Titre: Title:	Identification et réglage par réseaux neuronaux artificiels des systèmes dynamiques non-linéaires
Auteurs: Authors:	Hadi Kanaan
Date:	1999
Type:	Rapport / Report
Référence: Citation:	Kanaan, H. (1999). Identification et réglage par réseaux neuronaux artificiels des systèmes dynamiques non-linéaires (Rapport technique n° EPM-RT-99-10). https://publications.polymtl.ca/9888/

Institution:	École Polytechnique de Montréal
Numéro de rapport: Report number:	EPM-RT-99-10
URL officiel: Official URL:	
Mention légale: Legal notice:	

Département de Génie Électrique
et de Génie Informatique

Section Électronique et Énergie

IDENTIFICATION ET RÉGLAGE PAR RÉSEAUX NEURONAUX ARTIFICIELS DES SYSTÈMES DYNAMIQUES NON-LINÉAIRES

Par

Hadi KANAAN

Rapport technique EPM/RT-99/10

dirigé par

Professeur Gilles Roy
Professeur Xuan-Dai Do
Professeur Jean-Charles Bernard

École Polytechnique de Montréal
Mai 1999

Dépôt légal, Mai 1999
Bibliothèque nationale du Québec
Bibliothèque nationale du Canada

Identification et réglage par réseaux neuronaux artificiels
des systèmes dynamiques non-linéaires

Hadi Kanaan
Projet dirigé par Professeurs Gilles Roy,
Xuan-Dai Do et Jean-Charles Bernard
Département de Génie Électrique et Génie Informatique
Section Électronique et Énergie

Pour se procurer une copie de ce document, s'adresser au:

Service des Éditions
École Polytechnique de Montréal
Case Postale 6079, Succursale Centre-Ville
Montréal (Québec) H3C 3A7
Téléphone: (514) 340-4711 ext. 4473
Télécopie: (514) 340-3734

Compter 0,10$ par page et ajouter 3,00$ pour la couverture, les frais de poste et la manutention. Régler en dollars canadiens par chèque ou mandat-poste au nom de l'École Polytechnique de Montréal.

Nous n'honorerons que les commandes accompagnées d'un paiement, sauf s'il y a eu entente préalable dans le cas d'établissements d'enseignement, de sociétés ou d'organismes canadiens.

Remerciements

Je tiens à remercier M. Gilles ROY (Ecole Polytehnique de Montréal) pour le soutien qu'il m'a apporté tout au long de la rédaction du présent rapport, ainsi que MM. Jean-Jules BRAULT (Ecole Polytechnique de Montréal) et Maarouf SAAD (Ecole de Technologie Supérieure, ETS) pour leurs précieuses suggestions qui ont permis l'amélioration de ce rapport.

Sommaire

Le présent rapport décrit, d'une façon assez détaillée, des méthodes récentes pour l'identification et le réglage des processus dynamiques non-linéaires, basées sur les réseaux neuronaux artificiels.
Les principaux résultats, théoriques soient-ils ou pratiques, ces derniers étant basés sur des simulations numériques, sont les fruits d'un travail de recherche intense effectué au sein de Yale University par Kumpati S. Narendra et ses collaborateurs.

Cet ouvrage comporte principalement quatre parties:

- La première partie est consacrée à la théorie de base relative aux systèmes dynamiques. On y évoquera quelques définitions et théorèmes nécessaires à la suite de l'exposé, concernant surtout les notions de **stabilité**, **commandabilité**, **observabilité**, **linéarisation**, **ordre** et **degré relatif** d'un système dynamique. Comme on verra dans la suite, les définitions et théorèmes établis pour les systèmes non-linéaires apparaîtront comme une extension à ce type de systèmes de ceux relatifs aux systèmes linéaires.

- Dans la deuxième partie, on décrira d'abord deux types de réseaux de neurones artificiels universellement utilisés comme des approximateurs de fonctions: le **perceptron multi-couche** (Multi-Layer Perceptron, ou MLP) et le **réseau à base de fonctions radiales** (Radial Basis Functions, ou RBF). Pour chaque type de réseau, on évoquera les lois d'ajustement des paramètres internes.
Dans les travaux de simulation, on utilisera presque toujours le réseau MLP en raison de la simplicité de l'algorithme utilisé pour ajuster ses paramètres. Ce dernier est basé sur la rétro-propagation statique de l'erreur évaluée à la sortie du réseau; d'où son nom d'**Algorithme de Rétro-Propagation Statique**.
D'autre part, dans le but d'identifier un système dynamique non-linéaire, on sera amené à envisager des **réseaux neuronaux généralisés** constitués d'une association entre un ou plusieurs réseau(x) de neurones et un ou plusieurs élément(s) linéaires. Dans ces conditions, l'utilisation de l'algorithme de rétro-propagation statique pour l'ajustement de tous les paramètres internes de ce type de réseaux paraîtra insuffisante. Ce problème sera résolu en faisant appel à l'**Algorithme de Rétro-Propagation Dynamique** qui tient compte des blocs linéaires faisant partie de l'identificateur.

- La troisième partie présentera en détails les principales méthodes d'identification des systèmes dynamiques. On y étudiera quatre modèles d'identificateurs associés à quatre types de processus non-linéaires. De plus, pour chacun des modèles, deux modes d'identification seront expliqués: l'**Identification en Parallèle** et l'**Identification en Série-Parallèle**.

- La quatrième et dernière partie constitue celle la plus importante de ce rapport. Elle se propose d'exposer d'une façon assez détaillée les méthodes de réglage les plus récentes, utilisant des réseaux de neurones artificiels. On envisagera deux modes de réglage: le **Réglage Direct** et le **Réglage Indirect** qui, pour des raisons qui seront données ultérieurement, attirera le plus notre attention. En outre, trois différents objectifs de réglage seront proposés: la **stabilisation** qui permettra au système régulé d'atteindre le plus tôt possible son point d'équilibre en l'absence d'excitation externe, la **régulation** qui permettra de stabiliser la sortie du système excité à une valeur constante autre que celle correspondant au point d'équilibre, et la **poursuite** qui imposera une certaine forme à la réponse temporelle du même système. On étudiera ensuite l'effet des perturbations, externes soient-elles ou internes, sur le comportement du système réglé. Finalement, une brève description d'une nouvelle technique de réglage très sophistiquée sera donnée; il s'agira de la méthode des **Modèles Multiples** basée sur les phénomènes de **commutation** et d'**adaptation**. L'identificateur et le régulateur dans ce type de réglage seront constitués respectivement de plusieurs identificateurs et régulateurs élémentaires connectés en parallèle.

Il est à noter que la séparation entre les troisième et quatrième parties n'est qu'artificielle. En réalité, on verra dans la suite que la synthèse des régulateurs ne pourra se faire en pratique sans l'existence d'un identificateur pour le processus à commander.

Enfin, on tient à préciser encore une fois que presque tous les résultats cités dans cet ouvrage ne sont que des reproductions de ceux déjà évoqués dans le livre de Simon Haykin [14] et les articles de Kumpati S. Narendra et collaborateurs [1-13].

Table des Matières

Introduction

On a assisté, pendant les cinq dernières années, à une grande évolution dans le domaine du réglage des systèmes dynamiques par des réseaux de neurones artificiels, que ce soit du point de vue théorique ou pratique. Après une longue période de recherche et d'expérimentation, des régulateurs à base de réseaux neuronaux commencent à apparaître sur le marché et à être utilisés dans de nombreux domaines d'application.

La théorie mathématique des systèmes, dont font partie l'analyse et la synthèse des systèmes dynamiques, a amplement progressé durant les dix dernières années et est devenue, en conséquence, une importante discipline scientifique de vaste domaine d'application.
L'aspect le plus développé dans cette théorie concerne les systèmes linéaires. En effet, sachant que les techniques de réglage des systèmes dynamiques sont étroitement liées à la notion de stabilité, l'établissement des conditions nécessaires et suffisantes pour la stabilité des systèmes linéaires, effectué tout au long du dernier siècle, a permis la génération de méthodes rigoureuses de réglage pour ce type de systèmes.
Contrairement au cas précédent, il n'existe pas une théorie générale concernant la stabilité des systèmes dynamiques non-linéaires. Par conséquent, des méthodes de réglage, assurant à ces derniers une certaine stabilité, une robustesse vis à vis des perturbations internes ou externes et un comportement dynamique appréciable, ne sont pas encore disponibles.
Depuis les années 60, des progrès ont été faits au niveau de l'identification et du réglage des systèmes linéaires stationnaires, dont les paramètres sont totalement ou partiellement inconnus. Le choix des structures de l'identificateur et du régulateur est basé sur la théorie parfaitement établie des systèmes linéaires.

Le problème du réglage consiste à déterminer les lois de commande satisfaisant certains objectifs. Sachant qu'il existe déjà des méthodes précises et parfaitement structurées permettant la synthèse des lois de commande pour les systèmes linéaires, l'application de ces méthodes sur des processus industriels réels n'aboutira pas aux résultats souhaités. Ceci est dû à la complexité, la non-linéarité et l'incertitude caractérisant ces processus.

Le système de réglage qui sera considéré est représenté sous sa forme la plus générale à la figure 0.1. Le processus est généralement régi par des équations différentielles reliant les éléments du vecteur de sortie **y** à ceux du vecteur d'entrée **u**. Le régulateur permet de générer le vecteur de commande **u** à partir du vecteur d'erreur défini par l'écart entre la consigne appliquée au système par l'environnement extérieur et le vecteur de retour généré par la chaîne de rétro-action.

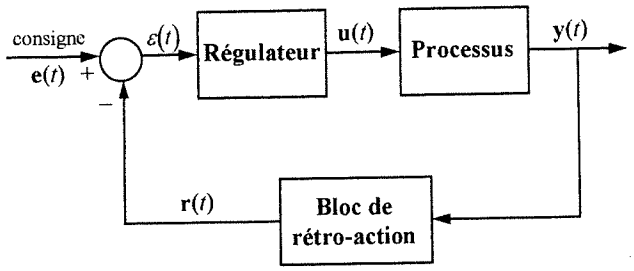

Figure 0.1: Structure générale du système de réglage

Le régulateur est généralement conçu de manière à assurer au système global un comportement en régimes statique et dynamique souhaitable et une robustesse satisfaisante vis-à-vis des perturbations internes ou externes.
Le bloc de rétro-action permet d'assurer la mise à l'échelle des signaux à la sortie du processus, ainsi que le filtrage des signaux parasites et des bruits pratiquement associés à la mesure de la réponse du système.

Trois types de problèmes de réglage seront considérés:

- si la consigne est identiquement nulle et l'on s'intéresse à ramener le point de fonctionnement du système à l'état d'équilibre à partir de conditions initiales quelconques et en un minimum de temps, on est en présence d'un problème de **stabilisation**;

- le problème de **régulation** consiste à synthétiser un régulateur qui imposera certaines caractéristiques statiques (erreur en régime permanent nulle) et dynamiques (dépassement et temps de réponse acceptables au niveau des signaux de sortie) au comportement du système en réponse à un échelon non nul de consigne;

- lorsqu'on applique à l'entrée du système une consigne $\mathbf{e}(t)$ variant dans le temps, et l'on désire que la réponse du système suive fidèlement la trajectoire imposée par cette consigne, on a affaire à un problème de **poursuite** où le régulateur doit être conçu de manière à ce que l'erreur temporelle entre la consigne et le vecteur de retour:

$$\varepsilon(t) = \mathbf{e}(t) - \mathbf{r}(t)$$

tende vers zéro le plus rapidement possible.

Dans le présent rapport, on s'intéressera en premier lieu aux processus dynamiques non-linéaires. On y essayera de démontrer l'efficacité de l'utilisation des réseaux de neurones artificiels dans l'identification et le réglage de ce type de systèmes.

Plusieurs modèles d'identificateurs ont récemment été proposés et utilisés, par la suite, dans la synthèse des régulateurs. L'efficacité de ces modèles a été mise en relief par des simulations numériques effectuées, en grande partie, par un groupe de recherche à Yale University, dirigé par K. S. Narendra [1-13].

Première Partie

Notions et résultats liés à la théorie mathématique des systèmes dynamiques

Dans cette section, plusieurs définitions, théorèmes et concepts de base, qui font partie de la théorie mathématique des systèmes et seront utilisés tout au long du présent rapport, sont introduits.
On définira successivement les notions de stabilité, commandabilité et observabilité des systèmes dynamiques. Dans le cas d'un système non-linéaire, ces notions prennent un caractère local, de sorte qu'elles ne peuvent être définies que dans un voisinage entourant le point d'équilibre du système.

1.1 - Données générales:

1.1.1 - Représentation des systèmes dynamiques dans l'espace d'état:

Les méthodes de représentation, qui s'appliquent à une grande classe de systèmes dynamiques continus ou discrets, sont parfaitement établies dans la théorie.

a) Cas d'un système continu:

Un système dynamique continu est représenté par des équations différentielles de la forme:

$$\frac{d\mathbf{x}(t)}{dt} = \dot{\mathbf{x}}(t) = \Phi[\mathbf{x}(t), \mathbf{u}(t)]$$
$$\mathbf{y}(t) = \Psi[\mathbf{x}(t)]$$

où $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$ représente le vecteur d'état du système, $\mathbf{u}(t) = [u_1(t), u_2(t), \ldots, u_p(t)]^T$ et $\mathbf{y}(t) = [y_1(t), y_2(t), \ldots, y_m(t)]^T$ sont respectivement les vecteurs d'entrée et de sortie, et t est un nombre réel positif représentant le temps. On peut ainsi représenter ce type de système par le schéma bloc de la figure 1.1 où I dénote la matrice identité d'ordre n.
Le système précédent est dit multivariable, d'ordre n, à p entrées et m sorties. Φ et Ψ sont sont deux fonctions statiques non-linéaires telles que:

$$\Phi: \Re^n \times \Re^p \rightarrow \Re^n \qquad \text{et} \qquad \Psi: \Re^n \rightarrow \Re^m$$

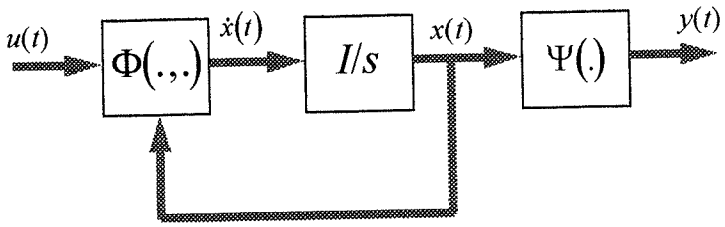

Figure 1.1: Schéma bloc d'un système dynamique continu

b) Cas d'un système discret:

Tout au long du présent rapport, on s'intéressera en particulier aux systèmes dynamiques non-linéaires discrets, notés Σ et définis par des équations d'état de la forme:

$$\Sigma : \begin{aligned} \mathbf{x}(k+1) &= f[\mathbf{x}(k), \mathbf{u}(k)] \\ \mathbf{y}(k) &= h[\mathbf{x}(k)] \end{aligned} \qquad (1)$$

où $\mathbf{x}(k) \in \Re^n$, $\mathbf{u}(k) \in \Re^p$ et $\mathbf{y}(k) \in \Re^m$ représentent respectivement les vecteurs d'entrée, d'état et de sortie du système (1) à l'instant k. n est l'ordre du système, et f et h sont des fonctions statiques, généralement non-linéaires. Le schéma bloc équivalent à ce type de système est donné à la figure 1.2.

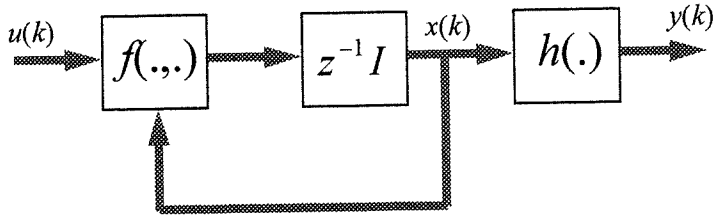

Figure 1.2: Schéma bloc d'un système dynamique discret

1.1.2 - Point d'équilibre et Stabilité des systèmes dynamiques:

On dit que $\mathbf{x}_0$ est un point d'équilibre du système dynamique (1) s'il existe une valeur $\mathbf{u}_0$ du vecteur d'entrée $\mathbf{u}$, telle que:

$$\mathbf{x}_0 = f(\mathbf{x}_0, \mathbf{u}_0)$$

Dans la suite, on supposera que la fonction f vérifie la condition suivante:

$$f(0,0) = 0$$

de sorte que l'origine définie par $\mathbf{x}$=0 correspond à un point d'équilibre du système.
De plus, tout système dynamique sera supposé avoir au moins un point d'équilibre.

Un système est dit stable lorsque de faibles variations au niveau des conditions initiales induisent de faibles changements dans la trajectoire du système dans l'espace d'état.
De plus, lorsque ces changements tendent asymptotiquement en fonction du temps vers zéro,on dit que le système est asymptotiquement stable.
En d'autres termes, l'origine est un point d'équilibre stable du système Σ si, quel que soit $V \subset \Re^n$, il existe un voisinage $W \subset V$ de l'origine tel que, si l'état initial $\mathbf{x}(0)$ appartient à W, alors $\mathbf{x}(k)$ appartient à V pour tout k>0. L'origine est asymptotiquement stable si on a en plus:

$$\lim_{k\to\infty} \mathbf{x}(k) = 0$$

Si W n'est autre que $\Re^n$, l'origine est globalement asymptotiquement stable.
Dans le cas où $\mathbf{x}(k)$ converge vers l'origine en un nombre fini de pas, l'origine est dite finiment stable.

1.1.3 - Théorie de stabilité de Lyapounov:

a) Définition d'une fonction définie positive:

Une fonction V($\mathbf{x}$) est dite définie positive dans une région W contenant l'origine si:

(i) V(0)=0
(ii) V($\mathbf{x}$)>0, $\forall\ \mathbf{x} \in W$ et $\mathbf{x} \neq 0$

b) Définition de la fonction de Lyapounov:

Soit W un ensemble inclus dans $\Re^n$ et contenant l'origine, et V une fonction de $\Re^n$ vers $\Re$. On dit que V est une fonction de Lyapounov associée au système:

$$\mathbf{x}(k+1) = F[\mathbf{x}(k)] \qquad (2)$$

sur l'ensemble W si:

(i) V est continue sur $\Re^n$

(ii) V est définie positive par rapport à l'origine dans W

(iii) $\Delta V(k)=V[\mathbf{x}(k+1)]-V[\mathbf{x}(k)]\leq 0$ tout au long de la trajectoire du système (2) dans l'espace d'état, pour tout $x\in W$

(iv) $V(\mathbf{x})$ tend vers l'infini lorsque $\|\mathbf{x}\|$ tend vers l'infini.

c) Théorème de stabilité de Lyapounov:

Si V est une fonction de Lyapounov associée au système (2) sur un certain voisinage du point d'équilibre $\mathbf{x}=0$, alors l'origine est stable.
Si, en plus, $-\Delta V$ est définie positive par rapport à $\mathbf{x}=0$, alors l'origine est asymptotiquement stable.

1.1.4 - Stabilité sous perturbations:

Dans certains cas, on est amené à modéliser ou identifier le système réel avant de synthétiser le régulateur qui fera répondre le système global conformément à un certain cahier des charges. Cependant, on ne peut pas espérer avoir un modèle idéal du système. Par conséquent, si le système est décrit par l'équation (2), le modèle sera caractérisé par:

$$\hat{\mathbf{x}}(k+1)=\hat{f}[\mathbf{x}(k)]=f[\mathbf{x}(k)]+\mathbf{e}[\mathbf{x}(k)]$$

où $\mathbf{e}[\mathbf{x}(k)]=\hat{f}[\mathbf{x}(k)]-f[\mathbf{x}(k)]$ est l'erreur introduite par la modélisation.
En considérant le système (1) au lieu de (2), l'erreur $\mathbf{e}$ sera fonction de k et de $\mathbf{x}(k)$:

$$\mathbf{e}=\mathbf{e}[k,\mathbf{x}(k)]$$

a) Définition d'une fonction Lipschitzienne:

Soit E et F deux espaces vectoriels normés, soit $D\subset E$, et soit T une fonction de D dans F. S'il existe une constante c telle que:

$$\|T(\mathbf{x}_1)-T(\mathbf{x}_2)\|\leq c\|\mathbf{x}_1-\mathbf{x}_2\|\ ,\ \forall\ \{\mathbf{x}_1,\mathbf{x}_2\}\subset D$$

alors on dit que T est une fonction Lipschitzienne. Dans le cas où $c<1$, T a un effet de contraction.

b) Théorème du point neutre:

Soit D un sous-ensemble fermé de l'espace vectoriel normé E, et soit $T : E \rightarrow E$ une fonction Lipschitzienne ayant un effet de contraction sur D. Alors, il existe un et un seul point $\mathbf{x}_0 \in D$ vérifiant $T(\mathbf{x}_0) = \mathbf{x}_0$. De plus, pour tout $\mathbf{x} \in D$, on a:

$$\lim_{k \to \infty} T^k(\mathbf{x}) = \mathbf{x}_0$$

c) Théorème de stabilité sous perturbations:

Soit $\mathbf{x}(\mathbf{x}_0, k)$ la solution du système (2) correspondant à la condition initiale $\mathbf{x}_0 = \mathbf{x}(\mathbf{x}_0, 0)$. L'origine $\mathbf{x}$=0 est stable sous perturbations si, pour tout ε>0, il existe $\delta_1(\varepsilon)$ et $\delta_2(\varepsilon)$ tels que les conditions $\|\mathbf{x}_0\| < \delta_1(\varepsilon)$ et $\|\mathbf{e}(k, \mathbf{x})\| < \delta_2(\varepsilon)$, pour tout k>0, impliquent $\|\mathbf{x}(\mathbf{x}_0, k)\| < \varepsilon$ pour tout $k \geq 0$.
Si en plus, pour tout ε>0, il existe deux nombres réels positifs r et $K(\varepsilon)$ tels que les conditions $\|\mathbf{x}_0\| < r$ et $\|\mathbf{e}(k, \mathbf{x})\| \leq \delta_2(\varepsilon)$, pour tout $k \geq 0$, impliquent $\|\mathbf{x}(\mathbf{x}_0, k)\| < \varepsilon$ pour tout $k \geq K(\varepsilon)$, alors l'origine est fortement stable sous perturbations.

1.1.5 - Commandabilité des systèmes dynamiques:

Un système dynamique est commandable si, pour tout $(\mathbf{x}_1, \mathbf{x}_2) \in \Re^n \times \Re^n$, il existe une séquence finie du signal d'entrée qui permet de transférer le système de l'état $\mathbf{x}_1$ vers l'état $\mathbf{x}_2$.
Dans le cas des systèmes non-linéaires, les conditions de commandabilité globale sont très difficiles à établir et vérifier. Par conséquent, l'attention sera portée sur les notions locales.

Un système est dit localement commandable autour d'un point d'équilibre $\mathbf{x}$=0 si, pour tout voisinage V de l'origine, il existe un certain voisinage W de l'origine tel que, pour tout $(\mathbf{x}_1, \mathbf{x}_2) \in W^2$, il existe une séquence finie du signal d'entrée qui permet de transférer l'état du système de $\mathbf{x}_1$ vers $\mathbf{x}_2$ sans que celui-ci quitte V.

La notion de commandabilité est liée à l'existence d'un vecteur de commande $\mathbf{u}$ qui fait passer le système d'un état à un autre en un nombre fini de pas. $\mathbf{u}$ peut être soit une fonction de k, soit une fonction de l'état $\mathbf{x}(k)$ du système à l'instant k.
Dans le premier cas, qui correspond en réalité à une commande en boucle ouverte, la seule connaissance de $\mathbf{x}_1 = \mathbf{x}(k_0)$, $\mathbf{x}_2 = \mathbf{x}(k_T)$, k_0 et k_T permet de préciser la valeur $\mathbf{u}(k)$ du vecteur de commande, où $k_0 < k < k_T$. Puisque le vecteur de commande à l'instant k n'est pas explicitement déterminé par l'état du système au même instant, il

s'en suit qu'un système commandé en boucle ouverte est sensible aux bruits et perturbations externes.
Contrairement à ce qui précède, la commande en boucle fermée, où le vecteur **u** est fonction de l'état **x**, est robuste vis à vis de telles perturbations. Dans ce cas, en posant:

$$\mathbf{u} = g(\mathbf{x}) \ ,$$

le système (1) devient autonome et sera caractérisé par une équation de la forme:

$$\mathbf{x}(k+1) = f[\mathbf{x}(k), g[\mathbf{x}(k)]] = F[\mathbf{x}(k)]$$

Le choix de la loi $\mathbf{u} = g(\mathbf{x})$ dépend du comportement qu'on désire imposer au système commandé.

En utilisant la notion de stabilité décrite précédemment, on aboutit au résultat important suivant: s'il existe une loi $\mathbf{u} = g(\mathbf{x})$ qui rend le point d'équilibre $\mathbf{x}=0$ asymptotiquement stable, alors le système est stabilisable autour de ce point.
Dans le cas des systèmes linéaires stationnaires, les notions de commandabilité et de stabilisation sont étroitement liées entre elles. En particulier, la commandabilité implique la stabilisation. On verra dans la suite que, sous certaines conditions, un résultat similaire peut être établi localement pour les systèmes non-linéaires.

1.1.6 - Théorèmes des fonctions inverse et implicite:

Les deux théorèmes suivants seront utilisés lors de la génération du signal de commande dans le cas où celui-ci dépend explicitement de l'état du système à régler.

a) Théorème de la fonction inverse:

Soit E et F deux espaces vectoriels normés, U un ouvert de E, a un élément de U, et f une fonction de U dans F de classe C^p. Si le jacobien de f en a, noté $Df(a)$: $E \rightarrow F$, est inversible, alors f est localement C^p - inversible en a.

b) Théorème de la fonction implicite:

Soit E, F et G trois espaces vectoriels normés, U un ouvert de $E \times F$, (a,b) un élément de U, et f une fonction de U dans G de classe C^p et vérifiant:

$$f(a,b) = 0$$

Si le jacobien de f par rapport à la variable $y \in F$, évalué au point (a,b) et noté $D_y f(a,b): F \to G$, est inversible, alors il existe une boule ouverte V dans E, centrée en a, et une fonction continue $g: V \to F$ telles que:

$$g(a) = b \quad \text{et} \quad f[x, g(x)] = 0 \ , \ \forall x \in V$$

Si le rayon de V est suffisamment faible, alors g est unique et de classe C^p.

1.1.7 - Observabilité des systèmes dynamiques:

L'espace des séquences d'entrée et de sortie, de longueur l, sera noté respectivement $\boldsymbol{U}_l$ et $\boldsymbol{Y}_l$. Les séquences d'entrée et de sortie, de longueur l et débutant à l'instant k seront notées respectivement:

$$U_l(k) = [\mathbf{u}(k), \mathbf{u}(k+1), \ldots, \mathbf{u}(k+l-1)]$$

et

$$Y_l(k) = [\mathbf{y}(k), \mathbf{y}(k+1), \ldots, \mathbf{y}(k+l-1)]$$

D'après la définition du vecteur d'état, $\mathbf{x}(k+l)$ peut être exprimé par:

$$\begin{aligned}
\mathbf{x}(k+l) &= f[\mathbf{x}(k+l-1), \mathbf{u}(k+l-1)] \\
&= f[f[\mathbf{x}(k+l-2), \mathbf{u}(k+l-2)], \mathbf{u}(k+l-1)] \\
&= \ldots \\
&= f[f[\ldots f[f[\mathbf{x}(k), \mathbf{u}(k)], \mathbf{u}(k+1)], \ldots, \mathbf{u}(k+l-2)], \mathbf{u}(k+l-1)] \\
&= F_l[\mathbf{x}(k), U_l(k)]
\end{aligned}$$

où F_l est une fonction de $\Re^n \times \boldsymbol{U}_l$ vers $\Re^n$.
D'une manière similaire, la sortie à l'instant $k+l$ s'écrit:

$$\mathbf{y}(k+l) = h[\mathbf{x}(k+l)] = h[F_l[\mathbf{x}(k), U_l(k)]]$$

et la séquence $Y_l(k)$ s'exprime par:

$$Y_l(k) = H_l[\mathbf{x}(k), U_{l-1}(k)]$$

où H_l est une fonction de $\Re^n \times \boldsymbol{U}_{l-1}$ vers $\boldsymbol{Y}_l$.

Le système dynamique (1) est dit observable si, pour tout $(\mathbf{x}_1, \mathbf{x}_2) \in \Re^n \times \Re^n$, il existe une séquence de longueur finie l du signal d'entrée, $U_l = [\mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(l-1)]$, telle que $Y_l(\mathbf{x}_1, U_l) \neq Y_l(\mathbf{x}_2, U_l)$, Y_l étant la séquence de sortie.

En se basant sur les observations entrée-sortie du système, l'efficacité de l'estimation ou de l'identification du vecteur d'état est étroitement liée aux propriétés d'observabilité de ce système.

Le système (1) est dit fortement observable si toute séquence d'entrée, de longueur l donnée, permet de déterminer d'une manière unique l'état du système.

D'autre part, s'il existe un nombre entier l tel que toute séquence d'entrée, de longueur supérieure ou égale à l, permet de déterminer d'une manière unique l'état du système (1), ce dernier sera dit génériquement observable.

1.2 - Linéarisation d'un système dynamique:

Sachant que la partie la plus développée dans la théorie des systèmes concerne les systèmes linéaires, on essayera dans ce paragraphe d'étendre, sous certaines conditions, les résultats établis pour ces systèmes au cas des systèmes dynamiques non-linéaires.
Pour cela, on fera appel à la notion de *linéarisation* d'un système dynamique au voisinage de son point d'équilibre supposé défini par $\mathbf{x} = 0$ et $\mathbf{u} = 0$.

La linéarisation autour du point d'équilibre du système Σ, décrit par les équations d'état (1), permet d'obtenir le système Σ_L défini comme suit:

$$\Sigma_L : \begin{array}{l} \mathbf{x}(k+1) = A.\mathbf{x}(k) + B.\mathbf{u}(k) \\ \mathbf{y}(k) = C.\mathbf{x}(k) \end{array} \qquad (3)$$

Les matrices A, B et C, de dimensions respectives (n,n), (n,p) et (m,n), sont définies par:

$$A = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\Big|_{(0,0)} \quad ; \quad B = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\Big|_{(0,0)} \quad \text{et} \quad C = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\Big|_{(0)}$$

La théorie des systèmes linéaires invariants (ou stationnaires), où les éléments des matrices A, B et C sont supposés entièrement connus et invariants dans le temps, est parfaitement développée, et les concepts de stabilité, commandabilité et observabilité, qui leur sont associés, ont été intensivement étudiés durant les quatre dernières décennies.

Des méthodes de calcul du vecteur de commande $\mathbf{u}$, permettant d'optimiser un certain critère de performance, sont de même parfaitement établies. Ceci est dû principalement au fait que ce type de problème peut être toujours ramené à une résolution d'un système linéaire de n équations à n inconnus.

1.2.1 - Stabilité:

La stabilité du système linéarisé Σ_L est entièrement déterminée par la matrice A. D'après la théorie des systèmes linéaires, Σ_L est asymptotiquement stable si les valeurs propres de A sont à l'intérieur du cercle unité dans le plan complexe.

1.2.2 - Commandabilité:

La commandabilité du système Σ_L dépend des matrices A et B. Σ_L est dit commandable si, et seulement si, pour tout état initial $\mathbf{x}_i$, il peut être amené à un état final $\mathbf{x}_f$ en un nombre fini de pas.

D'après la théorie des systèmes linéaires, cette condition peut être exprimée sous forme mathématique. En effet, le système Σ_L est commandable si, et seulement si, la matrice de dimension $(n,n\times p)$ définie par:

$$M_C = \left[B, A.B, \ldots, A^{n-1}.B\right]$$

est de rang n. M_C est appelée la *matrice de commandabilité* du système Σ_L.

Dans le cas d'un système linéaire monovariable ($p = 1$), M_C est une matrice carrée d'ordre n, dont la non-singularité est une condition nécessaire et suffisante pour la commandabilité du système correspondant.

Si Σ_L est commandable, il est aussi stabilisable via un retour d'état. En d'autres termes, il existe une matrice K telle que la loi de commande $\mathbf{u} = K.\mathbf{x}$ rend le système global asymptotiquement stable. De plus, un choix judicieux de K permet au système global défini par l'équation d'état:

$$\mathbf{x}(k+1) = (A + B.K).\mathbf{x}(k)$$

d'avoir des valeurs propres souhaitées.

Sous certaines conditions, la commandabilité du système linéarisé Σ_L peut s'étendre au cas du système principal Σ décrit par (1). En effet, si Σ est un système dynamique non-linéaire discret stationnaire dont le point d'équilibre est défini par $\mathbf{x} = 0$ et $\mathbf{u} = 0$, et si la linéarisation Σ_L de Σ autour de ce point est commandable, alors Σ est localement commandable, et il existe un voisinage $V \subset \Re^n$ autour de l'origine et une loi de commande continue $\mathbf{u}(k) = g[\mathbf{x}(k)]$ tels que le système bouclé décrit par:

$$\mathbf{x}(k+1) = f[\mathbf{x}(k), g[\mathbf{x}(k)]]$$

est n-finiment stable dans V, n étant l'ordre du système (en d'autres termes, tout point $\mathbf{x} \in V$ peut converger vers l'origine en n itérations au plus).

Ce théorème fut intensivement utilisé dans la synthèse des régulateurs visés à stabiliser des systèmes non-linéaires autour de leurs points d'équilibre instables. Pour le démontrer, on considère la fonction $G: \Re^{2n} \rightarrow \Re^{2n}$ définie par:

$$G[\mathbf{x}(k), U_n(k)] = \left(\mathbf{x}(k), \mathbf{x}(k+n)\right)$$

où $U_n(k) = [\mathbf{u}(k), \mathbf{u}(k+1), \ldots, \mathbf{u}(k+n-1)]$ est la séquence d'entrée de longueur n à l'instant k. La matrice jacobienne de G, évaluée au point (0,0), est donnée par:

$$D_{(0,0)}G = \begin{bmatrix} I & 0 \\ D_{\mathbf{x}(k)}\mathbf{x}(k+n)\big|_{(0,0)} & D_{U_n(k)}\mathbf{x}(k+n)\big|_{(0,0)} \end{bmatrix}$$

où I est la matrice identité de rang n et $D_{U_n(k)}\mathbf{x}(k+n)\big|_{(0,0)}$ est tel que:

$$D_{U_n(k)}\mathbf{x}(k+n)\big|_{(0,0)} = \left[\frac{\partial \mathbf{x}(k+n)}{\partial \mathbf{u}(k)} \quad \frac{\partial \mathbf{x}(k+n)}{\partial \mathbf{u}(k+1)} \quad \cdots \quad \frac{\partial \mathbf{x}(k+n)}{\partial \mathbf{u}(k+n-1)} \right]_{(0,0)}$$

D'après les résultats du paragraphe 1.1.7, $\mathbf{x}(k+n)$ est fonction de $\mathbf{x}(k)$ et $U_n(k)$:

$$\mathbf{x}(k+n) = F_n[\mathbf{x}(k), U_n(k)]$$

De plus, pour tout $i = 1, \ldots, n$, on a:

$$\frac{\partial F_n[\mathbf{x}(k), U_n(k)]}{\partial \mathbf{u}(k-1+i)}\Big|_{(0,0)} = A^{n-i}.B$$

Il apparaît donc que $D_{U_n(k)}\mathbf{x}(k+n)\big|_{(0,0)}$ n'est autre qu'un arrangement de la matrice de commandabilité M_C. Si celle-ci est de rang n, alors d'après le théorème de la fonction inverse (donné au paragraphe 1.1.6) il existe localement une fonction continue $\Psi = G^{-1}$ telle que:

$$\left(\mathbf{x}(k), U_n(k)\right) = \Psi[\mathbf{x}(k), \mathbf{x}(k+n)] = \left(\Psi_x[\mathbf{x}(k), \mathbf{x}(k+n)], \Psi_u[\mathbf{x}(k), \mathbf{x}(k+n)]\right)$$

En d'autres termes, étant donné deux points quelconques $\mathbf{x}_i$ et $\mathbf{x}_f$, il existe une séquence d'entrée unique $U_n = \Psi_u[\mathbf{x}_i, \mathbf{x}_f]$ qui fait passer l'état du système de $\mathbf{x}_i$ à $\mathbf{x}_f$ en n étapes. Pour tout voisinage $V \subset \Re^n$, si $\mathbf{x}_i$ et $\mathbf{x}_f$ sont choisis suffisamment proches

de l'origine, la trajectoire suivie par l'état du système pour passer de l'état $\mathbf{x}_i$ à l'état $\mathbf{x}_f$ se situera entièrement à l'intérieur de V.
En posant $\mathbf{x}_f = 0$, et en utilisant le théorème de la fonction implicite (donné au paragraphe 1.1.6) et la propriété d'unicité de la séquence d'entrée U_n (conduisant le système à son point d'équilibre après n étapes), on peut montrer qu'il existe une fonction continue $g: \Re^n \rightarrow \Re$ telle que, pour tout $\mathbf{x} \in V$, le système:

$$\mathbf{x}(k+1) = f\big(\mathbf{x}(k), g[\mathbf{x}(k)]\big)$$

converge vers l'origine en n étapes au plus.

