INTRODUCTION GÉNÉRALE

L'origine des nombreux travaux actuels sur les réseaux de neurones formels en Automatique peut être trouvée dans les travaux de Norbert Wiener, qui introduisit les méthodes statistiques dans les domaines de l'Automatique et des Télécommunications dès 1949, et dans ceux de Kalman et Bucy, qui combinèrent méthodes statistiques et représentation d'état. Néanmoins, si les concepts sont anciens, le véritable essor des réseaux de neurones en Automatique ne remonte qu'aux cinq dernières années. Cet essor a notamment été favorisé par les résultats des travaux effectués auparavant pour les applications des réseaux de neurones à la classification ; c'est ainsi que divers apports, qu'ils soient fondamentaux, tels que les théorèmes d'approximation universelle, ou plus techniques, tels que la rétropropagation, ont un impact direct sur les applications à l'Automatique. Néanmoins, cette dernière discipline a sa problématique et ses exigences propres, qui nécessitent et justifient des développements spécifiques.

Les réseaux de neurones, ensembles d'opérateurs non linéaires interconnectés, forment une famille de fonctions non linéaires, qui permet de construire, par apprentissage, une très large classe de modèles et de correcteurs. Néanmoins, pour la modélisation de processus, il n'est pas suffisant de disposer d'une famille de modèles très large ; encore faut-il déterminer les performances optimales réalisables par un modèle, compte tenu notamment de perturbations aléatoires non mesurables, afin de poser correctement le problème de l'apprentissage. De même, pour la commande, il est nécessaire de caractériser les propriétés des systèmes de commande en matière de stabilité et de performance, indépendamment de la réalisation éventuelle du correcteur par un réseau de neurones. Il s'agit là de questions qui n'avaient pas été abordées, ou seulement de façon très schématique, lorsque notre travail a débuté. D'autre part, il existe très peu d'applications opérationnelles des réseaux de neurones pour la commande de processus industriels. Nous avons tenté de combler ce double manque dans notre travail, qui comporte deux parties : une étude théorique consacrée à la modélisation et à la commande de processus non linéaires, et une étude appliquée, consacrée au pilotage de REMI - véhicule autonome tout-terrain conçu et réalisé par la société SAGEM - dans laquelle sont mis en œuvre les concepts et les méthodes introduits dans la première partie.

Sur le plan théorique, nous présentons la modélisation et la commande de processus par réseaux de neurones dans un cadre aussi général que possible, en les plaçant dans la perspective de l'Automatique classique, linéaire en particulier. Du point de vue de la modélisation, les résultats concernant les systèmes linéaires nous aident à formuler les prédicteurs non linéaires optimaux théoriques correspondant à diverses hypothèses sur le bruit intervenant dans le processus à

modéliser ; une méthodologie d'apprentissage spécifique fournit des prédicteurs neuronaux qui sont des réalisations des prédicteurs théoriques. Nous proposons ensuite une famille de systèmes de commande neuronaux, dont nous étudions les propriétés et les liens avec les systèmes de commande classique, linéaire ou non, en insistant notamment sur la robustesse ; ceci nous conduit à la commande avec modèle interne.

Sur le plan pratique, nous illustrons notre démarche par une application industrielle, le pilotage du véhicule REMI, dont volant, accélérateur et freins sont commandés par des réseaux neuronaux.

Les deux parties de ce mémoire sont organisées comme suit :

La première partie est consacrée à la présentation théorique de la modélisation et de la commande non adaptatives par réseaux de neurones, illustrée de nombreux exemples de mise en œuvre. Le chapitre 1 expose les propriétés mathématiques des réseaux de neurones, et les principes généraux de leur apprentissage pour la modélisation et la commande (les aspects pratiques de l'optimisation sont regroupés dans l'annexe I). Le chapitre 2 présente les différentes structures de modèles neuronaux que nous utilisons. Le chapitre 3 est consacré à la définition du système d'apprentissage associé à une structure de modèle donnée, et à la sélection d'un modèle parmi les modèles retenus. Le chapitre 4 montre la mise en œuvre des principes exposés pour la modélisation de processus non linéaires simulés, et pour celle de l'actionneur d'un bras de robot réel. Le chapitre 5 traite des systèmes de commande pour la régulation optimale de l'état d'un processus, ainsi que pour la poursuite de sa sortie. Nous y analysons les propriétés de ces derniers systèmes (une étude dans le cas linéaire est proposée en annexe II), puis nous introduisons les systèmes d'apprentissage pour la synthèse des correcteurs. Nous attachons une importance particulière aux propriétés de robustesse, et développons notamment la commande neuronale avec modèle interne. Enfin, le chapitre 6 illustre la mise en œuvre de l'ensemble des systèmes étudiés, à l'aide des processus simulés du chapitre 4.

