Modélisation et commande de processus par réseaux de neurones ; application au pilotage d'un véhicule autonome

Isabelle Rivals

THÈSE DE DOCTORAT DE L'UNIVERSITÉ PARIS 6

Spécialité :

PHYSIQUE

présentée

par **Isabelle RIVALS**

pour obtenir le titre de **DOCTEUR de l'UNIVERSITÉ PARIS 6**

Sujet de la thèse :

Modélisation et commande de processus par réseaux de neurones ; application au pilotage d'un véhicule autonome.

Soutenue le 20 janvier 1995

devant le jury composé de :

Mme	C.	ALQUIÉ	Présidente
M.	J.P.	CORRIOU	Rapporteur
M.	L.	LJUNG	Rapporteur
M.	M.	de CREMIERS	Examinateur
M.	A.V.	NAZIN	Examinateur
M.	G.	DREYFUS	Examinateur
M.	L.	PERSONNAZ	Examinateur
M.	J.-J.	SLOTINE	Invité

“ Wovon man nicht sprechen kann, darüber muß man schweigen. ”

Ludwig Wittgenstein.

Ce travail a été effectué dans le cadre d'un contrat CIFRE (N°234/91) entre la société SAGEM et le Laboratoire d'Électronique de l'École Supérieure de Physique et de Chimie Industrielles de la Ville de Paris.

L'initiative du projet de pilotage neuronal du robot REMI revient à Monsieur Michel de CREMIERS, Directeur de l'Unité de Recherche en Automatique de la SAGEM. Par l'attention qu'il a portée à mes travaux, et les moyens qu'il a mis à ma disposition, il a contribué efficacement à la réalisation de ce projet. Je le remercie vivement de couronner cette collaboration en acceptant d'être membre de mon jury. J'adresse aussi mes remerciements à Monsieur Gérard FRAPPIER, qui dirige les activités en Robotique de l'Unité de Recherche en Automatique de la SAGEM.

Ce travail a été effectué sous la direction de Monsieur le Professeur Gérard DREYFUS, directeur du Laboratoire d'Électronique de L'ESPCI. Je le remercie très sincèrement de m'avoir initiée aux applications des réseaux de neurones à la Classification, puis à l'Automatique, et pour la confiance qu'il m'a accordée en m'associant aux recherches de son laboratoire. Nos nombreuses et stimulantes discussions ont aussi beaucoup contribué à la clarification des notions ainsi qu'à la présentation du mémoire. Qu'il reçoive l'expression de ma très grande reconnaissance.

Mes plus vifs remerciements vont à Monsieur Léon PERSONNAZ, Maître de Conférences, qui m'a guidée et épaulée tout au long de ces années de thèse. L'ensemble de mon travail lui doit énormément, et en particulier ce mémoire, qui eût été plus concis cependant, si j'avais su tirer meilleur parti de son enseignement.. Qu'il soit salué pour son exigence et sa rigueur exemplaires, ainsi que pour son art accompli de la maïeutique, qui m'ont grandement aidée à structurer et peut-être même à " penser " mon travail de recherche.

Monsieur Daniel CANAS, Ingénieur à la SAGEM, a également suivi de très près mes recherches. Ses précieuses compétences en Automatique, alliées à un intérêt marqué pour les réseaux de neurones, m'ont conduite à approfondir plusieurs parties de mon travail, en particulier celles qui concernent la commande. Qu'il soit chaleureusement remercié pour son enthousiasme, ainsi que pour sa bonne humeur toute méridionale.

Je remercie très sincèrement Madame le Professeur Claude ALQUIÉ d'avoir accepté la présidence de mon jury.

La partie de mon travail concernant la modélisation de processus doit beaucoup aux travaux de Monsieur le Professeur Lennart LJUNG, qui font référence dans ce domaine. Je suis très sensible au grand honneur qu'il m'a fait en acceptant d'examiner mon mémoire.

Je tiens à exprimer ma reconnaissance à Monsieur le Professeur Jean-Pierre CORRIOU, qui a examiné mon mémoire avec attention et rigueur. Je formule le vœu que les concepts et les méthodes

développés dans ce travail puissent être rapidement appliquées au domaine du Génie Chimique qui est le sien.

J'adresse également mes remerciements à Monsieur le Professeur A. NAZIN, pour avoir accepté de faire partie de mon jury, et ainsi témoigné son intérêt pour mes recherches. J'espère que son prochain séjour au Laboratoire me permettra de bénéficier de ses grandes connaissances.

Je remercie vivement Monsieur le Professeur Jean-Jacques SLOTINE, dont les travaux font autorité en commande non linéaire, neuronale en particulier, ainsi qu'en robotique, et dont la participation à la soutenance en tant que membre invité m'honore beaucoup.

Ces années de thèse se sont enrichies de fructueux échanges avec mes collègues de l'ESPCI, dans une ambiance propice à la re-création autant qu'à la réflexion scientifique. Ainsi, mon travail de thèse prolonge et développe-t-il celui d'Olivier NERRAND. Le sujet voisin de Dominique URBANI nous a également conduits à de nombreuses discussions. Je tiens par ailleurs à remercier François ANSELME, qui a mené à son terme la conception du logiciel neuronal " simul_nn " entreprise par Olivier NERRAND, et créé un environnement informatique remarquable. Le bon déroulement de mes simulations sur ordinateur est aussi dû à l'aide efficace et tranquille de Pierre ROUSSEL, qui consacre une part importante de son temps à la bonne santé de notre réseau. Je remercie enfin Yacine OUSSAR pour sa lecture attentive du mémoire.

Je suis très reconnaissante à Monsieur Pierre CONSTANCIS d'avoir accompagné mes premiers pas à bord du robot REMI, ainsi qu'à Messieurs Philippe LEMOINE et Stéphane SALLÉ, Ingénieurs à la SAGEM. J'adresse tout particulièrement mes remerciements à Monsieur Thierry LEBRET, qui a pris un soin constant de REMI, immortalisé en vidéo les essais les plus périlleux, et énergiquement contribué à la réussite des expériences en tout-terrain sur les pistes humides et pentues de l'ÉTAS.

Mes remerciements les plus affectueux vont à Janine RIVALS, pour avoir traqué sans merci les nombreux germanismes, abus d'écriture, raisonnements approximatifs… qui émaillaient encore les dernières versions de ce mémoire.

Je tiens enfin à remercier Mesdames M. BRETAGNOLLE et V. FUCHS pour l'impression de ces volumes.

TABLE DES MATIÈRES

TABLE DES MATIÈRES DÉTAILLÉE

INTRODUCTION GÉNÉRALE

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

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

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

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

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

Les deux parties de ce mémoire sont organisées comme suit :

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

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

Première partie :

MODÉLISATION ET COMMANDE DE PROCESSUS PAR RÉSEAUX DE NEURONES.

Chapitre 1

RÉSEAUX DE NEURONES POUR LA MODÉLISATION ET LA COMMANDE DE PROCESSUS

INTRODUCTION.

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

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

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

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

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

où h est une fonction non linéaire.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

---

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

I. LES RÉSEAUX DE NEURONES.

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

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

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

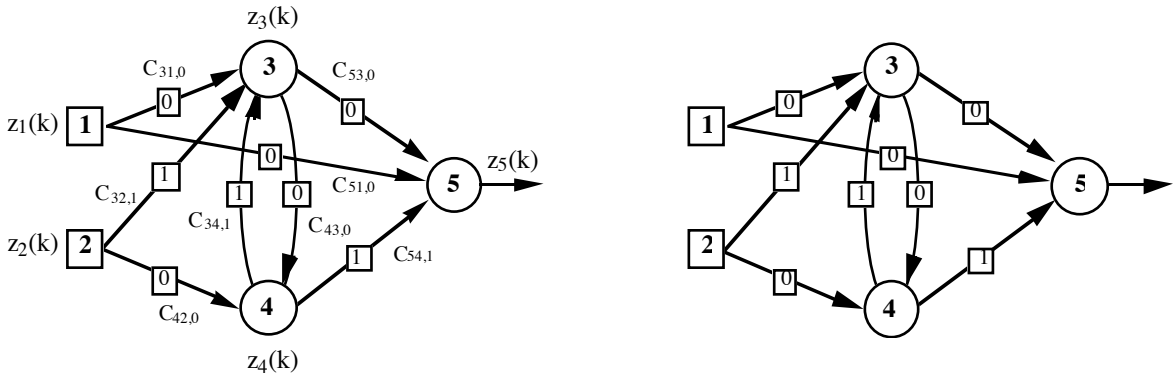


Figure 1.
Exemple de réseau de neurones et de son graphe.

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

Pour caractériser un réseau de neurones, il est pratique d'utiliser son graphe. Ses nœuds sont les neurones, ses racines les entrées du réseau, et les arcs sont les connexions pondérées par leur retard. S'il n'y a pas de cycle dans ce graphe, le réseau est non bouclé, sinon, il est bouclé [NER92b]. L'architecture d'un réseau est définie par le graphe du réseau, les coefficients de celui-ci, et par les fonctions d'activation des neurones. Le caractère bouclé ou non du réseau, ainsi que les fonctions d'activation, peuvent être fixés en fonction de la tâche que doit remplir le réseau de neurones. Les valeurs des coefficients sont en général déterminées par apprentissage, mais certaines d'entre elles peuvent être fixées à l'avance. Ainsi, dans le cas de la modélisation d'un processus, l'architecture

peut être partiellement déterminée par des connaissances *a priori* ; les valeurs de coefficients ayant une signification physique peuvent être fixées préalablement à l'apprentissage.

I.1. LES RÉSEAUX DE NEURONES NON BOUCLÉS.

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

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

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

Un réseau non bouclé réalise donc une transformation entrée/sortie non linéaire ψ_{RN} paramétrée par les coefficients C du réseau (voir figure 2) :

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

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

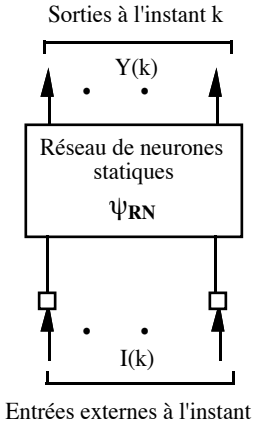


Figure 2.
Forme canonique d'un réseau de neurones non bouclé.

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

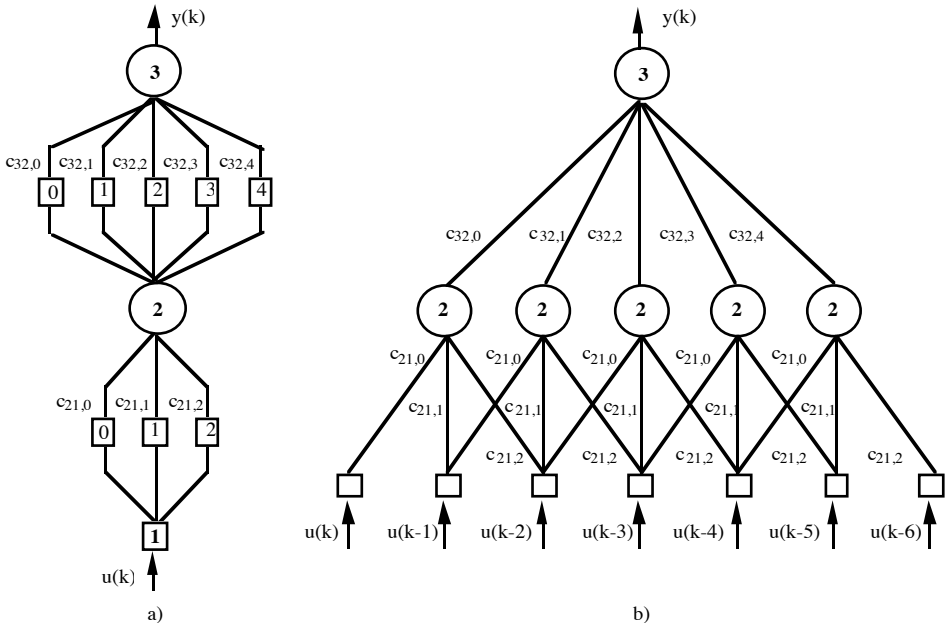


Figure 3.
Mise sous forme canonique d'un filtre transverse non linéaire à temps discret (TDNN).
$I(k) = \left(u(k), u(k-1), u(k-2), u(k-3), u(k-4), u(k-5), u(k-6)\right)^T$; $Y(k) = y(k)$.

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

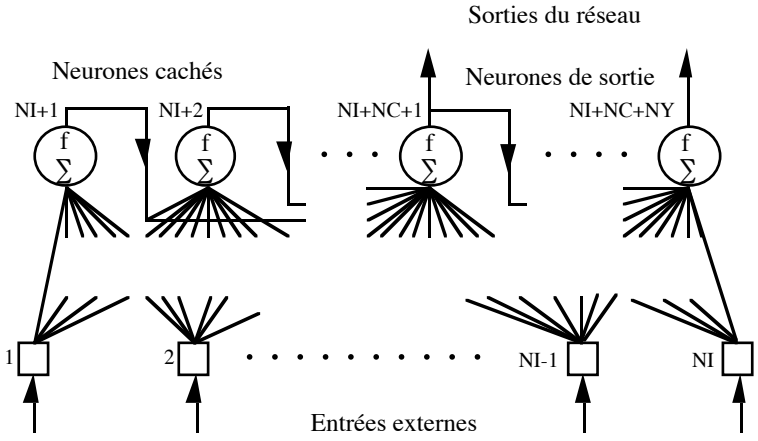


Figure 4.
Réseau de neurones non bouclé complètement connecté.

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

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

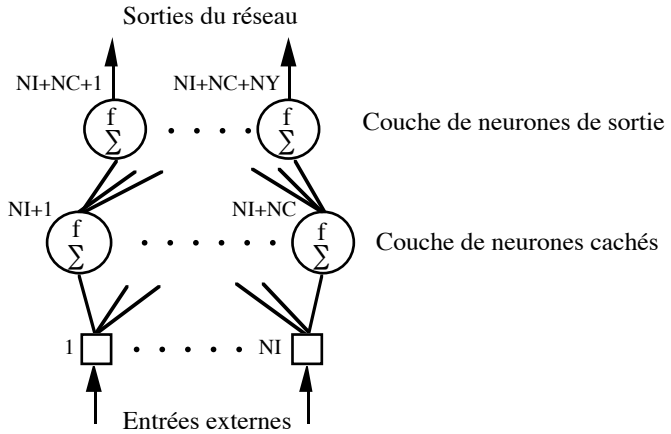


Figure 5.
Réseau de neurones non bouclé à une couche de neurones cachés.

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

I.2. LES RÉSEAUX DE NEURONES BOUCLÉS.

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

Le comportement dynamique d'un réseau de neurones bouclé peut être décrit par la représentation d'état paramétrée par les coefficients C (voir figure 6) :

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

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

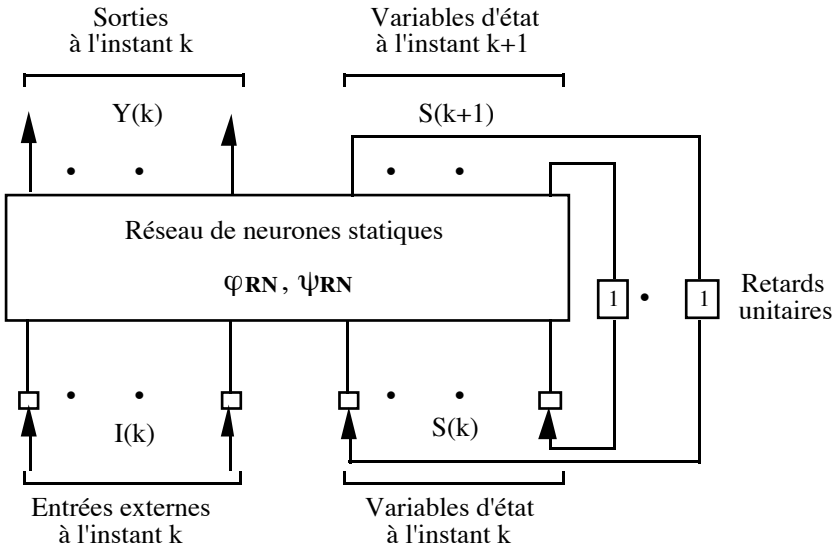


Figure 6.
Forme canonique d'un réseau de neurones bouclé.

