***LES RESEAUX DE NEURONES POUR LA MODELISATION ET LA COMMANDE DE PROCEDES, J.P. Corriou, coordonnateur (Lavoisier Tec et Doc, 1995)***

# MODÉLISATION, CLASSIFICATION ET COMMANDE PAR RÉSEAUX DE NEURONES : PRINCIPES FONDAMENTAUX, MÉTHODOLOGIE DE CONCEPTION ET ILLUSTRATIONS INDUSTRIELLES

**I. RIVALS, L. PERSONNAZ, G. DREYFUS**
Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris
Laboratoire d'Electronique
10, rue Vauquelin
75005 PARIS

**J.L. PLOIX**
NETRAL S.A.
14, rue Verdi
92130 ISSY LES MOULINEAUX

## INTRODUCTION

Les réseaux de neurones connaissent depuis quelques années un succès croissant dans divers domaines des Sciences de l'Ingénieur ; celui du génie des procédés n'échappe pas à cette règle. Malheureusement, la littérature fourmille d'exemples où la mise en œuvre des réseaux de neurones relève plus de la recette que d'une approche raisonnée. De plus, les connotations biologiques du terme "réseaux de neurones", et l'utilisation du terme d'"apprentissage", ont souvent introduit une grande confusion ; elles ont conduit à relier abusivement les réseaux de neurones à l'Intelligence Artificielle, alors qu'ils sont fondamentalement des outils *statistiques.* L'objectif de cet article est de montrer comment, à partir des notions fondamentales, il est possible d'aboutir à une véritable méthodologie de mise en œuvre, notamment dans le cadre de la modélisation des processus. Nous montrons en particulier que, contrairement à une croyance répandue, les réseaux de neurones ne sont pas nécessairement des "boîtes noires" : bien au contraire, il est parfaitement possible, et même vivement recommandé, d'introduire dans le réseau de neurones, dès sa conception, toutes les connaissances mathématiques disponibles concernant le processus à modéliser ou à commander.



La première partie de l'article rappelle les définitions, et introduit la propriété essentielle des réseaux de neurones : la propriété d'*approximation universelle parcimonieuse*. Nous replaçons ainsi les réseaux de neurones dans le contexte mathématique approprié, qui est celui de la régression non linéaire. Dans la deuxième partie, nous introduisons les notions nécessaires pour mettre en œuvre les algorithmes d'apprentissage ; nous insistons sur le caractère *générique* des algorithmes d'apprentissage, qui constitue l'un des atouts des réseaux de neurones dans le domaine de la modélisation et de la commande. La troisième partie indique brièvement le lien entre la propriété d'approximation universelle et l'utilisation des réseaux de neurones en classification. Nous abordons ensuite le domaine d'application essentiel de l'article, celui de la modélisation de processus ; nous décrivons, et justifions mathématiquement, une méthodologie pour le choix de la structure d'un modèle neuronal, notamment en fonction des hypothèses que l'on peut faire sur la manière dont le bruit intervient dans le processus à modéliser. Nous montrons que les modèles d'état constituent une classe de modèles particulièrement efficaces, et nous illustrons cette propriété par la modélisation d'un actionneur de bras de robot, effectuée par plusieurs techniques, par des équipes différentes. Les deux dernières parties de l'article décrivent des applications mettant en œuvre la modélisation dans deux grands domaines : d'une part, la modélisation de procédés industriels (calcination, distillation) pour la simulation et la détection d'anomalies ; d'autre part, la modélisation pour la commande, illustrée par l'exemple du pilotage d'un véhicule 4x4 autonome.

## I. LES RÉSEAUX DE NEURONES FORMELS : DÉFINITIONS, PROPRIÉTÉ FONDAMENTALE, GÉNÉRALITÉS SUR L'APPRENTISSAGE

### I.1. DÉFINITIONS

Un réseau de neurones formels à temps discret est un système composé de deux types d'éléments, ou "unités" : les entrées du réseau et les neurones eux-mêmes. Chaque neurone (déterministe) est un processeur non linéaire (généralement simulé sur ordinateur, parfois réalisé sous forme de circuit électronique) qui, à chaque instant discret $k$, calcule son potentiel $v_i(k)$ et son activité $z_i(k)$ de la façon suivante :

$$z_i(k) = f_i\,(v_i(k)) \quad \text{où} \quad v_i(k) = \sum_{j \in P_i} \sum_{\tau=0}^{q_{ij}} c_{ij,\tau}\, z_j(k-\tau)$$

$P_i$ est l'ensemble des indices des unités du réseau propageant leur activité au neurone $i$. Son potentiel $v_i(k)$ est une somme des valeurs des activités de ces unités, à l'instant $k$ ou à des instants précédents, pondérée par les coefficients $c_{ij,\tau}$. $q_{ij}$ est le retard maximal du neurone $i$ sur l'entrée ou le neurone $j$. Si $q_{ij}=0$ pour tout $j$, le neurone $i$ est statique. La fonction $f_i$, fonction d'activation du neurone $i$, est en général *non linéaire*. Ce peut être la distribution de Heaviside, une sigmoïde (c'est le type de fonction que nous utilisons dans ce travail), une fonction à base

radiale (RBF), etc. ; nous appelons “neurone linéaire” un simple sommateur, c’est-à-dire un neurone dont la fonction d’activation est l’identité.

Un réseau de neurones est conçu pour remplir une tâche que le concepteur définit par un ensemble de valeurs d’entrée, et par un ensemble de valeurs désirées correspondantes pour les activités de certains neurones du réseau que l’on appelle neurones de sortie (les éléments de ces ensembles sont appelés “exemples d’apprentissage”). Les neurones qui ne sont pas des neurones de sortie sont dits cachés. Le réseau de la figure I.1 possède deux entrées, deux neurones cachés, et un neurone de sortie.

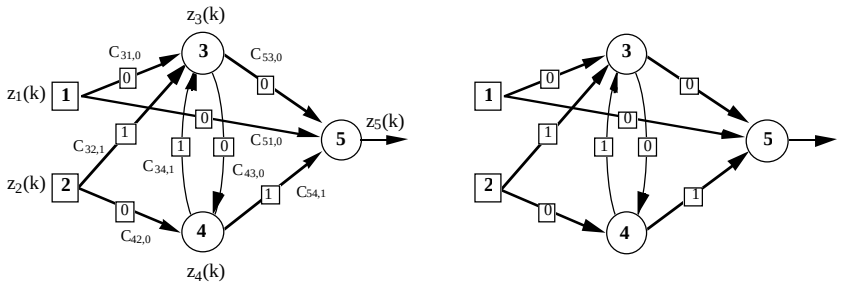


Figure I.1.
Exemple de réseau de neurones et de son graphe.

Pour caractériser un réseau de neurones, il est pratique d’utiliser son graphe. Ses nœuds sont les neurones, ses racines les entrées du réseau, et les arcs sont les connexions pondérées par leur retard. S’il n’y a pas de cycle dans ce graphe, le réseau est *non bouclé*, sinon il est *bouclé*. L’architecture d’un réseau est définie par le graphe du réseau, les coefficients de celui-ci, et par les fonctions d’activation des neurones. Le caractère bouclé ou non du réseau, ainsi que les fonctions d’activation, peuvent être fixés en fonction de la tâche que doit remplir le réseau de neurones. Les valeurs des coefficients sont en général déterminées par un processus algorithmique appelé apprentissage (voir le paragraphe I.3) à partir des exemples d’apprentissage, mais les valeurs de certains coefficients peuvent être fixées à l’avance. Ainsi, dans le cas de la modélisation d’un processus, l’architecture peut être partiellement déterminée par des connaissances *a priori* ; les valeurs de coefficients ayant une signification physique peuvent être fixées préalablement à l’apprentissage.



### I.1.1 Les réseaux de neurones non bouclés

Un réseau est non bouclé, ou statique, si son graphe ne possède pas de cycle. Il réalise donc, de manière générale, une relation *algébrique* non linéaire entre ses entrées et ses sorties.

*Dans le contexte du traitement du signal et de l’automatique*, un réseau non bouclé réalise un filtre transverse non linéaire à temps discret. Ce filtre peut posséder des synapses à retard. On a intérêt à mettre le réseau sous une forme équivalente, dite *forme canonique*, constituée uniquement de neurones à retard nul, ou neurones statiques : cette forme a l’avantage de faire apparaître les entrées effectives du réseau à chaque instant, et de faciliter l’apprentissage (car toutes les connexions sont de même type). Ses unités (entrées et neurones) sont ordonnées, et les connexions ne peuvent aller que d’une unité à un neurone dont l’indice est supérieur. Chaque neurone $i$ du réseau calcule à l’instant $k$ :

$$z_i(k) = f_i(v_i(k)) \quad \text{avec} \quad v_i(k) = \sum_{j \in P_i} c_{ij}\, z_j(k)$$

où $v_i(k)$ est le potentiel du neurone $i$ à l’instant $k$, $z_i(k)$ son activité, $P_i$ est l’ensemble des indices des unités propageant leur activité au neurone $i$, et $c_{ij}$ est le coefficient de la connexion reliant l’unité $j$ au neurone $i$.

Un réseau non bouclé réalise donc une transformation entrée-sortie non linéaire $\psi$ paramétrée par les coefficients $C$ du réseau (figure I.2) :

$$O(k) = \psi\,(I(k);\, C)$$

où $O(k) \in \mathbb{R}^{N_O}$ est le vecteur des sorties à l’instant $k$, c’est-à-dire des activités des neurones de sortie du réseau à l’instant $k$, et $I(k) \in \mathbb{R}^{N_I}$ est le vecteur des entrées externes à l’instant $k$. $\psi(.;\, C) : \mathbb{R}^{N_I} \rightarrow \mathbb{R}^{N_O}$, représente la fonction réalisée par les neurones du réseau interconnectés avec les coefficients $C$.

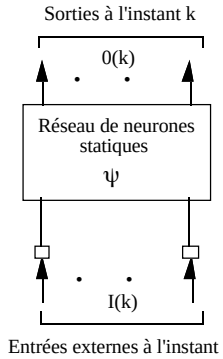


Figure I.2.
Forme canonique d’un réseau de neurones non bouclé.

La figure I.3 illustre l'utilisation de la forme canonique pour l'apprentissage de filtres en cascade : un réseau de type TDNN (Time Delay Neural Network [WAI89]) est représenté sur la figure I.3a, et sa forme canonique sur la figure I.3b. Cette forme canonique possède 7 entrées externes, 5 neurones cachés, et un neurone de sortie.

L'architecture de réseau non bouclé la plus générale est celle du réseau complètement connecté, représentée sur la figure I.4. Tous les neurones cachés et les neurones de sortie sont connectés aux unités d'indice inférieur.

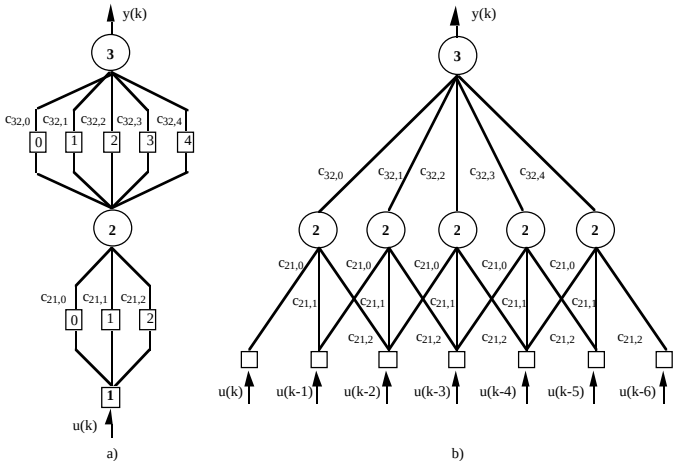


Figure I.3.
Mise sous forme canonique d'un filtre transverse non linéaire à temps discret (TDNN).
I(k) = [u(k), u(k-1), u(k-2), u(k-3), u(k-4), u(k-5), u(k-6)] ; O(k) = y(k) .

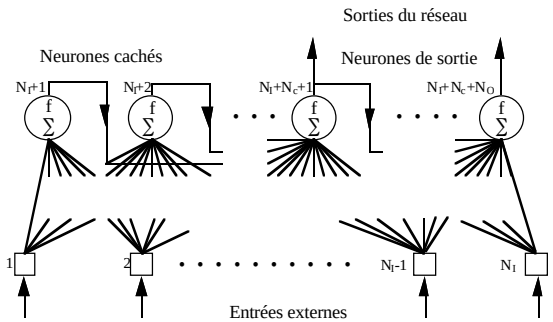


Figure I.4.
Réseau de neurones non bouclé complètement connecté.

Une architecture historiquement très utilisée, surtout en raison de sa pertinence en classification, est celle du réseau à couches (figure I.5). Les neurones cachés sont répartis en couches successives, les neurones appartenant à une couche donnée n'étant commandés que par les neurones de la couche précédente, et ceux de la première couche n'étant connectés qu'aux entrées externes. Mentionnons que la propriété d'approximation universelle des réseaux de neurones est valable pour la famille des réseaux possédant seulement une couche de neurones cachés (voir le paragraphe I.2).



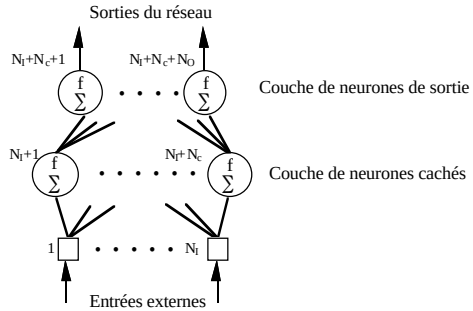


Figure I.5.
Réseau de neurones non bouclé à une couche de neurones cachés.