1.2.3 - Observabilité:

D'un point de vue qualitatif, le système linéarisé Σ_L est dit observable si tout vecteur d'état $\mathbf{x}(k)$, $k > 0$, peut être déterminé par un nombre fini de mesures effectuées à la sortie du système.

L'observabilité de Σ_L dépend des matrices A et C. En effet, d'après la théorie des systèmes linéaires, Σ_L est observable si, et seulement si, la matrice de dimension $(n \times m, n)$ définie par:

$$M_O = \begin{bmatrix} C \\ C.A \\ \vdots \\ C.A^{n-1} \end{bmatrix}$$

est de rang n. M_O est la *matrice d'observabilité* du système Σ_L.

Dans le cas d'un système monovariable $(m = 1)$, M_O est une matrice carrée d'ordre n, et sa non-singularité est une condition nécessaire et suffisante pour l'observabilité du système correspondant.

L'observabilité du système linéaire Σ_L s'étend, d'une certaine manière, au système principal Σ décrit par (1). Plus précisément, si Σ_L est observable, alors Σ est localement fortement observable.
En effet, d'après les résultats établis au paragraphe 1.1.7, la séquence de sortie $Y_n(k) = [\mathbf{y}(k), \mathbf{y}(k+1), \ldots, \mathbf{y}(k+n-1)]$ du système Σ d'ordre n peut s'exprimer en fonction du vecteur d'état $\mathbf{x}(k)$ et de la séquence d'entrée $U_{n-1}(k) = [\mathbf{u}(k), \mathbf{u}(k+1), \ldots, \mathbf{u}(k+n-2)]$ comme suit:

$$Y_n(k) = H_n\left[\mathbf{x}(k), U_{n-1}(k)\right]$$

Le jacobien de $Y_n(k)$ par rapport à $\mathbf{x}(k)$, noté $D_x Y_n(k)$, évalué à l'origine (supposée le point d'équilibre de Σ) n'est autre que la matrice d'observabilité M_O du système linéarisé Σ_L. Si $\widetilde{H} : \boldsymbol{U}_{n-1} \times \Re^n \rightarrow \boldsymbol{U}_{n-1} \times \boldsymbol{Y}_n$ est la fonction définie par:

$$\left[U_{n-1}(k), Y_n(k)\right] = \widetilde{H}\left[U_{n-1}(k), \mathbf{x}(k)\right]$$

$\boldsymbol{U}_{n-1}$ et $\boldsymbol{Y}_n$ étant les espaces des séquences d'entrée et de sortie de longueur n-1 et n respectivement, la matrice jacobienne de $\widetilde{H}$ à l'origine est donnée par:

$$D\widetilde{H}\Big|_{(0,0)} = \begin{bmatrix} I & 0 \\ D_{U_{n-1}} Y_n(k)\Big|_{(0,0)} & D_x Y_n(k)\Big|_{(0,0)} \end{bmatrix}$$

où I est la matrice identité et $D_{U_{n-1}} Y_n(k)\Big|_{(0,0)}$ le jacobien évalué à l'origine de $Y_n(k)$ par rapport à la séquence $U_{n-1}(k)$.

D'après sa forme spéciale, le déterminant de $D\widetilde{H}\Big|_{(0,0)}$ est égal au produit des déterminants de la matrice identité et du jacobien $D_x Y_n(k)\Big|_{(0,0)}$, donc au déterminant de M_O. Par conséquent, si M_O est de rang n (c'est-à-dire si Σ_L est observable), $D\widetilde{H}\Big|_{(0,0)}$ est non-singulière et, d'après le théorème de la fonction inverse, il existe un voisinage $V \subset \boldsymbol{U}_{n-1} \times \Re^n$ de l'origine sur lequel $\widetilde{H}$ est inversible.

En notant $\widetilde{\Phi} : \boldsymbol{U}_{n-1} \times \boldsymbol{Y}_n \rightarrow \boldsymbol{U}_{n-1} \times \Re^n$ la fonction inverse de $\widetilde{H}$, et $\overline{\Phi}$ la projection sur les n derniers composants de $\widetilde{\Phi}$, on obtient localement:

$$\mathbf{x}(k) = \overline{\Phi}\left[U_{n-1}(k), Y_n(k)\right]$$

1.3 - Représentation entrée-sortie des systèmes dynamiques:

1.3.1 - Cas du système linéarisé:

Le système linéaire discret multivariable Σ_L, ayant p entrées et m sorties et décrit par les équations (3), peut être représenté sous plusieurs formes équivalentes.

a) Matrice de transfert:

En faisant appel à la transformée en z, on peut écrire:

$$\mathbf{x}(k+1) = z.\mathbf{x}(k)$$

z étant l'opérateur de décalage unitaire de temps.
Par conséquent, en éliminant $\mathbf{x}(k)$ dans le système (3), Σ_L peut être décrit par sa matrice de transfert $W(z)$ liant la sortie $\mathbf{y}(k)$ à l'entrée $\mathbf{u}(k)$ et donnée par:

$$W(z) = \frac{\mathbf{y}(k)}{\mathbf{u}(k)} = C.(z.I - A)^{-1}.B$$

I étant la matrice identité de rang n.

Dans le cas d'un système monovariable ($p = m = 1$), B et C sont des vecteurs, et $W(z)$ est la *fonction de transfert* du système linéaire Σ_L.

b) Représentation temporelle - Modèle ARMA:

En appliquant successivement les équations (3), tout en supposant la matrice A non-singulière de sorte que sa matrice inverse A^{-1} puisse exister, on obtient:

$$\begin{aligned}
&\mathbf{y}(k) = C.\mathbf{x}(k)\\
&\mathbf{y}(k-1) = C.\mathbf{x}(k-1) = C.A^{-1}.\mathbf{x}(k) - C.A^{-1}.B.\mathbf{u}(k-1)\\
&\mathbf{y}(k-2) = C.A^{-2}.\mathbf{x}(k) - C.A^{-2}.B.\mathbf{u}(k-1) - C.A^{-1}.B.\mathbf{u}(k-2)\\
&\vdots\\
&\mathbf{y}(k-n+1) = C.A^{-n+1}.\mathbf{x}(k) - \sum_{i=1}^{n-1} C.A^{-i}.B.\mathbf{u}(k-n+i)
\end{aligned}$$

En notant respectivement $U_{n-1}(k-n+1)$ et $Y_n(k-n+1)$ les séquences d'entrée $[\mathbf{u}(k-n+1),\ldots,\mathbf{u}(k-1)]^T$ et de sortie $[\mathbf{y}(k-n+1),\ldots,\mathbf{y}(k)]^T$, les équations précédentes peuvent être regroupées sous forme matricielle comme suit:

$$Y_n(k-n+1) = \begin{bmatrix} C.A^{-n+1} \\ C.A^{-n+2} \\ \vdots \\ C.A^{-1} \end{bmatrix}.\mathbf{x}(k) - \begin{bmatrix} C.A^{-1}.B & C.A^{-2}.B & \cdots & C.A^{-n+1}.B \\ 0 & C.A^{-1}.B & \cdots & C.A^{-n+2}.B \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C.A^{-1}.B \\ 0 & 0 & \cdots & 0 \end{bmatrix}.U_{n-1}(k-n+1) \qquad (4)$$

Les matrices-colonnes $U_{n-1}(k-n+1)$ et $Y_n(k-n+1)$ sont de dimensions $((n-1)\times p,1)$ et $(n\times m,1)$ respectivement. Les termes $C.A^{-i}.B$, $i=0,\ldots,n-2$, sont appelés les paramètres *markoviens* du système Σ_L.

En s'intéressant à l'une des m sorties du système, on a pour tout $j=1,\ldots,m$:

$$Y_{j,n}(k-n+1) = M_j.\mathbf{x}(k) - N_j.U_{n-1}(k-n+1)$$

où:

$$M_j = \begin{bmatrix} C_j.A^{-n+1} \\ C_j.A^{-n+2} \\ \vdots \\ C_j.A^{-1} \\ C_j \end{bmatrix} \quad \text{et} \quad N_j = \begin{bmatrix} C_j.A^{-1}.B & C_j.A^{-2}.B & \cdots & C_j.A^{-n+1}.B \\ 0 & C_j.A^{-1}.B & \cdots & C_j.A^{-n+2}.B \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_j.A^{-1}.B \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Les matrices M_j et N_j sont de dimensions respectives (n,n) et $(n,(n-1)\times p)$. C_j est le j^{eme} vecteur-ligne de la matrice C.

Si le système Σ_L est observable, M_j est non-singulière et, par conséquent, l'expression de $\mathbf{x}(k)$ s'écrit:

$$\mathbf{x}(k) = M_j^{-1}.\left[Y_{j,n}(k-n+1) + N_j.U_{n-1}(k-n+1)\right]$$

On en déduit qu'à tout instant k le vecteur d'état dépend uniquement des n-1 anciennes valeurs des vecteurs d'entrée **u** et de sortie **y** du système, ainsi que de la récente valeur $\mathbf{y}(k)$ du vecteur de sortie.

D'après ce qui précède, pour tout instant k et pour tout $j=1,\ldots,m$, on a:

$$\begin{aligned} y_j(k+1) &= C_j.A.\mathbf{x}(k) + C_j.B.\mathbf{u}(k) \\ &= C_j.A.M_j^{-1}.Y_{j,n}(k-n+1) + C_j.A.M_j^{-1}.N_j.U_{n-1}(k-n+1) + C_j.B.\mathbf{u}(k) \end{aligned}$$

où les matrices $C_j.A.M_j^{-1}$, $C_j.A.M_j^{-1}.N_j$ et $C_j.B$ sont de dimensions respectives $(1,n)$, $(1,(n\text{-}1)\times p)$ et $(1,p)$.

L'expression précédente peut être arrangée de la manière suivante:

$$y_j(k+1) = \sum_{i=1}^{n} \delta_{j,i} . y_j(k-n+i) + \sum_{l=1}^{n} \Gamma_{j,l} . \mathbf{u}(k-n+l)$$

où les nombres réels $\delta_{j,i}$ sont les éléments de la matrice-ligne $C_j . A . M_j^{-1}$, les $\Gamma_{j,l}$ sont des sous-matrices de dimension (1,*p*) de $C_j . A . M_j^{-1} . N_j$, pour $l = 1,\ldots,n-1$, et $\Gamma_{j,n}$ n'est autre que $C_j . B$.

Le vecteur de sortie **y** à l'instant k+1 s'exprime alors par:

$$\mathbf{y}(k+1) = \sum_{i=1}^{n} \Delta_i . \mathbf{y}(k-n+i) + \sum_{l=1}^{n} \Gamma_l . \mathbf{u}(k-n+l) \qquad (5)$$

où, pour tout $i = 1,\ldots,n$, Δ_i est une matrice diagonale de dimension (*m*,*m*), dont les éléments diagonaux sont les $\delta_{j,i}$, et, pour tout $l = 1,\ldots,n$, Γ_l est une matrice de dimension (*m*,*p*) formée par les *m* vecteurs-lignes $\Gamma_{j,l}$.

On aboutit ainsi au résultat important suivant: les valeurs à un instant donné des signaux de sortie du système linéaire Σ_L sont entièrement déterminées par les *n* anciennes valeurs des signaux d'entrée et de sortie du système. L'équation (5) est la *représentation entrée-sortie temporelle* du système Σ_L. Dans la littérature, on dit souvent qu'elle constitue le *modèle ARMA* (AutoRegressive Moving Average) du système.
Dans le cas où le système Σ_L est monovariable ($p = m = 1$), les matrices Δ_i et Γ_l sont réduites à des constantes réelles, et la représentation entrée-sortie temporelle du système est complètement déterminée par 2*n* paramètres.

1.3.2 - <u>Cas des systèmes non-linéaires</u>:

a) <u>Extension du modèle ARMA au cas des systèmes non-linéaires - Modèle NARMA</u>:

Etant donné le système dynamique Σ décrit par les équations (1), on a par récurrence:

$$\begin{aligned}
&\mathbf{y}(k) = h[\mathbf{x}(k)] = \Phi_1[\mathbf{x}(k)] \\
&\mathbf{y}(k+1) = h[f[\mathbf{x}(k),\mathbf{u}(k)]] = \Phi_2[\mathbf{x}(k),\mathbf{u}(k)] \\
&\vdots \\
&\mathbf{y}(k+n-1) = \Phi_n[\mathbf{x}(k),\mathbf{u}(k),\mathbf{u}(k+1),\ldots,\mathbf{u}(k+n-2)]
\end{aligned}$$

En notant respectivement $U_{n-1}(k)$ et $Y_n(k)$ les séquences d'entrée $[\mathbf{u}(k),\ldots,\mathbf{u}(k+n-2)]$ et de sortie $[\mathbf{y}(k),\ldots,\mathbf{y}(k+n-1)]$, on peut écrire:

$$Y_n(k) = \Phi[\mathbf{x}(k), U_{n-1}(k)]$$

Si le jacobien $\left.\frac{\partial \Phi}{\partial \mathbf{x}(k)}\right|_{(0,0)}$ est non-singulier (ce qui revient à dire que le système linéarisé Σ_L est observable), on peut montrer, en appliquant le théorème de la fonction inverse, que $\mathbf{x}(k)$ peut être exprimé localement en termes de $Y_n(k)$ et $U_{n-1}(k)$:

$$\mathbf{x}(k) = \Theta[\mathbf{y}(k), \ldots, \mathbf{y}(k+n-1), \mathbf{u}(k), \ldots, \mathbf{u}(k+n-2)] \qquad (6)$$

D'autre part, on a:

$$\begin{aligned}
&\mathbf{x}(k+1) = f[\mathbf{x}(k), \mathbf{u}(k)] = \Psi_1[\mathbf{x}(k), \mathbf{u}(k)] \\
&\mathbf{x}(k+2) = f[f[\mathbf{x}(k), \mathbf{u}(k)], \mathbf{u}(k+1)] = \Psi_2[\mathbf{x}(k), \mathbf{u}(k), \mathbf{u}(k+1)] \\
&\vdots \\
&\mathbf{x}(k+n) = \Psi_n[\mathbf{x}(k), \mathbf{u}(k), \mathbf{u}(k+1), \ldots, \mathbf{u}(k+n-1)]
\end{aligned}$$

En combinant la dernière expression avec l'équation (6), puis en opérant un décalage temporel de n-1 pas en arrière, on aboutit à:

$$\mathbf{x}(k+1) = \Lambda[\mathbf{y}(k), \ldots, \mathbf{y}(k-n+1), \mathbf{u}(k), \ldots, \mathbf{u}(k-n+1)]$$

Par conséquent:

$$\mathbf{y}(k+1) = h[\mathbf{x}(k+1)] = \Im[\mathbf{y}(k), \ldots, \mathbf{y}(k-n+1), \mathbf{u}(k), \ldots, \mathbf{u}(k-n+1)] \qquad (7)$$

L'équation (7) constitue la *représentation entrée-sortie locale* (au voisinage du point d'équilibre) du système dynamique non-linéaire (1). Dans la littérature, elle est connue comme étant le *modèle NARMA* (Non-linear AutoRegressive Moving Average) du système.

b) Modèle NARMA généralisé:

Le modèle NARMA, établi précédemment, est de nature locale car il se base sur le théorème de la fonction implicite et sur l'hypothèse de non-singularité de la matrice $\frac{\partial \Phi}{\partial \mathbf{x}(k)}$ au voisinage du point d'équilibre $(\mathbf{x} = 0, \mathbf{u} = 0)$.

Des travaux de recherche furent menés par D. Aeyels [20], E. D. Sontag [21] et, tout récemment, A. U. Levin et K. S. Narendra [12] dans le but de déterminer des modèles NARMA globalement valides. D. Aeyels a montré que le système dynamique décrit par:

$$\dot{\mathbf{x}} = f(\mathbf{x}) \qquad \text{et} \qquad \mathbf{y} = h(\mathbf{x})$$

est, dans la plupart des cas, observable si $2n+1$ mesures sont effectuées à la sortie du système. E. D. Sontag étudia les conditions d'existence de représentations entrée-sortie globales dans le cas où les fonctions f et h du système (1) sont polynômiales.
D'après A. U. Levin et K. S. Narendra, si la fonction f est inversible de sorte que, pour tout vecteur $\mathbf{u}(k)$, $\mathbf{x}(k)$ puisse être déterminé par $\mathbf{x}(k+1)$, le vecteur de sortie du système (1) à l'instant $k+1$ est défini par $2n+1$ anciennes valeurs des signaux d'entrée et de sortie:

$$\mathbf{y}(k+1) = \hat{\mathfrak{I}}[\mathbf{y}(k),\ldots,\mathbf{y}(k-2n),\mathbf{u}(k),\ldots,\mathbf{u}(k-2n)] \tag{8}$$

L'équation (8) constitue le *modèle NARMA généralisé* du système (1).

c) Modèles NARMA approximatifs:

Dans certains problèmes de réglage, on se trouve contraint à utiliser des modèles simplifiés pour représenter un processus dynamique non-linéaire donné. Deux modèles approximatifs, NARMA-L1 et NARMA-L2, ont été proposés, chacun d'eux étant basé sur une utilisation différente de la série de Taylor.

c.1) Modèle NARMA-L1:

Le premier modèle est décrit par:

$$\mathbf{y}(k+1) = \mathfrak{I}_0[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1)] + \sum_{i=0}^{n-1} G_i[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1)].\mathbf{u}(k-i)$$

où:

$$\mathfrak{I}_0[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1)] = \mathfrak{I}[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),0,\ldots,0]$$

et:

$$G_i[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1)] = \frac{\partial \mathfrak{I}}{\partial \mathbf{u}(k-i)}\Big|_{(\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),0,\ldots,0)}$$

c.2) Modèle NARMA-L2:

Le second modèle est décrit par l'équation suivante:

$$\begin{aligned}\mathbf{y}(k+1) = {} & \mathfrak{I}_1[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),\mathbf{u}(k-1),\ldots,\mathbf{u}(k-n+1)] \\ & + \mathfrak{I}_2[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),\mathbf{u}(k-1),\ldots,\mathbf{u}(k-n+1)].\mathbf{u}(k)\end{aligned}$$

Cette dernière est obtenue par un développement en série de Taylor de la fonction $\Im$ autour de $(\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),0,\mathbf{u}(k-1),\ldots,\mathbf{u}(k-n+1))$.

1.3.3 - Notion de degré relatif:

Jusqu'à présent, on a toujours supposé que le signal de sortie, à l'instant k+1, d'un système dynamique est affecté par les valeurs précédentes (à l'instant k) des signaux d'entrée et de sortie.
Dans certains systèmes monovariables, ces mêmes valeurs n'affectent la sortie qu'après un délai de temps , noté d, de sorte qu'on peut écrire:

$$\mathbf{y}(k+d) = \Im[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),\mathbf{u}(k),\ldots,\mathbf{u}(k-n+1)]$$

d est appelé le *degré relatif* du système.

Dans le cas des systèmes multivariables, à p entrées et m sorties, il existe $p \times m$ fonctions de transfert élémentaires liant les signaux de sorties à ceux d'entrée, et ayant chacune un degré relatif différent de ceux des autres. Si $d_{i,j}$ dénote le délai entre la j^{eme} entrée et la i^{eme} sortie, le degré relatif de la i^{eme} sortie est défini par:

$$d_i = \min_j d_{i,j}$$

En affectant à i les valeurs 1,...,m, on obtient successivement les degrés relatifs de tous les signaux de sortie. Par conséquent, la représentation entrée-sortie du système multivariable devient:

$$\begin{bmatrix} y_1(k+d_1) \\ y_2(k+d_2) \\ \vdots \\ y_m(k+d_m) \end{bmatrix} = \begin{bmatrix} C_1.A^{d_1}.\mathbf{x}(k)+E_1.\mathbf{u}(k) \\ C_2.A^{d_2}.\mathbf{x}(k)+E_2.\mathbf{u}(k) \\ \vdots \\ C_m.A^{d_m}.\mathbf{x}(k)+E_m.\mathbf{u}(k) \end{bmatrix}$$

où $E_i = C_i.A^{d_i-1}.B$ et C_i est le i^{eme} vecteur-ligne de C, pour tout $i = 1,\ldots,m$.

1.3.4 - Système à minimum de phase:

Un système monovariable linéarisé est dit *à minimum de phase* si le numérateur de sa fonction de transfert $W(z)$ a toutes ses racines situées à l'intérieur du cercle unité dans le plan complexe. Dans ces conditions, tout signal de sortie borné correspond à une excitation bornée à l'entrée du système.

Dans le cas d'un système linéaire à phase non minimale, la réponse à un signal d'entrée non borné peut être bornée et même tendre vers zéro. Le réglage de ce type de systèmes s'avère très difficile, voire impossible, à réaliser.

Deuxième Partie

Réseaux neuronaux artificiels utilisés comme approximateurs de fonctions

Dans cette partie, deux grandes classes de réseaux de neurones artificiels seront étudiées séparément. Il s'agit, d'une part, des *perceptrons multi-couches* (MultiLayer Perceptrons, ou MLP) et, d'autre part, des *réseaux à base de fonctions radiales* (Radial Basis Function network, ou RBF). Ces deux types de réseaux neuronaux sont considérés comme des *approximateurs universels*; ils sont souvent utilisés dans la pratique pour approximer des fonctions continues complexes.
A chaque type de réseaux sont associés des algorithmes spécifiques permettant l'ajustement de ses paramètres internes. Ce phénomène d'ajustement, qui modélise en quelque sorte l'apprentissage du réseau, permettra à ce dernier de "mimer" aussi parfaitement que possible le processus qu'on désire identifier.
Avant d'exposer les caractéristiques et propriétés de chaque type de réseaux, ainsi que les algorithmes qui lui sont associés, on débutera par une brève description d'un simple neurone artificiel qui est l'élément de base des perceptrons multi-couches.
L'étude qui suit est basée sur les résultats donnés dans l'ouvrage de S. Haykin [14] auquel on conseille de se reporter.

2.1 - Description du neurone artificiel:

Un *neurone* est une unité de traitement de l'information, sur laquelle se base le fonctionnement global d'un réseau neuronal.

La structure du neurone est illustrée à la figure 2.1. Elle fait apparaître trois éléments essentiels:

- les *synapses* (ou liens de connexion), dont chacune est associée à une grandeur appelée *poids*: en particulier, un signal x_j, appliqué à l'entrée de la synapse j connectée au neurone k, est multiplié par le poids synaptique w_{kj}; si w_{kj} est positif, on dit que la synapse j correspondante est *excitatrice*, et s'il est négatif, la synapse j est dite *inhibitrice*;

- un *sommateur* permettant d'additionner les signaux d'entrée pondérés;

- une *fonction d'activation* φ limitant l'amplitude du signal à la sortie du neurone.

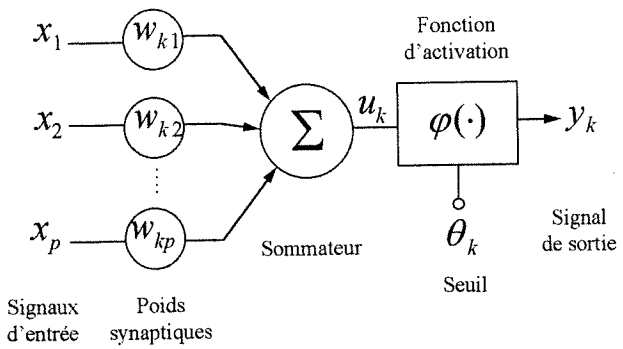

Figure 2.1: Modèle non-linéaire d'un neurone

De plus, dans le modèle du neurone donné à la figure 2.1, une grandeur externe θ_k s'applique au niveau de la fonction d'activation. Cette grandeur, appelée *seuil*, introduit un décalage de la fonction d'activation φ parallèlement à l'axe de ses arguments. Le fonctionnement du neurone k est alors régi par les deux équations suivantes:

$$u_k = \sum_{j=1}^{p} w_{kj} . x_j$$

et (1)

$$y_k = \varphi(u_k - \theta_k)$$

où, pour tout $j = 1,\ldots,p$, x_j est le signal appliqué à l'entrée de la j^{eme} synapse, et w_{kj} le poids associé à cette synapse.
u_k est le signal de sortie du sommateur, θ_k le seuil, φ la fonction d'activation et y_k le signal de sortie du neurone k.

En définissant le *niveau d'activité interne* du neurone k par:

$$v_k = u_k - \theta_k$$

les équations (1) se réécrivent comme suit:

$$v_k = \sum_{j=0}^{p} w_{kj} . x_j$$

et (2)

$$y_k = \varphi(v_k)$$

Dans l'expression précédente de v_k, on a ajouté une nouvelle synapse "fictive" correspondant à un signal d'entrée $x_0 = -1$ et un poids $w_{k0} = \theta_k$. Par conséquent, le neurone k de la figure 2.1 peut être représenté sous une autre forme équivalente donnée à la figure 2.2. L'effet du seuil est alors représenté, d'une part, par la création d'une nouvelle synapse dont le poids est égal au seuil et, d'autre part, par l'application d'un signal de valeur fixe égale à -1 à l'entrée de cette synapse.

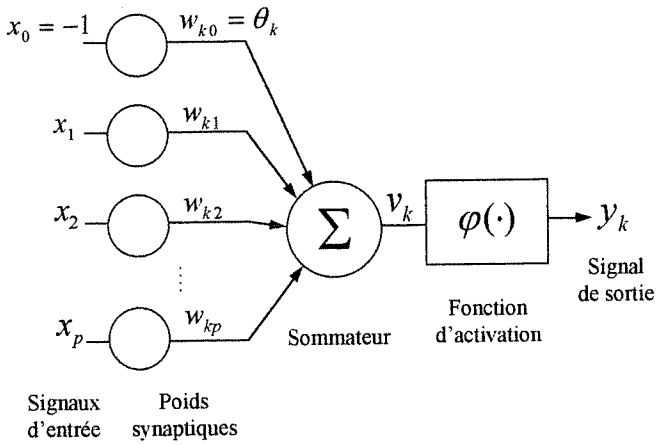

Figure 2.2: Autre modèle non-linéaire d'un neurone

La fonction d'activation φ peut prendre plusieurs formes différentes (linéaires ou non-linéaires). En pratique, on choisit généralement la fonction sigmoïdale *tangente-hyperbolique* définie par:

$$\forall v \in \Re \ , \quad \varphi(v) = a.\tanh(b.v) = a.\left(\frac{1-e^{-2.b.v}}{1+e^{-2.b.v}}\right) = \frac{2.a}{1+e^{-2.b.v}} - a$$

où a et b sont des constantes réelles.

Les principales propriétés de cette fonction sont les suivantes:

- Elle est assymétrique, c'est-à-dire:

$$\forall v \in \Re \ , \quad \varphi(-v) = -\varphi(v)$$

- Sa plage de variation est limitée par l'intervalle [-a;a].

- Sa fonction dérivée $\varphi'(v)$ s'exprime par:

$$\varphi'(v) = \frac{d\varphi(v)}{dv} = a.b.\left[1 - \tanh(b.v)\right].\left[1 + \tanh(b.v)\right] = \frac{b}{a}.\left[a - \varphi(v)\right].\left[a + \varphi(v)\right]$$

Donc, pour tout $v \in \Re$, la valeur de $\varphi'(v)$ peut être calculée à partir de celle de $\varphi(v)$. On verra dans le paragraphe suivant que cette dernière propriété simplifie énormément l'implantation de l'algorithme de rétro-propagation destiné à ajuster les poids synaptiques et seuils des neurones constituant le perceptron multi-couche.

2.2 - Perceptrons multi-couches:

Les perceptrons multi-couches (MLP) constituent une classe importante de réseaux de neurones artificiels. Ils sont les plus utilisés dans les applications concernant l'identification et le réglage des systèmes dynamiques non-linéaires.

2.2.1 - Structure générale des perceptrons multi-couches:

Tout perceptron multi-couches a les trois caractéristiques suivantes:

- le modèle de chaque neurone constituant le réseau est conforme à celui de la figure 2.2; la fonction d'activation φ doit être continue et différentiable sur tout $\Re$; une forme assez commune vérifiant ces critères est la fonction tangente-hyperbolique définie par:

$$\forall v \in \Re \ , \quad \varphi(v) = a.\tanh(b.v) = \frac{2.a}{1 + e^{-2.v.b}} - a$$

où les constantes a et b ont respectivement comme valeurs typiques [22]:

$$a = 1.716 \quad \text{et} \quad b = 2/3$$

- le réseau comporte une ou plusieurs couches de *neurones cachés* qui ne font pas partie des couches d'entrée et de sortie du réseau; la présence des *couches cachées*, dont l'effet est d'augmenter le degré de liberté du réseau, permet à celui-ci d'apprendre des tâches complexes; plus cette complexité s'accentue, plus le nombre des couches cachées à introduire dans le réseau s'avère important; cependant, une augmentation du nombre des couches cachées entraîne une diminution de la rapidité d'apprentissage du réseau; un compromis précision-rapidité existe donc quant au choix du nombre des couches cachées et des neurones les constituant;

- le réseau expose un haut degré de *connectivité* déterminé par ses synapses; tout changement dans la connectivité du réseau exige une variation dans la population des liens synaptiques ou leurs poids.

D'après la structure du perceptron multi-couche, les ports ou noeuds constituant la couche d'entrée représentent les signaux d'entrée relatifs à la première couche cachée. De même, les signaux de sortie débités par chaque couche cachée correspondent aux signaux d'entrée de la couche suivante.

Le réseau est dit *entièrement connecté* si tout noeud, que ce soit un port d'entrée ou un neurone caché, est relié à tous les neurones de la couche suivante (figure 2.3). Dans le cas contraire, on dit que le réseau est *partiellement connecté* (figure 2.4).

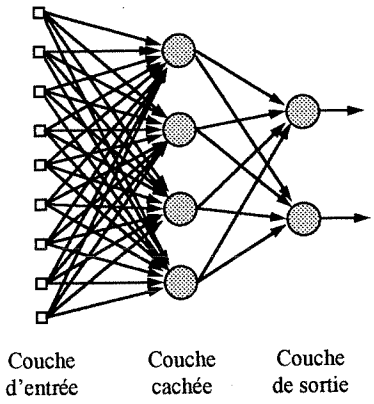

Figure 2.3: Perceptron à une couche cachée entièrement connecté

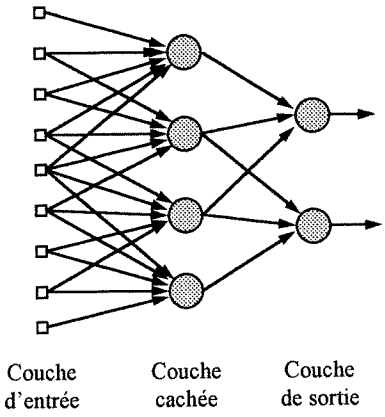

Figure 2.4: Perceptron à une couche cachée partiellement connecté

2.2.2 - Perceptron multi-couches utilisé comme approximateur:

On donne à la figure 2.5 le schéma détaillé d'un perceptron à trois couches (deux couches cachées et une couche de sortie). La fonction de transfert globale du réseau,

déterminée par la mise en cascade de plusieurs blocs fonctionnels élémentaires comme le montre la figure 2.6, s'écrit sous la forme suivante:

$$\mathbf{y} = \Phi[W_1.\Phi[W_2.\Phi[W_3.\mathbf{x}]]] = \hat{f}(\mathbf{x})$$

où $\mathbf{x}$ et $\mathbf{y}$ sont les vecteurs d'entrée et de sortie, de dimensions respectives p et m, W_i, $i = 1,2,3$, les vecteurs-poids ajustables et Φ la fonction matricielle définie par:

$$\Phi\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_l \end{bmatrix} = \begin{bmatrix} \varphi(v_1) \\ \varphi(v_2) \\ \vdots \\ \varphi(v_l) \end{bmatrix} \quad , \quad \forall (v_1, v_2, \ldots, v_l) \in \Re^l$$

D'une manière générale, pour un perceptron à N couches, on a:

$$\mathbf{y} = \hat{f}(\mathbf{x}) = \Phi[W_1.\Phi[W_2.\cdots\Phi[W_N.\mathbf{x}]\cdots]]$$

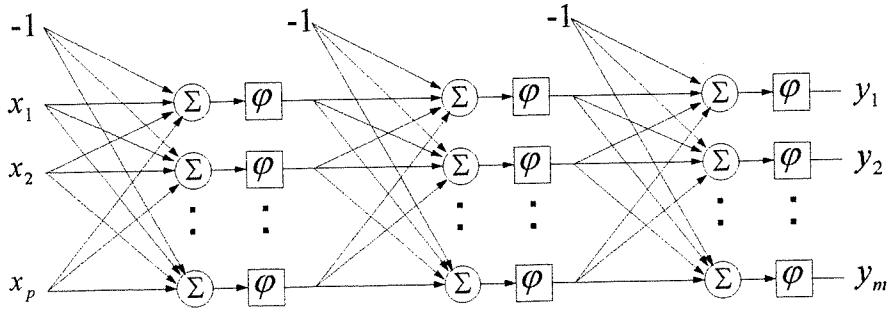

Figure 2.5: Perceptron à trois couches

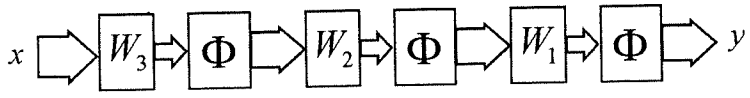

Figure 2.6: Schéma bloc d'un perceptron à trois couches

D'après le théorème de Weierstrass (qui sera vu d'une façon plus détaillée dans la troisième partie du présent rapport), toute fonction $f: \Re^p \rightarrow \Re^m$ peut être approximée, avec un haut degré de précision, par un polynôme tel que:

$$\forall k = 1,\ldots,m \ , \quad f_k(x_1,x_2,\ldots,x_p) \cong \alpha_{k,0} + \sum_i \alpha_{k,i}.x_i + \sum_i \sum_j \alpha_{k,i,j}.x_i.x_j + \cdots$$

où $\alpha_{k,0}$, $\alpha_{k,i}$ et $\alpha_{k,i,j}$ sont des nombres réels.