La deuxième partie de ce mémoire est consacrée au pilotage du véhicule tout-terrain REMI. Le chapitre 7, qui concerne sa modélisation, met en relief l'exploitation conjuguée de connaissances physiques sur le fonctionnement du véhicule, et des capacités de modélisation " boîte noire " des réseaux de neurones. Le chapitre 8 décrit le pilotage du véhicule, qui met en jeu deux systèmes d'asservissement : un système d'asservissement sur trajectoire qui illustre la mise en œuvre de réseaux de neurones pour la régulation optimale, et un système d'asservissement de vitesse qui montre l'intérêt et la faisabilité de la réalisation de systèmes de commande neuronaux avec modèle interne. Enfin, nous présentons les résultats obtenus avec le véhicule sur terrain plat et non accidenté, ainsi qu'en tout-terrain. Pour des raisons de confidentialité, les chapitres 7 et 8 sont regroupés dans un volume séparé. Deux articles publiés, consacrés au pilotage de REMI, sont néanmoins reproduits dans les annexes III et IV du volume non confidentiel du mémoire.

Première partie :

MODÉLISATION ET COMMANDE DE PROCESSUS PAR RÉSEAUX DE NEURONES.

Chapitre 1

RÉSEAUX DE NEURONES POUR LA MODÉLISATION ET LA COMMANDE DE PROCESSUS

INTRODUCTION.

L'objet de ce mémoire est l'utilisation de réseaux de neurones pour la modélisation et la commande de processus. Les tâches auxquelles ces réseaux sont destinés sont donc essentiellement celles de prédicteurs ou de modèles de simulation des processus à commander, ainsi que celles de régulateurs ou de correcteurs. Nous nous plaçons dans le cadre de modèles à temps discret des processus, cadre qui se prête bien à la commande numérique d'une part, et à l'utilisation de réseaux de neurones formels d'autre part. Nous supposons que le comportement dynamique des processus auxquels nous nous intéressons peut être décrit par la classe de modèles dynamiques suivante :

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

où $k \in \mathbb{Z}$ représente l'instant discret t=kT, T étant le pas d'échantillonnage, $u(k) \in \mathbb{R}^{n_u}$ est le vecteur des entrées externes du modèle, $y_p(k) \in \mathbb{R}^{n_y}$ est le vecteur de ses sorties, et $x_p(k) \in \mathbb{R}^{n_x}$ est le vecteur de ses variables d'état, à l'instant k ; f et g sont des fonctions non linéaires.

Ces processus sont commandés à l'aide de processeurs numériques, pour lesquels nous cherchons à synthétiser des lois de commande de la forme très générale :

$$(2) \qquad u(k) = h\left(y_p(k), y_p(k\text{-}1), \ldots, x_p(k), x_p(k\text{-}1), \ldots, u(k\text{-}1), u(k\text{-}2), \ldots\right)$$

où h est une fonction non linéaire.

Un réseau de neurones est un système d'opérateurs non linéaires interconnectés, recevant des signaux de l'extérieur par ses entrées, et délivrant des signaux de sortie, qui sont les activités de certains neurones. Pour les applications considérées dans ce mémoire (modélisation et commande à temps discret de processus), ces signaux d'entrée et de sortie d'un réseau de neurones sont constitués de suites numériques. Un réseau de neurones est donc considéré comme un filtre non linéaire à temps discret. Nous distinguons deux types de réseaux (voir tableau 1) :

- les réseaux statiques ou non bouclés : ils réalisent des fonctions algébriques. Dans ce mémoire, nous les utilisons comme filtres transverses, prédicteurs non récursifs, ou correcteurs par retour d'état statique.
- les réseaux dynamiques ou bouclés : ce sont des systèmes dynamiques qui sont décrits par un jeu d'équations aux différences non linéaires couplées. Ces réseaux sont utilisés dans ce travail comme filtres récursifs, simulateurs ou prédicteurs récursifs, ou encore comme correcteurs par retour d'état dynamique.

Dans le cas général, un réseau de neurones est un modèle dynamique paramétré décrit par :

$$(3) \qquad \begin{cases} S(k+1) = \varphi_{RN}\left(S(k), I(k); C\right) \\ Y(k) = \psi_{RN}\left(S(k), I(k); C\right) \end{cases}$$

où $k \in \mathbb{Z}$ est l'instant discret, $I(k) \in \mathbb{R}^{NI}$ est le vecteur des entrées externes du modèle, $Y(k) \in \mathbb{R}^{NY}$ est le vecteur de ses sorties, et $S(k) \in \mathbb{R}^{NS}$ est le vecteur de ses variables d'état, à l'instant k.

$\varphi_{RN}(.,.; C) : \mathbb{R}^{NS+NI} \rightarrow \mathbb{R}^{NS}$, et $\psi_{RN}(.,.; C) : \mathbb{R}^{NS+NI} \rightarrow \mathbb{R}^{NY}$, représentent les fonctions réalisées par les neurones du réseau, interconnectés avec les coefficients C.

Le cas particulier d'un réseau non bouclé correspond à NS=0, simplement régi par :

$$(3') \qquad Y(k) = \psi_{RN}\left(I(k); C\right)$$

où $\psi_{RN}(.; C) : \mathbb{R}^{NI} \rightarrow \mathbb{R}^{NY}$, représente la fonction réalisée par les neurones du réseau, interconnectés avec les coefficients C.