*Dans le contexte de la classification automatique*, un tel réseau réalise une estimation de la probabilité d'appartenance d'une forme à une classe ; nous justifions, dans la partie II, l'utilisation des réseaux de neurones dans ce domaine. La figure I.5 représente l'architecture généralement utilisée dans ce type d'application.

### I.1.2. Les réseaux de neurones bouclés

Un réseau est bouclé, ou dynamique, si son graphe possède au moins un cycle. Il constitue un filtre récursif non linéaire à temps discret. Comme pour les réseaux non bouclés, on a intérêt, pour l'apprentissage, à mettre le réseau sous une forme équivalente dite canonique, constituée de neurones statiques. En effet, tout réseau de neurones bouclé à temps discret d'ordre $N_S$ peut être représenté par un réseau dont la dynamique est décrite par $N_S$ équations aux différences couplées d'ordre 1, mettant en jeu $N_S$ variables d'état, et $N_I$ entrées externes. Cette forme canonique n'est en général pas unique.

Le comportement dynamique d'un réseau de neurones bouclé peut être décrit par la représentation d'état paramétrée par les coefficients $C$, représentée sur la figure I.6) :

$$\begin{cases} S(k+1) = \varphi\,(S(k),\, I(k);\, C) \\ O(k) = \psi\,(S(k),\, I(k);\, C) \end{cases}$$

où $I(k) \in \mathbb{R}^{N_I}$ est le vecteur des entrées externes, $S(k) \in \mathbb{R}^{N_S}$ le vecteur des variables d'état, $O(k) \in \mathbb{R}^{N_O}$ le vecteur des sorties, à l'instant $k$, et $S(k+1) \in \mathbb{R}^{N_S}$ est le vecteur des variables

d'état à l'instant $k$+1. $\varphi$ (., .; $C$) et $\psi$ (., .; $C$) représentent les fonctions réalisées par le réseau de neurones statiques de la forme canonique interconnectés avec les coefficients $C$.

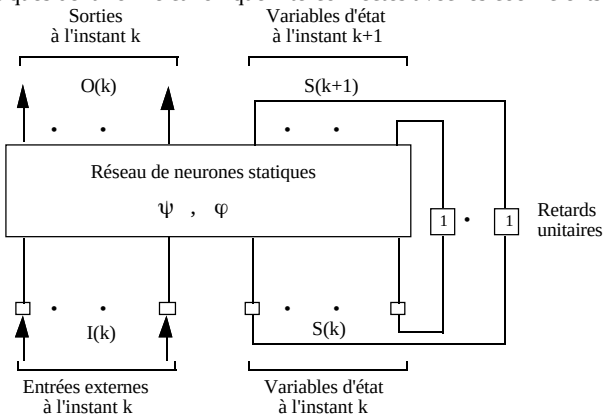


Figure I.6.
Forme canonique d'un réseau de neurones bouclé.

Une forme canonique d'un réseau de neurones bouclé est ainsi définie à partir d'un réseau non bouclé constitué de neurones statiques possédant $N_I$ entrées externes, $N_S$ entrées d'état (les variables d'état à l'instant k), $N_C$ neurones cachés et $N_O$ neurones de sortie (neurones pour les activités desquels il existe une valeur désirée). Les sorties du réseau à l'instant $k$ sont les activités des $N_O$ neurones de sortie, et les variables d'état à l'instant $k$+1 sont les activités de $N_S$ neurones que nous appelons *neurones d'état*. Ces neurones d'état sont soit des neurones cachés, soit des neurones de sortie.

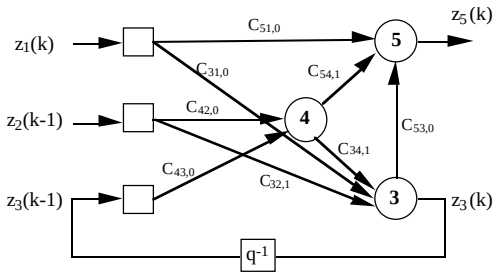


Figure I.7.
Mise sous forme canonique du réseau de la figure 1.
$I(k) = (z_1(k), z_2(k-1))^T$ ; $S(k) = z_3(k-1)$ ; $O(k) = z_5(k)$ .

Par exemple, mettons le réseau de la figure I.1, bouclé comme l'indique son graphe, sous une forme canonique (figure I.7). Ce réseau possède deux entrées externes ($N_I$=2) $z_1(k)$ et $z_2(k\text{-}1)$, une entrée d'état ($N_S$=1) $z_3(k\text{-}1)$, deux neurones cachés ($N_C$=2), dont un neurone



d'état ($N_S$=1) dont l'activité donne la nouvelle valeur de la variable d'état $z_3(k)$, et un neurone de sortie ($N_O$=1) dont l'activité donne la valeur de la sortie $z_5(k)$.

## I.2. PROPRIÉTÉ FONDAMENTALE DES RÉSEAUX DE NEURONES : L'APPROXIMATION UNIVERSELLE PARCIMONIEUSE

La propriété d'*approximation universelle parcimonieuse* justifie, dans une large mesure, l'utilisation des réseaux de neurones dans le domaine des Sciences de l'Ingénieur. Nous soulignons dans le paragraphe I.3 que le caractère *générique* des algorithmes d'apprentissage constitue la seconde justification de cette utilisation.

### I.2.1. Réseaux non bouclés

Nous avons vu dans le paragraphe précédent que tout réseau de neurones peut être mis sous une forme canonique comprenant un réseau de neurones statiques ; les propriétés générales des réseaux de neurones, bouclés ou non, dépendent donc des propriétés des réseaux de neurones non bouclés.

La propriété d'*approximation universelle* des réseaux de neurones peut s'énoncer comme suit : *pour toute fonction déterministe suffisamment régulière, il existe au moins un réseau de neurones non bouclé, possédant une couche de neurones cachés et un neurone de sortie linéaire, qui réalise une approximation de cette fonction et de ses dérivées successives, au sens des moindres carrés, avec une précision arbitraire.* Ce théorème ne s'applique que si les fonctions d'activation des neurones cachés possèdent des propriétés que nous ne développons pas ici ; les fonctions sigmoïdes et les fonctions à base radiale remplissent les conditions d'application du théorème.

Il va de soi que la propriété d'approximation universelle n'est pas spécifique aux réseaux de neurones : les polynômes, les séries de Fourier, les fonctions splines, possèdent cette même particularité. Ce qui différencie les réseaux de neurones des autres approximateurs universels usuels, c'est leur *parcimonie :* pour obtenir une approximation d'une précision donnée, les réseaux de neurones utilisent *moins de paramètres* que les approximateurs usuels. En particulier, *le nombre de paramètres varie essentiellement de manière linéaire en fonction du nombre de variables de la fonction que l'on cherche à approcher*, alors qu'il varie beaucoup plus rapidement (voire exponentiellement) avec la dimension de l'espace des entrées dans le cas des approximateurs usuels [HOR94].

*En pratique, les réseaux de neurones sont donc avantageux, par rapport aux approximateurs conventionnels, lorsque l'on cherche à résoudre un problème à plus d'une ou deux variables.*

Cet avantage est particulièrement intéressant du point de vue du temps de calcul, et surtout du point de vue de la quantité d'information nécessaire pour le calcul des coefficients :

- du point de vue du temps de calcul, l'estimation des coefficients du réseau (*l'apprentissage*), est d'autant plus rapide que le nombre de paramètres à calculer est petit ;
- lorsque la fonction que l'on cherche à approcher n'est pas connue analytiquement, mais seulement par l'intermédiaire de l'ensemble d'apprentissage, la taille de l'échantillon (donc le nombre d'exemples) nécessaire croît avec le nombre de poids. Le fait d'utiliser moins de coefficients que les méthodes classiques de régression permet donc une économie en nombre d'exemples, ce qui peut être particulièrement important lorsque l'acquisition des exemples est coûteuse ou lente.

En résumé :

- les réseaux de neurones non bouclés sont des outils statistiques de régression qui permettent l'approximation, au sens des moindres carrés, de toute fonction non linéaire suffisamment régulière,
- cette approximation est économe en nombre de coefficients, donc en nombre d'exemples,
- l'estimation des paramètres de la régression (les poids du réseau) à partir d'un ensemble d'échantillons constitue l'apprentissage.

Il doit donc être bien clair que le fonctionnement des réseaux de neurones est *fondamentalement différent* de celui des systèmes de traitement symbolique de l'information (tels que les systèmes experts) ; notons également que, par essence, les réseaux de neurones sont aptes à traiter des informations *numériques* beaucoup plus que des informations *symboliques*.

### I.2.2. Réseaux bouclés

La propriété d'approximation universelle des réseaux de neurones prend un sens différent s'il s'agit d'un réseau bouclé. En effet, considérons par exemple la modélisation d'un processus représenté par un modèle d'état :

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

où $f$ et $g$ sont des fonctions continues. Il existe bien un réseau bouclé tel que, pour des entrées données $x_p(k)$ et $u(k)$, les variables d'état $S(k+1)$ et les sorties $O(k)$ calculés par le réseau approchent avec une précision fixée l'état $x_p(k+1)$ et la sortie $y_p(k)$ du processus. En effet, il suffit pour cela que le réseau non bouclé de sa forme canonique soit constitué de deux sous-réseaux dont l'un approche la fonction $f$, et l'autre la fonction $g$, les conditions d'application du théorème d'approximation universelle étant remplies. Mais ceci n'implique pas nécessairement



que le réseau réalise l'approximation du système dynamique en toute circonstance. En effet, une propriété d'approximation universelle pour les réseaux bouclés peut s'énoncer de la manière suivante : pour tout système dynamique, pour toute précision désirée $\varepsilon$, pour un intervalle de temps fini [0,T], pour des entrées $u(.) : u(.) : [0,T] \rightarrow U \subseteq R^{n_u}$ et un état initial $x(0) \in X \subseteq R^{n_x}$, il existe un réseau de neurones bouclé qui approche le comportement entrée-sortie du système avec la précision $\varepsilon$ sur l'intervalle $[0,T]$ et sur les ensembles $U$ et $X$ [SON93]. La définition de l'approximation d'un système dynamique par un réseau de neurones bouclé est donc restreinte à un domaine de l'espace des entrées (comme pour les réseaux non bouclés), mais aussi à un domaine de l'espace des états, et à un intervalle de temps fini : un tel approximateur peut donc ne pas refléter des caractéristiques fondamentales du processus qu'il est censé approcher, sa stabilité par exemple.

## I.3. GÉNÉRALITÉS SUR L'APPRENTISSAGE

Le problème de l'approximation d'une fonction n'est qu'un aspect de l'apprentissage des réseaux de neurones, la propriété d'approximation universelle étant seulement une condition nécessaire à leur utilisation comme modèles et correcteurs non linéaires généraux.

Dans le cas où la tâche du réseau de neurones est une tâche de modélisation d'un processus physique, il est raisonnable de supposer que les sorties mesurées sur le processus obéissent à des lois déterministes, et de chercher une expression mathématique des fonctions $f$ et $g$. La propriété d'approximation universelle est donc une propriété nécessaire du modèle utilisé à cette fin, mais elle n'est pas suffisante. En effet :

- d'une part, dans la pratique, les fonctions à déterminer sont définies par un ensemble fini de couples {entrées-sorties mesurées}, qui ne permet pas de déterminer ces fonctions de façon univoque ; le but de l'apprentissage est alors de trouver la solution la plus parcimonieuse, passant par tous les points d'apprentissage, qui, si l'ensemble d'apprentissage est bien choisi, tendra vers les fonctions $f$ et $g$ supposées régir le fonctionnement du processus.
- d'autre part, comme nous le verrons dans le paragraphe consacré à la modélisation, on est souvent en présence de processus affectés de perturbations aléatoires ; dans ce cas, le but de l'apprentissage ne peut être de passer par tous les points de l'ensemble d'apprentissage : bien que l'on ne dispose pas des valeurs prises sur l'ensemble d'apprentissage par les fonctions $f$ et $g$ supposées régir le fonctionnement du processus, il doit ajuster les coefficients du réseau de façon que les fonctions qu'il réalise tendent vers $f$ et $g$. La mise au point d'un système d'apprentissage en fonction des hypothèses faites sur les lois déterministes et les perturbations aléatoires affectant un processus fait l'objet de la partie III.

Dans le cas où la tâche du réseau est de réaliser une loi de commande imposant une dynamique désirée à un processus pour lequel on dispose d'un modèle, la démonstration de l'existence d'une telle loi de commande est en elle-même un problème. En effet, si la synthèse

du correcteur est effectuée à l'aide d'un modèle neuronal, dont il est difficile de déterminer les caractéristiques de façon analytique, cette existence peut être difficile à établir. Ces problèmes spécifiques à la commande ne seront pas abordés ici.

Dans ce paragraphe, nous donnons les principes et décrivons la mise en œuvre des procédures d'apprentissage, indépendamment des considérations d'existence que nous venons d'évoquer.