D'une manière similaire, si $\phi_i(x_1,x_2,\ldots,x_p)$, $i = 1,2,\ldots,n$, sont les éléments d'un ensemble complet orthonormal, la fonction f peut être exprimée par:

$$f(x_1,x_2,\ldots,x_p) \cong \sum_{i=1}^{n} c_i.\phi_i(x_1,x_2,\ldots,x_p)$$

où $c_i \in \Re$, pour tout $i = 1,\ldots,n$.

En se basant sur le théorème précédent, K. Hornik, M. Stinchombe et H. White [17] ont récemment montré qu'un perceptron à deux couches, dont la couche cachée contient arbitrairement un grand nombre de neurones, peut approximer toute fonction f continue sur un sous-ensemble compact de $\Re^p$.

2.2.3 - Algorithme de rétro-propagation classique:

2.2.3.1 - Principe:

L'algorithme de rétro-propagation constitue une méthode itérative d'ajustement des paramètres d'un réseau MLP, basée sur le principe de minimisation d'une certaine fonction d'erreur mesurée à la sortie du réseau.

En notant respectivement $y_j(n)$ et $y_{d,j}(n)$ les valeurs à l'instant n des signaux réel et désiré à la sortie j du réseau, l'erreur de sortie correspondante est donnée par:

$$e_j(n) = y_j(n) - y_{d,j}(n)$$

On définit la valeur instantanée de l' *erreur quadratique* du neurone j par $\frac{1}{2}e_j^2(n)$. De même, la valeur instantanée $\xi(n)$ de la somme quadratique d'erreurs est obtenue en additionnant les termes $\frac{1}{2}e_j^2(n)$ correspondant à tous les neurones de la couche de sortie; ceux-ci sont les seuls neurones accessibles dont on peut calculer les signaux d'erreur.
Ainsi, la somme quadratique instantanée d'erreurs, relative à un réseau donné, a pour expression:

$$\xi(n) = \frac{1}{2}.\sum_{j \in C} e_j^2(n) \qquad (9)$$

où C est l'ensemble des indices des neurones de sortie.

Soit N_e le nombre total des données $\left(\mathbf{x}(i), \mathbf{y}_d(i)\right)$, $i = 1,\ldots,N_e$, utilisées dans la phase d'apprentissage du réseau. On définit l' *erreur quadratique moyenne* par:

$$\xi_{av} = \frac{1}{N_e}.\sum_{n=1}^{N_e}\xi(n)$$

La somme quadratique instantanée d'erreurs $\xi(n)$ et, par conséquent, l'erreur quadratique moyenne ξ_{av} dépendent de tous les paramètres libres (poids synaptiques et seuils) du réseau. ξ_{av}, qui peut être considérée comme une *fonction coût*, permet de mesurer la performance de l'apprentissage relatif à l'ensemble des données choisi pour cette fin.
L'objectif du procédé d'apprentissage est d'ajuster les paramètres libres du réseau afin de minimiser ξ_{av}. Ceci constitue le principe sur lequel se base l'algorithme de rétro-propagation. En effet, selon cet algorithme, la correction $\Delta w_{ji}(n)$ appliquée à $w_{ji}(n)$ à l'instant n est définie par la *loi delta*:

$$\Delta w_{ji}(n) = -\eta.\frac{\partial\xi(n)}{\partial w_{ji}(n)} \qquad (10)$$

où η est une constante positive représentant le *paramètre d'apprentissage* de l'algorithme. Notons que, dans l'expression précédente, w_{ji} n'est autre que le poids associé à la synapse reliant les noeuds d'indices i et j.
Si on considère tous les paramètres libres du réseau, on aboutit à l'expression condensée suivante:

$$\Delta W(n) = -\eta.\nabla_{W(n)}\xi(n)$$

où $W(n)$ est la matrice des poids synaptiques et seuils du réseau, et ∇ l'opérateur gradient. Les paramètres du réseau sont ajustés dans un sens opposé à celui du gradient de la fonction d'erreur. On dit alors que l'ajustement des paramètres se fait conformément à la méthode de la *descente du gradient*.

En développant l'équation (10) après avoir remplacé $\xi(n)$ par son expression donnée en (9), on aboutit à:

$$\Delta w_{ji}(n) = \eta.\delta_j(n).y_i(n) \qquad (11)$$

où:

$$\delta_j(n) = -\varphi'_j\left(v_j(n)\right).e_j(n) \quad \text{si } j \text{ correspond à un neurone de sortie}$$

et:

$$\delta_j(n) = \varphi'_j\left(v_j(n)\right).\sum_k \delta_k(n).w_{kj}(n) \quad \text{si } j \text{ correspond à un neurone caché}$$

Pour un neurone j donné, $\delta_j(n)$ est le *gradient local* qui lui est associé, $v_j(n)$ son niveau d'activité interne, $y_i(n)$ son i^{eme} signal d'entrée, φ'_j la dérivée de la fonction d'activation φ_j par rapport à son argument $v_j(n)$, et $e_j(n)$ le signal d'erreur mesuré à sa sortie, qui n'est autre que la différence entre le signal donné par le neurone et celui désiré.
Dans la deuxième expression de $\delta_j(n)$, correspondant au cas où le neurone j est caché, la sommation d'indice k regroupe tous les neurones k qui sont reliés au neurone j (par des synapses de poids w_{kj}) et qui appartiennent à la couche en aval de celle dont le neurone j fait partie. Si le neurone j appartient à la l^{eme} couche, les neurones k sont ceux de la $(l+1)^{eme}$ couche en liaison avec le neurone j.

2.2.3.2 - Représentation graphique de l'algorithme:

Les expressions du gradient local données dans (11) peuvent être représentées schématiquement, comme le montre la figure 2.7 dans le cas d'un réseau à trois couches. Le fonctionnement du réseau considéré comporte deux phases:

- dans la première phase, correspondant à la partie supérieure du schéma, le sens de propagation de l'information dans le réseau est *direct*, c'est-à-dire de la couche d'entrée vers la couche de sortie; cette propagation se fait couche par couche, et se manifeste par le calcul du vecteur à la sortie de chaque couche à partir du vecteur d'entrée correspondant;

- la deuxième phase, qui correspond à la partie inférieure du graphe, concerne la rétro-propagation (propagation dans le sens inverse) des signaux d'erreur évalués à la sortie du réseau.

Les parties supérieure et inférieure du graphe de la figure 2.7 représentent respectivement le *réseau principal* et le *réseau de sensibilité* associés au perceptron multi-couche considéré.

Du point de vue notation, on représente par $\mathbf{y}^{(l)}$ le vecteur de sortie correspondant à la l^{eme} couche du réseau. Si celui-ci comporte L couches, $\mathbf{y}^{(0)}$ et $\mathbf{y}^{(L)}$ ne sont autres que les vecteurs d'entrée et de sortie (notés respectivement **x** et **o** sur la figure 2.7).

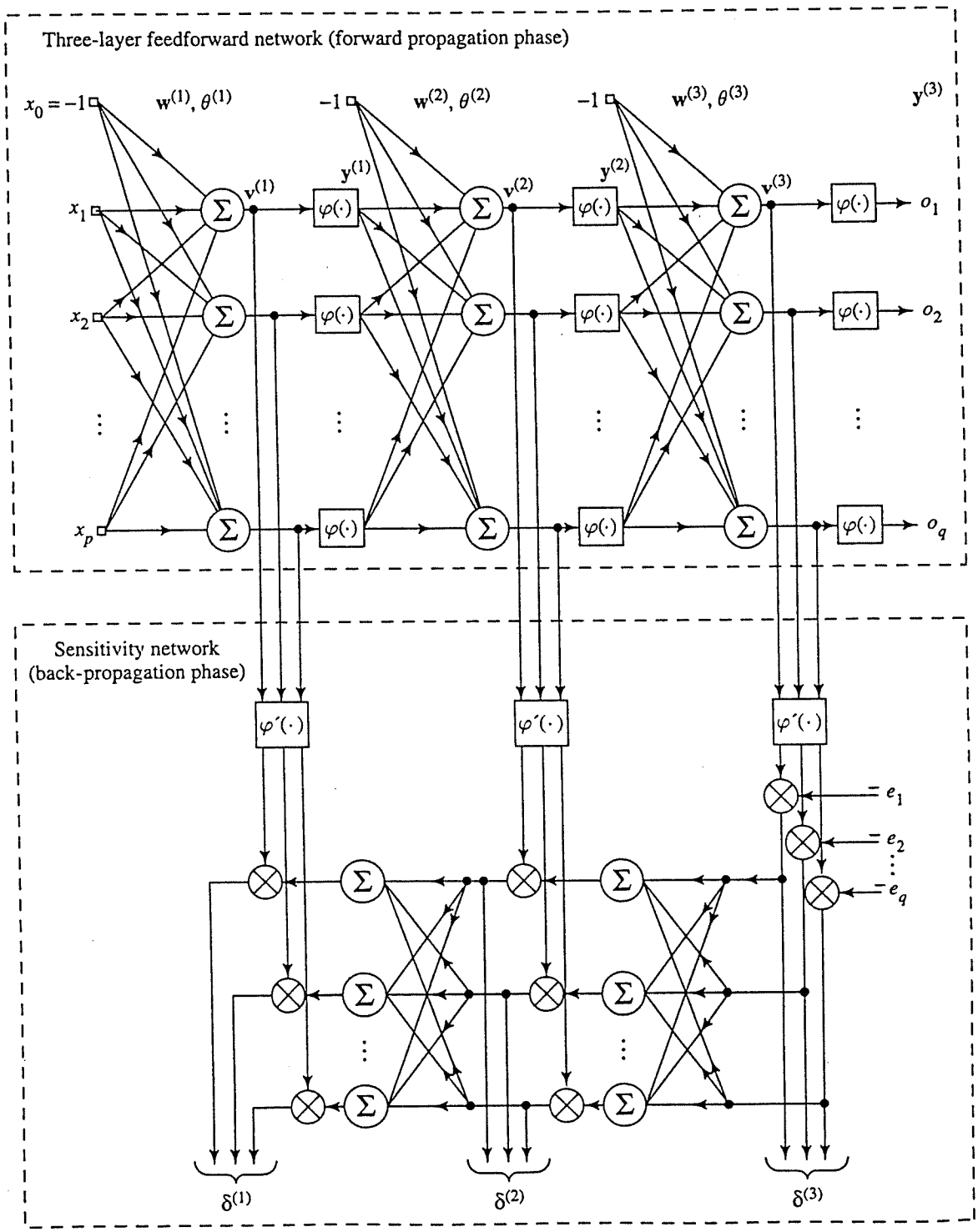

Figure 2.7: Représentation graphique de l'algorithme de rétro-propagation associé à un perceptron à trois couches [14]

2.2.3.3 - Amélioration de l'algorithme:

a) Notion de momentum:

Afin d'améliorer les performances du processus d'apprentissage (augmenter sa rapidité, diminuer le risque de convergence vers un minimum local) auquel est soumis un perceptron multi-couche donné, on a tendance à augmenter le paramètre d'apprentissage η. Cependant, une grande valeur de η peut rendre le réseau instable (régime oscillatoire). Un compromis existe donc entre la rapidité d'apprentissage du réseau et sa stabilité.

Une méthode simple, permettant d'accélérer l'apprentissage du réseau sans toutefois nuire à sa stabilité, consiste à inclure dans la loi delta donnée par (11) un nouveau terme appelé *momentum*. La loi delta modifiée s'écrit:

$$\Delta w_{ji}(n) = \alpha.\Delta w_{ji}(n-1) + \eta.\delta_j(n).y_i(n) \qquad (12)$$

où α, appelé *coefficient du momentum*, est un nombre réel positif. L'équation (12) constitue la *loi delta généralisée*; elle est équivalente à la loi delta dans le cas où α est nul.

b) Choix du paramètre d'apprentissage:

Il est préférable que tous les neurones constituant le perceptron multi-couche puissent avoir la même vitesse d'apprentissage.
D'une manière générale, les dernières couches (celles voisines de la couche de sortie) tendent à avoir des gradients locaux supérieurs à ceux des premières couches (celles voisines de la couche d'entrée). Par conséquent, la valeur donnée au paramètre d'apprentissage η doit être plus faible pour les dernières couches que les premières.
De plus, plus le nombre de synapses associées à un neurone est grand, plus le paramètre d'apprentissage relatif à ce neurone doit être faible.

2.2.3.4 - Implantation de l'algorithme:

L'implantation de l'algorithme de rétro-propagation suit les étapes suivantes:

a) Initialisation:

On commence par choisir une configuration raisonnable du réseau: on fixe d'abord le nombre des couches cachées et des neurones les constituant, puis on donne aux différents paramètres libres (poids synaptiques et seuils) du réseau des valeurs initiales aléatoires uniformément distribuées à l'intérieur de l'intervalle:

$$I = \left[-\frac{2.4}{F_j}, +\frac{2.4}{F_j} \right]$$

F_j étant le nombre total des synapses (ou *fan-in*) associées au neurone j.

L'initialisation des paramètres est donc effectuée au niveau de chaque neurone. Notons que l'intervalle I a été construit en supposant implicitement que la fonction d'activation φ de chaque neurone est la fonction tangente-hyperbolique définie par:

$$\forall v \in \Re \ , \quad \varphi(v) = a.\tanh(b.v) = \frac{2.a}{1+e^{-2.b.v}} - a$$

où $a = 1.716$ et $b = 2/3$.

b) Présentation des données d'apprentissage:

On fait présenter au réseau, d'une manière successive et dans un ordre aléatoire, les N_e éléments $(\mathbf{x}(i), \mathbf{y}_d(i))$, $i = 1,\ldots,N_e$, d'un ensemble de données préalablement choisi. Rappelons que $\mathbf{x}(i)$ est le i^{eme} vecteur d'entrée et $\mathbf{y}_d(i)$ le vecteur de sortie désiré qui lui correspond. Pour chaque couple $(\mathbf{x}(i), \mathbf{y}_d(i))$, on applique les deux points suivants [c) et d)].

c) Propagation directe de l'information:

On détermine, en commençant par la première couche, les différents signaux et potentiels d'activation du réseau. Le niveau d'activité interne du neurone j dans la couche l est:

$$v_j^{(l)}(n) = \sum_{i=0}^{p} w_{ji}^{(l)}(n).y_i^{(l-1)}(n)$$

où $y_i^{(l-1)}(n)$ est la valeur à l'instant n du signal provenant du noeud i dans la couche précédente (l-1), et $w_{ji}^{(l)}(n)$ le poids de la synapse reliant les neurones i et j entre eux. Il est à noter que, pour $l = 1$, $y_i^{(0)}(n)$ n'est autre que $x_i(n)$.
Pour $i = 0$, on a:

$$y_0^{(l-1)}(n) = -1 \qquad \text{et} \qquad w_{j0}^{(l)}(n) = \theta_j^{(l)}(n)$$

où $\theta_j^{(l)}(n)$ est le seuil relatif au neurone j.

Le signal de sortie du neurone j dans la couche l est donné par:

$$y_j^{(l)}(n) = \varphi\left(v_j^{(l)}(n)\right) = a.\tanh\left(b.v_j^{(l)}(n)\right)$$

Si le neurone j appartient à la couche de sortie L, le signal d'erreur qui lui correspond se calcule comme suit:

$$e_j(n) = y_j^{(L)}(n) - y_{d,j}(n)$$

où $y_{d,j}(n)$ est la j^{eme} composante du vecteur de sortie désiré à l'instant n.

d) Rétro-propagation de l'erreur:

On calcule, en commençant par la dernière couche, les gradients locaux du réseau donnés par:

- $$\delta_j^{(L)}(n) = -e_j^{(L)}(n).\varphi'\left(v_j^{(L)}(n)\right)$$
$$= -\frac{b}{a}.e_j^{(L)}(n).\left[a - \varphi\left(v_j^{(L)}(n)\right)\right].\left[a + \varphi\left(v_j^{(L)}(n)\right)\right]$$
$$= -\frac{b}{a}.e_j^{(L)}(n).\left[a - y_j^{(L)}(n)\right].\left[a + y_j^{(L)}(n)\right]$$

pour tout neurone j appartenant à la couche de sortie L;

- $$\delta_j^{(l)}(n) = \varphi'\left(v_j^{(l)}(n)\right).\sum_k \delta_k^{(l+1)}(n).w_{kj}^{(l+1)}$$
$$= \frac{b}{a}.\left[a - y_j^{(l)}(n)\right].\left[a + y_j^{(l)}(n)\right].\sum_k \delta_k^{(l+1)}(n).w_{kj}^{(l+1)}(n)$$

pour tout neurone j appartenant à la couche cachée l.

Ensuite, on ajuste les poids synaptiques du réseau en utilisant la loi delta généralisée:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha.\left[w_{ji}^{(l)}(n) - w_{ji}^{(l)}(n-1)\right] + \eta.\delta_j^{(l)}(n).y_i^{(l-1)}(n)$$

pour tout $l = 1,\ldots,L$. η est le paramètre d'apprentissage et α le coefficient du momentum.

e) Itération:

On répète les trois étapes précédentes [b), c) et d)] après avoir choisi un nouvel ensemble de données d'apprentissage. Ce procédé s'arrête lorsque les paramètres libres du système atteignent des valeurs stables et l'erreur quadratique moyenne ξ_{av}, évaluée en tenant compte de la totalité des données d'apprentissage, devient minimale ou suffisamment faible.

2.2.3.5 - La procédure de validation croisée:

Telle citée dans [23], la *validation croisée* (ou *cross-validation*) est une méthode (ou protocole) d'entraînement fort utile qui permet d'estimer les performances en généralisation d'un réseau de neurones après apprentissage. Elle consiste d'abord à mesurer l'évolution des performances du réseau sur l'ensemble des données d'apprentissage. En général, ces performances s'améliorent constamment au cours de l'apprentissage. Cependant, cette mesure ne fournit pas à elle seule une évaluation objective des performances réelles du réseau. En effet, il est possible que le réseau apprenne à traiter spécifiquement les informations contenues dans les données d'apprentissage, au détriment de ses performances sur le problème général. Ce phénomène survient souvent en fin d'apprentissage, et se nomme *surapprentissage* ou *apprentissage par coeur*.
Pour éviter ce problème, on mesure l'évolution des performances du réseau sur un second ensemble de données (ensemble *test* ou *de généralisation*), distinct de celui d'apprentissage. Dans la mesure où le réseau apprend effectivement à traiter le problème, ses performances s'améliorent à la fois sur l'ensemble des données d'apprentissage et sur l'ensemble test. S'il commence à apprendre par coeur, ses performances en test se dégraderont. Ce phénomène est illustré par l'exemple graphique donné à la figure 2.8; le point A (correspondant à des performances en test optimales) indique le moment où il serait judicieux d'interrompre l'apprentissage.

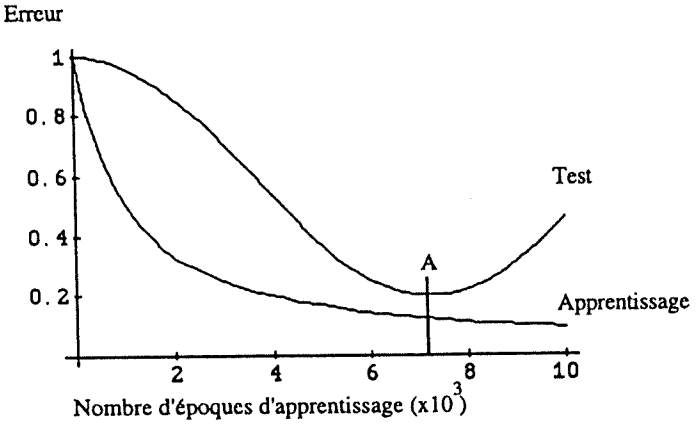

Figure 2.8: Evolution des erreurs moyennes en apprentissage et en test au cours de l'entraînement [23]

Notons que si l'ensemble test n'a pas été appris par le réseau, il a néanmoins été employé pour interrompre l'entraînement. Ceci peut induire un biais favorable dans l'évaluation des performances finales du réseau. Pour éviter ce biais, il est donc nécessaire de mesurer les performances du réseau en fin d'apprentissage sur un troisième ensemble de données (l'ensemble *de validation*). Cette dernière mesure constitue l'évaluation non biaisée des performances réelles du réseau.

La validation croisée procède donc en trois étapes:

- l'apprentissage même, pendant lequel on évalue le comportement du réseau sur l'ensemble des données d'apprentissage;

- une étape de test, où l'apprentissage est suspendu périodiquement, et où l'ensemble test est présenté au réseau (les performances ainsi obtenues constituent une première indication des performances réelles du réseau);

- une étape de validation *a posteriori*, où un troisième ensemble, dit de validation, est présenté au réseau à la fin de l'apprentissage (les performances du réseau sur celui-ci sont employées pour juger du succès ou de l'échec de l'apprentissage).

2.2.4 - Algorithme de Levenberg-Marquardt:

L'algorithme de rétro-propagation classique décrit précédemment est basé sur la méthode de la descente du gradient, selon laquelle les paramètres libres du réseau, tels les poids synaptiques et seuils, sont ajustés dans une direction opposée à celle du gradient de la fonction d'erreur. En pratique, les performances relatives à cette technique sont loin d'être optimales, et leur dégradation s'accentue lorsque la fonction à apprendre par le réseau devient de plus en plus complexe: le temps d'apprentissage devient long et il est très probable que la fonction d'erreur $\xi(n)$ atteigne un minimum local non désiré.

Un nouvel algorithme d'ajustement des paramètres du réseau a été conçu par Levenberg et Marquardt. Il présente par rapport au précédent les deux avantages suivants: une convergence plus rapide des paramètres du réseau et une valeur minimale beaucoup plus faible atteinte par la fonction d'erreur.

Selon l'algorithme de Levenberg-Marquardt, dont la construction est basée sur une approximation de la méthode de Newton, la loi d'ajustement des paramètres est donnée par:

$$\Delta W = -\left(J^T.J + \mu.I\right)^{-1}.J^T.\mathbf{e} \qquad (13)$$

où W est la matrice des paramètres du réseau, $\mathbf{e}$ le vecteur d'erreur défini comme précédemment par:

$$\mathbf{e} = \mathbf{y} - \mathbf{y}_d$$

J la matrice jacobienne des dérivées partielles des signaux d'erreur par rapport à chaque paramètre, I la matrice identité, et μ un nombre réel positif.
Si μ est très grand, l'expression (13) approxime la méthode de la descente du gradient; dans le cas contraire (μ est très faible), elle traduit la méthode de Gauss-Newton.

La méthode de Gauss-Newton devient plus performante (plus rapide et plus précise) que celle de la descente du gradient lorsque la fonction d'erreur est proche de son minimum. Il est conseillé donc de réduire la valeur de μ après chaque itération réussie (qui résulte en une diminution de la fonction d'erreur), et de l'aggrandir dans le cas contraire (lorsqu'une itération résulte en une augmentation de la fonction d'erreur).

2.3 - Perceptrons multi-couches généralisés:

Dans les problèmes d'identification des systèmes dynamiques partiellement et non totalement inconnus, l'identificateur utilisé est constitué, non pas d'un seul réseau neuronal, mais d'une association entre un ou plusieurs réseau neuronaux (qui ne sont autres que des perceptrons multi-couches) et un ou plusieurs blocs linéaires. On obtient ainsi une nouvelle classe de réseaux neuronaux, qui est celle des *perceptrons multi-couches généralisés*.
L'application de l'algorithme de rétro-propagation classique, pour l'ajustement des paramètres libres d'un perceptron multi-couche généralisé, s'avère insuffisante; d'où la nécessité de concevoir un nouveau type d'algorithme pour cette classe de réseaux.

D'après K. S. Narendra et K. Parthasarathy [9], tout réseau de la famille des perceptrons multi-couches généralisés peut être constitué par une combinaison de quatre structures simples, dites de base, appartenant à cette même famille. Celles-ci sont représentées à la figure 2.9. N, $N^{(1)}$ et $N^{(2)}$ représentent des perceptrons multi-couches, et $W(z)$ est une matrice de transfert linéaire discrète.
A chaque structure de base est associé un *algorithme de rétro-propagation dynamique*. Celui-ci, utilisant la méthode de la descente du gradient exposée précédemment, permet l'ajustement de tous les paramètres libres de la structure considérée.
Dans les lignes qui suivent, on décrira séparément les quatres structures de base, ainsi que l'algorithme associé à chacune d'elles.

2.3.1 - Etude de la Structure 1:

D'après cette structure, un perceptron multi-couche (MLP) à p entrées et l sorties, représenté par sa fonction non-linéaire N, est connecté en cascade avec un système multivariable linéaire stationnaire, à l entrées et m sorties, représenté par sa matrice de transfert discrète $W(z)$.

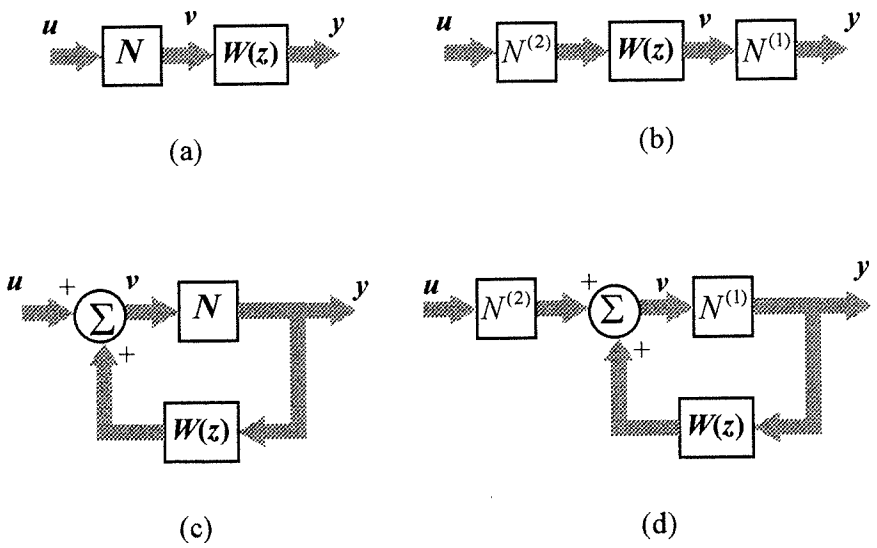

Figure 2.9: Structures de base.
a) Structure 1, b) Structure 2,
c) Structure 3, d) Structure 4

En notant respectivement **u** et **y** les vecteurs d'entrée et de sortie de la structure, on a la relation suivante:

$$\mathbf{y} = W(z).N[\mathbf{u}]$$

L'ajustement des paramètres libres du perceptron multi-couche est basé sur la méthode de la descente du gradient. Si θ_j est un paramètre libre de ce réseau, sa variation $\Delta\theta_j$ à chaque itération est proportionnelle à l'opposé de la dérivée partielle de la fonction d'erreur par rapport à θ_j. Cette fonction d'erreur, notée ξ, est définie à l'instant k par:

$$\xi(k) = \frac{1}{2}.\mathbf{e}^T(k).\mathbf{e}(k) = \frac{1}{2}\sum_{j=1}^{l} e_j^2(k)$$

où $\mathbf{e}(k)$ est le vecteur d'erreur évalué à l'instant k et redéfini comme étant l'écart entre le vecteur de sortie réel débité par la structure et celui désiré:

$$\mathbf{e}(k) = \mathbf{y}(k) - \mathbf{y}_d(k)$$

Ainsi, les ajustements qu'on doit apporter à chaque itération aux différents paramètres libres θ_j du réseau sont entièrement déterminés par la connaissance des termes:

$$\frac{\partial \mathbf{e}(k)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j}\left[\mathbf{y}(k) - \mathbf{y}_d(k)\right] = \frac{\partial \mathbf{y}(k)}{\partial \theta_j} = W(z).\frac{\partial \mathbf{v}(k)}{\partial \theta_j} \qquad (14)$$

où $\mathbf{v}(k) = N[\mathbf{u}(k)]$ est le vecteur de sortie du perceptron multi-couche.

Dans l'expression (14), les valeurs instantanées des termes $\partial \mathbf{v}(k)/\partial \theta_j$ sont générées par le réseau de sensibilité associé au perceptron multi-couche. Chaque vecteur $\partial \mathbf{e}/\partial \theta_j$ peut donc être obtenu à partir d'un système multivariable linéaire de matrice de transfert $W(z)$, dont le vecteur d'entrée est $\partial \mathbf{v}/\partial \theta_j$. On a besoin ainsi d'autant de systèmes multivariables linéaires identiques, caractérisés par $W(z)$, que de paramètres libres θ_j. Ceci permet de définir le *modèle de sensibilité* de la Structure 1, représenté à la figure 2.10.

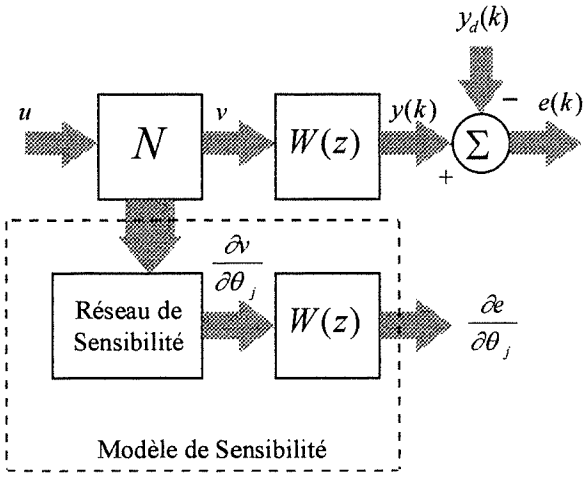

Figure 2.10: Génération du gradient d'erreur dans le cas de la Structure 1

2.3.2 - Etude de la Structure 2:

Dans ce cas, deux réseaux neuronaux de type MLP, représentés par $N^{(1)}$ et $N^{(2)}$, sont connectés en cascade de part et d'autre d'un système multivariable linéaire de matrice de transfert $W(z)$. Cette configuration est illustrée à la figure 2.10.

On note respectivement **u** et **y** les vecteurs d'entrée et de sortie de la structure, de dimensions respectives p et m, et v le vecteur intermédiaire de dimension l à la sortie du bloc linéaire. Dans ces conditions, la relation entrée-sortie caractérisant la structure est donnée par:

$$\mathbf{y} = N^{(1)}[\mathbf{v}] = N^{(1)}\left[W(z).N^{(2)}[\mathbf{u}]\right]$$

Les dérivées partielles des signaux de sortie y_i, $i = 1,\ldots,m$, par rapport à tout paramètre libre $\theta_j^{(1)}$ du réseau $N^{(1)}$ peuvent être évaluées en appliquant à ce dernier l'algorithme de rétro-propagation classique. Par contre, notre intérêt est de déterminer le gradient du vecteur de sortie **y** par rapport aux paramètres libres $\theta_j^{(2)}$ du réseau $N^{(2)}$ en amont du bloc linéaire représenté par $W(z)$. Pour tout $i = 1,\ldots,m$, on peut écrire:

$$\frac{\partial y_i}{\partial \theta_j^{(2)}} = \sum_l \frac{\partial y_i}{\partial v_l} \cdot \frac{\partial v_l}{\partial \theta_j^{(2)}}$$

La dérivée partielle $\partial y_i / \partial \theta_j^{(2)}$ s'obtient donc par la somme de l produits de deux termes: $\partial y_i / \partial v_l$ et $\partial v_l / \partial \theta_j^{(2)}$. Ceux-ci peuvent être déterminés par des méthodes connues jusqu'à présent. En effet, $\partial y_i / \partial v_l$ peut être donné par l'algorithme de rétro-propagation classique appliqué au réseau $N^{(2)}$, et la génération de $\partial v_l / \partial \theta_j^{(2)}$ est conforme à la méthode décrite précédemment (lors de l'étude de la Structure 1). La figure 2.11 illustre la méthode de génération du gradient de sortie par rapport aux paramètres libres du réseau neuronal $N^{(2)}$, et définit le modèle de sensibilité relatif à la structure considérée.

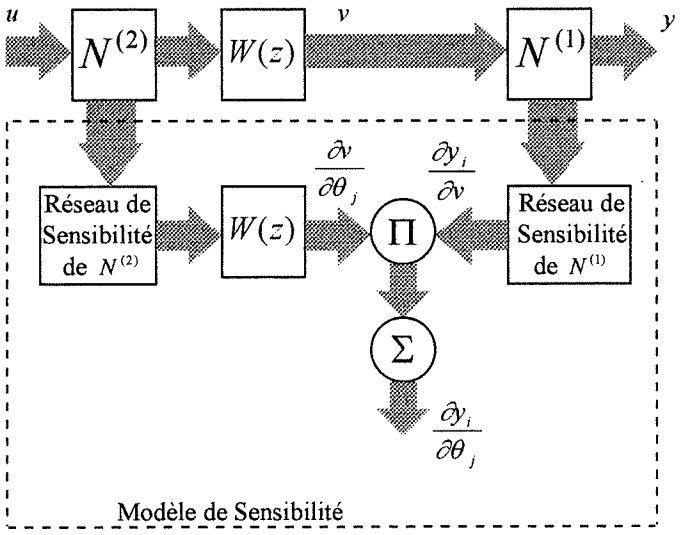

Figure 2.11: Génération du gradient de sortie dans le cas de la Structure 2

2.3.3 - Etude de la Structure 3:

La structure 3 est représentée par la partie de gauche du schéma de la figure 2.12. Un réseau neuronal N de type MLP est asservi à travers un système multivariable linéaire défini par sa matrice de transfert $W(z)$.

Les vecteurs d'entrée et de sortie du système asservi non-linéaire sont notés respectivement **u** et **y**. Leurs dimensions sont supposées égales respectivement à p et m. Ces deux vecteurs sont liés l'un à l'autre par la relation suivante:

$$\mathbf{y} = N[\mathbf{v}] = N\left[\mathbf{u} + W(z).\mathbf{y}\right] \tag{15}$$

où **v** est un vecteur intermédiaire de dimension p défini par:

$$\mathbf{v} = \mathbf{u} + W(z).\mathbf{y}$$

Contrairement aux deux cas précédents, l'expression entrée-sortie (15) de la Structure 3 montre que le vecteur de sortie **y** dépend implicitement (dû au terme **v**) et explicitement (dû à la fonction N) des paramètres libres θ_j du réseau MLP. Par conséquent, on s'intéressera non pas à la dérivée partielle des signaux de sortie y_i, $i = 1,\ldots,m$, par rapport aux paramètres θ_j, mais à leur dérivée totale notée $dy_i/d\theta_j$ et donnée sous forme condensée par:

$$\frac{d\mathbf{y}}{d\theta_j} = \frac{\partial \mathbf{y}}{\partial \mathbf{v}} \cdot \frac{d\mathbf{v}}{d\theta_j} + \frac{\partial \mathbf{y}}{\partial \theta_j}$$

En remplaçant **y** par son expression donnée en (15), on aboutit à:

$$\frac{d\mathbf{y}}{d\theta_j} = \frac{\partial N[\mathbf{v}]}{\partial \mathbf{v}} \cdot \frac{d\mathbf{v}}{d\theta_j} + \frac{\partial N[\mathbf{v}]}{\partial \theta_j} = \frac{\partial N[\mathbf{v}]}{\partial \mathbf{v}} \cdot W(z) \cdot \frac{d\mathbf{y}}{d\theta_j} + \frac{\partial N[\mathbf{v}]}{\partial \theta_j} \tag{16}$$

L'expression (16) représente une équation aux différences linéarisée dont les coefficients varient avec le temps. Le procédé dynamique, par lequel le gradient total $d\mathbf{y}/d\theta_j$ est généré, est montré à la figure 2.12. Le vecteur $\partial N[\mathbf{v}]/\partial \theta_j$ et la matrice jacobienne $\partial N[\mathbf{v}]/\partial \mathbf{v}$ sont évalués à chaque instant à partir de l'algorithme de rétro-propagation classique associé au réseau neuronal N.

