivantes :

$$\begin{aligned} &(C1) \quad E(z_i) = 0 \quad \text{pour } i = 1, \cdots, n_1 \\ &(C2) \quad \text{var}(z_i) = \sigma_z^2 \quad \text{pour } i = 1, \cdots, n_1 \end{aligned} \qquad \text{(III.15)}$$

où E désigne l'espérance mathématique et var la variance. Ces conditions signifient que les z_i sont supposés centrés et de variance σ_z^2. Cette valeur de variance sera fixée ultérieurement. la première condition a simplement pour but d'utiliser les fonctions d'activation sur l'ensemble de leurs plages de variation. La seconde condition permet un contrôle de la dispersion des potentiels initiaux. Une trop forte dispersion conduirait à un comportement très différents des neurones de la couche cachée et les neurones les moins adaptés au problème considéré resteraient inutilisés durant les premières étapes de l'apprentissage, d'où une grande lenteur de l'apprentissage.

La première condition va conduire à :

$$E(w_{ih}^1) = 0 \qquad \text{pour } i = 1, \cdots, n_1 \text{ et pour } h = 1, \cdots, n_0 \tag{III.16}$$

En effet, puisque les poids sont indépendants des entrées du réseau, nous avons la relation $E(z_i) = \sum_{h=1}^{n_0+1} E(w_{ih}^1)E(x_h^0)$. La solution la plus simple pour annuler cette somme consiste à initialiser les poids en respectant l'équation (III.16), et pour cela, il suffit de choisir $M_1=M_2=M$.

Ceci fixé, il est possible de définir la variance de z_i par :

$$\sigma_z^2 = \sum_{h=1}^{n_0+1} E((w_{ih}^1)^2)E((x_h^0)^2) \tag{III.17}$$

En effet, comme z_i est centré d'après (C1), on a :

$$\text{var}(z_i) = E(z_i^2) = \sum_{h=1}^{n_0+1} E((w_{ih}^1 x_h^0)^2) \sum_{h=1}^{n_0+1} E((w_{ih}^1)^2 (x_h^0)^2) + \sum_{k\neq m} E(w_{ik}^1 x_k^0 w_{im}^1 x_m^0)$$

Comme les poids sont indépendants entre eux et indépendant des entrées, on peut écrire :

$$\begin{aligned}\text{var}(z_i) &= \sum_{h=1}^{n_0+1} E((w_{ih}^1)^2 (x_h^0)^2) + \sum_{k\neq m} E(w_{ik}^1 x_k^0 w_{im}^1 x_m^0) \\ &= \sum_{h=1}^{n_0+1} E((w_{ih}^1)^2) E((x_h^0)^2) + \sum_{k\neq m} E(w_{ik}^1) E(w_{im}^1) E(x_k^0 x_m^0)\end{aligned}$$

Comme les poids sont centrés d'après (III.16), nous retrouvons bien la relation (III.17). Puisque les poids sont répartis uniformément entre -M et M, leur variance est connue :

$$E((w_{ih}^1)^2) = \frac{M^2}{3} \tag{III.18}$$

d'où le choix de la borne M :

$$M = \sqrt{\frac{3\sigma_z^2}{\sum_{h=1}^{n_0} E((x_h^0)^2)}} = \frac{0{,}87}{\sqrt{\sum_{h=1}^{n_0} E((x_h^0)^2)}} \tag{III.19}$$

en imposant $\sigma_z = 0{,}5$ puisque l'on peut considérer la tangente hyperbolique comme étant linéaire entre -0,5 et 0,5.

Ceci nous permet de déterminer les poids et biais connectant la couche d'entrée à la couche cachée. Burel ne précisant pas le choix des poids et biais connectant la couche cachée à la couche de sortie, nous les déterminerons aléatoirement de manière uniforme entre -1 et 1.

Cette méthode d'initialisation sera appelée Bu (Burel) dans les simulations ultérieures.

III.3.2. Initialisation par estimation linéaire des poids initiaux

III.3.2.1. Utilisation des moindres carrés simples

Cette méthode d'initialisation est due à Chen et Nutter (1991). Le principe de cette méthode est d'estimer les poids et biais initiaux de chaque couche successivement. Pour cela, ils proposent d'utiliser l'algorithme des moindres carrés simples.

La difficulté principale que l'on va rencontrer pour estimer les poids et biais du réseau correspond au fait que les sorties des neurones cachés sont inconnues. Pour contourner ce problème, Chen et Nutter (1991) proposent alors de donner dans un premier temps des valeurs aléatoires uniformément distribuées entre -1 et +1 aux poids et biais compris dans la matrice WB1. Il est alors aisé de calculer les sorties des neurones de la couche cachée regroupées dans la matrice $\vec{X}1$ (la flèche indiquant une propagation de l'information) à l'aide des équations (III.1) et (III.2) à partir des informations fournies en entrée.

Une fois $\vec{X}1$ obtenu, et en posant $G = [B \quad \vec{X}1]$, il est possible d'estimer les poids et biais WB2 connectant la couche cachée à la couche de sortie par la relation :

$$WB2 = (G^T G)^{-1} G^T D \qquad \text{(III.20)}$$

où D est le vecteur des sorties désirées.

En posant $H = \vec{X}1\, W2 = D - b^2 B$, il est possible de calculer $\vec{X}1$ (la flèche indiquant une rétropropagation de l'information) à partir des informations fournies en sortie du réseau par les relations :

$$\vec{X}1 = \begin{cases} HW2^{-1} & \text{si } \text{rang}([W2^T\ H^T]) = \text{rang}(W2^T) = n_1 \\ H(W2^T W2)^{-1} W2^T & \text{si } \text{rang}([W2^T\ H^T]) = \text{rang}(W2^T) < n_1 \\ HW2^T(W2\ W2^T)^{-1} & \text{si } \text{rang}([W2^T\ H^T]) \neq \text{rang}(W2^T) \end{cases} \qquad \text{(III.21)}$$

Afin de mixer les informations fournies en entrée et en sortie du réseau, il est possible de créer maintenant une matrice $X1 = \vec{X}1 + \lambda\vec{X}1$, avec λ, un paramètre aléatoire compris entre 0 et 1. Il faut remarquer qu'une telle procédure risque de donner à des éléments de $\vec{X}1$ des valeurs n'appartenant pas à l'intervalle [-1; 1]. Il faut donc borner les éléments de la matrice $\vec{X}1$.

A partir de cette matrice qui combine les informations fournies par l'entrée et la sortie du réseau, nous allons chercher à estimer la matrice WB1 par les relations :

$$WB1 = \begin{cases} A^{-1}K & \text{si rang}([A \;\; K]) = \text{rang}(A) = n_0 + 1 \\ A^T(AA^T)^{-1}K & \text{si rang}([A \;\; K]) = \text{rang}(A) < n_0 + 1 \\ (A^TA)^{-1}A^TK & \text{si rang}([A \;\; K]) \neq \text{rang}(A) \end{cases} \qquad \text{(III.22)}$$

avec A = [B X0] et $K=g^{-1}(X1)$ où g est la fonction d'activation utilisée, ici la tangente hyperbolique.

L'ensemble de la procédure est itérable. Partant du principe qu'une procédure d'initialisation doit utiliser peu de temps de calcul, nous n'effectuerons qu'une seule itération. Cette méthode sera appelée CN (Chen et Nutter) lors des tests de simulation.

Pour notre part, nous proposons d'utiliser une version simplifiée de cet algorithme. En effet, l'algorithme de Nguyen et Widrow (1990) modifié par Demuth et Beale (1994) fournit très rapidement une bonne initialisation des poids et biais connectant la couche d'entrée à la couche cachée à l'aide des équations (III.10), (III.11) et (III.12). Nous proposons donc d'utiliser ces poids et biais pour calculer comme précédemment la matrice $\bar{X}1$, puis d'utiliser l'équation (III.20) afin d'obtenir une estimation correcte des poids et biais connectant la couche cachée à la couche de sortie. Cet algorithme sera appelé NWMCN (Nguyen et Widrow Modifié avec Chen et Nutter) durant les simulations.

III.3.2.2. Utilisation des moindres carrés orthogonaux

Lehtokangas *et al.* (1995) considèrent, de la même manière que Chen *et al.* (1991) pour les réseaux à base radiale, les perceptrons multicouches comme un modèle de régression dont les sorties des neurones cachés sont les régresseurs. Dans la phase d'initialisation, le problème est de choisir les meilleurs régresseurs possibles. Le modèle de régression a la forme suivante :

$$D = R\,WB2 + \varepsilon \qquad \text{(III.23)}$$

où R = [B X1] et ε représente le vecteur des résidus.

La matrice R regroupe les différents régresseurs expliquant la sortie désirée. Généralement, certains de ces régresseurs sont corrélés entre eux et il est difficile de calculer leurs contributions individuelles à l'explication de la sortie. En les orthogonalisant, il devient cependant possible de calculer leurs contributions individuelles. Pour cela, Lehtokangas et al. (1995) proposent d'utiliser l'orthogonalisation classique de Gram-Schmidt.

Nous commencerons par poser :

$$R = H\,U \tag{III.24}$$

où U est une matrice carrée de dimension $n_1 + 1$ que l'on peut écrire :

$$U = \begin{bmatrix} 1 & \alpha_{12} & \alpha_{13} & \cdots & \alpha_{1(n_1+1)} \\ 0 & 1 & \alpha_{23} & \cdots & \alpha_{2(n_1+1)} \\ 0 & 0 & 1 & & \alpha_{3(n_1+1)} \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

et H est une matrice de dimension $(n \,.\, n_1 + 1)$ possédant des colonnes orthogonales h_j.

Le modèle de régression peut alors être réécrit sous la forme :

$$D = H\,V + \varepsilon \tag{III.25}$$

où $V = U\,WB2$ est le nouveau vecteur des paramètres que l'on peut calculer par :

$$V = (H^T H)^{-1} H^T D \tag{III.26}$$

Nous allons maintenant définir plus précisément la matrice H en calculant les vecteurs colonnes h_j la constituant :

$$\begin{aligned} h_1 &= r_1 \\ \alpha_{ij} &= \frac{h_i^T r_j}{h_i^T h_i} \qquad \text{pour } i = 1, \ldots, (j-1) \text{ et } j = 2, \ldots, (n_1+1) \\ h_j &= r_j - \sum_{i=1}^{j-1} \alpha_{ij} h_i \quad \text{pour } i = 1, \ldots, (j-1) \text{ et } j = 2, \ldots, (n_1+1) \end{aligned} \tag{III.27}$$

Nous pouvons maintenant définir la contribution de chacun des régresseurs afin de rechercher le régresseur possédant la contribution la plus importante :

$$Err_j = \frac{V_j^2 h_j^T h_j}{D^T D} \quad \text{pour } j = 1, \ldots, (n_1+1) \tag{III.28}$$

Pour l'initialisation proprement dite, il faut construire beaucoup plus de régresseurs que l'on ne désire de neurones sur la couche cachée (par exemple $10*n_1$). Pour les construire, Lehtokangas *et al.* (1995) préconisent d'initialiser les poids et biais connectant la couche d'entrée aux neurones de la couche cachée à des valeurs prises de manière uniformément aléatoire entre -4 et 4. Nous avons cependant constaté qu'une telle initialisation conduit de

manière quasi systématique à la saturation totale de tous les neurones de la couche cachée ce qui peut amener des problèmes de division par 0 dans le calcul des α_{ij} à l'équation (III.27).

Nous avons donc choisi de sélectionner les poids initiaux en utilisant l'algorithme NW.

Il ne reste plus alors qu'à calculer la contribution de chacun des régresseurs ainsi créés, sélectionner et retirer de la base le meilleur, puis recommencer la procédure jusqu'à l'obtention des n_l régresseurs.

Les régresseurs sélectionnés nous permettent de calculer les sorties des neurones cachés et l'obtention des poids et biais connectant la couche cachée à la couche de sortie s'effectue de la même manière que précédemment à l'aide de l'algorithme des moindres carrés simple par l'équation (III.20). Cet algorithme sera appelé LSKHMNW (Lehtokangas Saarinen Kaski Huuhtanen Modifié avec Nguyen Widrow) durant les simulations.

III.3.3. Initialisation à l'aide de prototypes

Cette méthode d'initialisation est due à Denoeux et Lengellé (1993). Elle consiste en l'utilisation de prototypes sélectionnés dans les données du jeu d'identification après avoir effectué une transformation des entrées du réseau. Cet algorithme est principalement conçu pour des problèmes de classification, mais Denoeux et Lengellé en ont également créé une version pour l'identification que nous allons présenter. Le principe de cet algorithme d'initialisation consiste dans le choix de n_l vecteurs de référence P_i, i = 1, ..., n_l dont on connaît la sortie correspondante. Ces vecteurs de référence P_i sont sélectionnés parmi le jeu de données d'identification. Ils vont constituer les poids des n_l neurones de la couche cachée. Par analogie avec la classification, ces neurones doivent calculer la distance euclidienne entre le vecteur d'entrée du réseau X correspondant au vecteur d'information $\phi(k)$ et les vecteurs des poids constitués par les prototypes P_i :

$$\|X - P_i\|^2 = \|X\|^2 + \|P_i\|^2 - 2X\,P_i \qquad \text{(III.29)}$$

Ce résultat n'offre un intérêt pratique que si $\|X\|^2 = 1 \quad \forall X$ et $\|P_i\|^2 = 1$, i = 1, ..., n_l. Dans ce cadre, l'ensemble de l'information se trouve concentrée dans le calcul de $X\,P_i$ que l'on peut comparer à un seuil constitué par le biais b_i^1. Ce calcul de $X\,P_i$ revient à une somme des éléments du vecteur d'entrée pondérés par les poids regroupés dans le vecteur prototype. Un tel calcul correspond bien au travail effectué par les neurones cachés d'un perceptron multicouches. On va donc chercher à rendre X et P_i de norme 1.

Pour cela, les vecteurs d'information $\phi(k) = [\phi_1(k) \cdots \phi_{n_0}(k)]^T$, k = 1, ..., n sont transformés en vecteurs d'entrée X(k) de norme 1 :

$$X(k) = \begin{bmatrix} \phi_1(k)/r \\ \vdots \\ \phi_{n_0}(k)/r \\ (1-\frac{1}{r^2}\sum_{h=1}^{n_0}\phi_h(k)^2)^{1/2} \end{bmatrix} \qquad W_i^1 = \begin{bmatrix} P_{i1}/r \\ \vdots \\ P_{in_0}/r \\ (1-\frac{1}{r^2}\sum_{h=1}^{n_0}P_{ih}^2)^{1/2} \end{bmatrix} \qquad \text{(III.30)}$$

avec r = max($\|\phi(k)\|$, k = 1, ..., n) où $\|\phi(k)\|$ est la norme euclidienne du vecteur d'information.

Un perceptron multicouches utilisant cette normalisation particulière des entrées du réseau, est alors comparable, dans son fonctionnement, à un réseau de neurones à fonction d'activation à base radiale.

Nous pouvons constater que le choix d'une telle normalisation introduit un neurone supplémentaire sur la couche d'entrée, et donc, de ce fait, n_1 paramètres supplémentaires. Nous obtenons alors, pour i = 1, ..., n_1 et pour h = 1, ..., n_0+1 les poids w_{ih}^1. Les biais b_i^1 constituant les seuils de comparaisons sont sélectionnés comme des valeurs aléatoires choisies uniformément entre -1 et 0.

Le choix des poids et biais connectant la couche cachée à la couche de sortie est simple. Le biais b^2 est pris égal à 0. Les poids w_i^2 sont pris égaux à la sortie désirée correspondant au prototype P_i.

Il est possible, afin de résoudre le problème causé par la présence de valeurs aberrantes dans le jeu d'identification, de ne prendre que 95% des données pour calculer r. Les données de norme supérieure à r sont, soit fixées de norme r, soit retirées du jeu de données. Le choix des vecteurs prototypes peut s'effectuer à l'aide de l'algorithme mis au point par Yin *et al.* (1990). Cet algorithme permet, non seulement de sélectionner les meilleurs prototypes pour modéliser le système, mais aussi de déterminer le nombre de prototypes à utiliser, et donc, le nombre de neurones cachés à introduire dans le perceptron multicouches. Cependant, l'utilisation d'un algorithme de sélection des prototypes risque de toujours fournir les mêmes prototypes, et alors l'apprentissage parviendrait toujours à un même minimum. Afin d'éviter ce problème, et dans le but de simplifier la procédure, nous avons préférer sélectionner aléatoirement nos prototypes.

Cet algorithme sera appelé DL (Denoeux et Lengellé) durant les tests effectués sur un exemple de simulation.

III.4. Application à des exemples de simulation

Afin de comparer les diverses méthodes d'initialisation des poids présentées ici, trois exemples de simulation vont être mis en place. Le premier de ces exemples correspond à l'apprentissage de la sortie d'un système non bruité construit sur la base d'un perceptron multicouches dont on a fixé les poids. Les deux autres exemples sont des modèles non-linéaires classiques tirés de la littérature.

L'ensemble des méthodes d'initialisation présentées sera également comparé à une initialisation aléatoire où l'ensemble des poids et biais sont choisis aléatoirement de manière uniforme entre -1 et 1. Cette méthode supplémentaire sera ici dénommée Random.

Pour chacune des méthodes comparées, l'identification et la validation sont réalisées avec plusieurs jeux de poids et biais initiaux différents, et ce, pour chaque exemple. Rappelons tout d'abord les dénominations des différentes méthodes comparées :

- random : initialisation aléatoire des poids de manière uniforme entre -1 et 1,
- NW : Nguyen Widrow (1990),
- NWM : Nguyen Widrow Modifié (méthode que nous avons mise au point),
- Bu : Burel (1991),
- CN : Chen Nutter (1991),
- NWMCN : Nguyen Widrow Modifié avec Chen Nutter (méthode que nous avons mise au point),
- LSKHMNW : Lehtokangas Saarinen Kaski Huuhtanen (1995) Modifié avec Nguyen Widrow,
- DL : Denoeux Lengellé (1993).

L'algorithme d'apprentissage utilisé est l'algorithme de Levenberg-Marquardt (Norgaard, 1995), décrit dans le prochain chapitre. Cet algorithme présente l'avantage d'utiliser un paramètre permettant d'obtenir une approximation de la matrice Hessienne toujours inversible. Ce coefficient évolue au cours de l'apprentissage et un critère d'arrêt de l'apprentissage lui est associé lorsqu'il devient trop grand ce qui correspond à l'obtention d'un minimum (local ou global).

Pour chacun des algorithmes d'initialisation, trois indicateurs sont observées :

- le nombre d'itérations nécessaire pour obtenir un minimum (nb. itér.),
- le critère résiduel après convergence en identification (résid. ident.),
- le critère résiduel après convergence en validation (résid. valid.).

Le nombre d'itérations fournit une indication précise du temps de calcul puisque l'algorithme de minimisation du critère utilisé est le même pour l'ensemble des expériences. Le critère résiduel est défini par :

$$V = \frac{1}{n} \sum_{k=1}^{n} (y(k) - \hat{y}(k))^2 \qquad \text{(III.31)}$$

où $\hat{y}(k)$ est la sortie prédite par le perceptron multicouches à l'instant k.

Pour chacun des indicateurs étudiés, nous présenterons son minimum, son maximum et également, sa moyenne. Pour les valeurs minimales et maximales de l'indicateur *résid. ident.*, nous présenterons également les valeurs prises par l'indicateur *résid. valid.* correspondante que nous nommerons *max. valid.* et *min. valid.* Nous présenterons réciproquement les valeurs prises par l'indicateur *résid. ident.* correspondant aux minima et maxima de l'indicateur *résid. valid.* que nous nommerons respectivement *max. ident.* et *min. ident.*

III.4.1. Premier exemple de simulation

Le système que nous allons étudier ici est un système non-linéaire non bruité. Ce système comporte 10 entrées et une sortie. Chacune de ces entrées est une séquence pseudo-aléatoire évoluant dans une plage de variation propre à chaque entrée afin de donner une influence différente à chacune des entrées. Ces plages de variation sont respectivement : [0; 3], [10; 30], [−5; 10], [−40; 35], [−4; 50], [−40; 35], [−10; 75], [120; 140], [12; 16] et [−13; 25].

Le système supposé inconnu que nous allons identifier est constitué par un perceptron multicouche possèdant 10 neurones sur la couche d'entrée, 5 neurones sur la couche cachée et un seul neurone sur la couche de sortie. La sortie obtenue est présentée à la figure (III.1).

Nous avons effectué cette expérience afin qu'aucun problème de structure, parfaitement connue ici, ne vienne influer sur les résultats de l'apprentissage. En effet, en supposant que nous disposons de suffisamment d'exemples, puisque la sortie du système est non bruitée, et que l'architecture du modèle correspond exactement à l'architecture du système, seul les problèmes liés à l'obtention de minima locaux ou à la saturation d'un ou plusieurs neurones cachés nous empêcherons de retrouver le système exact.

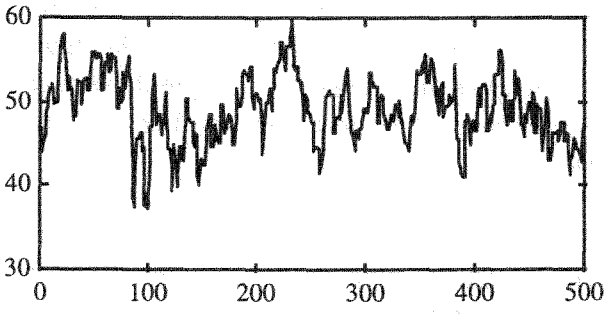


Fig. III.1. Sortie du système à étudier.

Pour chacune des méthodes d'initialisation comparées, l'identification est réalisée avec 50 jeux de poids et biais initiaux différents. Le nombre d'itération maximal est fixé à 500. Les résultats sont regroupés dans le tableau (III.1).

		Random	NW	NWM	Bu	CN	NWMCN	LSKHMNW	DL
nb. itér.	moyenne	405,38	493,82	375,88	490,54	399,5	298,88	490,12	488,68
	maximum	500	500	500	500	500	500	500	500
	minimum	40	191	45	207	9	35	6	185
rés. ident.	moyenne	1,5023	0,472	0,0662	0,8699	0,3212	0,0274	0,1466	0,0242
	maximum	12,9644	1,3224	0,2745	8,8162	1,3602	0,187	2,4002	0,0255
	minimum	0,1457	0,0372	$8{,}68.10^{-30}$	0,1562	0,0316	$8{,}28.10^{-30}$	0,0025	0,0213
	max. valid.	16,8335	2,7553	1,4064	17,3559	2,1972	1,0747	4,2965	0,1135
	min. valid.	0,7006	0,1541	$2{,}03.10^{-29}$	0,6748	0,3197	$2{,}78.10^{-29}$	0,0145	0,0942
rés. valid.	moyenne	4,6786	1,6019	0,3808	2,7729	1,2369	0,166	0,7997	0,1196
	maximum	54,1006	6,1677	1,4064	23,627	4,1174	1,0747	4,2965	0,1716
	minimum	0,7006	0,1541	$1{,}73.10^{-29}$	0,6381	0,3197	$2{,}06.10^{-29}$	0,0145	0,0595
	max. ident.	0,6168	0,9735	0,2745	1,0302	0,508	0,187	2,4002	0,022
	min. ident.	0,1457	0,0372	$1{,}04.10^{-29}$	0,2349	0,0316	$9{,}29.10^{-30}$	0,0025	0,0222
%(rés. ident. < 1)		94%	98%	100%	88%	92%	100%	98%	100%
%(rés. ident. < 10^{-1})		2%	4%	68%	0%	24%	94%	64%	100%
%(rés. ident. < 10^{-2})		0%	0%	32%	0%	0%	48%	4%	0%
%(rés. valid. < 1)		2%	20%	92%	6%	56%	96%	80%	100%
%(rés. valid. < 10^{-1})		0%	0%	32%	0%	0%	48%	4%	24%
%(rés. valid. < 10^{-2})		0%	0%	32%	0%	0%	40%	0%	0%

Tab. III.1. Résultats du premier exemple.

Ces premiers résultats nous montrent, tout d'abord, qu'une méthode d'initialisation des poids purement aléatoire (random) ainsi que les méthodes d'initialisation proposées par Nguyen et Widrow (1990) (NW) et Burel (1991) (Bu) ne semblent pas adaptées à l'initialisation du perceptron multicouches utilisé pour effectuer la modélisation de ce système. En effet, sur ce système, ces trois méthodes d'initialisation permettent rarement d'obtenir une valeur du critère résiduel en validation correct. Nous pouvons remarquer que ces trois algorithmes utilisent une initialisation des poids connectant la couche cachée à la couche de sortie aléatoire.

Seuls quatre algorithmes sont parvenus lors d'une expérience, au moins, à modéliser correctement le système étudié si l'on considère qu'un résultat acceptable de la modélisation correspond à un critère résiduel en validation inférieur à 10^{-1}. Nous pouvons remarquer que les résultats fournis par l'algorithme DL sont tous très proches les uns des autres. Cet algorithme est d'ailleurs le seul à fournir un critère résiduel en identification inférieur à 10^{-1} dans tous les cas.

Les figures (III.2) présentent la dispersion des 50 résultats pour chacune des méthodes comparées. L'histogramme noir présente le critère résiduel en identification, le gris, en validation.

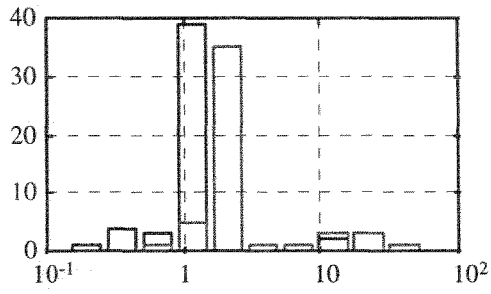


Fig. III.2a. Critères pour random.

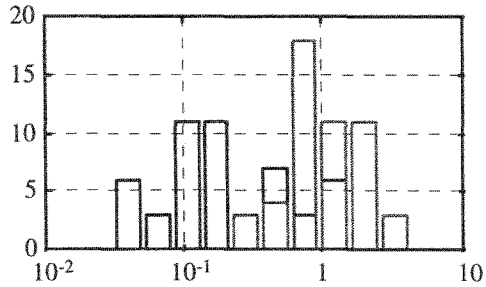


Fig. III.2e. Critères pour CN.

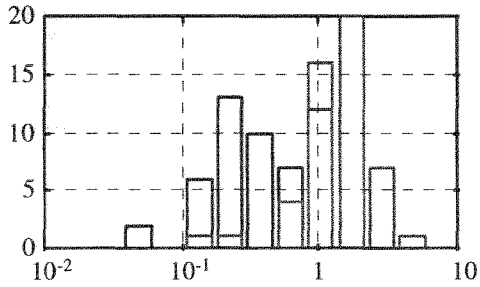


Fig. III.2b. Critères pour NW.

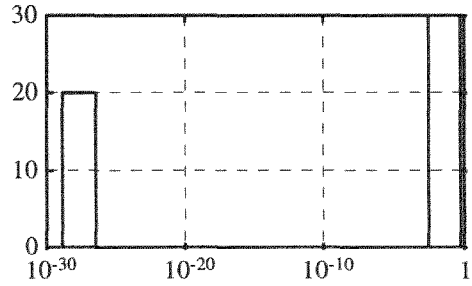


Fig. III.2f. Critères pour NWMCN.

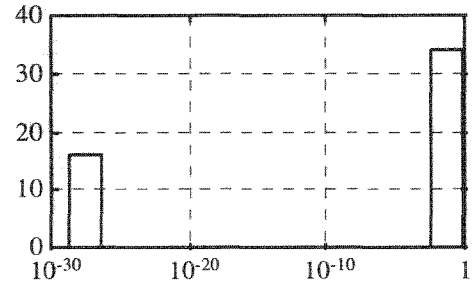


Fig. III.2c. Critères pour NWM.

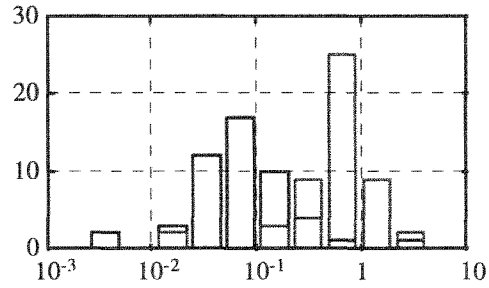


Fig. III.2g. Critères pour LSKHMNW.