### I.3.1. Principe général

L'architecture du réseau de neurones n'est souvent que partiellement imposée par la tâche à réaliser : les entrées, l'état, et les sorties du réseau peuvent être fixées en fonction de celle-ci par le concepteur, ainsi que le type et la connectivité des neurones. Mais le nombre de neurones ne peut être fixé *a priori*, et il est en général déterminé selon une procédure itérative, suivant le succès de l'apprentissage (il existe des méthodes systématiques de sélection de modèles dynamiques [LEO87] [URB94]). L'architecture du réseau étant fixée, le but de l'apprentissage est l'estimation des coefficients pour remplir au mieux la tâche à laquelle le réseau est destiné.

*Apprentissage d'un réseau de neurones.*

Comme mentionné plus haut, l'apprentissage d'un réseau est définie par :

- un *ensemble d'apprentissage* constitué de $N$ exemples, chacun étant constitué d'un vecteur $\{I(k)\}$ appliqué aux entrées du réseau, et du vecteur $\{D(k)\}$ des valeurs désirées correspondantes pour les sorties. Dans le cas de l'automatique et du traitement du signal, ces exemples sont ordonnés temporellement ; on parle alors de séquences d'apprentissage. Il est très important que cet ensemble d'apprentissage soit suffisamment riche : il faut qu'il couvre aussi complètement que possible le domaine de fonctionnement désiré pour le réseau, et que le nombre d'exemples soit grand devant le nombre de coefficients du réseau. La propriété de parcimonie des réseaux de neurones permet de faciliter la réalisation de cette condition. Notons que la nécessité de disposer d'exemples en nombre suffisant, et suffisamment représentatifs, est souvent présentée comme une limitation propre aux réseaux de neurones ; ceci est incorrect : c'est une limitation commune à *toute méthode statistique* ; grâce à la propriété de parcimonie, cette contrainte est plus facile à satisfaire lorsque l'on utilise les réseaux de neurones que lorsque l'on utilise d'autres méthodes de régression.
- une *fonction de coût* à minimiser : en effet, la tâche ne consiste pas nécessairement à rendre les sorties du réseau égales aux sorties désirées ou "proches" de celles-ci. Par exemple, pour un problème de régulation, on peut souhaiter minimiser également le coût énergétique de la commande. Le critère fera donc intervenir non seulement l'écart de la sortie à la valeur de consigne, mais également l'énergie dépensée. Ou bien, si le processus possède plusieurs sorties, on peut attacher plus d'importance à certaines d'entre elles ; cela se traduit par une pondération des différents termes de la fonction de coût (c'est une généralisation de la commande linéaire quadratique).



L'apprentissage d'un réseau de neurones est ainsi défini comme un problème d'optimisation qui consiste à trouver les coefficients du réseau minimisant une fonction de coût. L'apprentissage est mis en œuvre par un *système d'apprentissage* qui comprend le réseau dont les coefficients sont à estimer, les séquences d'apprentissage et leur processus générateur (processus, modèle de référence…), et l'algorithme utilisé pour l'estimation.

*Exemple de la modélisation d'un processus non linéaire.*

La figure I.8 représente le système d'apprentissage pour la modélisation d'un processus : le but est d'estimer les coefficients du réseau prédicteur. Les séquences d'apprentissage sont constituées de la séquence des commandes $\{u(k)\}$ appliquées au processus et au prédicteur, et de la séquence des sorties désirées pour le prédicteur, qui sont les sorties $\{y_p(k)\}$ mesurées sur le processus. Les coefficients sont estimés de manière à minimiser une fonction de coût définie à partir de l'écart $y_p - y$, à l'aide d'un algorithme d'apprentissage approprié.

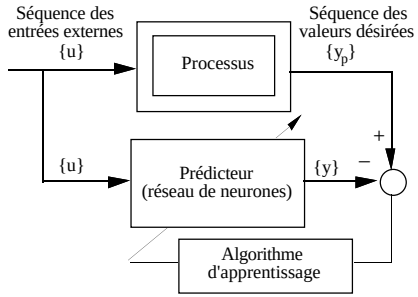


Figure I.8.
Système d'apprentissage pour la modélisation d'un processus.

### I.3.2. Expression de la fonction de coût

Dans cet article, nous nous intéressons à la mise au point de systèmes *non adaptatifs*[1] : la phase d'apprentissage et la phase d'utilisation des réseaux considérés sont distinctes. Ainsi, un classifieur non adaptatif ne subira plus de modifications de ses coefficients pendant son utilisation. Dans ce cadre non adaptatif, l'ensemble d'apprentissage est de taille finie.

Nous nous plaçons dans le cas où la fonction de coût est une fonction quadratique des erreurs $\{E(k)\}$, écarts entre les sorties du réseau $\{O(k)\}$ et les sorties désirées $\{D(k)\}$. La minimisation de cette fonction de coût est effectuée itérativement en modifiant les coefficients à

[1] Pour une présentation de l'apprentissage de systèmes adaptatifs entrée-sortie, voir [NER93].

chaque présentation de l'ensemble d'apprentissage : les erreurs $\{E(k)\}$ sont calculées à l'aide du réseau muni des coefficients disponibles à la fin de l'itération précédente, et de l'ensemble d'apprentissage. En classification, on parle généralement d'algorithme de minimisation du coût total. En automatique et en traitement du signal, un tel algorithme d'apprentissage est dit *itératif*, et *non récursif*.

En classification, on utilise fréquemment l'algorithme du coût partiel défini sur un ou plusieurs exemples de l'ensemble d'apprentissage. En automatique et en traitement du signal, le même type de démarche conduit à un algorithme dit *récursif* ; ces algorithmes ne présentent de véritable intérêt que pour les systèmes adaptatifs.

Cet article ayant pour objet la mise en œuvre de systèmes non adaptatifs, nous utiliserons dans les parties suivantes uniquement des algorithmes non récursifs et itératifs. L'expression de la fonction de coût à l'itération *i* définie sur la totalité de l'ensemble d'apprentissage est la suivante :

$$J(C, i) = \frac{1}{N} \sum_{k=1}^{N} E^i(k)^T W E^i(k) = \frac{1}{N} \sum_{k=1}^{N} \left(D(k) - O^i(k)\right)^T W \left(D(k) - O^i(k)\right)$$

où $C$ représente les coefficients du réseau disponibles à l'itération i, $E^i(k)$ le vecteur des erreurs pour l'exemple (ou l'instant) $k$ et à l'itération $i$, $W$ une matrice définie positive (qui sera le plus souvent choisie diagonale), $D(k)$ le vecteur des sorties désirées pour l'exemple $k$, et $O^i(k)$ le vecteur des sorties du réseau pour l'exemple $k$ et à l'itération $i$.

### I.3.3. Minimisation de la fonction de coût

Puisque nous utilisons des réseaux de neurones, les sorties du système subissant un apprentissage sont en général des fonctions non linéaires des coefficients à estimer. La recherche du minimum de la fonction de coût ne peut s'effectuer à l'aide des moindres carrés ordinaires, et demande donc l'utilisation de méthodes de programmation non linéaire. Les algorithmes de calcul du gradient de la fonction de coût ainsi que des méthodes de minimisation du premier et du second ordre sont présentées dans [RIV95].

Au-delà des aspects techniques de ces algorithmes, il faut retenir leur caractère *générique* : l'un des atouts majeurs des réseaux de neurones réside dans le fait que les algorithmes d'apprentissage sont applicables *pour tout réseau mis sous forme canonique*. Ainsi, le concepteur a toute liberté dans le choix de l'architecture la mieux adaptée au problème posé : il sait que, quelle que soit la structure du réseau, il pourra toujours utiliser la même panoplie d'algorithmes d'apprentissage ; cette souplesse permet notamment de concevoir des réseaux dont l'architecture est très fortement guidée par la physique du processus à modéliser et/ou à commander, exprimée par les équations des modèles de connaissance. La généricité des algorithmes rend ainsi possible la conception de "modèles neuronaux de connaissance" qui seront introduits dans la partie IV.



## II. CLASSIFICATION

Dans le domaine du génie des procédés, les réseaux de neurones ont été utilisés fréquemment en tant que classifieurs, par exemple pour l'identification d'anomalies. Dans cette brève partie, nous montrons comment les propriétés de classification des réseaux de neurones découlent de la propriété d'approximation universelle, et nous mettons en évidence les avantages et limitations des réseaux de neurones par rapport aux autres classifieurs.

### II.1. CLASSIFICATION ET APPROXIMATION DE FONCTION

Considérons un problème à deux classes *A* et *B*, et considérons la fonction $\Phi(\boldsymbol{x})$ qui vaut +1 si l'objet décrit par le vecteur $\boldsymbol{x}$ appartient à la classe *A*, et qui vaut zéro si cet objet appartient à la classe *B*. *L'approximation, au sens des moindres carrés, de la fonction $\Phi(\boldsymbol{x})$ n'est autre que la probabilité d'appartenance de l'objet $\boldsymbol{x}$ à la classe A.* Or, nous venons de voir que les réseaux de neurones constituent de bons approximateurs de fonctions non linéaires : ils sont donc de bons candidats pour l'évaluation d'une *probabilité d'appartenance,* suivie d'une décision (qui n'est pas nécessairement prise par le réseau de neurones lui-même) quant à l'appartenance de l'objet à l'une ou l'autre classe.

Contrairement à une croyance largement répandue, les réseaux de neurones ne sont donc *pas* limités à une prise de décision binaire ; bien au contraire, ils sont *intrinsèquement* capables de réaliser une estimation d'une probabilité d'appartenance. Cette propriété est très souvent ignorée, alors qu'elle est particulièrement précieuse lorsque l'on veut introduire un réseau de neurones comme outil de classification dans des systèmes complexes, contenant par exemple d'autres classifieurs, fondés sur des principes différents, ou utilisant des représentations différentes pour les formes à classer [PRI95].

### II.2. CLASSIFICATION NEURONALE ET NON NEURONALE

Nous venons de justifier mathématiquement le fait que les réseaux de neurones fournissent une estimation de la probabilité d'appartenance à une classe. Les réseaux de neurones ne sont pas les seuls à avoir cette propriété : il existe notamment des méthodes de classification, dites *paramétriques*, qui consistent à faire une hypothèse sur la forme de la distribution de probabilité d'appartenance, et à ajuster les paramètres de cette distribution. Le résultat final est donc, en principe, le même que celui que l'on obtient à l'aide d'un réseau de neurones. Il n'y a donc aucune raison *a priori* de penser que les réseaux de neurones sont supérieurs, en termes de taux de classification, à ceux, parmi les classifieurs conventionnels, qui permettent d'estimer de

manière précise les probabilités d'appartenance. Le seul avantage que peuvent présenter les réseaux de neurones dans ce domaine, c'est peut-être une certaine facilité de mise en œuvre, et certainement une parallélisation facile si le réseau est réalisé électroniquement, ou s'il est implanté sur un ordinateur parallèle.

# III. MODÉLISATION

## III.1. INTRODUCTION

La modélisation d'un processus consiste à représenter son comportement dynamique à l'aide d'un modèle mathématique paramétré. Ce modèle sera utilisé pour l'apprentissage d'un correcteur, ou au sein d'un système de commande, ou encore comme simulateur du processus.

*La première phase* d'une modélisation consiste à rassembler les connaissances dont on dispose sur le comportement du processus, à partir d'expériences et/ou d'une analyse théorique des phénomènes physiques mis en jeu. Ces connaissances conduisent à faire plusieurs hypothèses de modèles. Chacun de ces modèles dynamiques, appelés *modèles-hypothèse*, réalise des fonctions non linéaires entre ses variables de commande, d'état, et de sortie. Dans le cas où ces fonctions sont inconnues, on parle de modèle "boîte noire". Si certaines fonctions peuvent être fixées à partir de l'analyse physique, on parle alors de modèle de connaissance. L'approche "boîte noire" utilise, le plus souvent, des modèles *entrée-sortie*, alors qu'une *représentation d'état* est adoptée pour les modèles de connaissance.

On est ainsi conduit à un ensemble de modèles-hypothèse concurrents. La seconde phase, décrite ci-dessous, va permettre de sélectionner le meilleur d'entre eux.

*La seconde phase* de la modélisation, aussi appelée identification, consiste à estimer les paramètres des modèles concurrents. Pour chacun d'eux, on met en œuvre un *système d'apprentissage* constitué de la *forme prédicteur* du modèle-hypothèse, et d'un *algorithme d'apprentissage* (ou d'estimation). L'estimation des paramètres du modèle est effectuée en minimisant une *fonction de coût* définie à partir de l'écart entre les sorties mesurées du processus et les valeurs prédites (erreur de prédiction). La qualité de cette estimation dépend de la richesse des séquences d'apprentissage et de l'efficacité de l'algorithme utilisé.

À l'issue de l'identification de l'ensemble des modèles-hypothèse, on retient l'hypothèse correspondant au meilleur prédicteur neuronal obtenu ; la validation finale du modèle neuronal est effectuée en fonction de ses performances pour l'utilisation prévue (commande ou simulation).



Nous allons décrire les modèles-hypothèse considérés dans cet article, et définir la *forme prédicteur* associée à un *modèle-hypothèse* donné ; ce prédicteur est l'élément principal du *système d'apprentissage*.

### III.1.1. Modèles "boîte noire" et modèles de connaissance

La modélisation utilisant des modèles *boîte noire entrée-sortie* a été traitée dans [NER94]. Nous présentons dans la partie IV.4 une application de ce type pour la modélisation d'un procédé de calcination.