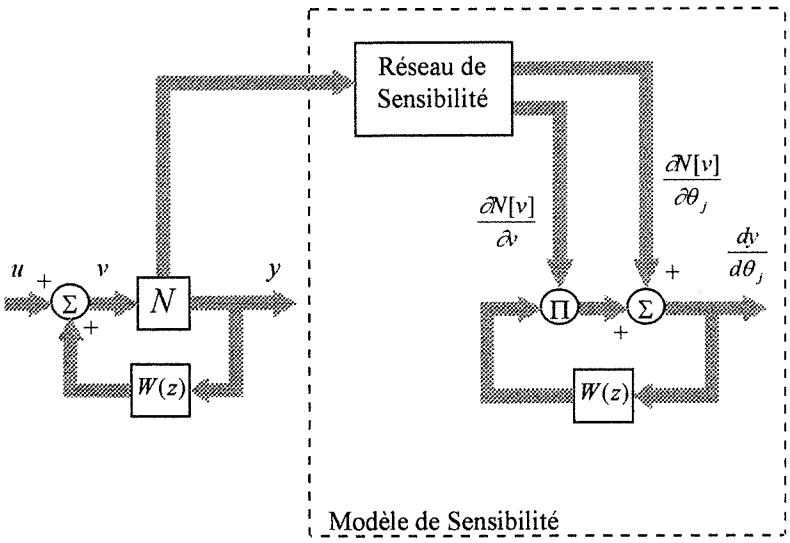

Figure 2.12: Génération du gradient total de sortie dans le cas de la Structure 3

2.3.4 - Etude de la Structure 4:

Le système asservi décrit précédemment est précédé dans ce cas par un réseau MLP noté $N^{(2)}$. La présence de $N^{(2)}$ n'affecte pas la génération des dérivées partielles des signaux de sortie par rapport aux paramètres libres du réseau $N^{(1)}$ présent dans la chaîne directe du système asservi, auquel cas la méthode établie pour la Structure 3 est adoptée. Par contre, pour déterminer le gradient du vecteur de sortie **y** par rapport aux paramètres du réseau $N^{(2)}$, on utilise la procédure décrite ci-dessous.

Sachant que la relation entrée-sortie relative à la Structure 4 est:

$$\mathbf{y} = N^{(1)}[\mathbf{v}] = N^{(1)}\left[N^{(2)}[\mathbf{u}] + W(z).\mathbf{y}\right] \tag{17}$$

u étant le vecteur d'entrée et **v** le vecteur intermédiaire défini par:

$$\mathbf{v} = N^{(2)}[\mathbf{u}] + W(z).\mathbf{y}$$

on a:

$$\frac{\partial \mathbf{y}}{\partial \theta_j} = \frac{\partial N^{(1)}[\mathbf{v}]}{\partial v} \cdot \frac{\partial v}{\partial \theta_j} = \frac{\partial N^{(1)}[\mathbf{v}]}{\partial v} \cdot \left[\frac{\partial N^{(2)}[\mathbf{u}]}{\partial \theta_j} + W(z) \cdot \frac{\partial \mathbf{y}}{\partial \theta_j}\right] \tag{18}$$

où θ_j représente un paramètre libre du réseau $N^{(2)}$.

Comme le montre la figure 2.13, $\partial y/\partial\theta_j$ peut être considéré comme le vecteur de sortie d'un système non-linéaire semblable à la Structure 3, ayant $\partial N^{(2)}[\mathbf{u}]/\partial\theta_j$ pour vecteur d'entrée. Le réseau neuronal N est alors remplacé par un bloc de gain $\partial N^{(1)}[\mathbf{v}]/\partial v$ linéarisé autour du point de fonctionnement courant. Notons que le vecteur $\partial N^{(2)}[\mathbf{u}]/\partial\theta_j$ et la matrice jacobienne $\partial N^{(1)}[\mathbf{v}]/\partial v$ sont générés à chaque instant par les algorithmes de rétro-propagation classiques relatifs à $N^{(2)}$ et $N^{(1)}$ respectivement.

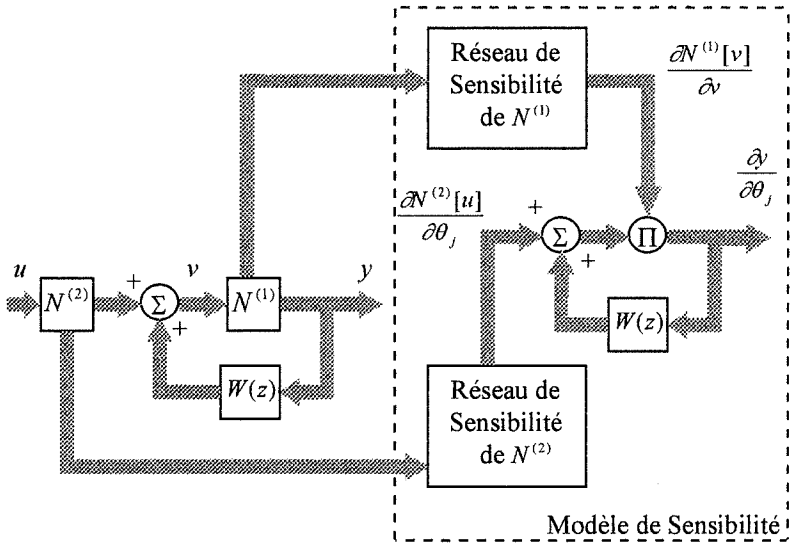

Figure 2.13: Génération du gradient de sortie dans le cas de la Structure 4

2.4 - Réseaux à base radiale:

On étudie dans ce paragraphe une autre classe de réseaux neuronaux artificiels utilisés comme approximateurs universels. Le principe sur lequel se repose ce type de réseaux consiste à approximer toute fonction réelle continue $f:\Re^p \rightarrow \Re$ par une somme pondérée de plusieurs fonctions dites *à base radiale*, de sorte qu'on peut écrire:

$$\forall \mathbf{x} \in \Re^p \ , \quad f(\mathbf{x}) \cong \sum_j w_j . \varphi_j(\mathbf{x}) = \hat{f}(\mathbf{x})$$

où, pour tout $j = 1,2,\ldots$, $\varphi_j(\mathbf{x})$ est une fonction à base radiale et w_j la constante réelle associée.

Les premiers travaux sur ce sujet furent amorcés par M. J. D. Powell [24]. Il est devenu aujourd'hui l'un des principaux domaines de recherche en analyse numérique. D. S. Broomhead et D. Lowe [25] ont été les premiers à exploiter l'utilisation des fonctions à

base radiale dans la conception des réseaux neuronaux. Autres contributions majeures au niveau de la théorie, la conception et l'application des réseaux à base radiale (RBF) ont été données par J. E. Moody et C. J. Darken [26], S. Renals [27], et T. Poggio et F. Girosi [28].

2.4.1 - Structure d'un réseau à base radiale:

Un réseau RBF est formé de trois couches différentes:

- une couche d'entrée qui comporte les noeuds de source;

- une couche cachée de très grande dimension, qui joue un rôle différent de celui des couches cachées faisant partie des perceptrons multi-couches;

- un neurone de sortie, qui débite le signal de réponse du réseau.

La figure 2.14 donne une description d'un réseau à base radiale typique. Le signal donné par le neurone de sortie est une *combinaison linéaire* de ceux provenant de la couche cachée. Par contre, ces derniers sont générés à partir des signaux d'entrée par une transformation non-linéaire. Tout neurone j appartenant à la couche cachée génère à sa sortie un signal de la forme:

$$\begin{aligned}\varphi_j(\mathbf{x}) &= G\left(\left\|\mathbf{x}-\mathbf{t}_j\right\|_{C_j}\right) \\ &= \exp\left[-\left(\mathbf{x}-\mathbf{t}_j\right)^T.C_j^T.C_j.\left(\mathbf{x}-\mathbf{t}_j\right)\right] \\ &= \exp\left[-\frac{1}{2}\cdot\left(\mathbf{x}-\mathbf{t}_j\right)^T\cdot\Sigma_j^{-1}\cdot\left(\mathbf{x}-\mathbf{t}_j\right)\right] \qquad (19)\end{aligned}$$

où $\mathbf{x}$ est le vecteur d'entrée de dimension p, $\|\cdot\|_{C_j}$ la norme euclidienne généralisée, dite *pondérée*, définie par:

$$\forall \mathbf{v}\in\Re^p\ ,\quad \|\mathbf{v}\|_{C_j}^2=\left(C_j.\mathbf{v}\right)^T.\left(C_j.\mathbf{v}\right)=\mathbf{v}^T.C_j^T.C_j.\mathbf{v}$$

C_j une *matrice de pondération*, de dimension (p,p), G la fonction gaussienne de centre $\mathbf{t}_j$, associée à la matrice C_j, et Σ_j la *matrice de covariance* de G, définie par:

$$\Sigma_j=\tfrac{1}{2}\cdot\left[C_j^T.C_j\right]^{-1}$$

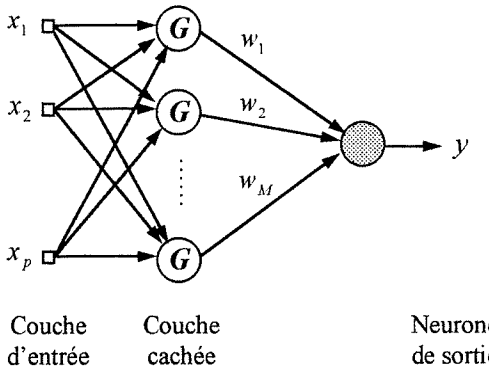

Figure 2.14: Structure de base d'un réseau RBF

Le signal de sortie y du réseau est alors donné par:

$$y = \hat{f}(\mathbf{x}) = \sum_{j=1}^{M} w_j . \varphi_j(\mathbf{x})$$

où M est le nombre de neurones cachés et, pour tout $j = 1,\ldots,M$, w_j est le poids synaptique associé à la $j^{ème}$ entrée du neurone de sortie.

2.4.2 - Stratégies d'apprentissage:

Il existe, en réalité, plusieurs stratégies qui, tout en ajustant d'une manière itérative les paramètres libres du réseau RBF considéré, visent à faire apprendre à celui-ci une certaine tâche. L'ajustement des paramètres dépend de l'ensemble des N_e données $\left(\mathbf{x}(i), y_d(i)\right)$, $i = 1,\ldots,N_e$, choisies pour entraîner le réseau. Deux parmi ces stratégies d'apprentissage, caractérisées par leur simplicité d'implantation, seront considérées.

2.4.2.1 - Stratégie des centres fixes:

L'approche la plus simple consiste à considérer des fonctions à base radiale (fonctions d'activation des neurones cachés) fixes, dont les centres peuvent être choisis *aléatoirement* parmi les $\mathbf{x}(i)$, $i = 1,\ldots,N_e$, de l'ensemble des données d'apprentissage. Comme fonctions à base radiale, on peut utiliser des fonctions gaussiennes *isotropes* dont la déviation standard est fixée par l'étendue des centres choisis. Ainsi, pour tout $j = 1,\ldots,M$, M étant le nombre des centres ou des neurones cachés, on a:

$$\varphi_j(\mathbf{x}) = \exp\left(-\frac{M}{d^2} \cdot \left\|\mathbf{x} - \mathbf{t}_j\right\|^2\right)$$

où d est la distance maximale entre les centres choisis. La déviation standard (largeur) commune à toutes les fonctions gaussiennes est fixée à:

$$\sigma = \frac{d}{\sqrt{2M}}$$

Un tel choix pour la déviation standard σ permet d'avoir des fonctions à base radiale dont la forme n'est ni trop pointue, ni trop plate; ces deux cas extrêmes sont à éviter.
Les seuls paramètres libres, dans cette approche, sont les poids synaptiques w_j, $j = 1,\ldots,M$. Une procédure assez directe permettant l'ajustement de ces paramètres est donnée par la *méthode de la pseudo-inverse* selon laquelle:

$$\mathbf{w} = G^{+}.\mathbf{y}_d$$

où $\mathbf{w} = \left[w_j\right]$ est le vecteur-poids, $\mathbf{y}_d = \left[y_d(i)\right]$ le vecteur des réponses désirées, et G^{+} la matrice pseudo-inverse de G qui est elle-même définie par:

$$G = \left[g_{ij}\right]$$

avec:

$$\forall i = 1,\ldots,N_e \ , \ \forall j = 1,\ldots,M \ , \quad g_{ij} = \exp\left(-\frac{M}{d^2}\cdot\left\|\mathbf{x}(i) - \mathbf{t}_j\right\|^2\right)$$

2.4.2.2 - Sélection supervisée des centres:

Dans cette deuxième approche, les centres des fonctions à base radiale, ainsi que tous les autres paramètres libres du réseau RBF considéré, sont ajustés par un procédé d'apprentissage supervisé, basé sur la correction d'erreur. L'implantation de ce type de procédé fait appel à la méthode de la descente du gradient déjà connue à ce stade.

La première étape dans le développement d'une telle procédure d'apprentissage consiste à définir la valeur instantanée de la fonction coût:

$$\xi(n) = \tfrac{1}{2}\cdot\sum_{i=1}^{N_e} e_i^2(n)$$

où N_e est le nombre des données d'apprentissage $\left(\mathbf{x}(i), y_d(i)\right)$, $i = 1,\ldots,N_e$, utilisées pour entraîner le réseau, et e_i le signal d'erreur défini par:

$$e_i(n) = y_i(n) - y_d(i) = \sum_{j=1}^{M} w_j(n).G\left(\left\|\mathbf{x}(i) - \mathbf{t}_j(n)\right\|_{C_j(n)}\right) - y_d(i)$$

Le but est de déterminer les valeurs des paramètres libres w_j, $\mathbf{t}_j$ et $\Sigma_j = \frac{1}{2}\cdot\left[C_j^T.C_j\right]^{-1}$, $j = 1,\ldots,M$, correspondant à une valeur minimale de ξ. Les résultats de cette minimisation sont donnés ci-dessous:

$\forall j = 1,\ldots,M$,

$$w_j(n+1) = w_j(n) - \eta_1 \cdot \frac{\partial \xi(n)}{\partial w_j(n)}$$
$$= w_j(n) - \eta_1 \cdot \sum_{i=1}^{N_e} e_i(n).G\left(\left\|\mathbf{x}(i) - \mathbf{t}_j(n)\right\|_{C_j(n)}\right) \qquad (20)$$

$$t_j(n+1) = t_j(n) - \eta_2 \cdot \frac{\partial \xi(n)}{\partial t_j(n)}$$
$$= t_j(n) + \eta_2.w_j(n).\sum_{i=1}^{N_e} e_i(n).G'\left(\left\|\mathbf{x}(i) - \mathbf{t}_j(n)\right\|_{C_j(n)}\right).\Sigma_j^{-1}.\left[\mathbf{x}(i) - \mathbf{t}_j(n)\right] \qquad (21)$$

$$\Sigma_j^{-1}(n+1) = \Sigma_j^{-1}(n) - \eta_3 \cdot \frac{\partial \xi(n)}{\partial \Sigma_j^{-1}(n)}$$
$$= \Sigma_j^{-1}(n) - \frac{1}{2}\eta_3.w_j(n).\sum_{i=1}^{N_e} e_i(n).G'\left(\left\|\mathbf{x}(i) - \mathbf{t}_j(n)\right\|_{C_j(n)}\right).Q_{ij}(n) \qquad (22)$$

avec $Q_{ij}(n) = \left[\mathbf{x}(i) - \mathbf{t}_j(n)\right]\cdot\left[\mathbf{x}(i) - \mathbf{t}_j(n)\right]^T$.

Dans les deux dernières expressions, $G'(\cdot)$ dénote la dérivée de la fonction gaussienne $G(\cdot)$ par rapport au carré de son argument:

$$G'\left(\left\|\mathbf{x}(i) - \mathbf{t}_j(n)\right\|_{C_j(n)}\right) = \frac{dG\left(\left\|\mathbf{x}(i) - \mathbf{t}_j(n)\right\|_{C_j(n)}\right)}{d\left(\left\|\mathbf{x}(i) - \mathbf{t}_j(n)\right\|^2_{C_j(n)}\right)}$$

Il s'avère utile de noter les points importants suivants:

- la fonction coût ξ est *convexe* par rapport aux paramètres w_j, mais *non-convexe* par rapport aux centres $\mathbf{t}_j$ et matrices Σ_j^{-1}; dans ce dernier cas, la recherche des valeurs optimales de $\mathbf{t}_j$ et Σ_j^{-1} peut mener à un minimum local non désiré dans l'espace d'état.

- l'ajustement de w_j, $\mathbf{t}_j$ et Σ_j^{-1} utilise des paramètres d'apprentissage différents (η_1, η_2 et η_3 respectivement).

- contrairement à l'algorithme de rétro-propagation, la méthode de la descente du gradient appliquée à un réseau RBF et décrite par les équations (20), (21) et (22) n'implique pas une rétro-propagation d'erreur.

2.5 - Comparaison entre un réseau à base radiale et un perceptron multi-couches:

Le réseau à base radiale (RBF) et le perceptron multi-couches (MLP) sont des exemples de réseaux multi-couches non-linéaires. Ils sont tous les deux considérés comme des approximateurs universels. Cependant, ces deux types de réseaux se diffèrent l'un de l'autre par plusieurs aspects dont les principaux sont cités ci-dessous:

- un réseau RBF possède une seule couche cachée, alors qu'un réseau MLP peut avoir une ou plusieurs couches cachées;

- tous les neurones constituant un réseau MLP, qu'ils soient dans les couches cachées ou celle de sortie, ont un même modèle; par contre, les neurones cachés d'un réseau RBF sont différents, tant au niveau de la structure que la fonction, du neurone de sortie;

- l'opération de chaque neurone caché dans un réseau RBF utilise la norme euclidienne (distance) pondérée entre le vecteur d'entrée et le centre du neurone; par contre, l'opération de chaque neurone caché dans un réseau MLP fait appel au produit scalaire du vecteur d'entrée et celui des poids synaptiques associés au neurone;

- les réseaux MLP permettent d'approximer *globalement* toute fonction continue non-linéaire; par conséquent, ils sont capables de *généraliser* dans les régions où aucune donnée d'apprentissage n'est disponible; par contre, les réseaux RBF, dont le fonctionnement se base entre autres sur les fonctions gaussiennes (non-linéarités exponentielles localisées), ne peuvent approximer que *localement* une fonction continue non-linéaire donnée, mais présentent l'avantage d'être capables d'apprendre rapidement et d'être peu sensibles à l'ordre de présentation des données d'apprentissage.

Troisième Partie

Identification des systèmes dynamiques par réseaux neuronaux

On se propose dans cette partie de décrire les principales méthodes d'identification des systèmes dynamiques non-linéaires, utilisant des réseaux neuronaux artificiels.
Le problème d'identification surgit chaque fois que le système dynamique qu'on désire régler est totalement ou partiellement inconnu, de sorte que sa mise en équations s'avère très difficile, voire quasi-impossible. L'objectif est de construire un *modèle d'identification* convenable qui, lorsqu'il est soumis au même signal d'entrée que le système principal, produit un signal de sortie qui approxime celui donné par le système. En d'autres termes, si on représente par f la fonction analytique réalisée par un système dynamique non-linéaire donné, et si on note respectivement $\mathbf{u}(k)$ et $\mathbf{y}(k)$ les vecteurs d'entrée et de sortie du système à l'instant k (figure 3.1), de sorte que:

$$\forall k \geq 0 \ , \quad \mathbf{y}(k) = f[\mathbf{u}(k)]$$

l'objectif est de déterminer une fonction $\hat{f}$ vérifiant:

$$\forall k \geq 0 \ , \quad \|\hat{\mathbf{y}}(k) - \mathbf{y}(k)\| = \left\|\hat{f}\left[\mathbf{u}(k)\right] - f\left[\mathbf{u}(k)\right]\right\| \leq \varepsilon$$

pour tout $\mathbf{u}$ appartenant à l'espace d'entrée, avec ε un nombre réel positif convenablement choisi. $\|\cdot\|$ dénote une norme définie sur l'espace de sortie, et $\hat{\mathbf{y}} = \hat{f}(\mathbf{u})$ représente le signal de sortie du modèle d'identification. Par suite, l'erreur entre la sortie générée par le modèle et celle observée $\mathbf{y}$ se définit à tout instant k par:

$$\mathbf{e}_i(k) = \hat{\mathbf{y}}(k) - \mathbf{y}(k)$$

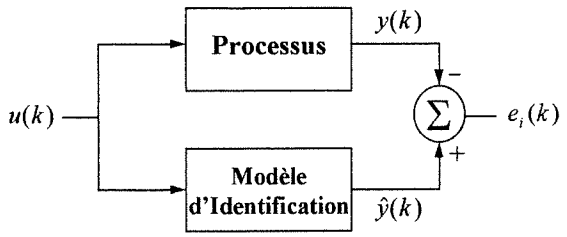

Figure 3.1: Identification d'un processus dynamique

Il existe une vaste littérature sur l'identification des systèmes linéaires. Pour de tels systèmes, si l'ordre est connu, la structure du modèle d'identification peut être fixée de telle sorte que le problème d'identification se ramène tout simplement à celui de l'estimation des paramètres du système. Ceci ne s'applique malheureusement pas au cas des systèmes non-linéaires, où la structure du modèle d'identification doit être justifiée. Le système réel étant totalement ou partiellement inconnu, on assumera toutefois qu'il existe toujours un modèle paramétrique qui peut théoriquement avoir le même comportement que le système. Présentée de cette manière, l'identification est réduite encore une fois à un problème d'estimation des paramètres.
Si le modèle d'identification utilisé comporte entre autres un ou plusieurs réseau(x) de neurones artificiel(s) supposé(s) du type perceptron multi-couche (MLP), la procédure d'identification consiste à ajuster les paramètres libres (poids synaptiques et seuils) du ou des réseau(x) afin d'optimiser une certaine fonction de performance basée sur l'erreur entre les réponses du modèle d'identification et du système principal à un même signal d'entrée. L'ajustement des paramètres est engendré par l'algorithme de rétro-propagation statique ou dynamique décrit dans la partie précédente.
Le procédé d'identification est surtout utilisé dans les travaux de simulation dont le but est l'élaboration d'un régulateur faisant répondre un système donné conformément à certains critères souhaités. Pour aboutir aux résultats voulus, il est nécessaire d'avoir à notre disposition un modèle identifiant le système réel qu'on désire commander. Plus l'écart entre le modèle d'identification et le système réel est faible, mieux seront les performances du régulateur synthétisé.
Dans les lignes qui suivent, quatre modèles de représentation des systèmes dynamiques non-linéaires monovariables, qui furent initialement introduits par K. S. Narendra et K. Parthasarathy [1], seront décrits. Il existe déjà, dans la littérature concernant les systèmes adaptatifs, des modèles équivalents qui sont utilisés pour l'identification des systèmes linéaires. Les modèles qui nous intéressent et qu'on se propose d'étudier dans la suite peuvent être considérés comme une généralisation aux systèmes non-linéaires de ceux qui précèdent.

3.1 - Théorèmes fondamentaux:

3.1.1 - Théorème de Weierstrass:

Soit $C([a;b],\Re)$ l'espace des fonctions réelles continues définies sur l'intervalle $[a;b]$, et définissons la norme d'une fonction $f \in C([a;b],\Re)$ par:

$$\|f\| = \sup_{t\in[a;b]} \{|f(t)|\}$$

D'après le théorème d'approximation de Weierstrass, toute fonction appartenant à $C([a;b],\Re)$ peut être approximée avec une certaine précision par un polynôme. Ainsi, l'ensemble des fonctions polynômiales est dense dans $C([a;b],\Re)$.
Naturellement, le théorème de Weierstrass et sa généralisation aux dimensions multiples trouvent un large domaine d'application, notamment dans les procédures d'approximation des fonctions continues $f:\Re^n \rightarrow \Re^m$ par des polynômes.

3.1.2 - Théorème de Stone-Weierstrass:

Ce théorème, qui constitue une généralisation du théorème de Weierstrass, peut être considéré comme le point de départ pour toute procédure d'approximation des systèmes dynamiques. Il s'énonce comme suit:

Soit $\boldsymbol{U}$ un espace métrique compact. Si $\hat{\Im}$ est une sous-algèbre de $C(\boldsymbol{U},\Re)$ qui contient les fonctions constantes, alors $\hat{\Im}$ est dense dans $C(\boldsymbol{U},\Re)$.

Dans les problèmes qui nous concernent, on supposera que la fonction f caractérisant le processus à identifier est bornée, continue, causale et invariante dans le temps. Si $\hat{\Im}$ satisfait les conditions évoquées par le théorème de Stone-Weierstrass, alors d'après ce théorème il existe toujours une fonction $\hat{f}$ appartenant à $\hat{\Im}$ qui peut être choisie pour approximer toute fonction spécifiée f.

Une vaste littérature existe sur la caractérisation des fonctions non-linéaires et englobe les travaux de Volterra, Wiener, Barret et Urysohn. En utilisant le théorème de Stone-Weierstass, on peut montrer que toute fonction non-linéaire peut être représentée sous certaines conditions par un développement en série correspondant tel que celui de Volterra ou de Wiener. En dépit de ce travail théorique impressionnant, peu de résultats ont été appliqués dans les procédés d'identification des grandes classes de systèmes physiques non-linéaires.

3.1.3 - Théorème d'identification:

Soit Σ un système non-linéaire discret d'ordre n, décrit par les équations d'état suivantes:

$$\Sigma : \begin{cases} \mathbf{x}(k+1) = \Phi[\mathbf{x}(k),\mathbf{u}(k)] \\ \mathbf{y}(k) = \Psi[\mathbf{x}(k)] \end{cases} \qquad (1)$$

où $\mathbf{x}(k) \in \Re^n$, $\mathbf{u}(k) \in \Re^p$ et $\mathbf{y}(k) \in \Re^m$ représentent successivement les vecteurs d'état, d'entrée et de sortie à l'instant k du système, et soit respectivement $\boldsymbol{U}_l$ et $\boldsymbol{Y}_l$ les

ensembles des séquences d'entrée et de sortie de longueur l. Si Σ est *localement fortement observable*, alors il existe localement une fonction continue $\hat{h}: \boldsymbol{Y}_n \times \boldsymbol{U}_n \rightarrow \Re^m$ telle que le modèle entrée-sortie récursif défini par:

$$\mathbf{y}(k+1) = \hat{h}\left[\mathbf{y}(k), \mathbf{y}(k-1), \ldots, \mathbf{y}(k-n+1), \mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-n+1)\right]$$

se comporte de la même manière que le système original Σ. La fonction $\hat{h}$ ainsi définie peut être réalisée par un réseau de neurones artificiels.

Si les conditions de forte observabilité sont satisfaites dans la région d'opération du système, la procédure d'identification utilisant un réseau neuronal devient évidente. A chaque instant, on applique à l'entrée du réseau les n anciennes valeurs des vecteurs d'entrée et de sortie ($2n$ valeurs au total). La sortie du réseau est comparée à celle du système pour engendrer le signal d'erreur. Les paramètres du réseau sont alors ajustés utilisant l'algorithme de rétro-propagation afin de minimiser la somme quadratique d'erreur.

3.2 - Modèles de représentation des systèmes dynamiques non-linéaires:

Comme on l'a déjà mentionné, la capabilité qu'ont les réseaux neuronaux d'approximer d'une manière assez précise de grandes classes de fonctions non-linéaires en fait des principaux candidats à être utilisés dans des modèles dynamiques pour la représentation des processus non-linéaires. L'emploi des algorithmes de rétro-propagation statique et dynamique pour l'ajustement de leurs paramètres rend plus attrayante leur utilisation au sein des identificateurs et des régulateurs.
Dans ce paragraphe, quatre modèles pour la représentation des processus monovariables sont introduits. Ils peuvent facilement être généralisés à des systèmes multivariables. Les modèles d'identification, contenant des réseaux neuronaux (du type perceptron multi-couche ou MLP) comme sous-systèmes, auront une structure semblable à celle des modèles de représentation qui leur sont associés. Ces modèles sont motivés par ceux déjà utilisés pour l'identification et le réglage des systèmes linéaires, et peuvent être considérés comme leur généralisation aux systèmes non-linéaires.
Dans ce qui suit, $u(k)$ et $y(k)$ représentent respectivement les valeurs des signaux d'entrée et de sortie du processus réel à l'instant k, et f, g et h sont des fonctions non-linéaires supposées différentiables par rapport à leurs arguments. Dans les quatre modèles de représentation, qui constituent en réalité des cas particuliers du modèle NARMA déjà vu au paragraphe 1.3.2, la sortie du processus à l'instant k+1 dépend de ses n anciennes valeurs $y(k-i)$, $i = 0,1,\ldots,n-1$, et des m anciennes valeurs $u(k-j)$, $j = 0,1,\ldots,m-1$, du signal d'entrée, avec $m \leq n$.

3.2.1 - Caractérisation du premier modèle de représentation:

Le premier modèle d'un système discret est décrit par l'équation aux différences non-linéaire suivante:

$$y(k+1) = \sum_{i=0}^{n-1} \alpha_i . y(k-i) + g\left[u(k), u(k-1), \ldots, u(k-m+1)\right] \qquad (2)$$

où g est une fonction de $\Re^m$ vers $\Re$ et les coefficients α_i, $i = 0,1,\ldots,n-1$, sont des nombres réels. Le schéma bloc de ce type de systèmes est donné à la figure 3.2 où α^T est la transposée du vecteur-colonne dont les éléments sont les α_i.
Le signal de sortie à l'instant k+1 dépend donc linéairement de ses n anciennes valeurs et non-linéairement des m anciennes valeurs du signal d'entrée.

Dans l'expression (2), seule la fonction g est supposée inconnue, et on se propose de l'identifier à l'aide d'un réseau de neurones de type MLP. De plus, le système défini par (2) est supposé stable de sorte que, pour tout signal u appartenant à l'espace d'entrée, le signal de sortie correspondant est toujours borné. En d'autres termes, les racines de l'équation caractéristique:

$$z^n - \alpha_0 . z^{n-1} - \cdots - \alpha_{n-2} . z - \alpha_{n-1} = 0$$

sont toutes situées à l'intérieur du cercle unité dans le plan complexe.

On sait déjà que, sous certaines conditions, un réseau neuronal artificiel peut être construit pour approximer la fonction g sur un ensemble compact de $\Re^m$. Par suite, le modèle d'identification associé à (2) a une structure semblable à celle du processus, mais contient un réseau neuronal de paramètres ajustables.
On définit à la figure 3.2 le vecteur d'état à l'instant k du modèle; il est représenté par les variables $x_l(k)$, $l = 1,\ldots,n+m-1$.

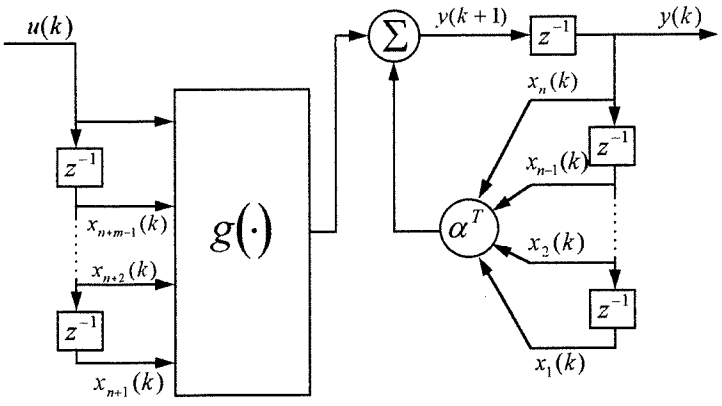

Figure 3.2: Représentation du premier modèle

3.2.2 - Caractérisation du deuxième modèle de représentation:

Le deuxième modèle, représenté à la figure 3.3, est caractérisé par l'équation aux différences non-linéaire suivante:

$$y(k+1) = f\left[y(k), y(k-1), \ldots, y(k-n+1)\right] + \sum_{j=0}^{m-1} \beta_j . u(k-j) \qquad (3)$$

où f est une fonction de $\Re^n$ dans $\Re$ et les coefficients β_j, $j = 0,1,\ldots,m-1$, sont des nombres réels. Dans la figure 3.3, β^T représente le vecteur-ligne $\left[\beta_0, \beta_1, \ldots, \beta_{m-1}\right]$.
Le signal de sortie y à l'instant k+1 dépend linéairement des m anciennes valeurs du signal d'entrée u et non-linéairement de ses n anciennes valeurs. D'une manière similaire au cas précédent, on suppose que dans l'expression (3) seule la fonction f est inconnue; un réseau neuronal de type MLP est utilisé pour l'approximer.
Ce modèle est particulièrement convenable pour les problèmes de réglage.

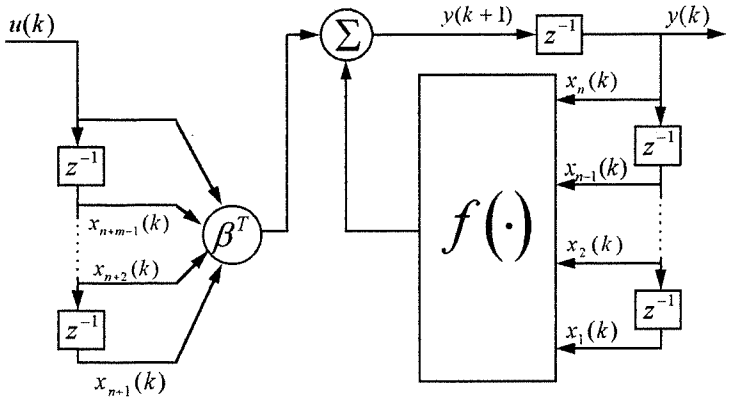

Figure 3.3: Représentation du deuxième modèle

3.2.3 - Caractérisation du troisième modèle de représentation:

L'équation aux différences non-linéaire, caractérisant le troisième modèle d'un système discret, est donnée par:

$$y(k+1) = f\left[y(k), y(k-1), \ldots, y(k-n+1)\right] + g\left[u(k), u(k-1), \ldots, u(k-m+1)\right] \qquad (4)$$

où f est une fonction de $\Re^n$ vers $\Re$ et g une fonction de $\Re^m$ dans $\Re$.

Le signal de sortie y à l'instant k+1 dépend non-linéairement de deux groupes séparés de variables, l'un contenant les n anciennes valeurs $y(k-i)$, $i = 0,1,\ldots,n-1$, de y et l'autre les m anciennes valeurs $u(k-j)$, $j = 0,1,\ldots,m-1$, du signal d'entrée u. Les fonctions f et g sont supposées toutes les deux inconnues et on se propose de les approximer à l'aide de deux réseaux de neurones de type MLP.
Le troisième modèle peut être considéré comme une généralisation des deux modèles précédents. Il est représenté à la figure 3.4.