Réseaux non bouclés	**Réseaux bouclés**
Filtres transverses	Filtres récursifs
Prédicteurs non récursifs	Prédicteurs récursifs
	Modèles de simulation
Correcteurs par retour d'état statique	Correcteurs par retour d'état dynamique

Tableau 1.
Tâches réalisées par des réseaux de neurones bouclés ou non[1] pour la modélisation et la commande de processus.

La famille de modèles paramétrés définie par (3) présente les attraits suivants :

- Il existe théoriquement toujours un réseau de neurones tel que les valeurs des fonctions φ_{RN} et ψ_{RN} approchent avec une précision fixée celles de fonctions continues dans un domaine donné de leurs arguments. Presque tout modèle de type (1) ou correcteur de type (2) peut donc être approché par un réseau de la famille (3), comme nous le verrons au §II de ce chapitre.
- Il est possible d'estimer les coefficients d'un réseau à l'aide de séquences de couples {entrées-sorties désirées}, disponibles ou déterminés par le concepteur à partir de la tâche à effectuer, de manière à satisfaire un indice de performance mesurant la ressemblance entre les sorties effectives du réseau et les sorties désirées. L'un des intérêts des réseaux de neurones réside dans le fait que cette estimation des coefficients est le résultat d'une procédure algorithmique, l'apprentissage, dont la mise en œuvre obéit à des règles indépendantes de l'architecture et de la complexité du réseau.

Dans le paragraphe qui suit, nous présentons les réseaux de neurones bouclés et non bouclés. Puis nous rappelons les résultats concernant l'approximation de fonctions ou de systèmes dynamiques par les réseaux de neurones. Enfin, nous introduisons la procédure algorithmique permettant d'estimer les coefficients d'un réseau destiné à une tâche donnée (voir également l'annexe I).

---

[1] Dans le présent mémoire, à propos de systèmes même non neuronaux, nous employons indifféremment les termes : statique, non récursif, *non bouclé*, et les termes : dynamique, récursif, *bouclé*.

I. LES RÉSEAUX DE NEURONES.

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 qui, à chaque instant discret k, calcule son potentiel $v_i(k)$ et son activité $z_i(k)$ de la façon suivante :

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

P_i est l'ensemble des indices des unités du réseau *p*ropageant leur activité au neurone i. Son potentiel $v_i(k)$ est une somme des valeurs 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, la fonction tangente hyperbolique ou une sigmoïde, une fonction à base radiale (RBF), ou encore la fonction identité.

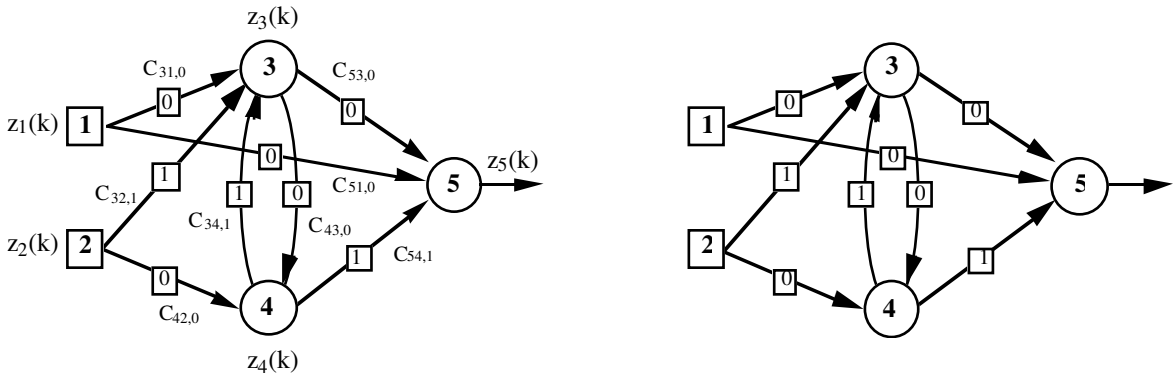

Figure 1.
Exemple de réseau de neurones et de son graphe.

Un réseau de neurones est conçu pour remplir une tâche que le concepteur définit par une séquence d'entrées, et par une séquence de valeurs désirées correspondantes pour les activités de certains neurones du réseau, les neurones de sortie. Les neurones qui ne sont pas des neurones de sortie sont dits cachés. Le réseau de la figure 1 possède deux entrées, deux neurones cachés, et un neurone de sortie.

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 [NER92b]. 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 apprentissage, mais certaines d'entre elles 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. LES RÉSEAUX DE NEURONES NON BOUCLÉS.

Un réseau est non bouclé, ou statique, si son graphe ne possède pas de cycle. Dans le contexte du traitement du signal et de l'automatique, il 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 :

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

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 *p*ropageant 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 ψ_{RN} paramétrée par les coefficients C du réseau (voir figure 2) :

$$Y(k) = \psi_{RN}\left(I(k);\, C\right)$$

où $Y(k) \in \mathbb{R}^{NY}$ 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 $I(k) \in \mathbb{R}^{NI}$ est le vecteur des entrées externes à l'instant k. $\psi_{RN}(.;\, C) : \mathbb{R}^{NI} \rightarrow \mathbb{R}^{NY}$, représente la fonction réalisée par les neurones du réseau interconnectés avec les coefficients C.