*Toujours dans le cadre d'une approche boîte noire, il peut être intéressant d'avoir recours à une représentation d'état*. En effet, les modèles d'état constituent une classe plus vaste de systèmes dynamiques que les modèles entrée-sortie : certains modèles d'état n'ont pas de représentation entrée-sortie globale équivalente. De plus, une représentation entrée-sortie équivalente à une représentation d'état donnée peut être d'ordre plus élevé. Plus précisément, soit un modèle d'état mono-entrée mono-sortie d'entrée $u$ et de sortie $y$ :

$$\begin{aligned} x_p(k) &= f\,(x_p(k\text{-}1),\, u(k\text{-}1)) \quad \text{(équation d'état)} \\ y_p(k) &= g\,(x_p(k)) \qquad\qquad \text{(équation de sortie)} \end{aligned} \tag{1}$$

où $x_p(k)$ est le vecteur d'état du modèle d'ordre $n_x$ ; $f$ et $g$ sont des fonctions non linéaires.

Il a été montré [LEV92] qu'un modèle entrée-sortie équivalent à ce modèle d'état est de la forme :

$$y_p(k) = h\,(y_p(k\text{-}1),\, ...,\, y_p(k\text{-}r),\, u(k\text{-}1),\, \ldots,\, u(k\text{-}r)) \tag{2}$$

avec $n_x \le r \le 2n_x+1$. Ainsi, les modèles entrée-sortie sont moins parcimonieux que les modèles d'état, en particulier parce qu'ils demandent un plus grand nombre d'arguments pour la fonction $h$ à estimer. Ceci sera illustré par la modélisation du comportement dynamique de l'actionneur d'un bras de robot au paragraphe III.5.

Dans le cas où un modèle de connaissance, même imparfait ou incomplet, est disponible, le concepteur a généralement intérêt à utiliser cette connaissance dès la conception d'un modèle d'état neuronal. Cette approche sera décrite plus en détail dans la partie IV.5 pour la modélisation d'une colonne à distiller.

### III.1.2. Définition de la forme prédicteur associée à un modèle-hypothèse

Dans tout ce qui suit, les perturbations mesurées sont considérées comme des entrées (notées $u$), et seules des perturbations non mesurées de type bruit sont envisagées.

La forme prédicteur associée à un modèle-hypothèse est le prédicteur qui fournit l'erreur de prédiction minimale si l'hypothèse est vraie.

- Dans le cas d'un modèle sans perturbation aléatoire, trouver la forme prédicteur consiste essentiellement à trouver les arguments de la fonction $f$ (modèle d'état (1)) ou $h$ (modèle entrée-sortie (2)) du modèle-hypothèse. La forme prédicteur fournit une erreur nulle si l'hypothèse est vraie.
- Pour les modèles avec perturbations aléatoires que nous allons présenter, la forme prédicteur dépend également de la manière dont les perturbations sont prises en considération par le modèle-hypothèse. La forme prédicteur est le prédicteur optimal au sens de la variance de l'erreur de prédiction, si l'hypothèse est vraie.

Nous présentons successivement les modèles entrée-sortie et les modèles d'état.

### III.2. MODÉLISATION ENTRÉE-SORTIE

Plusieurs modèles dynamiques non linéaires avec bruit ont été présentés dans [NER94] : les modèles NARX, NBSX, et NARMAX. Nous rappelons ces modèles, dans le cas mono-entrée mono-sortie.

Le ***modèle NARX*** (Non linéaire Auto-Régressif avec entrée eXogène) est le modèle non linéaire avec bruit qui conduit au prédicteur le plus simple. Dans le cas linéaire ARX, ce modèle est encore appelé "equation error" [LJU87]. Il s'écrit :

$$y_p(k) = h\,(y_p(k\text{-}1), ..., y_p(k\text{-}n), u(k\text{-}1), \ldots, u(k\text{-}m)) + w(k) \qquad (3)$$

où $\{w(k)\}$ est une séquence de variables aléatoires indépendantes à valeur moyenne nulle (bruit pseudo-blanc).

La forme prédicteur associée au modèle NARX est la suivante :

$$y(k\text{+}1) = h\,(y_p(k), ..., y_p(k\text{-}n\text{+}1), u(k), \ldots, u(k\text{-}m\text{+}1)) \qquad (4)$$

On vérifie aisément que, si l'hypothèse NARX est vraie, l'erreur de prédiction $e(k\text{+}1) = y_p(k\text{+}1) - y(k\text{+}1)$ est égale au bruit, par définition imprédictible. La variance de l'erreur est alors minimale.

Le système d'apprentissage doit utiliser un réseau prédicteur non bouclé de la forme :

$$y(k\text{+}1) = \varphi\,(y_p(k), ..., y_p(k\text{-}n\text{+}1), u(k), \ldots, u(k\text{-}m\text{+}1); C) \qquad (5)$$

où $\varphi$ est la fonction non linéaire réalisée par le réseau non bouclé muni des coefficients $C$. Si l'hypothèse NARX est vraie et si les autres conditions d'une bonne identification sont réunies (séquences d'apprentissage représentatives, algorithme adéquat et performant), alors la fonction $\varphi$ est une bonne approximation de $h$.

Le ***modèle NBSX*** (Non linéaire avec Bruit de Sortie additif et entrée eXogène) peut représenter un processus affecté d'un bruit de mesure additif non corrélé. Dans le cas linéaire, ce modèle est appelé "output error" [LJU87]. Il s'écrit de la manière suivante :

$$\begin{aligned} z_p(k) &= h\,(z_p(k\text{-}1), \ldots, z_p(k\text{-}n), u(k\text{-}1), \ldots, u(k\text{-}m)) \\ y_p(k) &= z_p(k) + w(k) \end{aligned} \qquad (6)$$



Le prédicteur associé est un prédicteur d'ordre $n$, dont l'état est constitué de la sortie et des $n$-1 valeurs précédentes de celle-ci :

$$y(k\text{+}1) = h\,(y(k), ..., y(k\text{-}n\text{+}1), u(k), \ldots, u(k\text{-}m\text{+}1)) \qquad (7)$$

Si l'hypothèse est vraie, l'erreur de prédiction est égale au bruit.

Le système d'apprentissage doit ici utiliser un réseau de neurones prédicteur d'ordre $n$ bouclé sur sa sortie :

$$y(k\text{+}1) = \varphi\,(y(k), ..., y(k\text{-}n\text{+}1), u(k), \ldots, u(k\text{-}m\text{+}1); C) \qquad (8)$$

où $\varphi$ est la fonction non linéaire réalisée par la partie non bouclée du réseau de coefficients $C$. Ici encore, si l'hypothèse est vraie, et si toutes les conditions d'une bonne identification sont réunies, alors la fonction $\varphi$ est une bonne approximation de $h$.

Dans le domaine du traitement du signal et de l'automatique, les modèles précédents sont souvent insuffisants pour décrire correctement les processus. Le modèle-hypothèse non linéaire le plus général que nous considérons est le ***modèle NARMAX*** (Non linéaire Auto-Régressif à Moyenne Ajustée avec entrée eXogène). Ce modèle est une extension au non linéaire du modèle linéaire ARMAX [LEO85] [CHE89] [CHE90a&b]. Il est de la forme :

$$y_p(k) = h\,(y_p(k\text{-}1), ..., y_p(k\text{-}n), u(k\text{-}1), \ldots, u(k\text{-}m), w(k\text{-}1), \ldots, w(k\text{-}p)) + w(k) \qquad (9)$$

Le prédicteur associé est d'ordre $p$, ses variables d'état étant les $p$ erreurs de prédiction passées :

$$y(k\text{+}1) = h\,(y_p(k), ..., y_p(k\text{-}n\text{+}1), u(k), \ldots, u(k\text{-}m\text{+}1), e(k), \ldots, e(k\text{-}p\text{+}1)) \qquad (10)$$

où $e(k) = y_p(k) - y(k)$. Son erreur de prédiction est égale au bruit. Le système d'apprentissage doit donc utiliser un réseau de neurones prédicteur bouclé d'ordre $p$ donc les variables d'état sont les p erreurs de prédiction passées :

$$y(k\text{+}1) = \varphi\,(y_p(k), ..., y_p(k\text{-}n\text{+}1), u(k), \ldots, u(k\text{-}m\text{+}1), e(k), \ldots, e(k\text{-}p\text{+}1); C) \qquad (11)$$

où $\varphi$ est la fonction non linéaire réalisée par la partie non bouclée du réseau de coefficients $C$. De même, si l'hypothèse est vraie, et si toutes les conditions d'une bonne identification sont remplies, alors la fonction $\varphi$ est une bonne approximation de $h$.

### III.3. MODÉLISATION D'ÉTAT

Les modèles neuronaux d'état sont utilisés, comme nous l'avons dit, soit pour une modélisation de type boîte noire, dans le cas où un modèle entrée-sortie semble insuffisamment performant, soit dans le cas où une étude physique conduit à envisager le modèle comme un ensemble de modules dont certains sont "physiques", et d'autres des réseaux de neurones.

Pour une représentation d'état donnée, nous distinguons les variables d'état mesurées et celles qui ne le sont pas, mais nous n'abordons pas le problème de l'observation des variables d'état non mesurées. La présentation est ici plus délicate que pour les modèles entrée-sortie, c'est pourquoi nous commençons par des modèles d'état sans perturbation aléatoire.

### III.3.1. Modélisation d'état sans perturbation aléatoire

Considérons le modèle-hypothèse suivant :

$$\begin{aligned} x_p(k) &= f\,(x_p(k\text{-}1),\, u(k\text{-}1)) \quad \text{(équation d'état)} \\ y_p(k) &= g\,(x_p(k)) \quad \text{(équation d'observation)} \end{aligned} \tag{12}$$

On déduit de ce modèle une forme prédicteur théorique (on ne connaît pas a priori les fonctions $f$ et $g$) :

$$\begin{aligned} x(k+1) &= f\,(x(k),\, u(k)) \\ y(k+1) &= g\,(x(k+1)) \end{aligned} \tag{13}$$

où $x$ est la prédiction de l'état $x_p$.

Le prédicteur neuronal bouclé d'état $\xi$ et de sortie $y$ correspondant est utilisé pour l'apprentissage :

$$\begin{aligned} \xi(k+1) &= \varphi\left(\xi(k),\, u(k);\, C_\varphi\right) \\ y(k+1) &= \psi\left(\xi(k+1);\, C_\psi\right) \end{aligned} \tag{14}$$

où $\varphi$ et $\psi$ sont des fonctions non linéaires réalisées par deux sous-réseaux en cascade de coefficients $C_\varphi$ et $C_\psi$. L'apprentissage est effectué à l'aide de la séquence d'entrée $\{u(k)\}$ et de celle de la sortie $\{y_p(k)\}$. Les valeurs désirées pour les $\{y(k+1)\}$ sont les sorties mesurées $\{y_p(k+1)\}$.

***Dans le cas général où l'état n'est pas mesuré***, l'état $\xi$ du prédicteur neuronal n'ayant pas de valeurs désirées, $\xi$ n'a donc pas de signification particulière. En effet, tout prédicteur neuronal dont l'état vérifie $\xi = \Phi(x_p)$, où $\Phi$ est inversible, et de la forme :

$$\begin{cases} \xi(k+1) = \tilde{f}\left(\xi(k),\, u(k)\right) \\ y(k+1) = \tilde{g}\left(\xi(k+1)\right) \end{cases} \quad \text{avec} \quad \begin{cases} \tilde{f}\left(\xi,u\right) = \Phi\left(f\left(\Phi^{-1}\left(\xi\right),u\right)\right) \\ \tilde{g}\left(.\right) = g\left(\Phi^{-1}(.)\right) \end{cases} \tag{15}$$

prédit sans erreur la sortie du processus. Il existe donc plusieurs formes prédicteur différentes, qui ont le même comportement entrée-sortie que le modèle-hypothèse, mais dont l'état ne coïncide généralement pas avec celui du modèle-hypothèse, et dont les fonctions $\varphi$ et $\psi$ ne sont pas des approximations des fonctions $f$ et $g$ du modèle-hypothèse. Néanmoins, l'utilisation du prédicteur comme modèle de simulation boîte noire se ramène à celle d'un modèle entrée-sortie ; elle peut cependant poser des problèmes à l'initialisation.

***Si l'état est complètement mesuré***, nous distinguons dans le vecteur de sortie $y_p$, l'état mesuré $x_p$ d'une part, et les autres variables de sortie $y_p$* d'autre part :

$$y_p(k) = \begin{bmatrix} x_p(k) \\ y_p^*(k) \end{bmatrix} = \begin{bmatrix} x_p(k) \\ g^*(x_p(k)) \end{bmatrix} \tag{16}$$

La situation est analogue au cas entrée-sortie : chaque composante de $x$ possède une valeur désirée. Il est donc possible d'estimer la fonction $f$ à l'aide soit d'un prédicteur neuronal de l'état non bouclé :



$$x(k+1) = \varphi\,(x_p(k), u(k);\, C) \tag{16}$$

soit d'un prédicteur bouclé :

$$x(k+1) = \varphi\,(x(k), u(k);\, C) \tag{17}$$

Un réseau non bouclé est utilisé pour estimer la fonction $g$* :

$$y^*(k) = \psi\,(x_p(k);\, C) \tag{18}$$

L'apprentissage est réalisé à l'aide de la séquence de commande $\{u(k)\}$ et de celle des sorties $\{y_p(k)\}$, qui comprennent l'état mesuré $\{x_p(k)\}$.

***Si l'état n'est pas complètement mesuré***, le concepteur a évidemment intérêt à utiliser les variables d'état mesurées comme valeurs désirées.