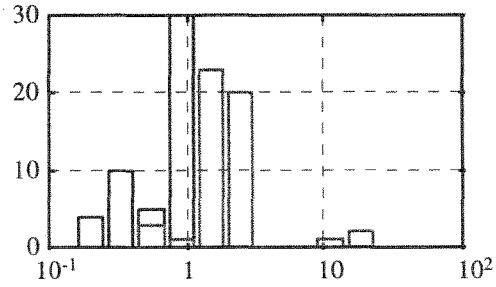


Fig. III.2d. Critère pour Bu.

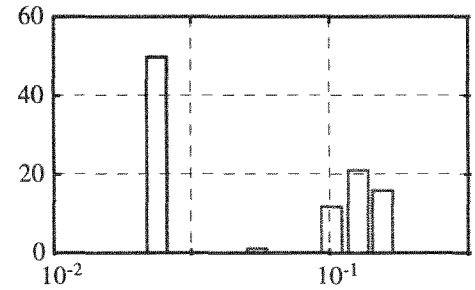


Fig. III.2h. Critères pour DL.

Il est remarquable que les algorithmes NWM et NWMCN présentent de grandes similitudes dans la distribution de leurs résultats. Par contre, LSKHMNW, qui fonctionne sur un principe très similaire à NWMCN, fournit des résultats nettement plus regroupés. La différence principale entre ces deux algorithmes se réduit au fait que LSKHMNW effectue le choix des meilleurs poids initiaux connectant la couche d'entrée à la couche cachée parmi un grand nombre de possibilités. Un tel algorithme d'initialisation risque alors de toujours placer

le début de l'apprentissage dans une même zone qui peut se trouver dans un bassin d'attraction d'un minimum local.

L'algorithme DL fournit toujours des résultats acceptables et très regroupés. Cependant, ce sont les algorithmes NWM et NWMCN qui fournissent les meilleurs résultats. Il est notable que l'algorithme NWMCN fourni dans 40% des cas un résultat très proche de 0.

Nous allons maintenant comparer les méthodes d'initialisation proposées sur deux exemples bruités.

III.4.2. Deuxième exemple de simulation

Le deuxième exemple que nous avons utilisé est un système non-linéaire NARX (modèle non-linéaire auto régressif à entrées exogènes) introduit par Chen *et al.* (1990) :

$$\begin{aligned} y(k) = {} & [0{,}8 - 0{,}5e^{-y^2(k-1)}]y(k-1) - [0{,}3 + 0{,}9e^{-y^2(k-1)}]y(k-2) \\ & + u(k-1) + 0{,}2\,u(k-2) + 0{,}1\,u(k-1)\,u(k-2) + e(k) \end{aligned} \qquad \text{(III.32)}$$

où u(k) et y(k) sont respectivement l'entrée et la sortie système à l'instant k et e(k) est un bruit normal centré de variance 0,16. Le vecteur d'entrée du réseau $\phi(k)$ est constitué des deux entrées retardées u(k-1) et u(k-2) et des deux sorties retardées y(k-1) et y(k-2).

Deux jeux de données sont créés, un pour l'apprentissage et un pour la validation, et sont exactement les mêmes pour l'ensemble des méthodes d'initialisation et pour l'ensemble des jeux d'initialisations de chaque méthode. Les entrées u(k) pour les deux jeux de données sont constituées par des valeurs aléatoires uniformément distribuées entre 5 et 10. Les jeux de données contiennent tous les deux n = 500 données. L'entrée et la sortie pour le jeu d'identification du système sont présentées respectivement aux figures (III.3) et (III.4) :

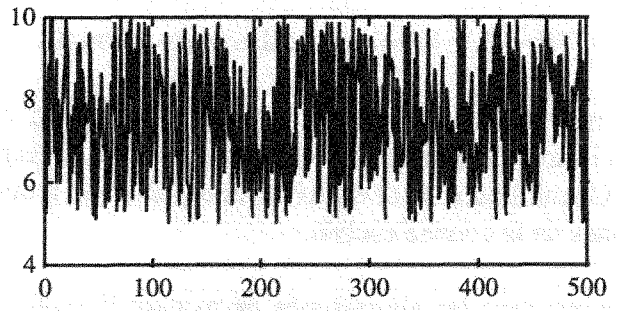


Fig. III.3. Entrée du système.

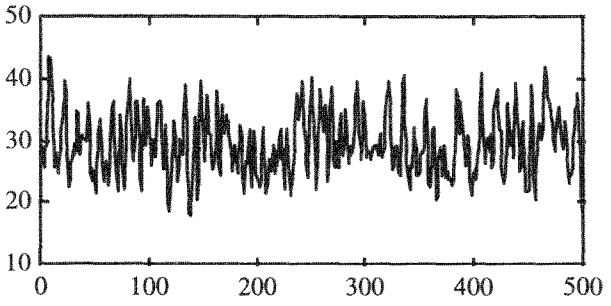


Fig. III.4. Sortie du système.

Pour chacune des méthodes comparées, l'identification est effectuée avec un réseau de neurones possédant 5 neurones sur la couche cachée. Les nombres d'initialisations différentes et d'itérations ont été fixés respectivement à 20 et 500 car ils sont apparus comme sufisant pour identifier correctement le système.

Les résultats obtenus sont regroupés dans le tableau (III.2).

		Random	NW	NWM	Bu	CN	NWMCN	LSKHMNW	DL
nb. itér.	moyenne	475,2	500	462,8	403,15	441,85	488,75	500	500
	maximum	500	500	500	500	500	500	500	500
	minimum	4	500	168	5	6	275	500	500
rés. ident.	moyenne	1,3657	0,1635	0,4276	5,7103	1,3734	0,1498	0,1512	0,1395
	maximum	24,1177	0,1884	3,3812	24,1177	24,1177	0,1671	0,1659	0,1407
	minimum	0,1468	0,1395	0,138	0,1618	0,1462	0,141	0,1458	0,1363
	max. valid.	20,1772	0,198	2,799	20,1772	20,1772	0,182	0,1824	0,17
	min. valid.	0,1784	0,1844	0,1815	0,1736	0,1757	0,1731	0,1714	0,1746
rés. valid.	moyenne	1,9798	0,2827	0,386	4,732	1,1871	0,1747	0,1751	0,1746
	maximum	20,1772	2,1055	2,799	20,1772	20,1772	0,1844	0,1905	0,1836
	minimum	0,1697	0,1706	0,1708	0,1736	0,1671	0,1691	0,1699	0,1699
	max. ident.	24,1177	0,1618	3,3812	24,1177	24,1177	0,1657	0,1573	0,1371
	min. ident.	0,1526	0,1519	0,1442	0,1618	0,152	0,151	0,1481	0,1407
%(rés. ident. < 0,18)		70%	75%	85%	50%	50%	100%	100%	100%
%(rés. valid. < 0,18)		50%	35%	60%	40%	35%	85%	75%	95%
%(rés. ident. & rés. valid. < 0,18)		50%	35%	60%	40%	35%	85%	75%	95%

Tab. III.2. Résultats du deuxième exemple.

Tout d'abord, nous remarquons que les algorithmes random, Bu et CN tombent tous les trois dans un minimum local très éloigné du minimum global et correspondant à une valeur de *rés. ident.* de 24,1177. Ce minimum local est très remarquable car il correspond à la saturation totale de tous les neurones de la couche cachée.

Nous pouvons noter que tous les algorithmes permettent d'approcher au moins une fois, avec 20 initialisations différentes, la valeur théorique du critère résiduel à obtenir correspondant à la variance du bruit introduit sur le système : 0,16.

Les algorithmes random, NW, Bu et CN fournissent cependant des résultats équivalents, où moins de 50% des initialisations aboutissent à des résultats acceptables.

L'utilisation de l'algorithme NWM permet d'améliorer fortement la probabilité d'obtenir un résultat correct. Cependant c'est avec l'algorithme DL et, dans une moindre mesure, NWMCN et LSKHMNW que nous obtenons les meilleurs résultats. Cependant, l'algorithme DL, comme nous l'avons déjà dit, impose d'utiliser une normalisation particulière et implique l'utilisation d'une entrée supplémentaire, et donc, des paramètres supplémentaires. Il est remarquable que, contrairement à l'exemple précédent, NWMCN et LSKHMNW fournissent des résultats très similaires. Cependant l'algorithme LSKHMNW est beaucoup plus complexe et long à mettre en place que l'algorithme NWMCN. Nous préférerons donc utiliser l'algorithme NWMCN pour initialiser ce type de système.

III.4.3. Troisième exemple de simulation

L'exemple de simulation que nous avons utilisé ici est également un système non-linéaire NARX (modèle non-linéaire auto régressif à entrées exogènes) introduit cette fois par Narendra et Parthasarathy (1990) :

$$y(k) = \frac{y(k-1)}{1+y^2(k-1)} + u^3(k-1) + e(k) \qquad \text{(III.33)}$$

où u(k) et y(k) sont respectivement l'entrée et la sortie système à l'instant k et e(k) est un bruit normal centré de variance 0,16. Le vecteur d'entrée du réseau $\phi(k)$ est constitué de l'entrée retardée u(k-1) et de la sortie retardée y(k-1).

Comme précédemment, deux jeux de données sont créés, un pour l'apprentissage et un pour la validation, et sont exactement les mêmes pour l'ensemble des méthodes d'initialisation et pour l'ensemble des jeux d'initialisations de chaque méthode. Les entrées u(k) du jeu d'identification sont constituées par des valeurs aléatoires uniformément distribuées entre 0 et 5. Les données du jeu de validation sont créées avec des entrées suivant la relation :

$$u(k) = 2{,}5\sin(2\pi k/25) + 0{,}5\sin(2\pi k/10) \qquad k = 1, \ldots, n \qquad \text{(III.34)}$$

Les jeux de données contiennent tous les deux n = 500 données. L'entrée et la sortie du système du jeu de validation sont présentées respectivement aux figures (III.5) et (III.6) :

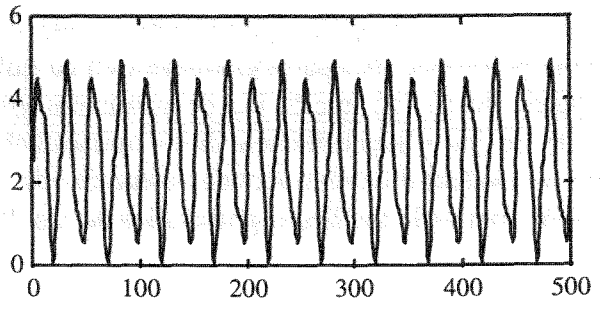


Fig. III.5. Entrée du système.

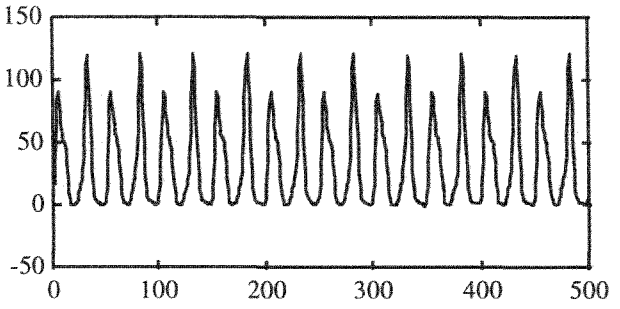


Fig. III.6. Sortie du système.

Pour chacune des méthodes comparées, l'identification et la validation sont réalisées avec 20 jeux de poids et biais initiaux différents. Pour les deux exemples de simulations, l'identification est effectuée avec un réseau de neurone possédant 5 neurones sur la couche cachée, une telle architecture de réseau étant suffisante pour identifier le système. Le nombre maximal d'itérations est ici fixé à 500.

Les résultats obtenus sont regroupés dans le tableau (III.3).

		Random	NW	NWM	Bu	CN	NWMCN	LSKHMNW	DL
nb. itér.	moyenne	485,7	500	500	500	480,75	483,95	500	457,3
	maximum	500	500	500	500	500	500	500	500
	minimum	214	500	500	500	118	179	500	67
rés. ident.	moyenne	0,3584	0,2281	0,1665	0,1754	0,5309	0,1688	0,2235	0,2169
	maximum	0,4789	0,4732	0,176	0,2011	2,7143	0,1852	0,4627	0,4829
	minimum	0,1622	0,1608	0,1585	0,1519	0,1513	0,1513	0,1515	0,149
	max. valid.	0,4821	0,4907	0,1991	0,218	2,2313	0,1984	0,5465	0,5045
	min. valid.	0,1887	0,183	0,1861	0,1704	0,1697	0,1724	0,1685	0,1705
rés. valid.	moyenne	0,3782	0,243	0,1922	0,1996	0,5213	0,1888	0,2449	0,2446
	maximum	0,5412	0,4907	0,2046	0,218	2,2386	0,1984	0,5465	0,6388
	minimum	0,1887	0,183	0,184	0,1704	0,1697	0,1683	0,1685	0,1705
	max. ident.	0,4765	0,4732	0,1723	0,2011	2,3473	0,1852	0,4627	0,4703
	min. ident.	0,1622	0,1608	0,1655	0,1519	0,1513	0,152	0,1515	0,149
%(rés. ident. < 0,18)		30%	30%	100%	65%	50%	80%	65%	75%
%(rés. valid. < 0,18)		0%	0%	0%	10%	5%	20%	10%	15%
%(rés. ident. & rés. valid. < 0,18)		0%	0%	0%	10%	5%	20%	10%	15%
%(rés. ident. < 0,19)		30%	55%	100%	80%	60%	100%	80%	80%
%(rés. valid. < 0,19)		5%	15%	50%	10%	15%	45%	20%	55%
%(rés. ident. & rés. valid. < 0,19)		5%	15%	50%	10%	15%	45%	20%	55%

Tab. III.3. Tableau récapitulatif des résultats du troisième exemple.

A partir de ces résultats, nous remarquons premièrement que ce système semble plus complexe à identifier que le précédent, tout particulièrement en ce qui concerne la validation du modèle. Ceci est en partie du au fait que le jeu de validation est ici construit avec des entrées de forme différente du jeu d'identification. Il est remarquable que les algorithmes random, NW et NWM ne permettent d'obtenir aucune valeur de *rés. valid.* inférieure à 0,18. Pourtant, l'algorithme NWM fournit la meilleure probabilité de résultats inférieurs à 0,18 en identification.

Même en étudiant les modèles fournissant des valeurs de *rés. valid.* inférieures à 0,19, seul les algorithmes NWM, Bu, NWMCN et DL fournissent une probabilité correcte d'obtenir un modèle efficace, aussi bien sur le jeu d'identification que celui de validation. Enfin, nous pouvons constater que parmi ces algorithmes, l'algorithme NWMCN fourni en moyenne les meilleures valeurs de *rés. valid.*

III.5. Conclusion

Au cours de ce chapitre, nous nous sommes intéressés aux problèmes liés à l'initialisation du perceptron multicouches. Ces problèmes sont principalement posés par la présence de minima locaux très éloignés du minimum global. De plus, une mauvaise initialisation des paramètres peut conduire à la saturation des fonctions d'activations des neurones de la couche cachée ce qui va compliquer la tâche de l'algorithme de minimisation du critère utilisé.

Pour pallier ces difficultés, divers auteurs ont proposé des algorithmes d'initialisation des poids que nous avons présentés. Ces algorithmes ont été classés en deux grandes classes. La première regroupe les algorithmes utilisables pour la modélisation en ligne de systèmes dont on ne dispose pas de jeu de données avant l'identification. La deuxième classe regroupe les méthodes nécessitant un jeu de données avant de commencer l'identification.

Parmi les 7 algorithmes présentés, 5 (NW, NWM, Bu, NWMCN, LSKH) ont en commun d'utiliser des poids et biais initiaux connectant la couche d'entrée à la couche cachée construits de manière uniforme.

Les 7 algorithmes ont été ensuite comparés sur trois exemples de simulation afin de sélectionner celui qui nous semble le plus adapté à l'initialisation du perceptron multicouches.

Il ressort de ces expériences que les algorithmes les plus efficaces et qui fournissent les résultats les plus constants sont les algorithmes NWM, NWMCN, que nous avons mis au point, DL mis au point par Denoeux et Lengellé (1993) et, dans une moindre mesure, LSKHMNW mis au point par Lehtokangas *et al.* (1995). Parmi ces algorithmes, seul NWM ne nécessite pas de jeu de données pour fonctionner. D'autre part, l'algorithme DL nécessite une normalisation particulière des entrées qui introduit une entrée supplémentaire, et donc, des poids supplémentaires dans le réseau.

L'initialisation n'est cependant que la première phase de l'identification. Le corps du problème est constitué par la recherche des paramètres du modèle. Cette recherche s'effectue à l'aide d'un algorithme de minimisation du critère. Nous présenterons donc, au chapitre suivant, les divers algorithmes de minimisation du critère utilisables ainsi que les problèmes qui leurs sont reliés.

III.6. Références

BILLINGS S.A., H.B. JAMALUDDIN (1991) 'A comparison of the backpropagation and recursive prediction error algorithms for training neural networks', *Mechanical Systems and Signal Processing*, Vol.5, 3, 233-255.

BUREL G. (1991) *Réseaux de neurones en traitement d'images : des modèles théoriques aux applications industrielles,* Thèse de doctorat de l'université de Bretagne Occidentale, spécialité électronique.

CHEN C.L., R.S. NUTTER (1991) 'Improving the training speed of three-layer feedforward neural nets by optimal estimation of the initial weights', *Proc. of International Joint Conf. on Neural networks*, Seatle, U.S.A., 8-12 Juil., 2063-2068.

CHEN S., S.A. BILLINGS, P.M. GRANT (1990) 'Non-linear system identification using neural networks', *Int. J. Control*, Vol.51, 6, 1191-1214.

CHEN S., C.F.N. COWAN, P.M. GRANT (1991) 'Orthogonal least squares learning algorithm for radial basis function networks', *IEEE Trans. on Neural Networks*, Vol.2, 2, 302-309.

CHEN Y., F. BASTANI (1992) 'ANN with two-dendrite neurons and its weight initialization', *Proc. of International Joint Conf. on Neural networks*, Baltimore, U.S.A., 7-11 Juin, 139-146.

LE CUN Y. (1985) 'Une procédure d'apprentissage pour réseaux à seuil asymétrique', *Proceedings of Cognitiva'85*, Paris, France, 599-604.

DEMUTH H., M. BEALE (1994) *Neural network toolbox user's guide*, V2.0, The MathWorks, Inc.

DENOEUX T, R. LENGELLE (1993) 'Initializing back propagation networks with prototypes', *Neural Networks*, Vol.6, 351-363.

HAGAN M.T., H. DEMUTH, M. BEALE (1995) *Neural network design*, PWS publishing company, Boston, U.S.A.

HERAULT J., C. JUTTEN (1994) *Réseaux neuronaux et traitement du signal,* Hermès, Paris, France.

KAROUIA M., R. LENGELLE, D. DENOEUX (1994) 'Weight initialization in BP networks using discriminant analysis techniques', *Proc. of NEURONIMES'94.*

KAROUIA M., D. DENOEUX, R. LENGELLE (1995a) 'Influence of weight initialization on multilayer perceptron performance', *Proc. of the International Conf. on Artificial Neural Networks ICANN'95*, Paris, France, 9-13 Oct., 33-38.

KAROUIA M., R. LENGELLE, D. DENOEUX (1995b) 'Performance analysis weight initialization algorithm', *Proc. of the European Symp. on Artificial Neural Networks ESANN'95*, Bruxelle, Belgique, 19-21 Avril, 347-352.

KIM L.S. (1993) 'Initializing weights to a hidden layer of a multilayer neural network by linear programming', *Proc. of International Joint Conf. on Neural networks*, Nagoya, Japan, 25-29 Oct., Vol.2, 1701-1704.

LARI-NAJAFI H., M. NASIRUDDIN, T. SAMAD (1989) 'Effect of initial weights on back-propagation and its variations', *IEEE International Conf. on Systems Man. and Cybernetics*, Vol.1, 218-219.

LEHTOKANGAS M., J. SAARINEN, K. KASKI, P. HUUHTANEN (1995) 'Initializing weights of a multilayer perceptron network using the orthogonal least squares algorithm', *Neural Computation*, Vol.7, 982-999.

NARENDRA K.S., K. PARTHASARATHY (1990) 'Identification and control of dynamical systems using neural networks', *IEEE Trans. on Neural Networks*, Vol.1, 4-27.

NGUYEN D., B. WIDROW (1990) 'Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights', *Proc. of International Joint Conf. on Neural networks*, Vol.3, 21-26.

RUMELHART D.E., J.L. MCCLELLAND (1986) *Parallel distributed processing,* The MIT Press, Cambridge, England.

SJÖBERG J., Q. ZHANG, L. LJUNG, A. BENVENISTE, B. DELYON, P.Y. GLORENNEC, H. HJALMARSSON, A. JUDITSKY (1995) 'Nonlinear black-box modeling in system identification : A unified overview', *Automatica*, Vol.31, 12, 1691-1724.

VAN DER SMAGT P.P., B.J.A. KRÖSE (1991) 'A real time neural robot controller', *Proceedings of the International Conference on Artificial Neural Networks*, Espoo, Finland, Elsevier Science Publishers, 351-356.

WESSELS L.F.A., E. BARNARD (1992) 'Avoiding false local minima by proper initialization of connections', *IEEE trans. on Neural Networks*, Vol.3, 6, 899-905.

YIN H., R. LENGELLE, P. GAILLARD (1990) 'NeoART : une variante du réseau ART2 pour la classification', *Proc. of NEURONIMES'90*, Paris, France, 171-179.

Chapitre IV

Algorithmes de minimisation du critère

IV.1. Introduction

IV.1.1. Problématique

La rétropropagation du gradient est l'une des plus importantes avancée dans l'histoire du calcul neuronal. Cet algorithme a été mis au point au milieu des années 80 par deux équipes indépendantes, en France et aux Etats-Unis (Le Cun, 1985; Rumelhart et McClelland, 1986), et il a permis de répondre aux limites du perceptron démontrées par Minsky et Papert (1969). La rétropropagation du gradient constitue actuellement la clef de voûte de l'apprentissage pour les réseaux de neurones et a été appliquée avec succès à une grande variété de problèmes tels que la classification, la reconnaissance de formes et, bien sûr, l'identification de systèmes non-linéaires.

Cependant, les algorithmes de minimisation du critère utilisant la rétropropagation du gradient se heurtent encore à un certain nombre de problèmes. Tout d'abord, ils sont souvent trop lents pour être utilisés dans certaines applications réclamant une grande rapidité de calcul (Han *et al.*, 1996). D'autre part, ces algorithmes n'effectuant pas une recherche globale du minimum d'un critère, contrairement, par exemple, aux algorithmes génétiques, ou encore à l'algorithme par recherche aléatoire adaptative (Walter et Pronzato, 1994; Sjöberg *et al.*, 1995), nous ne pouvons jamais être assurés d'avoir atteint le minimum global, mais seulement d'obtenir un minimum local. Ce dernier problème a été partiellement résolu au chapitre précédent, et peut l'être plus complètement en utilisant une méthode d'optimisation globale comme les algorithmes génétiques. Les algorithmes génétiques présentent cependant l'inconvénient d'être beaucoup plus long en temps de calcul que ceux utilisant la rétropropagation du gradient. Les problèmes posés par le risques de lenteur de calcul ont conduit à une évolution permanente des algorithmes de minimisation du critère depuis Rumelhart *et al.* (1986) (Billings et Jamaluddin, 1991; Ochiai *et al.*, 1992; Van der Smagt, 1994; Petridis et Paraschidis, 1995; Han *et al.*, 1996).

Au cours de ce chapitre, nous présenterons le principe de la rétropropagation du gradient ainsi que la plupart des méthodes de minimisation du critère reposant sur ce principe. Ces méthodes peuvent se diviser en deux grandes classes suivant leur cadre d'utilisation : les méthodes hors ligne (batch) et les méthodes en ligne (récursives). Il est notable que l'on peut toujours utiliser les méthodes récursives de manière hors ligne en les itérant. Nous présenterons également une méthode récursive que nous avons mise au point et qui, contrairement aux méthodes traditionnelles, n'utilise pas le principe de la rétropropagation du gradient. Nous finirons ce chapitre sur une étude de simulation présentant les intérêts découlant de l'utilisation de ce dernier algorithme.

IV.1.2. Rappel de l'architecture du réseau

Nous avons vu dans le chapitre II que nous restreignons notre étude aux perceptrons multicouches. Nous rappelons que, pour cette architecture, la première couche est constituée de neurones "transparents" qui distribuent juste les entrées du réseau aux neurones de la couche suivante, appelée couche cachée. Ces neurones calculent dans un premier temps la

somme pondérée des entrées du réseau, puis transforment cette somme à l'aide d'une fonction d'activation g qui est une tangente hyperbolique. La sortie x_i^1 du neurone i de la couche cachée peut alors être décrite par l'équation :

$$x_i^1 = g(\sum_{h=1}^{n_0} w_{ih}^1 x_h^0 + b_i^1) \qquad \text{(IV.1)}$$

où x_h^0, h = 1, ..., n_0 sont les entrées du réseau, w_{ih}^1 et b_i^1 sont les poids et biais correspondant au neurone i.

La couche de sortie est constituée d'un seul neurone n'effectuant qu'une somme pondérée des sorties des neurones cachées. La sortie z du réseau est alors décrite par l'équation :

$$z = \sum_{i=1}^{n_1} w_i^2 x_i^1 + b^2 = \sum_{i=1}^{n_1} w_i^2 g(\sum_{h=1}^{n_0} w_{ih}^1 x_h^0 + b_i^1) + b^2 \qquad \text{(IV.2)}$$

où x_i^1, i = 1, ..., n_1 sont les sorties des neurones cachés, w_i^2 et b^2 sont les poids et biais correspondant au neurone de sortie. L'ensemble des poids et biais du réseau peuvent être regroupé dans le vecteur des paramètres θ.

IV.2. Principe de la rétropropagation du gradient

Ce terme de rétropropagation du gradient peut être utilisé aussi bien pour décrire l'algorithme complet d'apprentissage utilisant le calcul du gradient adapté aux réseaux de neurones que pour décrire uniquement le calcul du gradient. Nous avons décider dans ce mémoire d'utiliser la deuxième possibilité.

L'algorithme de rétropropagation du gradient, également appelé "backpropagation", a été créé pour généraliser la règle de Widrow-Hoff dans les réseaux multicouches. Il a été mis au point simultanément par deux équipes indépendantes, en France et aux Etats-Unis (Le Cun, 1985; Rumelhart et McClelland, 1986). La nécessité d'un tel algorithme s'est fait sentir lorsque Minsky et Papert (1969) ont démontré les limites du perceptron.

Les méthodes d'adaptation de poids sont basées sur un apprentissage supervisé, c'est-à-dire qu'à chaque instant de l'apprentissage, on compare la sortie fournie par le réseau à la sortie réelle, puis on modifie les poids de manière à faire concorder les deux sorties.

Mais si l'on introduit une couche supplémentaire de neurones entre la couche d'entrée et la couche de sortie, comment évaluer l'impact de la modification d'un poids de la première couche sur le résultat? Ce problème se rattache au "credit-assignment problem" commun à l'ensemble des systèmes à apprentissage, qu'ils soient naturels ou artificiels (Davalo et Naïm, 1989).

Pour les perceptrons multicouches, ce problème peut se résumer à cette simple question : comment répercuter sur chacune des connexions le signal d'erreur qui n'est disponible que sur la couche de sortie, après avoir traversé plusieurs étapes non-linéaires?

C'est en réponse à cette question qu'a été créé l'algorithme de rétropropagation de l'erreur (Rumelhart et McClelland, 1986). La base de cet algorithme est d'utiliser comme fonctions d'activations des fonctions continues, continûment dérivables contrairement à la fonction seuil précédemment utilisée. Les fonctions d'activation utilisées sont, généralement, des fonctions sigmoïdes comme, par exemple, la tangente hyperbolique.

Cet algorithme repose sur le principe que, de même que l'on est capable de propager un signal venant de la couche d'entrée vers la couche de sortie, on peut, en suivant le chemin inverse, rétropropager l'erreur commise en sortie vers les couches internes. Mathématiquement, cet algorithme utilise simplement les règles de dérivation composées.