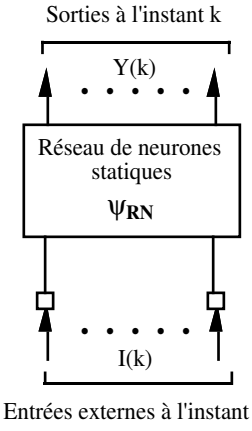

Figure 2.
Forme canonique d'un réseau de neurones non bouclé.

La figure 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 3a, et sa forme canonique sur la figure 3b. Cette forme canonique possède 7 entrées externes, 5 neurones cachés, et un neurone de sortie.

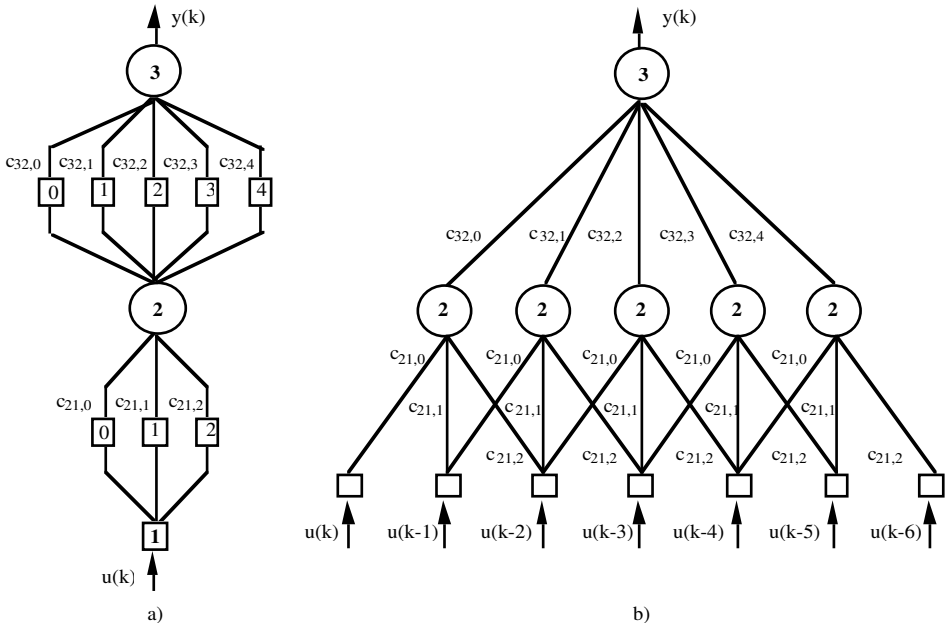

Figure 3.
Mise sous forme canonique d'un filtre transverse non linéaire à temps discret (TDNN).
$I(k) = \left(u(k), u(k-1), u(k-2), u(k-3), u(k-4), u(k-5), u(k-6)\right)^T$; $Y(k) = y(k)$.

L'architecture de réseau non bouclé la plus générale est celle du réseau complètement connecté (voir figure 4). Toutes les neurones cachés et les neurones de sortie sont connectés aux unités d'indice inférieur[2].

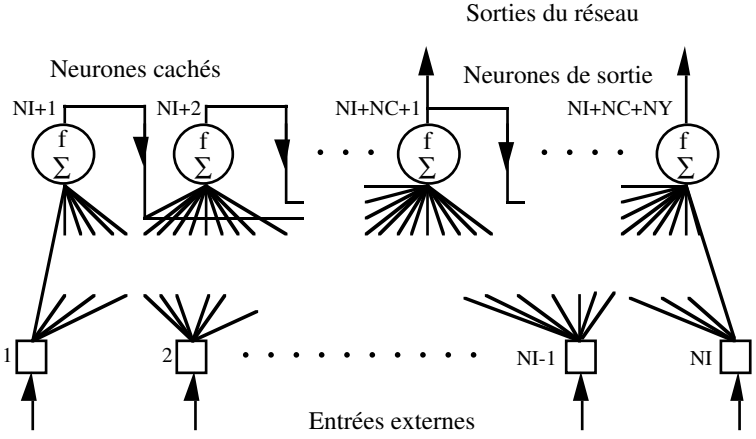

Figure 4.
Réseau de neurones non bouclé complètement connecté.

[2] Un réseau non bouclé réalise une tranformation entrée/sortie ; il n'est donc pas nécessaire que ses sorties soient fonctions les unes des autres. Dans ce travail, les neurones de sortie d'un réseau complètement connecté ne sont ainsi pas connectés entre eux.

Les entrées externes sont numérotées de 1 à NI, les neurones cachés de NI+1 à NI+NC, et les neurones de sortie de NI+NC+1 à NI+NC+NY.