***Remarque :***

Pour de nombreuses applications, on peut désirer modéliser non seulement le comportement entrée-sortie du processus, mais aussi l'évolution de son état. Or, nous avons vu ci-dessus qu'en l'absence d'informations sur le processus, les fonctions $\varphi$ et $\psi$ obtenues par apprentissage ne sont des approximations des fonctions $f$ et $g$ du modèle-hypothèse que si toutes les variables d'état sont mesurées et utilisées comme valeurs désirées. Néanmoins, si des connaissances *a priori* permettent d'imposer la structure mathématique des fonctions $f$ et $g$ et des contraintes sur les variables d'état, il est possible de réaliser une estimation exploitable des fonctions $f$ et $g$ et donc de l'état, même sans la mesure complète de l'état. Ceci est illustré par la modélisation d'état d'une colonne à distiller dans la partie IV.5.

### III.3.2. Modélisation d'état avec perturbations aléatoires

Rappelons qu'en présence de perturbations aléatoires, la *forme prédicteur* associée au modèle-hypothèse est le prédicteur qui est optimal (i.e. qui fournit une erreur de prédiction à variance minimale) si l'hypothèse est vraie.

Soit un ***modèle-hypothèse d'état avec bruit de sortie additif*** :

$$\begin{aligned} x_p(k) &= f\,(x_p(k\text{-}1),\, u(k\text{-}1)) \\ y_p(k) &= g\,(x_p(k)) + w(k) \end{aligned} \tag{19}$$

On déduit de ce modèle une forme prédicteur théorique :

$$\begin{aligned} x(k+1) &= f\,(x(k),\, u(k)) \\ y(k+1) &= g\,(x(k+1)) \end{aligned} \tag{20}$$

Si l'hypothèse est vraie, l'erreur de prédiction est égale au bruit : ce prédicteur est bien optimal. Le prédicteur neuronal s'en déduit, de la même façon qu'en l'absence de perturbation aléatoire : il est de la forme (14). Comme précédemment, si certaines variables d'état sont mesurées, ces mesures doivent être utilisées comme valeurs désirées.

Pour tout modèle-hypothèse autre qu'avec bruit de sortie additif, les prédicteurs associés font intervenir l'état mesuré ; si l'état n'est pas mesuré, on ne peut qu'utiliser un prédicteur d'état neuronal de la forme (14), qui est donc sous-optimal.

***Remarque :***

Rappelons que les modèles d'état peuvent représenter une plus grande catégorie de systèmes dynamiques que les modèles entrée-sortie. Cet avantage est généralement déterminant, même si ces modèles sont sous-optimaux en présence de perturbations aléatoires autres qu'un bruit de sortie additif. Ceci est illustré au paragraphe III.5 avec la modélisation de l'actionneur d'un bras de robot.

## III.4. APPRENTISSAGE

Nous avons vu dans la partie I qu'une forme canonique d'un réseau de neurones est :

$$S(k+1) = \varphi\,(S(k),\, I(k);\, C)$$
$$O(k) = \psi\,(S(k),\, I(k);\, C) \qquad (21)$$

où $I(k)$ est le vecteur des entrées externes du réseau, $S(k)$ est le vecteur d'état, et $O(k)$ est le vecteur de sortie. Les fonctions $\varphi$ et $\psi$ sont réalisées par la partie non bouclée du réseau munie des coefficients $C$. Pour réaliser des prédicteurs d'état de la forme (14), on peut aussi bien utiliser un réseau composé de deux sous-réseaux distincts en cascade :

$$S(k+1) = \varphi\,(S(k),\, I(k);\, C_{\varphi})$$
$$O(k) = \psi'\,(S(k+1);\, C_{\psi'}) \qquad (22)$$

où $C_{\varphi}$ et $C_{\psi'}$ sont les coefficients des deux sous-réseaux.

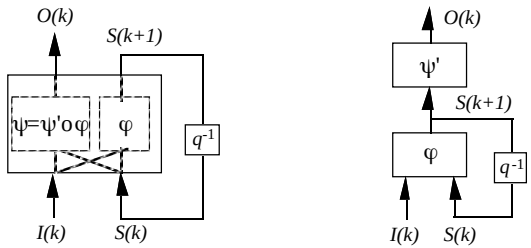


Figure III.1.
a) Formes canonique d'un réseau de neurones ; b) Forme équivalente.

*Exemple.*

Soit le modèle-hypothèse NARMAX (9) ; le prédicteur associé est donné par (10). Ceci impose l'utilisation d'un prédicteur neuronal de la forme (11) :

$$y(k+1) = \varphi\,(y_p(k), ..., y_p(k-n+1), u(k), \ldots, u(k-m+1), e(k), \ldots, e(k-p+1);\, C) \qquad (11)$$

où :



$$I(k) = [y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1)]$$
$$S(k) = [e(k), \ldots, e(k-p+1)] \qquad (23)$$
$$O(k) = y(k+1)$$

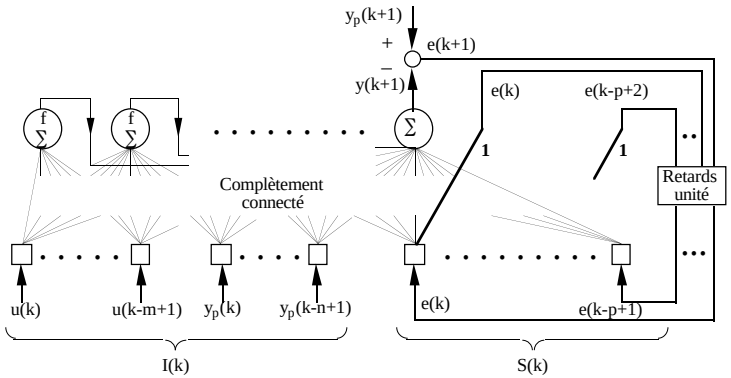


Figure III.2.
Exemple de réseau prédicteur NARMAX.

L'apprentissage du réseau est défini :

- par une séquence d'apprentissage $\{I(k)\}$ appliquée aux entrées externes, et une séquence de valeurs désirées correspondantes $\{D(k)\}$ pour les sorties du réseau ;
- par une fonction de coût, définie sur la fenêtre fixe de taille $N$ :

$$J(C, i) = \frac{1}{N}\sum_{k=1}^{N} E^i(k)^T W\, E^i(k) = \frac{1}{N}\sum_{k=1}^{N} \left(D(k) - O^i(k)\right)^T W \left(D(k) - O^i(k)\right) \qquad (24)$$

où $O^i(k)$ sont les sorties à l'instant $k$, calculées avec les coefficients disponibles à l'itération $i$ ; $E^i(k)$ est le vecteur des erreurs à l'instant $k$ et à l'itération $i$, et $D(k)$ est le vecteur des sorties désirées à l'instant $k$ ; $W$ est une matrice définie positive.

La fonction de coût est minimisée par une méthode itérative utilisant la valeur de la fonction de coût et celle de son gradient, cette dernière étant calculée par l'algorithme de rétropropagation. Les méthodes quasi-Newtoniennes à pas asservi sont particulièrement efficaces dans ce cadre.

L'algorithme de calcul de la fonction de coût et de son gradient est dit "dirigé" si le réseau est non bouclé, "semi-dirigé" sinon. Ceci signifie, dans le premier cas, que le réseau n'a pas d'état, et qu'il est donc entièrement "dirigé" par les entrées externes, et, dans le second cas, que l'état du réseau ne lui est imposé qu'au début de la fenêtre de la fonction d'apprentissage.

Pour tout apprentissage de prédicteur neuronal, si le modèle-hypothèse est vrai, si les séquences d'entrée d'apprentissage sont sensibilisantes, si la famille de fonctions réalisables par

le réseau est assez vaste (c'est-à-dire si le réseau possède suffisamment de neurones), et si l'algorithme d'apprentissage est efficace, alors le réseau prédicteur est une bonne réalisation du prédicteur optimal.

### III.5. MODÉLISATION DE L'ACTIONNEUR D'UN BRAS DE ROBOT

La position d'un bras de robot est commandée par un actionneur hydraulique. La position du bras dépend de la pression d'huile dans l'actionneur, pression commandée par l'ouverture d'une vanne. Les variations de l'ouverture de la vanne, c'est-à-dire la séquence de commande {u(k)}, et la pression d'huile correspondante, c'est-à-dire la séquence de sortie {$y_p$(k)}, sont montrées sur la figure III.3. Ce fichier de données[2] contient 1024 points de mesure : la première moitié d'entre eux est utilisée pour l'apprentissage, la seconde pour l'estimation de la performance (séquence de test).

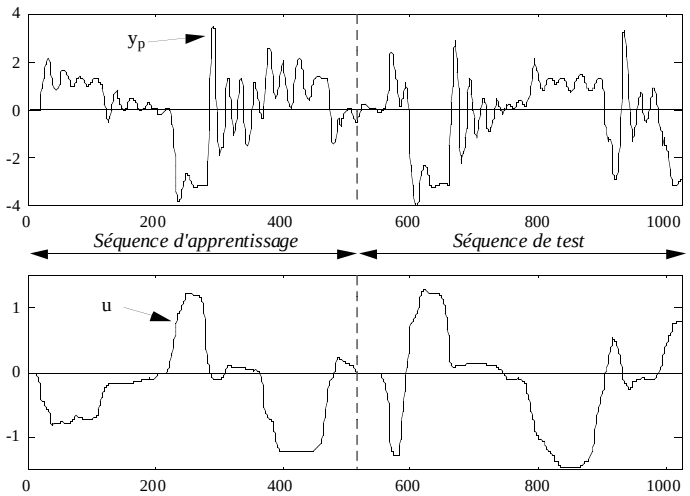


Figure III.3.
Séquences d'apprentissage et de test pour la modélisation d'un bras de robot.

L'examen des données montre que les séquences d'apprentissage et de test n'explorent qu'approximativement le même domaine de fonctionnement (signaux de sortie et de commande de même type et de même amplitude). On note qu'aux instants 600 et 850 environ de la séquence de test, l'amplitude de la commande dépasse les amplitudes maximales atteintes sur la

[2] Ces données proviennent de la Division of Oil Hydraulics and Pneumatics, Dept. of Mechanical Eng., Linköping University, et nous ont été aimablement communiquées par P.-Y. Glorennec.



séquence d'apprentissage. De plus, pour des entrées similaires, les oscillations du processus sont beaucoup plus entretenues sur la séquence d'apprentissage que sur la séquence de test.

Nous ne disposons d'aucune connaissance sur le processus : la modélisation est donc entièrement "boîte noire". Nous avons tout d'abord envisagé des modèles entrée-sortie, puis des modèles d'état.

#### III.5.1. Modélisation entrée-sortie de l'actionneur hydraulique du bras de robot

Sous l'hypothèse entrée-sortie NBSX, les meilleurs résultats sont obtenus pour un modèle du second ordre. Le prédicteur neuronal est de la forme :

$$y(k+1) = \varphi\,(y(k), y(k-1), u(k); C) \qquad (25)$$

avec trois neurones cachés. Il est représenté sur la figure III.4.

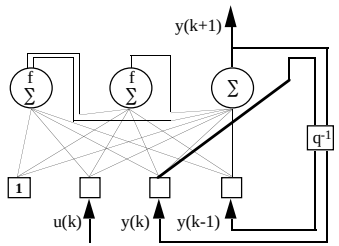


Figure III.4.
Prédicteur neuronal entrée-sortie pour l'actionneur hydraulique.

L'erreur quadratique moyenne obtenue à l'aide du prédicteur de la figure III.4 est de 0,081 sur la séquence d'apprentissage, et de 0,305 sur la séquence de test. L'ajout de neurones supplémentaires conduit à un surajustement (overfitting). Les résultats obtenus sur la séquence de test sont montrés sur la figure III.5. On peut observer que l'erreur n'est pas un bruit blanc.

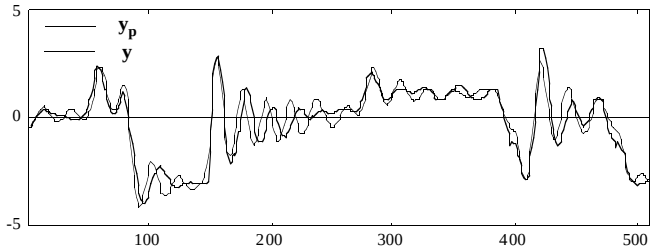


Figure III.5.
Modélisation entrée-sortie de l'actionneur hydraulique.

Sous l'hypothèse NARX, les prédicteurs obtenus donnent tous des résultats beaucoup moins satisfaisants s'ils sont testés bouclés (rappelons que la forme prédicteur associée à un modèle NARX est non bouclée). Ces prédicteurs ne peuvent donc pas être utilisés comme simulateurs.

### III.5.2. Modélisation d'état de l'actionneur hydraulique du bras de robot

Quelles que soient les hypothèses concernant les perturbations aléatoires, la seule possibilité est d'utiliser un modèle d'état bouclé (rappelons qu'il s'agit de modélisation boîte noire). Le prédicteur neuronal est de la forme :

$$\begin{cases} x_1(k+1) = \varphi_1(x_1(k),\, x_2(k),\, u(k);\, C) \\ x_2(k+1) = \varphi_2(x_1(k),\, x_2(k),\, u(k);\, C) \\ y(k+1) = \psi(x_1(k),\, x_2(k),\, u(k);\, C) \end{cases} \qquad (26)$$

avec deux neurones cachés. Il est représenté sur la figure III.6.