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

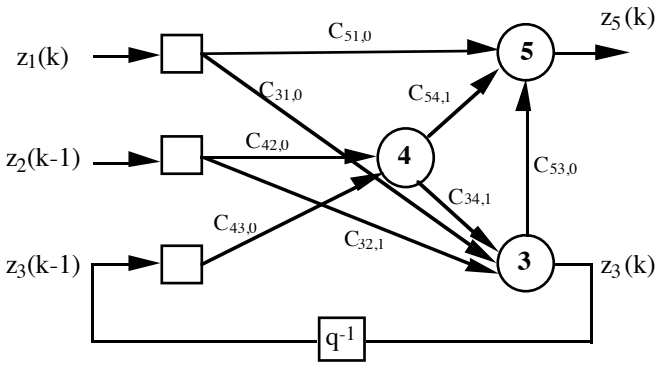


Figure 7.
Mise sous forme canonique du réseau de la figure 1.
$I(k) = \left(z_1(k), z_2(k-1)\right)^T$; $S(k) = z_3(k-1)$; $Y(k) = z_5(k)$.

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

---

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

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

II. PROPRIÉTÉS D'APPROXIMATION DES RÉSEAUX DE NEURONES.

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

II.1. RÉSEAUX NON BOUCLÉS.

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

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

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

II.2. RÉSEAUX BOUCLÉS.

La propriété d'approximation universelle des réseaux de neurones prend un sens différent s'il s'agit d'un réseau bouclé. En effet, considérons un processus représenté par un modèle de type (1) :

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

où f et g sont des fonctions continues. Il existe bien un réseau bouclé tel que, pour des entrées données $x_p(k)$ et u(k), les variables d'état S(k+1) et les sorties Y(k) calculés par le réseau approchent avec une précision fixée l'état $x_p(k+1)$ et la sortie $y_p(k)$ du processus. En effet, il suffit pour cela que le réseau non bouclé de sa forme canonique soit constitué de deux sous-réseaux dont l'un approche la fonction f, et l'autre la fonction g, les conditions d'applications des résultats ci-dessus étant remplies. Mais ceci n'est pas nécessaire pour que le réseau reproduise le comportement entrée-sortie du processus (voir chapitre 2 §I.2.2.2), ni pertinent pour l'approximation d'un système dynamique. En effet, une propriété d'approximation universelle pour les réseaux bouclés peut s'énoncer de la manière suivante : pour tout système dynamique (1), pour toute précision désirée ε, pour un intervalle de temps fini [0;T], pour des entrées u(.) : $[0,T]\rightarrow\mathbb{U}\subseteq\mathbb{R}^{n_u}$ et un état initial $x(0)\in\mathbb{X}\subseteq R^{n_x}$, il existe un réseau de neurones bouclé qui approche le comportement entrée-sortie du système (1) avec la précision ε sur l'intervalle [0;T] et sur les ensembles $\mathbb{U}$ et $\mathbb{X}$ [SON93]. La définition de l'approximation d'un système dynamique par un réseau de neurones bouclé n'est donc pas globale, mais restreinte à un domaine des espaces d'état et d'entrée, sur un intervalle de temps fini : un tel approximateur peut donc ne pas refléter des caractéristiques fondamentales du processus qu'il est censé approcher, sa stabilité par exemple.

III. APPRENTISSAGE DES RÉSEAUX DE NEURONES.

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

Dans le cas où la tâche du réseau de neurones est une tâche de modélisation d'un processus physique, il semble raisonnable de supposer que les sorties mesurées sur le processus obéissent à des lois déterministes, de type (1) par exemple, et de chercher une expression mathématique des fonctions f et g. La propriété d'approximation universelle est donc une propriété nécessaire du modèle utilisé à cette fin, mais elle n'est pas suffisante. En effet :

- d'une part, dans la pratique, les fonctions à déterminer sont définies par un ensemble fini de couples {entrées-sorties mesurées}, qui ne permet pas de déterminer ces fonctions de façon univoque ; le but de l'apprentissage est alors de trouver la solution la plus parcimonieuse, passant par tous les points d'apprentissage, qui, si l'ensemble d'apprentissage est bien choisi, tendra vers les fonctions f et g supposées régir le fonctionnement du processus.

- d'autre part, comme nous le verrons au chapitre 2 consacré à la modélisation, on est souvent en présence de processus affectés de perturbations aléatoires ; dans ce cas, le but de l'apprentissage ne peut être de passer par tous les points de l'ensemble d'apprentissage : bien que le système d'apprentissage ne dispose pas des valeurs prises sur l'ensemble d'apprentissage par les fonctions f et g supposées régir le fonctionnement du processus, il doit ajuster les coefficients du réseau de façon que les fonctions qu'il réalise tendent vers ces fonctions. La mise au point d'un tel système d'apprentissage en fonction des hypothèses faites sur les lois déterministes et les perturbations aléatoires affectant un processus fait l'objet des chapitres 2 et 3 de ce mémoire.

Dans le cas où la tâche du réseau est de réaliser une loi de commande imposant une dynamique désirée à un processus pour lequel on dispose d'un modèle, la démonstration de l'existence d'une telle loi de commande est en elle-même un problème. En effet, si la synthèse du correcteur est effectuée à l'aide d'un modèle neuronal, dont il est difficile de déterminer les caractéristiques de façon analytique, cette existence peut être difficile à établir. Ces problèmes spécifiques à la commande sont abordés au chapitre 5.

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

III.1. PRINCIPE GÉNÉRAL.

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

Tâche d'un réseau de neurones.

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

- deux séquences de nombres, une séquence appliquée aux entrées externes {I(k)}, et une séquence de valeurs désirées correspondantes {D(k)} pour les sorties. Ces séquences constituent *les séquences d'apprentissage*.
- une *fonction de coût* à minimiser : en effet, la tâche ne consiste pas nécessairement à rendre les sorties du réseau égales aux sorties désirées ou " proches " de celles-ci. Par exemple, pour un problème de régulation, on peut souhaiter minimiser également le coût énergétique de la commande. Le critère fera donc intervenir non seulement l'écart de la sortie à la valeur de consigne, mais également l'énergie dépensée. Ou bien, si le processus possède plusieurs sorties,

on peut attacher plus d'importance à certaines d'entre elles ; cela se traduit par une pondération des différents termes de la fonction de coût (c'est une généralisation de la commande linéaire quadratique, voir chapitre 5).

L'apprentissage d'un réseau de neurones est ainsi défini comme un problème d'optimisation qui consiste à trouver les coefficients du réseau minimisant une fonction de coût.

Exemples.

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

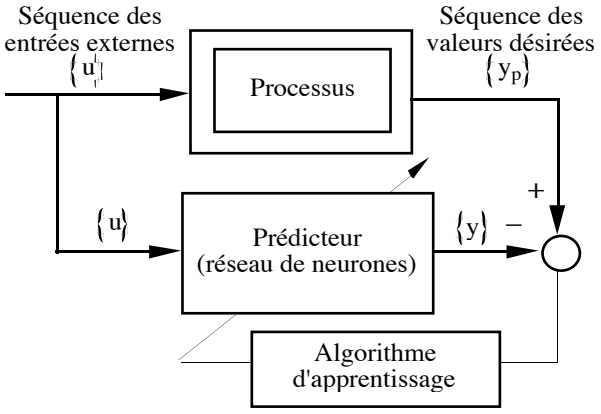


Figure 8.
Système d'apprentissage pour la modélisation d'un processus.

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

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

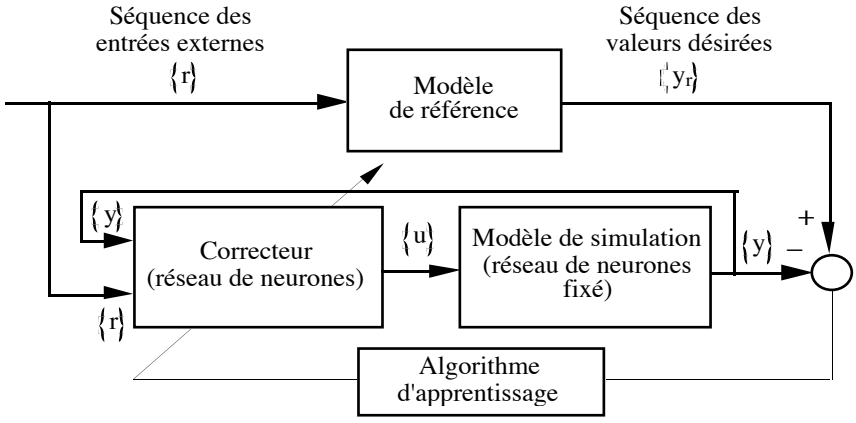


Figure 9.
Système d'apprentissage pour la commande d'un processus.

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

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

III.2. EXPRESSION DE LA FONCTION DE COÛT.

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

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

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

---

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

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

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

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

où C représente les coefficients du réseau C(i-1) disponibles à l'itération i, $E^i(k)$ le vecteur des erreurs à l'instant k et à l'itération i, W une matrice définie positive (qui sera le plus souvent choisie diagonale), D(k) le vecteur des sorties désirées à l'instant k, et $Y^i(k)$ le vecteur des sorties du réseau à l'instant k et à l'itération i.

III.3. MINIMISATION DE LA FONCTION DE COÛT.

Puisque nous utilisons des réseaux de neurones, les sorties du système subissant un apprentissage sont en général des fonctions non linéaires des coefficients à estimer. La recherche du minimum de la fonction de coût ne peut s'effectuer à l'aide des moindres carrés ordinaires, et demande donc l'utilisation de méthodes de programmation non linéaire. Nous présentons dans l'annexe I un algorithme général de calcul du gradient de la fonction de coût dans le cas de l'utilisation de réseaux de neurones, gradient qui est utilisé soit comme direction de descente, soit pour calculer une direction de descente permettant une convergence plus rapide (méthode quasi newtonienne par exemple). L'algorithme est présenté pour tout réseau de neurones bouclé de la forme générale du §I.2, dont le réseau non bouclé est un cas particulier. La présentation est élargie aux réseaux dont l'état n'est pas constitué exclusivement des valeurs retardées des sorties (dits " réseaux d'état ", voir les notations du chapitre 2 §I.1).

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

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

CONCLUSION.

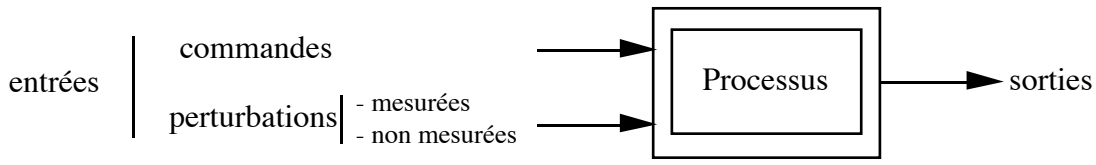
Dans ce chapitre introductif, complété par l'annexe I pour les aspects pratiques de l'optimisation, nous avons présenté les modèles dynamiques universels que sont les réseaux de neurones, et le cadre général de leur apprentissage, élargi aux réseaux (dits " réseaux d'état ", cf chapitre 2 §I.1) dont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

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

PROBLÉMATIQUE DE LA CONCEPTION D'UN SYSTÈME DE COMMANDE

LE PROCESSUS À COMMANDER.

Les processus que nous considérons sont des systèmes physiques qui évoluent au cours du temps, sous l'effet d'influences internes et externes, et sur lesquels on peut faire des observations, c'est-à-dire des mesures. Les signaux qui nous intéressent sont appelées *variables de sortie*. Les grandeurs agissant sur le processus, et donc sur ses sorties, sont appelées *variables d'entrée*. Le processus est affecté par deux types de variables d'entrée : les *commandes*, sur lesquelles on peut agir, et les *perturbations*, sur lesquelles on n'a pas d'action. Parmi ces dernières, on distingue les *perturbations mesurées* et les *perturbations non mesurées*.



Le processus à commander.

Nous nous intéressons à des *processus dynamiques*, c'est-à-dire des processus dont la valeur des sorties à un instant donné ne dépend pas uniquement des valeurs de ses entrées à ce même instant, mais aussi de leurs valeurs passées. Le concept de processus, ou de système physique, est en fait inséparable du concept de *modèle*, conçu comme *système représentatif d'un système physique*. Nous supposons que les processus qui nous intéressent peuvent être convenablement décrits par des modèles mathématiques. De tels modèles sont caractérisés par des *variables d'état*, qui constituent l'information minimale nécessaire au calcul de l'évolution des sorties, si toutes les entrées sont connues.

Commander un processus, c'est déterminer les commandes à lui appliquer, de manière à assurer aux variables d'état ou aux sorties qui nous intéressent un comportement précisé par un cahier des charges. Ces commandes sont délivrées par un *organe de commande* ; le processus et son organe de commande constituent le *système de commande*. L'élaboration de l'organe de commande s'articule en plusieurs étapes.

LES ÉTAPES DE LA CONCEPTION D'UN ORGANE DE COMMANDE.

Choix d'un modèle du processus.

Un modèle du processus est nécessaire à la synthèse de l'organe de commande. La modélisation consiste à rassembler les connaissances que l'on a du comportement dynamique du processus, par une analyse physique des phénomènes mis en jeu, et une analyse des données expérimentales. Ces analyses conduisent à la définition des grandeurs caractérisant le processus, c'est-à-dire ses entrées, ses variables d'état, ses sorties, et aussi les perturbations, mesurables ou non, auxquelles il est soumis. Dans le cadre de ce document, nous faisons l'hypothèse que les sorties du processus sont des fonctions déterministes d'arguments qui sont son état, les commandes, et les perturbations, qu'elles soient mesurées ou non, déterministes ou non. La structure de modèle dont on fait ainsi l'hypothèse qu'elle décrit correctement le processus est appelée *modèle-hypothèse*. En général, la démarche de modélisation conduit à plusieurs modèles-hypothèse concurrents.

Estimation des paramètres du modèle (identification).

Pour un modèle-hypothèse donné parmi ceux retenus, le but de cette étape est de configurer et de sélectionner le meilleur modèle parmi différents modèles de la structure du modèle-hypothèse, sur la base d'un critère de performances. Bien entendu, comme le véritable but de notre démarche est la conception d'un organe de commande à partir d'un modèle, le meilleur d'entre eux est celui qui conduit aux meilleures performances du système de commande, au sens du cahier des charges. Il est évidemment plus économique, et donc préférable, de se fonder sur un critère qui ne nécessite pas la réalisation complète du système de commande pour sélectionner ce modèle : le meilleur modèle est défini comme celui dont l'erreur de prédiction est la plus faible. Pour obtenir ce modèle, il faut le chercher au sein d'une famille de modèles paramétrés. L'estimation des paramètres d'un modèle est donc effectuée de manière à *minimiser l'erreur de prédiction*, à partir de mesures effectuées sur le processus (ensemble d'apprentissage). La famille de modèles paramétrés que nous considérons dans ce travail est celle des réseaux de neurones, qui a l'intérêt de posséder la propriété d'approximation universelle. Leurs paramètres sont les coefficients des réseaux, et leur estimation correspond à la *phase d'apprentissage* dans le " jargon neuronal ". Dans le cadre de ce mémoire, le réseau obtenu en fin d'identification est essentiellement utilisé comme *modèle de simulation* pour la synthèse hors-ligne d'un organe de commande. Nous nous intéressons à des modèles dont l'apprentissage est réalisé *préalablement* à leur utilisation pour la synthèse de l'organe de commande (système non adaptatif).

Conception de l'organe de commande.

Cette étape est celle du choix de l'architecture de l'organe de commande et de la structure des éléments qui le composent, choix effectué en fonction du modèle du processus mis au point et du cahier des charges, qui spécifie les performances désirées pour le système de commande en régulation et, le cas échéant, en poursuite. L'organe de commande comprend nécessairement un élément, le *correcteur,* qui effectue le calcul de la commande à appliquer au processus à partir de la

consigne et de l'état du processus, par exemple. Il comprend en général aussi d'autres éléments : un modèle de référence, un observateur, ou encore un " modèle interne ". Le correcteur peut avoir la structure d'un PID, ou celle d'un correcteur par retour d'état linéaire ou non… : cette structure est choisie en fonction de la tâche, définie par le cahier des charges, que doit remplir le système de commande.

Estimation des paramètres du correcteur.

La dernière étape consiste à configurer le correcteur pour que l'organe de commande assure la tâche définie à l'étape précédente. Cette configuration est effectuée hors-ligne à l'aide du modèle : ceci caractérise les méthodes indirectes de synthèse du correcteur. Comme pour la modélisation, nous considérons des correcteurs réalisés par des réseaux de neurones, dont la structure a été fixée lors de la définition de l'organe de commande. L'estimation des coefficients du correcteur correspond à la *phase d'apprentissage* du réseau de neurones. Nous nous intéressons à la synthèse d'organes de commande *non adaptatifs*, c'est-à-dire pour lesquels l'apprentissage du correcteur est entièrement réalisé *préalablement* à son utilisation avec le processus.

En réalité, la conception d'un organe de commande est une procédure itérative, surtout lorsque celui-ci n'est pas adaptatif. Car, comme nous l'avons signalé plus haut, un réseau modèle ne peut être définitivement rejeté ou validé qu'en fonction des performances du système de commande élaboré à partir de ce modèle.