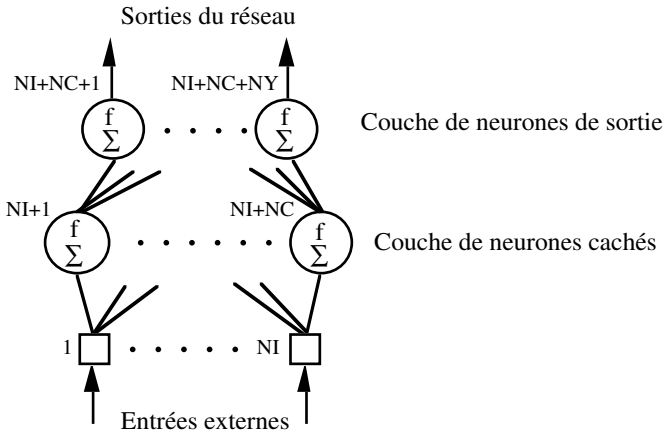

Figure 5.
Réseau de neurones non bouclé à une couche de neurones cachés.

Une architecture très utilisée, historiquement en raison surtout de sa pertinence en classification, est celle du réseau à couches (voir figure 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 §II).

I.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 NS peut être représenté par un réseau dont la dynamique est décrite par NS équations aux différences couplées d'ordre 1, mettant en jeu NS variables d'état, et NI 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 (voir figure 6) :

$$\begin{cases} S(k+1) = \varphi_{RN}\left(S(k), I(k); C\right) \\ Y(k) = \psi_{RN}\left(S(k), I(k); C\right) \end{cases}$$

où $I(k) \in \mathbb{R}^{NI}$ est le vecteur des entrées externes, $S(k) \in \mathbb{R}^{NS}$ le vecteur des variables d'état, $Y(k) \in \mathbb{R}^{NY}$ le vecteur des sorties, à l'instant k, et $S(k+1) \in \mathbb{R}^{NS}$ est le vecteur des variables d'état à l'instant k+1. φ_{RN} (., .; C) et ψ_{RN} (., .; C) représentent les fonctions réalisées par le réseau de neurones statiques de la forme canonique interconnectés avec les coefficients C.

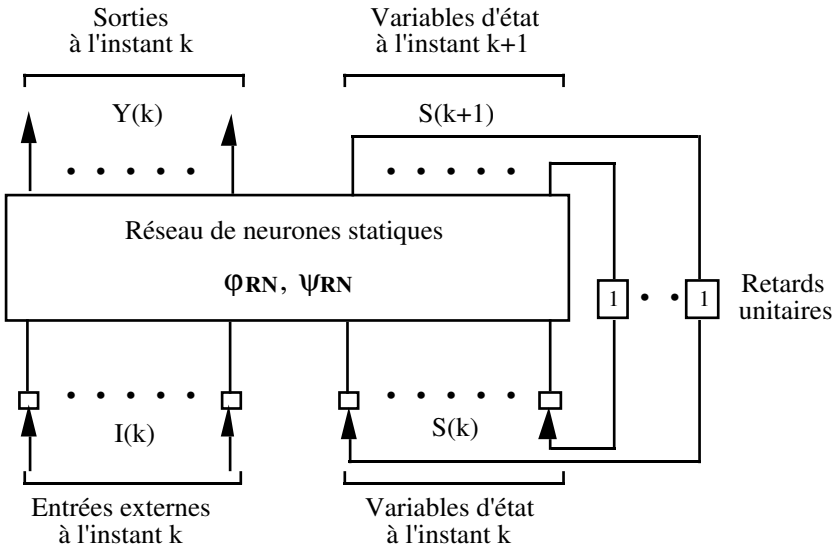

Figure 6.
Forme canonique d'un réseau de neurones bouclé.

Une forme canonique d'un réseau de neurones bouclé est ainsi définie à partir d'un réseau non bouclé constitué de neurones statiques possédant NI entrées externes, NS entrées d'état (les variables d'état à l'instant k), NC neurones cachés et NY 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 NY neurones de sortie, et les variables d'état à l'instant k+1 sont les activités de NS neurones que nous appelons neurones d'état[3]. Ces neurones d'état sont soit des neurones cachés, soit des neurones de sortie.

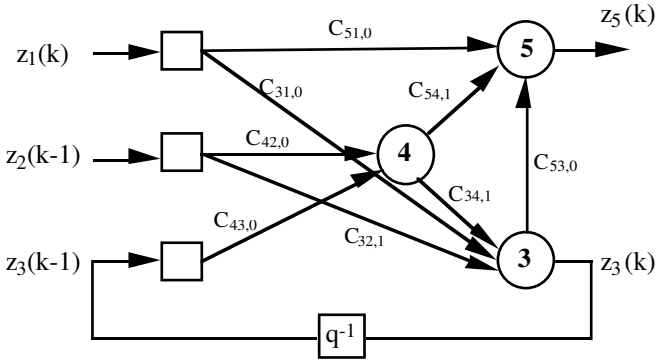

Figure 7.
Mise sous forme canonique du réseau de la figure 1.
$I(k) = \left(z_1(k), z_2(k-1)\right)^T$; $S(k) = z_3(k-1)$; $Y(k) = z_5(k)$.