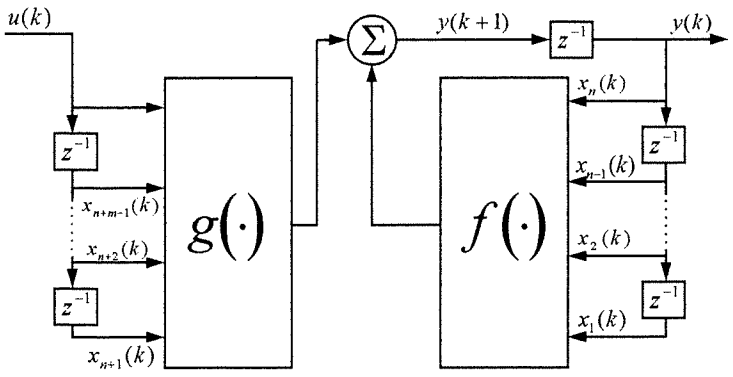

Figure 3.4: Représentation du troisième modèle

3.2.4 - Caractérisation du quatrième modèle de représentation:

Le quatrième modèle, représenté à la figure 3.5, est décrit par l'équation aux différences non-linéaire suivante:

$$y(k+1) = h\left[y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-m+1)\right] \qquad (5)$$

où h est une fonction de $\Re^{n+m}$ dans $\Re$, supposée inconnue. Il constitue une généralisation des trois modèles précédents, et peut représenter tout système dynamique non-linéaire monovariable décrit par (1) et vérifiant les conditions d'observabilité.

En raison de sa généralité, ce modèle est cependant le moins utilisé en pratique. Toutefois, son application peut être simplifiée en faisant appel aux modèles NARMA approximatifs, NARMA-L1 et NARMA-L2, déjà étudiés au paragraphe 1.3.2.c) auquel il est conseillé de se référer.

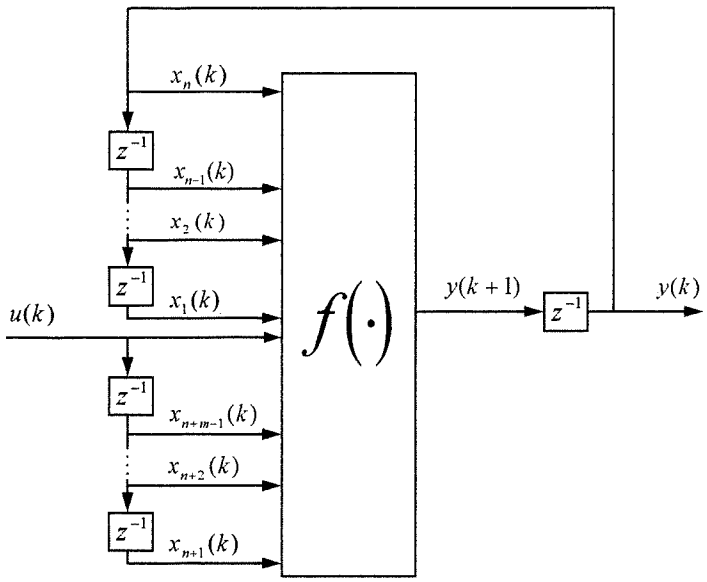

Figure 3.5: Représentation du quatrième modèle

3.3 - Modèles d'identification:

Comme on peut le constater dans le paragraphe précédent, le modèle d'identification, correspondant à l'un quelconque des quatres modèles de représentation des systèmes dynamiques non-linéaires, est constitué d'une association d'un ou plusieurs réseau(x) de neurones artificiels avec des blocs linéaires. Par conséquent, on peut le considérer comme un réseau neuronal généralisé. D'après le paragraphe 2.3 consacré à l'étude de ce type de réseaux, l'ajustement des paramètres libres du modèle d'identification se fait à l'aide de l'algorithme de rétro-propagation dynamique dont l'implantation s'avère très complexe dans la plupart des cas pratiques.
Dans les lignes qui suivent, on décrira deux modes d'identification: l'identification parallèle et l'identification série-parallèle. On verra que le mode série-parallèle est le plus avantageux, dans la mesure où il simplifie énormément l'implantation de l'algorithme de rétro-propagation dynamique.

3.3.1 - Mode parallèle:

Dans le mode d'identification parallèle, la structure du modèle d'identification est *identique* à celle du processus; les fonctions non-linéaires inconnues caractérisant le fonctionnement du processus sont toutefois remplacées par des réseaux neuronaux de type MLP. Le modèle d'identification parallèle est donc caractérisé dans le cas le plus général par l'équation suivante:

$$\hat{y}(k+1) = \hat{h}\left[\hat{y}(k), \hat{y}(k-1), \ldots, \hat{y}(k-n+1), u(k), u(k-1), \ldots, u(k-m+1)\right]$$

où $\hat{h}$ est la fonction réalisée par le réseau neuronal utilisé et $\hat{y}$ le signal de sortie du modèle.

On montre aux figures 3.6, 3.7, 3.8 et 3.9 les modèles d'identification parallèle correspondant aux quatre modèles de représentation décrits précédemment. Les blocs TDL_p (Tapped Delay Line) utilisés dans ces figures dénotent des lignes à retard dont le vecteur de sortie est formé par les p anciennes valeurs de leur signal d'entrée.
Dans le cas des figures 3.7 et 3.8, la présence d'un réseau de neurones dans une boucle rend compliqué et fastidieux le procédé d'implantation de l'algorithme de rétro-propagation dynamique destiné à ajuster les paramètres du modèle d'identification. De plus, rien ne peut garantir la convergence de ces paramètres vers leurs valeurs souhaitées. Malgré vingt ans de recherche, les conditions assurant une convergence des paramètres du modèle d'identification parallèle, même dans le cas des systèmes linéaires, restent indéterminées. Par conséquent, pour pouvoir aboutir à de meilleurs résultats, le mode série-parallèle décrit ci-après sera utilisé.

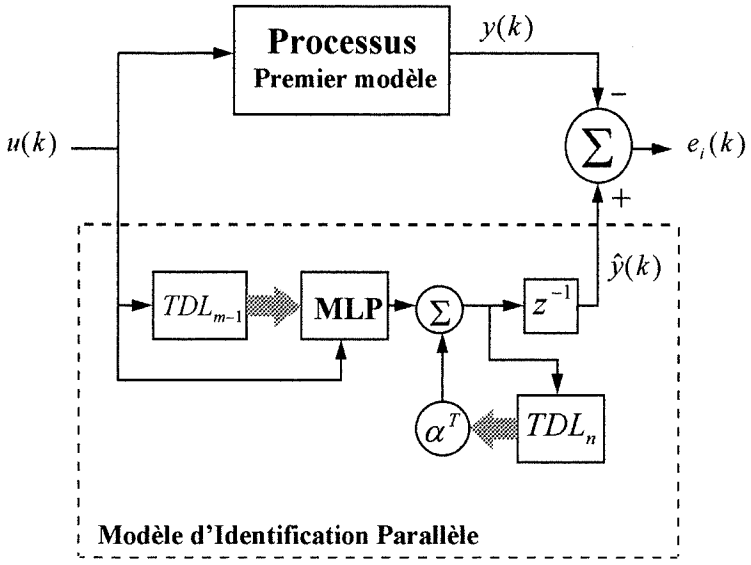

Figure 3.6: Identification parallèle du premier modèle de représentation

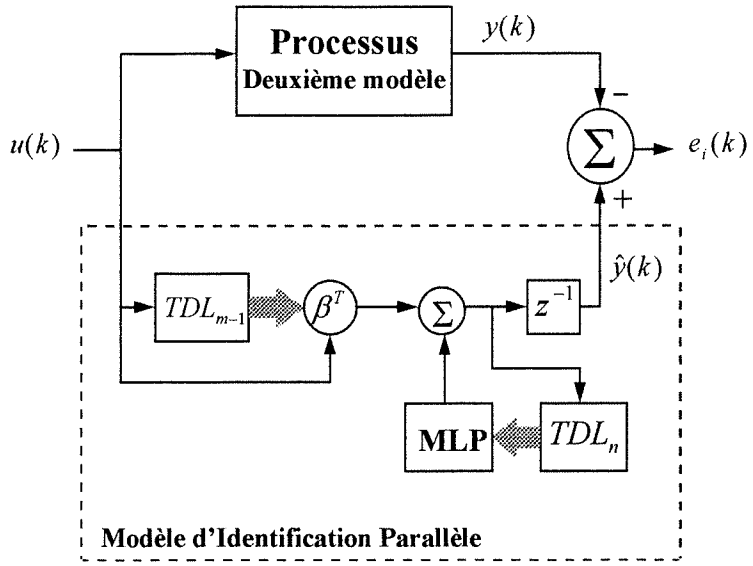

Figure 3.7: Identification parallèle du deuxième modèle de représentation

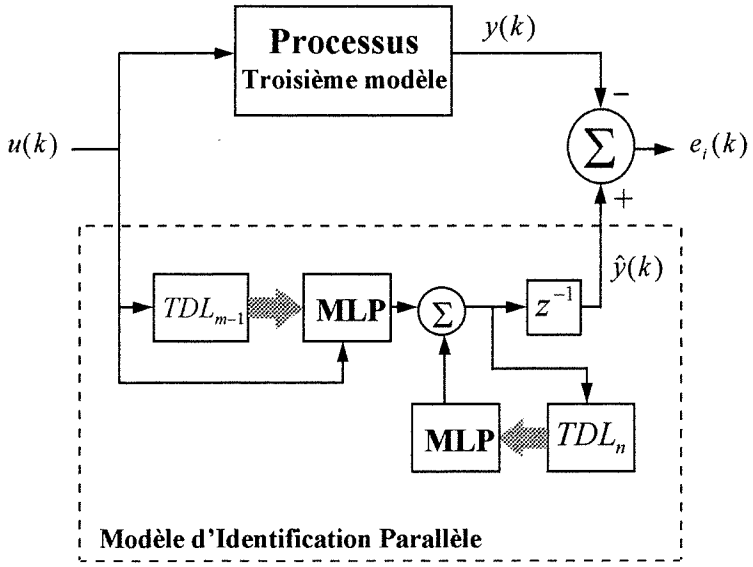

Figure 3.8: Identification parallèle du troisième modèle de représentation

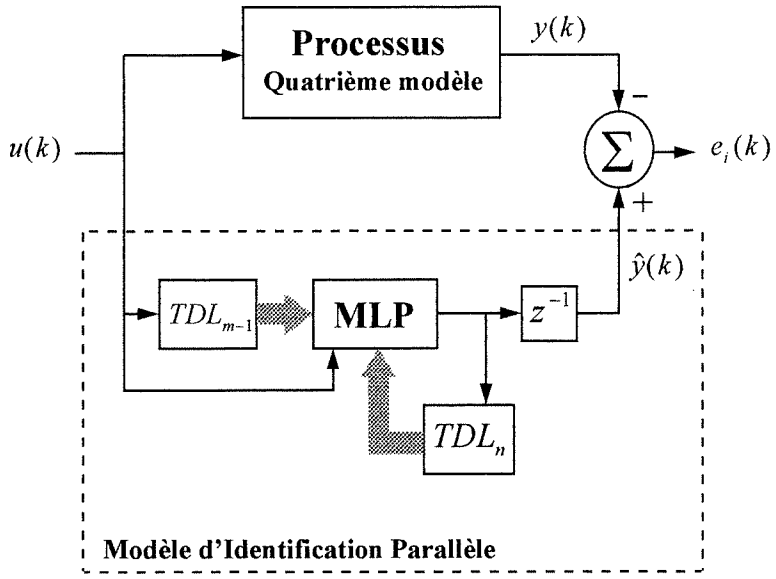

Figure 3.9: Identification parallèle du quatrième modèle de représentation

3.3.2 - Mode série-parallèle:

Contrairement au mode parallèle décrit précédemment, dans le mode série-parallèle ce sont les anciennes valeurs de la sortie y du processus réel (et non pas celles données par le modèle d'identification) qui sont utilisées dans la génération du signal de sortie $\hat{y}$ du modèle. Dans sa forme la plus générale, l'équation caractérisant le modèle d'identification série-parallèle s'écrit comme suit:

$$\hat{y}(k+1) = \hat{h}\left[y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-n+1)\right]$$

Le mode série-parallèle présente plusieurs avantages par rapport au mode parallèle. En particulier, étant donné qu'aucune boucle n'existe dans ce modèle, l'implantation de l'algorithme de rétro-propagation dynamique s'avère beaucoup plus simple que dans le cas du mode parallèle. D'autre part, des études ont montré que le mode série-parallèle est globalement stable, alors que la stabilité du mode parallèle n'est pas évidente en général. On représente aux figures 3.10, 3.11, 3.12 et 3.13 les modèles d'identification série-parallèle correspondant aux quatre modèles de représentation déjà vus.

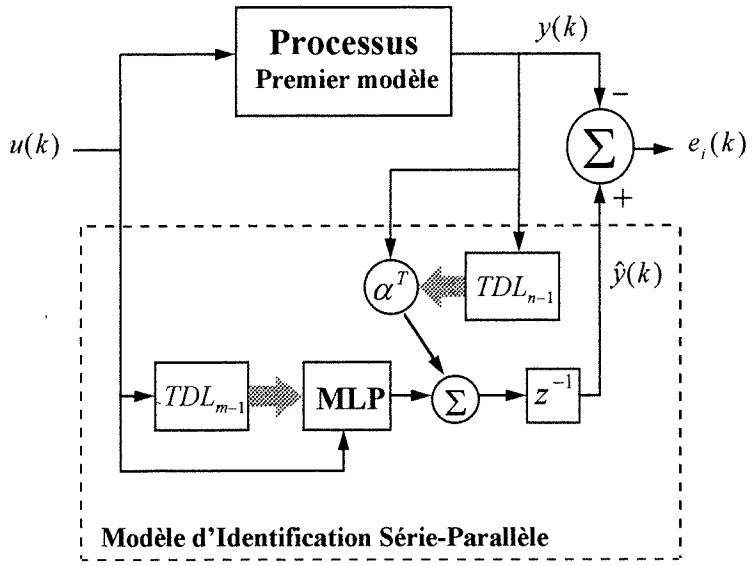

Figure 3.10: Identification série-parallèle du premier modèle de représentation

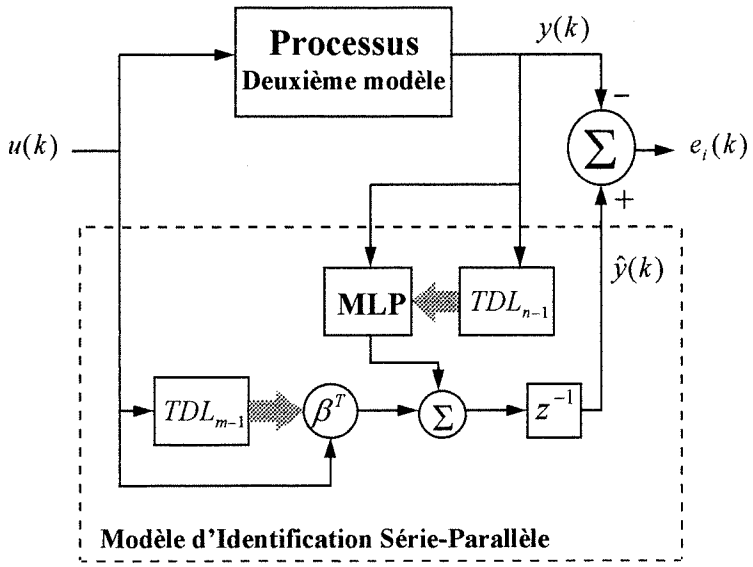

Figure 3.11: Identification série-parallèle du deuxième modèle de représentation

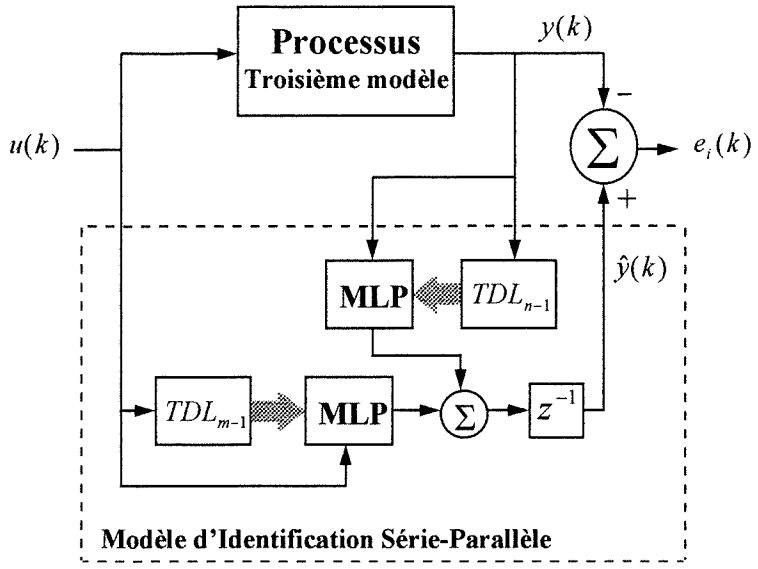

Figure 3.12: Identification série-parallèle du troisième modèle de représentation

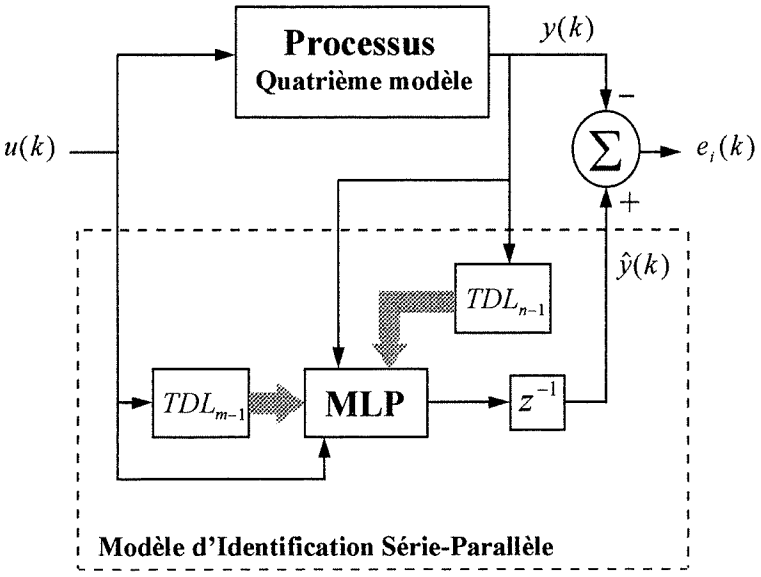

Figure 3.13: Identification série-parallèle du quatrième modèle de représentation

3.4 - Identification utilisant le vecteur d'état:

Tous les modèles d'identification construits jusqu'à présent se basent sur la représentation entrée-sortie des systèmes dynamiques monovariables. Aucun intérêt n'a été porté sur le vecteur d'état de ces systèmes.
Dans le présent paragraphe, on se propose d'établir une nouvelle méthode d'identification faisant utilité du vecteur d'état dans le cas où celui-ci est disponible. Cette procédure d'élaboration passe par deux phases principales: la première consiste à créer un bloc fonctionnel jouant le rôle d'*estimateur d'état*, et la deuxième correspond à la construction du modèle d'identification du système en se basant sur le vecteur d'état estimé.

3.4.1 - Estimateur d'état:

D'après les résultats mathématiques établis aux paragraphes 1.2.3 et 1.3.2 auxquels il est conseillé de se reporter, on sait que, si le système Σ décrit par (1) est fortement observable dans sa région d'opération, alors il existe une fonction Θ telle que:

$$\mathbf{x}(k) = \Theta\left[Y_n(k), U_{n-1}(k)\right] \tag{6}$$

où n représente l'ordre du système, $\mathbf{x}(k)$ le vecteur d'état, et $U_{n-1}(k)$ et $Y_n(k)$ des séquences d'entrée et de sortie de longueur n-1 et n respectivement.
En combinant l'équation (6) avec la relation connue suivante:

$$\mathbf{x}(k+n-1) = \Psi_{n-1}\left[\mathbf{x}(k), U_{n-1}(k)\right]$$

et en opérant un décalage de temps convenable, on aboutit à:

$$\mathbf{x}(k) = \Lambda\left[Y_n(k-n+1), U_{n-1}(k-n+1)\right] \tag{7}$$

Pour pouvoir estimer à tout instant k le vecteur d'état $\mathbf{x}$ du système, un réseau neuronal de type MLP est utilisé pour approximer la fonction Λ. Ce réseau comporte $2n$-1 noeuds d'entrée (auxquels on applique, à chaque instant k, les anciennes valeurs $u(k-j)$, $j = 1,\ldots,n-1$, du signal d'entrée et les anciennes valeurs $y(k-i)$, $i = 0,1,\ldots,n-1$, du signal de sortie) et n noeuds de sortie (débitant la valeur estimée $\hat{\mathbf{x}}$ à l'instant k du vecteur d'état, la dimension de celui-ci étant égale à l'ordre du système, c'est-à-dire n).

L'estimateur d'état ainsi constitué est entraîné à l'aide de l'algorithme de rétro-propagation basé sur le vecteur d'erreur $\mathbf{e}_x(k)$ défini par:

$$\mathbf{e}_x(k) = \hat{\mathbf{x}}(k) - \mathbf{x}(k) = \hat{\Lambda}\left[Y_n(k-n+1), U_{n-1}(k-n+1)\right] - \mathbf{x}(k)$$

La procédure d'estimation du vecteur d'état est décrite à la figure 3.14. On verra, dans la quatrième et dernière partie du rapport, comment l'estimateur d'état peut être utilisé dans la synthèse des régulateurs.

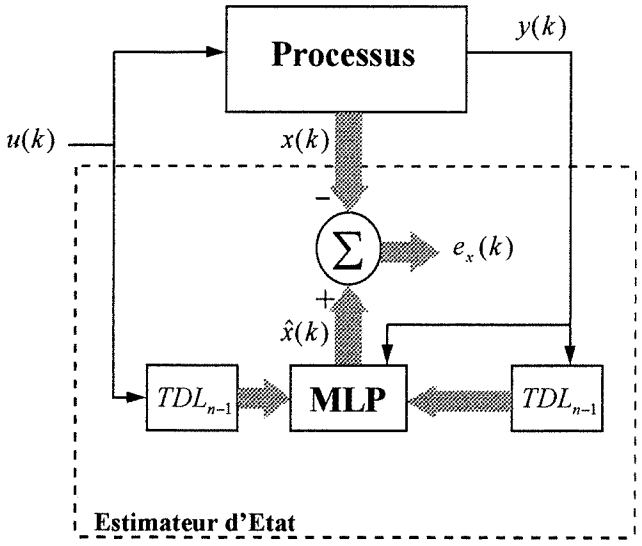

Figure 3.14: Estimateur d'état

3.4.2 - Modèle d'identification basé sur l'estimateur d'état:

Dans le modèle d'identification du système Σ décrit par (1), deux réseaux neuronaux MLP_1 et MLP_2 sont utilisés pour approximer respectivement les fonctions Φ et Ψ supposées inconnues. Ce modèle, représenté à la figure 3.15, est choisi du type série-parallèle et se caractérise par les équations suivantes:

$$\hat{\mathbf{x}}(k+1) = \hat{\Phi}\left[\hat{\mathbf{x}}(k), u(k)\right]$$
$$\hat{y}(k) = \hat{\Psi}\left[\hat{\mathbf{x}}(k)\right]$$

Les paramètres des réseaux MLP_1 et MLP_2 sont ajustés d'une manière indépendante. Cet ajustement, réalisé à l'aide de l'algorithme de rétro-propagation, dépend du vecteur d'erreur $\mathbf{e}_1(k) = \hat{\hat{\mathbf{x}}}(k) - \hat{\mathbf{x}}(k)$ dans le cas du réseau MLP_1, et de $e_2(k) = \hat{y}(k) - y(k)$ dans le cas du réseau MLP_2.

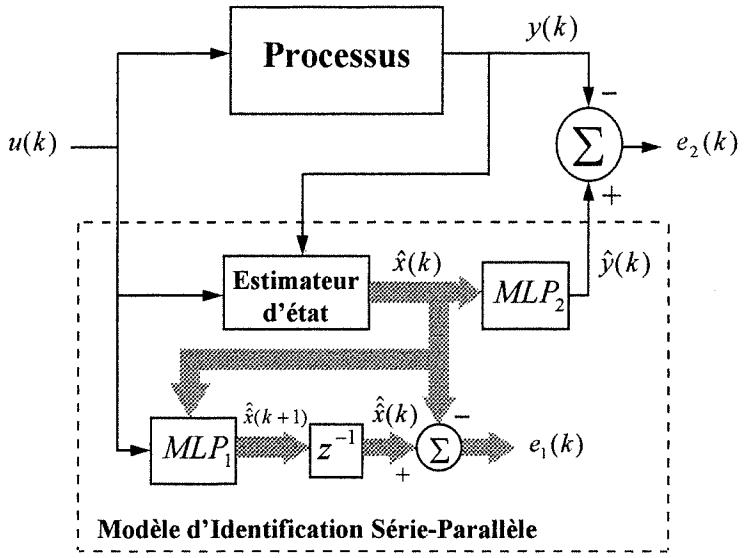

Figure 3.15: Modèle d'identification

Quatrième Partie

Réglage des systèmes dynamiques par réseaux neuronaux

La théorie du réglage concerne, en général, l'analyse et la synthèse de systèmes dynamiques dans lesquels une ou plusieurs variables sont gardées dans des limites prédéfinies. Pour un processus Σ décrit par:

$$\Sigma : \begin{cases} \mathbf{x}(k+1) = f[\mathbf{x}(k), \mathbf{u}(k)] \\ \mathbf{y}(k) = h[\mathbf{x}(k)] \end{cases} \qquad (1)$$

si les fonctions f et h sont connues, le problème de réglage consiste à construire un régulateur qui génère le vecteur de commande $\mathbf{u}(k)$ désiré, en se basant sur toute information disponible à l'instant k.

Plusieurs techniques élaborées dans les domaines temporel et fréquentiel existent pour la synthèse de régulateurs dans le cas des systèmes linéaires. Des méthodes similaires n'existent malheureusement pas pour les systèmes non-linéaires, même lorsque les fonctions f et h sont spécifiées.
Durant les trois dernières décennies, un grand intérêt a été porté sur le réglage des processus incertains. Un effort considérable fut mis surtout dans l'étude du réglage adaptatif des processus linéaires stationnaires (invariants dans le temps) dont les paramètres sont inconnus. Dans cette quatrième partie du rapport, on s'intéressera principalement au problème de réglage des systèmes dynamiques non-linéaires totalement ou partiellement inconnus. La synthèse de régulateurs pour ce type de systèmes sera basée sur des réseaux neuronaux artificiels.

Dans les applications pratiques, on préfère toujours choisir le régulateur le plus simple qui satisfait toutes les contraintes imposées par le concepteur. Ceci est généralement dû à l'existence d'une forte liaison entre la simplicité, la robustesse et le faible coût. Les régulateurs à base de réseaux neuronaux sont caractérisés par un grand nombre de paramètres, rendant leur utilisation très complexe. Par conséquent, des régulateurs plus simples, tels que ceux à actions proportionnelle, intégrale (PI) et dérivée (PID), ceux à retour d'état et les régulateurs adaptatifs linéaires, doivent toujours être essayés et trouvés non adéquats avant d'envisager l'utilisation des réseaux neuronaux.

Il existe trois différentes classes de problèmes de réglage où l'emploi des réseaux neuronaux s'avère désirable:

- La première classe concerne les processus dont les équations régissant le fonctionnement sont complètement spécifiées. Dans ce cas, la construction d'un régulateur est théoriquement tout à fait possible. Cependant, dans la mesure où la détermination d'un tel régulateur pourrait s'avérer dans certains cas pratiques trop complexe, il serait préférable d'utiliser dans ces conditions des réseaux neuronaux opérant d'une manière adaptative.

- Dans certaines situations, le processus non-linéaire à régler est stable, mais présente un comportement dynamique indésirable. L'application d'un réglage linéaire à ce type de processus ne serait pas satisfaisante pour toute condition initiale appartenant au domaine d'intérêt. Dans ce cas, l'identification par réseaux neuronaux du processus sur une longue période de temps, suivie de l'utilisation d'un régulateur non-linéaire, pourrait mener à de meilleurs résultats.

- Dans certains cas, le processus doit être capable d'opérer en plusieurs points d'équilibre. Le rôle du régulateur adaptatif dans ce type de problème est de stabiliser le système autour de ces états d'équilibre, suivant les informations disponibles à tout instant. Très peu de travaux théoriques ont été publiés dans ce domaine; mais certains résultats de simulation ont déjà prouvé l'efficacité des régulateurs à base de réseaux neuronaux.

Dans les années 80, un grand nombre de schémas approximatifs, basés sur des concepts simples, furent proposés dans la littérature relative au neuro-réglage pour que la sortie **y** du processus puisse suivre un signal désiré $\mathbf{y}_d$ avec une faible erreur. L'idée de base consiste à déterminer l'inverse de la fonction statique F caractérisant le processus, puis à l'utiliser comme régulateur afin que l'ensemble régulateur-processus puisse approximer la fonction constante unitaire sur le domaine de variation de la consigne. Les différentes méthodes de calcul de l'inverse ont fait naître différentes architectures de réglage, dont quelques-unes sont données à la figure 4.1.
Lors d'un *réglage inverse direct* (figure 4.1.a), un réseau neuronal *MLP* utilisé comme régulateur est entraîné pour approximer l'inverse de la fonction F du processus; l'ajustement des paramètres du réseau est basé sur l'écart ε entre la sortie **y** du système et la consigne **r**; ce type d'architecture suppose implicitement que la fonction F est connue mais complexe. Dans le cas du *réglage inverse indirect* (figure 4.1.b), un réseau neuronal MLP_2 est entraîné de manière à approximer F; un régulateur MLP_1 est ensuite ajusté afin que l'association MLP_2 - MLP_1 puisse approximer l'opérateur unité. L'avantage de cette configuration par rapport à la précédente devient évident lorsqu'on considère la fonction F inconnue; dans ce cas, le calcul du gradient d'erreur:

$$\nabla_W\left(\frac{1}{2}\mathbf{e}^T\mathbf{e}\right) = (\mathbf{y}-\mathbf{r})^T.\nabla_u F(\mathbf{u}).\nabla_W \mathbf{u}$$

correspondant au réseau *MLP* de la figure 4.1.a s'avère impossible, vu que le terme $\nabla_u F(\mathbf{u})$ est indéterminé. Pour remédier à ce problème, on remplace d'abord la sortie réelle $\mathbf{y}$ du processus par la réponse estimée $\hat{\mathbf{y}}$ de l'identificateur (qui n'est autre que MLP_2 dans le cas de la figure 4.1.b), puis on fait propager l'erreur $\mathbf{e}_1 = \mathbf{y}-\mathbf{r}$ d'une manière réciproque à travers l'identificateur avant de l'appliquer au réseau MLP_1. On peut écrire ainsi:

$$\nabla_{W_1}\left(\frac{1}{2}\mathbf{e}_1^T\mathbf{e}_1\right) = (\hat{\mathbf{y}}-\mathbf{r})^T.\nabla_u\hat{\mathbf{y}}.\nabla_{W_1}\mathbf{u}$$

Le terme $\nabla_u\hat{\mathbf{y}}$ peut facilement être déterminé, connaissant les paramètres $W_2 = \left[w_{2,ji}\right]$ du réseau MLP_2. Le calcul du gradient $\nabla_{W_1}\mathbf{u}$ est implanté dans le réseau de sensibilité de MLP_1. Il est important de noter que l'apprentissage du réseau MLP_2 doit être beaucoup plus rapide que celui de MLP_1, afin que l'on puisse considérer $\hat{\mathbf{y}}$ pratiquement égal à $\mathbf{y}$ tout au long de la phase d'apprentissage de MLP_1, ce qui permet d'accélérer sa convergence et d'améliorer, par conséquent, ses performances.

Toutes les approches décrites précédemment constituent des approximations dans la mesure où le caractère dynamique du processus n'a toujours pas été pris en considération (seule la fonction statique F du processus a été utilisée).
A cause de la complexité du problème de réglage non-linéaire, on fait souvent appel dans les applications au raisonnement heuristique. Des schémas simples, tels ceux représentés à la figure 4.1, ont permis d'obtenir par simulation des résultats impressionnants. Toutefois, leur application dans des conditions d'opération différentes n'est pas garantie.

Dans les lignes qui suivent, on commencera par décrire le principe général du problème de réglage. Un *modèle de référence*, dont le rôle est d'imposer au système réglé des comportements statique et dynamique désirés, sera utilisé. Deux approches distinctes pour le réglage adaptatif seront ensuite envisagées, à savoir les réglages *direct* et *indirect*. Puis, on tentera d'établir des méthodes de réglage pour trois types de problèmes: la *stabilisation*, la *régulation* et la *poursuite*. Pour chacun de ces problèmes, les deux modes de représentation du processus, à savoir le modèle d'état et la représentation entrée-sortie (ou modèle NARMA), seront considérés séparément. Enfin, on décrira brièvement un nouveau mode de réglage adaptatif, dit *à modèles multiples*, qui utilise les techniques de *commutation* et d'*adaptation*. Sauf indication du contraire, on considèrera toujours des processus monovariables afin d'alléger les notations. Dans ces conditions, les signaux d'entrée et de sortie du processus, ainsi que la consigne de référence, sont tous scalaires et seront notés respectivement par la suite u, y et r.

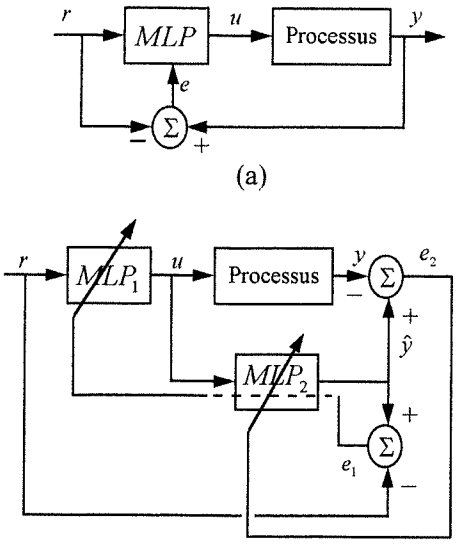

Figure 4.1: Architectures de réglage
a) Réglage inverse direct
b) Réglage inverse indirect

4.1 - Modèle de référence:

Le but du réglage est de permettre à un processus donné d'avoir une bonne réponse dynamique et un comportement statique convenable. Ce type de fonctionnement souhaité est imposé par un modèle de référence comme le montre la figure 4.2.
En effet, en notant respectivement $u(k)$ et $y(k)$ les signaux d'entrée et de sortie du processus à régler et $r(k)$ et $y_m(k)$ ceux du modèle de référence, l'objectif est de déterminer l'entrée de commande $u(k)$, pour tout $k \geq k_0$, de façon à obtenir:

$$\lim_{k\to\infty}|e_c(k)| = \lim_{k\to\infty}|y(k) - y_m(k)| \leq \varepsilon$$

avec ε une constante positive spécifiée.