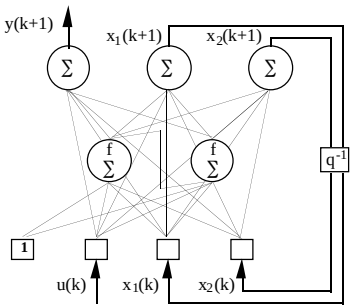


Figure III.6.
Prédicteur neuronal d'état pour l'actionneur hydraulique.

Au début des séquences d'apprentissage et de test, l'état du prédicteur est initialisé à zéro. L'erreur quadratique moyenne obtenue avec le prédicteur de la figure III.6 est de 0,103 sur la séquence d'apprentissage, et de 0,114 sur la séquence de test, ce qui constitue une bien meilleure performance que celle des prédicteurs entrée-sortie.

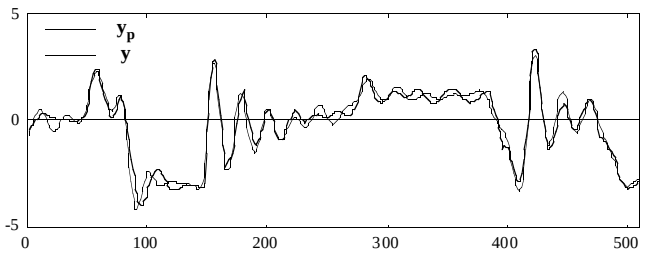


Figure III.7.
Modélisation d'état de l'actionneur hydraulique.



Ces résultats illustrent très nettement l'avantage de prédicteurs d'état pour la modélisation boîte noire lorsque les séquences d'apprentissage disponibles sont de petite taille : puisque qu'ils nécessitent moins d'arguments, ils nécessitent aussi moins de paramètres - ils sont plus parcimonieux, et sont donc moins enclins au surajustement. Les résultats obtenus sur la séquence de test sont représentés sur la figure III.7.

### III.5.3. Comparaisons

La comparaison de nos résultats avec ceux d'autres équipes est très favorable :

- [SJÖ94] réalise l'apprentissage dirigé d'un prédicteur NARX avec $n$=3, $m$=2 et 10 neurones cachés. Le test est effectué avec le prédicteur obtenu bouclé (modèle de simulation associé au prédicteur), et fournit une erreur quadratique moyenne de 0,585 sur la séquence de test.
- Une performance analogue est obtenue dans les mêmes conditions dans [BEN94] à l'aide d'un modèle à ondelettes.
- [SJÖ93] réalise une erreur quadratique moyenne de 0,465 sur la séquence de test à l'aide du modèle de simulation associé à un prédicteur NARX avec $n$=2, $m$=2 et 8 neurones cachés.

Les résultats que nous avons obtenus à l'aide d'un prédicteur d'état à deux neurones cachés, bien meilleurs que les précédents (erreur quadratique moyenne de 0,114 sur la séquence de test), démontrent donc l'intérêt de ces prédicteurs très généraux, ainsi que la faisabilité de leur apprentissage.

## III.6. CONCLUSION

Cette partie présente les principes de la modélisation neuronale. Outre la modélisation entrée-sortie, déjà décrite dans [NER94], et qui sera illustrée sur l'exemple d'un procédé de calcination dans la partie IV.4 suivante, nous avons exposé la modélisation à l'aide de modèles d'état neuronaux. En effet, l'intérêt de ce type de modèles est considérable, pour une modélisation boîte noire comme pour une modélisation physique.

Dans le cas de la modélisation boîte noire, l'intérêt de modèles d'état neuronaux par rapport à des modèles entrée-sortie est dû au fait qu'ils forment une classe plus vaste, et à ce qu'ils sont plus parcimonieux : (i) certains processus non linéaires ne sont pas modélisables globalement par des modèles entrée-sortie ; (ii) des modèles entrée-sortie nécessitent en général plus d'arguments et donc de paramètres que des prédicteurs d'état équivalents. Ceci a été illustré par l'exemple de la modélisation de l'actionneur d'un bras de robot.

Dans le cas où l'on peur réaliser une modélisation physique, qui conduit donc à un ensemble d'équations d'état, des modèles neuronaux d'état peuvent avantageusement être conçus en vue de prendre en considération ces connaissances a priori. Leur mise en œuvre se distingue de celle des modèles neuronaux entrée-sortie boîte noire par le fait que leur structure est partiellement

imposée : certaines variables d'état sont mesurées, et certaines des équations d'état sont éventuellement connues (fonctions fixées ou famille de fonctions dont les paramètres, physiques, sont inconnus). Les fonctions inconnues peuvent être réalisées par des modules neuronaux. Ces considérations sont développées dans la partie IV.5 avec la modélisation d'état d'une colonne à distiller.

Comme nous l'avons précisé au début de cette partie, une modélisation neuronale peut être effectuée en vue de la conception d'un système de commande d'un processus non linéaire, pour élaborer un correcteur, où encore pour simuler le processus au sein du système de commande, comme c'est le cas pour la commande avec modèle interne. Un exposé succinct de l'utilisation de modèles neuronaux pour la conception de systèmes de commande neuronaux fera l'objet de la partie V.

# IV. MODÉLISATION POUR LA SIMULATION DE PROCÉDÉS INDUSTRIELS

## IV.1. INTRODUCTION

La modélisation de procédés industriels pour la simulation connaît deux grandes classes d'applications : la détection d'anomalies, et l'aide à la conception ou à la conduite d'un processus. Nous développons le principe de la conception d'un modèle neuronal de connaissances, et nous illustrons cette partie par deux applications industrielles.

### IV.1.1. Modélisation pour la détection d'anomalies

Un processus industriel moderne est souvent très instrumenté ; il fournit en permanence les résultats de nombreuses mesures, apportant à son pilote une grande quantité d'informations, quasiment en temps réel, qui sont souvent difficiles à exploiter : c'est notamment le cas lorsqu'une anomalie de fonctionnement se traduit par de faibles écarts simultanés sur plusieurs mesures, voire par des évolutions normales pour chaque mesure prise séparément, mais dont l'apparition simultanée traduit cette anomalie.

Pour automatiser la détection de telles situations, plusieurs approches peuvent être envisagées :

- l'utilisation de systèmes experts (éventuellement flous) : une telle approche nécessite une formalisation (éventuellement incertaine) de l'expertise ; elle est difficilement applicable lorsqu'un grand nombre de grandeurs doivent être prises en considération ;
- l'utilisation de classifieurs (éventuellement neuronaux) capables de discriminer les situations normales de situations anormales ; la limitation fondamentale de cette approche réside dans le fait que ces méthodes statistiques exigent des bases d'exemples ; or, s'il est très facile de trouver de nombreux exemples de situations normales, il est souvent - heureusement - très difficile de disposer, en nombre équivalent, d'exemples des différentes situations anormales susceptibles de se présenter (on ne les connaît d'ailleurs pas forcément) ;
- l'utilisation d'un modèle précis du comportement normal du processus, les anomalies étant détectées en comparant les prédictions du modèle et les mesures effectuées sur le processus ; cette méthode nécessite évidemment un modèle précis en temps réel, mais elle évite de collecter des exemples de fonctionnement anormal. L'ensemble d'apprentissage doit couvrir l'ensemble du domaine de comportement normal du processus.



### IV.1.2. Modélisation pour l'aide à la conception ou à la formation

On peut également mettre en œuvre des réseaux de neurones pour réaliser des simulateurs destinés à l'aide à la conception de systèmes, ou pour aider à la formation de personnels de conduite de procédés. Dans ce cas, il n'est pas nécessaire que le simulateur soit extrêmement précis, mais il doit pouvoir simuler une grande variété de situations, même si celles-ci ne sont pas très vraisemblables ou fréquentes.

## IV.2. SYSTÈMES ADAPTATIFS

Comme nous l'avons indiqué dans la partie I, les réseaux de neurones peuvent être mis en œuvre au sein de systèmes adaptatifs, c'est-à-dire de systèmes qui apprennent en permanence. Cette propriété impose évidemment une charge de calcul en temps réel plus importante que celle de systèmes non adaptatifs, mais elle peut être indispensable dans les cas où le processus est l'objet d'évolutions lentes telles que l'encrassement d'un conduit, l'usure d'un palier, la dérive d'un capteur, etc. ; si l'évolution est suffisamment lente, une remise à jour régulière, mais non permanente, des coefficients du modèle peut suffire. Dans le cas d'une modélisation à base de connaissances, ces coefficients peuvent d'ailleurs être utilisés comme indicateurs de l'évolution d'un processus, à des fins de surveillance ; dans le cas d'une modélisation "boîte noire", il convient de prévoir un suivi de ces évolutions.

## IV.3. LES SYSTÈMES NEURONAUX DE CONNAISSANCE

Une croyance populaire veut que les réseaux de neurones soient des "boîtes noires", "incapables d'expliquer leur comportement". C'est généralement faux. Comme nous l'avons mentionné à plusieurs reprises, il est parfaitement possible, et même souhaitable, d'utiliser, dans la conception d'un réseau de neurones, la plus grande quantité possible d'informations *mathématiques* concernant le processus à modéliser. On peut ainsi réaliser de véritables "modèles neuronaux de connaissance".

Lorsque l'on cherche à modéliser un processus dynamique, plusieurs situations peuvent se présenter :

- aucune connaissance mathématique, suffisamment précise pour être exploitable, n'est disponible : on a alors nécessairement recours à une modélisation "boîte noire", qui doit être effectuée selon les principes exposés dans la partie III, et qui est illustrée par la modélisation de l'actionneur du bras de robot au paragraphe III.5.
- une analyse physique permet de déterminer les variables d'état d'un modèle, qui peut être exprimé sous la forme d'un ensemble d'équations d'état non linéaires, mais qui n'est pas suffisamment précis pour être utilisable en détection d'anomalies :

$$\frac{dx}{dt} = f(x(t), u(t))$$

L'imprécision de ce modèle peut être due à des approximations faites dans l'analyse physique, ou à une méconnaissance de certains phénomènes, qui se traduisent par des incertitudes importantes sur les paramètres du modèle. Néanmoins, ce modèle est suffisamment pertinent pour introduire des contraintes dans la conception du prédicteur neuronal ; l'apprentissage devrait donc fournir un modèle plus précis.

Après discrétisation de cette équation par la méthode d'Euler, les équations d'état deviennent :

$$x(k+1) = x(k) + f(x(k), u(k))$$

(d'autres méthodes que la méthode d'Euler peuvent être utilisées). Le réseau de neurones représenté sur la figure IV.1 (qui est sous forme canonique) obéirait à ces équations aux différences si le réseau de neurones statique réalisait la fonction *f*.

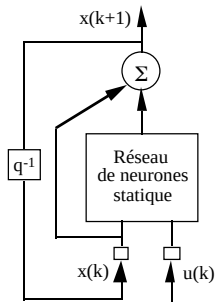


Figure IV.1.
Réalisation "neuronale" des équations d'état discrétisées.

Pour bénéficier de l'apprentissage, on peut organiser le réseau statique en deux types de sous-réseaux :

(i) les fonctions connues analytiquement, mais dont le calcul exact nécessite une quantité d'opérations incompatibles avec les exigences de fonctionnement en temps réel (par exemple, le calcul de la valeur de certaines composantes de *f* pour *x* et *u* donnés peut nécessiter la solution d'une équation algébrique ou différentielle) sont approchées par certaines parties de ce réseau après un apprentissage hors-ligne à partir de données *calculées*. Si nécessaire, un apprentissage final, à partir de données *mesurées*, peut affiner le résultat.

(ii) les comportements non connus analytiquement sont modélisés par des sous-réseaux "boîtes noires".



Notons que les fonctions connues qui sont suffisamment simples peuvent être incluses directement dans le réseau ; dans ce cas, le seul intérêt de la mise sous forme neuronale réside dans le fait qu'elle permet l'intégration de ces fonctions dans l'ensemble du modèle de manière homogène et facilement manipulable dans le cadre d'un modèle neuronal global.

Il faut noter que, si les équations différentielles constituant le modèle ne sont pas sous forme d'équations d'état, le réseau obtenu après discrétisation n'est pas sous forme canonique. Le réseau doit alors être mis sous forme canonique avant l'apprentissage.

L'un des intérêts majeurs du modèle de connaissance réside dans le fait qu'il résulte généralement d'une décomposition du problème global en sous-problèmes de plus petite taille ; cette décomposition se retrouve dans la structure du réseau de neurones, qui devient ainsi "lisible" . Cette lisibilité permet notamment de vérifier que chaque sous-réseau réalise bien la fonction qui lui est affectée, et de détecter, le cas échéant, des problèmes de non identifiabilité. Ainsi, l'apprentissage peut fournir les valeurs de paramètres physiques qui sont mal connus, ou qui présentent des déviations par rapport aux valeurs théoriques.

Le concepteur d'un modèle neuronal possède donc une gamme étendue de possibilités, qui lui permettent de bénéficier de connaissances acquises, tout en mettant à profit, de plusieurs manières, les capacités d'apprentissage des réseaux.

## IV.4. EXEMPLE DE MODÉLISATION "BOÎTE NOIRE" : UN PROCÉDÉ DE CALCINATION

Un procédé de calcination met en jeu des réactions chimiques entre solides, et entre solides et gaz. Ces réactions sont toujours difficilement maîtrisées, car de nombreux paramètres pertinents ne sont pas accessibles à la mesure. L'état de division du solide, la taille des grains, leurs formes, ou leur état de surface, par exemple, sont très difficiles à caractériser. Les connaissances théoriques sont à peu près inutilisables.