Pour cela, il nous faut connaître le gradient de la sortie estimée en fonction des différents paramètres du réseau. Pour les paramètres de la couche de sortie, nous obtenons:

$$\begin{aligned} \frac{\partial z}{\partial w_i^2} &= x_h^1 \\ \frac{\partial z}{\partial b^2} &= 1 \end{aligned} \qquad \text{(IV.3)}$$

Nous allons maintenant rechercher le gradient de la sortie estimée en fonction des paramètres de la couche cachée. Nous constatons qu'au cours de ce calcul intervient la dérivée de la fonction d'activation de la couche cachée $g(z_i^1)$ par rapport à z_i^1 notée $g'(z_i^1)$. Nous obtenons:

$$\begin{aligned} \frac{\partial z}{\partial w_{ih}^1} &= \frac{\partial z}{\partial x_i^1}\frac{\partial x_i^1}{\partial z_i^1}\frac{\partial z_i^1}{\partial w_{ih}^1} = w_i^2 g'(z_i^1) x_h^0 \\ \frac{\partial z}{\partial b_i^1} &= \frac{\partial z}{\partial x_i^1}\frac{\partial x_i^1}{\partial z_i^1}\frac{\partial z_i^1}{\partial b_i^1} = w_i^2 g'(z_i^1) \end{aligned} \qquad \text{(IV.4)}$$

Dans notre cas, la fonction d'activation utilisée étant la fonction tangente hyperbolique, sa dérivée peut s'écrire sous la forme simple : $g'(x) = 1 - g^2(x)$.

Ce gradient de la sortie estimée sera ensuite utilisé dans un algorithme de minimisation du critère.

IV.3. Identification hors-ligne ou 'batch'

IV.3.1. Méthodes utilisant la rétropropagation du gradient

Considérons un système dynamique SISO non linéaire discret possédant une forme générale de type NARX :

$$y(k) = f(\phi(k,\theta^\circ)) + e(k) \tag{IV.5}$$

où y(k) est la sortie du système
e(k) est un bruit de structure blanc et de moyenne nulle
f est une fonction non linéaire arbitraire
θ° est le vecteur des paramètres réels du système non-linéaire
$\phi(k)$ est le vecteur d'informations défini par :

$$\phi(k) = \left[y(k-1),\cdots,y(k-n_a),u(k-n_k),\cdots,u(k-n_k-n_b+1)\right]^T \tag{IV.6}$$

où u(k) est l'entrée du système à l'instant k
n_k représente le retard pur
n_a et n_b sont les ordres associés respectivement à y et u.

La modélisation à l'aide de perceptrons multicouches consiste à trouver les paramètres permettant d'émuler la fonction non linéaire f. Le prédicteur neuronal peut alors être écrit comme :

$$\hat{y}(k,\theta) = NN(\phi(k),\theta) \tag{IV.7}$$

où θ est le vecteur des paramètres correspondant aux différents poids du réseau. De la même manière que pour une méthode paramétrique, nous pouvons introduire l'erreur de prédiction :

$$\varepsilon(k,\theta) = y(k) - NN(\phi(k),\theta) = y(k) - \hat{y}(k,\theta) \tag{IV.8}$$

Afin de minimiser cette erreur, il est nécessaire de faire évoluer les paramètres du réseau. Pour cela, il nous faut sélectionner un critère à minimiser. Plusieurs critères possibles seront présentés lors du prochain chapitre. Nous allons donc introduire un critère assez général à minimiser :

$$V_n(\theta) = \frac{1}{n}\sum_{k=1}^{n} L(\varepsilon(k,\theta)) \tag{IV.9}$$

où L(.) est une fonction coût par observation (case cost function) à valeur scalaire.

Dans un algorithme itératif ou 'batch', l'estimateur $\hat{\theta}_n^i$ de θ est alors défini par la minimisation du critère précédent à l'itération i. Donc, connaissant $\hat{\theta}_n^i$, nous allons chercher à obtenir un $\hat{\theta}_n^{i+1}$ qui minimise le critère $V_n(\theta)$. Pour cela, nous allons supposer que $\hat{\theta}_n^i$ approche les paramètres réels à l'itération i ce qui nous permet d'effectuer un développement de Taylor d'ordre 2 autour de $\hat{\theta}_n^i$, nous obtenons (Ljung et Söderström, 1983) :

$$\begin{aligned} V_n(\hat{\theta}_n^{i+1}) &= V_n(\hat{\theta}_n^i) + V_n'(\hat{\theta}_n^i)^T\left[\hat{\theta}_n^{i+1} - \hat{\theta}_n^i\right] \\ &\quad + 1/2\left[\hat{\theta}_n^{i+1} - \hat{\theta}_n^i\right]^T V_n''(\hat{\theta}_n^i)\left[\hat{\theta}_n^{i+1} - \hat{\theta}_n^i\right] + o\left(\left|\hat{\theta}_n^{i+1} - \hat{\theta}_n^i\right|^2\right) \end{aligned} \qquad \text{(IV.10)}$$

où $o(x)$ représente une fonction telle que : $\lim_{|x|\to 0} o(x)/|x| \to 0$ et où $V_n'(\theta)$ et $V_n''(\theta)$ sont respectivement les dérivées première et seconde par rapport à θ de $V_n(\theta)$.

En supposant que la prochaine estimation $\hat{\theta}_n^{i+1}$ restera proche de $\hat{\theta}_n^i$, nous pouvons alors négliger le terme $o\left(\left|\hat{\theta}_n^{i+1} - \hat{\theta}_n^i\right|^2\right)$ dans l'équation (IV.10) (Ljung, 1987).

Le fait de négliger le troisième terme de l'équation (IV.10) conduit à des méthodes d'optimisation des paramètres du premier ordre (Van der Smagt, 1994). Ces méthodes sont connues sous le nom d'algorithme de descente du gradient ou "steepest descent". Dans les perceptrons multicouches, on lui donne le nom de "error back propagation" (Rumelhart et McClelland, 1986).

Pour rechercher la condition de stationnarité du critère, nous pouvons écrire la relation suivante :

$$\begin{aligned} \frac{\partial\left(V_n(\hat{\theta}_n^{i+1}) - V_n(\hat{\theta}_n^i)\right)}{\partial(\hat{\theta}_n^{i+1} - \hat{\theta}_n^i)} &= 0 \\ &= V_n'(\hat{\theta}_n^i) + V_n''(\hat{\theta}_n^i)\left[\hat{\theta}_n^{i+1} - \hat{\theta}_n^i\right] \end{aligned} \qquad \text{(IV.11)}$$

ce qui implique que:

$$\hat{\theta}_n^{i+1} = \hat{\theta}_n^i - \left[V_n''(\hat{\theta}_n^i)\right]^{-1}\left[V_n'(\hat{\theta}_n^i)\right] \qquad \text{(IV.12)}$$

Le choix d'utiliser une méthode du premier ordre va alors conduire à une simplification de l'équation (IV.12) (Narendra et Parthasarathy, 1991) :

$$\hat{\theta}_n^{i+1} = \hat{\theta}_n^i + \alpha_i \Delta_i \qquad \text{(IV.13)}$$

où α_i représente le pas de l'algorithme généralement choisi constant et Δ_i représente la direction de recherche.

Si le pas de l'algorithme varie au cours de l'identification, l'algorithme devient un algorithme du gradient avec pas adaptatif (Setiono et Liu, 1996). Pour l'algorithme "error back propagation" la direction de recherche Δ_i est choisie comme :

$$\Delta_i = -V_n'(\hat{\theta}_n^i) \tag{IV.14}$$

La direction de recherche Δ peut être améliorée en utilisant un terme additionnel pour transformer l'algorithme en 'error back propagation with momentum' (Rumelhart *et al.* 1986) :

$$\Delta_i = -V_n'(\hat{\theta}_n^i) + \beta_i \Delta_{i-1} \tag{IV.15}$$

Le terme additionnel a pour objet d'éviter les oscillations autour du minimum. Il est bien évident que cet algorithme est très sensible au choix des paramètres α_i et β_i (Sarkar, 1995). Une évolution de ces méthodes du gradient peut conduire à la mise en place de la méthode du gradient conjugué (Van der Smagt et Kröse, 1991; Battiti, 1992; Barnard, 1992). Dans cette méthode, la direction de la recherche est toujours choisie de manière que les minimisations effectuées aux étapes précédentes dans leurs directions respectives ne soient pas altérées. Ceci signifie que, lorsque à l'itération i, une minimisation du critère est effectuée dans la direction Δ_i, et conduisant aux paramètres $\hat{\theta}_n^{i+1}$, alors la direction de recherche Δ_{i+1} correspondant à $\hat{\theta}_n^{i+1}$ doit être orthogonale à Δ_i, Δ_{i-1}, ..., Δ_0. Cette méthode est simplement obtenue en choisissant β_i dans l'équation (IV.15) comme :

$$\beta_i = \frac{\left[V_n'(\hat{\theta}_n^i)\right]^T \left[V_n'(\hat{\theta}_n^i)\right]}{\left[V_n'(\hat{\theta}_n^{i-1})\right]^T \left[V_n'(\hat{\theta}_n^{i-1})\right]} \tag{IV.16}$$

Si l'on note $\psi(k,\hat{\theta})$ le gradient de la sortie estimée par le réseau $\hat{y}(k,\hat{\theta})$ par rapport au vecteur des paramètres $\hat{\theta}$:

$$\psi(k,\hat{\theta}) = \frac{\partial(\hat{y}(k,\hat{\theta}))}{\partial\hat{\theta}} \tag{IV.17}$$

Le gradient du critère s'écrit :

$$V_n'(\hat{\theta}_n^i) = \frac{\partial V}{\partial\theta} = -\frac{1}{n}\sum_{k=1}^{n} \psi(k,\hat{\theta}_n^i) L'(\varepsilon(k,\hat{\theta}_n^i)) \tag{IV.18}$$

où $L' = \frac{\partial L(\varepsilon)}{\partial\varepsilon}$. L'utilisation de l'expression (IV.18) dans les équations (IV.13) à (IV.16) complète les diverses méthodes du gradient.

En différentiant une seconde fois avec $L''=\frac{\partial^2 L(\varepsilon)}{\partial\varepsilon\partial\varepsilon^T}$, nous obtenons la matrice Hessienne du critère :

$$V_n''(\hat{\theta}_n^i)=\frac{\partial^2 V}{\partial\theta\partial\theta^T}=\frac{1}{n}\sum_{k=1}^{n}\left[\psi(k,\hat{\theta}_n^i)L''(\varepsilon(k,\hat{\theta}_n^i))\psi^T(k,\hat{\theta}_n^i)-\frac{\partial\psi(k,\hat{\theta}_n^i)}{\partial\theta^T}L'(\varepsilon(k,\hat{\theta}_n^i))\right] \quad \text{(IV.19)}$$

Pour estimer $\hat{\theta}_n^{i+1}$ en fonction de $\hat{\theta}_n^i$, avec une méthode du deuxième ordre, la méthode de Gauss-Newton (Sjöberg *et al.*, 1994), nous allons supposer que nous sommes proches des paramètres réels à l'itération i. Nous pouvons alors dire que $L'(\varepsilon(k,\hat{\theta}_n^i))$ est un bruit blanc non corrélé avec $\frac{\partial\psi(k,\hat{\theta}_n^i)}{\partial\theta^T}$, d'où :

$$\frac{\partial\psi(k,\hat{\theta}_n^i)}{\partial\theta^T}L'(\varepsilon(k,\hat{\theta}_n^i))=0 \quad \text{(IV.20)}$$

A l'aide de cette dernière hypothèse, nous pouvons écrire une approximation du Hessien que l'on notera R_n^i :

$$R_n^i=\frac{1}{n}\sum_{k=1}^{n}\psi(k,\hat{\theta}_n^i)L''(\varepsilon(k,\hat{\theta}_n^i))\psi^T(k,\hat{\theta}_n^i) \quad \text{(IV.21)}$$

L'utilisation des équations (IV.18) et (IV.21) dans l'équation (IV.12) nous donne la méthode de Gauss-Newton (Bloch *et al.*, 1996). Le point faible de cette méthode réside dans le fait que l'approximation de la matrice hessienne doit être inversée à chaque itération ce qui peut introduire des problèmes numériques lors de l'identification dans le cas où la matrice Hessienne est singulière, en particulier au voisinage de la solution. Pour pallier ce problème, on peut choisir d'utiliser l'algorithme de Levenberg-Marquardt (Levenberg, 1944; Marquardt, 1963; Declerq et DeKeyser, 1995, Hagan *et al.*, 1995) en utilisant l'approximation du Hessien suivante :

$$R_n^i=\frac{1}{n}\sum_{k=1}^{n}(\psi(k,\hat{\theta}_n^i)L''(\varepsilon(k,\hat{\theta}_n^i))\psi^T(k,\hat{\theta}_n^i)+\alpha I) \quad \text{(IV.22)}$$

où I est la matrice identité et α un scalaire positif ou nul. L'utilisation des expressions (IV.18) et (IV.22) conduit à l'algorithme suivant :

$$\hat{\theta}_n^{i+1}=\hat{\theta}_n^i-\left[\sum_{k=1}^{n}\psi(k,\hat{\theta}_n^i)L''(\varepsilon(k,\hat{\theta}_n^i))\psi^T(k,\hat{\theta}_n^i)+\alpha I\right]^{-1}\left[\sum_{k=1}^{n}\psi(k,\hat{\theta}_n^i)L'(\varepsilon(k,\hat{\theta}_n^i))\right]^T \quad \text{(IV.23)}$$

Nous obtenons alors un algorithme d'estimation hors-ligne des paramètres alliant la rapidité de convergence des méthodes du deuxième ordre à la sûreté de convergence vers un minimum

local des méthodes du premier ordre. En effet, le paramètre α peut varier au cours de la minimisation du critère et, lorsqu'il prend de faibles valeurs, l'algorithme de Levenberg-Marquardt se comporte comme l'algorithme de Gauss-Newton. Au contraire, lorsque le paramètre α prend de grandes valeurs, l'algorithme se comporte comme un algorithme du gradient (Hagan *et al.*, 1995).

Le paramètre α doit être initialisé à une faible valeur. Il existe plusieurs méthodes pour faire évoluer le paramètre α au cours de l'apprentissage.

La première consiste à tester l'évolution du critère au cours de l'itération. Si ce critère ne diminue pas, on multiplie α par un coefficient supérieur à 1 et on recommence l'itération. On effectue cette procédure tant que le critère ne diminue pas. Au contraire, lorsque le critère décroît, le paramètre α est alors multiplié par un coefficient positif inférieur à 1 en vue de la prochaine itération. Cette procédure a été proposée par Hagan *et al.* (1995).

Cet algorithme, déjà implanté dans Matlab (Demuth et Beale, 1994), sera celui que nous utiliserons lors d'identification hors-ligne.

De son coté Norgaard (1995) utilise un indicateur r(i) (Fletcher, 1987) pour gérer l'évolution du α :

$$r(i) = \frac{V_n(\hat{\theta}_n^i) - V_n(\hat{\theta}_n^{i+1})}{V_n(\hat{\theta}_n^i) - \sum_{k=1}^{n} \left[y(k) - \hat{y}\left(k\middle|\hat{\theta}_n^i\right) + \left[[R_n^i]^{-1} V_n'(\hat{\theta}_n^i)\right]^T \psi(k,\hat{\theta}_n^i)\right]^2} \qquad \text{(IV.24)}$$

Si cet indicateur donne une valeur supérieur à 0,75, on divise le paramètre α par 2. Au contraire, si r(i) est inférieur à 0,25, on multiplie α par 2.

IV.3.2. Algorithme effectuant une recherche globale

L'utilisation d'un algorithme génétique a pour objet de déterminer les poids correspondant au minimum global du critère. L'algorithme évolutionniste utilisé pour optimiser les paramètres des perceptrons multicouches que nous allons plus particulièrement présenter est du à Gégout et Rossi (1994). D'autres algorithmes utilisant le même principe existent (Thierens *et al.*, 1993; Whitley *et al.*, 1993).

Les algorithmes évolutionnistes sont basés sur l'étude du processus d'évolution naturelle. Les principes importants de ce type d'algorithme sont les suivants.

L'algorithme travaille sur une population d'individus. Chaque individu correspond à un point de recherche dans un espace de solutions (espace dans lequel on veut obtenir une solution optimale).

La population est initialisée aléatoirement. Elle évolue par le biais d'opérations telles que la mutation d'un individu ou la recombinaison d'individus.

L'adaptation d'un individu à son environnement est mesurée grâce à une fonction appelée ajustement (fitness) associant à chaque individu un réel positif.

L'algorithme que proposent Gégout et Rossi (1994) suit bien évidemment les lois générales précédemment décrites. Dans notre cas, nous utiliserons le critère quadratique comme fonction force.

Nous allons maintenant décrire les opérations de cet algorithme :

- L'opérateur de recombinaison.

L'algorithme effectue une recherche sur un espace vectoriel. C'est pourquoi les opérateurs sélectionnés doivent être indépendants du choix de la base de cet espace vectoriel. L'opérateur de recombinaison utilisé est communément appelé l'appariement arithmétique. Si deux individus I_1 et I_2 sont caractérisés par leurs extrémités u et v, l'opérateur arithmétique construit un descendant I_3 à partir de I_1 et I_2 tel que le vecteur caractérisant I_3 appartienne au segment [u, v].

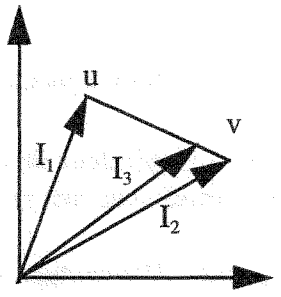


Fig. IV.1. L'opérateur de recombinaison.

- L'opérateur de mutation.

Si un individu I_1 est caractérisé par un vecteur U d'extremité u, l'opérateur de mutation construit à partir de I_1 un individu I_2 caractérisé par un vecteur V d'extremité v obtenu en effectuant une translation du point u d'un vecteur λW où $\lambda \in [0, D]$ et W est choisi aléatoirement sur la sphère unité de l'espace vectoriel. Le paramètre D est fixé au début de l'algorithme. On a donc $V = U + \lambda W$.

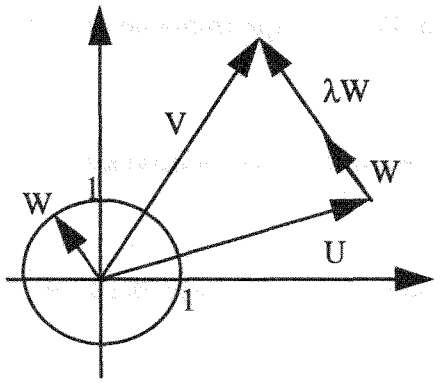


Fig. IV.2. L'opérateur de mutation.

Une fois les opérateurs définis, il nous reste à décrire le déroulement de l'algorithme.

- Etape 1 :
Définir aléatoirement une population initiale $\Pi_{t=0}$ contenant N individus.

- Etape 2 :
Répéter N fois les trois opérations suivantes :
- Sélectionner 2 individus aléatoirement dans Π_t.
- Produire un couple de descendant avec les parents avec les opérateurs de recombinaison ou de mutation.
- Mettre le meilleur de ces descendants déterminé à l'aide la fonction force dans une population temporaire $\tilde{\Pi}$.

- Etape 3 :
Comparer les meilleurs éléments de $\tilde{\Pi}$ et de Π_t et garder les N meilleurs des deux dans $\tilde{\Pi}$.

- Etape 4 :
Effectuer $\Pi_{t+1} = \tilde{\Pi}$, t = t + 1, et boucler à l'étape 2 jusqu'à l'obtention du minimum global.

La probabilité d'utiliser dans l'étape 2 l'opérateur de mutation peut être prise égale à 0,6. L'opérateur de recombinaison est alors utilisé avec une probabilité de 0,4.

Ce type d'algorithme présente le gros désavantage d'être particulièrement coûteux en temps de calcul par rapport aux algorithmes de recherche locale, malgré le fait que les algorithmes de recherche locale nécessitent généralement plusieurs initialisations différentes avant d'obtenir un résultat acceptable.

IV.4. Identification en-ligne ou "recursive"

IV.4.1. Méthodes utilisant la rétropropagation du gradient

Nous nous plaçons dans le même cadre que celui considéré dans le chapitre précédent et décrit par les équations (IV.5) à (IV.9), ce qui implique que nous pouvons introduire un critère à minimiser assez général :

$$V_t(\theta) = \gamma(t) \sum_{k=1}^{t} \beta(t,k) L(\varepsilon(k,\theta)) \qquad \text{(IV.25)}$$

où $\beta(t,k)$ est un coefficient d'oubli approprié avec $\beta(t,k) = \prod_{j=k+1}^{t} \lambda(j)$, $\beta(t,t) = 1$, $\gamma(t)$ est un facteur de normalisation déterminé par $\gamma(t) = \left[\sum_{k=1}^{t} \beta(t,k) \right]^{-1}$ et $\lambda(t) = \frac{\gamma(t-1)}{\gamma(t)} [1 - \gamma(t)]$,

$\varepsilon(k,\theta)$ est l'erreur de prédiction et L(.) est une fonction à valeur scalaire (Ljung et Söderström, 1983).

De manière similaire au cas itératif, nous pouvons utiliser des algorithmes du premier ordre et du second ordre. Les algorithmes du premier ordre sont de la forme :

$$\hat{\theta}(t) = \hat{\theta}(t-1) - \eta(t)\left[V_t'(\hat{\theta}(t-1))\right] \qquad \text{(IV.26)}$$

où $\eta(t)$ est le pas du gradient.

Cependant, comme pour le cas itératif, nous nous intéresserons principalement aux algorithmes du second ordre. En effectuant les mêmes hypothèses que dans le cas itératif, nous pouvons écrire que :

$$\hat{\theta}(t) = \hat{\theta}(t-1) - \left[V_t''(\hat{\theta}(t-1))\right]^{-1}\left[V_t'(\hat{\theta}(t-1))\right] \qquad \text{(IV.27)}$$

Si l'on note $\psi(k,\hat{\theta})$ le gradient de la sortie estimée par le réseau $\hat{y}(k,\hat{\theta})$ par rapport au vecteur des paramètres $\hat{\theta}$:

$$\psi(t,\hat{\theta}) = \frac{\partial(\hat{y}(t,\hat{\theta}))}{\partial\hat{\theta}} \qquad \text{(IV.28)}$$

Nous avons alors:

$$\begin{aligned} V_t'(\hat{\theta}(t-1)) &= -\gamma(t)\sum_{k=1}^{t}\beta(t,k)\,\psi(k,\hat{\theta}(t-1))\;L'(\varepsilon(k,\hat{\theta}(t-1))) \\ &= -\gamma(t)\lambda(t)\sum_{k=1}^{t-1}\beta(t-1,k)\,\psi(k,\hat{\theta}(t-1))\;L'(\varepsilon(k,\hat{\theta}(t-1))) \\ &\quad -\gamma(t)\psi(t,\hat{\theta}(t-1))\;L'(\varepsilon(t,\hat{\theta}(t-1))) \\ &= [1-\gamma(t)]\;V_{t-1}'(\hat{\theta}(t-1)) - \gamma(t)\;\psi(t,\hat{\theta}(t-1))\;L'(\varepsilon(t,\hat{\theta}(t-1))) \end{aligned} \qquad \text{(IV.29)}$$

où $L' = \frac{\partial L}{\partial \varepsilon}$, et en différenciant une seconde fois avec $L'' = \frac{\partial^2 L}{\partial\varepsilon\partial\varepsilon^T}$:

$$\begin{aligned} V_t''(\hat{\theta}(t-1)) &= [1-\gamma(t)]\;V_{t-1}''(\hat{\theta}(t-1)) - \gamma(t)\frac{\partial\psi(t,\hat{\theta}(t-1))}{\partial\theta^T}L'(\varepsilon(t,\hat{\theta}(t-1))) \\ &\quad +\gamma(t)\;\psi(t,\hat{\theta}(t-1))\;L''(\varepsilon(t,\hat{\theta}(t-1)))\;\psi^T(t,\hat{\theta}(t-1)) \end{aligned} \qquad \text{(IV.30)}$$

Pour estimer $\hat{\theta}(t)$ en fonction de $\hat{\theta}(t-1)$, nous allons effectuer un certain nombre d'hypothèses. Tout d'abord, nous allons supposer que la prochaine estimation $\hat{\theta}(t)$ restera proche de $\hat{\theta}(t-1)$ ce qui nous permet de prendre $V_t^{''}(\hat{\theta}(t)) = V_t^{''}(\hat{\theta}(t-1))$. Nous supposons maintenant que $\hat{\theta}(t-1)$ est déjà un estimateur correct à l'instant t-1 de telle sorte que : $V_{t-1}^{'}(\hat{\theta}(t-1)) = 0$. Finalement, comme nous supposons que nous sommes proches des paramètres réels à l'instant t-1, nous pouvons supposer que $L^{'}(\varepsilon(t,\hat{\theta}(t-1)))$ est un bruit blanc non corrélé avec $\frac{\partial\psi(t,\hat{\theta}(t-1))}{\partial\theta^T}$, d'où :

$$\frac{\partial\psi(t,\hat{\theta}(t-1))}{\partial\theta^T}L^{'}(\varepsilon(t,\hat{\theta}(t-1))) = 0 \qquad \text{(IV.31)}$$

A l'aide de ces différentes hypothèses, nous pouvons écrire une approximation du Hessien que l'on notera R(t) :

$$R(t) = [1-\gamma(t)]R(t-1) + \gamma(t)\,\psi(t,\hat{\theta}(t-1))\,L^{''}(\varepsilon(t,\hat{\theta}(t-1)))\,\psi^T(t,\hat{\theta}(t-1)) \qquad \text{(IV.32)}$$

ainsi qu'une approximation du gradient :

$$V_t^{'}(\hat{\theta}(t-1)) = -\gamma(t)\,\psi(t,\hat{\theta}(t-1))L^{'}(\varepsilon(t,\hat{\theta}(t-1))) \qquad \text{(IV.33)}$$

En utilisant les expressions (IV.32) et (IV.33) dans l'équation (IV.27), nous obtenons :

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \gamma(t)[R(t)]^{-1}\psi(t,\hat{\theta}(t-1))L^{'}(\varepsilon(t,\hat{\theta}(t-1))) \qquad \text{(IV.34)}$$

Maintenant, il apparaît que pour une forme récursive, il est préférable de ne pas inverser une matrice à chaque itération (Ljung, 1987; Chen *et al.*, 1990; Billings et Jamaluddin, 1991). Aussi, en posant $P(t) = \gamma(t)R^{-1}(t)$ et en utilisant le lemme d'inversion matricielle :

$$[A+BCD]^{-1} = A^{-1} - A^{-1}B\left[DA^{-1}B + C^{-1}\right]^{-1}DA^{-1} \qquad \text{(IV.35)}$$

avec A = [1 - γ(t)] R(t-1), B = $\psi(t,\hat{\theta})$ = D^T et C = γ(t) L"(ε(t,θ)), on obtient l'algorithme récursif.

L'algorithme récursif est décrit par les relations suivantes :

$$P(t) = \gamma(t)\left[\frac{\gamma(t-1)}{\gamma(t-1)[1-\gamma(t)]R(t-1)}\right.$$
$$\left.-\frac{\dfrac{\gamma(t-1)}{\gamma(t-1)[1-\gamma(t)]R(t-1)}\psi(t,\hat{\theta}(t-1))\,\psi^T(t,\hat{\theta}(t-1))\dfrac{\gamma(t-1)}{\gamma(t-1)[1-\gamma(t)]R(t-1)}}{\dfrac{1}{\gamma(t)L''(\varepsilon(t,\hat{\theta}(t-1)))}+\psi^T(t,\hat{\theta}(t-1))\dfrac{\gamma(t-1)}{\gamma(t-1)[1-\gamma(t)]R(t-1)}\psi(t,\hat{\theta}(t-1))}\right] \quad \text{(IV.36a)}$$
$$=\frac{1}{\lambda(t)}\left[P(t-1)-\frac{P(t-1)\,\psi(t,\hat{\theta}(t-1))\,\psi^T(t,\hat{\theta}(t-1))\,P(t-1)}{\dfrac{\lambda(t)}{L''(\varepsilon(t,\hat{\theta}(t-1)))}+\psi^T(t,\hat{\theta}(t-1))\,P(t-1)\,\psi(t,\hat{\theta}(t-1))}\right]$$

et :

$$\hat{\theta}(t) = \hat{\theta}(t-1) + P(t)\psi(t,\hat{\theta}(t-1))L'(\varepsilon(t,\hat{\theta}(t-1))) \quad \text{(IV.36b)}$$

Cet algorithme s'initialise en choisissant une matrice de variance-covariance initiale P(0) égale à la matrice identité I multipliée par un scalaire positif. Le choix de $\lambda(t)$ compris entre 0 et 1 permet d'accroître sensiblement les facultés d'adaptations du modèle lors d'une évolution de paramètres du système (Chen *et al.*, 1990). Il est possible de choisir pour $\lambda(t)$ une forme exponentielle ($\lambda(t) = \lambda_0\,\lambda(t\text{-}1) + (1 - \lambda_0)$) (Theilliol, 1993).