Le modèle de référence est supposé toujours linéaire et stable, et le signal d'entrée correspondant $r(k)$ est une fonction bornée dans $\Re$. Remarquons aussi que le signal de sortie $y_m(k)$ du modèle n'est autre que la réponse désirée du système réglé.

Les systèmes adaptatifs utilisant des modèles de référence ont fait l'objet de plusieurs études approfondies durant les vingt dernières années. Dans la littérature, tels systèmes sont souvent dits de type MRAC (Model Reference Adaptive Control systems). La

formulation d'un problème du type MRAC suppose implicitement que le concepteur connaît suffisamment le processus concerné et peut ainsi spécifier, en termes du signal de sortie du modèle de référence, le comportement désiré du système réglé.

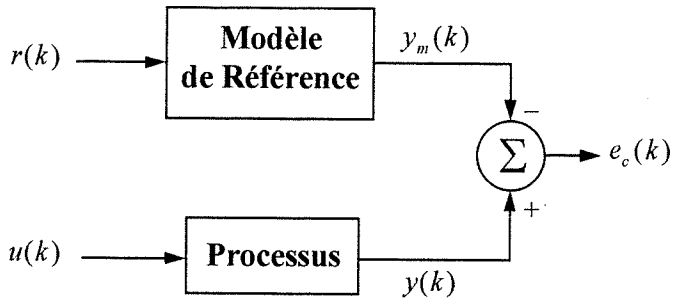

Figure 4.2: Réglage adaptatif avec un modèle de référence

Comme c'est déjà mentionné, le modèle de référence est supposé toujours linéaire, ceci étant dû au fait que la théorie des systèmes linéaires est bien développée et des méthodes de synthèse de modèles linéaires ayant des propriétés désirées sont bien établies. Des méthodes générales permettant de déterminer des modèles non-linéaires, ayant des propriétés désirées en régimes statique et dynamique, ne sont pas encore disponibles.

4.2 - Modes de réglage:

Dans ce qui suit, le processus à régler est supposé être un système stationnaire dont les paramètres internes sont inconnus. Ces paramètres sont considérés comme les éléments d'un vecteur **p**. Si **p** est connu, les paramètres du régulateur peuvent être choisis de manière à ce que l'ensemble régulateur-processus se comporte comme un modèle de référence décrit par une équation aux différences linéaire à coefficients constants. Par contre, si **p** est inconnu, les paramètres du régulateur doivent être ajustés d'une manière adaptative utilisant toute information disponible concernant le système. Lorsque le processus est non-linéaire et dynamique (ce qui est souvent le cas), le régulateur contient des réseaux neuronaux artificiels.
Depuis plus d'une vingtaine d'années, deux approches distinctes pour le réglage adaptatif des processus inconnus ont été utilisées; ce sont les réglages *direct* et *indirect*.

4.2.1 - Réglage direct:

Dans un réglage direct (figure 4.3), les paramètres du régulateur sont ajustés directement de manière à réduire une certaine norme de l'erreur de sortie. A présent, des méthodes permettant un tel ajustement direct et stable des paramètres n'existent pas.

Ceci est dû principalement à la nature non-linéaire du processus et du régulateur associé. Dans le cas où le régulateur est un réseau neuronal de type MLP (perceptron multi-couche), l'algorithme de rétro-propagation visé à ajuster les paramètres libres (poids synaptiques et seuils) du réseau ne peut être appliqué directement, étant donné que le processus est inconnu et ne peut donc être utilisé dans la génération des dérivées partielles désirées. Ainsi, en attendant que des méthodes de réglage direct soient développées, le réglage adaptatif des systèmes dynamiques non-linéaires se fait suivant des méthodes indirectes.

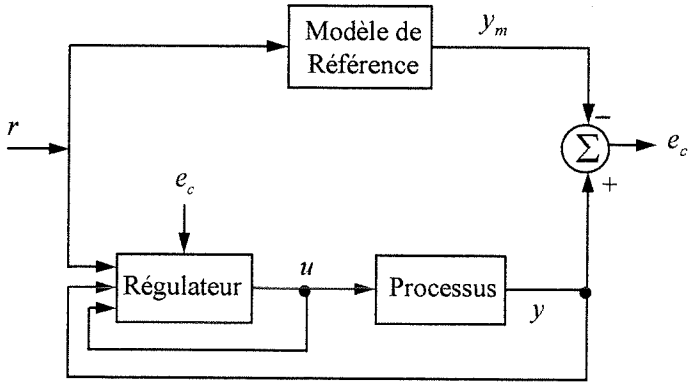

Figure 4.3: Réglage adaptatif direct

4.2.2 - Réglage indirect:

Lors d'un réglage indirect, les paramètres supposés inconnus du processus à régler sont estimés à chaque instant k à l'aide d'un modèle d'identification, comme le montre la figure 4.4. Ces paramètres estimés sont considérés comme les éléments d'un vecteur $\hat{\mathbf{p}}(k)$. Quant aux paramètres du régulateur, ils sont par la suite choisis en considérant que $\hat{\mathbf{p}}(k)$ représente la vraie valeur du vecteur $\mathbf{p}$ des paramètres du processus.

Si le système ne se trouve pas soumis à des perturbations ou bruits externes, il s'avère raisonnable d'ajuster les paramètres du régulateur et du modèle d'identification d'une manière synchrone. En revanche, lorsque ces perturbations ou bruits externes sont présents au niveau du système, le procédé d'identification s'active à chaque instant, tandis que l'ajustement des paramètres du régulateur se fait plus lentement (à chaque intervalle de temps); ceci permet d'augmenter la robustesse du système adaptatif.

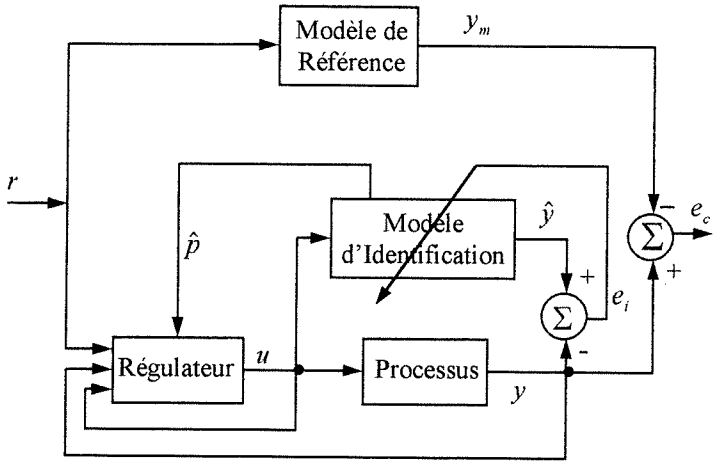

Figure 4.4: Réglage adaptatif indirect

4.3 - Types de réglage:

On étudiera dans ce paragraphe trois types de réglage, à savoir la *stabilisation*, la *régulation* et la *poursuite*. La stabilisation a pour but de garder le point de fonctionnement du système réglé autour d'un état d'équilibre (pouvant être stable ou instable) en l'absence de toute consigne externe. La régulation, quant à elle, permet de stabiliser ce point de fonctionnement autour d'un état fixe imposé par la consigne constante appliquée à l'entrée du système. Dans le problème de poursuite, on impose au signal de sortie du système une certaine forme temporelle désirée qui est donnée en général par un modèle de référence judicieusement choisi.
Pour chacun des types de réglage mentionnés ci-dessus, on considèrera séparément deux modes de représentation du processus: le modèle d'état et la représentation entrée-sortie (ou modèle NARMA). Le processus sera toujours considéré monovariable et décrit par les équations (1), où f est supposée être une fonction continûment différentiable, lipschitzienne et bornée pour tout $(\mathbf{x}, u)$ appartenant au domaine d'intérêt.

4.3.1 - Problème de stabilisation:

4.3.1.1 - Cas du modèle d'état:

Dans cette section, trois méthodes de stabilisation du processus non-linéaire Σ autour d'un état d'équilibre seront décrites. Elles sont toutes basées sur l'utilisation du vecteur d'état $\mathbf{x}(k)$ du système. Dans ces conditions (le vecteur d'état étant supposé accessible à tout instant), si les fonctions f et h caractérisant le système Σ sont inconnues, deux réseaux neuronaux MLP_f et MLP_h seront utilisés respectivement pour les approximer, conformément à la méthode décrite au paragraphe 3.4.2.

4.3.1.1.1 - Stabilisation par un régulateur linéaire:

La méthode de stabilisation la plus simple consiste à utiliser un régulateur linéaire. Elle fait appel à la linéarisation Σ_L de Σ autour du point d'équilibre $(\mathbf{x}_0, u_0) = (0,0)$, définie par:

$$\Sigma_L : \begin{array}{l} \delta\mathbf{x}(k+1) = A.\delta\mathbf{x}(k) + B.\delta u(k) \\ \delta y(k) = C.\delta\mathbf{x}(k) \end{array} \qquad (2)$$

où:

$$A = \frac{\partial f(\mathbf{x},u)}{\partial \mathbf{x}}\Big|_{(0,0)} \ , \ B = \frac{\partial f(\mathbf{x},u)}{\partial u}\Big|_{(0,0)} \text{ et } C = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\Big|_{(0)}$$

A est une matrice carrée d'ordre n, et B et C sont deux vecteurs de dimensions $(n,1)$ et $(1,n)$ respectivement.

D'après la théorie des systèmes linéaires, si Σ_L est commandable, alors il existe une loi de commande linéaire de la forme:

$$u = K.\mathbf{x} \qquad (3)$$

qui stabilise Σ_L autour de l'origine. Sachant que Σ_L constitue une approximation du premier ordre du système principal Σ, cette loi de commande peut rendre l'origine un point localement asymptotiquement stable de Σ. Ce résultat est équivalent à celui évoqué au paragraphe 1.2.2 auquel on conseille de se reporter.

La synthèse d'un régulateur linéaire n'exige pas l'utilisation de réseaux neuronaux. D'après la loi de commande (3), le système linéaire global devient caractérisé par l'équation d'état suivante:

$$\delta\mathbf{x}(k+1) = \left(A + B.K\right).\delta\mathbf{x}(k)$$

et sera asymptotiquement stable si les valeurs propres de $A + B.K$ sont à l'intérieur du cercle unité dans le plan complexe.

Le réglage linéaire permet donc une stabilisation des systèmes non-linéaires autour de leur point d'équilibre. Cependant, leur domaine de stabilité au voisinage de ce point peut être très limité. On pourrait espérer un élargissement du domaine de stabilité en utilisant des régulateurs non-linéaires appropriés. Tels régulateurs feront l'objet des deux paragraphes suivants.

4.3.1.1.2 - Stabilisation à travers une linéarisation par retour d'état:

a) Principe:

Soit Σ un système non-linéaire monovariable décrit par les équations (1). Le principe de *linéarisation par retour d'état* consiste à trouver deux transformations dont:

- la première effectue un changement de coordonnées dans l'espace d'état $\mathbf{z} = \Phi(\mathbf{x})$, où $\Phi: \Re^n \rightarrow \Re^n$ est inversible et continûment différentiable,

- la deuxième constitue la loi de commande $u(k) = \Psi[\mathbf{x}(k), v(k)]$

qui rendent Σ équivalent localement à un système linéaire. Si cette procédure est accomplie, on pourra alors se servir des outils qu'offre la théorie des systèmes linéaires pour régler et stabiliser le système Σ autour du point d'équilibre désiré.

En appliquant les deux transformations précédentes à la première équation du système (1), on obtient la nouvelle équation d'état:

$$\mathbf{z}(k+1) = \Phi\left[\mathbf{x}(k+1)\right] = \Phi\left[f\left(\Phi^{-1}\left[\mathbf{z}(k)\right], \Psi\left[\Phi^{-1}\left[\mathbf{z}(k)\right], v(k)\right]\right)\right] \qquad (4)$$

où $\mathbf{z}(k)$ est le nouveau vecteur d'état et $v(k)$ le nouveau signal d'entrée. Pour que le système Σ soit linéarisable par retour d'état, il faut que l'équation (4) soit linéaire. Si les transformations Φ et Ψ existent uniquement au voisinage du point (0,0), le système est alors dit *localement linéarisable par retour d'état* à l'origine.
Il existe des conditions nécessaires et suffisantes pour qu'un système soit linéarisable par retour d'état. Elles sont intimement liées à la notion de *distribution*.

b) Définition d'une distribution:

Soit V un sous-ensemble de $\Re^n$ sur lequel sont définies d fonctions dérivables $s_i: V \rightarrow \Re^n$, $i = 1,\ldots,d$. Pour tout point $\mathbf{x} \in V$, les vecteurs $s_1(\mathbf{x}), s_2(\mathbf{x}), \ldots, s_d(\mathbf{x})$ forment une base d'un sous-espace vectoriel $\Delta(\mathbf{x})$ de $\Re^n$. La fonction qui, à tout point $\mathbf{x} \in V$, associe un espace vectoriel $\Delta(\mathbf{x})$ est appelée une *distribution*. On note:

$$\Delta(\mathbf{x}) = \left(s_1(\mathbf{x}), s_2(\mathbf{x}), \ldots, s_d(\mathbf{x})\right), \quad \text{pour tout } \mathbf{x} \in V$$

En considérant le système (1), et en posant:

$$f_x(\mathbf{x},u) = \frac{\partial}{\partial \mathbf{x}} f(\mathbf{x},u) \quad \text{et} \quad f_u(\mathbf{x},u) = \frac{\partial}{\partial u} f(\mathbf{x},u)$$

on définit d'une manière itérative sur $\Re^n$ les distributions suivantes (dépendant du paramètre u):

$$\Delta_0(\mathbf{x},u) = 0$$
$$\Delta_1(\mathbf{x},u) = f_x^{-1}(\mathbf{x},u).\mathrm{Im}\big(f_u(\mathbf{x},u)\big)$$
$$\Delta_{i+1}(\mathbf{x},u) = f_x^{-1}(\mathbf{x},u).\Big[\Delta_i\big(f(\mathbf{x},u),u\big) + \mathrm{Im}\big(f_u(\mathbf{x},u)\big)\Big]$$

où $\mathrm{Im}\big(f_u(\mathbf{x},u)\big)$ dénote l'image de l'espace vectoriel $\Re^n$ par la fonction f_u et $f_x^{-1}.\Omega$ l'image inverse du sous-espace Ω par la fonction f_x. On peut montrer que:

$$\Delta_0(\mathbf{x},u) \subset \Delta_1(\mathbf{x},u) \subset \Delta_2(\mathbf{x},u) \cdots$$

et le rang de Δ_i atteint sa valeur maximale (qui n'est autre que n) après n itérations au plus.

c) Théorème de linéarisation locale par retour d'état:

Soit $(\mathbf{x}_0,u_0) = (0,0)$ un point d'équilibre du système (1), et $Df = (f_x, f_u)$ le jacobien de f par rapport à $\mathbf{x}$ et u. Supposons que $Df(0,0)$ est de rang n. Le système (1) est localement linéarisable par retour d'état au point (0,0) si, et seulement si:

i) les distributions $\Delta_i(\mathbf{x},u)$, $i = 0,1,2,\ldots$, ont une dimension constante et indépendante de u dans un voisinage de (0,0),

ii) Δ_n est de dimension n au voisinage de (0,0).

d) Cas particulier d'un système du second ordre:

Soit le système non-linéaire du second ordre décrit par:

$$\Sigma_2 : \begin{array}{l} x_1(k+1) = f_1\big[x_1(k),x_2(k)\big] \\ x_2(k+1) = f_2\big[x_1(k),x_2(k),u(k)\big] \end{array}$$

Pour ce système, on a:

$$f_x(\mathbf{x},u) = \begin{bmatrix} f_{11}(\mathbf{x}) & f_{12}(\mathbf{x}) \\ f_{21}(\mathbf{x},u) & f_{22}(\mathbf{x},u) \end{bmatrix} \quad \text{et} \quad f_u(\mathbf{x},u) = \begin{bmatrix} 0 \\ f_{2u}(\mathbf{x},u) \end{bmatrix}$$

où $f_{ij} = \dfrac{\partial f_i}{\partial x_j}$, pour tout $(i,j) \in \{1,2\}^2$, $f_{2u} = \dfrac{\partial f_2}{\partial u}$ et $\mathbf{x} = (x_1, x_2)$.

En supposant implicitement que l'origine est un point d'équilibre du système (c'est-à-dire $f_1(0,0) = f_2(0,0,0) = 0$) et que la matrice f_x est non-singulière dans un voisinage V de ce point, on obtient facilement:

$$f_x^{-1}(\mathbf{x},u) = \frac{1}{\det f_x(\mathbf{x},u)} \begin{bmatrix} f_{22}(\mathbf{x},u) & -f_{12}(\mathbf{x}) \\ -f_{21}(\mathbf{x},u) & f_{11}(\mathbf{x}) \end{bmatrix}$$

où $\det f_x(\mathbf{x},u) = f_{11}(\mathbf{x}).f_{22}(\mathbf{x},u) - f_{12}(\mathbf{x}).f_{21}(\mathbf{x},u)$ est le déterminant de f_x. Par conséquent, les distributions Δ_0 et Δ_1 sont données par:

$$\Delta_0(\mathbf{x},u) = 0$$

et

$$\Delta_1(\mathbf{x},u) = f_x^{-1}(\mathbf{x},u).\mathrm{Im}\left(f_u(\mathbf{x},u)\right) = \left(\begin{bmatrix} -f_{12}(\mathbf{x}).f_{2u}(\mathbf{x},u) \\ f_{11}(\mathbf{x}).f_{2u}(\mathbf{x},u) \end{bmatrix} \right)$$

Pour que Δ_1 soit de dimension 1, il suffit que le produit $f_{12}(\mathbf{x}).f_{2u}(\mathbf{x},u)$ soit non nul au voisinage de (0,0), ce qui revient à dire:

$$f_{12}(\mathbf{x}) \neq 0 \quad \text{et} \quad f_{2u}(\mathbf{x},u) \neq 0\ , \quad \forall \mathbf{x} \in V$$

Dans ces conditions, Δ_1 est une droite de vecteur directeur $\left[1, -f_{11}(\mathbf{x})/f_{12}(\mathbf{x})\right]^T$ dépendant uniquement de $\mathbf{x}$, et Δ_2 n'est autre que le plan $\Re^2$. Le système Σ_2 est ainsi localement linéarisable par retour d'état.

e) Implantation:

Si les équations (1) caractérisant le processus à régler sont connues et vérifient les conditions de linéarisation par retour d'état, on aura la tâche de chercher deux fonctions:

$$\Phi: \Re^n \to \Re^n \quad \text{et} \quad \Psi: \Re^{n+1} \to \Re$$

telles que:

$$\mathbf{z} = \Phi(\mathbf{x}) \quad , \quad u = \Psi(\mathbf{x}, v)$$

et

$$\mathbf{z}(k+1) = \Phi[\mathbf{x}(k+1)] = \Phi\left[f\left(\mathbf{x}(k), \Psi\left[\mathbf{x}(k), v(k)\right]\right)\right] = A_L.\mathbf{z}(k) + B_L.v(k)$$

où les matrices constantes A_L et B_L correspondent à un système linéaire stable et commandable. En particulier, on peut choisir A_L et B_L de la manière suivante:

$$A_L = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \quad \text{et} \quad B_L = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{5}$$

Si, par contre, on ne dispose que d'un modèle du processus réel, décrit par:

$$\hat{\Sigma} : \begin{aligned} \hat{\mathbf{x}}(k+1) &= \hat{f}[\hat{\mathbf{x}}(k), u(k)] \\ \hat{y}(k) &= \hat{h}[\hat{\mathbf{x}}(k)] \end{aligned} \tag{6}$$

où $\hat{f}$ et $\hat{h}$ sont des approximations des fonctions réelles f et h, représentées par deux réseaux neuronaux MLP_f et MLP_h, il est toujours possible d'étendre à ce modèle la propriété de linéarisation par retour d'état. En effet, en supposant que:

$$\left\| \hat{f}(\mathbf{x}, u) - f(\mathbf{x}, u) \right\| = \left\| \mathbf{e}(\mathbf{x}, u) \right\| < \varepsilon << 1$$

pour tout point $(\mathbf{x}, u)$ appartenant au domaine d'opération D, on obtient:

$$\Phi\left[\hat{f}\left(\mathbf{x}(k), \Psi\left[\mathbf{x}(k), v(k)\right]\right)\right] = \Phi\left[f\left(\mathbf{x}(k), \Psi\left[\mathbf{x}(k), v(k)\right]\right) + \mathbf{e}\left(\mathbf{x}(k), \Psi\left[\mathbf{x}(k), v(k)\right]\right)\right] \tag{7}$$

Vu la continuité de la fonction Φ, si $\left\|\mathbf{e}(\mathbf{x}, u)\right\| < \varepsilon$ où ε est un nombre réel positif voisin de zéro, la relation (7) peut être réécrite comme suit:

$$\begin{aligned} \Phi\left[\hat{f}\left(\mathbf{x}, \Psi\left[\mathbf{x}, v\right]\right)\right] &\cong \Phi\left[f\left(\mathbf{x}, \Psi\left[\mathbf{x}, v\right]\right)\right] + \mathbf{e}_{l1}\left[\mathbf{x}, u, \Psi[\cdot], \Phi[\cdot]\right] \\ &\cong A_L . \mathbf{z} + B_L . v + \mathbf{e}_{l1}\left[\mathbf{x}, u, \Psi[\cdot], \Phi[\cdot]\right] \end{aligned} \tag{8}$$

avec:

$$\mathbf{e}_{l1} = \frac{\partial \Phi}{\partial \mathbf{e}} \cdot \mathbf{e}$$

La borne supérieure de $\mathbf{e}_{l1}$ est fonction de ε et $\sup\left\|\partial\Phi/\partial\mathbf{x}\right\|$. Par suite, si le modèle $\hat{\Sigma}$ est suffisamment précis, il peut être transformé en un système presque linéaire décrit par (8).

A ce stade, l'objectif est d'entraîner simultanément deux réseaux neuronaux MLP_Φ et MLP_Ψ de telle sorte que, lorsqu'on applique au système (6) le signal $\hat{u}(k) = \hat{\Psi}\left[\hat{\mathbf{x}}(k), v(k)\right]$, le vecteur $\hat{\mathbf{z}}(k) = \hat{\Phi}\left[\hat{\mathbf{x}}(k)\right]$ puisse approximer le vecteur $\mathbf{z}(k)$ donné par le modèle linéaire:

$$\mathbf{z}(k+1) = A_L.\mathbf{z}(k) + B_L.v(k)$$

où A_L et B_L sont donnés en (5). Ce procédé est illustré à la figure 4.5.

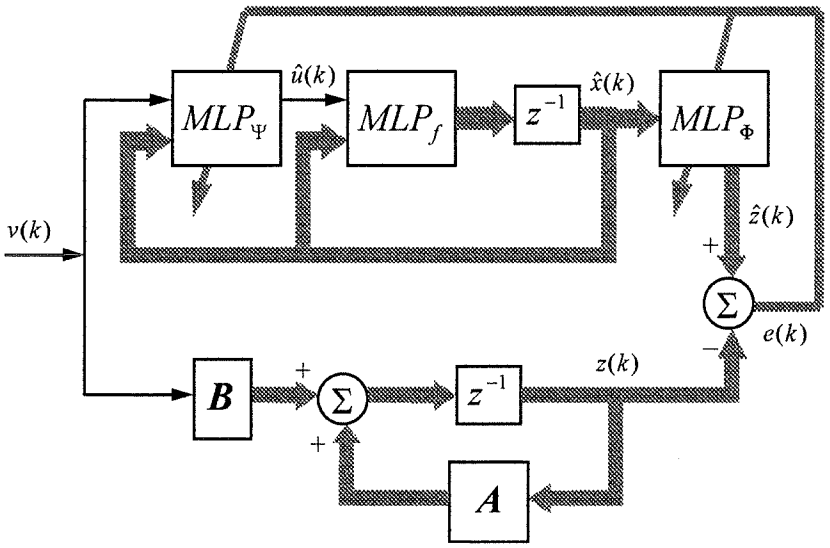

Figure 4.5: Linéarisation par retour d'état

En supposant $\Phi(0) = 0$, et en considérant l'origine comme étant l'état initial dans l'architecture de la figure 4.5, l'erreur instantanée est donnée par:

$$\mathbf{e}(k) = \hat{\mathbf{z}}(k) - \mathbf{z}(k)$$

et la performance du système, évaluée sur un intervalle limité de temps, peut être représentée par la fonction d'erreur J définie par:

$$J = \frac{1}{2}\sum_k \left\|\hat{\mathbf{z}}(k) - \mathbf{z}(k)\right\|^2 = \frac{1}{2}\sum_k \left\|\mathbf{e}(k)\right\|^2 = \frac{1}{2}\sum_k \left[\mathbf{e}(k)^T\right].\mathbf{e}(k)$$

Les paramètres libres du réseau MLP_Φ sont ajustés à l'aide de l'algorithme de rétro-propagation statique, étant donné qu'il est directement connecté à la sortie du système. Par contre, pour calculer les gradients de la fonction d'erreur J par rapport aux paramètres de MLP_Ψ, l'utilisation de l'algorithme de rétro-propagation dynamique est

obligatoire. En notant θ_l un paramètre quelconque de MLP_Ψ, le gradient de J par rapport à θ_l est donné par:

$$\frac{dJ}{d\theta_l} = \sum_k \left[\hat{\mathbf{z}}(k) - \mathbf{z}(k)\right]^T \cdot \frac{d\hat{\mathbf{z}}(k)}{d\theta_l} \tag{9}$$

avec:

$$\frac{d\hat{\mathbf{z}}(k)}{d\theta_l} = \frac{d\hat{\Phi}[\hat{\mathbf{x}}(k)]}{d\theta_l} = \frac{\partial\hat{\Phi}[\hat{\mathbf{x}}(k)]}{\partial\hat{\mathbf{x}}(k)} \cdot \frac{d\hat{\mathbf{x}}(k)}{d\theta_l} \tag{10}$$

$$\begin{aligned}\frac{d\hat{\mathbf{x}}(k)}{d\theta_l} &= \frac{d\hat{f}\left[\hat{\mathbf{x}}(k-1), \hat{u}(k-1)\right]}{d\theta_l} \\ &= \frac{\partial\hat{f}\left[\hat{\mathbf{x}}(k-1), \hat{u}(k-1)\right]}{\partial\hat{\mathbf{x}}(k-1)} \cdot \frac{d\hat{\mathbf{x}}(k-1)}{d\theta_l} + \frac{\partial\hat{f}\left[\hat{\mathbf{x}}(k-1), \hat{u}(k-1)\right]}{\partial\hat{u}(k-1)} \cdot \frac{d\hat{u}(k-1)}{d\theta_l}\end{aligned} \tag{11}$$

et:

$$\begin{aligned}\frac{d\hat{u}(k-1)}{d\theta_l} &= \frac{d\hat{\Psi}\left[\hat{\mathbf{x}}(k-1), v(k-1)\right]}{d\theta_l} \\ &= \frac{\partial\hat{\Psi}\left[\hat{\mathbf{x}}(k-1), v(k-1)\right]}{\partial\hat{\mathbf{x}}(k-1)} \cdot \frac{d\hat{\mathbf{x}}(k-1)}{d\theta_l} + \frac{\partial\hat{\Psi}\left[\hat{\mathbf{x}}(k-1), v(k-1)\right]}{\partial\theta_l}\end{aligned} \tag{12}$$

Les équations (10), (11) et (12) peuvent être regroupées pour former le système suivant:

$$\Sigma_G : \begin{cases} \dfrac{d\hat{\mathbf{x}}(k+1)}{d\theta_l} = A_G(k) \cdot \dfrac{d\hat{\mathbf{x}}(k)}{d\theta_l} + B_G(k) \cdot \dfrac{\partial\hat{u}(k)}{\partial\theta_l} \\ \dfrac{d\hat{\mathbf{z}}(k)}{d\theta_l} = C_G(k) \cdot \dfrac{d\hat{\mathbf{x}}(k)}{d\theta_l} \end{cases} \tag{13}$$

avec:

$$A_G(k) = \frac{\partial\hat{f}\left[\hat{\mathbf{x}}(k), \hat{u}(k)\right]}{\partial\hat{\mathbf{x}}(k)} + \frac{\partial\hat{f}\left[\hat{\mathbf{x}}(k), \hat{u}(k)\right]}{\partial\hat{u}(k)} \cdot \frac{\partial\hat{\Psi}\left[\hat{\mathbf{x}}(k), v(k)\right]}{\partial\hat{\mathbf{x}}(k)}$$

$$B_G(k) = \frac{\partial\hat{f}\left[\hat{\mathbf{x}}(k), \hat{u}(k)\right]}{\partial\hat{u}(k)} \quad \text{et} \quad C_G(k) = \frac{\partial\hat{\Phi}[\hat{\mathbf{x}}(k)]}{\partial\hat{\mathbf{x}}(k)}$$

Les équations (13) sont bel et bien celles d'un système linéaire multivariable *non-stationnaire* (à coefficients variant avec le temps), dont les vecteurs d'état, d'entrée et de sortie sont respectivement $d\hat{\mathbf{x}}(k)/d\theta_l$, $\partial\hat{u}(k)/\partial\theta_l$ et $d\hat{\mathbf{z}}(k)/d\theta_l$. $A_G(k)$ et $C_G(k)$ sont toutes deux des matrices carrées d'ordre n, et $B_G(k)$ est un vecteur de dimension n.
Le système Σ_G ainsi défini constitue le modèle de sensibilité relatif au réseau MLP_Ψ. Il est représenté à la figure 4.6.

Une fois la phase d'apprentissage de MLP_Φ et MLP_Ψ est terminée, le vecteur $\hat{\mathbf{z}}(k)$ sera donné par:

$$\begin{aligned}\hat{\mathbf{z}}(k+1) &= \hat{\Phi}\Big[\hat{f}\Big[\hat{\mathbf{x}}(k),\hat{\Psi}\big[\hat{\mathbf{x}}(k),v(k)\big]\Big]\Big] \\ &\cong A_L.\hat{\mathbf{z}}(k)+B_L.v(k)+\mathbf{e}_{l2}\big[\hat{\mathbf{x}}(k),v(k)\big] \qquad (14)\end{aligned}$$

où $\mathbf{e}_{l2}$ est une faible erreur représentant la déviation du système transformé par rapport à un modèle parfaitement linéaire. D'autre part, pour un signal d'entrée $v(k)$ toujours nul, le système linéaire:

$$\mathbf{z}(k+1) = A_L.\mathbf{z}(k)$$

est asymptotiquement stable. En utilisant les résultats établis au paragraphe 1.1.4, on peut montrer qu'il est aussi *fortement stable sous perturbations*; en d'autres termes, pour tout $\varepsilon_0 > 0$, il existe deux nombres réels positifs $\varepsilon_l(\varepsilon_0)$ et $r(\varepsilon_0)$ tels que, si:

$$\|\mathbf{e}_{l2}(\hat{\mathbf{x}},0)\| < \varepsilon_l \quad \text{pour tout } \|\hat{\mathbf{x}}\| < r$$

alors:

$$\hat{\mathbf{z}}(k+1) = A_L.\hat{\mathbf{z}}(k)+\mathbf{e}_{l2}\big(\hat{\mathbf{x}}(k),0\big)$$

convergera vers la boule B_0 de rayon ε_0 autour de l'origine. Cette convergence est obtenue après n étapes au plus; ceci est dû au fait que:

$$A_L^p.\hat{\mathbf{z}}(0) = 0 \quad \text{pour tout } p \geq n$$

Finalement, dans l'hypothèse où $\hat{\Phi}(0) = 0$, le vecteur:

$$\hat{\mathbf{x}}(k+1) = \hat{f}\Big[\hat{\mathbf{x}}(k),\hat{\Psi}\big[\hat{\mathbf{x}}(k),0\big]\Big]$$

convergera de même vers la boule B_0' de rayon ε_0' autour de l'origine, donnée par:

$$B_0' = \hat{\Phi}^{-1}\big(B_0\big)$$

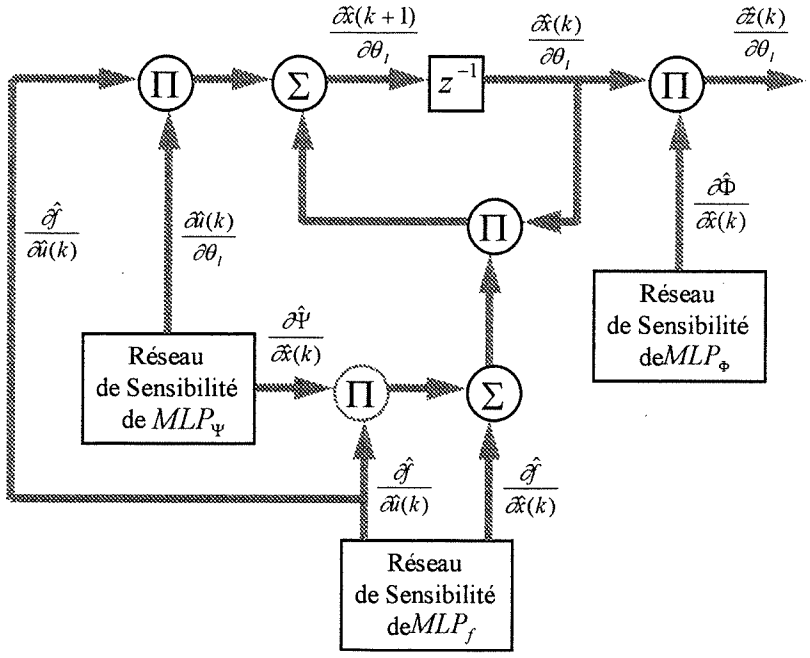

Figure 4.6: Génération du gradient de sortie

4.3.1.1.3 - Stabilisation directe:

La méthode de linéarisation par retour d'état, décrite précédemment, ne peut s'appliquer que pour une certaine classe limitée de processus. De plus, son implantation utilise, dans le cas où c'est possible, l'algorithme de rétro-propagation dynamique qui, comme on l'a déjà mentionné, s'avère très compliqué à mettre au point.
Dans ce paragraphe, des régulateurs non-linéaires permettant de stabiliser d'une manière directe un processus donné seront présentés et implantés à l'aide des réseaux neuronaux artificiels.

a) Exemple d'un système du second ordre:

Soit le système dynamique non-linéaire du second ordre défini par les équations d'état suivantes:

$$\Sigma_2 : \begin{cases} x_1(k+1) = x_1(k) + K.\sin\left[4.u(k) - 2.x_2(k)\right] \\ x_2(k+1) = x_2(k) - 2.u(k) \end{cases} \qquad (15)$$

$u(k)$ est la valeur à l'instant k du signal d'entrée (ou de commande) du système, $\mathbf{x}(k) = [x_1(k), x_2(k)]^T$ le vecteur d'état et K une constante réelle positive caractérisant le

système. Les équations (15) sont celles d'un *jongleur idéal* constitué d'une planche plate sur laquelle une balle rebondit sans cesse, tout en restant dans un plan vertical (figure 4.7). $x_1(k)$ et $x_2(k)$ représentent respectivement la position horizontale de la balle et l'angle que fait sa trajectoire avec l'axe vertical juste avant sa collision élastique, à l'instant k, avec la planche. La direction de la balle est réglée en ajustant la position angulaire de la planche (par rapport au plan horizontal), représentée par $u(k)$ et considérée comme la variable de commande du système.