SUITE DE LA PREMIÈRE PARTIE.

La suite de cette première partie suit le fil conducteur qui vient d'être exposé. Le chapitre 2 présente les différents modèles de processus que nous utilisons. Au chapitre 3, nous traitons de l'estimation des paramètres d'un modèle donné, c'est-à-dire de l'apprentissage et de la sélection du meilleur modèle. Le chapitre 4 expose la mise en œuvre de la modélisation pour deux processus simulés, ainsi que pour l'actionneur d'un bras de robot.

Nous présentons au chapitre 5 les divers systèmes de commande que nous avons étudiés, et l'apprentissage des correcteurs utilisés par ces systèmes. Le chapitre 6 enfin, est consacré à la commande des processus simulés introduits au chapitre 4.

Chapitre 2

MODÈLES DE PROCESSUS

INTRODUCTION.

La modélisation d'un processus consiste à trouver un modèle paramétré dont le comportement dynamique approche celui du processus. Ce modèle sera utilisé pour effectuer des prédictions de la sortie du processus, ou pour l'apprentissage d'un correcteur, ou encore pour simuler le processus au sein d'un système de commande.

La première phase d'une modélisation consiste à rassembler les connaissances que l'on a du comportement dynamique du processus (d'après des expériences et/ou une analyse théorique des phénomènes physiques mis en jeu), ce qui conduit à faire plusieurs hypothèses de structures de modèles susceptibles de décrire ce comportement. Ces structures de modèles, appelées *modèles-hypothèse*, sont caractérisées par le nombre et la nature de leurs variables d'entrée (entrées de commandes ou perturbatrices), d'état et de sortie, et éventuellement par les relations entre ces variables. Par exemple, si l'on a des connaissances précises sur le processus, ces relations peuvent être l'expression de lois physiques. Si ces connaissances sont rudimentaires, on est conduit à choisir des modèles de type " boîte noire " ; on peut également combiner les deux approches au sein d'un même modèle. On est ainsi conduit à un ensemble de modèles-hypothèse concurrents. Chacun d'eux est défini à partir d'une ou de plusieurs fonctions inconnues (ou partiellement connues) d'arguments déterminés, qui vont être réalisées par des fonctions paramétrées, ici des réseaux de neurones.

La seconde phase de la modélisation, aussi dite identification, consiste à estimer les paramètres du modèle. Pour cela, on met en œuvre un système d'apprentissage constitué d'un prédicteur de la sortie du processus associé au modèle-hypothèse, et un algorithme d'apprentissage. L'estimation des paramètres du modèle est effectuée en minimisant une fonction de coût définie à partir de l'écart entre les sorties mesurées du processus (séquences d'apprentissages) et les valeurs prédites. La qualité de cette estimation dépend du modèle-hypothèse choisi, de la richesse des séquences d'apprentissage et de l'efficacité de l'algorithme utilisé.

À l'issue de l'identification relative à chaque modèle-hypothèse, on choisit le modèle neuronal donnant la meilleure performance pour l'utilisation prévue (prédiction, simulation, commande).

Le présent chapitre est consacré à la description des modèles-hypothèse considérés dans ce mémoire, et à la définition du prédicteur associé à un modèle-hypothèse donné, qui est l'élément principal du système d'apprentissage. Le chapitre 3 aborde le problème spécifique de l'estimation des paramètres (identification).

I. MODÈLES-HYPOTHÈSE SANS BRUIT.

I.1. MODÈLES D'ÉTAT ET MODÈLES ENTRÉE-SORTIE.

Dans ce paragraphe, nous précisons les types de modèles-hypothèse sans bruit, c'est-à-dire sans perturbation non mesurée, que nous allons considérer. Les entrées de ces modèles-hypothèse sont donc constituées uniquement des commandes et des perturbations mesurées. Pour la modélisation, ces deux types d'entrées sont désignés globalement sous le nom de commande. La distinction sera réintroduite ultérieurement pour la commande du processus.

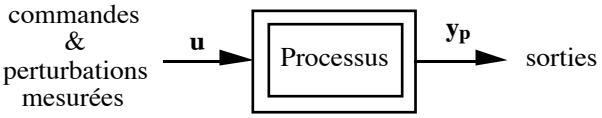


Figure 1.
Modélisation d'un processus sans perturbation non mesurée.

Nous considérons deux classes de modèles-hypothèse :

- la classe des *modèles d'état* :

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

où $x_p \in \mathbb{R}^{n_x}$ est le vecteur d'état, $u \in \mathbb{R}^{n_u}$ est l'entrée de commande, et $y_p \in \mathbb{R}^{n_y}$ est la sortie ; f et g sont des fonctions non linéaires. Nous supposons que tout processus peut être décrit par un modèle-hypothèse d'état.

- la classe des *modèles entrée-sortie* :

$$y_p(k) = h\left(y_p(k\text{-}1), ..., y_p(k\text{-}n), u(k\text{-}1), ...u(k\text{-}m)\right)$$

où $u \in \mathbb{R}^{n_u}$ est l'entrée de commande, et $y_p \in \mathbb{R}^{n_y}$ est la sortie ; h est une fonction non linéaire. Cette classe de modèles est moins vaste que la précédente, mais les modèles entrée-sortie sont souvent plus faciles à mettre en œuvre que les modèles d'état, en particulier si l'état n'est pas mesuré.

Pour simplifier la présentation, nous prendrons $n_u=n_y=1$, et $n_x=n$ (mono-entrée/mono-sortie).

Choix d'un modèle-hypothèse.

Nous faisons l'hypothèse de modèles d'état ou entrée-sortie essentiellement en fonction des possibilités d'analyse physique du processus. Deux cas sont envisageables :

a) Une modélisation physique approfondie peut être effectuée.

Cette modélisation conduit à l'hypothèse d'un modèle d'état, dont les variables d'état ont une signification physique.

Si les variables d'état sont mesurées, les données utilisées pour la modélisation sont la séquence des entrées, et les séquences des variables d'état et des sorties mesurées (séquences d'apprentissage). On peut estimer la fonction f de l'équation d'état à l'aide d'un modèle entrée-sortie, dont les sorties sont les variables d'état du modèle d'état ; l'équation d'observation (fonction g) est modélisée séparément.

Si les variables d'état ne sont pas mesurées, les séquences d'apprentissage sont alors la séquence des entrées et la séquence des sorties mesurées. Deux démarches sont possibles :

- on peut chercher à modéliser le comportement entrée-sortie du processus à l'aide d'un modèle d'état, mais dont l'état n'est pas imposé ;
- on peut faire l'hypothèse plus restrictive d'un modèle entrée-sortie.

Notons que dans ce dernier cas, la meilleure solution, si elle est praticable, est quand même de mettre des capteurs pour mesurer les variables d'état. C'est-à-dire que les nécessités de la commande doivent être prises en considération par l'automaticien dès la conception du processus.

b) Le processus est trop complexe pour être modélisé physiquement.

En pratique, cette situation est équivalente au cas où une modélisation physique est possible, mais où l'état n'est pas mesuré. En effet, les mêmes démarches sont envisageables :

- tenter de modéliser le comportement global du processus à l'aide d'un modèle d'état, sans imposer l'état.
- faire l'hypothèse plus restrictive d'un modèle-hypothèse entrée-sortie.

La mise au point d'un modèle entrée-sortie ou d'un modèle d'état dans ces dernières conditions repose donc surtout sur les résultats expérimentaux. Cependant, comme nous l'avons indiqué dans l'introduction, il peut être très utile, voire indispensable dans le cas de processus complexes, de tirer profit de connaissances *a priori*, même partielles, sur les phénomènes physiques dont le processus est le siège, pour établir la structure de ces modèles. En particulier, ces connaissances peuvent être utilisées pour évaluer l'ordre du modèle.

Notations.

Dans le cas d'un modèle-hypothèse d'état, nous notons $x_p \in \mathbb{R}^n$ l'état du processus et $y_p \in \mathbb{R}$ sa sortie, $x \in \mathbb{R}^n$ l'état du prédicteur et $y \in \mathbb{R}$ sa sortie, et $u \in \mathbb{R}$ la commande. Dans le cas d'un modèle-hypothèse entrée-sortie, nous notons $y_p \in \mathbb{R}$ la sortie du processus et $y \in \mathbb{R}$ celle du prédicteur, et $u \in \mathbb{R}$ la commande.

a) La notation :

$$y(k+1) = \psi_{RN}(\ \ y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1); C)$$

désigne un réseau prédicteur non bouclé où ψ_{RN} est la fonction réalisée par le réseau muni des coefficients C (voir figure 2).

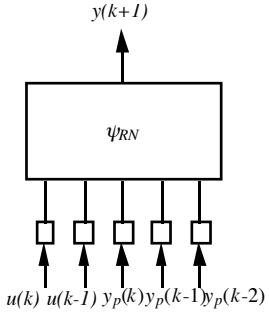


Figure 2.
Exemple de réseau non bouclé avec : n=3 ; m=2.

b) La notation :

$$y(k+1) = \varphi_{RN}(\ \ y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-m+1); C)$$

désigne de façon compacte un réseau prédicteur bouclé d'ordre n, dont les n variables d'état sont la sortie et n-1 valeurs retardées de celle-ci. φ_{RN} est la fonction réalisée par la partie non bouclée du réseau munie des coefficients C (les connexions à coefficients fixés égaux à 1 pour la propagation des n-1 variables d'état sont implicites dans cette écriture, voir annexe I). Un tel réseau est appelé *réseau bouclé entrée-sortie* (voir figure 3).

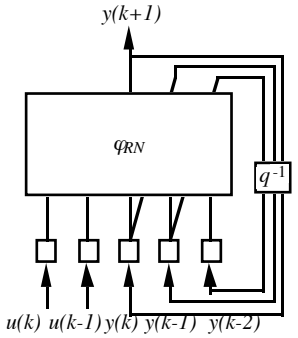


Figure 3.
Exemple de réseau bouclé entrée-sortie avec : n=3 ; m=2.

c) La notation :

$$y(k+1) = \varphi_{RN}(\ \ \ y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1), e(k), \ldots, e(k-p+1); C)$$

désigne un réseau prédicteur bouclé d'ordre p, dont les p variables d'état sont les p valeurs successives de l'erreur $e(k) = y_p(k) - y(k)$. φ_{RN} est la fonction réalisée par la partie non bouclée du réseau munie des coefficients C (les connexions à coefficients fixés égaux à 1 pour la propagation des p-1 variables d'état sont implicites dans cette écriture). Un tel réseau est appelé *réseau bouclé entrée-erreur* (voir figure 4).

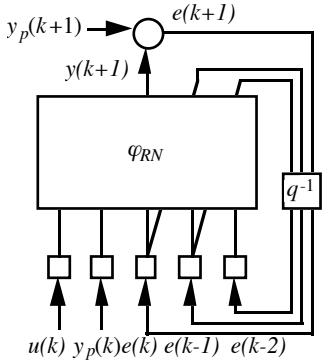


Figure 4.
Exemple de réseau bouclé entrée-erreur avec : n=1 ; m=1 ; p=3.

d) La notation :

$$\begin{cases} x(k+1) = \varphi_{RN}(\ x(k), u(k); C) \\ y(k+1) = \psi_{RN}(\ x(k), u(k); C) \end{cases}$$

désigne un réseau prédicteur bouclé dont les variables d'état sont les activités de neurones d'état distincts des neurones de sortie, donc pour lesquels il n'y a pas de valeurs désirées. φ_{RN} et ψ_{RN} sont les fonctions réalisées par la partie non bouclée du réseau munie des coefficients C. Un tel réseau est appelé *réseau d'état*. (voir figure 5a).

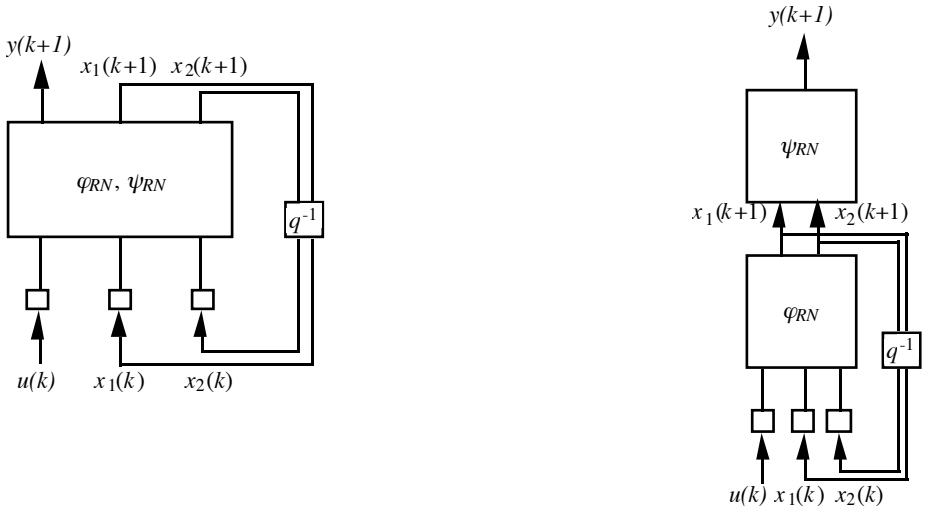


Figure 5.
Exemple de réseau prédicteur d'état avec : n=2.

Lorsque l'on impose une dépendance exclusive de y(k+1) en les variables d'état x(k+1), on note le réseau (voir figure 5b) :

$$\begin{cases} x(k+1) = \varphi_{RN}\left(x(k), u(k); C\right) \\ y(k+1) = \omega_{RN}\left(x(k+1); C\right) \end{cases}$$

I.2. PRÉDICTEURS ASSOCIÉS AUX MODÈLES-HYPOTHÈSE SANS BRUIT.

Définition du prédicteur associé au modèle-hypothèse

Dans le cas où le modèle-hypothèse est complètement déterministe, la notion de prédicteur associé est évidente : c'est un prédicteur qui, si l'hypothèse est vraie, permet de prédire *sans erreur* la sortie du processus.

Pour la clarté de l'exposé, nous commençons par la présentation des modèles-hypothèse entrée-sortie. Comme nous l'avons vu au §I.1, dans le cas d'un modèle-hypothèse d'état, si l'état défini par la modélisation est mesuré, la modélisation est effectuée à l'aide d'un modèle entrée-sortie. Sinon, la modélisation peut être menée d'une manière spécifique que nous décrivons. Enfin, nous montrons qu'il est presque toujours légitime de faire l'hypothèse d'une représentation entrée-sortie, et donc de réaliser la modélisation correspondante, mais au prix d'une augmentation du nombre des arguments du prédicteur associé (§I.2.2.3).

I.2.1. Modélisation entrée-sortie.

Nous appelons ces modèles NDARMA (Non Linéaire Déterministe AutoRégressif à Moyenne Ajustée). Ces modèles sont de la forme :

$$\boldsymbol{y_p(k)} = \boldsymbol{h}\left(\boldsymbol{y_p(k\text{-}1)}, \ldots, \boldsymbol{y_p(k\text{-}n)}, \boldsymbol{u(k\text{-}1)}, \ldots, \boldsymbol{u(k\text{-}m)}\right)$$

Deux prédicteurs sont possibles :

* le prédicteur associé non bouclé :

$$\boldsymbol{y(k+1)} = \boldsymbol{h}\left(\boldsymbol{y_p(k)}, \ldots, \boldsymbol{y_p(k\text{-}n+1)}, \boldsymbol{u(k)}, \ldots, \boldsymbol{u(k\text{-}m+1)}\right)$$

* le prédicteur associé bouclé :

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

avec des conditions initiales correctes.

h est donc la fonction inconnue que l'on désire approcher à l'aide d'un réseau de neurones. Lors de la phase d'identification, le système d'apprentissage peut indifféremment utiliser les prédicteurs neuronaux suivants :

* un prédicteur neuronal non bouclé :

$$y(k+1) = \psi_{RN}\left(y_p(k), \ldots, y_p(k\text{-}n+1), u(k), \ldots, u(k\text{-}m+1); C\right)$$

* un prédicteur neuronal bouclé entrée-sortie :

$$y(k+1) = \varphi_{RN}\left(y(k), \ldots, y(k\text{-}n+1), u(k), \ldots, u(k\text{-}m+1); C\right)$$

Si l'hypothèse est vraie, si la fonction h est dans la famille définie par le réseau de neurones, si l'ensemble d'apprentissage est suffisamment riche, et si l'algorithme est performant, alors la prédiction de la sortie du processus y_p sera très bonne dans le domaine des entrées du réseau défini par l'ensemble d'apprentissage.

I.2.2. Modélisation d'état.