En posant $\lambda(t) = 1$, nous obtenons l'algorithme récursif traditionnel de l'erreur de prédiction que nous appellerons RPE dans la suite de ce mémoire (Thomas et Bloch, 1996). Nous n'étudierons pas au cours de ce travail l'influence du facteur d'oubli $\lambda(t)$. Aussi, c'est l'algorithme RPE que nous utiliserons lors d'identifications récursives.

IV.4.2. Méthode d'apprentissage par répartition des tâches

Cet algorithme, que nous avons développé, n'utilise pas la rétropropagation du gradient, mais exploite l'architecture du réseau pour faire travailler chaque neurone de manière individuelle.

Il est inspiré de l'algorithme d'estimation paramétrique des systèmes linéaires appelé MCMV (Moindres Carrés Multi-Variable) mis au point par Mielcarek (1990) et Mielcarek *et al.* (1990). Cet algorithme d'estimation paramétrique pour les systèmes linéaires permet l'identification de systèmes multi-entrées, multi-sorties, décomposable ou non en sous-systèmes multi-entrées, mono-sortie.

Dans la méthode présentée ici, chaque neurone peut être vu comme un sous-système à part entière avec ses entrées, sa sortie et bien sur ses paramètres. Les sorties des neurones cachés, qui ne sont pas mesurées, peuvent être calculées aussi bien à partir des entrées du réseau qu'en partant de la sortie du réseau.

Cependant, au début de l'identification, les poids initiaux ne permettent pas de calculer précisément les sorties des neurones cachés. La difficulté va donc être de propager l'information disponible à l'entrée du réseau vers la sortie, mais également de rétropropager l'information disponible à la sortie vers l'entrée du réseau.

L'initialisation de cet algorithme va consister à calculer, pour chaque neurone caché i et à partir des poids initiaux connectant la couche d'entrée et la couche cachée, la somme pondérée $\vec{z}_i^1(1)$ (la flèche indiquant le sens de propagation de l'information, ici, propagation avant) à l'instant 1 par la relation :

$$\vec{z}_i^1(1) = \sum_{h=1}^{n_0} w_{ih}^1 x_h^0(1) + b_i^1 \qquad \text{(IV.37)}$$

Par la suite, pour chaque instant t, l'algorithme va faire évoluer alternativement les poids et biais des neurones de la couche cachée, puis les poids et le biais de la couche de sortie. Il se décompose comme suit.

Pour chaque neurone i de la couche cachée, on fait évoluer les poids w_{ih}^1 et le biais b_i^1 en utilisant l'algorithme de l'erreur de prédiction RPE décrit par les relations (IV.36) avec, comme vecteur d'entrée $x_h^0(t)$ et comme sortie désirée $\bar{z}_i^1(t)$ (rétropropagation de l'information), la somme pondérée calculée à partir des poids connectant la couche cachée à la couche de sortie en utilisant la sortie désirée du réseau en écrivant la relation (IV.2) sous la forme :

$$\bar{z}_i^1(t) = g^{-1}(y(t) - b^2 - \sum_{\substack{j=1 \\ j\neq i}}^{n_1} w_j^2 . g(\bar{z}_j^1(t))) \qquad \text{(IV.38)}$$

où y(t) est la sortie désirée du réseau à l'instant t.

La difficulté va consister dans le fait que lorsque l'on calcule la somme pondérée du neurone caché i $\bar{z}_i^1(t)$ à partir de l'information existante en sortie du réseau, nous n'avons pas à notre disposition les sorties des autres neurones cachés j $\bar{z}_j^1(t)$ obtenues à partir de l'information fournie en sortie du réseau. Cependant, nous pouvons supposer que $\hat{\theta}(t)$ approche les paramètres réels à l'instant t ce qui nous permet d'écrire $\bar{z}_j^1(t) = \vec{z}_j^1(t)$. L'utilisation de cette relation dans l'équation (IV.38) va alors nous permettre d'estimer $\bar{z}_i^1(t)$.

Cette approximation peut cependant introduire certains problèmes dus à l'inversion de la fonction d'activation. Aussi nous estimerons $\bar{z}_i^1(t)$ par les relations :

$$\bar{z}_i^1(t) = g^{-1}(y(t) - b^2 - \sum_{\substack{j=1 \\ j\neq i}}^{n_1} w_j^2 . g(\bar{z}_j^1(t))) \qquad \text{(IV.39a)}$$

ou :

$$\bar{z}_i^1(t) = Cte \quad si \quad |y(t) - b^2 - \sum_{\substack{j=1 \\ j\neq i}}^{n_1} w_j^2 . g(\bar{z}_j^1(t))| > 1 \qquad \text{(IV.39b)}$$

où la constante Cte est ici prise égale à 1.

Ensuite, on calcule les nouvelles sommes pondérées $\bar{z}_i^1(t)$ par la relation (IV.37) en utilisant l'information fournie par l'entrée du réseau, puis, on fait évoluer les poids w_i^2 et le biais b^2 du neurone de la couche de sortie en utilisant l'algorithme de l'erreur de prédiction RPE avec, comme vecteur d'entrée $\bar{z}_i^1(t)$ et comme sortie désirée y(t). Finalement, on reconstruit les sommes pondérées $\bar{z}_i^1(t+1)$ à partir des équations (IV.39a) et (IV.39b) en utilisant l'information fournie par la sortie désirée du réseau y(t+1), et il ne nous reste plus qu'à recommencer l'algorithme au début avec les données fournies à l'instant suivant. Le déroulement de cet algorithme peut être résumé par l'organigramme suivant :

initialisation : Calcul des sommes pondérées des neurones cachés en effectuant une propagation 'avant' de l'information par l'équation (IV.37).

étape 1 : Calcul des sommes pondérées des neurones cachés en effectuant une propagation 'arrière' de l'information par les équations (IV.39a, b).

étape 2 : Evolution des poids connectant la couche d'entrée à la couche cachée en utilisant l'algorithme de l'erreur de prédiction et les sommes pondérées des neurones cachés calculées à l'étape 1. Cette évolution se fait neurone caché par neurone caché.

étape 3 : Calcul des sommes pondérées des neurones cachés en effectuant une propagation 'avant' de l'information par les équations (IV.37) et en utilisant les poids calculés à l'étape 2.

étape 4 : Evolution des poids connectant la couche cachée à la couche de sortie en utilisant l'algorithme de l'erreur de prédiction et les sommes pondérées des neurones cachés calculées à l'étape 3.

étape 5 : Test sur le résultat. Si le résultat est satisfaisant : fin de l'algorithme. Sinon : bouclage sur l'étape 1.

Nous pouvons remarquer que cet algorithme, et tout particulièrement les étapes 1 à 3, peut être facilement implanté et utilisé sur des ordinateurs à architecture parallèle ce qui améliore fortement le temps de calcul.

Les exemples de simulations seront cependant effectués avec une version séquentielle de l'algorithme. Cet algorithme sera appelé MRT (Méthode par Répartition des Tâches) durant les tests de simulation.

IV.5. Application à un exemple de simulation

Durant cette étude de simulation, nous allons comparer les algorithmes récursifs MRT (que nous avons mis au point) et RPE (Recursive Prediction Error). Le système que nous avons utilisé est un système non-linéaire NARX (modèle non-linéaire auto régressif à entrées exogènes) introduit par Chen *et al.* (1990) et déjà utilisé au chapitre précédent :

$$\begin{aligned} y(k) = {} & (0{,}8 - 0{,}5\, e^{-y^2(k-1)})\, y(k-1) \\ & -(0{,}3 + 0{,}9\, e^{-y^2(k-1)})\, y(k-2) \\ & +u(k-1) + 0{,}2\, u(k-2) + 0{,}1\, u(k-1)\, u(k-2) + e(k) \end{aligned} \qquad \text{(IV.40)}$$

où u(k), une séquence pseudo aléatoire d'amplitude comprise entre -1 et 2, et y(k) sont respectivement l'entrée et la sortie du système à l'instant k, e(k) est un bruit normal centré de variance 0,16. Le vecteur d'entrée du réseau $\phi(k)$ est constitué des deux entrées retardées u(k-1) et u(k-2) et des deux sorties retardées y(k-1) et y(k-2).

Le critère à minimiser que nous allons utiliser dans la phase d'apprentissage est le critère quadratique traditionnel.

Nous nous sommes tout d'abord intéressés au problème du temps de calcul. En effet, nous avons vu que lors de l'implantation de tels algorithmes sur un système industriel, il est important que le modèle fournisse la prédiction de la sortie suffisamment rapidement pour permettre d'autres traitements, comme une phase de diagnostic ou de contrôle, dans l'espace d'un cycle du système.

Bien que l'algorithme MRT puisse être implanté dans une architecture parallèle ce qui permet d'améliorer fortement le temps de calcul, nous allons comparer les temps de calcul d'une version séquentielle de cet algorithme avec les temps de calcul de l'algorithme RPE en fonction du nombre de neurones sur la couche cachée. La couche d'entrée comporte elle quatre neurones. Les résultats sont regroupés dans le tableau (IV.1).

nb. neurones	1	2	3	4	5	6	7
MRT	0'34"	1'0,1"	1'51"	3'14,5"	5'9,06"	6'57,3"	9'5,2"
RPE	3'23,7"	3'40,1"	3'51,1"	4'5,1"	4'29,6"	4'59,8"	5'33"

Tab. IV.1. Temps de calcul des deux algorithmes MRT et RPE.

Ces temps de calcul ont été obtenus en utilisant le logiciel MATLAB sur un Pentium 90 pour des jeux de données identiques comportant 5000 données. On constate immédiatement que l'algorithme MRT est plus rapide que l'algorithme RPE pour les perceptrons multicouches possèdant peu de neurones sur la couche cachée, mais que la tendance s'inverse lorsque le nombre de neurones sur la couche cachée augmente.

Nous allons maintenant nous intéresser aux performances comparées de ces deux algorithmes pour apprendre en ligne les paramètres d'un modèle neuronal du système décrit par l'équation (IV.40). Ce modèle neuronal possède 5 neurones sur sa couche cachée et les deux algorithmes utilisent les mêmes poids initiaux. La figure IV.3 présente l'évolution du critère quadratique au cours de l'apprentissage pour les deux algorithmes.

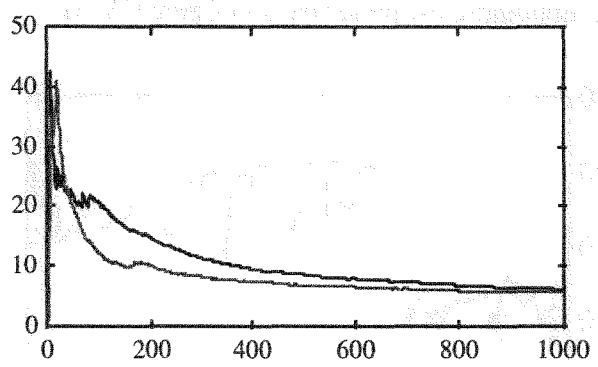


Fig. IV.3. Evolution du critère quadratique.
(MRT : gris - RPE : noir)

Nous constatons tout d'abord que les deux algorithmes ont des performances similaires au tout début de l'apprentissage, mais l'algorithme MRT est par la suite beaucoup plus rapide.

Ces deux algorithmes étant récursifs, ils ont pour but de pouvoir s'adapter rapidement à une évolution du système due, par exemple, à l'usure de certains éléments, à l'évolution de paramètres extérieurs, ou à un changement de point de fonctionnement non encore appris. Nous allons, dans les deux sections suivantes, nous intéresser à la comparaison de ces deux algorithmes face à ce type de problème.

IV.5.1. Apprentissage d'un nouveau point de fonctionnement

L'initialisation du perceptron mutlicouches est ici obtenu en effectuant un apprentissage batch au cours duquel le point de fonctionnement suplémentaire n'est pas appris. L'apprentissage hors ligne est donc effectué en utilisant des entrées comprises entre -1 et 2. Le perceptron multicouches que nous avons utilisé pour modéliser le système possède 5 neurones sur la couche cachée.

L'entrée du système comprenant le changement de point de fonctionnement à partir du point 300 est présentée à la figure (IV.4).

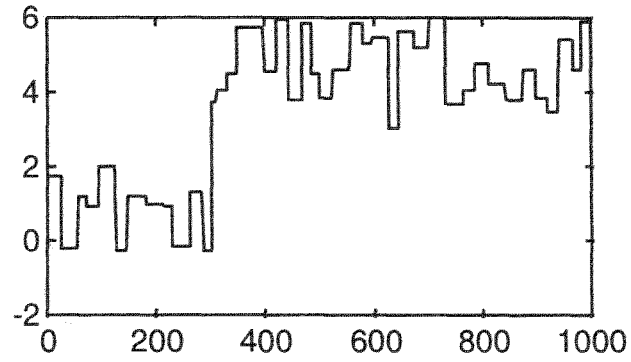


Fig. IV.4. Entrée du système.

La sortie du système à apprendre est présentée à la figure (IV.5).

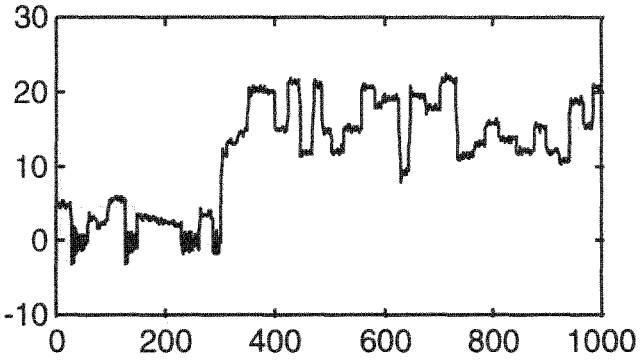


Fig. IV.5. Sortie du système.

La sortie nous montre que le changement de point de fonctionnement est conséquent. De plus, cette expérience valide notre choix d'utiliser une fonction identité comme fonction d'activation car l'utilisation d'une fonction du type sigmoïdale aurait imposé d'utiliser une normalisation de la sortie ce qui aurait rendu le modèle incapable de s'adapter à un tel changement de point de fonctionnement.

Les figures (IV.6) et (IV.7) vont respectivement présenter les résidus obtenus au cours du temps pour les algorithmes MRT et RPE.

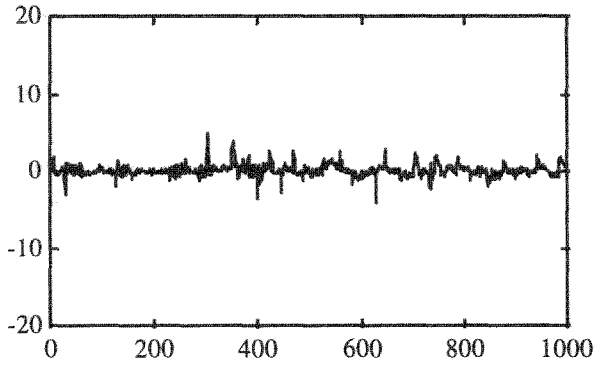


Fig. IV.6. Résidus obtenus avec l'algorithme MRT.

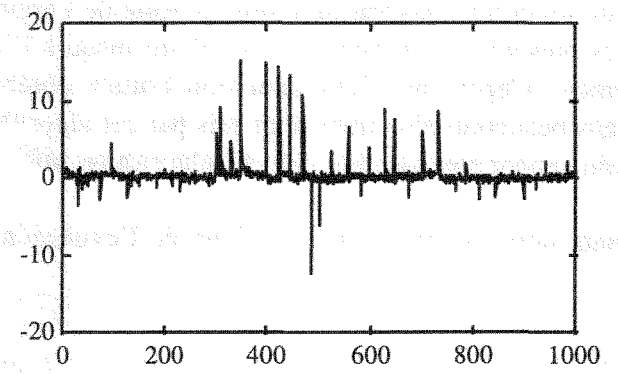


Fig. IV.7. Résidus obtenus avec l'algorithme RPE.

En comparant ces deux résidus, nous constatons que l'algorithme MRT permet d'apprendre beaucoup plus rapidement le nouveau point de fonctionnement que l'algorithme RPE. En effet, l'algorithme RPE fourni des résidus plus importants que l'algorithme MRT jusqu'à l'instant 700. L'algorithme MRT est donc plus rapide pour généraliser le modèle à d'autres points de fonctionnement.

Les figures (IV.8) et (IV.9) présentent respectivement l'évolution du critère quadratique au cours de l'apprentissage pour les deux algorithmes.

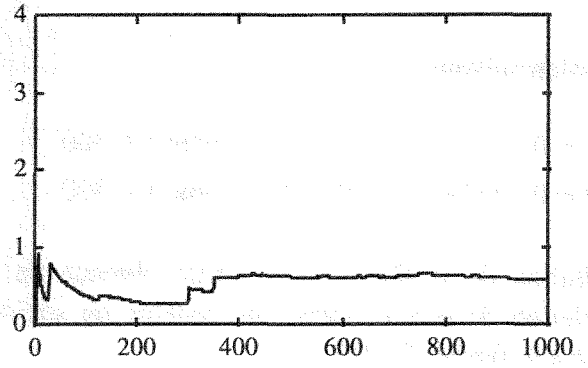


Fig. IV.8. Evolution du critère quadratique obtenue avec l'algorithme MRT.

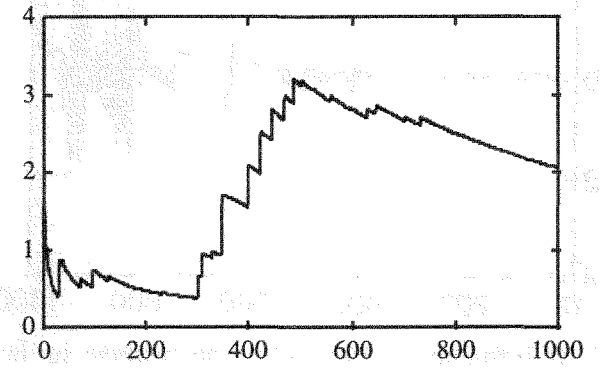


Fig. IV.9. Evolution du critère quadratique obtenue avec l'algorithme RPE.

En étudiant l'évolution du critère quadratrique tout au long de l'apprentissage, on constate que les deux algorithmes présentent une évolution similaire jusqu'à l'instant du changement de point de fonctionnement. Cependant, l'augmentation brutale observée pour l'algorithme RPE montre bien le temps beaucoup plus important pris par cet algorithme pour apprendre le nouveau point de fonctionnement comparativement à l'algorithme MRT.

Nous allons maintenant nous intéresser au problème de l'évolution du système en cours d'utilisation.

IV.5.2. Problème de l'évolution du système en cours d'utilisation

L'initialisation du perceptron multicouches, utilisant 5 neurones sur sa couche cachée, est ici également obtenue en effectuant un apprentissage batch du système initial. L'équation du système étudié est une modification du système précédemment décrit dans lequel l'évolution d'un paramètre a été introduite :

$$\begin{aligned} y(k) = {} & (0{,}8 - 0{,}5\, e^{-y^2(k-1)})\, y(k-1) \\ & -(0{,}3 + 0{,}9\, e^{-y^2(k-1)})\, y(k-2) \\ & +u(k-1) + \alpha\, u(k-2) + 0{,}1\, u(k-1)\, u(k-2) + e(k) \end{aligned} \qquad \text{(IV.41)}$$

où α est décrit par la relation suivante :

$$\begin{aligned} \alpha &= 0{,}2 && \text{pour } t \le 300 \\ \alpha &= 0{,}2 * (1 + (t - 300) / 700 && \text{pour } t > 300 \end{aligned} \qquad \text{(IV.42)}$$

L'influence de l'évolution du système sur la sortie obtenue en comparant la sortie du système subissant l'évolution de α à la sortie du système ne subissant pas l'évolution du paramètre α est présentée à la figure (IV.10).

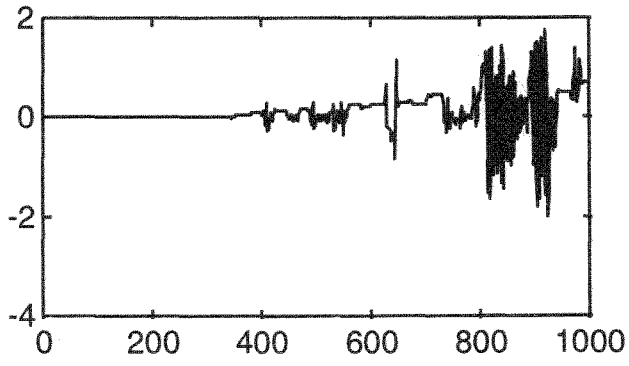


Fig. IV.10. Influence de l'évolution du système sur la sortie.

Les figures (IV.11) et (IV.12) présentent respectivement l'évolution du résidu obtenu avec les algorithmes MRT et RPE au cours du temps.

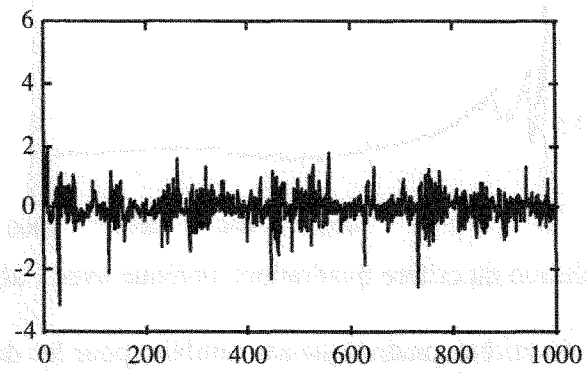


Fig. IV.11. Résidus obtenus avec MRT.

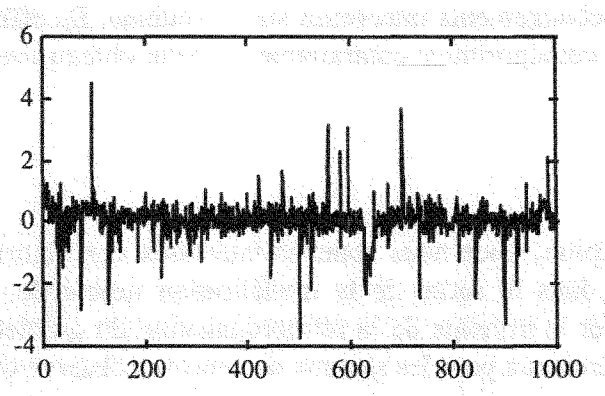


Fig. IV.12. Résidus obtenus avec RPE.

En comparant ces deux résidus, on constate que celui obtenu avec l'algorithme MRT est de meilleur qualité que celui obtenu avec l'algorithme RPE. Cependant, les capacités de ces deux algorithmes restent très proches.

Les figures (IV.13) et (IV.14) vont respectivement présenter l'évolution du critère quadratique au cours de l'apprentissage.

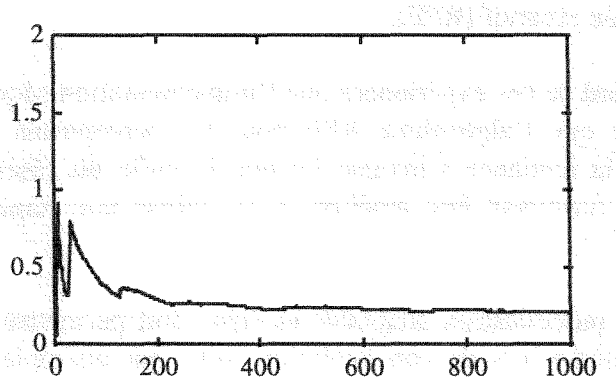


Fig. IV.13. Evolution du critère quadratique obtenue avec l'algorithme MRT.

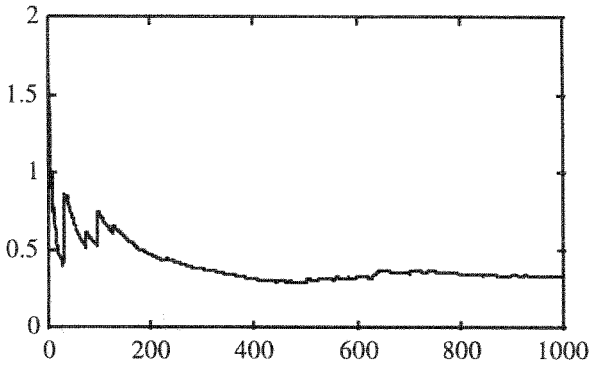


Fig. IV.14. Evolution du critère quadratique obtenue avec l'algorithme RPE.

Là encore l'évolution du critère quadratique est similaire pour les deux algorithmes jusqu'à ce que le changement de modèle devienne important. L'évolution lente du système est parfaitement prise en compte par l'algorithme MRT qui s'adapte plus rapidement que l'algorithme RPE aux changements intervenus sur le système. En effet, le critère quadratique ne se dégrade pas pour cet algorithme contrairement à celui obtenu avec l'algorithme RPE.

IV.6. Conclusion

Au cours de ce chapitre, nous nous sommes intéressés aux algorithmes d'estimation des paramètres utilisables dans le cadre de la modélisation neuronale. Pour cela, nous avons commencé par présenter le principe de la rétropropagation du gradient qui est à l'origine du regain d'intérêt des chercheurs pour les réseaux de neurones observé durant les années 80.

Nous avons ensuite présenté divers algorithmes de minimisation du critère que nous avons rangés en deux grandes classes. La première regroupe les algorithmes utilisables uniquement pour la modélisation hors-ligne (batch) des systèmes non-linéaires. La deuxième classe regroupe les algorithmes utilisables lors d'une modélisation en-ligne.

Parmi ces derniers se trouve un algorithme MRT que nous avons mis au point et qui n'utilise pas le principe de la rétropropagation du gradient. Afin de juger des capacités de cet algorithme, nous l'avons comparé sur un exemple de simulation avec l'algorithme classique de l'erreur de prédiction récursif (RPE).

Il ressort tout d'abord de ces expériences que l'implémentation séquentielle de l'algorithme MRT est plus rapide que l'algorithme RPE pour les perceptrons multicouches de faible dimension mais que la tendance s'inverse lorsque la taille du réseau croît. Cependant, ce temps de calcul peut fortement être amélioré si on utilise une implémentation parallèle de l'algorithme MRT.

L'utilisation d'une modélisation adaptable en-ligne doit permettre au modèle de suivre le comportement du système lors de son évolution suite, par exemple, à l'usure de certaines pièces. Un tel modèle doit également pouvoir, en cours d'utilisation, étendre son domaine de

validation à des points de fonctionnement non encore appris. Nous avons donc comparé le comportement des deux algorithmes MRT et RPE face à ces deux problèmes.

Nous avons constaté que, si les deux algorithmes permettent d'adapter le modèle en ligne, l'utilisation de l'algorithme MRT permet d'accroître fortement cette capacité d'adaptation du modèle. Il semble donc préférable d'utiliser l'algorithme MRT dans le cadre d'une stratégie de controle adaptatif.

Cependant un problème important rencontré lors de la modélisation de systèmes industriels est due à la présence de valeurs aberrantes dans les données. Une modélisation ne prenant pas en compte ce problème risque de produire des biais sur les paramètres et nous obtiendrons alors un modèle peu ou pas efficace. Nous présenterons donc, au chapitre suivant, divers critères robustes à la présence de valeurs aberrantes dans les données.