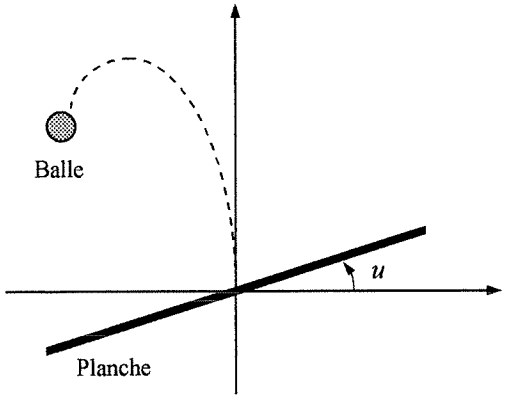

Figure 4.7: Jongleur idéal

a.1) Commande non-linéaire en boucle ouverte:

L'origine du plan d'état, définie par $x_1 = 0$ et $x_2 = 0$, est un point d'équilibre du système Σ_2. La linéarisation Σ_{2L} de Σ_2 autour de ce point est donnée sous forme condensée par:

$$\Sigma_{2L} : \delta\mathbf{x}(k+1) = A.\delta\mathbf{x}(k) + B.\delta u(k)$$

où:

$$\delta\mathbf{x} = \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & -2.K \\ 0 & 1 \end{bmatrix} \quad \text{et} \quad B = \begin{bmatrix} 4.K \\ -2 \end{bmatrix}$$

La matrice de commandabilité associé associée à Σ_{2L} est par définition:

$$M_C = [B|A.B] = \begin{bmatrix} 4.K & 8.K \\ -2 & -2 \end{bmatrix}$$

et son déterminant (égal à $8.K$) est non nul. Par suite, M_C est de rang 2 et le système linéarisé Σ_{2L} est commandable.
D'après les résultats du paragraphe 1.2.2, le système Σ_2 est localement commandable, et il existe un voisinage $V \subset \Re^2$ autour de l'origine et une séquence de commande unique $U_2 = \Psi_u[\mathbf{x}_0, 0]$ qui, à partir d'un état initial $\mathbf{x}_0 = (x_{10}, x_{20})$ appartenant à V, permet de ramener le système à son point d'équilibre en deux étapes au plus. Cette loi de commande est la suivante:

$$u(0) = \frac{1}{4} \cdot \left[2.x_{20} - \arcsin\left(\frac{x_{10}}{K}\right) \right]$$

$$u(1) = \frac{1}{4} \cdot \arcsin\left(\frac{x_{10}}{K}\right)$$

et: $$u(k) = 0 \quad , \quad \text{pour tout } k \geq 2$$

L'inconvénient majeur de la commande en boucle ouverte réside dans le fait qu'elle suppose une connaissance parfaite du processus et, par suite, ne tient pas compte des bruits et perturbations externes pouvant être générés par l'environnement entourant le processus. Pour que le système de réglage soit robuste vis à vis des perturbations, une commande en boucle fermée sera envisagée.

a.2) Stabilisation en boucle fermée:

On a déjà établi la propriété de commandabilité pour le système linéarisé Σ_{2L}. Par conséquent, en se référant au paragraphe 1.2.2, il existe une loi de commande continue $u(k) = g[\mathbf{x}(k)]$ qui rend le système original Σ_2 2-finiment stable dans un voisinage $V \subset \Re^2$ autour de l'origine. Elle est donnée par:

$$u(k) = g\left[x_1(k), x_2(k)\right] = \frac{1}{4} \cdot \left[2.x_2(k) - \arcsin\left(\frac{x_1(k)}{K}\right) \right]$$

b) Elargissement du domaine de commandabilité:

Soit Σ un système localement commandable de la forme (1). Il existe alors un voisinage $V \subset \Re^n$ autour de l'origine et une loi de commande continue $u(k) = g[\mathbf{x}(k)]$ tels que le système bouclé décrit par:

$$\mathbf{x}(k+1) = f\left[\mathbf{x}(k), g\left[\mathbf{x}(k)\right]\right]$$

est n-finiment stable dans V, n étant l'ordre du système. Par conséquent, en posant $F_g(\mathbf{x}) = f^n[\mathbf{x}, g(\mathbf{x})]$, on peut écrire:

$$\forall\ \mathbf{x} \in V\ ,\quad F_g(\mathbf{x}) = 0$$

La fonction F_g étant continue, il existe un ensemble ouvert W tel que $V \subset W$ et, pour tout $\mathbf{x} \in W$:

$$\left\| F_g(\mathbf{x}) - F_g(0) \right\| < \left\| \mathbf{x} - 0 \right\|$$

Sachant que $F_g(0) = 0$, l'inégalité précédente se réduit à:

$$\forall\ \mathbf{x} \in W\ ,\quad \left\| F_g(\mathbf{x}) \right\| < \left\| \mathbf{x} \right\|$$

On en déduit que F_g est une fonction lipschitzienne de contraction et, d'après le théorème du point neutre (donné au paragraphe 1.1.4), on a:

$$\forall\ \mathbf{x} \in W\ ,\quad \lim_{l \to \infty} F_g^l(\mathbf{x}) \equiv \lim_{l \to \infty} f^{nl}\left[\mathbf{x}, g(\mathbf{x})\right] = 0$$

De plus, en définissant pour toute fonction g le système dynamique autonome décrit par:

$$\tilde{\mathbf{x}}(k+1) = F_g\left[\tilde{\mathbf{x}}(k)\right] \qquad (16)$$

alors, d'après ce qui précède, $L(\mathbf{x}) = \left\| \tilde{\mathbf{x}} \right\|$ est une fonction de Lyapounov associée au système (16) sur l'ensemble W.

Notre but est de déterminer une loi de commande $g(\mathbf{x})$ qui rend l'ensemble W aussi large que possible. A cette fin, g doit être choisie de manière à avoir:

$$\left\| \tilde{\mathbf{x}}(k+1) \right\| < \left\| \tilde{\mathbf{x}}(k) \right\|\ ,\quad \text{pour tout } \tilde{\mathbf{x}} \in W$$

Notons que la région W dépend du processus et est au moins aussi large que celle obtenue par un régulateur linéaire.

b.1) Loi de commande globale:

Soit $g_0(\mathbf{x})$ la loi de commande qui rend le système Σ n-finiment stable dans un voisinage $V_0 \subset \Re^n$ entourant l'origine. Pour tout état initial $\mathbf{x}_0$ appartenant à V_0, le système bouclé:

$$\mathbf{x}(k+1) = f\left[\mathbf{x}(k), g_0\left[\mathbf{x}(k)\right]\right]$$

est ramené à son état d'équilibre après n itérations au plus. En d'autres termes, on peut écrire:

$$\forall\ \mathbf{x}_0 \in V_0\ ,\quad F_{g_0}(\mathbf{x}_0) \equiv f^n\left[\mathbf{x}_0, g_0(\mathbf{x}_0)\right] = 0$$

Définissons l'ensemble V_1 de tous les points qui peuvent être amenés dans V_0 après une seule itération par une loi de commande $u_0(\mathbf{x})$, $\mathbf{x} \in V_1$. Il est clair que $V_0 \subset V_1$ et la loi de commande $g_1 : V_1 \rightarrow \Re$ définie par:

$$\forall\ \mathbf{x} \in V_1\ ,\quad g_1(\mathbf{x}) = \begin{cases} g_0(\mathbf{x}) & \forall \mathbf{x} \in V_0 \\ u_0(\mathbf{x}) & \textit{autrement} \end{cases}$$

rend le système Σ (n+1)-finiment stable à l'origine.

D'une manière similaire, on peut définir l'ensemble V_{l+1} des points qui peuvent être conduits dans V_l en une seule étape, et la loi de commande sur V_{l+1} sera telle que:

$$\forall\ \mathbf{x} \in V_{l+1}\ ,\quad g_{l+1}(\mathbf{x}) = \begin{cases} g_l(\mathbf{x}) & \forall \mathbf{x} \in V_l \\ u_l(\mathbf{x}) & \textit{autrement} \end{cases}$$

En faisant tendre l vers l'infini, on aboutit à une loi de commande globale g définie sur un ensemble W aussi large que possible. Les régulateurs réalisant la fonction g ainsi construite sont souvent appelés des *régulateurs à temps minimal*. L'implantation à l'aide de réseaux neuronaux artificiels de ce type de régulateurs n'est possible que dans le cas où la loi de commande g est continue sur W. Cette dernière hypothèse sera maintenue dans les lignes qui suivent.

b.2) Implantation par réseaux neuronaux:

Supposons le système Σ localement commandable, et notons S le domaine de $\Re^n$ où l'on désire stabiliser le système. On se propose de réaliser un régulateur, à base de réseaux neuronaux de type MLP, rendant Σ finiment stable dans S.

On a déjà établi l'existence d'une loi de commande $u = g(\mathbf{x})$ pour laquelle:

i) il existe un ensemble ouvert $V \subset \Re^n$ contenant l'origine et vérifiant:

$$\forall\ \mathbf{x} \in V\ ,\quad F_g(\mathbf{x}) \equiv f^n\left[\mathbf{x}, g(\mathbf{x})\right] = 0$$

ii) il existe un ensemble ouvert W contenant V, sur lequel F_g est une fonction lipschitzienne de contraction.

On supposera par la suite que:

$$S \subset W$$

Notons MLP_f et MLP_g les réseaux neuronaux destinés à approximer respectivement les fonctions f et g. L'ajustement des paramètres de ces deux réseaux se fait d'une manière indépendante: on procède d'abord à l'apprentissage de MLP_f, en utilisant l'une des méthodes exposées dans la troisième partie du rapport, et celui de MLP_g sera ensuite réalisé conformément au schéma de la figure 4.8.
En représentant par $\hat{f}$ et $\hat{g}$ les fonctions réalisées par les réseaux MLP_f et MLP_g respectivement, pour tout point initial $\mathbf{x}_0 \in S$, le vecteur $\hat{\mathbf{x}}_n = \hat{f}^n[\mathbf{x}_0, \hat{g}(\mathbf{x}_0)]$ est obtenu à la sortie d'une association en cascade de n blocs élémentaires, dont chacun est caractérisé par:

$$\hat{\mathbf{x}}(k+1) = \hat{f}\left[\hat{\mathbf{x}}(k), \hat{g}\left[\hat{\mathbf{x}}(k)\right]\right]$$

En rappelant que $\hat{\mathbf{x}}_n$ ne peut être confondu avec l'origine que lorsque $\mathbf{x}_0$ appartient à l'ensemble V (supposé inconnu), l'erreur à la sortie du système d'apprentissage de la figure 4.8 doit être choisie comme suit:

$$\mathbf{e}(\mathbf{x}_0) = \hat{\mathbf{x}}_n \quad \text{si} \quad \|\mathbf{x}_0\| < \rho \quad \text{ou} \quad \|\hat{\mathbf{x}}_n\| > \lambda.\|\mathbf{x}_0\|$$

$$\mathbf{e}(\mathbf{x}_0) = 0 \qquad \text{autrement}$$

où $\rho>0$ et $0<\lambda<1$. ρ est initialement choisi faible et le paramètre λ, caractérisant la contraction sur W, est initialement choisi proche de 1.
Les paramètres libres du système d'apprentissage sont ajustés à l'aide de l'algorithme de rétro-propagation statique. Si celui-ci ne converge pas au bout d'un nombre prédéterminé d'itérations, on diminue la valeur de ρ (ce qui revient à réduire l'ensemble V). Par contre, si les paramètres libres convergent vers des valeurs stables, une augmentation de ρ entraînerait une augmentation de la rapidité et de la robustesse du système de réglage.
De même, λ est initialement choisi voisin de 1 afin de maximiser l'ensemble W. Si la phase d'apprentissage peut être accomplie avec une valeur plus faible de λ, le régulateur obtenu sera plus rapide.

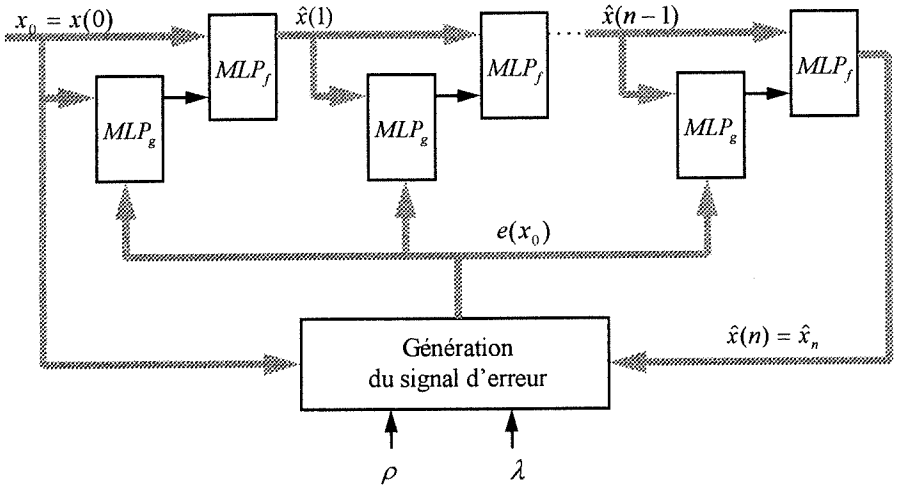

Figure 4.8: Apprentissage d'un stabilisateur direct

Une fois que le régulateur MLP_g a appris à stabiliser le système:

$$\hat{\mathbf{x}}(k+1) = \hat{f}\left[\hat{\mathbf{x}}(k), \hat{g}\left[\hat{\mathbf{x}}(k)\right]\right]$$

sur le domaine d'intérêt S, on pourra l'appliquer au système réel Σ conformément au schéma de la figure 4.9. L'erreur de réglage introduite entre la fonction $\hat{g}$ et celle désirée g dépend de la précision du modèle d'identification MLP_f. Plus $\hat{f}$ est proche de f, moindre sera l'erreur de réglage.

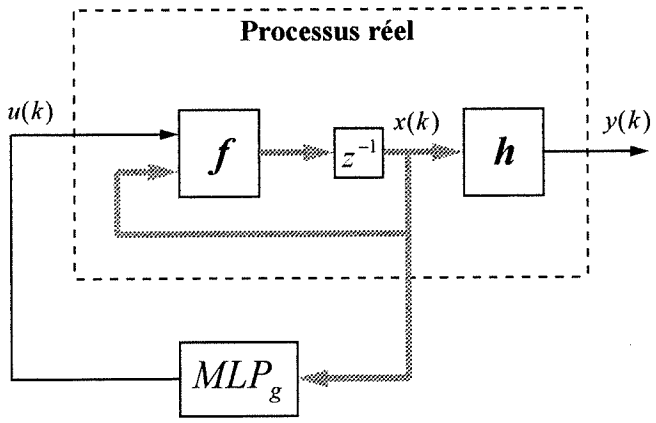

Figure 4.9: Stabilisation directe

4.3.1.2 - Cas du modèle NARMA:

On se propose ci-dessous de décrire une méthode de stabilisation du processus non-linéaire Σ autour de l'origine, dans le cas où celui-ci est représenté par son modèle NARMA donné par:

$$y(k+1) = F\left[y(k),\ldots,y(k-n+1),u(k),\ldots,u(k-n+1)\right] \qquad (17)$$

On sait déjà que, si la linéarisation Σ_L du système Σ est commandable, il existe localement une loi de commande de la forme:

$$u(k) = g\left[\mathbf{x}(k)\right] \qquad (18)$$

qui stabilise Σ autour de l'origine.

D'autre part, si Σ_L est observable, il existe alors localement une fonction continue Λ (se référer au paragraphe 3.4.1) telle que pour toute séquence d'entrée $U_{n-1}(k-n+1)$:

$$\mathbf{x}(k) = \Lambda\left[Y_n(k-n+1),U_{n-1}(k-n+1)\right] \qquad (19)$$

En combinant ensemble les équations (18) et (19), on aboutit à l'expression suivante du signal de commande:

$$u(k) = g \circ \Lambda\left[y(k),\ldots,y(k-n+1),u(k-1),\ldots,u(k-n+1)\right] \qquad (20)$$

La relation (20) représente la loi de commande exprimée en termes d'entrée et de sortie du système auquel on désire l'appliquer.

D'une manière plus générale, en utilisant la notion d'*observabilité générique* définie au paragraphe 1.1.7, la stabilisation de Σ autour de l'origine peut être réalisée à partir de la loi de commande locale:

$$u(k) = g \circ \Lambda'\left[y(k),\ldots,y(k-2.n),u(k-1),\ldots,u(k-2.n)\right] \qquad (21)$$

Dans l'expression (21), les séquences d'entrée et de sortie utilisées pour la génération du signal de commande u sont plus longues que celles introduites dans l'équation (20). On pourra montrer facilement, en se reportant au paragraphe 1.3.2.b), que la validité de l'expression (21) est plus globale que celle de (20).

L'implantation par réseaux neuronaux (perceptrons multi-couches) de la loi de commande (20) suit deux étapes indépendantes. La première étape consiste à synthétiser un estimateur d'état, noté MLP_Λ, conformément à la méthode décrite au paragraphe 3.4.1; ceci ne peut se faire que lorsque le vecteur d'état du processus est accessible. A

l'étape suivante, on procède à l'apprentissage du régulateur MLP_g par la méthode illustrée à la figure 4.8 du paragraphe précédent.

Une fois leur apprentissage est terminé, MLP_Λ et MLP_g sont combinés ensemble et introduits dans le système de réglage illustré à la figure 4.10. Les blocs TDL_p présents dans cette figure représentent des lignes à retard (Tapped Delay Line) dont le vecteur de sortie est formé par les p anciennes valeurs de leur signal d'entrée. Les performances du système de réglage dépendent de l'écart instantané entre l'état estimé $\hat{\mathbf{x}}(k)$ et l'état réel $\mathbf{x}(k)$ du processus. Plus cet écart est faible, meilleure sera la stabilisation.

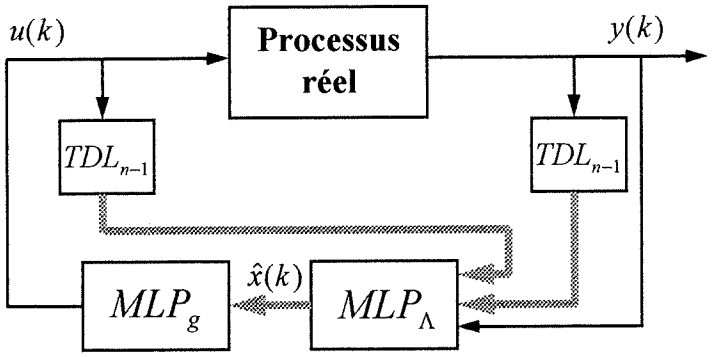

Figure 4.10: Stabilisation utilisant le modèle NARMA du processus

4.3.2 - Problème de régulation:

Le problème de régulation se définit de la manière suivante. Soit Σ un processus non-linéaire décrit par les équations (1), et soit y^* la réponse désirée du système, supposée fixe et prédéterminée. Le but de la régulation est de déterminer une séquence de commande $u(k)$ telle que:

$$\lim_{k\to\infty}\left|y(k)-y^*\right|=0$$

où $y(k)$ est la valeur réelle de sortie du système à l'instant k. Dans le cas particulier où la valeur désirée y^* est nulle, le problème peut être ramené à celui d'une stabilisation.

Le problème de régulation concerne donc la détermination d'une loi de commande $u(k)$ qui permet à la sortie du système Σ de se rapprocher d'une valeur de référence désirée. Cette loi de commande pourrait être générée par le système inverse Σ^{-1} de Σ, dans le cas où celui-ci existe. En effet, le système formé par l'association en cascade de Σ^{-1} et Σ admet un gain unitaire et, par conséquent, débite à sa sortie un signal y dont la valeur est théoriquement égale, à tout instant, à celle de la consigne appliquée à son entrée (choisie égale à y^*). Dans ces conditions, le problème de régulation devient celui de construction d'un système inverse.

L'idée d'utiliser le système inverse d'un processus donné, pour en permettre la régulation, n'est pas nouvelle. Dans les années 60, plusieurs chercheurs ont appliqué cette méthode au cas des processus linéaires. Cette idée fut ensuite généralisée pour permettre la construction de régulateurs pour différentes classes de processus non-linéaires. Récemment, plusieurs auteurs ont suggéré l'utilisation des réseaux neuronaux artificiels pour émuler les systèmes inverses. Cette solution fut appliquée spécifiquement au cas des systèmes statiques pour lesquels un système inverse existe toujours. Celui-ci est considéré comme un système indépendant, et le signal qu'il génère est utilisé pour commander le processus auquel il s'applique.
Cette stratégie de commande en boucle ouverte ne peut cependant être appliquée au cas des systèmes dynamiques, vu qu'elle n'assure pas une robustesse vis à vis des perturbations. Pour remédier à ce fait, la méthode du système inverse sera intégrée dans une stratégie de commande en boucle fermée. Dans ces conditions, la synthèse du régulateur dépendra de l'état $\mathbf{x}(k)$ du processus et de la valeur de référence désirée y^*.

On a vu précédemment que l'association entre le processus et son système inverse engendre un système global de gain unitaire, c'est-à-dire:

$$\forall k \geq 0 \ , \qquad y(k) = y^* \tag{22}$$

où $y(k)$ est la sortie réelle du système et y^* la consigne appliquée à son entrée.

Pour le système dynamique Σ décrit par (1), un système inverse vérifiant la condition (22) ne peut exister car le signal de sortie y ne dépend pas explicitement de celui d'entrée u. On propose alors de faire appel à une variante de la méthode du système inverse, dite *méthode du système inverse retardé*, dont le but est de générer la séquence de commande nécessaire qui permet de ramener la sortie réelle du processus, après un certain temps, à sa valeur désirée imposée par la consigne de référence.

4.3.2.1 - Cas du modèle d'état:

Considérons un processus non-linéaire Σ, d'ordre n, représenté par des équations d'état de la forme (1), et soit d le degré relatif (ou retard) de Σ autour de l'origine. En se

reportant au paragraphe 1.3.3, on déduit qu'il existe une région $\Omega \subset \Re^n \times \Re$ autour de l'origine (supposée asymptotiquement stable) telle que pour tout $(\mathbf{x}, u) \in \Omega$:

$$\left.\frac{\partial h \circ f^k}{\partial u}\right|_{(\mathbf{x},u)} = 0 \qquad \text{pour tout } k < d$$

et:

$$\left.\frac{\partial h \circ f^d}{\partial u}\right|_{(\mathbf{x},u)} \neq 0$$

d correspond intuitivement au premier instant où la sortie du processus se trouve affectée par la valeur du signal d'entrée appliquée à l'instant initial $k = 0$.

D'autre part, la valeur de sortie du système Σ est donnée à l'instant $k + d$ par:

$$y(k+d) = h \circ f\Big[f\Big[\cdots f\Big[f\big[\mathbf{x}(k), u(k)\big], u(k+1)\Big], \cdots\Big], u(k+d-1)\Big] \qquad (23)$$

ou:

$$y(k+d) = h_d\big[\mathbf{x}(k), u(k), \ldots, u(k+d-1)\big] \qquad (24)$$

Puisque $\partial y(k)/\partial u(k-i) = 0$ pour tout $i < d$, l'expression (24) se réduit à:

$$y(k+d) = h_d\big[\mathbf{x}(k), u(k)\big] \qquad (25)$$

Sachant que $\partial y(k+d)/\partial u(k) \neq 0$, il existe localement, d'après le théorème de la fonction implicite (donné au paragraphe 1.1.6), une loi de commande:

$$u^*(k) = g\big[\mathbf{x}(k), y^*\big] \qquad (26)$$

telle que:

$$y(k+d) = y^* \ , \quad \text{pour tout } k \geq 0$$

L'implantation par réseaux neuronaux de la loi de commande (26) se fait en deux étapes:

i) On procède d'abord à l'identification du système régi par l'équation (25). En supposant que deux réseaux neuronaux MLP_f et MLP_h sont ajustés de manière à approximer respectivement, avec une bonne précision, les fonctions f et h caractéristiques du système Σ, l'apprentissage du réseau MLP_{h_d} destiné à identifier la relation (25) est réalisé conformément au schéma de la figure 4.11. Le réseau MLP_{h_d} étant choisi du type perceptron multi-couches, ses paramètres libres sont ajustés à partir de l'algorithme de rétro-propagation statique. Il est à noter que, dans la mesure où la réponse du processus à l'instant $k + d$ est indépendante des valeurs du signal d'entrée

introduites après l'instant k, celles-ci peuvent être supposées nulles de sorte que l'on peut écrire d'après (23):

$$y(k+d) = h \circ f\Big[f\Big[\cdots f\Big[f\big[\mathbf{x}(k), u(k)\big], 0\Big]\cdots\Big], 0\Big]$$

Une fois la phase d'apprentissage subie par MLP_{h_d} est terminée, on peut alors supposer que:

$$\left|\hat{y}(k+d) - y(k+d)\right| = \left|e_d(k)\right| <<< 1 \qquad \text{pour tout } k$$

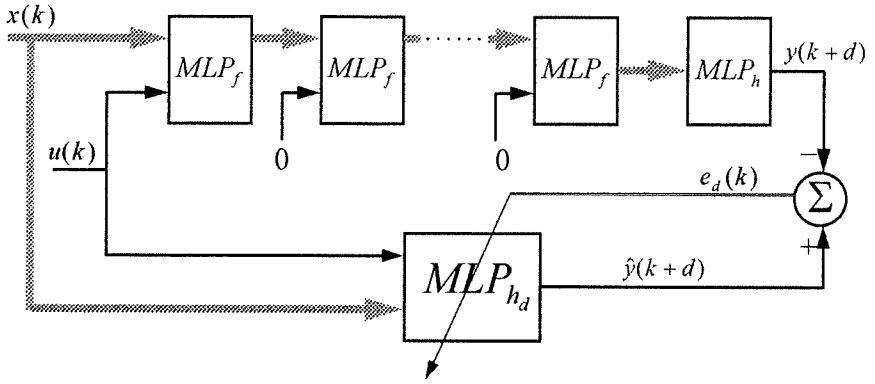

Figure 4.11: Procédé d'identification

ii) La procédure d'implantation de la règle (26) par un réseau neuronal noté MLP_g est illustrée à la figure 4.12. Les paramètres du réseau MLP_{h_d}, qui sert à modéliser le processus, sont gardés fixes, alors que ceux du régulateur MLP_g sont ajustés à l'aide de l'algorithme de rétro-propagation. Si θ_j dénote un paramètre libre quelconque de MLP_g, il est ajusté itérativement suivant la loi de descente du gradient décrite par:

$$\theta_j(k+1) = \theta_j(k) + \eta . e_g(k) \frac{\partial \hat{y}(k+d)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial \theta_j}\bigg|_{\theta_j = \theta_j(k)}$$

où $\eta>0$ est le paramètre d'apprentissage et $e_g(k)$ l'erreur de réglage donnée par:

$$e_g(k) = \hat{y}(k+d) - y^* = \hat{h}_d\Big[\mathbf{x}(k), \hat{g}\big[\mathbf{x}(k), y^*\big]\Big] - y^*$$

$\hat{h}_d$ et $\hat{g}$ représentent les fonctions réalisées par les réseaux MLP_{h_d} et MLP_g respectivement.

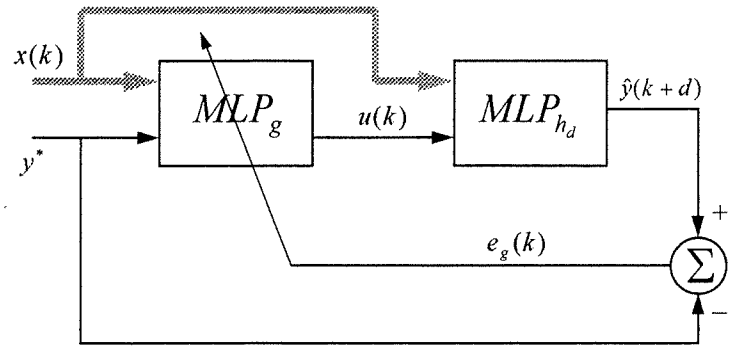

Figure 4.12: Synthèse du régulateur

4.3.2.2 - Cas du modèle NARMA:

Supposons le système Σ fortement observable. Dans ces conditions, le vecteur d'état $\mathbf{x}(k)$ peut être exprimé en fonction d'anciennes observations au niveau de l'entrée et la sortie du système. D'une manière plus spécifique, en se référant au paragraphe 3.4.1, on peut écrire:

$$\mathbf{x}(k) = \Lambda\left[y(k),\ldots,y(k-n+1),u(k-1),\ldots,u(k-n+1)\right] \qquad (27)$$

En substituant la relation (27) dans l'expression (26), on aboutit à une nouvelle forme de la loi de commande:

$$\begin{aligned} u^*(k) &= g\left[\Lambda\left[y(k),\ldots,y(k-n+1),u(k-1),\ldots,u(k-n+1)\right],y^*\right] \\ &= g_\Lambda\left[y(k),\ldots,y(k-n+1),u(k-1),\ldots,u(k-n+1),y^*\right] \end{aligned} \qquad (28)$$

Comme précédemment, la procédure d'implantation de la loi (28) suit les deux étapes suivantes:

i) On commence par identifier à l'aide d'un réseau neuronal $MLP_{\Im_d}$, en tenant compte du degré relatif d, la caractéristique entrée-sortie du processus, donnée par:

$$y(k+d) = \Im_d\left[y(k),\ldots,y(k-n+1),u(k),\ldots,u(k-n+1)\right] \qquad (29)$$

Cette démarche est illustrée à la figure 4.13 où z^{-d} représente un retard de d unités de temps.

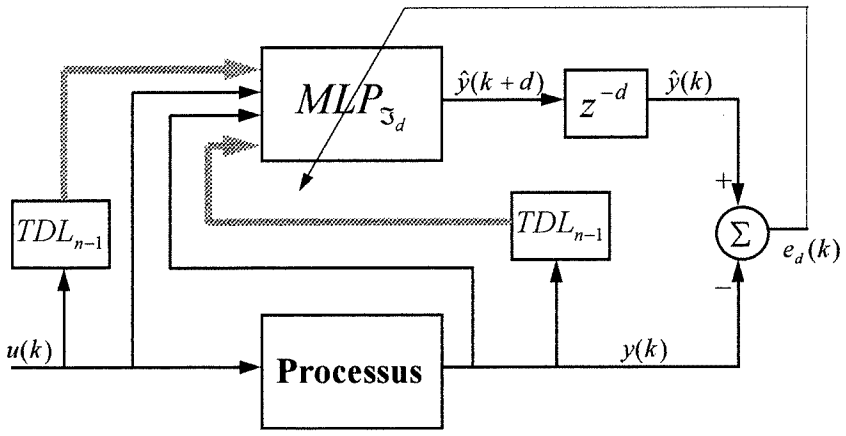

Figure 4.13: Procédé d'identification

ii) Le régulateur défini par (28) est réalisé d'une manière similaire au cas où le processus est représenté par son modèle d'état. Il faudra toutefois remplacer $\mathbf{x}(k)$ par son expression donnée par (27). L'implantation de la loi (28) à l'aide d'un réseau neuronal MLP_{g_Λ} est décrite par le schéma de la figure 4.14.

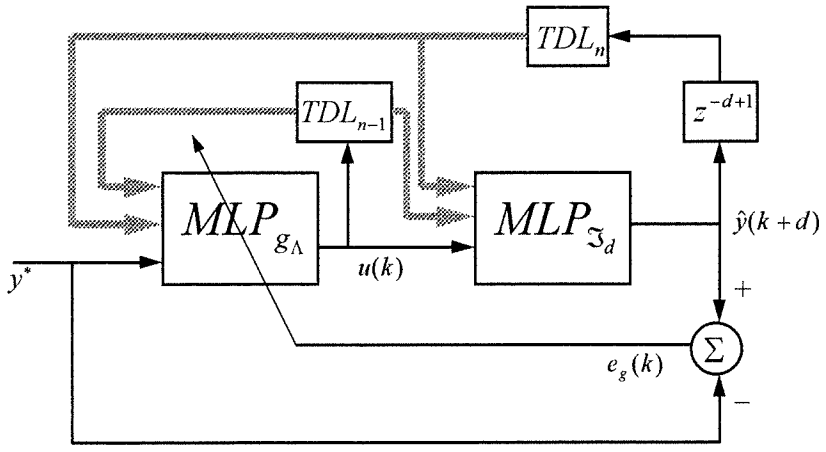

Figure 4.14: Synthèse du régulateur

4.3.3 - Problème de poursuite:

Soit Σ un système non-linéaire de la forme (1), et soit $y^*(k)$ la séquence (supposée uniformément bornée) qu'on désire obtenir à sa sortie. Le problème de poursuite consiste à déterminer une loi de commande telle que:

$$\lim_{k\to\infty}\left|y(k)-y^*(k)\right| = 0 \qquad (30)$$

où $y(k)$ est la valeur réelle du signal de sortie de Σ à l'instant k.

Dans le cas particulier où $y^*(k)$ est constant, quel que soit $k \geq 0$, le problème de poursuite se ramène à celui de régulation décrit au paragraphe précédent.

L'étude théorique du problème de poursuite est identique à celle menée dans le cas d'une régulation, sauf qu'il faut toutefois remplacer la consigne constante y^* par un signal de référence $y^*(k)$ variant avec le temps. En particulier, la méthode du système inverse retardé sera utilisée pour générer la loi de commande vérifiant la condition (30). On suppose que Σ admet un degré relatif d bien défini, et que l'origine est un point d'équilibre asymptotiquement stable du système. Dans ces conditions, en se référant au paragraphe précédent, on peut affirmer qu'il existe localement une séquence de commande $u^*(k)$ telle que:

$$y(k+d) = y^*(k+d) \quad \text{pour tout } k \geq 0$$

Cette loi de commande est de la forme:

$$u^*(k) = g\left[\mathbf{x}(k), y^*(k+d)\right] \qquad (31)$$

En se basant uniquement sur des observations à l'entrée et la sortie du système, on obtient une nouvelle forme de la loi $u^*(k)$ donnée par:

$$u^*(k) = g_\Lambda\left[y(k),\ldots,y(k-n+1),u(k-1),\ldots,u(k-n+1),y^*(k+d)\right] \qquad (32)$$

4.3.3.1 - Définition de la reproductibilité fonctionnelle:

Si la sortie $y(k)$ d'un système Σ peut suivre un signal de référence donné $y^*(k)$, on dit que $y^*(k)$ est *fonctionnellement reproductible* par Σ.

Dans le cas des systèmes linéaires dont le degré relatif est une propriété globale, on peut facilement montrer que toute trajectoire de référence est fonctionnellement reproductible

si le système en question est à minimum de phase (c'est-à-dire, tous les zéro de sa fonction de transfert discrète se situent à l'intérieur du cercle unité dans le plan complexe). Par contre, aucune méthode simple, permettant de déterminer les conditions générales de reproductibilité fonctionnelle, n'existe pour les systèmes non-linéaires. On peut toutefois se contenter du théorème suivant.

4.3.3.2 - Théorème de reproductibilité fonctionnelle:

Soit Σ un système dynamique non-linéaire monovariable de la forme (1). Si Σ est asymptotiquement stable par rapport à l'origine et admet un degré relatif bien défini d dans une région Ω_x autour de l'origine, alors il existe un voisinage Ω_y tel que toute séquence $y^*(k) \in \Omega_y$ est fonctionnellement reproductible par Σ pour tout $k \geq d$.