Nous faisons l'hypothèse qu'il existe une relation de la forme :

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

Nous envisageons successivement l'hypothèse où l'état est mesuré, et celle où il ne l'est pas. Pour des raisons que nous explicitons, la deuxième hypothèse conduit à un modèle difficile à manipuler pour la commande du processus. Nous établissons donc ensuite à quelle condition il existe un modèle-hypothèse entrée-sortie équivalent au modèle-hypothèse d'état (§I.2.2.3), mieux adapté à la synthèse et à la mise en œuvre de systèmes de commande. Ce problème a été abordé notamment dans [LEV92] et [NAR92], dont nous utilisons les résultats.

I.2.2.1. L'état est mesuré.

Si l'état est mesuré, il est possible de prédire l'état et la sortie du processus. Deux prédicteurs sont associés au modèle-hypothèse d'état :

* le prédicteur associé non bouclé :

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

* le prédicteur associé bouclé entrée-sortie :

$$\begin{cases} \boldsymbol{x}(\boldsymbol{k}+1) = \boldsymbol{f}\left(\boldsymbol{x}(\boldsymbol{k}), \boldsymbol{u}(\boldsymbol{k})\right) \\ \boldsymbol{y}(\boldsymbol{k}+1) = \boldsymbol{g}\left(\boldsymbol{x}(\boldsymbol{k}+1)\right) \end{cases}$$

avec des conditions initiales correctes.

Le système d'apprentissage utilise donc un premier réseau non bouclé ψ^f_{RN} pour approcher la fonction f :

$$x(k+1) = \psi^f_{RN}\left(x_p(k), u(k); C_f\right)$$

ou un réseau bouclé entrée-sortie φ^f_{RN} :

$$x(k+1) = \varphi^f_{RN}\left(x(k), u(k); C_f\right)$$

Il utilise également un second réseau non bouclé ψ^g_{RN} pour approcher la fonction g de l'équation d'observation :

$$y(k) = \psi^g_{RN}\left(x_p(k); C_g\right)$$

Les séquences d'apprentissage sont constituées de la séquence des entrées {u(k)}, et des séquences des variables d'état $\{x_p(k)\}$ et des sorties mesurées $\{y_p(k)\}$ du processus.

I.2.2.2. L'état n'est pas mesuré.

Si l'état n'est pas mesuré, seule la modélisation du comportement entrée-sortie du processus peut être réalisée. Comme nous l'avons vu au §I.2.2.1, un prédicteur d'état bouclé associé au modèle-hypothèse d'état est évidemment :

$$\begin{cases} \boldsymbol{x}(\boldsymbol{k}+1) = \boldsymbol{f}\left(\boldsymbol{x}(\boldsymbol{k}), \boldsymbol{u}(\boldsymbol{k})\right) \\ \boldsymbol{y}(\boldsymbol{k}+1) = \boldsymbol{g}\left(\boldsymbol{x}(\boldsymbol{k}+1)\right) \end{cases}$$

où x est l'état du prédicteur bouclé.

Le comportement entrée-sortie du processus peut donc théoriquement toujours être identifié à l'aide d'un réseau d'état de la forme :

$$\begin{cases} x(k+1) = \varphi_{RN}\left(x(k), u(k); C\right) \\ y(k+1) = \omega_{RN}\left(x(k+1); C\right) \end{cases}$$

On peut en effet approcher les fonctions f et g avec la précision désirée par la partie non bouclée du réseau φ_{RN} et par le réseau ω_{RN}. Cependant, les séquences d'apprentissage étant constituées de la séquence des entrées {u(k)}, et de la séquence des sorties mesurées $\{y_p(k)\}$ du processus, le même comportement entrée-sortie peut être obtenu avec d'autres couples de fonctions. En effet, pour toute transformation inversible $x = \Phi(x_p)$ de l'état, les prédicteurs de la forme suivante sont équivalents :

$$\begin{cases} x(k+1) = f'\left(x(k), u(k)\right) \\ y(k+1) = g'\left(x(k+1)\right) \end{cases} \text{ avec } \begin{cases} f'\left(. , .\right) = \boldsymbol{\Phi}\left(f\left(\boldsymbol{\Phi}^{-1}(.), .\right)\right) \\ g'\left(.\right) = g\left(\boldsymbol{\Phi}^{-1}(.)\right) \end{cases}$$

L'identification au moyen d'un prédicteur neuronal d'état est donc susceptible de conduire à un modèle dont le comportement entrée-sortie est proche de celui du processus, mais rien n'impose à l'état du réseau de neurones d'être identique à celui du processus (on sait seulement qu'il doit exister entre les états x et x_p une transformation Φ respectant les relations ci-dessus).

Remarque pratique.

Le prédicteur associé peut également s'écrire :

$$\begin{cases} x(k+1) = f\big(x(k), u(k)\big) \\ y(k+1) = g\big(f\big(x(k), u(k)\big)\big) \end{cases}$$

Il est donc également possible d'utiliser un réseau de neurones de la forme générale :

$$\begin{cases} x(k+1) = \varphi_{RN}\big(x(k), u(k); C\big) \\ y(k+1) = \psi_{RN}\big(x(k), u(k); C\big) \end{cases}$$

Dans la pratique, nous préférons cette forme à la précédente, qui n'impose pas l'utilisation de deux sous-réseaux distincts (voir les notations du §I.1, et les exemples des chapitres 3 et 4).

L'utilisation d'un modèle d'état est intéressante si le but de la modélisation est uniquement de décrire le comportement entrée-sortie du processus. Cependant, si l'on souhaite utiliser le prédicteur d'état pour élaborer un correcteur par retour d'état pour le processus, le correcteur ne pourra être utilisé qu'au sein d'un système de commande utilisant le même prédicteur pour prédire l'état, c'est-à-dire un système avec *modèle interne*. Or il n'est pas toujours possible de mettre en œuvre ces systèmes de commande avec des réseaux de neurones, en particulier si l'inverse du prédicteur est instable (cf. chapitre 5). Pour ne pas limiter les possibilités de commande, il est donc souhaitable de se ramener à une représentation entrée-sortie du processus. Nous étudions maintenant cette alternative.

I.2.2.3. Existence d'un modèle entrée-sortie équivalent.

Le but de ce paragraphe est de donner les conditions pour que l'hypothèse d'existence d'un modèle entrée-sortie soit vraie, et de caractériser cette représentation entrée-sortie. Les conditions pour qu'un modèle d'état non linéaire possède une représentation entrée-sortie dans un domaine donné de son espace d'état sont intimement liées à l'observabilité du modèle. En effet, écrivons le comportement de sa sortie sur l'intervalle [k, k+m-1], on obtient :

$$\begin{cases} y_p(k) = g\big(x_p(k)\big) \\ y_p(k+1) = g\big(f\big(x_p(k), u(k)\big)\big) \\ \dots \\ y_p(k+m\text{-}1) = g\big(f\big(\dots\big(f\big(x_p(k), u(k)\big), u(k+1)\big)\dots, u(k+m\text{-}2)\big)\big) \end{cases} \qquad (1)$$

Dans le cas d'un modèle linéaire d'ordre n, observable, on peut résoudre ce système pour m=n, et l'état $x_p(k)$ peut donc être calculé à partir de n-1 valeurs de l'entrée (u(k), …, u(k+n-2)), et de n valeurs de la sortie ($y_p(k)$, …, $y_p(k+n-1)$). Tout modèle d'état linéaire observable possède donc une représentation entrée-sortie. Dans le cas non linéaire, il est clair que le système (1) ne peut être résolu que si les fonctions f et g vérifient certaines conditions.

[LEV92] propose des conditions d'existence de représentations entrée-sortie *globales*, reposant sur la propriété d'observabilité générique[1]. Un modèle d'état non linéaire est *génériquement*

---

[1] Des conditions suffisantes concernant l'existence de représentations entrée-sortie *locales* sont données dans [LEO85a], établies à partir de la matrice de Hankel du système. Dans le domaine des réseaux de neurones, un autre

observable s'il existe un entier p tel que presque toute séquence d'entrées et de sorties de longueur supérieure ou égale à p détermine l'état de manière unique à partir de (1)[2]. En s'appuyant sur les résultats d'Ayeles [AYE81], il commence par montrer que presque tout système d'état est génériquement observable avec p=2n+1 (n est l'ordre du système, supposé connu, ou sa limite supérieure)[3]. Pour un tel système, il établit l'existence d'un observateur global de la forme :

$$x_p(k) = \Phi_{glo}\left(\quad y_p(k+2n), \ldots, y_p(k), u(k+2n), \ldots, u(k)\right)$$

Supposons que nous sommes dans ce cas. La sortie du processus à l'instant k+2n+1 s'écrit :

$$y_p(k+2n+1) = g\left(f\left(\ldots\left(f\left(x_p(k), u(k)\right), u(k+1)\right)\ldots, u(k+2n)\right)\right)$$

En remplaçant $x_p(k)$ par son expression dans $y_p(k+2n+1)$, on déduit de l'existence de l'observateur celle d'une représentation entrée-sortie globale de la forme :

$$y_p(k+2n+1) = \Psi_{glo}\left(\quad y_p(k+2n), \ldots, y_p(k), u(k+2n), \ldots, u(k)\right)$$

ou encore, en réarrangeant les indices :

$$y_p(k) = \Psi_{glo}\left(\quad y_p(k-1), \ldots, y_p(k-2n-1), u(k-1), \ldots, u(k-2n-1)\right)$$

et donc celle d'un réseau de neurones approchant Ψ_{glo} avec la précision désirée.

Ces résultats nous autorisent donc à considérer que les conditions nécessaires à l'existence d'une représentation entrée-sortie globale pour un modèle-hypothèse d'état sont souvent remplies, et donc à remplacer l'hypothèse d'un modèle d'état par celle d'un modèle entrée-sortie de la forme :

$$\boldsymbol{y_p(k)} = \boldsymbol{h}\left(\boldsymbol{y_p(k-1)}, \ldots, \boldsymbol{y_p(k-p)}, \boldsymbol{u(k-1)}, \ldots, \boldsymbol{u(k-p)}\right)$$

où p est un entier compris entre n_x, l'ordre du modèle-hypothèse d'état, et $2n_x+1$.

Le système d'apprentissage doit utiliser un réseau de neurones dont la structure est celle des prédicteurs associés, soit :

- un réseau prédicteur non bouclé :

$$y(k+1) = \psi_{RN}\left(\quad y_p(k), \ldots, y_p(k-p+1), u(k), \ldots, u(k-p+1); C\right)$$

- un réseau prédicteur bouclé entrée-sortie :

$$y(k+1) = \varphi_{RN}\left(\quad y(k), \ldots, y(k-p+1), u(k), \ldots, u(k-p+1); C\right)$$

où p est un entier que l'on fera varier entre n_x, l'ordre du modèle-hypothèse d'état, et $2n_x+1$.

L'inconvénient de la modélisation entrée-sortie par rapport à la modélisation d'état est de nécessiter un prédicteur possédant un plus grand nombre d'arguments, et donc de coefficients

---

résultat est donné dans [LEV92] ; il repose sur les propriétés d'observabilité du modèle linéarisé autour d'un point d'équilibre.

[2] Presque toute séquence : au sens où l'ensemble des séquences de longueur supérieure ou égale à p qui permettent de reconstituer l'état est dense dans l'ensemble des séquences de longueur supérieure ou égale à p.

[3] Presque tout système : au sens où , si g est une fonction de Morse C^∞ (c'est-à-dire ne possède que des points critiques isolés), l'ensemble des fonctions f C^∞ pour lesquelles le système :

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

possède la propriété d'observabilité générique avec p=2n+1, est dense dans C^∞.

ajustables, ce qui pose un problème si les séquences d'apprentissage sont peu riches. Nous montrons sur un tel exemple, réel (modélisation d'un bras de robot, chapitre 4 §III), que la modélisation d'état peut dans ce cas s'avérer bien plus efficace que la modélisation entrée-sortie, même en augmentant l'ordre du modèle entrée-sortie.

II. MODÈLES-HYPOTHÈSE AVEC BRUIT.

L'expérience montre que des modèles déterministes comme ceux que nous venons de présenter sont insuffisants pour décrire la plupart des processus réels, parce que l'on ne mesure jamais tous les signaux qui les influencent, et en raison de l'incertitude qui affecte toute mesure. Nous présentons dans ce paragraphe des modèles qui rendent mieux compte de l'incertitude qui affecte la connaissance de tout processus physique.

II.1. PERTURBATIONS NON MESURÉES.

On distingue deux classes de perturbations non mesurées : les perturbations déterministes et les perturbations de type bruit.

Perturbation déterministe.

Une perturbation déterministe peut être modélisée par une entrée (échelon, rampe, sinus,…) qui survient ou se modifie à des instants aléatoires. Par exemple, l'encrassement d'un appareil, ou le changement de la masse de carburant d'un véhicule, peuvent être modélisés par une entrée de type rampe. Dans ce cas, on peut concevoir un système d'identification adaptatif, qui sera l'un des éléments d'un système de commande adaptatif.

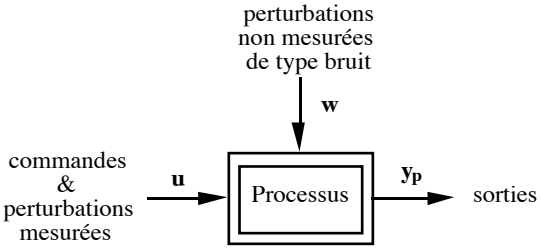


Figure 6.
Modélisation d'un processus avec perturbations non mesurées de type bruit.

Perturbation de type bruit.

Nous appelons perturbation de type bruit toute entrée modélisée par une séquence de variables aléatoires. Par exemple, toute mesure est entachée d'incertitude. On peut ne pas prendre du tout en considération ce caractère incertain, et utiliser des modèles déterministes de variables déterministes, comme nous l'avons fait dans les paragraphes précédents. On peut, à l'autre extrême, considérer

toute variable d'entrée, d'état ou de sortie d'un modèle comme une réalisation de processus aléatoire. Nous nous intéressons à une catégorie de modèles intermédiaire, des modèles déterministes dont certaines entrées sont des séquences de variables aléatoires.

Dans le cadre de ce mémoire, nous ne considérons pour la modélisation que des perturbations non mesurées de type bruit. Les perturbations non mesurées déterministes doivent être compensées par l'organe de commande (voir chapitre 5 §II).

II.2. PRÉDICTEURS ASSOCIÉS AUX MODÈLES-HYPOTHÈSE AVEC BRUIT.

Définition d'un prédicteur associé à un modèle hypothèse avec bruit.

Le prédicteur associé fournit l'espérance mathématique de la sortie du processus à l'instant k+1 conditionnée par les mesures jusqu'à l'instant k. Ce prédicteur théorique minimise la variance de l'erreur de prédiction. Le prédicteur réel (un réseau de neurones), dont les paramètres sont estimés en minimisant l'erreur de prédiction quadratique moyenne, calcule une estimation de cette espérance mathématique conditionnelle. Nous dirons encore que le prédicteur associé est optimal, au sens de la variance de l'erreur de prédiction.

Exemple : prédicteur associé au modèle-hypothèse linéaire ARMAX.

Soit le modèle-hypothèse ARMAX (Auto-Régressif à Moyenne Ajustée avec entrée eXogène) :

$$A(q)\, y_p(k) = B(q)\, u(k) + C(q)\, w(k) = q^{-d}\, B'(q)\, u(k) + C(q)\, w(k)$$

où A est un polynôme monique[4] de degré n, B un polynôme de degré $m = d+m'$, et C un polynôme monique de degré p ; {w(k)} est une séquence de variables aléatoires indépendantes (bruit blanc) à valeur moyenne nulle. Goodwin [GOO84] montre que, si les zéros de C sont à l'intérieur du cercle unité, la prédiction optimale à d pas y(k+d) satisfait :

$$C(q)\, y(k+d) = G(q)\, y_p(k) + F(q)\, B'(q)\, u(k)$$

où :

$$y_p(k+d) - y(k+d) = F(q)\, w(k+d)$$

et où F (monique) et G sont les uniques polynômes de degrés respectifs d-1 et n-1 tels que :

$$C(q) = F(q)\, A(q) + q^{-d}\, G(q)$$

La prédiction à un pas, donc avec d=1, conduit à F(q) = 1, et donc à :

$$y_p(k+1) - y(k+1) = w(k+1).$$

N. B. Cette forme prédicteur du modèle ARMAX n'est qu'une écriture équivalente à la forme prédicteur du filtre de Kalman [GOO84].

---

[4] Un polynôme en q^{-1} ***monique*** est de la forme : $\mathbf{1} + a_1 q^{-1} + a_2 q^{-2} \ldots$

Cet exemple met en lumière deux faits importants :

a) La variance de l'erreur de prédiction obtenue avec le prédicteur associé est bien minimale, puisque c'est celle du bruit blanc (qui est non prédictible par définition).

b) Faisons apparaître la parenté du *prédicteur à 1 pas* avec le modèle-hypothèse :

On a $F(q) = 1$ et $G(q) = q\left(C(q) - A(q)\right)$, soit : $C(q)\, y(k+1) = q\,(C(q) - A(q))\, y_p(k) + B'(q)\, u(k)$.