IV.7. Références

BARNARD E. (1992) 'Optimization for training neural nets', *IEEE Trans. on Neural Networks*, Vol.3, 2, 232-240.

BATTITI T. (1992) 'First and second-order methods for learning : between steepest descent and Newton's method', *Neural Computation*, Vol.4, 2, 141-166.

BILLINGS S.A., H.B. JAMALUDDIN (1991) 'A comparison of the backpropagation and recursive prediction error algorithms for training neural networks', *Mechanical Systems and Signal Processing*, Vol.5, 3, 233-255.

BLOCH G., P. THOMAS, M. OULADSINE, M. LAIRI (1996) 'On several outlier-robust training rules of neural networks for identification of non-linear systems', *preprints of 8th International Conference on Neural Networks and their Applications NEURAP'96,* Marseille, France, 20-22 March, 13-19.

CHEN S., S.A. BILLINGS, P.M. GRANT (1990) 'Non-linear system identification using neural networks', *Int. J. Control* , Vol.51, 6, 1191-1214.

LE CUN Y. (1985) 'Une procédure d'apprentissage pour réseaux à seuil asymétrique', *Proceedings of Cognitiva'85*, Paris, France, 599-604.

DAVALO E., P. NAÏM (1989) *Des Réseaux de Neurones*, Eyrolles, Paris, France.

DECLERCQ F., R. DEKEYSER (1995) 'Using Levenberg-Marquardt minimization in neural model based predicive control', *Preprints of International Workshop on Artificial Intelligence in Real-time Control*, Bled, Slovenia, 29 november - 1 december, 275-279.

DEMUTH H., M. BEALE (1994) *Neural network toolbox user's guide*, V2.0, The MathWorks, Inc.

FLETCHER R. (1987) *Practical methods of optimization*, Wiley.

GEGOUT C., F. ROSSI (1994) 'Initialisation des réseaux de neurones non récurents à coefficients réels par algorithmes évolutioninstes', *Proc. of Neural Networks and their Applications NEURAP'94*, Marseille, France, 15-16 déc., 416-424.

HAGAN M.T., H. DEMUTH, M. BEALE (1995) *Neural network design*, PWS publishing company, Boston, U.S.A.

HAN J., C. MORAGO, S. SINNE (1996) 'Optimization of feedforward neural networks', *Engng. Applic. Artif. Intell.*, Vol.9, 2, 109-119.

LEVENBERG K. (1944) 'A method for the solution of certain nonlinear problem in least squares', Quart. Appl. Math., Vol.2, 164-168.

LJUNG L. (1987) *System identification : theory for the user*, Prentice-Hall, Englewood Cliffs, N.J.

LJUNG L., T. SÖDERSTRÖM (1983) *Theory and practice of recursive identification* , The MIT press, London, England.

MARQUARDT D.W. (1963) 'An algorithm for least-squares estimation of nonlinear parameters', *J. Soc. Appl. Math.*, Vol 11, 2, 431-441.

MIELCAREK D. (1990) *Etude et développement de méthodes d'identification multi-variables - Application à un procédé chimique,* thèse de l'institut polytechnique de Lorraine, Spécialité automatique.

MIELCAREK D., C. LOFFLER, J. RAGOT, G. BLOCH (1990) 'A comparative study of some recursive parameters estimation algorithms for multivariable systems', *Proceedings of IMACS Annals on Computing and applied mathematics MIM-S^2'90.*, Bruxelle, Belgique, 3-7 sept.

MINSKY M., S. PAPERT (1969) *Perceptrons,* MIT Press, Cambridge.

NARENDRA K.S., K. PARTHASARATHY (1991) 'Gradient methods for the optimization of dynamical systems containing neural networks', *IEEE Trans. on Neural Networks*, Vol.2, 2, 252-262.

OCHIAI K., N. TODA, S. USUI (1992) 'New accelerated learning algorithm to reduce the oscillation of weights in multilayered neural networks', *Proceedings of the International Joint Conference on Neural Networks IJCNN'92*, Baltimore, U.S.A., june 7-11, Vol.1, 914-919.

PETRIDIS V., K. PARASCHIDIS (1995) 'On the properties of the feedforward method : a simple training law for on-chip learning', *IEEE Trans. on Neural Networks*, Vol.6, 6, 1536-1541.

RUMELHART D.E., G.E. HINTON, R.J. WILLIAMS (1986) 'Learning representations by back-propagating errors', *Nature*, Vol.323, 533-536.

RUMELHART D.E., J.L. MCCLELLAND (1986) *Parallel distributed processing,* The MIT Press, Cambridge, England.

SARKAR D. (1995) 'Methods to speed up error back-propagation learning algorithm', *ACM Computing Surveys*, Vol.27, 4, 519-542.

SETIONO R., H. LIU (1996) 'Improving backpropagation with feature selection', *Applied Intelligence*, Vol.6, 129-139.

SJÖBERG J., H. HJALMARSSON, L. LJUNG (1994) 'Neural networks in system identification', *preprints of 10th IFAC Symp. on System Identification SYSID '94,* Copenhagen, Denmark, 4-6 july, Vol.2, 49-72.

THEILLIOL D. (1993) *Identification de systèmes SISO linéaires et non linéaies par réseaux de neurones multicouches*, Thèse de doctorat de l'université Nancy I, spécialité automatique.

THIERENS D., J. SUYKENS, J. VANDEWALLE, B. DE MOOR (1993) 'Genetic weight optimization of a feedforward neural network controller', *Artificial Neural Nets and Genetic Algorithms Proc. of the International Conf.*, Innsbruck, Autriche, 658-663.

THOMAS P., G. BLOCH (1996) 'From batch to recursive outlier-robust identification of non-linear dynamic systems with neural networks', *Proceedings of the International Conference on Neural Networks ICNN'96,* Washington, D.C., U.S.A., 3-6 june, Vol. 1, 178-183.

VAN DER SMAGT P.P. (1994) 'Minimisation methods for training feedforward neural networks', *Neural Networks*, Vol.7, 1, 1-11.

VAN DER SMAGT P.P., B.J.A. KRÖSE (1991) 'A real time neural robot controller', *Proceedings of the In ternational Conference on Artificial Neural networks*, Espoo, Finland, Elsevier Science Publishers, 351-356.

WALTER E., L. PRONZATO (1994) *Identification de modèles paramétriques à partir de données expérimentales,* Masson, Paris, France.

WHITLEY D., S. DOMINIC, R. DAS, C.W. ANDERSON (1993) 'Genetic reinforcement learning for neurocontrol problems', *Machine Learning*, Vol.13, 2-3, 259-284.

Chapitre V

Choix du critère à minimiser
Robustesse aux valeurs aberrantes

V.1. Introduction

V.1.1. Problématique

La modélisation d'un système industriel repose principalement sur l'utilisation de jeux de données récoltées sur le système. Cependant, cette phase de modélisation impose généralement de choisir un certain nombre d'hypothèses initiales. Ces hypothèses portent notamment sur le modèle de distribution du bruit, l'indépendance des résidus. Ces hypothèses ne sont cependant pas toujours exactement vérifiées (Huber, 1981). Des divergences des données par rapport aux hypothèses peuvent alors être perçues comme des valeurs aberrantes. C'est dans ce cadre que Huber (1964) a défini la notion de robustesse aux incertitudes sur la distribution du bruit.

D'autre part, les données elles-mêmes ne sont pas exemptes de problèmes. En effet, les capteurs et actionneurs qui fournissent ces données peuvent subir divers événements comme un parasitage, une évolution d'un paramètre extérieur tel que la température, une usure, qui tous conduisent soit à une erreur systématique, biais ou dérive, soit à une erreur accidentelle ou valeur aberrante (Ragot *et al.*, 1990). De surcroît, le développement des moyens informatiques a induit une numérisation croissante des signaux, d'où l'utilisation de convertisseurs analogiques-numériques, de systèmes de transmission et de stockage de données pouvant également produire des valeurs aberrantes. Certains auteurs ont évalué le pourcentage de valeurs aberrantes dans les données industrielles dans une fourchette allant de 1 à 10% (Hampel *et al.*, 1987). Face à ce problème, Mosteller et Tukey (1977) ont défini la notion de résistance aux valeurs aberrantes.

Or les valeurs aberrantes sont des défauts difficilement détectables avant l'identification et peuvent produire des biais importants sur les paramètres du modèle. Il nous faut cependant trouver des mécanismes permettant de nous accommoder de ces erreurs lors de l'identification. En effet, les méthodes conventionnelles d'estimation de paramètres dans le cadre statistique, comme l'estimateur des moindres carrés, sont depuis longtemps connues comme très sensibles aux valeurs aberrantes (Poljak et Tsypkin, 1980). C'est pourquoi, lors de leur utilisation, ces estimateurs furent modifiés pour s'accomoder de la présence de valeurs aberrantes dans les données à l'aide de techniques statistiques dérivées des notions de "Winsorized Mean" ou "Trimmed Mean". Ces outils statistiques furent longtemps utilisés de manière heuristique, et ce n'est que vers le début des années soixante que ce problème attira l'attention de statisticiens tels que Tukey (1962) et surtout Huber (1964) qui proposa la notion d'estimation au sens du minimax asymptotique.

Le problème posé par la présence de valeurs aberrantes dans les données pour l'identification de systèmes non-linéaires n'a été abordé que depuis le début des années 90, et n'est encore aujourd'hui que peu considéré parmi les utilisateurs de réseaux de neurones (Chen et Jain, 1991; Pollard *et al.* 1992; Cichocki et Unbehauen, 1993; Bloch *et al.*, 1994; Liano, 1996). Notre objectif dans ce chapitre est d'introduire une approche robuste permettant de minimiser l'influence des valeurs aberrantes sur les paramètres d'un modèle neuronal. Nous présenterons donc les deux approches traditionnelles permettant de s'accommoder de la présence de valeurs aberrantes dans les données. Ces deux approches sont le principe du

maximum de vraissemblance et le principe de la fonction d'influence. Nous présenterons plus particulièrement trois critères robustes que nous avons adaptés à l'identification de systèmes non-linéaires à l'aide de perceptrons multicouches. Nous finirons par une étude de simulation présentant l'efficacité de ces critères et par une application industrielle sur un système statique. Etant donné que, dans la pratique, les valeurs aberrantes induites par des incertitudes sur les hypothèses et celles réellement incluses dans les données ne peuvent pas être différenciées, nous considérerons que les notions de résistance et de robustesse sont équivalentes (Hampel, 1971).

V.1.2. Définitions

Biais : Un biais est, au même titre qu'une dérive, une erreur systématique. Pour une valeur donnée de la grandeur à mesurer, une erreur systématique est soit constante (biais), soit à variation lente par rapport à la durée de mesure (dérive). Elle introduit donc un décalage entre valeur vraie et valeur mesurée. Les causes les plus fréquentes d'erreurs systématiques sont : erreurs sur la valeur d'une grandeur de référence, erreurs sur la sensibilité ou sur la courbe d'étalonnage du (des) capteurs, erreurs dues aux modes ou aux conditions d'emploi des instruments, erreurs dues à l'usure des instruments (Ragot *et al.*, 1990).

Minimax : Un estimateur robuste au sens du minimax est un estimateur qui minimise l'erreur effectuée sur les paramètres du modèle lorsque l'on maximise l'erreur effectuée sur les hypothèses de distribution du bruit (Huber, 1981).

Point d'effondrement : Le point d'effondrement (Hampel *et al.*, 1987) d'un estimateur est la proportion maximale des données pour laquelle il existe une borne dans le changement du résultat de l'estimateur quand cette partie des données est modifiée sans restriction. Un estimateur est résistant seulement si son point d'effondrement est supérieur à 0. Le point d'effondrement d'un estimateur est nul s'il n'existe pas de borne dans le changement des paramètres du modèle en modifiant une seule donnée : il suffit de prendre une seule donnée et de lui donner une valeur arbitrairement très grande pour changer le résultat de l'estimation de manière significative. Le point d'effondrement d'un estimateur fournit le pourcentage maximal de valeurs aberrantes dans les données pour lequel l'estimateur fournit encore des résultats satisfaisants. Par exemple, le point d'effondrement de la moyenne d'un signal est nul. La moyenne n'est pas résistante (Walter et Pronzato, 1994; Rapacchi, 1994).

Résistance : La "résistance" permet d'avoir une certaine insensibilité aux quelques erreurs localisées qu'il peut y avoir dans les données. Une méthode résistante doit produire un résultat qui ne changera que légèrement si une petite partie des données est remplacée par de nouvelles valeurs qui peuvent être très différentes des originales. Les méthodes résistantes doivent accorder une importance assez grande à la partie principale des données et laisser une part très faible aux possibles "lointains". Par exemple, la médiane est une statistique résistante tandis que la moyenne statistique ne l'est pas (Mosteller et Tukey, 1977).

Robustesse : La "robustesse" traduit l'insensibilité d'un critère à une non-conformité aux hypothèses sous-jacentes à un modèle probabiliste. Le plus souvent, nous voulons obtenir des

méthodes dont les hypothèses d'application ne soient pas trop restrictives. Les méthodes classiques d'estimation ou d'analyse statistique nécessitent très souvent une distribution Gaussienne des données. Les méthodes robustes garantiront que le résultat est bon pour une très grande collection de distributions sans pour autant être les meilleures pour une en particulier. Cette notion est conceptuellement distincte de la notion de robustesse. Cependant, les deux notions sont pour l'ensemble des applications pratiques synonymes (Hampel, 1971). L'analyse de variance classique, par exemple, nécessite que les résidus suivent une loi Gaussienne avec un écart type pour chacun des groupes étudiés; ce n'est pas une méthode robuste (Huber, 1981; Rapacchi, 1994).

Trimming mean : Si l'on suppose que les n observations x_i d'une statistique ont été ordonnées dans l'ordre croissant, $x_1 \leq x_2 \leq \cdots \leq x_n$, alors la statistique :

$$T = \frac{x_{g+1} + x_{g+2} + \cdots + x_{n-h}}{n - (g + h)}$$

est appelée la trimming mean, obtenue en ébarbant les g observations les plus petites et les h observations les plus grandes (Barnett et Lewis, 1984).

Valeur aberrante : Du point de vue statistique, une valeur aberrante est une valeur ne correspondant pas aux hypothèses émises pour construire le modèle. Du point de vue industriel, une erreur accidentelle, dont l'apparition, comme l'amplitude et le signe, sont aléatoires, est considérée comme une valeur aberrante. Les principales causes de ces erreurs sont : indéterminations intrinsèques des caractéristiques instrumentales, prise en compte de signaux parasites de caractère aléatoire, malfonctionnement d'un convertisseur analogique-numérique, erreurs de transmission (Ragot *et al.* 1990). Il est notable que le concept statistique de valeur aberrante inclut celui plus restreint du domaine industriel et nous retiendrons donc le concept statistique dans la suite de ce mémoire.

Winsorizing mean : Si l'on suppose que les n observations x_i d'une statistique ont été ordonnées dans l'ordre croissant, $x_1 \leq x_2 \leq \cdots \leq x_n$, alors la statistique :

$$T = \frac{g x_{g+1} + x_{g+1} + x_{g+2} + \cdots + x_{n-h} + h x_{n-h}}{n}$$

est appelée la winsorizing mean, obtenue en fenêtrant les g observations les plus petites et les h observations les plus grandes (Barnett et Lewis, 1984).

V.1.3. Algorithme de minimisation du critère

Nous avons présenté durant le chapitre IV, plusieurs algorithmes permettant de minimiser un critère général. De ce fait, il devient aisé d'introduire les divers critères robustes que nous allons présenter dans ce chapitre afin d'estimer les paramètres du réseau tout en évitant les biais sur les paramètres du modèle neuronal provoqués par les valeurs aberrantes.

L'ensemble des critères que nous allons présenter est bien entendu adaptables à chacun des algorithmes de minimisation présentés précédemment. Cependant, pour tester ces divers critères dans un exemple de simulation, nous avons sélectionné deux algorithmes de minimisation du critère, un hors-ligne et un en-ligne.

Nous rappelons tout d'abord que l'erreur de prédiction est représentée par l'expression suivante :

$$\varepsilon(k,\theta) = y(k) - \hat{y}(k,\theta) \tag{V.1}$$

où y(k) représente la sortie réelle du système à l'instant k et $\hat{y}(k,\theta)$ son estimée. θ est le vecteur rassemblant l'ensemble des biais et poids du perceptron multicouches.

Nous avons introduit un critère assez général à minimiser :

$$V_n(\theta) = \frac{1}{n}\sum_{k=1}^{n} L(\varepsilon(k,\theta)) \tag{V.2}$$

où n représente le nombre d'observation dans le jeu de données et L(.) est une fonction à valeur scalaire.

L'estimateur $\hat{\theta}_n^i$ de θ est alors défini par la minimisation du critère précédent à la $i^{ième}$ itération. Cette minimisation peut alors s'effectuer, dans le cas hors-ligne, à l'aide de l'algorithme de Levenberg-Marquardt :

$$\hat{\theta}_n^{i+1} = \hat{\theta}_n^i - \left[\sum_{k=1}^{n} \psi(k,\hat{\theta}_n^i) L''(\varepsilon(k,\hat{\theta}_n^i)) \psi^T(k,\hat{\theta}_n^i) + \alpha I\right]^{-1} \left[\sum_{k=1}^{n} \psi(k,\hat{\theta}_n^i) L'(\varepsilon(k,\hat{\theta}_n^i))\right]^T \tag{V.3}$$

où I est la matrice identité et α un scalaire positif ou nul.

Dans le cas où nous effectuons une estimation des poids du réseau de manière récursive, le critère à minimiser peut alors s'exprimer ainsi :

$$V_t(\theta) = \gamma(t)\sum_{k=1}^{t} \beta(t,k) L(\varepsilon(k,\theta)) \tag{V.4}$$

où $\beta(t,k)$ est un coefficient d'oubli approprié avec $\beta(t,k) = \prod_{j=k+1}^{t} \lambda(j)$, $\beta(t,t) = 1$, $\gamma(t)$ est un facteur de normalisation défini par $\gamma(t) = \left[\sum_{k=1}^{t} \beta(t,k)\right]^{-1}$ et $\lambda(t) = \frac{\gamma(t-1)}{\gamma(t)}\left[1-\gamma(t)\right]$. $\varepsilon(k,\Theta)$ est l'erreur de prédiction et où L(.) est une fonction à valeur scalaire.

Dans notre cas précis, nous ne désirons pas employer de coefficient d'oubli. Aussi, pour cela, il suffit de poser $\lambda(t) = 1$.

L'estimateur $\hat{\theta}(t)$ de θ est alors défini par la minimisation du critère précédent. Cette minimisation peut alors s'effectuer à l'aide de l'algorithme récursif de l'erreur de prédiction :

$$P(t) = P(t-1) - \frac{P(t-1)\psi(t,\hat{\theta}(t-1))\psi^T(t,\hat{\theta}(t-1))P(t-1)}{\frac{1}{L''(\varepsilon(t,\hat{\theta}(t-1)))} + \psi^T(t,\hat{\theta}(t-1))P(t-1)\psi(t,\hat{\theta}(t-1))} \qquad \text{(V.5a)}$$

et :

$$\hat{\theta}(t) = \hat{\theta}(t-1) + P(t)\psi(t,\hat{\theta}(t-1))L'(\varepsilon(t,\hat{\theta}(t-1))) \qquad \text{(V.5b)}$$

Nous allons maintenant présenter divers critères permettant de ne pas tenir compte des valeurs aberrantes dans l'estimation des poids d'un modèle neuronal d'un système non-linéaire. Deux approches différentes sont généralement utilisées afin de s'accommoder de la présence de valeurs aberrantes dans les données lors de l'identification d'un système non-linéaire: le principe du maximum de vraisemblance et la fonction d'influence.

V.2. Maximum de vraisemblance et robustesse

V.2.1. Principe du maximum de vraisemblance

La méthode du maximum de vraisemblance consiste, dans son principe, à choisir, parmi les différentes valeurs possibles des paramètres à estimer, ceux qui maximisent la probabilité d'obtenir les données qui ont été obtenues en respectant les hypothèses posées sur la distribution du bruit. Plus précisément, pour un vecteur de mesures Z et un vecteur de paramètres θ, on définit la fonction de vraisemblance à partir de la densité de probabilité conditionnelle de Z connaissant θ. Ces deux fonctions s'écrivent de façon identique, mais la densité de probabilité est une fonction de Z, alors que la fonction de vraisemblance F est une fonction de θ. Dans le cas d'une distribution continue, la fonction de vraisemblance du modèle est donnée par la fonction densité de probabilité jointe des données d'observations. A partir des erreurs de prédiction $\varepsilon(k)$ (V.1), supposées indépendantes entre elles et aléatoirement distribuées, nous pouvons définir la fonction de vraisemblance est le produit des densités de probabilité de chaque point :

$$F(\theta) = \prod_{k=1}^{n} \mathcal{P}(\varepsilon(k,\theta)) \qquad \text{(V.6)}$$

où $\mathcal{P}(\varepsilon)$ est la fonction densité de probabilité de ε.

Maximiser la fonction de vraisemblance précédente revient également à maximiser son logarithme, ou encore, minimiser l'opposé de son logarithme en fonction des paramètres :

$$\underset{\theta}{\text{Min}}\left(\sum_{k=1}^{n} -\log(\mathcal{P}(\varepsilon(k,\theta)))\right) = \text{Min}\left(\sum_{k=1}^{n} L(\varepsilon(k,\theta))\right) \qquad \text{(V.7)}$$

Les deux équations (V.6) et (V.7) peuvent bien entendu être également étendues au cas récursif. La fonction densité de probabilité la plus communément employée, dans ce cadre, est la distribution gaussienne. Cette fonction densité de probabilité conduit à un algorithme utilisant le critère classique des moindres carrés. De son côté, Liano (1996) a proposé d'utiliser comme fonction densité de probabilité la distribution de Cauchy :

$$\mathcal{P}(\varepsilon) = \frac{1}{\pi(1+\varepsilon^2)} \qquad \text{(V.8)}$$

Cette fonction densité de probabilité peut être représentée par la figure suivante :

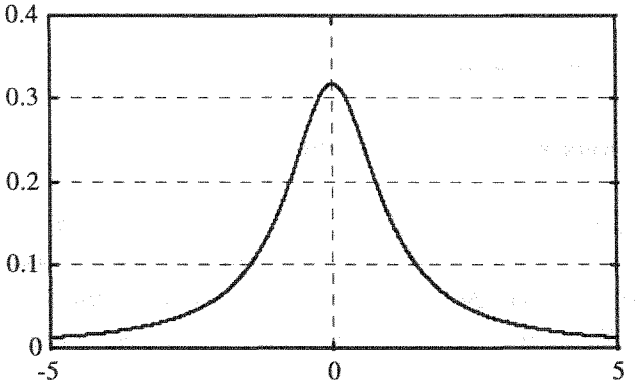


Fig. V.1. Fonction densité de probabilité de Cauchy

Elle présente la particularité, par rapport à une distribution gaussienne, d'avoir une forme très étendue. Si par hypothèse, le bruit suit une telle distribution, les résidus de grande amplitude ne seront pas considérés comme des valeurs aberrantes contrairement au cas où l'on suppose que le bruit suit une distribution Gaussienne.

L'utilisation de cette fonction densité de probabilité conduit à une fonction coût par observation permettant une accommodation très progressive aux résidus de grande amplitude généralement considérés comme des valeurs aberrantes. Cette fonction coût, utilisable dans les équations (V.2) et (V.4), s'écrit :

$$L(\varepsilon) = \log(1+\frac{1}{2}r^2) \qquad \text{(V.9a)}$$

La première dérivée de cette fonction coût s'écrit :

$$L'(\varepsilon) = \frac{r}{1+\frac{1}{2}r^2} \qquad \text{(V.9b)}$$

La forme de cette fonction est présentée à la figure(V.2).

Cependant, le choix d'une telle fonction coût implique obligatoirement une normalisation particulière de la sortie pour ramener l'écart-type du bruit vers une amplitude d'ordre 1. En effet, si le bruit est de variance très grande, le résidu sera également toujours d'amplitude supérieure à 1, et sera donc considéré comme une valeur aberrante par la fonction coût. Si, au contraire, le bruit est de variance très faible, le résidu sera toujours de faible amplitude et l'influence des valeurs aberrantes ne sera pas, ou peu, éliminée.

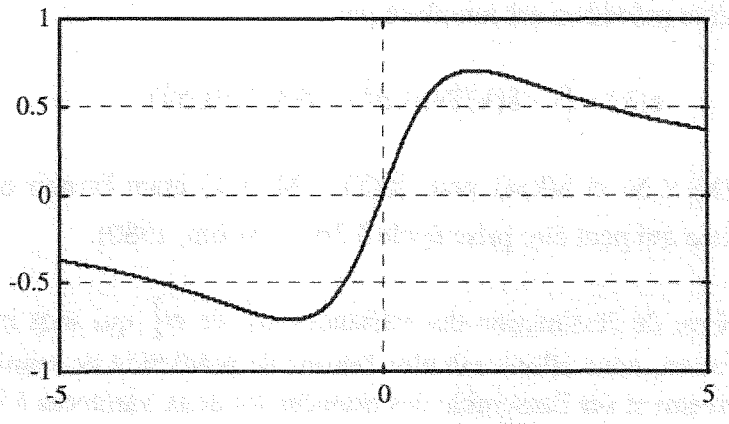


Fig. V.2. Dérivée de la fonction critère

Nous allons présenter deux autres fonctions densité de probabilité conduisant à des critères robustes plus aisés d'emploi.

V.2.2. Pondération par l'inverse de la variance du bruit

Ce critère repose principalement sur les travaux de Puthenpura et Sinha (1990), qui ont développé une méthode d'identification récursive robuste pour les systèmes dynamiques linéaires. Nous avons donc adapté cette méthode à la méthode de l'erreur de prédiction et ce, pour les systèmes non-linéaires. Elle est adaptable aussi bien au cas itératif (Bloch *et al.*, 1996) qu'au cas récursif (Bloch *et al.*, 1997). Cette méthode est basée sur l'utilisation d'un modèle particulier du bruit imputable à Huber (1964).

Nous considérons que le bruit e(t) qui contamine la sortie non bruitée appartient à la famille de distribution :

$$\mathcal{P}_\mu(e) = \left\{\mathcal{P} \middle| \mathcal{P} = (1-\mu)G + \mu H,\ 0 \le \mu \le 1\right\} \qquad (V.10)$$

où G est une distribution gaussienne correspondant au bruit sans valeur aberrante, H est une distribution symétrique arbitraire à forte dispersion représentant les valeurs aberrantes et μ représente la probabilité d'occurrence d'une valeur aberrante. En fait, H est supposée être également normale, mais avec une variance plus grande que celle de G :

$$e(k) \sim (1-\mu)N(0,\sigma_1^2) + \mu N(0,\sigma_2^2) \qquad (V.11)$$

où $N(0, \sigma^2)$ représente une distribution normale centrée de variance σ^2, avec $\sigma_1^2 \ll \sigma_2^2$.

Cependant, la probabilité μ d'occurrence d'une valeur aberrante est le plus souvent inconnue, et le modèle précédent est remplacé par :

$$\varepsilon(k) \approx [1-\delta(k)]N(0,\sigma_1^2) + \delta(k)N(0,\sigma_2^2) \qquad (V.12)$$

où $\delta(k)=0$ pour $|\varepsilon(k)| \le M$ et $\delta(k)=1$ pour $|\varepsilon(k)| > M$, $\varepsilon(k)$ étant l'erreur de prédiction et M une borne préassignée qui peut être prise égale à $3\sigma_1$ (Aström, 1980).