4.3.3.3 - Implantation de la loi de commande:

Comme dans le cas de la régulation, deux réseaux neuronaux MLP_g et MLP_{g_Λ}, de type perceptron multi-couche, peuvent être utilisés pour identifier les lois de commande (31) et (32) respectivement. Cette procédure est illustrée aux figures 4.15 et 4.16. Les réseaux MLP_{h_d} et $MLP_{\Im_d}$ sont supposés déjà ajustés de manière à approximer les fonctions h_d et $\Im_d$ définies en (25) et (29) respectivement.
Remarquons que la valeur à l'instant k de la consigne $y^*(k+d)$, appliquée à l'entrée des régulateurs MLP_g et MLP_{g_Λ}, correspond à une future valeur (à l'instant $k+d$) du signal de sortie désiré. Un changement au niveau de la sortie du système doit être donc planifié à l'avance.

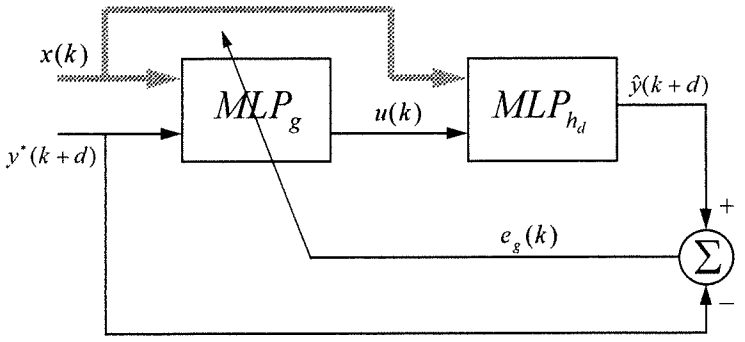

Figure 4.15: Synthèse du régulateur dans le cas d'une représentation en modèle d'état du processus

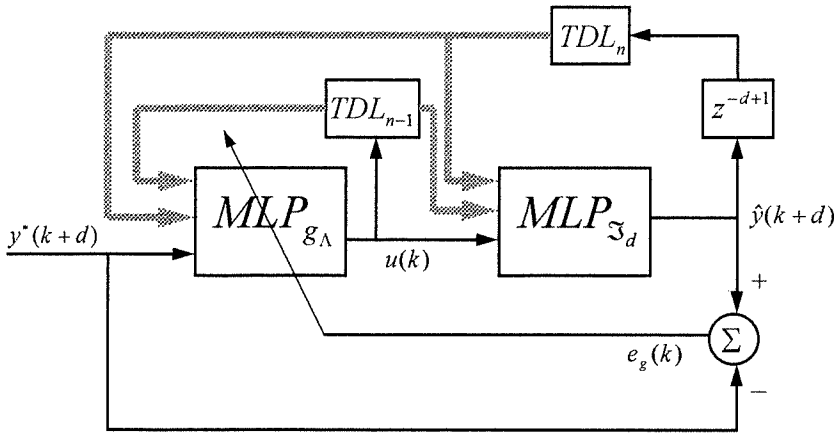

Figure 4.16: Synthèse du régulateur dans le cas d'une représentation en modèle NARMA du processus

4.3.3.4 - Réglage avec modèle de référence:

Considérons le système de réglage représenté à la figure 4.17. Le réseau neuronal $MLP_{\Im_d}$ représente le modèle NARMA, donné en (29), du processus qu'on désire régler. Celui-ci est supposé de la forme (1) et de degré relatif $d \geq 1$. Le signal de sortie désiré y^*, généré par un modèle de référence linéaire, s'exprime à l'instant $k+d$ par:

$$y^*(k+d) = y_m(k) = \sum_{i=1}^{n} \alpha_{mi} . y_m(k-i) + \sum_{j=1}^{m} \beta_{mj} . r(k-j)$$

où r est la consigne à l'entrée du système, et α_{mi} et β_{mj}, $i = 1,\ldots,n$, $j = 1,\ldots,m$, sont des constantes réelles. Le modèle de référence est supposé stable et à minimum de phase; en d'autres termes, les pôles et zéro de sa fonction de transfert discrète sont à l'intérieur du cercle unitaire dans le plan complexe.

On désire obtenir $\hat{y}(k+d) = y_m(k)$ pour tout $k \geq 0$. Par conséquent, d'après la relation (32), l'expression du signal de commande u doit être de la forme:

$$u(k) = g_\Lambda[\hat{y}(k),\ldots,\hat{y}(k-n+1),u(k-1),\ldots,u(k-n+1), \sum_{i=1}^{n} \alpha_{mi} . \hat{y}(k+d-i) + \sum_{j=1}^{m} \beta_{mj} . r(k-j)] \qquad (33)$$

ou:

$$u(k) = g_{m\Lambda}[\hat{y}(k+d-1),\ldots,\hat{y}(k+d-n),\hat{y}(k),\ldots,\hat{y}(k-n+1), u(k-1),\ldots,u(k-n+1),r(k-1),\ldots,r(k-m)] \qquad (34)$$

Si $d > n$, le nombre de noeuds d'entrée que doit avoir le régulateur est maximal et égal à $3.n+m-1$ (c'est le cas de la figure 4.17). Celui-ci devient minimal (égal à $2.n+m-1$) lorsque d est égal à 1.

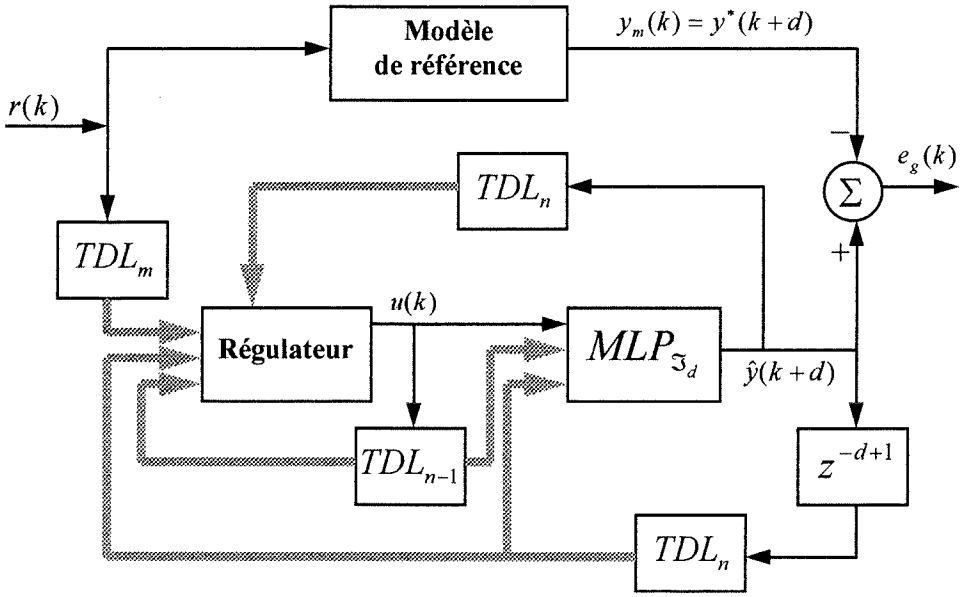

Figure 4.17: Système de réglage par modèle de référence

4.4 - Réjection des perturbations dans les systèmes de réglage:

Dans tous les problèmes de réglage étudiés jusqu'à présent, on n'a fait aucune allusion aux perturbations qui peuvent prendre naissance à l'intérieur ou à l'extérieur du processus à régler. Les paramètres de ce dernier ont été toujours supposés constants de sorte que le système de réglage associé puisse répondre, au moins théoriquement, conformément à un signal de référence désiré $y^*(k)$.

Cependant, en pratique, il existe toujours des perturbations au niveau du processus. Celles-ci peuvent être de nature externe, auquel cas elles sont générées par l'environnement entourant le processus, ou interne, auquel cas elles correspondent aux variations d'un ou plusieurs paramètres internes du processus. La raison d'être du réglage adaptatif est d'adapter la loi de commande aux changements subis par le processus qu'on désire régler. Les performances du système de réglage adaptatif doivent donc être jugées en présence des perturbations, externes soient-elles ou internes.

Il existe déjà, dans la littérature scientifique, des méthodes permettant de rendre *robustes* des systèmes de réglage adaptatif linéaires en présence de perturbations externes bornées et de paramètres variant avec le temps. Un système de réglage est dit *robuste* si, lorsqu'il est soumis à des perturbations de faible amplitude, la déviation entre sa réponse et celle obtenue en l'absence des perturbations est faible. Dans le cas linéaire, la robustesse des systèmes de réglage peut être obtenue en modifiant judicieusement les lois de commande. Dans cette section, on présentera une tentative de généralisation au cas non-linéaire des résultats concernant le réglage adaptatif des processus linéaires, récemment faite par K. S. Narendra et ses collaborateurs [11], appliquée à l'un des problèmes les plus importants rencontrés en pratique, celui de la réjection des perturbations externes.

4.4.1 - Représentation d'un système dynamique perturbé:

4.4.1.1 - Modèle d'état:

Un système dynamique non-linéaire monovariable, soumis à une perturbation externe $v(k)$, est décrit par des équations d'état de la forme:

$$\Sigma_{P(v)}: \quad \begin{aligned} \mathbf{x}(k+1) &= f\left[\mathbf{x}(k), u(k), v(k)\right] \\ y(k) &= h\left[\mathbf{x}(k)\right] \end{aligned} \tag{35}$$

où $\mathbf{x}(k) \in \Re^n$ est le vecteur d'état du système, $u(k) \in \Re$ son signal d'entrée et $y(k) \in \Re$ son signal de sortie. La perturbation $v(k) \in \Re$ est supposée générée par un système dynamique autonome stable, d'ordre m, tel que:

$$\Sigma_v: \quad \begin{aligned} \mathbf{x}_v(k+1) &= f_v\left[\mathbf{x}_v(k)\right] \\ v(k) &= h_v\left[\mathbf{x}_v(k)\right] \end{aligned} \tag{36}$$

avec $\mathbf{x}_v(k) \in \Re^m$.

En combinant les équations (35) et (36), on aboutit au système suivant:

$$\Sigma'_{P(v)}: \quad \begin{aligned} \mathbf{x}'(k+1) &= f'\left[\mathbf{x}'(k), u(k)\right] \\ y(k) &= h'\left[\mathbf{x}'(k)\right] \end{aligned} \tag{37}$$

où $\mathbf{x}'(k) = \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{x}_v(k) \end{bmatrix}$, et les fonctions f' et h' sont telles que:

$$f'\left[\mathbf{x}'(k), u(k)\right] = \begin{bmatrix} f\left[\mathbf{x}(k), u(k), h_v\left[\mathbf{x}_v(k)\right]\right] \\ f_v\left[\mathbf{x}_v(k)\right] \end{bmatrix}$$

et:

$$h'[\mathbf{x}'(k)] = h[\mathbf{x}(k)]$$

La perturbation n'apparaît plus dans le système $\Sigma'_{P(v)}$. Ses effets sont tenus compte par le nouveau vecteur d'état $\mathbf{x}'(k)$ de dimension $n+m$.

4.4.1.2 - Modèle NARMA:

Considérons les linéarisations $\Sigma_{L,P(v)}$ et $\Sigma_{L,v}$ des systèmes (35) et (36) autour de l'origine, définies respectivement par:

$$\Sigma_{L,P(v)}: \begin{array}{l} \mathbf{x}(k+1) = A.\mathbf{x}(k) + B_1.u(k) + B_2.v(k) \\ y(k) = C.\mathbf{x}(k) \end{array} \qquad (38)$$

et:

$$\Sigma_{L,v}: \begin{array}{l} \mathbf{x}_v(k+1) = A_v.\mathbf{x}_v(k) \\ v(k) = C_v.\mathbf{x}_v(k) \end{array} \qquad (39)$$

avec:

$$A = \frac{\partial f[\mathbf{x},u,v]}{\partial \mathbf{x}}\Big|_{(0,0,0)}, \quad B_1 = \frac{\partial f[\mathbf{x},u,v]}{\partial u}\Big|_{(0,0,0)}, \quad B_2 = \frac{\partial f[\mathbf{x},u,v]}{\partial v}\Big|_{(0,0,0)},$$

$$C = \frac{\partial h[\mathbf{x}]}{\partial \mathbf{x}}\Big|_{(0)}, \quad A_v = \frac{\partial f_v[\mathbf{x}_v]}{\partial \mathbf{x}_v}\Big|_{(0)} \text{ et } C_v = \frac{\partial h_v[\mathbf{x}_v]}{\partial \mathbf{x}_v}\Big|_{(0)}$$

On peut montrer que, si les systèmes linéarisés $\Sigma_{L,P(v)}$ et $\Sigma_{L,v}$ sont observables, et si les valeurs propres de la matrice A_v sont différentes des zéro de la fonction de transfert $C.[z.I - A]^{-1}.B_2$, I étant la matrice identité d'ordre n, le système global $\Sigma'_{P(v)}$ est localement observable et peut alors être représenté par un modèle NARMA de la forme:

$$y(k+1) = \Im[y(k), y(k-1), \ldots, y(k-m-n+1), u(k), u(k-1), \ldots, u(k-m-n+1)] \qquad (40)$$

qui est indépendant de la perturbation $v(k)$.

4.4.2 - Réglage du système perturbé:

L'objectif du réglage est de permettre au système global de suivre exactement une certaine trajectoire de référence désirée $y^*(k)$, même en présence de perturbations. Ce but peut être atteint en appliquant à l'entrée du processus une séquence de commande $u(k)$ judicieusement choisie.

En pratique, le vecteur d'état $\mathbf{x}(k)$ du processus et la perturbation $v(k)$ ne sont pas accessibles. La loi de commande sera donc établie à partir du modèle NARMA du processus, donné en (40). En se reportant au paragraphe 4.3.3, cette loi de commande aura la forme suivante:

$$u(k) = g\left[y(k),\ldots,y(k-m-n+1),u(k-1),\ldots,u(k-m-n+1),y^*(k+1)\right] \qquad (41)$$

Dans l'expression (41), on suppose implicitement que le système global $\Sigma'_{P(v)}$ a un degré relatif unitaire. L'implantation par des réseaux neuronaux de cette loi se fait conformément aux méthodes déjà exposées au paragraphe 4.3.3.

4.5 - Réglage adaptatif des systèmes dynamiques multivariables:

Dans la plupart des cas pratiques, le processus physique à régler présente plusieurs entrées et plusieurs sorties. Les méthodes de réglage relativement simples, déjà établies pour les processus dynamiques monovariables, ne peuvent être facilement généralisées au cas des processus multivariables, d'où la nécessité d'une théorie bien développée concernant le réglage de ce type de systèmes.

Même dans le cas d'un processus linéaire de paramètres connus, le problème du réglage des systèmes multivariables s'avère très difficile. Ceci est dû au couplage qui existe entre les différentes entrées et sorties du processus. De même, il n'est guère facile de représenter mathématiquement un système multivariable à cause des différents retards pouvant exister entre chaque paire d'entrée-sortie.

Dans les années 80, le problème du réglage adaptatif des systèmes linéaires multivariables fut intensivement étudié, et des conditions suffisantes pour qu'un système multivariable de paramètres inconnus puisse suivre les signaux de sortie d'un modèle de référence furent établies. Si le processus est non-linéaire, multivariable et de caractéristiques inconnues, on est en présence d'un problème de réglage adaptatif non-linéaire multivariable. Des procédures constructives, permettant de synthétiser des régulateurs satisfaisants pour ce type de problèmes, furent pratiquement inexistantes avant l'apparition des réseaux neuronaux artificiels. C'est dans le réglage adaptatif des systèmes non-linéaires multivariables que les réseaux neuronaux se montrent particulièrement puissants.

Soit Σ un système dynamique à m entrées et m sorties, décrit par les équations d'état (1), et soit Σ_L sa linéarisation autour de l'origine (supposée être un point d'équilibre asymptotiquement stable de Σ), donnée en (2). D'après le paragraphe 1.3.3, Σ_L peut être représentée comme suit:

$$\mathbf{y}_L(k+d) = \begin{bmatrix} y_{L1}(k+d_1) \\ y_{L2}(k+d_2) \\ \vdots \\ y_{Lm}(k+d_m) \end{bmatrix} = \begin{bmatrix} C_1.A^{d_1}.\mathbf{x}(k) + E_1.\mathbf{u}(k) \\ C_2.A^{d_2}.\mathbf{x}(k) + E_2.\mathbf{u}(k) \\ \vdots \\ C_m.A^{d_m}.\mathbf{x}(k) + E_m.\mathbf{u}(k) \end{bmatrix} \tag{42}$$

où, pour tout $i = 1,\ldots,m$, $E_i = C_i.A^{d_i-1}.B$, C_i est le i^{eme} vecteur-ligne de C, et d_i est le degré relatif correspondant à la sortie y_{Li}. Rappelons que les matrices A, B et C, définissant le système linéarisé Σ_L, sont données par:

$$A = \frac{\partial f[\mathbf{x},\mathbf{u}]}{\partial \mathbf{x}}\Big|_{(0,0)} \quad , \quad B = \frac{\partial f[\mathbf{x},\mathbf{u}]}{\partial \mathbf{u}}\Big|_{(0,0)} \quad \text{et} \quad C = \frac{\partial h[\mathbf{x}]}{\partial \mathbf{x}}\Big|_{(0)}$$

Si la matrice carrée E d'ordre m, formée par les m vecteurs-lignes E_i, est non-singulière, on peut choisir un vecteur de commande $\mathbf{u}(k)$ formé d'une combinaison linéaire de $\mathbf{x}(k)$ et du vecteur de sortie désiré $\mathbf{y}_L^*(k+d)$:

$$\mathbf{u}(k) = E^{-1}.\begin{bmatrix} y_{L1}^*(k+d_1) - C_1.A^{d_1}.\mathbf{x}(k) \\ y_{L2}^*(k+d_2) - C_2.A^{d_2}.\mathbf{x}(k) \\ \vdots \\ y_{Lm}^*(k+d_m) - C_m.A^{d_m}.\mathbf{x}(k) \end{bmatrix} \tag{43}$$

En utilisant la même démarche suivie dans le cas des systèmes linéaires, on peut montrer à partir des équations (1) que le vecteur de sortie $\mathbf{y}(k+d)$ du système original Σ peut être exprimé en termes des vecteurs $\mathbf{x}(k)$ et $\mathbf{u}(k)$ comme suit:

$$\mathbf{y}(k+d) = \begin{bmatrix} y_1(k+d_1) \\ y_2(k+d_2) \\ \vdots \\ y_m(k+d_m) \end{bmatrix} = \begin{bmatrix} \Phi_1[\mathbf{x}(k),\mathbf{u}(k)] \\ \Phi_2[\mathbf{x}(k),\mathbf{u}(k)] \\ \vdots \\ \Phi_m[\mathbf{x}(k),\mathbf{u}(k)] \end{bmatrix} \tag{44}$$

Si Σ_L est commandable, alors il existe une loi de la forme:

$$\mathbf{u}(k) = G[\mathbf{x}(k),\mathbf{y}^*(k+d)] = \begin{bmatrix} G_1[\mathbf{x}(k),y_1^*(k+d_1)] \\ G_2[\mathbf{x}(k),y_2^*(k+d_2)] \\ \vdots \\ G_m[\mathbf{x}(k),y_m^*(k+d_m)] \end{bmatrix} \tag{45}$$

qui permet aux sorties du système Σ de suivre des trajectoires de référence désirées après un nombre fini d'étapes.

De plus, si le système Σ est localement observable, $\mathbf{x}(k)$ est une fonction non-linéaire de $\mathbf{y}(k)$ et de ses (n-1) anciennes valeurs, ainsi que des (n-1) anciennes valeurs de $\mathbf{u}(k)$. Ceci nous mène aux représentations entrée-sortie du processus et du régulateur données respectivement par:

$$\mathbf{y}(k+d) = \Im\left[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),\mathbf{u}(k),\ldots,\mathbf{u}(k-n+1)\right] \tag{46}$$

et:

$$\mathbf{u}(k) = \Gamma\left[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),\mathbf{u}(k-1),\ldots,\mathbf{u}(k-n+1),\mathbf{y}^*(k+d)\right] \tag{47}$$

Dans le cas où les fonctions $\Im$ et Γ existent, des réseaux neuronaux peuvent être utilisés pour les approximer.

Dans le but de simplifier le problème de réglage, une version multivariable du modèle approximatif NARMA-L2 (introduit au paragraphe 1.3.2) est souvent utilisée pour représenter le processus Σ. Elle se caractérise par la forme générale:

$$\begin{aligned} y_i(k+d_i) = {} & \Theta_i\left[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),\mathbf{u}(k-1),\ldots,\mathbf{u}(k-n+1)\right] \\ & + \sum_{j=1}^{m} \Psi_{ij}\left[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),\mathbf{u}(k-1),\ldots,\mathbf{u}(k-n+1)\right].u_j(k) \end{aligned} \tag{48}$$

pour tout $i = 1,\ldots,m$. Dans ces conditions, le vecteur de commande $\mathbf{u}(k)$ peut être calculé algébriquement par:

$$\mathbf{u}(k) = \Psi^{-1}.\begin{bmatrix} y_1^*(k+d_1) - \Theta_1\left[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),\mathbf{u}(k-1),\ldots,\mathbf{u}(k-n+1)\right] \\ y_2^*(k+d_2) - \Theta_2\left[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),\mathbf{u}(k-1),\ldots,\mathbf{u}(k-n+1)\right] \\ \vdots \\ y_m^*(k+d_m) - \Theta_m\left[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),\mathbf{u}(k-1),\ldots,\mathbf{u}(k-n+1)\right] \end{bmatrix} \tag{49}$$

où $\Psi = \left[\Psi_{ij}\left[\mathbf{y}(k),\ldots,\mathbf{y}(k-n+1),\mathbf{u}(k-1),\mathbf{u}(k-n+1)\right]\right]$ est la matrice carrée d'ordre m dont les éléments sont donnés en (48).

4.6 - Réglage adaptatif par modèles multiples:

Le réglage adaptatif des systèmes dynamiques complexes pouvant opérer dans plusieurs environnements différents constitue présentement un domaine de recherche très actif. Les variations brusques des paramètres du système, les défauts intervenus au niveau des sous-systèmes et les perturbations externes peuvent tous engendrer des environnements particuliers dans lesquels on souhaite régler le système. Dans de telles situations, étant donné que différents modèles mathématiques seront nécessaires pour représenter le comportement du processus dans chaque environnement, une nouvelle technique de réglage, dite par *modèles multiples*, est utilisée.

4.6.1 - Nécessité des modèles multiples:

La nécessité d'utiliser des modèles multiples dans le réglage des systèmes dynamiques peut avoir plusieurs raisons parmi lesquelles on cite:

i) Plusieurs systèmes physiques peuvent être représentés par une interpolation entre des modèles locaux; le paradigme de réglage connu sous le nom de *planification du gain* est basé sur ce concept.

ii) Des modèles multiples peuvent être utiles pour détecter les différents changements qui se produisent au sein du processus, et initier par la suite l'action de commande appropriée.

iii) Dans certains cas, les données concernant le processus (telles son ordre et son degré relatif), nécessaires à la génération de la loi de commande, ne sont pas toutes disponibles; des modèles multiples peuvent être utilisés pour obtenir l'information désirée.

iv) L'utilisation des modèles multiples permet de combiner leurs avantages; certains modèles pourraient améliorer la stabilité du système, alors que d'autres permettraient d'obtenir de meilleures performances dynamiques; une combinaison appropriée de ces modèles pourrait résulter en un système stable de performances améliorées.

Si des modèles multiples sont utilisés pour l'une des raisons mentionnées ci-dessus, le but du réglage est de détecter, à chaque instant, le modèle qui correspond le plus au processus, et d'appliquer par la suite la séquence de commande appropriée. On est donc amené à introduire deux techniques sur lesquelles se base ce type de réglage, celles de *commutation* et d'*adaptation*. La première permet de prendre des mesures rapides afin d'éviter des défauts catastrophiques, tandis que la dernière permet d'améliorer les performances de la nouvelle configuration du système.

4.6.2 - Structure du régulateur:

L'architecture du système de commutation et d'adaptation est montrée à la figure 4.18. $I_1, I_2, \ldots, I_N$ constituent des modèles d'identification du processus, obtenus après observation du système sur une longue période de temps, et $C_1, C_2, \ldots, C_N$ sont les régulateurs correspondants, générés par les méthodes heuristiques présentées au paragraphe 4.3.

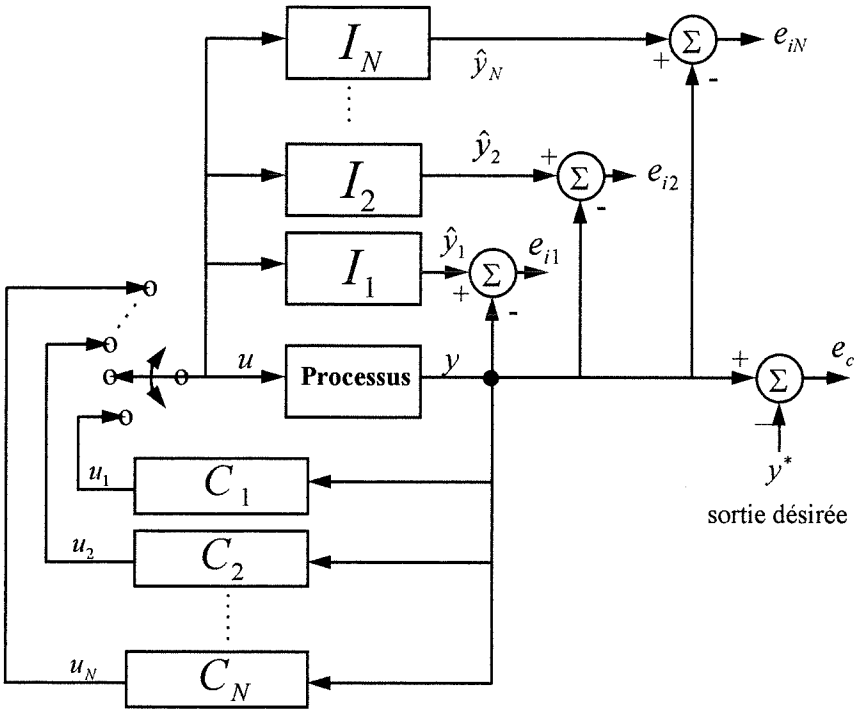

Figure 4.18: Réglage par modèles multiples

En notant $\hat{y}_j(k)$ le signal de sortie du modèle I_j, l'erreur d'identification relative à ce modèle est définie par:

$$e_{ij}(k) = \hat{y}_j(k) - y(k)$$

où $y(k)$ est la sortie réelle du processus. Le choix instantané du modèle approprié du système est basé sur une fonction d'erreur $J(e_{ij})$ évaluée pour $j = 1, \ldots, N$.

Si:

$$J_l(k) \equiv J\left[e_{il}(k)\right] = \min_j J\left[e_{ij}(k)\right]$$

le modèle I_l et le régulateur correspondant C_l sont choisis à l'instant k. Ceci correspond à la *phase de commutation* du réglage adaptatif. De plus, si des réseaux de neurones artificiels sont utilisés pour constituer le système de réglage, l'*adaptation* des régulateurs est nécessaire pour obtenir de bonnes performances du système.

Ainsi, la commutation peut être considérée comme nécessaire au choix des conditions initiales des régulateurs, et l'adaptation s'avère nécessaire au comportement adaptatif subséquent.

Conclusion

Une recherche bibliographique, récemment menée par K. S. Narendra, a révélé que 1960 articles sur les problèmes de réglage par réseaux neuronaux ont été publiés entre 1990 et 1995, dont seulement 353 sont dans le domaine des applications. Parmi ceux-ci, 45% se rapportent à la théorie, 28% décrivent des travaux de simulation, 23% sont en rapport avec des expériences en laboratoire et seulement 4% (14 articles) concernent des applications réelles.
Les applications dans les nouveaux domaines technologiques, tels la robotique, l'aéronautique et l'instrumentation médicale, ont créé un large spectre de problèmes de réglage dans lesquels la non-linéarité, l'incertitude et la complexité jouent un rôle primordial. Afin de pouvoir résoudre ce type de problèmes, des techniques basées sur les réseaux neuronaux artificiels commencent à émerger de manière à compléter les techniques de réglage conventionnelles, et paraissent dans certains cas comme la seule solution pouvant subsister.

Les principes théoriques proposés dans ce rapport, sur lesquels se repose la construction d'identificateurs et de régulateurs par des réseaux neuronaux artificiels, sont basés sur des résultats bien établis dans les domaines du réglage adaptatif linéaire et du réglage non-linéaire. En utilisant les théorèmes de la fonction inverse et de la fonction implicite, les propriétés (stabilité, commandabilité, observabilité) du système linéarisé autour de l'état d'équilibre peuvent être généralisées au domaine non-linéaire. Ceci permet, par la suite, d'étendre au cas des systèmes dynamiques non-linéaires certains résultats établis dans la théorie du réglage linéaire, tels la réjection des perturbations, le découplage des systèmes multivariables et l'adaptation utilisant des modèles multiples.

Beaucoup de travaux de simulation ont révélé que les réseaux neuronaux possèdent un grand potentiel dans les problèmes pratiques de réglage. Les processus dynamiques utilisés sont exclusivement des systèmes non-linéaires auxquels les méthodes de réglage linéaire classiques ne peuvent s'appliquer. Le point le plus impressionnant que ces travaux laissent montrer concerne la capacité remarquable qu'ont les réseaux neuronaux d'identifier et régler des systèmes non-linéaires multivariables complexes à l'aide des données entrée-sortie uniquement.

Les paramètres des modèles d'identification et des régulateurs sont ajustés en utilisant la technique de rétro-propagation ou d'autres méthodes équivalentes basées sur le principe de la descente du gradient. Tandis que la rétro-propagation statique paraît adéquate pour l'identification, l'ajustement des paramètres du régulateur nécessite en général une technique beaucoup plus complexe faisant appel à la rétro-propagation dynamique.

Une meilleure compréhension de la puissance et des limitations des neuro-régulateurs commence à se manifester. Le succès apporté par les études de simulation a encouragé les chercheurs à se tourner vers les problèmes réels, et à exploiter les grandes quantités de données expérimentales rassemblées dans des centres industriels et des laboratoires de recherche. Ceci favorise, par conséquent, l'utilisation des réseaux neuronaux pour l'identification et le réglage en temps réel des systèmes physiques. Dans de telles applications, les réseaux neuronaux seront utilisés pour compléter, et non pas remplacer, les identificateurs et régulateurs linéaires déjà mis en place.

En pratique, la réalisation des systèmes de réglage est orientée suivant les exigences opérationnelles et économiques, et on se rend compte déjà que le passage des principes théoriques, d'une part, au développement, à la mise à l'épreuve et à l'implantation, d'autre part, se fait lentement. Toutefois, en faisant allusion aux progrès déjà faits, on a raison de croire que les systèmes de réglage à base de réseaux neuronaux seront développés dans plusieurs domaines industriels durant les prochaines années, et s'adresseront à une grande variété de problèmes de réglage complexes.

Bibliographie

[1] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks", *IEEE Trans. Neur. Networks*, vol. 1, pp. 4-27, Mar. 1990

[2] A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks: Controllability and stabilization", *IEEE Trans. Neur. Networks*, vol. 4, no. 2, Mar. 1993

[3] S. Mukhopadhyay and K. S. Narendra, "Disturbance rejection in nonlinear systems using neural networks", *IEEE Trans. Neur. Networks*, pp. 63-72, Jan. 1993

[4] K. S. Narendra and S. Mukhopadhyay, "Adaptive control of nonlinear multivariable systems using neural networks", *Neur. Networks*, vol. 7, no. 5, pp. 737-752, 1994

[5] K. S. Narendra, J. Balakrishnan, and K. Ciliz, "Adaptation and learning using multiple models, switching and tuning", *IEEE Contr. Syst. Mag.*, pp. 37-51, June 1995

[6] K. S. Narendra and J. Balakrishnan, "Improving transient response of adaptive control systems using multiple models and switching", *IEEE Trans. Automat. Contr.*, vol. 39, pp.1861-1866, Sept. 1994

[7] A. U. Levin and K. S. Narendra, "Recursive identification using feedforward neural networks", *Int. J. Contr.*, vol. 61, no. 3, pp. 533-547, 1995

[8] K. S. Narendra and S. Mukhopadhyay, "Intelligent control using neural networks", *IEEE Contr. Syst. Mag.*, pp. 11-18, Apr. 1992

[9] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks", *IEEE Trans. Neur. Networks*, vol. 2, pp. 252-262, Mar. 1991

[10] K. S. Narendra and S.-M. Li, "Neural networks in control systems", in *Mathematical Perspectives on Neural Networks*, Smolensky, Mozer, and Rumelhard, Eds. Hillsdale, NJ: L. Erlbaum Assoc.

[11] K. S. Narendra, "Neural networks for control: Theory and practice", *Proc. IEEE*, vol. 84, no. 10, pp. 1385-1406, Oct. 1996

[12] A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks - Part II: Observability, identification, and control", *IEEE Trans. Neural Networks*, vol. 7, no. 1, Jan. 1996

[13] A. U. Levin, K. S. Narendra and D. E. Elliot, "Identification of nonlinear dynamical systems using neural networks", in *Neural Networks for Control*, Norwood, NJ: Ablex, 1994

[14] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, Inc., 1994

[15] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, The MIT Press, 1995

[16] J. Hérault et C. Jutten, *Réseaux Neuronaux et Traitement du Signal*, Hermès, 1994

[17] K. Hornik, M. Stinchcombe, and H. White, "Multi-layer feedforward networks are universal approximators", *Neur. Networks*, vol. 2, pp. 359-366, 1989

[18] T. Miller, R. Sutton, and P. Werbos, *Neural Networks for Control*, Eds. Cambridge, MA: MIT Press, 1990

[19] P. J. Werbos, "Backpropagation through time: What it does and how to do it", *Proc. IEEE*, vol. 78, pp.1550-1560, Oct. 1990

[20] D. Aeyels, "Generic observability of differentiable systems", *SIAM J. Contr. and Opt.*, vol. 19, pp. 595-603, 1981

[21] E. D. Sontag, "On the observability of polynomial systems", *SIAM J. Contr. and Opt.*, vol. 17, pp.139-151, 1979

[22] I. Guyon, "Applications of neural networks to character recognition", *Int. J. of Pattern Recognition and Artificial Intelligence*, vol. 5, pp. 353-382, 1991

[23] J.-F. Jodouin, *Les Réseaux de Neurones: Principes et Définitions*, Hermès, 1994

[24] M. J. D. Powell, "Radial basis functions for multivariable interpolation: a review", *IMA Conf. on Algorithms for the Approximation of Functions and Data*, pp. 143 -167, 1985

[25] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks", *Complex Systems*, vol. 2, pp. 321-355, 1988

[26] J. E. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units", *Neural Computation*, vol. 1, pp. 281-294, 1989

[27] S. Renals, "Radial basis function network for speech pattern classification", *Electronics Letters*, vol. 25, pp. 437-439, 1989

[28] T. Poggio and F. Girosi, "Networks for approximation and learning", *Proceedings of the IEEE*, vol. 78, pp. 1481-1497, 1990

École Polytechnique de Montréal
C.P. 6079, Succ. Centre-ville
Montréal (Québec)
H3C 3A7

ARTIFICIELS DES SYSTÈMES DYNAMIQUES NON-LINÉAIRES