Par exemple, mettons le réseau de la figure 1, bouclé comme l'indique son graphe, sous une forme canonique (voir figure 7). Ce réseau possède deux entrées externes (NI=2) $z_1(k)$ et $z_2(k-1)$, une entrée d'état (NS=1) $z_3(k-1)$, deux neurones cachés (NC=2), dont un neurone d'état (NS=1)

---

[3] Dans le présent mémoire, comme pour les réseaux non bouclés, les neurones de sortie ne sont pas reliés entre eux. Les neurones d'état ne le sont pas non plus (voir la note 2, ainsi que les exemples de réseaux des chapitres 3 et 4).

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

II. PROPRIÉTÉS D'APPROXIMATION DES RÉSEAUX DE NEURONES.

Indépendamment de tout problème d'apprentissage, la question que nous nous posons dans ce paragraphe est la suivante : quelles fonctions, ou quels systèmes dynamiques, peuvent être réalisés par les réseaux de neurones non bouclés et bouclés ?

II.1. RÉSEAUX NON BOUCLÉS.

Les résultats qui présentent un intérêt pour la modélisation et la commande de processus sont ceux qui concernent l'approximation de fonctions à valeurs continues. Nous laissons donc de côté la possibilité de réaliser des fonctions booléennes à l'aide de réseaux de neurones, démontrée anciennement par McCulloch et Pitts [MCU43], ainsi que celle de réaliser une frontière de séparation, solution d'un problème de classification.

Les travaux de Cybenko [CYB89] et Funahashi [FUN89] ont prouvé la possibilité d'approcher des fonctions continues, au sens de la norme uniforme sur les compacts, par des réseaux de neurones. Les réseaux considérés sont de type réseau à une couche de neurones cachés à fonction d'activation non linéaire, et à neurones de sortie linéaires. Dans le cas du théorème de Cybenko, l'hypothèse sur la fonction d'activation est qu'elle a pour limite 0 en $-\infty$ et 1 en $+\infty$, dans celui de Funahashi, qu'elle est croissante, non constante et bornée. Les fonctions non continues, mais mesurables, peuvent aussi être approchées, mais dans un sens moins fort [HOR90]. Il existe par ailleurs quelques résultats sur le nombre de neurones requis pour approcher une fonction avec une précision fixée, pour certaines classes de fonctions [BAR93] [SON93].

Ces résultats affirment donc, pour toute fonction déterministe usuelle, l'existence d'une approximation par un réseau de neurones. Les réseaux complètement connectés ou à couches, et à neurones cachés sigmoïdaux, remplissent les conditions d'application des théorèmes. Dans ce travail, nous utilisons systématiquement ce type de réseaux, à l'exclusion par exemple des réseaux utilisant des fonctions à base radiale (RBF). Une raison en est que, même si ces réseaux jouissent également de propriétés d'approximation intéressantes, et même si leur apprentissage peut être réalisé à l'aide de la méthode des moindres-carrés ordinaires (MCO), leur utilisation est souvent beaucoup moins économique, ou " parcimonieuse " du point de vue du nombre de connexions, que celle des réseaux à sigmoïdes. En toute rigueur, les réseaux à RBF peuvent être aussi parcimonieux que les réseaux à sigmoïdes, mais à condition d'ajuster la position des centres des RBF de manière non linéaire [SON93], ce qui supprime le principal intérêt des RBF : la simplicité de l'apprentissage par la méthode des MCO.

II.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 un processus représenté par un modèle de type (1) :

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

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 Y(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'applications des résultats ci-dessus étant remplies. Mais ceci n'est pas nécessaire pour que le réseau reproduise le comportement entrée-sortie du processus (voir chapitre 2 §I.2.2.2), ni pertinent pour l'approximation d'un système dynamique. En effet, une propriété d'approximation universelle pour les réseaux bouclés peut s'énoncer de la manière suivante : pour tout système dynamique (1), pour toute précision désirée ε, pour un intervalle de temps fini [0;T], pour des entrées u(.) : $[0,T]\rightarrow\mathbb{U}\subseteq\mathbb{R}^{n_u}$ et un état initial $x(0)\in\mathbb{X}\subseteq R^{n_x}$, il existe un réseau de neurones bouclé qui approche le comportement entrée-sortie du système (1) avec la précision ε sur l'intervalle [0;T] et sur les ensembles $\mathbb{U}$ et $\mathbb{X}$ [SON93]. La définition de l'approximation d'un système dynamique par un réseau de neurones bouclé n'est donc pas globale, mais restreinte à un domaine des espaces d'état et d'entrée, sur 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.

III. APPRENTISSAGE DES RÉSEAUX DE NEURONES.

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 semble raisonnable de supposer que les sorties mesurées sur le processus obéissent à des lois déterministes, de type (1) par exemple, 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 au chapitre 2 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 le système d'apprentissage 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 ces fonctions. La mise au point d'un tel système d'apprentissage en fonction des hypothèses faites sur les lois déterministes et les perturbations aléatoires affectant un processus fait l'objet des chapitres 2 et 3 de ce mémoire.

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 sont abordés au chapitre 5.

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.

III.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 (comme nous l'avons précisé au paragraphe précédent, nous utilisons dans ce travail des réseaux à neurones cachés à fonction d'activation tangente hyperbolique, complètement connectés). 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 [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.