Il reste le problème de l'estimation des variances σ_1^2 et σ_2^2 qui sont inconnues. Pour ce faire, à chaque itération, nous allons calculer l'erreur de prédiction de manière traditionnelle, puis estimer récursivement sur l'ensemble des données les deux variances à l'aide des relations suivantes :

$$\begin{aligned} &\sigma_1^2(k) = \sigma_1^2(k-1) + \frac{1}{k-\tau}(\varepsilon^2(k) - \sigma_1^2(k-1)) && \text{pour } |\varepsilon(k)| \le 3\sigma_1(k-1) \\ &\sigma_1^2(k) = \sigma_1^2(k-1) && \text{autrement} \end{aligned} \qquad (V.13a)$$

et :

$$\begin{aligned} &\sigma_2^2(k) = \sigma_2^2(k-1) + \frac{1}{\tau}(\varepsilon^2(k) - \sigma_2^2(k-1)) && \text{pour } |\varepsilon(k)| > 3\sigma_1(k-1) \\ &\sigma_2^2(k) = \sigma_2^2(k-1) && \text{autrement} \end{aligned} \qquad (V.13b)$$

en prenant $\tau=0$, pour k=1 et $\tau=\tau+1$ lorsque $|\varepsilon(k)| > 3\sigma_1(k-1)$. $\sigma_2^2(0)$ peut être choisi égal à $\sigma_1^2(0)$. Initialement $\sigma_1^2(0)$ peut être choisi comme étant la variance des résidus obtenus avec les poids initiaux sur l'ensemble de jeu d'apprentissage. Comme nous pouvons le remarquer, τ est l'estimation du nombre de valeurs aberrantes.

Avec l'estimation (V.13) de la variance, nous pouvons construire la forme particulière du critère quadratique L dans les équations (V.2) pour la méthode itérative et (V.4) pour la méthode récursive correspondant à la fonction densité de probabilité définie en (V.10) :

$$L(\varepsilon(k,\theta)) = \frac{1}{\sigma^2(k)}\varepsilon^2(k,\theta) \qquad (V.14)$$

La valeur de la pondération $\frac{1}{\sigma^2(k)}$ utilisée dans l'équation (V.14) est choisie dans le but d'atténuer l'influence des valeurs aberrantes. Dans le cas d'une identification batch, le choix de la pondération est :

$$\frac{1}{\sigma^2(k)} = \frac{1}{[1-\delta(k)]\sigma_1^2(n)+\delta(k)\sigma_2^2(n)} \qquad (V.15a)$$

Dans le cas d'une identification récursive, nous choisissons :

$$\frac{1}{\sigma^2(k)} = \frac{1}{[1-\delta(k)]\sigma_1^2(k)+\delta(k)\sigma_2^2(k)} \qquad (V.15b)$$

Ces choix nous permettent de déterminer les valeurs de $L'(\varepsilon(k,\theta))$ et $L''(\varepsilon(k,\theta))$ dans les algorithmes (V.3) pour la méthode itérative et (V.5a et b) pour la méthode récursive :

$$L'(\varepsilon(k,\theta)) = \frac{1}{\sigma^2(k)}\varepsilon(k,\theta) \qquad (V.16a)$$

et :

$$L''(\varepsilon(k,\theta)) = \frac{1}{\sigma^2(k)} \qquad (V.16b)$$

Cet algorithme sera appelé Weighted Neural Networks (WNN) pour la version itérative et Recursive Weighted Neural Netwoks (RWNN) pour la version récursive dans les applications.

V.2.3. Utilisation de la fonction puissance exponentielle

Dans le cadre de l'algorithme du maximum de vraisemblance, il est possible, toujours pour s'accommoder de la présence de valeurs aberrantes dans les données, d'utiliser une famille de fonction distribution de puissance exponentielle présentée par McMichael (1990).

Dans le cas de présence de valeurs aberrantes dans le bruit de structure blanc, il est possible de modéliser ce bruit par une fonction densité de probabilité puissance exponentielle :

$$\mathcal{P}(\varepsilon) = \frac{q \exp(-|\varepsilon/a|^q)}{2a\Gamma(1/q)} \qquad (V.17)$$

où a est un paramètre de dispersion positif, et où Γ est la fonction gamma, définie sur l'intervalle $]0, +\infty[$ par l'intégrale :

$$\Gamma(1/q) = \int_0^{+\infty} x^{(1/q)-1} e^x dx \qquad (V.18)$$

La fonction (V.17) peut fournir différents degrés de contamination par des valeurs aberrantes par le choix du paramètre q $(1 < q \leq 2)$. q est donc un paramètre de dimensionnement affectant l'aplatissement de la distribution qui peut aller d'une distribution normale possédant une variance égale à $1/\sqrt{2}$ quand q = 2 à une distribution Laplacienne lorsque q = 1. La figure V.3 présente cette famille de distribution avec a = 1 et q prenant 3 valeurs comprises entre 1 et 2.

L'utilisation d'une telle famille de fonction densité de probabilité dans un algorithme du maximum de vraisemblance conduit à la minimisation d'un critère Lp. Une telle fonction coût s'appelle fonction erreur de Minkowski-q (Bishop, 1995) :

$$L_q(\varepsilon) = (|\varepsilon|)^q \qquad (V.19a)$$

Cependant, un tel critère conduit à une fonction coût par observation dont la dérivée seconde n'est pas bornée pour $\varepsilon(k) = 0$ lorsque q = 1. Une section quadratique autour de l'origine est donc introduite :

$$L_q(\varepsilon) = \begin{cases} (|\varepsilon| - \rho\hat{\sigma}(1 - q/2))^q & |\varepsilon| \geq \rho\hat{\sigma} \\ \varepsilon^2 (q/2)^q (\rho\hat{\sigma})^{q-2} & |\varepsilon| < \rho\hat{\sigma} \end{cases} \qquad (V.19b)$$

où la continuité de la fonction coût et de sa dérivée et l'existence de sa dérivée seconde est assurée et où $\hat{\sigma}$ est l'estimation de l'écart type de l'erreur de prédiction, ρ étant un scalaire compris entre : $1 \leq \rho \leq 1.8$. Nous pouvons constater que pour $1 < q \leq 2$, cette fonction est continûment dérivable deux fois.

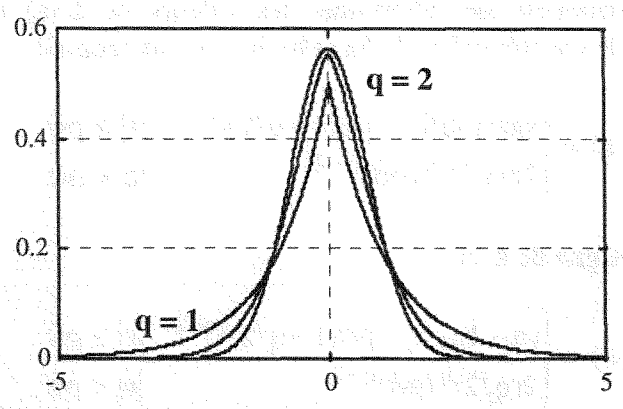


Fig.V.3. Fonction densité de probabilité puissance exponentielle pour q = 1; 1,5; 2 avec a = 1.

L'estimation $\hat{\sigma}$ de l'écart-type de l'erreur de prédiction ne doit pas être perturbée par la présence de valeurs aberrantes. Aussi, nous avons utilisé un estimateur robuste. Dans le cas itératif présenté par Bloch *et al.* (1996), nous sélectionnons classiquement $\hat{\sigma}$ par (Ljung, 1987) :

$$\hat{\sigma} = \frac{MAD}{0.7} \tag{V.20}$$

où MAD représente la médiane de $\left\{\left|\varepsilon(k) - \tilde{\varepsilon}\right|\right\}$ avec $\tilde{\varepsilon}$ la médiane de $\varepsilon(k)$.

Dans le cas d'une identification récursive présentée par Thomas et Bloch (1996), nous avons utilisé la méthode d'estimation de la variance proposée par Puthenpura et Sinha (1990) et déjà utilisée en (V.13a) :

$$\begin{aligned} &\hat{\sigma}^2(k) = \hat{\sigma}^2(k-1) + \frac{1}{k-\tau}(\varepsilon^2(k) - \hat{\sigma}^2(k-1)) \quad \text{pour } |\varepsilon(k)| \le \rho\hat{\sigma}(k-1) \\ &\hat{\sigma}^2(k) = \hat{\sigma}^2(k-1) \qquad\qquad\qquad\qquad\qquad\qquad \text{autrement} \end{aligned} \tag{V.21}$$

en prenant τ=0, pour k=1 et τ=τ+1 lorsque $|\varepsilon(k)| > \rho\hat{\sigma}(k-1)$.

La seule difficulté résultante se trouve dans le choix de q. En effet, un choix de q égal à 2 fait correspondre la fonction coût (V.19) à :

$$L_2(\varepsilon(k,\theta)) = \varepsilon^2(k,\theta) \tag{V.22}$$

qui n'aboutit pas à un algorithme robuste.

Au contraire, lorsque q se rapproche de 1, l'influence des valeurs aberrantes diminue. La stratégie la plus sûre pour obtenir un bon choix de q est d'utiliser une méthode du minimax ce qui augmente fortement le temps de calcul.

Ces choix nous permettent de déterminer les valeurs de L'(ε) et de L"(ε) dans les algorithmes (V.3) dans le cas itératif et (V.5a et b) dans le cas récursif :

$$L'_q(\varepsilon) = \begin{cases} \mathrm{sgn}(\varepsilon) q(|\varepsilon| - \rho\hat{\sigma}(1-q/2))^{q-1} & |\varepsilon| \geq \rho\hat{\sigma} \\ 2\varepsilon(q/2)^q(\rho\hat{\sigma})^{q-2} & |\varepsilon| < \rho\hat{\sigma} \end{cases} \qquad \text{(V.23a)}$$

où sgn(ε) représente le signe de ε, et :

$$L''_q(\varepsilon) = \begin{cases} (q-1)q(|\varepsilon| - \rho\hat{\sigma}(1-q/2))^{q-2} & |\varepsilon| \geq \rho\hat{\sigma} \\ 2(q/2)^q(\rho\hat{\sigma})^{q-2} & |\varepsilon| < \rho\hat{\sigma} \end{cases} \qquad \text{(V.23b)}$$

La fonction d'influence L'(ε) est présentée à la figure suivante pour des valeurs de q variant de 1 à 2 :

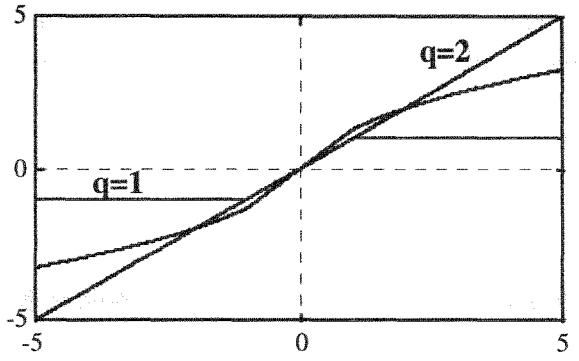


Fig.V.4. Fonction d'influence L'(ε)
pour q = 1; 1,5; 2.

Cet algorithme sera appelé Lp estimator Neural Networks (LpNN) pour la méthode itérative et Recursive Lp estimator Neural Networks (RLpNN) pour la méthode récursive.

V.3. Fonction d'influence et robustesse

V.3.1. Principe de la fonction d'influence

Nous avons vu dans le chapitre précédent que le principe du maximum de vraisemblance repose sur le choix d'une fonction densité de probabilité permettant d'expliquer la présence de valeurs aberrantes dans les données. Le principe de la fonction d'influence correspond plutôt à la question : comment limiter l'influence de valeurs aberrantes, potentiellement présentes dans les données, sur le résultat de l'identification ? Pour ce faire, nous allons étudier la raison pour laquelle un faible nombre de valeurs aberrantes peut avoir un impact important sur le résultat de l'identification.

Rappelons tout d'abord que l'estimation des paramètres d'un modèle revient à minimiser une fonction coût de la forme :

$$V_n(\theta) = \frac{1}{n}\sum_{k=1}^{n} L(\varepsilon(k,\theta)) \qquad (V.24)$$

où $L(\varepsilon(k,\theta))$ est une fonction de l'erreur à l'instant k, définie par l'équation (V.1), symétrique et continue. La dérivée de ce critère par rapport aux poids θ du réseau peut alors s'écrire :

$$\frac{\partial V_n(\theta)}{\partial \theta} = \frac{1}{n}\sum_{k=1}^{n} L'(\varepsilon(k,\theta))\frac{\partial \varepsilon(k,\theta)}{\partial \theta} \qquad (V.25)$$

qui est égale à 0 au minimum, et où :

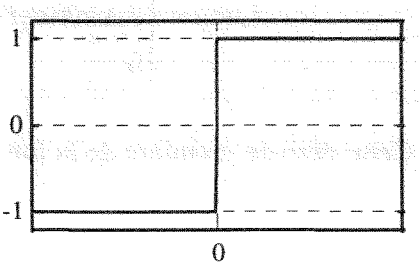
$$L'(\varepsilon(k,\theta)) = \frac{\partial L(\varepsilon(k,\theta))}{\partial \varepsilon(k,\theta)} \qquad (V.26)$$

est la fonction d'influence. Si la dérivée du critère (V.25) est interprétée comme la somme pondérée de $\frac{\partial \varepsilon(k,\theta)}{\partial \theta}$, $L'(\varepsilon(k,\theta))$ représente le poids ou influence du point k, d'où le nom de fonction d'influence. Il est remarquable que pour le critère quadratique standard, cette fonction d'influence s'écrit simplement :

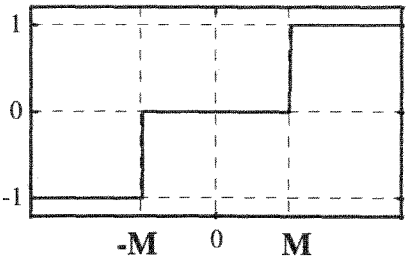
$$L'(\varepsilon(k,\theta)) = \varepsilon(k,\theta) \qquad (V.27)$$

Plus l'erreur de prédiction est importante, plus le poids dans l'équation (V.25) est grand et plus cette erreur aura d'influence sur l'estimation des paramètres du modèle. Pour limiter l'influence des valeurs aberrantes, une méthode simple consiste à sélectionner comme fonction d'influence (la dérivée de la fonction coût) une fonction mathématique approximativement quadratique pour les faibles valeurs de résidus mais fournissant une influence limitée des résidus dans le cas de fortes valeurs. De nombreuses fonctions de ce type ont été proposées par des statisticiens (Ershov, 1978; Poljak et Tsypkin, 1980), dont certaines dérivées des notions de la "trimmed mean" ou de la "winzorized mean" (Barnett et Lewis, 1984). Les figures et équations suivantes présentent succinctement plusieurs choix de fonctions d'influence :

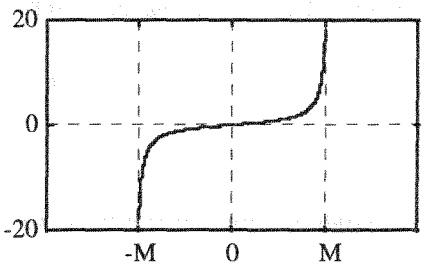
$$L'(\varepsilon(k,\theta)) = \mathrm{sgn}(\varepsilon(k,\theta))$$



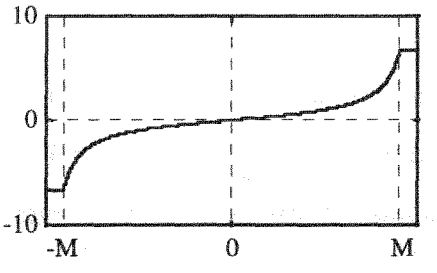
$$L'(\varepsilon(k,\theta)) = \begin{cases} 0 & |\varepsilon| < M \\ \mathrm{sgn}(\varepsilon) & |\varepsilon| \geq M \end{cases}$$



$$L'(\varepsilon(k,\theta)) = \tan(b\,\varepsilon)$$



$$L'(\varepsilon(k,\theta)) = \begin{cases} \tan(b_1\varepsilon) & |\varepsilon| < M \\ b_2\ \mathrm{sgn}(\varepsilon) & |\varepsilon| \geq M \end{cases}$$



Cette stratégie est la plus généralement employée dans le cadre de l'identification de systèmes linéaires. C'est aussi ce type de critères qu'ont proposé Chen et Jain (1991) ou encore Cichocki et Unbehauen (1993). Certaines fonctions d'influences sont obtenues par des hypothèses a priori sur le bruit (Puthenpura et Sinha, 1990; Liano, 1996).

La fonction d'influence proposée par Chen et Jain (1991) a pour objet, comme nous l'avons précédemment signalé, de se comporter comme le critère quadratique pour les résidus de faible amplitude, et d'agir comme un limiteur d'influence pour les résidus d'amplitude moyenne, mais aussi de supprimer toute influence des forts résidus sur l'estimation des paramètres du réseau comme le montre son équation :

$$L'(\varepsilon) = \begin{cases} \varepsilon & |\varepsilon| < M_1 \\ c_1 \tanh(c_2(M_2 - |\varepsilon|))\,\mathrm{sgn}(\varepsilon) & M_1 \leq |\varepsilon| \leq M_2 \\ 0 & |\varepsilon| > M_2 \end{cases} \qquad \text{(V.28)}$$

Cette dérivée première de la fonction coût est présentée par la figure (V.5).

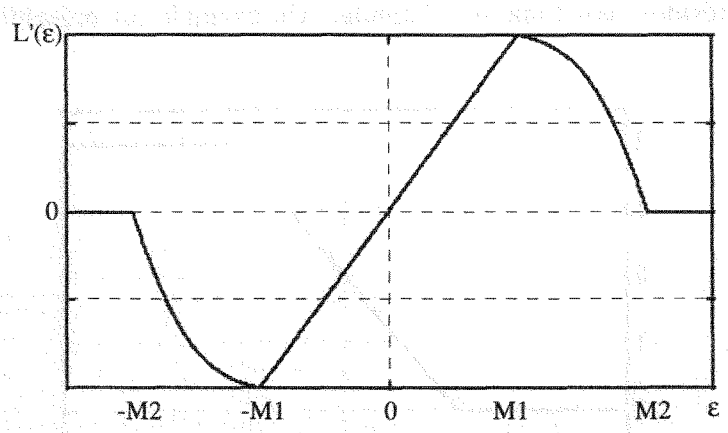


Fig. V.5. Dérivée de la fonction critère

Les bornes M1 et M2, qui déterminent la forme de la fonction d'influence, peuvent être déterminées de diverses manières. Chen et Jain (1991) proposent une méthode utilisant le pourcentage de valeurs aberrantes présentes dans les données d'apprentissage. Si l'on suppose que ce pourcentage est de Q%, la borne M1(k) est calculée comme étant la Qième plus grande amplitude du résidu à l'instant k. La borne M2(k) est alors choisie comme étant égale à deux fois M1(k). La valeur Q étant déterminée de manière heuristique.

V.3.2. Méthode par fenêtrage des résidus.

Contrairement à Chen et Jain, nous avons préféré choisir une fonction coût de forme plus simple et se rapprochant de la notion de ''winzorizing''. Aussi, nous avons choisi comme fonction d'influence la fonction suivante :

$$L'(\varepsilon) = \begin{cases} \varepsilon & |\varepsilon| < M \\ M & \varepsilon \geq M \\ -M & \varepsilon \leq M \end{cases} \qquad \text{(V.29a)}$$

M étant une borne préassignée.

On obtient alors pour sa primitive correspondant à la fonction coût l'équation suivante :

$$L(\varepsilon) = \begin{cases} \frac{1}{2}\varepsilon^2 & |\varepsilon| < M \\ M\varepsilon - \frac{1}{2}M^2 & \varepsilon \geq M \\ M\varepsilon + \frac{1}{2}M^2 & \varepsilon \leq M \end{cases} \qquad \text{(V.29b)}$$

La fonction d'influence présentée à la figure suivante se comporte toujours de manière similaire au critère quadratique pour les faibles valeurs de résidus, mais limite l'influence des

fortes valeurs de résidus sans toutefois l'annuler. Un exemple est présentée à la figure (V.6) pour M=2.

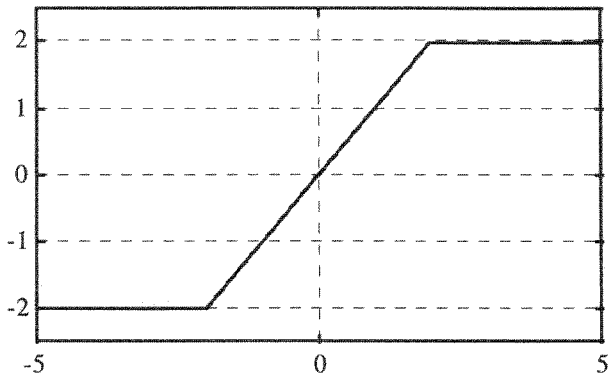


Fig. V.6. Représentation de la dérivée de la fonction coût.

La fonction coût que nous avons choisie est celle recommandée par Ljung (1987) et qui est déjà utilisée par l'algorithme PEM (Prediction Error Method) de Matlab (Ljung, 1992). Ce choix consiste à prendre pour borne M :

$$M = \rho\hat{\sigma} \tag{V.30}$$

où $\hat{\sigma}$ est l'estimation de l'écart-type de l'erreur de prédiction et ρ un coefficient tel que : $1 \leq \rho \leq 1.8$. ρ est en fait un coefficient de sensibilité aux valeurs aberrantes. Il faut noter que l'estimation $\hat{\sigma}$ de l'écart-type de l'erreur de prédiction ne doit pas être perturbée par la présence de valeurs aberrantes. Aussi, nous avons utilisé un estimateur robuste. Dans le cas itératif présenté par Bloch *et al.* (1996), nous sélectionnons classiquement $\hat{\sigma}$ par l'équation (V.20). Dans le cas d'une identification récursive nous allons utiliser la méthode d'estimation de la variance présentée par l'équation (V.21).

Le problème principal entraîné par ce critère est qu'il n'est pas deux fois continûment dérivable ce qui implique que nous ne pouvons pas obtenir une fonction L"(ε(k)) continue non nulle ce qui peut impliquer des problèmes de convergence de l'algorithme. Aussi, pour remédier à cet état de fait, nous avons apporté la même modification à l'algorithme que celle apportée dans PEM par Ljung (1992) en remplaçant L"(ε(k)) par la fonction constante 1. Ces choix nous permettent de déterminer les valeurs de L'(ε) et de L"(ε) dans les algorithmes (V.3) dans le cas itératif et (V.5a et b) dans le cas récursif :

$$L'(\varepsilon) = \begin{cases} \varepsilon & |\varepsilon| < \rho\hat{\sigma} \\ \rho\hat{\sigma} & \varepsilon \geq \rho\hat{\sigma} \\ -\rho\hat{\sigma} & \varepsilon \leq \rho\hat{\sigma} \end{cases} \tag{V.31a}$$

et :

$$L''(\varepsilon)=1 \tag{V.31b}$$

Cet algorithme sera appelé Influence Neural Networks (INN) pour la méthode itérative et Recursive Influence Neural Networks (RINN) pour la méthode récursive.

V.5. Application à un exemple de simulation

Le système que nous allons employer est un système non-linéaire NARX introduit par Chen *et al.* (1990) et déjà utilisé dans les deux chapitres précédents :

$$\begin{aligned} y(k) = &(0.8 - 0.5\, e^{-y^2(k-1)})\ y(k-1) \\ &-(0.3 + 0.9\, e^{-y^2(k-1)})\ y(k-2) \\ &+u(k-1) + 0.2\, u(k-2) + 0.1\, u(k-1)\, u(k-2) + e(k) \end{aligned} \qquad (V.32)$$

où u(k) est le signal d'entrée du système à l'instant k, choisi comme une séquence aléatoire uniformément distribuée de moyenne 0 et de variance 1. y(k) est la sortie du système et e(k) est un bruit gaussien tel que $e \sim N(0,\sigma_1^2)$ lorsqu'il n'y a pas de valeur aberrante présente dans le signal. Pour montrer l'influence des valeurs aberrantes, la figure V.7 présente, pour $\sigma_1^2 = 0.04$, la sortie du système contaminée par 50 valeurs aberrantes aléatoirement reportées dans le signal. Ces valeurs aberrantes sont simplement simulées en multipliant la valeur originale du bruit par un facteur multiplicatif f de 15. Ce type de bruit peut être décrit par le modèle introduit par Huber (1964) et présenté à l'équation (V.10).

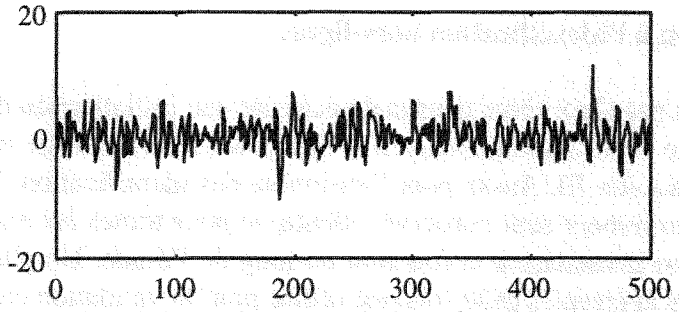


Fig V.7. Signal de sortie contaminé par les valeurs aberrantes.

La figure V.8 présente la différence entre la figure V.7 et la simulation obtenue dans les mêmes conditions mais sans valeur aberrante. Cette figure a pour objet de présenter l'impact important des valeurs aberrantes sur la sortie du procédé :

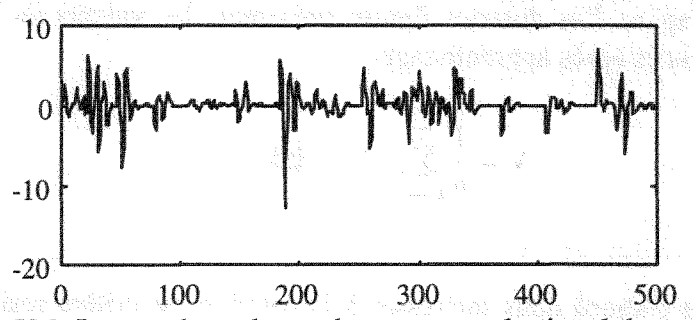


Fig V.8. Impact des valeurs aberrantes sur le signal de sortie.

La figure V.9 présente, elle, la différence entre la sortie du système, obtenue sans valeur aberrante et la simulation obtenue sans bruit ni valeur aberrante. Cette figure a pour objet de

présenter l'amplitude du bruit filtré par le système, amplitude nettement plus petite que pour la figure précédente :

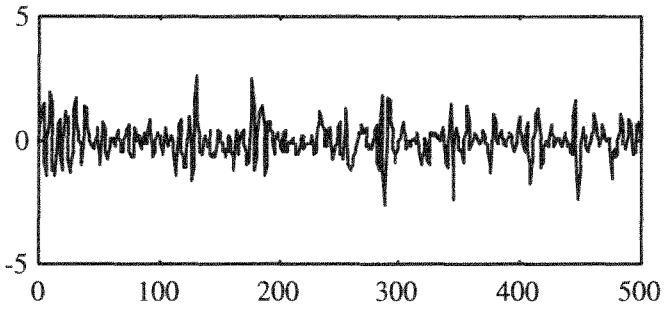


Fig V.9. Bruit filtré par le système.

Pour l'ensemble des expériences qui vont suivre, le prédicteur neuronal possédera la même structure : 4 (+ le biais) neurones sur la couche d'entrée, 3 (+ le biais) neurones sur la couche cachée et un seul sur la couche de sortie. De plus, le vecteur d'information $\phi(k) = [y(k-1)\ y(k-2)\ u(k-1)\ u(k-2)]^T$ est conservé identique pour l'ensemble des expériences, que ce soit dans le cadre de l'identification hors-ligne ou en-ligne.

V.5.1. Application à l'identification hors-ligne.

Une identification par algorithme neuronal nécessite une initialisation des poids effectuée à l'aide de l'algorithme traditionel de Nguyen et Widrow (1990) modifié par Demuth et Beale (1994) présenté au chapitre III. Aussi, pour l'ensemble des identifications hors ligne, les poids initiaux choisis aléatoirement sont conservés identique pour toutes les expériences afin de ne faire varier qu'un seul paramètre à la fois tout au long de l'étude. De plus, un second jeu de données (sans valeurs aberrantes cette fois) est utilisé pour la validation croisée. Ce second jeu de données est appelé jeu de validation et a des caractéristiques similaires au jeu d'identification concernant la forme des entrées, la distribution du bruit et sa variance mais ne possède pas de valeurs aberrantes. Les résultats des divers algorithmes sont comparés avec ceux obtenus par l'algorithme hors ligne de Levenberg-Marquard programmé dans la Toolbox Matlab Neural Networks (Demuth et Beale, 1994; Hagan et al., 1995) qui est appelé LMNN dans la suite du chapitre. Les diverses figures présentent les valeurs du critère résiduel dans les diverses expériences après apprentissage :

$$V = \frac{1}{n}\sum_{k=1}^{n}(y(k) - \hat{y}(k))^2 \qquad (V.33)$$

avec n=500.