En réarrangeant les termes, il vient :

$y(k+1) = (1 - A(q))\, y_p(k+1) + B'(q)\, u(k) + (C(q) - 1)\, e(k+1)$

Quel que soit le retard d du processus, le calcul des paramètres du prédicteur à 1 pas associé au modèle-hypothèse fournit donc directement les coefficients des polynômes A, B et C, c'est-à-dire les paramètres du modèle-hypothèse lui-même, puisque ce dernier s'écrit :

$$y_p(k+1) = \left(1 - A(q)\right)\, y_p(k+1) + B'(q)\, u(k) + \left(C(q) - 1\right)\, w(k+1) + w(k+1)$$

De même, pour la modélisation d'un processus non linéaire, si le prédicteur neuronal du système d'apprentissage a la structure du prédicteur à 1 pas associé au modèle-hypothèse, si ses coefficients sont modifiés de manière à minimiser une estimation de la variance de l'erreur de prédiction, si l'hypothèse est vraie, si le prédicteur neuronal est de taille suffisante, si les séquences d'apprentissage sont bien choisies (i.e. si l'estimation de la variance est bonne), et enfin si l'algorithme est performant, alors le réseau prédicteur donnera une bonne approximation de la (des) fonction(s) du modèle-hypothèse. *L'utilisation de réseaux de neurones permet donc d'étendre à une large classe de modèles non linéaires la forme prédicteur du filtre de Kalman.*

II.2.1 Modélisation entrée-sortie avec bruit.

Soit {w(k)} une séquence de variables aléatoires indépendantes (bruit blanc) à valeur moyenne nulle, intervenant dans le modèle-hypothèse d'une manière que nous allons préciser. Le modèle-hypothèse entrée-sortie avec bruit le plus général que nous considérons est le modèle NARMAX, (Non linéaire Auto-Régressif à Moyenne Ajustée avec entrée eXogène), extension non linéaire du modèle ARMAX. Cependant, des cas particuliers de ce modèle sont extrêmement utiles, et les prédicteurs associés faciles à manipuler. L'étude théorique des différents modèles-hypothèse et des prédicteurs associés a déjà été effectuée pour les réseaux de neurones dans [NER92a&b]. Cependant, il n'existe pas à notre connaissance de mise en œuvre de prédicteurs associés à des modèles généraux NARMAX, mise en œuvre beaucoup plus délicate que celle des cas particuliers. Des exemples de processus NARMAX et de l'apprentissage des prédicteurs associés sont donnés dans [LEO87], mais pour des modèles (et des processus !) polynômiaux. [BIL92] présente la modélisation neuronale d'un processus NARMAX dans le cas particulier d'un bruit MA (voir §II.2.1.3), mais n'utilise pas le prédicteur associé. Pour combler cette lacune, nous spécifions ici le prédicteur qui s'impose pour la modélisation NARMAX, et en donnons des exemples au chapitre 4.

Les différents modèles-hypothèse avec bruit que nous présentons sont classés *selon le type de prédicteur neuronal* qu'il faut mettre en œuvre pour réaliser le prédicteur associé, et *dans l'ordre de la complexité croissante de ce prédicteur.*

II.2.1.1. Modélisation avec bruit d'état additif NARX.

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

$$\boldsymbol{y_p(k)} = \boldsymbol{h}\left(\boldsymbol{y_p(k\text{-}1)}, \ldots, \boldsymbol{y_p(k\text{-}n)}, \boldsymbol{u(k\text{-}1)}, \ldots, \boldsymbol{u(k\text{-}m)}\right) + \boldsymbol{w(k)}$$

Le prédicteur associé est non bouclé :

$$\boldsymbol{y(k+1)} = \boldsymbol{h}\left(\boldsymbol{y_p(k)}, \ldots, \boldsymbol{y_p(k\text{-}n+1)}, \boldsymbol{u(k)}, \ldots, \boldsymbol{u(k\text{-}m+1)}\right)$$

Ce prédicteur est effectivement celui qui minimise la variance de l'erreur de prédiction puisque :
$y_p(k+1) - y(k+1) = w(k+1)$

Le système d'apprentissage doit donc utiliser un réseau de neurones prédicteur ayant la structure du prédicteur associé, soit un réseau non bouclé de la forme :

$$y(k+1) = \psi_{RN}\left(y_p(k), \ldots, y_p(k\text{-}n+1), u(k), \ldots, u(k\text{-}m+1); C\right)$$

Si l'hypothèse est vraie et si les autres conditions d'une bonne identification sont réunies (séquences d'apprentissage représentatives, algorithme adéquat et performant), alors la fonction réalisée par le réseau de neurones sera une bonne approximation de h.

II.2.1.2. Modélisation avec bruit de sortie additif NBSX.

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

$$\begin{cases} \boldsymbol{x_p(k)} = \boldsymbol{h}\left(\boldsymbol{x_p(k\text{-}1)}, \ldots, \boldsymbol{x_p(k\text{-}n)}, \boldsymbol{u(k\text{-}1)}, \ldots, \boldsymbol{u(k\text{-}m)}\right) \\ \boldsymbol{y_p(k)} = \boldsymbol{x_p(k)} + \boldsymbol{w(k)} \end{cases}$$

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

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

Ce prédicteur est optimal puisque si l'on suppose les erreurs de prédictions précédentes égales au bruit, on a :

$$\begin{cases} y_p(k) - y(k) = w(k) \\ \ldots \\ y_p(k\text{-}n+1) - y(k\text{-}n+1) = w(k\text{-}n+1) \end{cases} \quad avec \quad \begin{cases} y_p(k) = x_p(k) + w(k) \\ \ldots \\ y_p(k\text{-}n+1) = x_p(k\text{-}n+1) + w(k\text{-}n+1) \end{cases} \quad \Rightarrow \begin{cases} y(k) = x_p(k) \\ \ldots \\ y(k\text{-}n+1) = x_p(k\text{-}n+1) \end{cases}$$

alors :

$$\begin{aligned} y(k+1) &= h\left(y(k), \ldots, y(k\text{-}n+1), u(k), \ldots, u(k\text{-}m+1) \right) \\ &= h\left(x_p(k), \ldots, x_p(k\text{-}n+1), u(k), \ldots, u(k\text{-}m+1) \right) \\ &= x_p(k+1) \end{aligned}$$

et l'on a bien :
$y_p(k+1) - y(k+1) = y_p(k+1) - x_p(k+1) = w(k+1)$

Le système d'apprentissage doit ici utiliser un réseau de neurones prédicteur bouclé entrée-sortie d'ordre n :

$$y(k+1) = \varphi_{RN}\left(y(k), \ldots, y(k\text{-}n+1), u(k), \ldots, u(k\text{-}m+1); C\right)$$

Ici encore, si l'hypothèse est vraie, et si toutes les conditions d'une bonne identification sont réunies, alors la fonction réalisée par la partie non bouclée du réseau de neurones sera une bonne approximation de h.

II.2.1.3. Modélisation NARMAX.

Dans le domaine du Traitement du Signal et de l'Automatique, les modèles précédents sont souvent insuffisants pour décrire correctement les processus. Le modèle-hypothèse non linéaire le plus général que nous considérons est le modèle NARMAX (Non linéaire Auto-Régressif à Moyenne Ajustée avec entrée eXogène). Ce modèle est une extension au non linéaire du modèle linéaire ARMAX [LEO85b] [CHE89] [CHE90a&b]. Il est de la forme :

$$\boldsymbol{y_p(k)} = \boldsymbol{h}\left(\boldsymbol{y_p(k\text{-}1)}, \ldots, \boldsymbol{y_p(k\text{-}n)}, \boldsymbol{u(k\text{-}1)}, \ldots, \boldsymbol{u(k\text{-}m)}, \boldsymbol{w(k\text{-}1)}, \ldots, \boldsymbol{w(k\text{-}p)} \right) + \boldsymbol{w(k)}$$

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

$$\boldsymbol{y(k+1)} = \boldsymbol{h}\left(\boldsymbol{y_p(k)}, \ldots, \boldsymbol{y_p(k\text{-}n+1)}, \boldsymbol{u(k)}, \ldots, \boldsymbol{u(k\text{-}m+1)}, \boldsymbol{e(k)}, \ldots, \boldsymbol{e(k\text{-}p+1)} \right)$$

où $e(k) = y_p(k) - y(k)$

Ce prédicteur est optimal puisque si l'on suppose les erreurs de prédiction passées égales au bruit :

$e(k) = w(k), \ldots, e(k\text{-}p+1) = w(k\text{-}p+1)$

alors l'erreur de prédiction à k+1 est elle aussi égale au bruit :

$e(k+1) = y_p(k+1) - y(k+1) = w(k+1)$

Le système d'apprentissage doit donc utiliser un réseau de neurones prédicteur bouclé d'ordre p donc les variables d'état sont les p erreurs de prédiction passées (réseau bouclé entrée-erreur) :

$$y(k+1) = \varphi_{RN}\left(y_p(k), \ldots, y_p(k\text{-}n+1), u(k), \ldots, u(k\text{-}m+1), e(k), \ldots, e(k\text{-}p+1); C \right)$$

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

Un cas particulier de l'hypothèse NARMAX est le cas d'un bruit corrélé additif ARMA (Auto-Régressif à moyenne Ajustée) :

$$\boldsymbol{A(q)\ y_p(k)} = \boldsymbol{h}\left(\boldsymbol{y_p(k\text{-}1)}, \ldots, \boldsymbol{y_p(k\text{-}n)}, \boldsymbol{u(k\text{-}1)}, \ldots, \boldsymbol{u(k\text{-}m)} \right) + \boldsymbol{C(q)\ w(k)}$$

où A et C sont deux polynômes moniques de degrés respectifs n et p. Le prédicteur associé est :

$$\boldsymbol{y(k+1)} = \boldsymbol{h}\left(\boldsymbol{y_p(k)}, \ldots, \boldsymbol{y_p(k\text{-}n+1)}, \boldsymbol{u(k)}, \ldots, \boldsymbol{u(k\text{-}m+1)}\right) + \left(1 - \boldsymbol{A(q)}\right)\boldsymbol{y_p(k+1)} + \left(\boldsymbol{C(q)} - 1\right)\boldsymbol{e(k+1)}$$

Le réseau prédicteur utilisé pour l'apprentissage sera simplifié en imposant une dépendance linéaire de sa sortie par rapport à l'erreur de prédiction. Il faut utiliser un réseau composé de deux sous-réseaux, ou de deux réseaux distincts :

$$y(k+1) = \psi^{nl}_{RN}\left(y_p(k), \ldots, y_p(k\text{-}n+1), u(k), \ldots, u(k\text{-}m+1); C_{nl} \right) + \varphi^{lin}_{RN}\left(e(k+1); C_{lin} \right)$$

où le réseau bouclé φ^{lin}_{RN} est composé d'un seul neurone linéaire, destiné à estimer les coefficients de C(q)–1. Le réseau non bouclé ψ^{nl}_{RN} doit réaliser la somme de la fonction h et de (1–A(q)) y_p(k+1) (il est donc conseillé d'utiliser un réseau complètement connecté). Le même type de réseau sera utilisé dans le cas de la modélisation d'un bruit coloré MA (A(q)=1), le prédicteur associé étant :

$$y(k+1) = h\left(y_p(k), \ldots, y_p(k\text{-}n+1), u(k), \ldots, u(k\text{-}m+1) \right) + \left(C(q) - 1\right) e(k+1)$$

Le prédicteur associé au modèle-hypothèse non linéaire affecté d'un bruit additif ARMA est d'ailleurs donné dans [GOO84] sous une forme équivalente[5].

II.2.2. Modélisation d'état avec bruit.

II.2.2.1. L'état est mesuré.

Si l'état est mesuré, quelle que soit la nature du bruit, il est possible de se ramener à deux modèles-hypothèse entrée-sortie, comme dans le cas déterministe. Le système d'apprentissage doit alors utiliser un réseau de neurones ayant la structure du prédicteur associé à l'équation d'état, et un réseau statique destiné à modéliser l'équation de sortie. Les séquences d'apprentissage sont constituées de la séquence de commande, de la séquence des variables d'état mesurées et de la séquence des sorties mesurées sur le processus.

II.2.2.2. L'état n'est pas mesuré.

II.2.2.2.1. Modélisation d'état NBSX.

Ce modèle-hypothèse se prête bien à la description d'un modèle d'état avec bruit de mesure :

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

Un prédicteur optimal associé est :

$$\begin{cases} \boldsymbol{x}(\boldsymbol{k}+1) = \boldsymbol{f}\left(\boldsymbol{x}(\boldsymbol{k}), \boldsymbol{u}(\boldsymbol{k})\right) \\ \boldsymbol{y}(\boldsymbol{k}+1) = \boldsymbol{g}\left(\boldsymbol{x}(\boldsymbol{k}+1)\right) \end{cases}$$

En effet, si l'on suppose que $x(k) = x_p(k)$, on a $x(k+1) = x_p(k+1)$, et :

$e(k+1) = y_p(k+1) - y(k+1) = w(k+1)$

Comme dans le cas déterministe, tous les prédicteurs dont l'état est une transformation de x_p vérifiant les conditions du §I.2.2.2 sont également optimaux.

Le système d'apprentissage doit donc utiliser un réseau de neurones prédicteur bouclé sur l'état non imposé (réseau d'état) de la forme :

$$\begin{cases} x(k+1) = \varphi_{RN}\left(x(k), u(k); C\right) \\ y(k+1) = \omega_{RN}\left(x(k+1); C\right) \end{cases}$$

ou bien encore :

$$\begin{cases} x(k+1) = \varphi_{RN}\left(x(k), u(k); C\right) \\ y(k+1) = \psi_{RN}\left(x(k), u(k); C\right) \end{cases}$$

---

[5] L'auteur ne fait jamais apparaître l'erreur de prédiction dans l'expression du prédicteur associé ; le prédicteur non linéaire avec bruit ARMA est ainsi donné sous la forme :

$$y(k+1) = h\left(y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1)\right) + \left(A(q) - C(q)\right) y_p(k+1) + \left(C(q) - 1\right) y(k+1)$$

II.2.2.2.2. Modélisation d'état avec autre bruit.

Si le bruit n'est pas purement un bruit de sortie, le prédicteur associé au modèle-hypothèse a nécessairement pour arguments les variables d'état du processus. Ces variables n'étant pas mesurées, le système d'apprentissage, qui ne peut qu'utiliser un réseau de neurones prédicteur bouclé sur l'état comme au paragraphe précédent, conduira à un prédicteur sous-optimal, et donc à un moins bon modèle. La meilleure solution est dans ce cas de faire l'hypothèse NARX ou NARMAX (voir §II.2.2.3 pour la validité de cette hypothèse), et d'effectuer l'identification avec un prédicteur neuronal ayant la structure du prédicteur associé.

II.2.2.3. Existence d'un modèle entrée-sortie équivalent.

Des conditions de l'existence d'une représentation entrée-sortie *locale* de type NARMAX pour des modèles d'états discrets affectés de perturbations aléatoires sont données par Leontaritis et Billings dans [LEO85b] et sont les seules à notre connaissance pour des modèles non linéaires généraux. Il reste donc possible de supposer ces conditions réunies dans le domaine considéré de l'espace d'état, et de faire l'hypothèse NARMAX. Le système d'apprentissage devra donc utiliser le prédicteur associé à l'hypothèse NARMAX.

CONCLUSION.

Dans ce chapitre, nous avons déterminé la structure du prédicteur associé à un modèle-hypothèse donné qui doit être utilisé par le système d'apprentissage, ceci aussi bien pour des modèles entrée-sortie que pour des modèles d'état, que l'état soit mesuré ou non. Nous avons aussi donné brièvement les conditions du succès de l'apprentissage du prédicteur.

Le chapitre 3 suivant précise ces conditions en examinant la mise en œuvre du système d'apprentissage complet qui utilise : un prédicteur associé au modèle-hypothèse, et dont les coefficients sont à estimer, des séquences et un algorithme d'apprentissage. Nous abordons également le problème de la sélection d'un prédicteur parmi les prédicteurs associés à différentes hypothèses. Nous définissons enfin les utilisations possibles du prédicteur retenu (prédiction, simulation, élaboration d'un correcteur pour le processus).

Chapitre 3

ESTIMATION DES PARAMÈTRES D'UN MODÈLE

INTRODUCTION.