Dans le cadre d'une étude pour la Société Ciments Lafarge, une modélisation "boîte noire" a été effectuée. Les entrées retenues sont les grandeurs physiques reconnues pertinentes par les

experts du domaine. La sortie est représentative de la qualité du produit final. Les modèles théoriques établis par divers auteurs mettent en jeu des grandeurs qui ne sont pas mesurées ; elles font intervenir des coefficients dont les valeurs numériques publiées ont une dispersion de plusieurs ordres de grandeurs : il n'est donc pas possible d'utiliser ces connaissances pour modéliser le procédé.

Les mesures sont effectuées toutes les minutes, et le temps de réponse typique du processus est de l'ordre de 30 minutes. Les meilleurs résultats ont été obtenus avec un prédicteur entrée-sortie du premier ordre à six neurones cachés. Ce prédicteur a été mis en œuvre, après un apprentissage semi-dirigé, pour prévoir la sortie du processus sur divers horizons, en supposant les entrées constantes sur ces horizons. Un exemple de prévision à 10 minutes est représenté sur la figure IV.2, sur laquelle sont portées les mesures effectuées à l'instant *t*, et la prévision effectuée à l'instant *t* – 10 minutes.

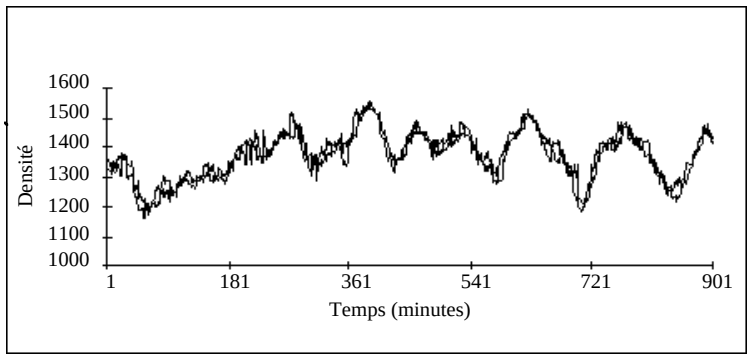


Figure IV.2.
Exemple de prévision à dix minutes de la sortie du procédé de calcination.

## IV.5. Exemple de modèle neuronal de connaissances : un procédé de distillation

Le processus de distillation a été très largement étudié, et il est bien connu. Il met en jeu des équilibres thermodynamiques entre liquide et vapeur dont l'analyse théorique est faite depuis longtemps. Les colonnes à distiller industrielles sont souvent très bien instrumentées, et le processus lui-même est peu bruité.

Dans le cadre d'une étude effectuée pour Elf Atochem, nous avons réalisé un simulateur, fonctionnant en temps réel sur PC 486, dont la précision est suffisante pour permettre la mise en évidence d'anomalies de fonctionnement. Dans cette application, nous avons utilisé la méthode décrite au paragraphe IV.3 pour effectuer la conception d'un modèle neuronal de connaissance,



qui comporte 8 entrées, 10 sorties, et 100 variables d'état, dont seulement 8 sont mesurées (de manière indirecte). Le profil de température dans la colonne est reconstitué par le réseau. Notons que, malgré le grand nombre de neurones utilisés (un millier), seuls 32 poids sont calculés par l'apprentissage, grâce à l'utilisation des connaissances physiques. Les résultats obtenus permettent de détecter des anomalies par comparaison entre les températures mesurées et celles qui sont prédites par le modèle. Un résultat typique est présenté ci-dessous.

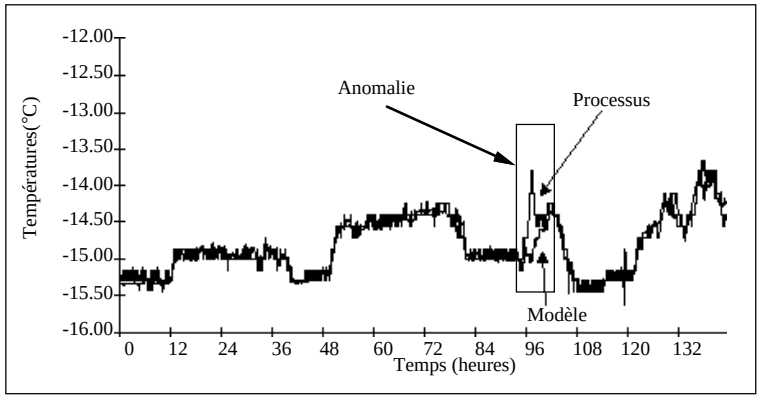


Figure IV.3.
Exemple de détection d'anomalies.

# V. MODÉLISATION POUR LA COMMANDE

La commande de processus par réseaux de neurones est abordée ici de façon succincte, afin de faire apparaître les aspects spécifiques de la modélisation neuronale pour la commande, ainsi que l'apport des réseaux de neurones pour une telle utilisation.

## V.1. Présentation

Commander un processus, c'est déterminer les commandes à lui appliquer, afin de lui assurer un comportement donné, en dépit de perturbations. Ces commandes sont délivrées par un organe de commande, qu'on élabore en plusieurs étapes.

Tout d'abord, un modèle du comportement du processus est nécessaire à la synthèse de l'organe de commande : un tel modèle sera appelé *modèle de commande*. Les qualités requises pour un modèle de commande ne sont évidemment pas les mêmes que pour un modèle de simulation ; on privilégiera notamment une structure de modèle assez simple pour faciliter la synthèse d'une commande, la rapidité du calcul, par rapport à la validité parfaite sur un très

large domaine de fonctionnement qui est nécessaire à un modèle de simulation (pour la détection d'anomalies par exemple). Il faut ensuite choisir l'architecture du correcteur (correcteur par retour d'état, PID…), en fonction du modèle et du cahier des charges, correcteur qui peut également être réalisé par un réseau de neurones.

Nous nous intéressons ici à des systèmes de commande *non adaptatifs*, c'est-à-dire des systèmes dont les paramètres sont calculés lors d'une phase de synthèse préalable à leur utilisation. L'apprentissage des correcteurs nécessaires aux systèmes de commande présentés est donc réalisé hors-ligne à l'aide du modèle de commande, comme le montre la figure V.1 : ceci caractérise les méthodes de commande *indirectes*.

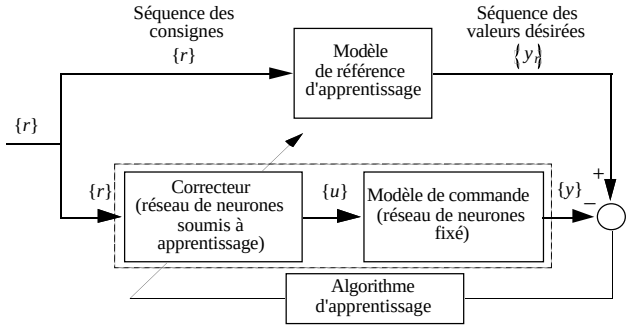


Figure V.1.
Système d'apprentissage d'un correcteur neuronal (non adaptatif indirect).

Dans ce cadre, le "réseau" pour lequel la tâche est définie (entrées externes, sorties désirées, et fonction de coût à minimiser) est donc composé du réseau correcteur dont les coefficients sont à estimer, et du modèle de commande dont les coefficients sont fixés. La séquence des entrées externes est constituée des consignes $\{r\}$ ; la séquence des sorties désirées pour le système {correcteur+modèle de commande} est constituée des sorties $\{y_r(k)\}$ d'un modèle de référence. La dynamique de ce dernier traduit les exigences du cahier des charges pour le comportement en boucle fermée du système de commande.

Enfin, certains systèmes de commande utilisent le modèle de commande lui-même : c'est le cas notamment des systèmes avec modèle interne, que nous décrivons au paragraphe V.2.2.

Comme nous l'avons montré dans la première partie, les réseaux de neurones se prêtent bien à la réalisation de modèles non linéaires, car ils peuvent approcher avec une précision arbitraire le comportement de n'importe quel système statique ou dynamique. Ce sont de plus des approximateurs parcimonieux. Ces propriétés ont deux conséquences :

- La possibilité de modéliser indifféremment des non-linéarités très diverses, qu'elles soient inhérentes au processus lui-même (non-linéarité de la cinématique d'un robot mobile, ou d'un moteur à explosion) ou à ses actionneurs (non-linéarités géométriques de type saturation, hystérésis…) ;
- La parcimonie d'un modèle neuronal de commande permet de réduire le temps de calcul lorsque le système de commande utilise ce modèle de commande (en temps réel), comme c'est le cas pour les systèmes avec modèle interne, ou de commande prédictive.



## V.2. QUELQUES SYSTÈMES DE COMMANDE

Nous illustrons ces avantages sur l'exemple de la commande d'un véhicule 4x4 Mercedes équipé des actionneurs et des capteurs nécessaires, le robot REMI de la SAGEM. Le problème de l'asservissement du véhicule sur une trajectoire de consigne avec un profil de vitesse imposé peut être résolu deux temps :

- l'asservissement de la position du véhicule sur la trajectoire indépendamment de sa vitesse à l'aide du volant, ce qui nécessite la modélisation du comportement latéral du véhicule : nous posons ce problème en termes de *régulation*.
- l'asservissement de la vitesse du véhicule sur le profil de vitesse imposé à l'aide des commandes des freins et de l'accélérateur, qui nécessite la modélisation du comportement longitudinal du véhicule : il s'agit d'un problème de *poursuite*.

### V.2.1. Régulation optimale neuronale

La régulation optimale avec coût quadratique à horizon infini se prête bien à l'utilisation de réseaux de neurones, dont l'apprentissage repose justement sur la minimisation d'une fonction de coût. En effet, le but de cette méthode est de trouver une loi de commande minimisant la fonction de coût J suivante :

$$J\big(x(0)\big) = \sum_{k=0}^{+\infty} x^T(k)\, Q\, x(k) + r\, u(k)^2$$

à partir d'un état initial $x(0)$ quelconque, où $Q$ est une matrice de pondération définie positive, et $r$ est un réel positif ou nul. L'intérêt des réseaux de neurones est de permettre facilement la minimisation de la fonction de coût pour n'importe quel modèle non linéaire du processus. Un système de régulation optimale utilisant un régulateur neuronal par retour d'état statique est représenté sur la figure V.2.

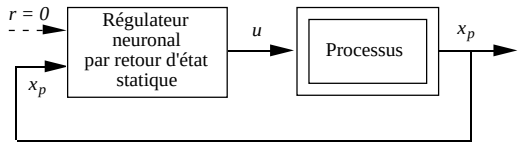


Figure V.2.
Système de régulation neuronal.

*Exemple.*

Un régulateur neuronal optimal de la position latérale et de l'orientation du véhicule REMI par rapport à la trajectoire de consigne a été mis au point [RIV93], à l'aide d'un modèle neuronal du comportement latéral du véhicule. Cette modélisation a tiré parti d'une analyse physique, et a notamment tenu compte des saturations en position et en vitesse angulaires de l'actionneur du volant. Le pilotage latéral était auparavant réalisé à la SAGEM avec des techniques linéaires : la prise en considération des non-linéarités principales a permis d'améliorer sensiblement les performances (erreur latérale maximale de 40 cm, pour des courbures jusqu'à 0,1 $m^{-1}$, et des dévers jusqu'à 30%, sur route et en tout-terrain).

Les performances du système de commande au cours d'un asservissement sur une trajectoire de consigne longue de 600 m (figure V.3) sont présentés sur la figure V.4 ; la période d'asservissement est de 0,05 secondes.

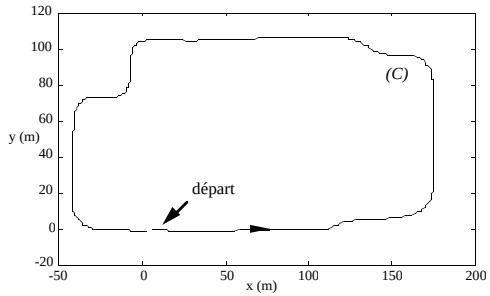


Figure V.3.
Trajectoire de consigne.



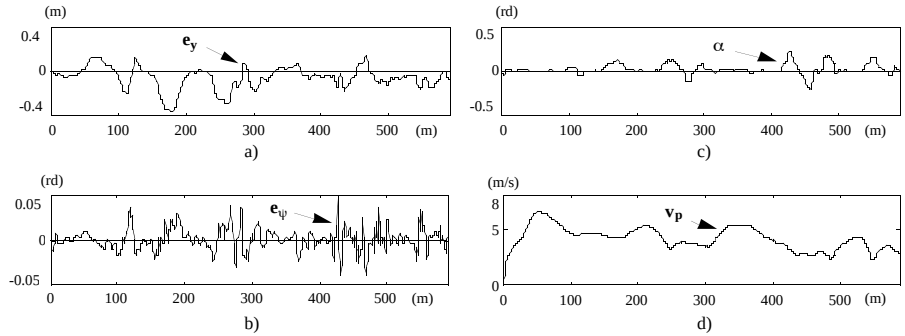


Figure V.4.
Asservissement latéral sur trajectoire : a) erreur latérale, b) erreur de cap, c) commande du volant, d) vitesse, en fonction de l'abscisse curviligne le long de la trajectoire de consigne.

### V.2.2. Asservissement de poursuite neuronal

La majeure partie des systèmes de commande neuronaux pour l'asservissement de poursuite appartient à la catégorie des systèmes dits *par simple bouclage*, dont un exemple est représenté sur la figure V.5. Dans ces systèmes, le signal de rétroaction est la sortie du processus (ou son état mesuré…).