Nous allons tout d'abord nous intéresser à l'évolution du critère résiduel en fonction du nombre de valeurs aberrantes présentes dans le signal. Aussi, pour une variance du bruit de $\sigma_1^2 = 0.04$, nous allons faire varier le nombre des valeurs aberrantes de 0 (pas de valeur aberrante) à 100. Le facteur multiplicatif f est pris égal à 15 pour l'ensemble des expériences.

La figure V.10 présente l'évolution du critère résiduel pour l'ensemble des algorithmes lors de l'identification :

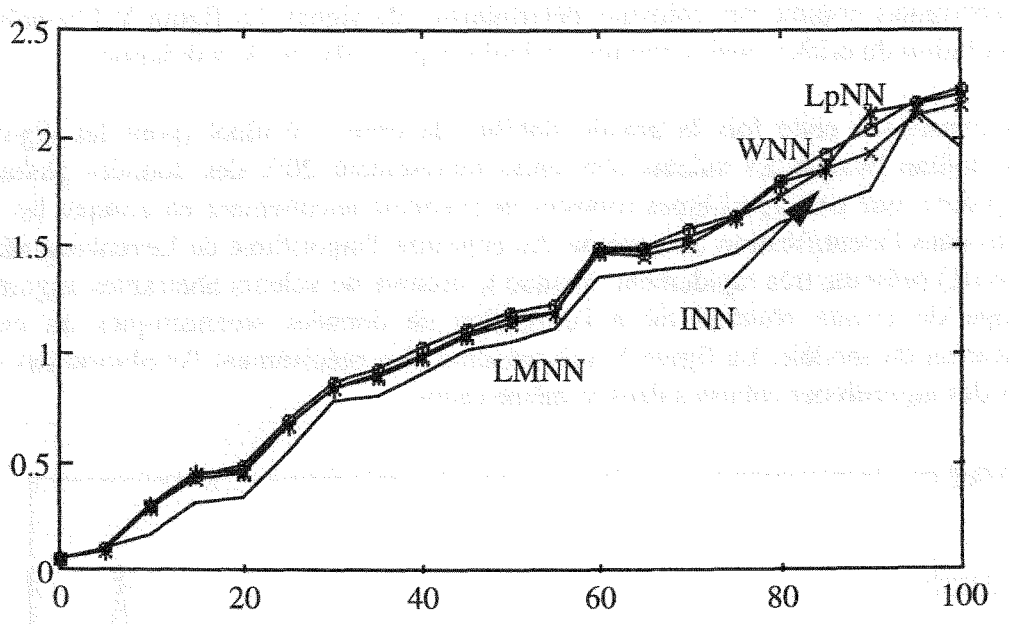


Fig V.10. Critère résiduel en fonction du nombre de valeurs aberrantes.
(jeu d'identification)

Nous pouvons ici constater l'accroissement constant de ce critère pour l'ensemble des algorithmes dû tout simplement à l'importance croissante de la partie stochastique dans le signal.

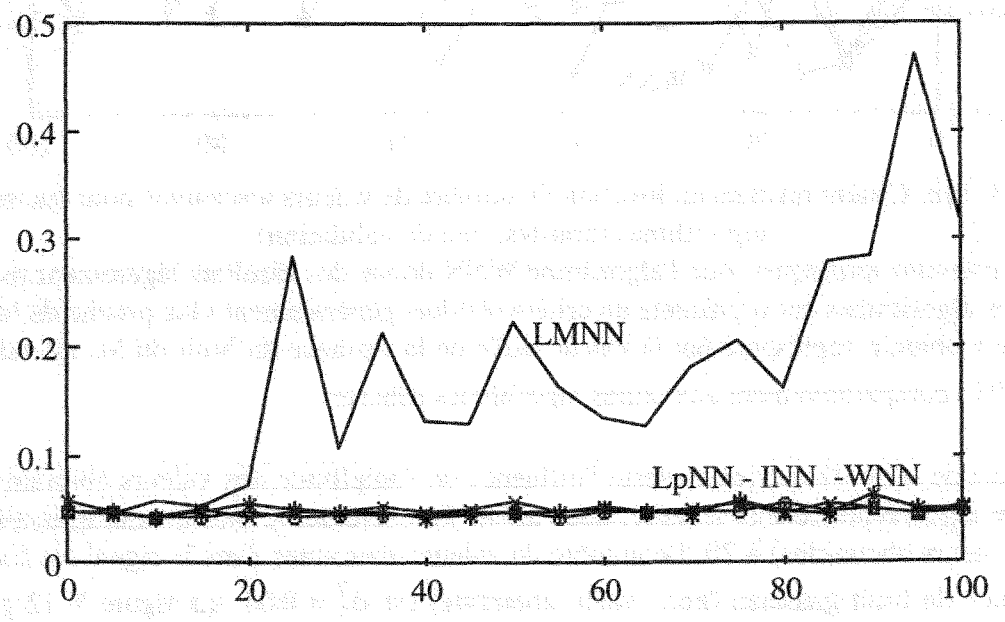


Fig V.11a. Critère résiduel en fonction du nombre de valeurs aberrantes.
(jeu de validation)

Il est à noter cependant que l'algorithme traditionnel de Levenberg-Marquard présente généralement un critère résiduel plus faible que les autres méthodes dans le cas de présence de valeurs aberrantes. Ceci s'explique par le fait que cet algorithme a tendance à comprendre ces valeurs aberrantes comme des éléments déterministes du signal. La figure V.11a présente la même évolution du critère, mais cette fois, calculée à partir du jeu de validation.

Nous constatons cette fois la grande stabilité du critère résiduel (pour les algorithmes robustes) même lorsque les valeurs aberrantes représentent 20% des données globales. Ce résultat prouve que ces algorithmes robustes ne prennent aucunement en compte les valeurs aberrantes dans l'identification du système. Au contraire, l'algorithme de Levenberg-Marquard (non robuste) présente très rapidement, lorsque le nombre de valeurs aberrantes augmente, un décrochage du critère résiduel dû à l'utilisation de données stochastiques au cours de l'identification du modèle. La figure V.11b présente plus précisément l'évolution des critères résiduels des algorithmes robustes dans le même cadre :

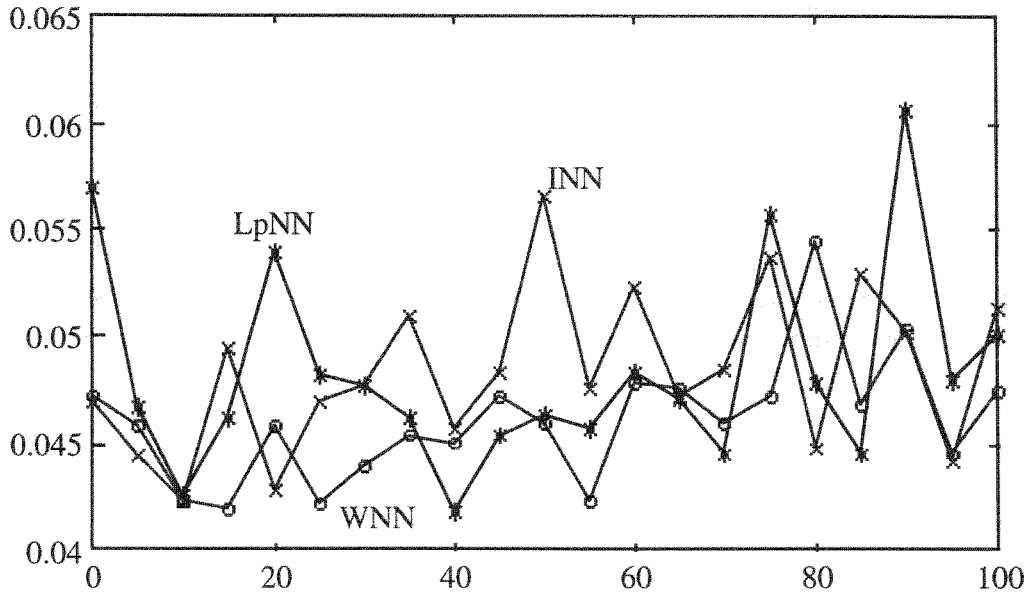


Fig V.11b. Critère résiduel en fonction du nombre de valeurs aberrantes pour les trois algorithmes robustes. (jeu de validation)

Nous pouvons remarquer, que l'algorithme WNN donne des résultats légèrement meilleurs aux autres algorithmes car il présente un critère résiduel généralement plus proche de la valeur théorique à obtenir, représenté par la valeur réelle de la variance du bruit du jeu de validation ($\sigma_1^2 = 0.04$) comparativement aux autres algorithmes robustes.

La seconde série d'expériences teste l'influence de l'amplitude des valeurs aberrantes pour les quatre algorithmes. Dans ce cadre, nous allons faire évoluer le facteur multiplicatif f de 1 (pas de valeurs aberrantes) à 20. Le nombre de valeurs aberrantes dans le signal est fixé à 50. La variance du bruit gaussien (hors valeur aberrante) est $\sigma_1^2 = 0.04$. La figure V.12 présente l'évolution du critère résiduel en fonction du facteur multiplicatif pour le jeu d'identification.

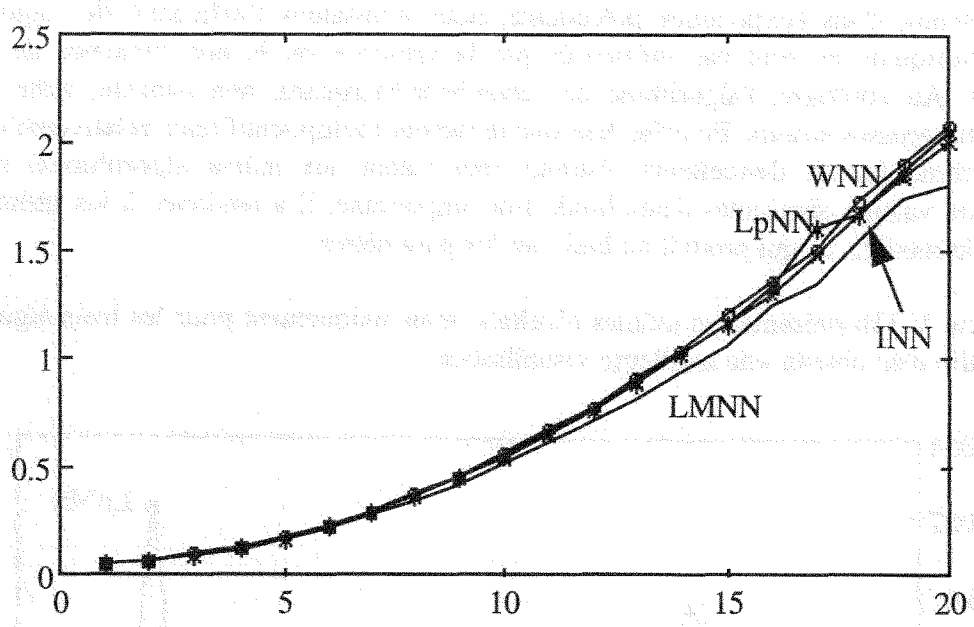


Fig V.12. Critère résiduel en fonction du facteur multiplicatif f.
(jeu d'identification)

Nous constatons ici des résultats tout à fait similaires à l'expérience précédente. Nous retrouvons en effet l'accroissement constant du critère résiduel (suivant ici sensiblement la forme d'une parabole) en fonction du facteur multiplicatif f. De même que pour l'expérience précédente, l'algorithme LMNN présente une évolution du critère résiduel moins forte que pour les autres algorithmes et semble une fois de plus tenter d'apprendre le bruit lors de la construction du modèle. La figure V.13a présente la même évolution du critère mais calculée cette fois à partir du jeu de validation.

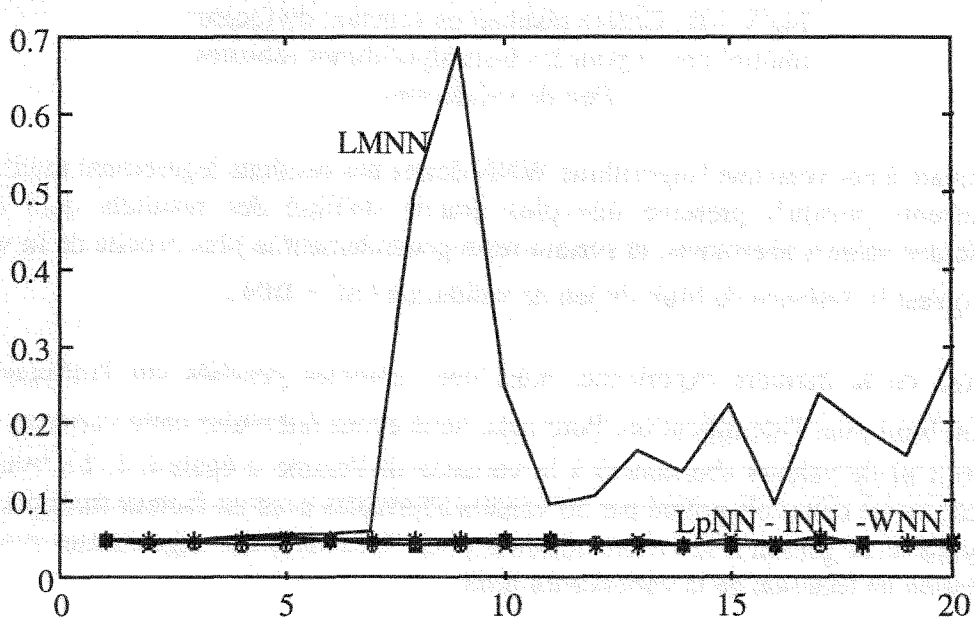


Fig V.13a. Critère résiduel en fonction du facteur multiplicatif f.
(jeu de validation)

Tout comme dans l'expérience précédente, nous constatons l'efficacité des algorithmes robustes puisqu'ils ne sont pas influencés par la présence ou la non présence de valeurs aberrantes. Au contraire, l'algorithme de Levenberg-Marquard, non robuste, reste lui très sensible aux grosses erreurs. En effet, tant que le facteur multiplicatif reste relativement faible, cet algorithme fournit d'excellents résultats (équivalent aux autres algorithmes) mais en présence de valeurs aberrantes d'amplitude trop importante, il a tendance à les utiliser pour identifier le modèle, ce qui produit un biais sur les paramètres.

La figure V.13b présente ces mêmes résultats, mais uniquement pour les trois algorithmes robustes afin d'en obtenir une meilleure visualisation :

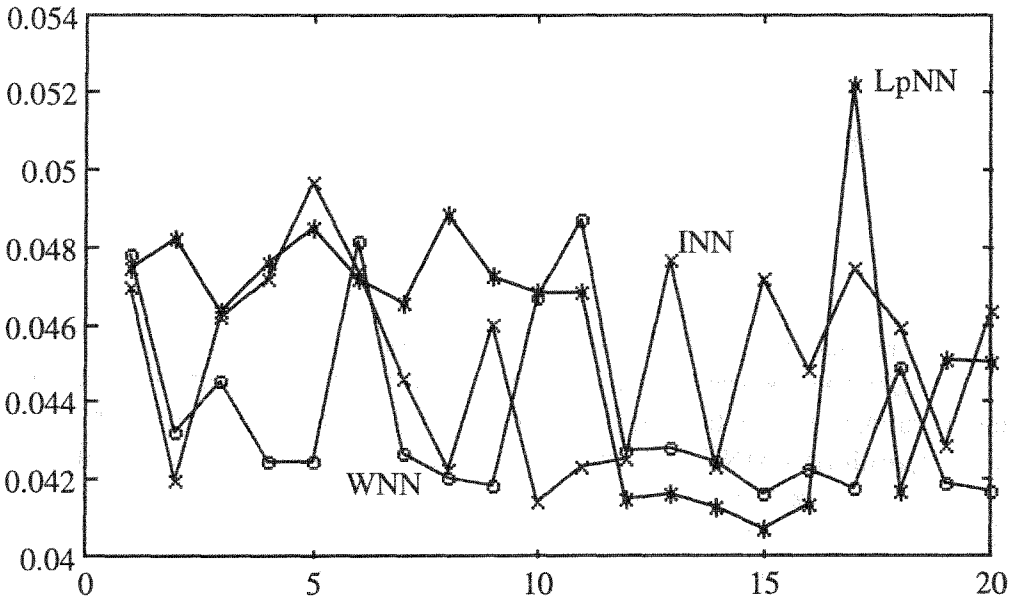


Fig V.13b. Critère résiduel en fonction du facteur multiplicatif f pour les trois algorithmes robustes. (jeu de validation)

On constate à nouveau que l'algorithme WNN donne des résultats légèrement meilleurs que ses concurrents puisqu'il présente une plus grande stabilité des résultats quel que soit l'amplitude des valeurs aberrantes, et surtout reste généralement le plus proche de la valeur de référence qu'est la variance du bruit du jeu de validation ($\sigma_1^2 = 0.04$).

Au cours de la dernière expérience, nous nous sommes penchés sur l'influence de la variance du bruit pour l'identification. Pour cela, nous avons fait varier cette variance σ_1^2 de 0 (pas de bruit ni de valeurs aberrantes) à la variance de l'entrée u égale à 1. Le bruit du jeu d'identification est cette fois pollué par 50 valeurs aberrantes avec un facteur multiplicatif f de 15. La figure V.14 présente le critère résiduel pour l'ensemble des algorithmes pour le jeu d'identification en fonction de la variance du bruit.

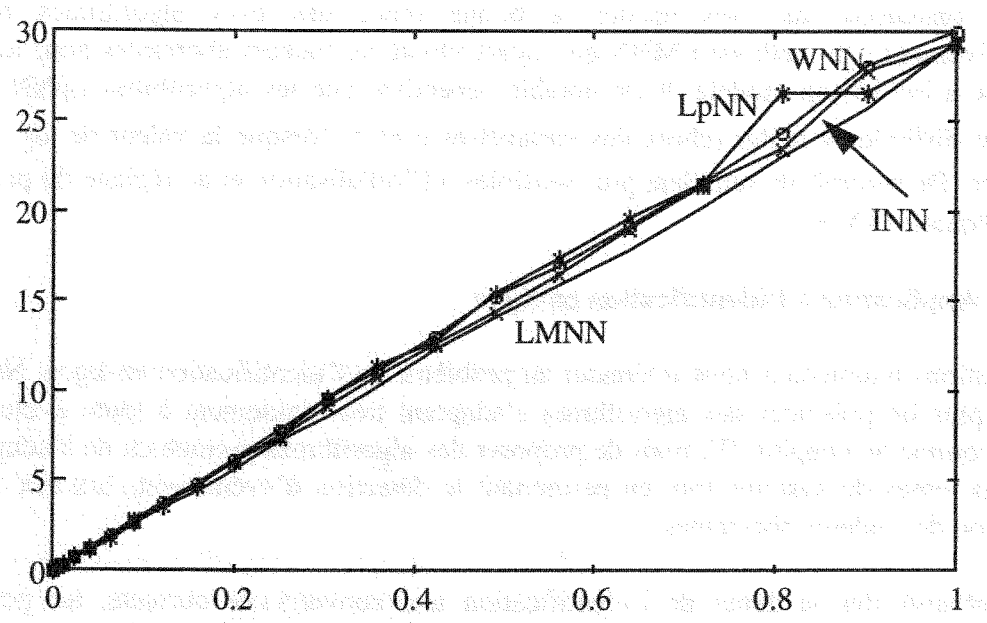


Fig V.14. Critère résiduel en fonction de la variance du bruit.
(jeu d'identification)

De même que dans les deux premières expériences, nous constatons que l'évolution du critère résiduel suit l'importance croissante de la partie stochastique dans le signal. Une fois de plus, l'algorithme de Levenberg-Marquard semble apprendre les valeurs aberrantes en même temps que le modèle. La figure V.15 présente la même évolution du critère, mais calculée cette fois avec le jeu de validation.

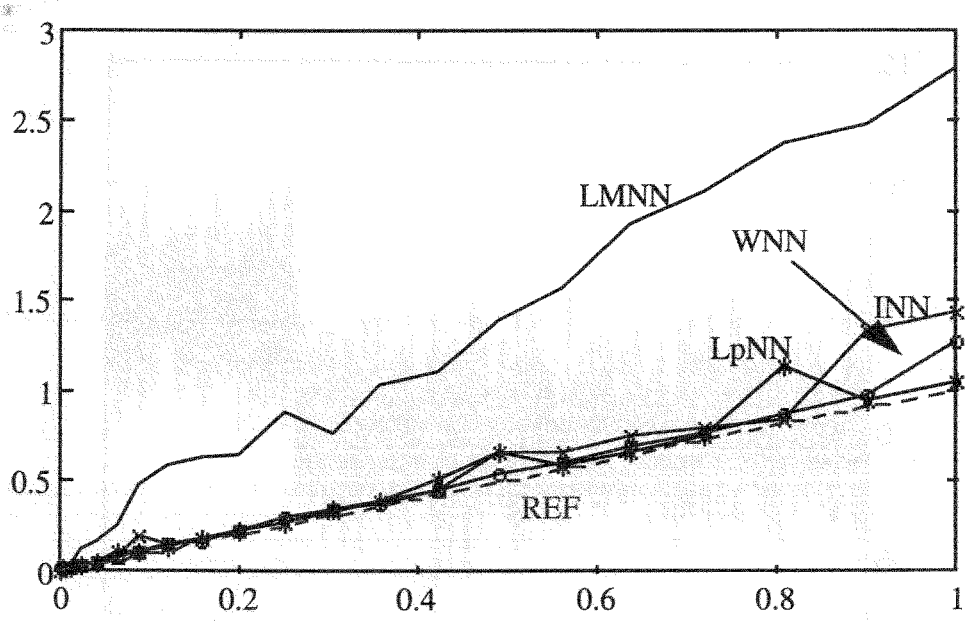


Fig V.15. Critère résiduel en fonction de la variance du bruit.
(jeu de validation)

Nous constatons une fois encore la bonne tenue des trois algorithmes robustes comparativement à la méthode LMNN qui, ayant utilisé les valeurs aberrantes pour identifier le système a biaisé son modèle. Il est notable cependant que les algorithmes LpNN et INN sont assez difficiles à régler (choix des paramètres q et ρ) lorsque la valeur de σ_1^2 devient importante. De surcroît ils semblent plus sensibles a l'initialisation et au réglage du paramètre α dans l'équation (V.3).

V.5.2. Application à l'identification en-ligne

Nous allons maintenant nous intéresser au problème de l'identification en-ligne. Notre but ici n'est plus de présenter des algorithmes s'adaptant très rapidement à toute évolution du système comme au chapitre IV, mais de proposer des algorithmes permettant de s'adapter aux évolutions lentes du système tout en permettant la détection d'événements brutaux tels que des biais ou des valeurs aberrantes.

Pour obtenir dès le début de l'identification une convergence correcte, les poids son initialisés par le résultat d'une identification hors-ligne précédemment effectuée sans valeur aberrante. Le jeu de données sur lequel nous allons effectuer l'identification récursive du modèle est constitué de 2000 valeurs de sortie correspondant à 2000 valeurs d'entrée choisies comme une séquence de valeurs aléatoires uniformément distribuées entre -1 et 1. Afin de présenter la robustesse des critères présentés, nous avons introduit un certain nombre de défauts dans les données. Ces défauts sont constitués de six valeurs aberrantes aléatoirement distribuées sur le bruit de variance 0,16 et d'un biais additif d'amplitude 5, introduit sur la sortie à partir de l'instant 1500, jusqu'à l'instant 2000, et simulant un défaut de capteur. La figure V.16 présente la sortie du système pollué par les défauts.

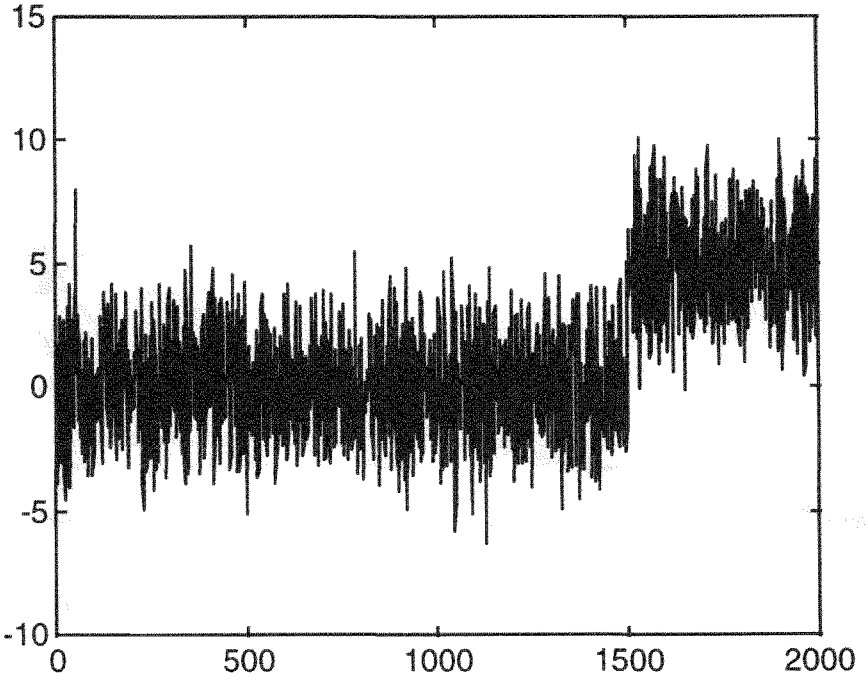


Fig V.16. Sortie polluée par les défauts.

L'impact de ces défauts est présenté à la figure V.17. Cet impact est calculé en effectuant la différence entre la sortie obtenue, polluée par les défauts, et celle que l'on aurait obtenue en absence de défauts. Cette figure permet notamment de constater la dynamique du système qui se trouve appliquée à l'ensemble des valeurs aberrantes.

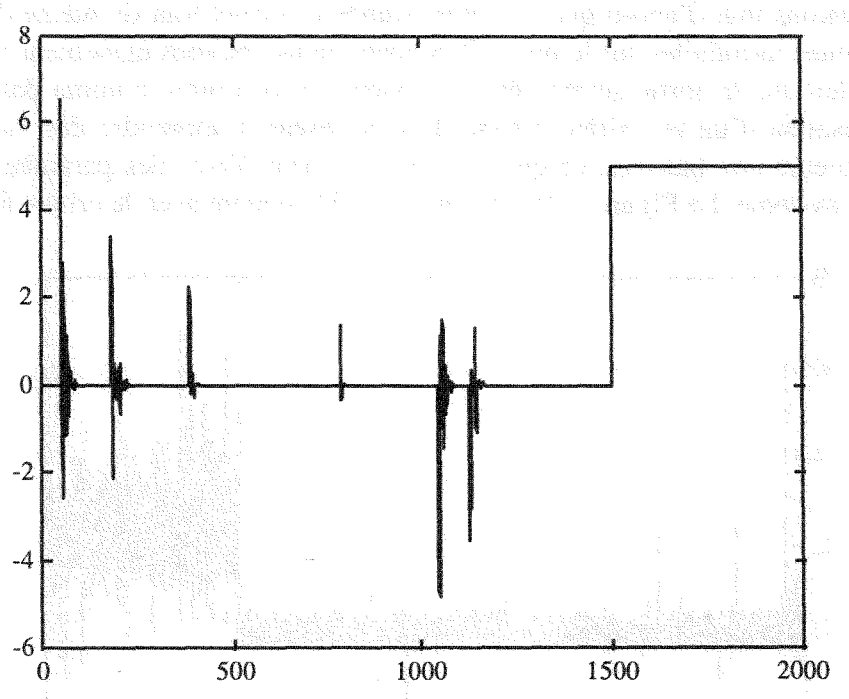


Fig V.17. Impact des défauts sur la sortie.