Le chapitre précédent fournit le prédicteur théorique associé à un modèle-hypothèse donné ; ceci permet, pour chaque modèle envisagé, de définir un ensemble de réseaux de neurones candidats pour l'estimation du prédicteur théorique. En effet, celui-ci définit la structure bouclée ou non, et les arguments de la ou des fonctions que doit réaliser un prédicteur neuronal candidat. Or sa définition complète comporte celle de son architecture : nombre et type de neurones, connectivité, et valeurs des coefficients. Pour un type de réseau donné, chaque candidat est caractérisé par le nombre de ses neurones. L'estimation des coefficients de chacun, ou identification, est obtenue par apprentissage. Nous traitons dans ce chapitre du choix de séquences d'apprentissage appropriées et de l'algorithme d'apprentissage adéquat. Le meilleur candidat de l'hypothèse considérée est sélectionné à l'issue d'une procédure itérative. Cette procédure, appliquée à toutes les hypothèses, conduit à la sélection du meilleur réseau de neurones. Pour terminer, nous précisons les modalités d'utilisation du prédicteur obtenu pour l'application à laquelle il est destiné (prédiction, simulation, commande).

I. SÉQUENCES D'APPRENTISSAGE.

Ce paragraphe donne quelques indications sur le choix des séquences d'apprentissage.

I.1. SÉQUENCE DES ENTRÉES DE COMMANDE.

* *Contraintes sur les entrées de commande*.

Elles portent sur l'amplitude et le type de signaux de commande que le processus est susceptible de recevoir pendant son fonctionnement. Les amplitudes maximales sont en général faciles à déterminer, car leur ordre de grandeur correspond aux valeurs de saturation des actionneurs, qui peuvent être estimées physiquement (puissance maximale que peut délivrer un moteur, pression maximale d'un circuit de freinage ou d'un mécanisme hydraulique…, cf. chapitre 7).

En ce qui concerne le type de signaux à utiliser, un principe général est que les signaux utilisés pour l'identification doivent être de même nature que ceux qui seront calculés par l'organe de commande pendant l'utilisation du processus. L'idéal serait d'utiliser pour l'identification le correcteur même qui sera synthétisé à l'aide du prédicteur identifié ! *Si le bruit est négligeable*, une bonne démarche consiste à effectuer les expériences *en asservissement*, avec un correcteur simple (par

exemple linéaire). Cette démarche, ou identification *en boucle fermée*, permet d'explorer le domaine de fonctionnement désiré, en imposant une séquence de consigne correspondant au cahier des charges. C'est ainsi que nous avons procédé pour la modélisation du véhicule REMI (cf. chapitre 7). Une autre démarche, couramment utilisée, consiste à explorer " au mieux " le domaine de fonctionnement, par exemple avec des créneaux de commande (riches en fréquences), d'amplitudes et de durées diverses. Notons que cette dernière solution n'est pas praticable s'il existe des contraintes sur des sorties ou des variables internes du processus, et en particulier si le processus est instable.

* *Fréquence d'échantillonnage*.

Les réseaux de neurones étant des modèles non linéaires, il n'est pas possible de passer simplement d'un modèle discret, valable à une fréquence d'échantillonnage donnée, à un autre : il est donc nécessaire d'effectuer l'identification à la fréquence qui sera utilisée pour la commande du processus. Si des contraintes diverses, par exemple le temps de calcul de la commande, nécessitent le choix d'une fréquence d'asservissement plus basse que la fréquence utilisée pour l'identification, il faut procéder à une nouvelle identification.

I.2. SÉQUENCES D'APPRENTISSAGE ET ESTIMATION DE LA PERFORMANCE.

Comme nous l'avons précisé au chapitre 1, l'apprentissage consiste à ajuster la (les) fonction(s) du prédicteur à un ensemble de points définis par des séquences d'apprentissage, ceci en minimisant une fonction de coût. Il y a surajustement (overfitting) lorsque l'apprentissage conduit à annuler quasiment la fonction de coût sans que pour autant la(les) fonction(s) réalisée(s) par la partie non bouclée du réseau prédicteur approche(nt) celle(s) du modèle-hypothèse auquel le prédicteur est associé. Pour des séquences d'apprentissage données, le surajustement se produit si le réseau prédicteur possède trop de coefficients, c'est-à-dire définit une famille de fonctions trop riche.

Pour sélectionner un réseau candidat pour une hypothèse donnée, il est donc nécessaire de répartir les données disponibles en une séquence d'apprentissage et une séquence d'estimation de la performance, dite séquence de test. Une séquence de test, de même type (issue de la même population) que la séquence d'apprentissage, conduit à une meilleure estimation de la variance de l'erreur de prédiction (erreur quadratique moyenne de test, notée EQMT) que celle obtenue avec la séquence d'apprentissage (notée EQMA). L'évolution de EQMA et de EQMT lorsque l'on augmente le nombre de neurones cachés, permet de détecter un surajustement, et de sélectionner le meilleur réseau de neurones parmi les candidats : dès qu'ajouter un neurone supplémentaire au réseau fait augmenter l'EQMT, même si l'EQMA continue à décroître, le nombre optimal de neurones est atteint. Cette méthode, à condition d'utiliser des méthodes d'optimisation performantes (par exemple les méthodes quasi newtonniennes proposées en annexe I §II.3), permet de sélectionner le réseau le plus parcimonieux.

Remarque importante.

Dans la littérature ([SJÖ94] par exemple), la séquence de test est souvent utilisée pour arrêter l'apprentissage d'un réseau donné : dès qu'une itération d'apprentissage augmente l'EQMT, celui-ci est arrêté. Dans notre travail, *la séquence de test sert exclusivement à estimer la performance des réseaux après apprentissage, et n'est en aucun cas utilisée pour interrompre l'apprentissage*. L'apprentissage n'est arrêté que lorsqu'un minimum est atteint : l'arrêt est décidé en fonction de la valeur de la fonction de coût et de celle de la norme de son gradient. Si le nombre de neurones est augmenté de façon incrémentale comme nous l'indiquons ci-dessus, on obtient forcément le prédicteur le plus parcimonieux réalisant la meilleure performance sans surajustement.

II. ALGORITHMES D'APPRENTISSAGE.

Si le modèle-hypothèse est juste, si le prédicteur neuronal possède bien la structure du prédicteur associé, et si le choix des séquences d'apprentissage est conforme aux principes que nous venons d'énoncer, il est encore nécessaire d'utiliser l'algorithme d'apprentissage adéquat. Celui-ci est en fait fixé par les affectations des entrées externes I et des sorties Y du prédicteur, et, s'il est bouclé, de ses entrées d'état S^{in} et de ses sorties d'état S^{out}. Les calculs de la fonction de coût, de son gradient et des modifications des coefficients, s'effectuent selon la présentation de l'annexe I. Comme au chapitre 2, nous considérons les modèles-hypothèse mono-entrée/mono-sortie suivants :

- des modèles-hypothèse entrée-sortie :

$$y_p(k) = h\left(y_p(k\text{-}1), \ldots, y_p(k\text{-}n), u(k\text{-}1), \ldots, u(k\text{-}m)\right)$$

où $y_p \in \mathbb{R}$, $u \in \mathbb{R}$, et h est une fonction non linéaire inconnue.

- des modèles-hypothèse d'état :

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

où $y_p \in \mathbb{R}$, $u \in \mathbb{R}$, $x_p \in \mathbb{R}^n$, et f et g sont des fonctions non linéaires inconnues.

La fonction de coût s'écrit :

- dans le cas d'un prédicteur de la sortie du processus :

$$J(C) = \frac{1}{N} \sum_{k=1}^{N} \left(y_p(k) - y(k)\right)^2 = \frac{1}{N} \sum_{k=1}^{N} \left(e(k)\right)^2$$

où N est le nombre total d'instants de l'ensemble d'apprentissage, constitué de plusieurs séquences correspondant à des initialisations différentes de l'état du processus. Pour simplifier la présentation des algorithmes, et sans perte de généralité, nous supposons la séquence d'apprentissage obtenue à partir d'un seul état initial.

- dans le cas d'un prédicteur de l'état (mesuré) du processus :

$$J(C) = \frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{n} w_{ii} \left(x_{ip}(k) - x_i(k)\right)^2 = \frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{n} w_{ii} \left(e_i(k)\right)^2$$

Les coefficients w_{ii} sont les coefficients de pondération de la fonction de coût, choisie ici diagonale. Pour les modèles-hypothèse et les prédicteurs associés considérés dans ce travail, le choix de la pondération n'est pas crucial.

II.1. PRÉDICTEURS NON BOUCLÉS.

Nous distinguons ici le cas de l'identification de prédicteurs associés à des modèles-hypothèse entrée-sortie, qui constitue un problème d'identification mono-sortie, du cas de l'identification de prédicteurs associés à des modèles-hypothèse d'état lorsque l'on mesure l'état du processus, qui est un problème d'identification multi-sortie (puisque les composantes de l'état sont considérées comme des " sorties " pour l'apprentissage - on dispose d'une valeur désirée pour chacune d'elles).

II.1.1. Prédicteur associé à un modèle-hypothèse entrée-sortie.

Un modèle-hypothèse NARX impose un prédicteur non bouclé (chapitre 2 §II.2.1.1). Un tel prédicteur peut, par exemple, être réalisé à l'aide du réseau complètement connecté de la figure 1.

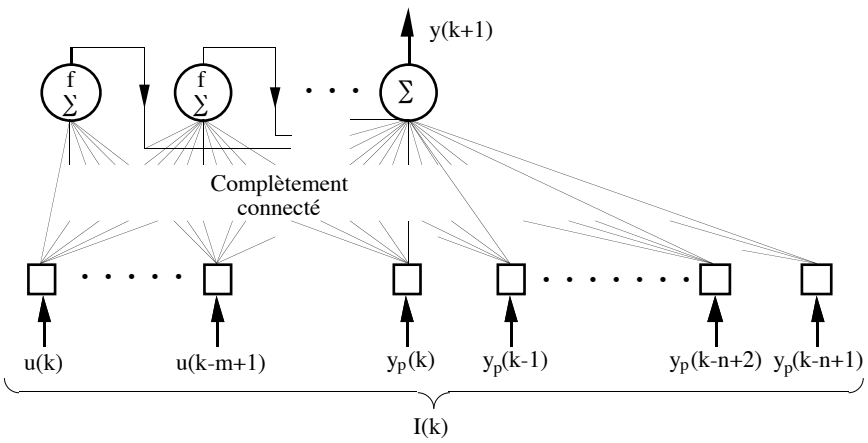


Figure 1.
Réalisation d'un prédicteur NARX à l'aide d'un réseau non bouclé complètement connecté.

Le réseau de la figure 1 définit la copie du réseau à l'instant k. Ce réseau, non bouclé, n'a pas d'état (N_S, dimension de S^{in} et S^{out}, est nul). Aux entrées externes sont affectées les composantes du vecteur I(k), de dimension m+n, qui sont les suivantes :

$I_i(k) = u(k\text{-}i\text{+}1) \quad i \in [1;m],\ k \in [1;N]$

$I_i(k) = y_p(k\text{+}m\text{-}i\text{+}1) \quad i \in [m\text{+}1;\ m\text{+}n],\ k \in [1;N]$

$Y(k) = y(k\text{+}1) \quad k \in [1;N]$

où N est la taille de la séquence d'apprentissage. On suppose les séquences d'apprentissage numérotées de telle manière que u et y_p soient définies pour les indices ci-dessus.

Le système d'apprentissage utilisant ce réseau est représenté schématiquement sur la figure 2. Le prédicteur est dit " *dirigé* " par le processus, car l'état de celui-ci est imposé à ses entrées externes à chaque instant de la fenêtre d'apprentissage. Les entrées externes sont indépendantes des coefficients

du réseau. On est donc en présence de N copies indépendantes du réseau dont chacune fournit en sortie une erreur intervenant dans la fonction de coût. Chacune de ces erreurs est l'entrée d'un réseau de rétro-propagation, et le gradient est la somme des N gradients partiels calculés par rétro-propagation. Cet algorithme est encore appelé " teacher-forcing " dans la littérature [JOR85].

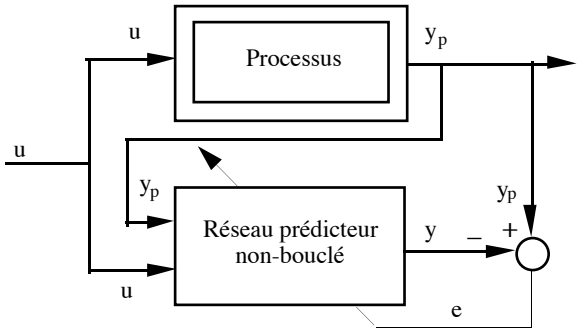
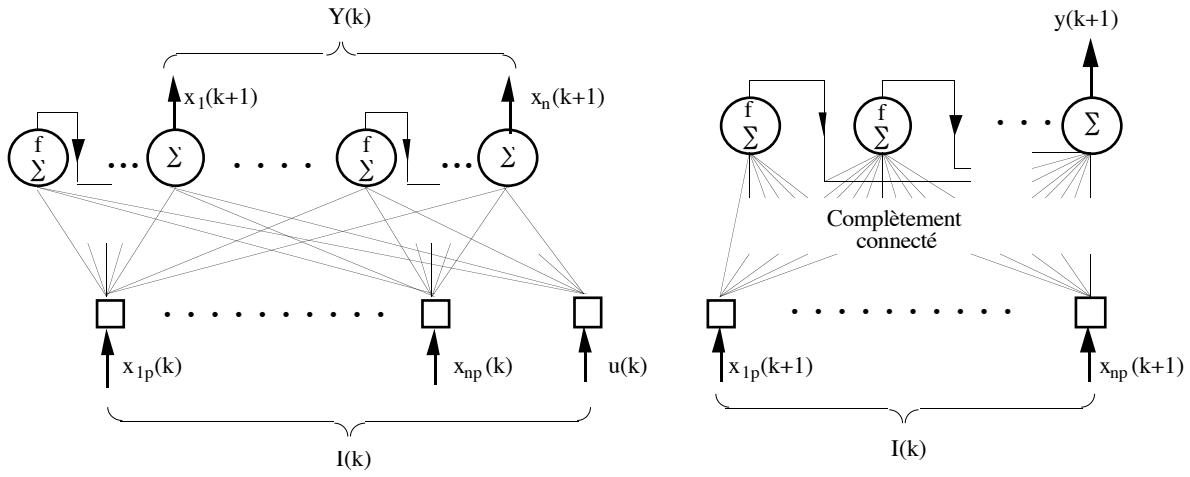


Figure 2.
Modèle-hypothèse entrée-sortie NARX :
le système d'apprentissage utilise un prédicteur non bouclé et un algorithme dirigé.

II.1.2. Prédicteur associé à un modèle-hypothèse d'état.

L'identification du prédicteur associé à un modèle-hypothèse d'état affecté d'un bruit d'état additif lorsque l'état du processus est mesuré impose aussi un prédicteur entrée-sortie non bouclé (chapitre 2 §II.2.2.1). Pour réaliser ce prédicteur, nous avons vu que l'on peut utiliser deux réseaux de neurones distincts, l'un réalisant la prédiction de l'état, et l'autre l'équation d'observation. Par exemple, le prédicteur associé au problème peut être réalisé à l'aide du réseau composé de n sous-réseaux complètement connectés, et du réseau complètement connecté de la figure 3.



Réseau non bouclé N°1 Réseau non bouclé N°2

Figure 3.
Réalisation d'un prédicteur associé à un modèle-hypothèse d'état affecté d'un bruit d'état, à l'aide d'un réseau composé de n sous-réseaux complètement connectés (réseau N°1), et d'un réseau complètement connecté (réseau N°2).

Réseau N°1 :

$I_i(k) = x_{i\,p}(k) \quad i \in [1;n],\ k \in [1;N]$

$I_{n+1}(k) = u(k) \quad k \in [1;N]$

$Y_i(k) = x_i(k+1) \quad k \in [1;N]$

Ce réseau est un prédicteur non bouclé.

Réseau N°2 :

$I_i(k) = x_{i\,p}(k+1) \quad i \in [1;n],\ k \in [1;N]$

$Y(k) = y(k+1) \quad k \in [1;N]$

Ce réseau non bouclé réalise une transformation algébrique de ses entrées.

Le système d'apprentissage utilisant ces deux réseaux est représenté sur la figure 4.