Tâche d'un réseau de neurones.

Comme mentionné plus haut, la tâche du réseau est définie par :

- deux séquences de nombres, une séquence appliquée aux entrées externes {I(k)}, et une séquence de valeurs désirées correspondantes {D(k)} pour les sorties. Ces séquences constituent *les séquences d'apprentissage*.
- 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, voir chapitre 5).

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.

Exemples.

a) Apprentissage du prédicteur associé à un processus mono-entrée/mono-sortie :

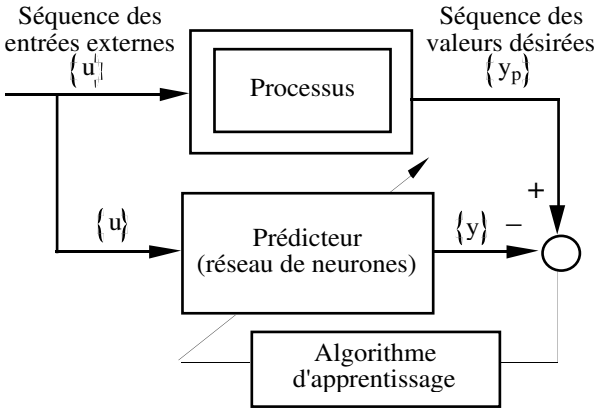

Figure 8.
Système d'apprentissage pour la modélisation d'un processus.

La séquence des entrées externes est constituée des commandes {u(k)} appliquées au processus, et la séquence des sorties désirées des sorties $\{y_p(k)\}$ mesurées sur le processus. La figure 8 représente le schéma-bloc d'un système d'apprentissage pour la modélisation du processus : le but est d'estimer les coefficients du réseau prédicteur de façon que ses sorties soient " aussi proches que possible " de celles du processus.

b) Apprentissage du correcteur d'un processus par une méthode de commande indirecte.

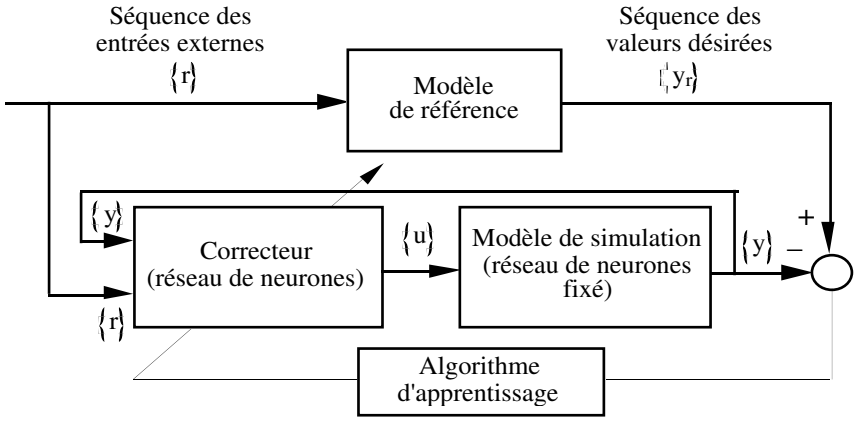

Figure 9.
Système d'apprentissage pour la commande d'un processus.

On dispose d'un modèle de simulation du processus, par exemple un réseau de neurones. La séquence des entrées externes est constituée des consignes {r(k)} (r(k) = cte ∀k s'il s'agit de régulation). La séquence des sorties désirées pour le système {correcteur + modèle de simulation} est la séquence des sorties $\{y_r(k)\}$ d'un modèle de référence dont la dynamique traduit les exigences du cahier des charges sur le comportement en boucle fermée du système de commande, c'est-à-dire du processus réel avec son organe de commande. La figure 9 suivante représente le schéma-bloc du système d'apprentissage.

Notons que dans le cadre de la commande indirecte, c'est-à-dire utilisant pour l'apprentissage un modèle de simulation du processus, le " réseau " pour lequel la tâche est définie (entrées externes, sorties désirées, et fonction de coût à minimiser) est composé du réseau dont les coefficients sont à estimer, le correcteur, et d'un réseau dont les coefficients sont fixés, le modèle de simulation.

III.2. EXPRESSION DE LA FONCTION DE COÛT.

Dans ce mémoire, nous nous intéressons à la mise au point de systèmes, modèles ou correcteurs, *non adaptatifs*[4] : la phase d'apprentissage et la phase d'utilisation des réseaux considérés sont distinctes. Ainsi, un correcteur appris hors-ligne à l'aide d'un modèle de simulation du processus à commander ne subira plus de modifications de ses coefficients pendant son utilisation avec le processus. Dans ce cadre non adaptatif, les séquences d'apprentissage sont de taille finie, disponibles dans une base de données. La fonction de coût à minimiser porte donc sur un nombre fini d'instants : elle est en général fonction croissante des écarts entre les sorties du réseau et les sorties désirées correspondantes.