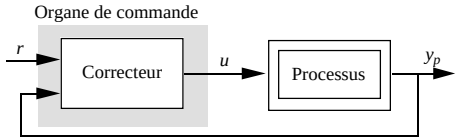


Figure V.5.
Système de commande par simple bouclage.

Dans les systèmes de commande *avec modèle interne* (qui est le modèle de commande utilisé pour l'apprentissage), le signal de rétroaction principal est la différence entre la sortie du modèle interne et celle du processus. L'intérêt de cette différence est qu'elle est représentative des défauts de modélisation et de l'effet de perturbations. Grâce à cela, la conception d'un système de commande neuronal *robuste* vis-à-vis de ces défauts et de perturbations est plus facile ainsi qu'en simple bouclage. De plus, l'utilisation de réseaux de neurones comme modèle interne peuvent contribuer à l'amélioration de ces systèmes de commande dans le cas de processus non linéaires.

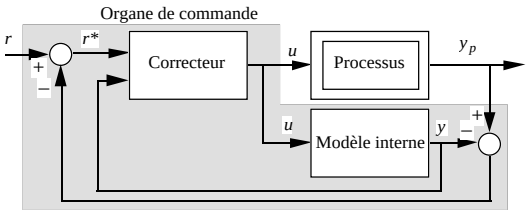


Figure V.6.
Système de commande avec modèle interne.

*Exemple.*

Le système avec modèle interne de la figure V.6 a été mis en œuvre pour l'asservissement de vitesse de REMI, après modélisation neuronale du comportement longitudinal de celui-ci, modélisation ici encore étayée par une analyse physique imposant partiellement l'architecture du réseau [RIV94]. Aucune technique linéaire ne s'était avérée satisfaisante pour ce problème : le système de commande neuronal a obtenu des performances comparables à celle du système également non linéaire de la SAGEM (compensation de l'effet de pentes jusqu'à 40%, sur route et en tout-terrain). Le système de commande neuronal a l'avantage d'être beaucoup plus compact que le précédent. Pour cette raison, le modèle neuronal est aussi préféré pour le test en laboratoire de systèmes de commande.

Nous présentons ci-dessous une simulation de la vitesse de REMI avec ce modèle (figure V.7). Le réseau est bouclé pendant toute la durée de la simulation qui est de 7 minutes environ ; la période d'échantillonnage est de 0,05 secondes.



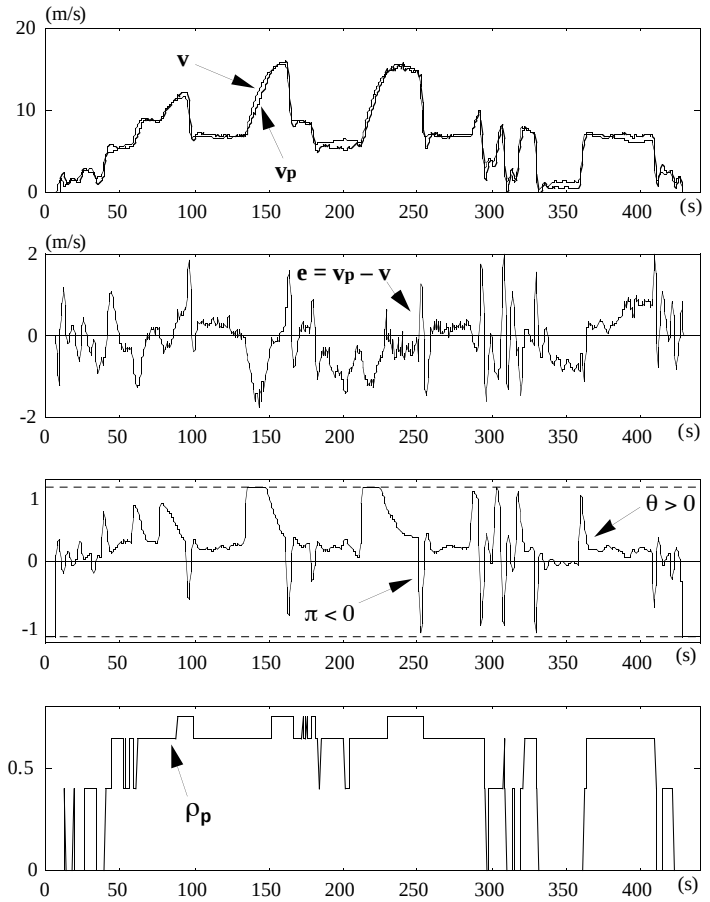


Figure V.7.
Modélisation du comportement longitudinal de REMI avec un modèle neuronal entrée-sortie du premier ordre. $v_p$ est la vitesse du processus, $v$ celle du modèle ; $\theta$ est la commande de l'angle papillon, $\pi$ la commande de la pression de freinage (grandeurs normalisées), et $\rho_p$ représente le rapport des vitesses (0 en 1ère ; 0,75 en 4ème).

## CONCLUSION

Dans cet article, nous avons, en premier lieu, procédé à une présentation des réseaux de neurones à partir des définitions et de la propriété fondamentale d'approximation universelle parcimonieuse ; à partir du moment où cette propriété est établie, la nature essentiellement *statistique* des réseaux de neurones apparaît très clairement. Garder constamment à l'esprit cette caractéristique fondamentale des réseaux de neurones, c'est se prémunir contre les biais introduits naturellement par les termes de "réseaux de neurones" et d'"apprentissage", qui

pourraient faire croire à des parentés entre les réseaux de neurones d'une part, et les méthodes et la problématique de l'Intelligence Artificielle (systèmes experts ou systèmes flous) d'autre part : une commande neuronale est beaucoup plus proche d'une commande PID que d'un système expert, ou beaucoup plus proche d'un filtre de Kalman que d'une commande floue. Nous avons insisté sur l'intérêt pratique de la parcimonie des réseaux de neurones, parcimonie qui est largement illustrée par les exemples d'applications présentées dans l'article. Nous avons également exposé, en termes très généraux, les principes de l'estimation des coefficients d'un réseau de neurones ou apprentissage, et avons montré le caractère générique des algorithmes d'apprentissage, qui s'appliquent indépendamment de l'architecture, aussi compliquée soit-elle, des réseaux utilisés.

Bien que ce ne fût pas là l'essentiel de notre propos, nous avons tenu a exposer brièvement le principe de fonctionnement des classifieurs neuronaux ; en effet, le lien entre la propriété d'approximation universelle et les propriétés de classification n'est pas toujours bien mis en évidence dans la littérature. Nous montrons notamment que, contrairement à une opinion très largement répandue, les réseaux de neurones sont intrinsèquement aptes à estimer la probabilité d'appartenance d'une forme inconnue, alors qu'ils sont souvent présentés comme capables seulement de déterminer des séparations "dures" entre les classes. Nous avons également montré les limitations des réseaux de neurones dans le domaine de la classification, et insisté sur le fait que les réseaux de neurones entrent dans la catégorie des meilleurs classifieurs, mais qu'ils n'ont aucune raison d'être systématiquement les meilleurs de cette catégorie.

Nous avons ensuite consacré une large place à la modélisation de processus dynamiques par réseaux de neurones. La démarche à adopter pour réussir une modélisation a été expliquée en détail ; nous avons particulièrement insisté sur l'intérêt des prédicteurs d'état par rapport aux prédicteurs entrée-sortie qui sont habituellement présentés dans la littérature, et nous l'avons illustré par le problème de la modélisation d'un bras de robot, qui a été abordé indépendamment par plusieurs équipes ; nous avons montré que la modélisation par réseaux de neurones est particulièrement parcimonieuse (deux neurones dans la couche cachée), et environ six fois plus précise que d'autres approches de ce même problème.

Nous avons expliqué la problématique et les principes de mise en œuvre de modèles neuronaux de procédés dans un contexte industriel. Nous avons illustré ce propos par deux exemples de modélisation de procédés industriels complexes : la modélisation "boîte noire" d'un procédé de calcination, et la modélisation par "modèle neuronal de connaissances" d'une colonne à distiller.

Nous avons également abordé, de manière plus brève, le problème de la modélisation pour la commande neuronale. Nous l'avons illustré par un exemple industriel : l'asservissement sur trajectoire d'un véhicule 4x4 autonome, par action sur le volant, l'accélérateur, et le frein.

L'objectif essentiel de cet article était de montrer que les réseaux de neurones reposent à présent sur des bases mathématiques solides qui permettent d'envisager des applications



industrielles à grande échelle, notamment dans le domaine de la modélisation. Si la résolution de problèmes difficiles nécessite toujours - et nécessitera encore très longtemps - beaucoup de travail et un éventail étendu de connaissances en statistiques, traitement du signal, automatique, etc., il n'est pas douteux que les réseaux de neurones peuvent alléger considérablement la tâche des ingénieurs en permettant une approche efficace et générique des problèmes non linéaires.

## REMERCIEMENTS

Les résultats présentés dans cet article n'auraient pû être obtenus sans les contributions décisives de Jean-Pierre Corriou (ENSIC Nancy), Michel de Cremiers et Daniel Canas (SAGEM), Marc Lavabre et Dieudonné Malanda (NETRAL).

## RÉFÉRENCES

[BEN94] Benveniste A., Juditsky A., Delyon B., Zhang Q.& Glorennec P.-Y. (1994) "Wavelets in identification", Preprints of the 10th IFAC Symposium on Identification, Copenhague, 4-6 juillet 1994.

[CHE89] Chen S. & Billings (1989) "Recursive prediction error parameter estimation for non linear models", Int. J. Control Vol.49 No.2, pp. 569-594.

[CHE90a] Chen S., Billings S. A. & Grant P. M. (1990) "Nonlinear system identification using neural networks", Int. J. Control Vol.51 No.6, pp. 1191-1214.

[CHE90b] Chen S., Cowan C.F.N., Billings S. A. & Grant P. M. (1990) "Parallel recursive prediction error algorithm for training neural networks", Int. J. Control Vol.51 No.6, pp. 1215-1228.

[HOR94] Hornik K., Stinchcombe M., White H. & Auer P. (1994) "Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives", Neural Computation Vol.6, pp. 1262-1275.

[LEO85] Leontaritis I. J. & Billings S. A. (1985) "Input-output parametric models for nonlinear systems. Part II : stochastic nonlinear systems", Int. J. Control Vol.41 No.2, pp. 329-344.

[LEO87] Leontaritis I. J. & Billings S.A. (1987) "Model selection and validation for nonlinear systems", Int. J. Control Vol.1, pp. 311-341.

[LEV92] Levin A. U. (1992) "Neural networks in dynamical systems; a system theoretic approach", PhD Thesis, Yale University.

[LJU87] Ljung L. (1987) System Identification; Theory for the User, Prentice Hall.

[NER93] Nerrand O., Roussel-Ragot P., Personnaz L. & Dreyfus G. (1993) “Neural networks and nonlinear adaptive filtering : unifying concepts and new algorithms”, Neural Computation Vol.5 No.2, pp. 165-199.

[NER94] Nerrand O., Roussel-Ragot P., Urbani D., Personnaz L. & Dreyfus G. (1994) “Training Recurrent Neural Networks: Why and How? An Illustration in Process Modeling”, IEEE Trans. Neural Networks 5, pp. 178-184.

[PLO94] Ploix J.-L., Dreyfus G., Corriou J.-P. & Pascal D. (1994) “From Knowledge-based Models to Recurrent Networks: an Application to an Industrial Distillation Process”, NeuroNîmes’94.

[PRI95] Price D., Knerr S., Personnaz L. & Dreyfus G. (1995) “Pairwise neural network classifiers with probabilistic outputs”, Neural Information Processing Systems, Morgan Kaufmann.

[RIV93] Rivals I., Personnaz L., Dreyfus G. & Canas D. (1993) “Real-time control of an autonomous vehicle : a neural network approach to the path-following problem”, 5th International Conference on Neural Networks and their Applications, pp. 219-229 (NeuroNîmes’93).

[RIV94] Rivals I., Canas D., Personnaz L. & Dreyfus G. (1994) “Modeling and control of mobile robots and intelligent vehicles by neural networks”, IEEE Conference on Intelligent Vehicles, 24-26 octobre 1994, Paris, pp. 137-142.

[RIV95] Rivals I. (1995) “Modélisation et commande de processus par réseaux de neurones ; application au pilotage d’un véhicule autonome”, Thèse de Doctorat de l’Université Paris 6.

[SJÖ93] Sjöberg J. (1993) “Regularization issues in neural network models of dynamical systems”, LiU-TEK-LIC-1993:08, ISBN 91-7871-072-3, ISSN 0280-7971.

[SJÖ94] Sjöberg J., Hjalmarsson H. & Ljung L. (1994) “Neural networks in system identification”, Report LiTH-ISY-R-1622.

[SON93] Sontag E. D. (1993) “Neural networks for control”, in [TRE93], pp. 339-380.

[TRE93] Trentelman H. L. & Willems J. C. eds. (1993) Essays on control : perspectives in the theory and its applications, Birkhäuser , Boston.

[URB94] Urbani D., Roussel-Ragot P., Personnaz L. & Dreyfus G. (1994) “The selection of neural models of nonlinear dynamical systems by statistical tests”, Neural Networks for Signal Processing, Proceedings of the 1994 IEEE Workshop, pp. 229-237.

[WHS92] White D. A. & Sofge D. A. (1992) Handbook of intelligent control : neural, fuzzy and adaptive approaches, Van Nostrand Reinhold, New York.