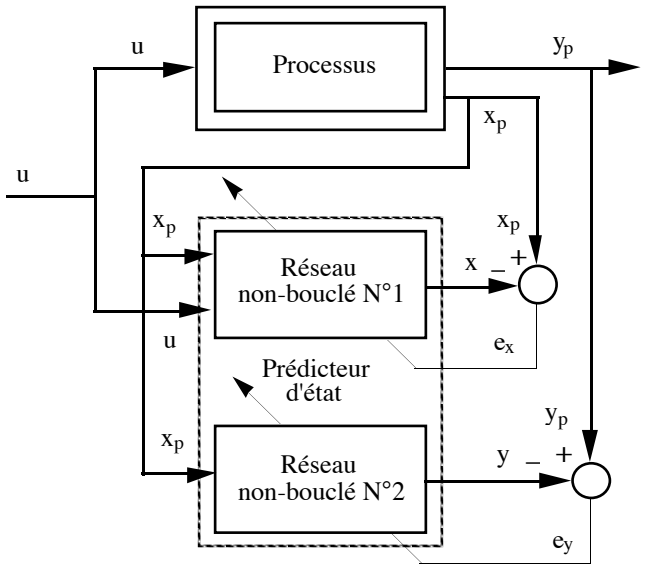


Figure 4.
Modèle-hypothèse d'état avec bruit d'état additif (ou sans perturbation) :
le système d'apprentissage utilise un prédicteur composé de deux réseaux non bouclés et un algorithme dirigé.

L'apprentissage des deux réseaux est dirigé par les variables d'état mesurées du processus. On est encore une fois en présence de N copies indépendantes de chacun des réseaux prédicteurs dont chacune fournit en sortie une erreur, e_x de dimension n et e_y de dimension 1, intervenant dans les 2 fonctions de coût. Comme précédemment, chacune de ces erreurs est l'entrée d'un réseau de rétro-propagation, et le gradient est la somme des N gradients partiels calculés par rétro-propagation. Pour éviter le problème de la pondération de la fonction de coût définie pour le réseau 1, on peut également utiliser n réseaux distincts.

II.2. PRÉDICTEURS BOUCLÉS.

Dans ce paragraphe, nous présentons les systèmes d'apprentissage pour l'identification de prédicteurs bouclés. Nous étudions séparément les prédicteurs associés à des modèles-hypothèse entrée-sortie (les composantes de l'état du réseau correspondent à des sorties successives du processus) et ceux qui sont associés à des modèles-hypothèse d'état (il n'y a pas de valeurs désirées pour leurs variables d'état).

II.2.1. Prédicteur associé à un modèle-hypothèse entrée-sortie.

a) Modèle-hypothèse bruit de sortie additif (NBSX).

L'identification d'un prédicteur NBSX impose l'utilisation d'un réseau de neurones bouclé (chapitre 2 §II.2.1.2). Un tel prédicteur peut être réalisé, par exemple, à l'aide du réseau complètement connecté de la figure 5.

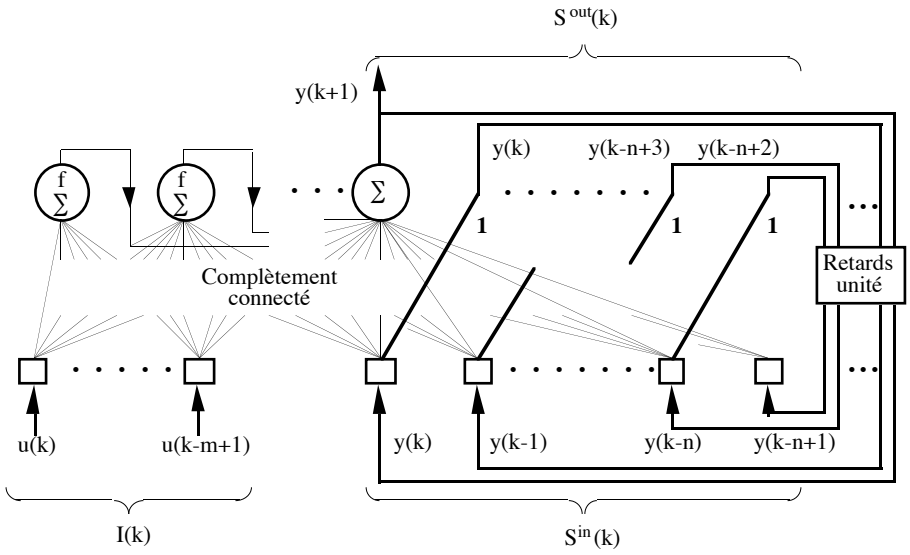


Figure 5.
Réalisation d'un prédicteur NBSX à l'aide d'un réseau bouclé complètement connecté.

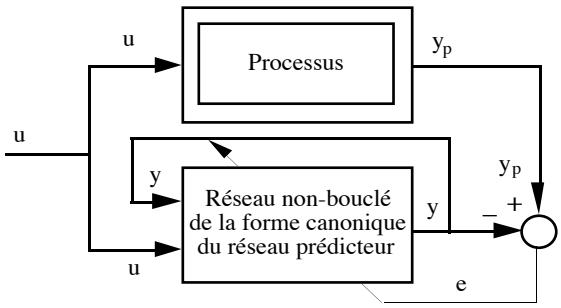


Figure 6.
Modèle-hypothèse entrée-sortie NBSX :
le système d'apprentissage utilise un prédicteur bouclé sur sa sortie et un algorithme semi-dirigé.

L'état du prédicteur correspond à celui du modèle-hypothèse et est donc constitué de n sorties successives du prédicteur. Les valeurs passées de la commande sont affectées aux entrées externes. On a :

$I_i(k) = u(k\text{-}i\text{+}1) \quad i \in [1;m],\ k \in [1;N]$

$S_i^{in}(1) = y_p(\text{-}i\text{+}1) \quad i \in [1;n] \quad (\textit{initialisation})$

$S_i^{in}(k) = S_i^{out}(k\text{-}1) \quad i \in [1;n],\ k \in [2;N]$

$S_i^{out}(k) = S_{i\text{-}1}^{in}(k) \quad i \in [2;n],\ k \in [1;N]$

$Y(k) = S_1^{out}(k) = y(k\text{+}1) \quad k \in [1;N]$

Y(k) est la seule valeur réellement calculée par le réseau : les autres variables d'état sont des sorties décalées dans le temps. Les entrées d'état de la première copie doivent être fixées par le concepteur. Le choix le plus raisonnable consiste à leur affecter les valeurs précédentes des sorties du processus.

Le système d'apprentissage utilisant ce prédicteur est représenté schématiquement sur la figure 6. Le prédicteur est dit " *semi-dirigé* " car les valeurs de ses entrées d'état ne sont imposées qu'au début de la fenêtre de la fonction de coût. Cette fois, les entrées d'état dépendent des coefficients du réseau, sauf pour la première copie. Pour le calcul du gradient, la rétro-propagation des N erreurs intervenant dans la fonction de coût doit donc être effectuée sur le réseau composé du dépliement spatial des N copies en cascade : l'algorithme de calcul de la fonction de coût et de son gradient est semi-dirigé.

b) Modèle-hypothèse NARMAX.

L'identification d'un prédicteur NARMAX impose l'utilisation d'un réseau de neurones bouclé (chapitre 2 §II.2.1.3). Un tel prédicteur peut par exemple être réalisé à l'aide du réseau complètement connecté de la figure 7.

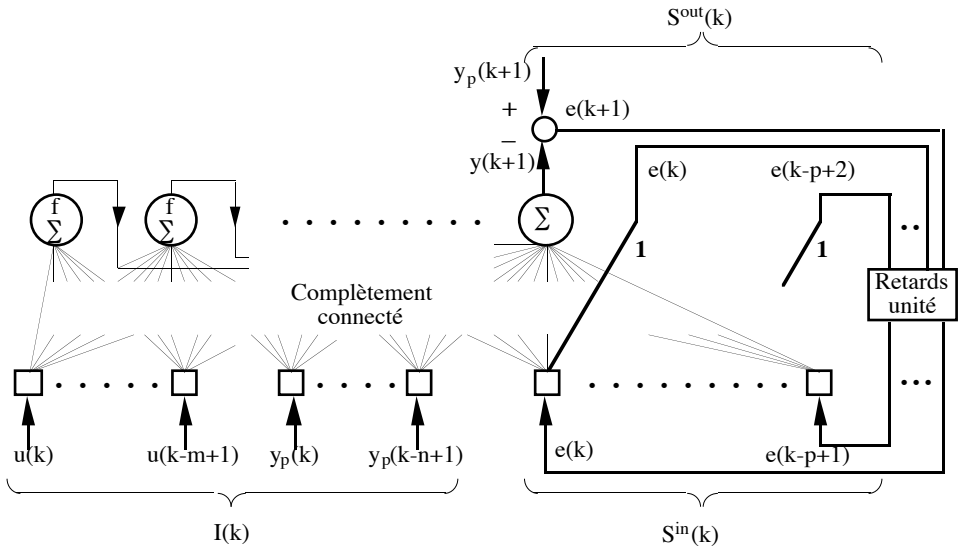


Figure 7.
Prédicteur associé à un modèle-hypothèse NARMAX réalisé à l'aide d'un réseau bouclé complètement connecté.

L'état du prédicteur n'est pas celui du modèle-hypothèse : il est constitué des p dernières erreurs du prédicteur. Les entrées externes I sont les commandes et les sorties mesurées du processus :

$I_i(k) = u(k\text{-}i\text{+}1) \quad i \in [1;m],\ k \in [1;N]$

$I_i(k) = y_p(k\text{-}i\text{-}m\text{+}1) \quad i \in [m\text{+}1;m\text{+}n],\ k \in [1;N]$

$S_i^{in}(1) = 0 \quad i \in [1;p] \quad (\textit{initialisation})$

$S_i^{in}(k) = S_i^{out}(k\text{-}1) \quad i \in [1;p],\ k \in [2;N]$

$S_i^{out}(k) = S_{i\text{-}1}^{in}(k) \quad i \in [2;p],\ k \in [1;N]$

$Y(k) = y(k\text{+}1) \quad k \in [1;N]$

$S_1^{out}(k) = y_p(k\text{+}1) - y(k\text{+}1) \quad k \in [1;N]$

Les entrées d'état de la première copie doivent ici encore être fixées par le concepteur. Ces entrées d'état représentent cette fois l'erreur de prédiction. Un choix raisonnable consiste à leur affecter la valeur zéro. Le système d'apprentissage utilisant ce prédicteur est représenté sur la figure 8.

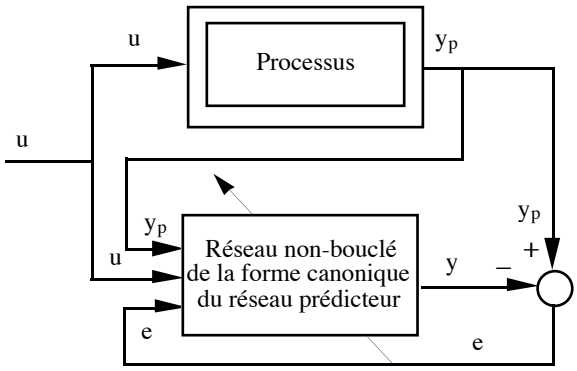


Figure 8.
Modèle-hypothèse entrée-sortie NARMAX :
le système d'apprentissage utilise un prédicteur bouclé sur l'erreur de prédiction et un algorithme semi-dirigé.

Les p entrées d'état dépendent des coefficients du réseau, sauf pour la première copie. Pour le calcul du gradient, la rétro-propagation des N erreurs intervenant dans la fonction de coût doit donc être effectuée sur le réseau composé du dépliement spatial des N copies en cascade : l'algorithme est semi-dirigé.

II.2.2. Prédicteur associé à un modèle-hypothèse d'état.

L'identification du prédicteur associé à un modèle-hypothèse d'état lorsque l'état du processus n'est pas mesuré impose l'utilisation d'un réseau de neurones bouclé (chapitre 2 §I.2.2.2 et §II.2.2.2). Ce prédicteur peut, par exemple, être réalisé à l'aide du réseau composé de n+1 sous-réseaux à une couche de neurones cachés de la figure 9.

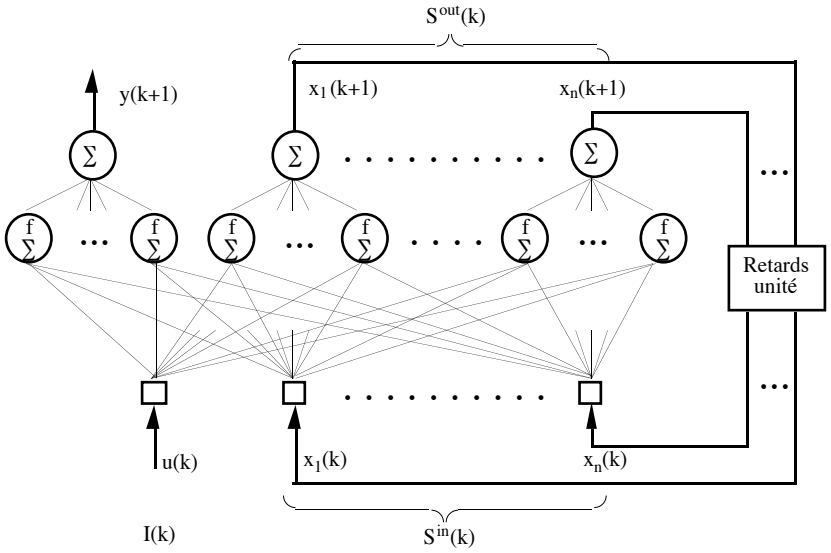


Figure 9.
Prédicteur associé à à un modèle-hypothèse d'état, lorsqu'on ne mesure pas l'état du processus,
réalisé à l'aide d'un réseau composé de n+1 sous-réseaux à une couche de neurones cachés.

Comme nous le verrons au chapitre 4 suivant, il n'est pas forcément nécessaire de fragmenter le réseau en sous-réseaux pour chaque variable d'état (surtout si les fonctions à réaliser par le réseau non bouclé de la forme canonique sont simples).

$I_1(k) = u(k) \quad k \in [1;N]$

$S_i^{in}(1) = 0 \quad i \in [1;n] \quad (initialisation)$

$S_i^{in}(k) = S_i^{out}(k\text{-}1) \quad i \in [1;n], k \in [2;N]$

$Y_1(k) = y(k+1) \quad k \in [1;N]$

Pour la première copie, des valeurs arbitraires (0 par exemple, des valeurs particulières si l'on dispose d'informations) doivent être affectées à toutes les entrées d'état du prédicteur. Des erreurs d'initialisation sont inévitables, puisque l'on ne mesure pas l'état. Le système d'apprentissage utilisant le prédicteur est représenté schématiquement sur la figure 10.

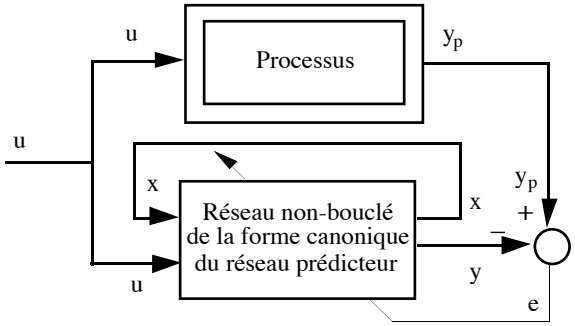


Figure 10.
Modèle-hypothèse d'état, l'état du processus n'est pas mesuré :
le système d'apprentissage utilise un prédicteur bouclé sur l'état (non imposé) et un algorithme semi-dirigé.

Le prédicteur est encore dit semi-dirigé car les valeurs de ses entrées d'état ne sont imposées qu'au début de la séquence d'apprentissage. Comme pour un prédicteur entrée-sortie bouclé, le calcul du gradient doit être effectué sur le réseau composé du dépliement spatial des N copies en cascade : l'algorithme de calcul de la fonction de coût et de son gradient est semi-dirigé.

III. SÉLECTION ET UTILISATIONS DU PRÉDICTEUR.

III.1. SÉLECTION DU PRÉDICTEUR.

Pour modéliser un processus, il est donc nécessaire de :

- formuler un ou plusieurs modèles-hypothèse compatibles avec les connaissances *a priori* que l'on a du processus (sur la nature de ses entrées, de ses sorties, éventuellement de son ordre, de son retard…) ; déterminer les prédicteurs théoriques associés à chacun de ces modèles-hypothèse ;
- pour chacune des hypothèses, mettre en œuvre le système d'apprentissage adéquat (séquences et algorithme d'apprentissage) pour les réseaux prédicteurs candidats. À partir de la performance de chaque candidat sur la séquence de test, déterminer le meilleur candidat associé à l'hypothèse.
- sélectionner la meilleure hypothèse, à partir des meilleurs candidats de chacune d'elles.

La validation finale du prédicteur est effectuée dans le cadre de son utilisation.

III.2. UTILISATIONS DU PRÉDICTEUR.

Le prédicteur obtenu, c'est-à-dire essentiellement la fonction ψ_{RN} réalisée par la partie non bouclée du réseau (ou φ_{RN}, ou les fonctions φ_{RN} et ψ_{RN} selon les cas) peut être utilisé comme modèle de simulation pour la commande, comme prédicteur, ou comme simulateur du processus.