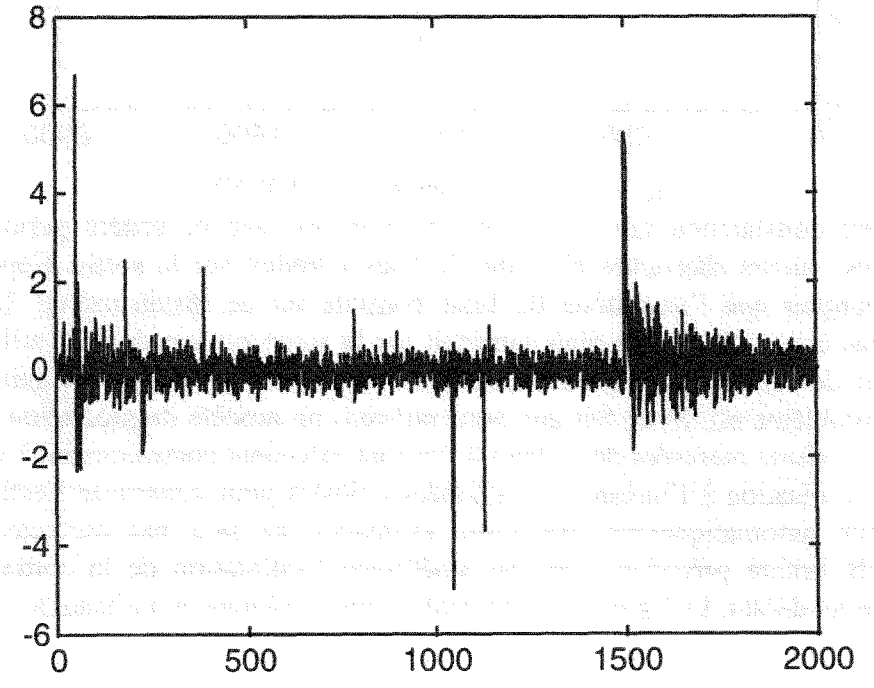


Fig V.18. Résidu obtenu avec le critère standard RPE.

Nous allons maintenant présenter les résidus $(y-\hat{y})$ obtenus pour l'ensemble du jeu de données à l'aide de chacun des critères proposés et également avec le critère quadratique standard que nous appellerons RPE par la suite. La figure V.18 présente le résidu obtenu avec le critère standard RPE.

Nous constatons tout d'abord que ce critère standard permet tout de même de détecter les valeurs aberrantes introduites sur le bruit. Cependant, nous pouvons également remarquer que le biais introduit sur la sortie semble être considéré par ce critère comme des valeurs sans défaut. L'utilisation d'un tel critère pousse donc le réseau à apprendre des valeurs biaisées comme des valeurs non biaisées, ce qui va conduire à une dérive des paramètres du modèle par rapport au système. La Figure V.19 présente le résidu obtenu avec le critère RWNN.

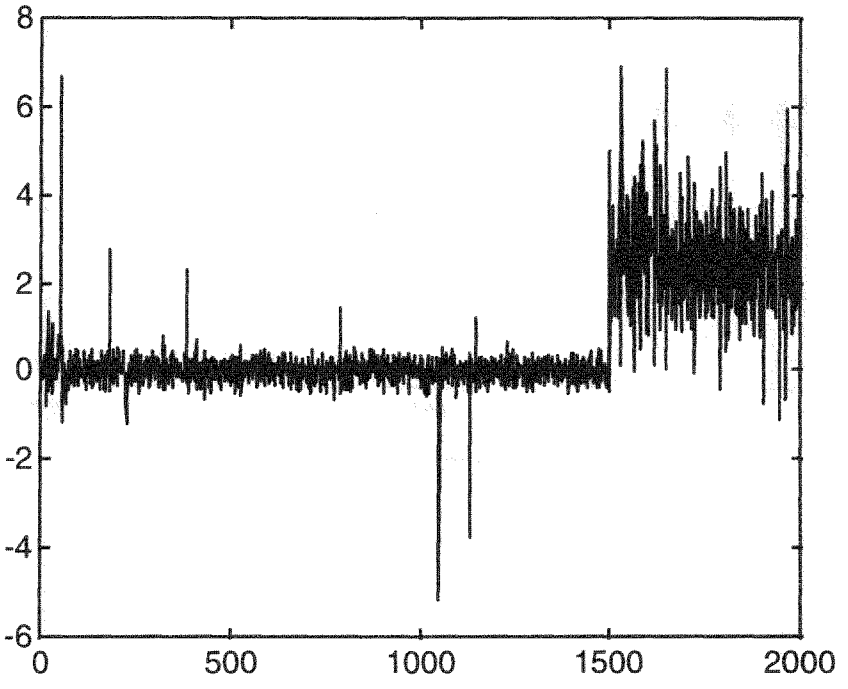


Fig V.19. Résidu obtenu avec RWNN.

La première constatation que nous pouvons faire est que ce critère permet de rejeter l'ensemble des valeurs aberrantes ainsi que le biais introduit sur la sortie. Cependant, nous pouvons remarquer que l'amplitude du biais constaté sur ce résidu est de 2,5 ce qui ne correspond pas à l'amplitude du biais introduit sur la sortie qui est de 5. L'utilisation de ce critère permet de retrouver un biais introduit sur la sortie pondéré par la gain statique du système. Ce problème est dû au fait que nous utilisons un modèle de prédiction qui utilise, à l'instant k, les valeurs mesurées des sorties à l'instant précédent contrairement à un modèle de simulation. L'utilisation à l'instant k de données polluées pour construire l'estimation de la sortie implique automatiquement que cette estimation ne sera pas correcte. Cependant, l'utilisation du critère précédent, tout en améliorant l'estimation de la sortie, favorise la détection d'un tel défaut. La figure V.20 présente le résidu obtenu en utilisant le critère RINN.

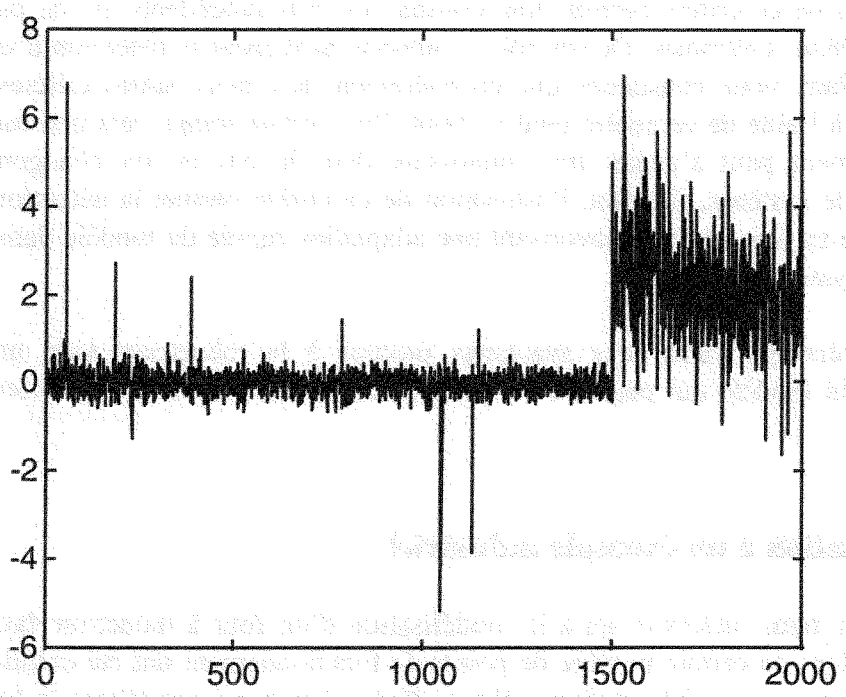


Fig V.20. Résidu obtenu avec RINN.

De même que pour les deux précédents résidus, nous constatons que ce résidu tend à reproduire les diverses valeurs aberrantes. De plus, l'utilisation du critère RINN permet, comme avec le critère RWNN, de reproduire le biais sur la sortie. Les deux critères RWNN et RINN fournissent des résultats très sensiblement semblables et sont tous les deux adaptés à ce type de problème. La figure V.21 présente le résidu obtenu avec le critère RLpNN.

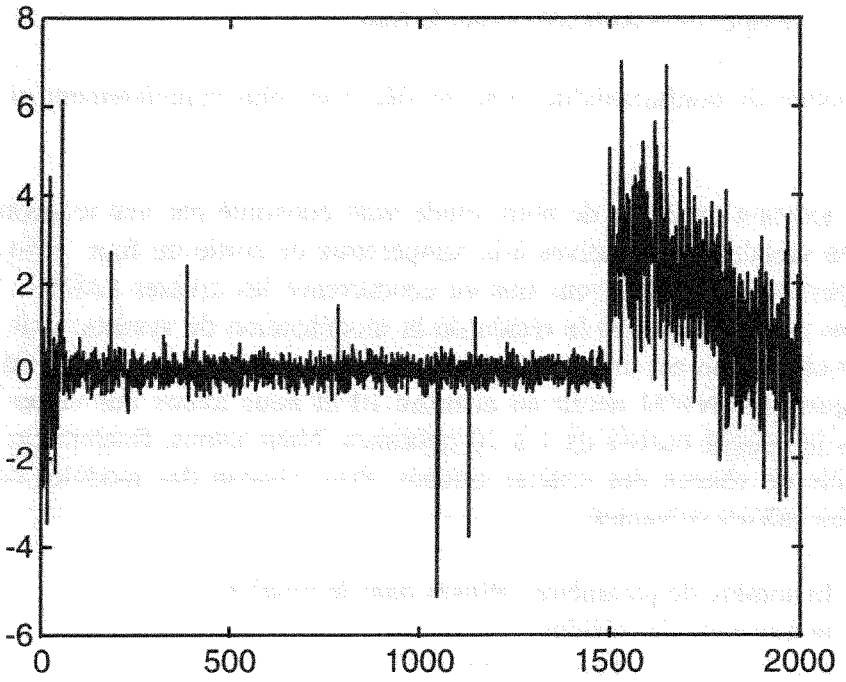


Fig V.21. Résidu obtenu avec RLpNN.

L'utilisation de ce critère permet, tout comme les trois précédents, de ne pas prendre en compte les valeurs aberrantes. De surcroît, il autorise également la détection d'un biais sur la sortie. Cependant, nous constatons que contrairement aux deux autres critères robustes, le résidu obtenu à l'aide de ce critère tend au bout d'un certain temps vers une moyenne nulle. Ce comportement peut s'avérer très intéressant dans le cas où un changement brusque intervient sur le système. En effet, l'utilisation de ce critère permet la détection d'un défaut éventuel sur le système, tout en favorisant une adaptation rapide du modèle dans le cas d'une évolution du système.

Un tel critère allie donc des avantages propres à la robustesse avec une possibilité d'adaptation du modèle qui peut encore être accrue dans le cas de l'utilisation d'un facteur d'oubli.

V.6. Application à un exemple industriel

Nous allons nous intéresser ici à la modélisation d'un four à induction dans le domaine sidérurgique dont un certain nombre de points de fonctionnement ont été extraits. Le modèle obtenu sera donc un modèle statique. Nous allons chercher à modéliser la température de sortie du four T en fonction de 7 variables explicatives :

largeur : largeur de la tôle,
épaisseur : épaisseur de la tôle,
résistivité : indice de résistivité de la tôle,
Zn : poids de zinc déposé par m2 de tôle,
Pgal : puissance appliquée au four à induction,
vitesse : vitesse de la bande de tôle,
Te : température de la tôle avant le four.

Pour des raisons de confidentialité, nous ne décrirons plus complètement ni le procédé, ni les variables.

Le modèle existant au début de notre étude était constitué par une relation linéaire liant directement les variables explicatives à la température de sortie du four. Afin de trouver un modèle plus performant, nous avons mis en concurrence les critères LMNN, WNN, INN et LpNN car nous ne savons pas si le résidu de la modélisation du système doit suivre une loi normale. Pour chacun de ces critères, nous avons utilisé 20 jeux de poids initiaux obtenus à l'aide de l'algorithme NWM décrit au chapitre III et nous avons fait varier le nombre de neurones dans la couche cachée de 1 à 10 neurones. Nous avons, finalement, sélectionné le meilleur modèle de chacun des critères utilisés. Pour chacun des modèles comparés, nous présenterons les valeurs suivantes :

#para : le nombre de paramètres utilisés dans le modèle,
Mrés. : la moyenne des résidus,
V : le critère résiduel correspondant à la variance des résidus,
max : l'erreur maximale obtenue.

Ces résultats sont rassemblés au tableau V.1.

		identification			validation		
Modèle	#para	Mrés.	V	max	Mrés.	V	max
LMNN	55	0,0002	4,451	6,331	-0,3618	8,26	15,184
WNN	64	0,0637	6,209	7,535	-0,9425	8,839	9,311
INN	46	-0,013	5,331	7,992	-0,5891	10,013	10,072
LpNN	37	0,0443	5,602	8,082	-0,1145	8,42	10,561

Tab V.1. Comparaison des différents modèles.

On constate immédiatement que le modèle obtenu avec LpNN est le plus performant en nombre de paramètres mais aussi fournit les meilleurs résultats en validation. Nous avons donc choisi ce modèle sur lequel nous allons appliquer un algorithme d'élimination des poids inutiles décrit au chapitre II (Hansen et Pedersen, 1994; Norgaard, 1995). Les résultats de ce modèle sont présentés et comparés à ceux du modèle linéaire au tableau V.2.

		identification			validation		
Modèle	#para	Mrés.	V	max	Mrés.	V	max
Linéaire	8	0	12,443	13,833	-0,7337	8,721	10,081
LpNN	31	$4{,}62.10^{-3}$	3,606	6,556	0,6679	3,944	5,867

Tab V.2. Comparaison entre le modèle sélectionné et le modèle linéaire.

Tout d'abord, l'élimination des poids inutiles dans le réseau permet non seulement de diminuer le nombre de paramètres mais aussi, améliore fortement les résultats. de plus, l'utilisation d'un modèle neuronal améliore fortement la qualité de la modélisation. Un tel modèle permet notamment de diviser par deux l'erreur maximale que l'on peut faire sur l'estimation de la température de sortie du four. Ces quelques résultats illustrent l'amélioration que peut apporter l'utilisation d'un modèle non-linéaire et, en particulier, neuronal, dans le diagnostic et le contrôle d'installations industrielles utilisant actuellement un modèle linéaire.

V.7. Conclusion.

Au cours de ce chapitre, nous nous sommes intéressés au problème posé par la présence de valeurs aberrantes dans les données. En effet, les données industrielles sont généralement polluées par des valeurs aberrantes qui peuvent conduire lors de la modélisation à des biais sur des paramètres, et donc, à des modèles de mauvaise qualité.

Ce problème bien connu et étudié pour la modélisation linéaire n'est que depuis peu étudié pour la modélisation neuronale. Dans ce chapitre, sont présentés divers critères à minimiser, robustes aux valeurs aberrantes et utilisables dans le cadre de la modélisation neuronale.

Ces critères reposent sur deux grands principes assez proche l'un de l'autre, le principe du maximum de vraisemblance et les fonctions d'influences. Les divers critères présentés sont ensuite comparés sur un exemple de simulation, aussi bien au cours d'une identification hors-ligne que en-ligne.

Il ressort de ces expériences que chaque critère semble plus ou moins adapté à chaque cas et qu'en l'absence de connaissance a priori, il peut s'avérer nécessaire de tester chacun des critères pour pouvoir sélectionner le plus adapté au cas considéré.

Nous avons finalement utilisé ces critères pour modéliser un système statique industriel. L'extraction des points de fonctionnement permettant de minimiser l'influence des valeurs aberrantes, l'utilisation de critères robustes nous a surtout permis de modéliser notre système sans faire forcément l'hypothèse d'une distribution du bruit normale.

V.8. Références.

ASTRÖM (1980) K.J. 'Maximum likelihood and prediction error methods', *Automatica*, Vol.16, 551-574.

BARNETT V., T. LEWIS (1984) *Outliers in statistical data,* J. Wiley, New-York, 2nd edition.

BLOCH G., D. THEILLIOL, P. THOMAS (1994) 'Robust identification of non-linear SISO Systems with Neural Networks', *preprints of 10th IFAC Symp. on System Identification SYSID '94,* Copenhagen, Denmark, 4-6 july, Vol. 3, 483-488.

BLOCH G., P. THOMAS, M. OULADSINE, M. LAIRI (1996) 'On several outlier-robust training rules of neural networks for identification of non-linear systems', *preprints of 8th International Conference on Neural Networks and their Applications NEURAP'96,* Marseille, France, 20-22 March, 13-19.

BLOCH G., P. THOMAS, D. THEILLIOL (1997) 'Accomodation to outliers in identification of non-linear SISO systems with neural networks', *Neurocomputing,* Vol.14, 1, 85-99.

CHEN D.S., R.C. JAIN (1991) 'A robust back propagation learning algorithm for function approximation', *Proc. Third Int. Workshop on Artificial Intelligence and Statistics*, Fort Lauderdale, janvier, 218-239.

CHEN S., S.A. BILLINGS, P.M. GRANT (1990) 'Non-linear system identification using neural networks', *Int. J. Control* , Vol.51, 6, 1191-1214.

CICHOCKI A;, R. UNBEHAUEN (1993) *Neural networks for optimization and signal processing* , John Willey and Sons, New-York, U.S.A.

DEMUTH H., M. BEALE (1994) *Neural network toolbox user's guide*, V2.0, The MathWorks, Inc.

ERSHOV A.A. (1978) 'Stable methods of estimating parameters (Survey)', *Automation and Remote Control*, vol.39, 7, 1152-1181.

HAGAN M.T., H. DEMUTH, M. BEALE (1995) *Neural network design*, PWS publishing company, Boston, U.S.A.

HAMPEL F.R. (1971) 'A general qualitative definition of robustness', *Ann. Math. Statistics.*, Vol.42, 1887-1896.

HAMPEL F.R., E.N. RONCHETTI, P.J. ROUSSEEUW, W.A. STAHEL (1987) *Robust statistics - The approach based on influence functions*, John Wiley, New-York, U.S.A.

HANSEN L.K., M.W. PEDERSEN (1994) 'Controlled growth of cascade correlation nets', *Int. Conf. on Artificial Neural Networks ICANN'94*, Sorento, Italie, 797-800.

HUBER P.J. (1964) 'Robust estimation of a location parameter', *Ann. Math. Stat.*, Vol.35, 73-101.

HUBER P.J. (1981) *Robust statistics*, John Wiley, New-york, U.S.A.

LIANO K. (1996) 'Robust error for supervised neural network learning with outliers', *IEEE Transactions on Neural Networks*, Vol.7, 1, 246-250.

LJUNG L. (1987) *System identification : theory for the user*, Prentice-Hall, Englewood Cliffs, N.J., U.S.A.

LJUNG L. (1992) *System identification toolbox user's guide*, V3.0a, The MathWorks, Inc.

MCMICHAEL D.W. (1990) 'Robust recursive Lp estimation', *IEE Proc.*, Vol.137, Pt. D, 2, 67-76.

MOSTELLER F., J.W. TUKEY (1977) *Data analysis and regression* , Addison-Wesley, Reading, Mass.

NGUYEN D., B. WIDROW (1990) 'Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights', *Proc. of International Joint Conf. on Neural networks*, Vol.3, 21-26.

NORGAARD M. (1995) *Neural network based system identification toolbox*, Tech. Repport. 95-E-773, Institute of Automation, Technical University of Denmark.

POLJAK B.T., Y.Z. TSYPKIN (1980) 'Robust identification', *Automatica*, Vol.16, 53-63.

POLLARD J.F., M.R. BROUSSARD, D.B. GARRISON, K.Y. SAN (1992) 'Process identification using neural networks', *Computers Chem. Engng.*, Vol.16, 4, 253-270.

PUTHENPURA S., N.K. SINHA (1990) 'A robust recursive identification method', *Control-Theory and Advanced Technology*, Vol.6, 4, 683-695.

RAGOT J., D. MAQUIN, M. DAROUACH, G. BLOCH (1990) *Validation de données et diagnostic,* Hermès, Paris, France.

RAPACCHI B. (1994) *Une introduction à la notion de robustesse,* rapport interne du Centre Interuniversitaire de Calcul de Grenoble, France, http://www.grenet.fr/cicg/ntcm/cours/.

THOMAS P., G. BLOCH (1996) 'From batch to recursive outlier-robust identification of non-linear dynamic systems with neural networks', *Proceedings of the International Conference on Neural Networks ICNN'96,* Washington, D.C., U.S.A., 3-6 june, Vol. 1, 178-183.

TUKEY J.W. (1962) 'The future of data analysis', *Ann. Math. Statist.*, Vol.3, 1-67.

WALTER E., L. PRONZATO (1994) *Identification de modèles paramétriques à partir de données expérimentales,* Masson, Paris, France.

Conclusions et perspectives

Au cours de ce travail, nous avons démontré les liens important existant entre la modélisation neuronale et les méthodes classiques d'identification de modèles non-linéaires. Tout particulièrement, nous avons démontré l'équivalence entre le perceptron multicouches et un modèle paramétrique discret. De plus ces deux types de modélisation nécessitent des outils similaires tant au niveau des méthodes d'estimation des paramètres qu'au niveau de la détermination du vecteur de régression. Cependant, l'utilisation des réseaux de neurones pour la modélisation de systèmes non-linéaires présente l'énorme avantage de ne pas nécessiter la description analytique de la non-linéarité.

Par contre, l'utilisation d'un modèle neuronal pour identifier un système non-linéaire impose d'effectuer un certain nombre de choix concernant, par exemple, le nombre de couches cachées, le nombre de neurones dans la (les) couche(s) cachée(s), les fonctions d'activation utilisées par les neurones des couches cachée(s) et de sortie...

Nous nous sommes tout particulièrement intéressés ici à la détermination des poids initiaux du réseau, au choix de l'algorithme d'apprentissage et également au choix du critère à minimiser. Sur chacun de ces trois points, nous avons présenté les problèmes rencontrés par les utilisateurs de réseaux de neurones. Nous avons ainsi mis en évidence les points suivants :

- un mauvais choix des poids initiaux du réseau peut conduire à l'obtention d'un minimum local éloigné du minimum local recherché ou encore, fortement accroître le nombre d'itération, et donc, le temps de calcul, nécessaire pour que l'apprentissage converge vers un minimum,

- l'algorithme d'apprentissage va lui aussi influer sur la rapidité de convergence et également la convergence elle même. D'une part, la plupart des algorithmes de minimisation du critère sont des algorithmes de recherche locale du minimum et n'assurent pas l'obtention du minimum global. D'autre part certains algorithmes sont particulièrement lents et d'autres peuvent même diverger. De plus, il est notable que lors d'une modélisation en-ligne le temps de calcul utilisé par l'algorithme d'apprentissage doit être le plus court possible,

- enfin, la présence de valeurs aberrantes dans les données peut conduire, avec le critère quadratique classique, à l'obtention d'un modèle biaisé, et donc inadapté.

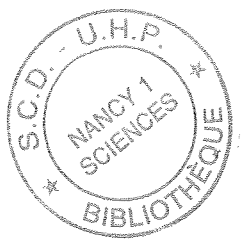
Face à chacune de ces difficultés, nous avons présenté et comparé plusieurs algorithmes permettant d'offrir une solution adaptée à la modélisation de tout système non-linéaire.

Ce travail a pour objet de présenter, à tout automaticien désireux d'utiliser un perceptron multicouches pour modéliser un système non-linéaire, les différents outils qui sont à sa disposition ainsi que des éléments de choix pour déterminer l'outil adapté à son cas. Cependant, certains problèmes important n'ont été abordés que succinctement dans ce mémoire.

Nous envisageons dans le futur d'aborder plus précisément les problèmes suivants.

- La détermination de la structure optimale du réseau, que ce soit par des méthodes constructives ou par élimination des connexions inutiles.
- La détermination du vecteur de régression.

Des travaux sur l'utilisation de modèles neuronaux pour la commande de systèmes non-linéaires sont actuellement effectués au sein du laboratoire. Diverses applications industrielles des perceptrons multicouches sont également en cours de réalisation.



Université HENRI POINCARÉ NANCY 1

Nom : **THOMAS**

Prénom : **Philippe**

DOCTORAT de l'UNIVERSITE HENRI POINCARE, NANCY-I

en AUTOMATIQUE

VU, APPROUVÉ ET PERMIS D'IMPRIMER

Nancy, le 26 MAR 1997 UHP 025/97

Le Président de l'Université

UNIVERSITÉ HENRI POINCARÉ, NANCY 1 * LE PRÉSIDENT

J.P. FINANCE

3C-F9

UNIVERSITÉ HENRI POINCARÉ · NANCY 1
24-30, rue Lionnois B.P.3069 54013 Nancy cedex
Tél : 83 32 81 81 Fax : 83 32 95 90
Adresse électronique : _____ @presn1.u-nancy.fr

Résumé :
Cette thèse est consacrée à l'identification de systèmes dynamiques non-linéaires SISO et MISO à l'aide de réseaux de neurones multicouches non-récurrents. Dans un premier temps, une présentation succincte de l'ensemble des méthodes d'identification non-linéaire est effectuée. Nous poursuivons notre étude par un bref historique des réseaux de neurones ainsi que par une présentation des divers modèles neuronaux existants. Une fois posé le cadre de ce travail, nous définissons plus particulièrement l'architecture générale du réseau de neurone retenu, puis nous présentons les algorithmes permettant d'adapter cette architecture générale à un cas précis. Ces choix concernent notamment le vecteur de régression et le nombre de neurones utilisés dans la couche cachée. Notre étude des réseaux de neurones s'effectuant dans le domaine précis de l'identification de systèmes, les liens importants existant entre la modélisation neuronale et les modèles non-linéaires classiques sont démontrés. Les divers critères de validation de modèles non-linéaires utilisables pour les réseaux neuronaux sont alors présentés. Le reste de ce travail est consacré aux diverses difficultés rencontrées lors de l'identification par réseau de neurones. La première difficulté se présente dès l'initialisation des poids du réseau. En effet, un mauvais choix des poids initiaux peut conduire à l'obtention d'un minimum local très éloigné du minimum global, à la saturation des neurones de la couche cachée, à une convergence lente. Afin de résoudre ce problème, divers algorithmes ont été proposés et comparés sur divers exemples. Les problèmes de lenteur de convergence, ou même de divergence, peuvent également être dus à l'algorithme d'apprentissage utilisé. Nous proposerons alors un nouvel algorithme permettant de s'affranchir de cette seconde difficulté. Ce nouvel algorithme sera alors comparé à l'algorithme plus classique RPE sur un exemple de simulation. Nous finirons notre étude en nous intéressant au troisième problème qui est posé par la présence de valeurs aberrantes dans les données d'identification. En effet, cette présence de valeurs aberrantes risque de conduire à des biais sur les paramètres. Nous proposons alors divers critères d'apprentissage qui sont robustes aux valeurs aberrantes. Ces critères sont comparés sur des exemples de simulation et des données industrielles réelles.

Mots Clefs :
Identification, systèmes non-linéaires, réseaux de neurones, perceptron multicouches, initialisation, minimisation du critère, robustesse.

Abstract:
This thesis deals with the identification of dynamical non-linear SISO and MISO systems with multilayer feedforward neural networks. Firstly, a short presentation of the non-linear identification methods is proposed and the neural network are reviewed. Secondly, the general architecture of the neural network used is more precisely defined. Some methods are presented to adapt this architecture to a particular case. These methods give the regressors and the number of neurons in the hidden layer. The relationships between neural identification and the most classical non-linear models are then shown. The validation criteria of non-linear models usable for the neural identification are presented. Three difficulties encountered in neural identification are investigated in the sequel. The first one is due to the initialisation of the parameters of the network. A bad choice of these initial parameters can lead to local minima very far from of the global minimum, to saturation of the hidden neurons, or to slow convergence. Two new algorithms are proposed to solve this problem and compared with others on three different examples. The slow convergence can be the result of the learning algorithm used. One algorithm is proposed to deal with this second difficulty. This algorithm is compared with the more classical RPE algorithm. This study is ends with the third studied problem which is posed by the presence of outliers in the identification data set. Indeed, outliers can produce biases on estimated parameters. Three robust criteria are then proposed and are compared with the classical quadratic criterion on a simulation example and on a real industrial data set.

Key words:
Identification, non-linear systems, neural networks, multilayers perceptron, initialisation, criterion minimisation, robustness.