Pour cette présentation, 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 {Y(k)} et les sorties désirées {D(k)} ; cette erreur est définie sur une fenêtre temporelle dont la taille est égale à la taille N des séquences d'apprentissage. La minimisation de cette fonction de coût est effectuée itérativement en modifiant les coefficients à chaque présentation de la séquence : les erreurs {E(k)} sont calculées à l'aide du réseau muni des coefficients disponibles à la fin de l'itération précédente, et des séquences d'apprentissage. Un tel algorithme d'apprentissage est *itératif*, et *non récursif*.

On peut éventuellement réaliser la minimisation en utilisant un algorithme *récursif*. Dans ce cas, à chaque instant k de la fenêtre totale de taille N, on considère une fonction auxiliaire, dite fonction d'apprentissage, définie sur une fenêtre glissante de taille $N_c << N$ correspondant aux instants passés (de $k-N_c+1$ à k), et l'on modifie les coefficients à chaque instant k afin de diminuer, en moyenne, la fonction de coût. Si plusieurs itérations sont effectuées à chaque instant k, l'algorithme est *récursif* et *itératif*. Lorsque la séquence d'apprentissage a été parcourue, on replace la fenêtre glissante à son début. L'utilisation d'un algorithme récursif permet aussi de minimiser la fonction de coût, mais ne peut garantir la convergence des coefficients qu'en moyenne. *Ces algorithmes ne*

---

[4] Pour une présentation de l'apprentissage de systèmes adaptatifs entrée-sortie, voir [NER92a].

présentent de véritable intérêt que pour les systèmes adaptatifs, en particulier en traitement du signal. Ce chapitre (ainsi que les suivants) ayant pour objet l'apprentissage de systèmes non adapatifs, nous présentons uniquement des algorithmes *non récursifs* et *itératifs*.

L'expression de la fonction de coût à l'itération i sur une fenêtre fixe englobant toute la longueur N de la séquence d'apprentissage est la suivante :

$$J(C, i) = \frac{1}{2} \sum_{k=1}^{N} E^i(k)^T W E^i(k) = \frac{1}{2} \sum_{k=1}^{N} \left(D(k) - Y^i(k)\right)^T W \left(D(k) - Y^i(k)\right)$$

où C représente les coefficients du réseau C(i-1) disponibles à l'itération i, $E^i(k)$ le vecteur des erreurs à 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 à l'instant k, et $Y^i(k)$ le vecteur des sorties du réseau à l'instant k et à l'itération i.

III.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. Nous présentons dans l'annexe I un algorithme général de calcul du gradient de la fonction de coût dans le cas de l'utilisation de réseaux de neurones, gradient qui est utilisé soit comme direction de descente, soit pour calculer une direction de descente permettant une convergence plus rapide (méthode quasi newtonienne par exemple). L'algorithme est présenté pour tout réseau de neurones bouclé de la forme générale du §I.2, dont le réseau non bouclé est un cas particulier. La présentation est élargie aux réseaux dont l'état n'est pas constitué exclusivement des valeurs retardées des sorties (dits " réseaux d'état ", voir les notations du chapitre 2 §I.1).

Nous avons signalé que, dans le cas de l'utilisation de méthodes de commande indirecte, l'apprentissage d'un correcteur différait de celui d'un modèle prédictif : le système à considérer pour l'apprentissage est constitué non du seul réseau, mais du système global constitué du correcteur et du modèle de simulation, puisque les valeurs désirées concernent la sortie du modèle. L'algorithme de calcul du gradient que nous proposons en annexe reste bien sûr applicable dans ce cas. En effet, nous verrons qu'il repose sur le calcul des dérivées de la fonction de coût par rapport aux sorties de tous les neurones. Or, le calcul des dérivées par rapport aux sorties des neurones du réseau modèle fournit le Jacobien de celui-ci, nécessaire à l'évaluation du gradient par rapport aux coefficients du correcteur. Les dérivées par rapport aux sorties des neurones du réseau correcteur sont ensuite utilisées pour calculer le gradient par rapport à ses coefficients, et pour effectuer les modifications de ces coefficients. Le principe de l'apprentissage d'un correcteur avec un modèle neuronal du processus est généralisable à tout modèle, neuronal ou non, soit en mettant le modèle sous forme de réseau avec les coefficients et les fonctions d'activations appropriées, soit en

calculant directement le Jacobien du modèle. Les modalités de l'apprentissage d'un correcteur sont amplement développées dans le chapitre 5 consacré à la commande.

CONCLUSION.

Dans ce chapitre introductif, complété par l'annexe I pour les aspects pratiques de l'optimisation, nous avons présenté les modèles dynamiques universels que sont les réseaux de neurones, et le cadre général de leur apprentissage, élargi aux réseaux (dits " réseaux d'état ", cf chapitre 2 §I.1) dont l'état n'est pas constitué exclusivement des valeurs retardées des sorties.

L'utilisation des réseaux de neurones pour la modélisation de processus est développée dans les chapitres 2 à 4, et elle est appliquée à un problème industriel dans la deuxième partie de ce mémoire, avec la modélisation du véhicule autonome REMI (chapitre 7). La commande neuronale de processus est présentée dans les chapitres 5 et 6, et, dans la deuxième partie, les systèmes de commande préconisés sont appliqués au pilotage de REMI (chapitre 8).