Modèle de simulation pour la commande.

La mise au point d'un système de commande nécessite un modèle de simulation du processus (cf. chapitre 5). Dans ce travail, nous nous limitons à des modèles dont les entrées de bruit sont mises à zéro. De tels modèles sont en effet nécessaires :

- pour effectuer l'apprentissage d'un correcteur ;
- pour simuler le processus au sein d'un système de commande, en particulier dans un système de commande avec modèle interne.

Les modèles de simulation correspondant aux divers modèles-hypothèse étudiés sont obtenus de la manière suivante à partir des prédicteurs optimaux identifiés :

a) Prédicteur neuronal associé à un modèle-hypothèse NARX :

$$y(k+1) = \psi_{RN}\left(y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1);\, C\right)$$

Pour obtenir le modèle de simulation associé, il suffit de boucler le réseau :

$$y(k) = \varphi_{RN}\left(y(k-1), \ldots, y(k-n), u(k-1), \ldots, u(k-m);\, C\right)$$

b) Prédicteur neuronal associé à un modèle-hypothèse NARMAX :

$$y(k+1) = \psi_{RN}\left(y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1), e(k), \ldots, e(k-p);\, C\right)$$

Le modèle de simulation associé est le suivant :

$$y(k) = \varphi_{RN}\left(y(k-1), \ldots, y(k-n), u(k-1), \ldots, u(k-m), 0, \ldots, 0;\, C\right)$$

c) Prédicteur neuronal associé à un modèle-hypothèse NBSX :

$$y(k+1) = \varphi_{RN}\left(y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-m+1);\, C\right)$$

Il est utilisé tel quel comme modèle de simulation du processus :

$$y(k) = \varphi_{RN}\left(y(k-1), \ldots, y(k-n), u(k-1), \ldots, u(k-m);\, C\right)$$

d) De même, un prédicteur neuronal bouclé associé à un modèle d'état est utilisé tel quel comme modèle de simulation. Un prédicteur neuronal associé à un modèle d'état composé de deux réseaux non bouclés est utilisé à cette fin en bouclant le réseau des équations d'état.

Remarque importante.

Dans la pratique, il est fréquent que certaines entrées ne soit pas mesurées (température d'un moteur thermique, paramètres de la combustion), ou que des perturbations non mesurées agissent sur le processus. Ce cas ne correspond à aucune des hypothèses du chapitre 2, et corollairement, *aucun prédicteur associé* ne fournit une erreur de prédiction qui est un bruit blanc. La sélection du prédicteur

ne doit alors pas s'appuyer sur cette erreur, car elle favorise le prédicteur NARX, *non bouclé* , qui donne presque toujours une erreur de prédiction plus faible et moins corrélée que celle d'un prédicteur *bouclé* NARMAX ou NBSX (voir l'explication du chapitre 4 §I.3.1 et §I.3.2). *Il est donc fortement recommandé, dans une telle situation, de sélectionner le prédicteur en fonction de la performance du modèle de simulation (bouclé) qui lui est associé*.

Prédicteur.

Des systèmes de commande, tels que les systèmes à variance minimale, nécessitent l'utilisation, au sein même du système de commande, du prédicteur optimal [GOO84]. C'est ici une utilisation immédiate du prédicteur obtenu. Nous donnons un exemple d'une telle utilisation au chapitre 6 §II.

Simulateur.

Nous avons montré au chapitre 2 §II.2 que la fonction intervenant dans le prédicteur optimal (le réseau non bouclé de sa forme canonique) permet d'estimer le modèle-hypothèse, et donc de réaliser un simulateur du processus. Celui-ci peut être utilisé notamment pour simuler le processus en fonctionnement " normal " et détecter les anomalies en comparant l'état du simulateur à celui du processus (voir [PLO94] pour une telle utilisation du simulateur neuronal d'une colonne à distiller).

CONCLUSION.

La procédure de modélisation qui vient d'être proposée conduit à des réseaux conciliant universalité et parcimonie. En ce qui concerne l'universalité, des non-linéarités très simples telles que des saturations peuvent ainsi être très difficiles à modéliser par d'autres moyens [NOR88]. En revanche, nous verrons que deux saturations en cascade peuvent être parfaitement identifiées à l'aide d'un réseau de neurones à deux neurones cachés sigmoïdaux (chapitre 7). Du point de vue de la parcimonie, contrairement à d'autres modèles universels tels que les modèles polynômiaux, le nombre de coefficients d'un réseau de neurones à sigmoïdes n'augmente pas nécessairement de façon importante avec la complexité de la fonction et la dimension de l'espace d'entrée.

En contrepartie, la sensibilité de la sortie d'un réseau de neurones par rapport à un coefficient particulier ne s'exprime pas de manière élémentaire (par rapport à celle d'un modèle linéaire, polynômial, ou encore bilinéaire), et rend complexe l'analyse du comportement dynamique d'un réseau de neurones bouclé.

Insistons enfin sur le caractère générique des algorithmes d'apprentissage que nous avons présentés : ils ne dépendent ni de la complexité, ni du caractère bouclé ou non des prédicteurs utilisés. Alors que dans la " culture linéaire ", on cherche souvent par des techniques *ad-hoc* à se ramener à une méthode de type moindres carrés, justifiée dans sa version ordinaire pour le prédicteur ARX seulement, l'identification par réseaux de neurones non récursive (et même récursive [NER92]) telle qu'elle a été exposée aux chapitres 2 et 3 présente une grande homogénéité.

Chapitre 4

EXEMPLES DE MODÉLISATION DE PROCESSUS

INTRODUCTION.

Dans les chapitres 2 et 3, nous avons décrit les principes de la modélisation de processus par réseaux de neurones, principes qui ont été appliqués à un problème industriel : la modélisation du véhicule REMI, décrite au chapitre 7 (confidentiel) du mémoire. Afin d'illustrer aussi complètement que possible le cadre général proposé, nous avons choisi d'étudier également deux processus simulés, et un processus réel, l'actionneur hydraulique d'un bras de robot :

- Le premier processus, de type entrée-sortie, est simulé par une équation aux différences non linéaire du second ordre, et illustre la modélisation en présence de perturbations aléatoires. Ce processus est utilisé au chapitre 6 au sujet de la commande à erreur de prédiction minimale.
- Le second processus est simulé avec une représentation d'état du second ordre. Nous réalisons tout d'abord l'apprentissage de prédicteurs d'état, afin de démontrer la faisabilité de cet apprentissage, et les intérêts de ces prédicteurs. Ensuite, nous utilisons des prédicteurs entrée-sortie. L'un d'eux est mis en œuvre au chapitre 6 comme modèle de simulation pour l'apprentissage de correcteurs, et comme modèle interne.
- Enfin, nous réalisons la modélisation d'un actionneur hydraulique. La modélisation de cet actionneur ayant donné lieu à plusieurs travaux récents [SJÖ93] [SJÖ94] [BEN94], nous comparons nos méthodes et les performances obtenues à celles qui sont décrites dans ces publications.

I. MODÉLISATION D'UN PROCESSUS SIMULÉ PAR UN MODÈLE ENTRÉE-SORTIE.

I.1. PRÉSENTATION.

L'objectif est ici de montrer l'influence des hypothèses faites quant à la nature des perturbations aléatoires affectant le processus à identifier. En particulier, nous montrons que, si l'hypothèse est vraie, on obtient un prédicteur très précis. Nous montrons aussi que le modèle de simulation obtenu à partir de ce prédicteur, comme nous l'avons indiqué au chapitre 3 §III.2, est une excellente approximation de la partie déterministe du processus, et peut donc servir à l'élaboration hors-ligne d'un organe de commande pour le processus.

Processus sans bruit.

Le processus est simulé par l'équation aux différences non linéaire du second ordre suivante :

$$y_p(k) = h\left(y_p(k\text{-}1), y_p(k\text{-}2), u(k\text{-}1)\right) = 50\ tanh\left[2\ 10^{-3}\left(\frac{\left(24+y_p(k\text{-}1)\right)}{3} y_p(k\text{-}1) - 8\ \frac{u(k\text{-}1)^2}{1+u(k\text{-}1)^2}\ y_p(k\text{-}2)\right)\right] + 0{,}5\ u(k\text{-}1)$$

La dynamique de ce processus est intéressante, car son comportement est celui (i) d'un filtre passe-bas linéaire du premier ordre pour des amplitudes de 0,1 environ, et (ii) celui d'un système du second ordre oscillatoire, linéaire pour de faibles amplitudes de la commande ($0{,}1 < |u| < 0{,}5$) , et non linéaire pour les grandes amplitudes ($0{,}5 < |u| < 5$). De plus, il n'est pas symétrique par rapport à l'origine. Nous supposons connus les arguments $y_p(k\text{-}1)$, $y_p(k\text{-}2)$, $u(k\text{-}1)$ du processus sans bruit (déterminés par exemple par la méthode de sélection de modèles présentée dans [URB94] pour le cas NARX, dont l'auteur propose le processus ci-dessus).

Séquences utilisées pour l'identification.

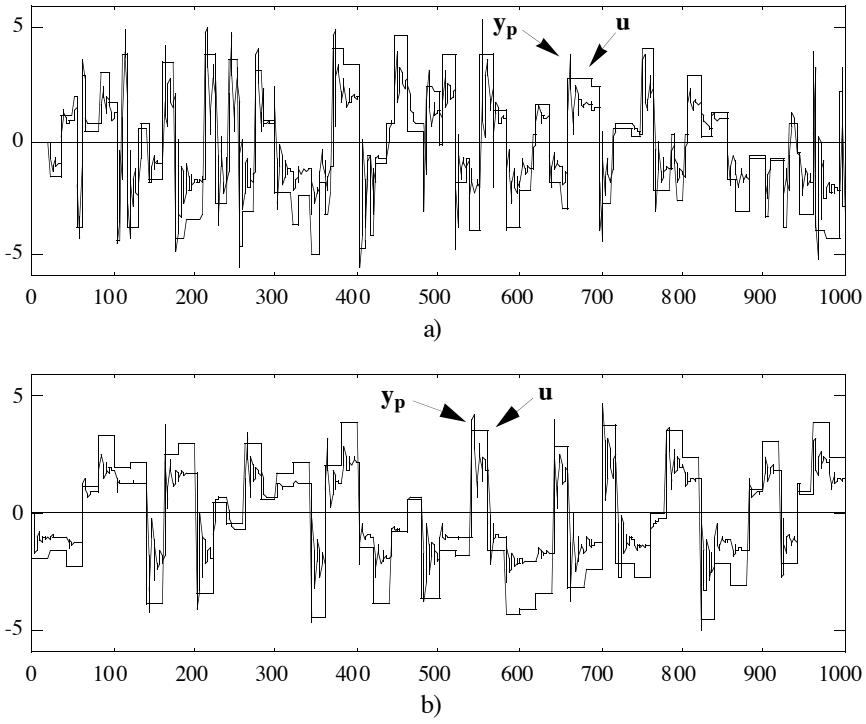


Figure 1.
a) Séquences d'apprentissage ; b) Séquences de test.

a) Séquences d'apprentissage :

La séquence de commande utilisée pour l'apprentissage des réseaux prédicteurs est constituée de créneaux d'amplitudes aléatoires entre ±5, de durées aléatoires entre 1 et 20 pas d'échantillonnage. La séquence totale comporte 1000 pas d'échantillonnage ; elle est représentée sur la figure 1a. Nous notons EQMA l'erreur quadratique moyenne sur la séquence d'apprentissage.

b) Séquences d'estimation de la performance (ou séquences de test) :

La séquence de commande pour l'estimation de la performance des prédicteurs est constituée de créneaux d'amplitudes aléatoires entre ±5, de durée constante 20 pas d'échantillonnage. La séquence

totale comporte également 1000 pas d'échantillonnage ; elle est représentée sur la figure 1b. Nous notons EQMT l'erreur quadratique moyenne sur la séquence de test.

Prédicteurs.

Les prédicteurs utilisés sont des réseaux complètement connectés, à neurones cachés dont la fonction d'activation est la tangente hyperbolique, et à neurone de sortie linéaire.

Processus avec bruit.

Soit w un bruit blanc à valeur moyenne nulle et de distribution uniforme[1] d'amplitude 0,2 (donc de variance $3{,}33\ 10^{-3}$). Nous considérons les processus avec bruit suivants :

* Processus avec perturbation additive d'état (NARX) :

$$y_p(k) = h\ \left(y_p(k\text{-}1),\ y_p(k\text{-}2), u(k\text{-}1)\right) + w(k)$$

* Processus avec perturbation additive de sortie (NBSX) :

$$\begin{cases} x_p(k) = h\left(x_p(k\text{-}1), x_p(k\text{-}2), u(k\text{-}1)\right) \\ y_p(k) = x_p(k) + w(k) \end{cases}$$

* Processus NARMAX :

$$y_p(k) = h\ \left(y_p(k\text{-}1),\ y_p(k\text{-}2), u(k\text{-}1)\right) + 0{,}5\ y_p(k\text{-}1)\ w(k\text{-}1) + w(k)$$

I.2. MODÉLISATION DU PROCESSUS SANS BRUIT.

Afin d'évaluer la difficulté du problème, nous avons préalablement procédé à l'identification de prédicteurs du processus sans bruit. Nous avons vu au chapitre 2 §I.2.1, que le prédicteur associé à un tel modèle-hypothèse peut être indifféremment bouclé ou non. *La performance qui nous intéresse étant celle d'un modèle bouclé (pour l'apprentissage d'un correcteur avec ce modèle), l'apprentissage a été effectué avec des modèles bouclés, donc en semi-dirigé.* Le tableau 1 donne les performances obtenues.

Nombre de neurones cachés	EQMA	EQMT
3	$9{,}6\ 10^{-2}$	$2{,}9\ 10^{-2}$
4	$4{,}0\ 10^{-4}$	$2{,}1\ 10^{-4}$
5	$6{,}1\ 10^{-6}$	$3{,}7\ 10^{-6}$
6	$4{,}2\ 10^{-6}$	$3{,}2\ 10^{-6}$

Tableau 1.
Processus sans bruit : résultats de l'identification de réseaux prédicteurs bouclés.

[1] Une distribution du bruit suivant une loi gaussienne eût-elle été plus réaliste ? " Tout le monde y croit fermement parce que les mathématiciens s'imaginent que c'est un fait d'observation, et les observateurs que c'est un théorème de mathématiques. " [POI02].

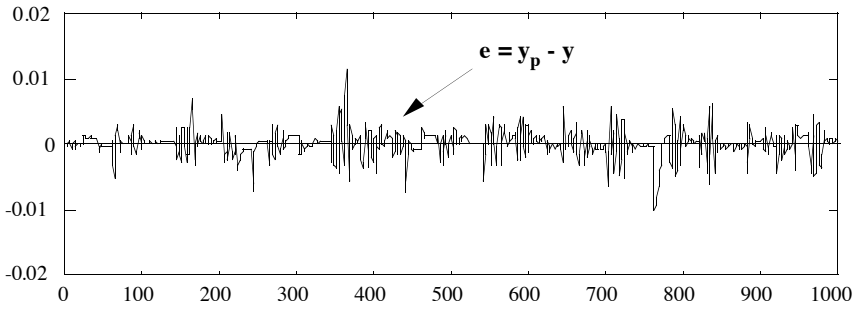


Figure 2.
Processus sans bruit : erreur de prédiction obtenue sur la séquence de test
avec le réseau de neurones bouclé à 5 neurones cachés.

L'ajout de neurones supplémentaires n'améliore pas sensiblement la précision. Ceci permet de conclure que 5 neurones cachés sont nécessaires et suffisent pour obtenir un prédicteur d'une précision satisfaisante. La figure 2 montre l'erreur du prédicteur bouclé à 5 neurones cachés, sur la séquence de test.

I.3. MODÉLISATION DU PROCESSUS AVEC BRUIT.

En présence de perturbations aléatoires, le prédicteur optimal est le prédicteur donnant une erreur de prédiction de variance minimale. Comme il s'agit d'un prédicteur à 1 pas (chapitre 2 §II.2), la variance minimale de l'erreur de prédiction obtenue avec ce prédicteur optimal est la variance de la perturbation aléatoire, qui vaut ici $3{,}33\ 10^{-3}$. Les erreurs quadratiques moyennes EQMA et EQMT doivent donc avoir une valeur voisine de celle-ci. Si l'EQMA est plus petite, il y a sur-apprentissage. Si l'EQMT est plus grande que la variance alors que la valeur de l'EQMA est correcte, c'est que l'apprentissage n'a pas été effectué dans un domaine représentatif du domaine de fonctionnement souhaité. Les précautions prises dans le choix de la séquence d'apprentissage nous ont permis d'éviter cette situation, comme le montrent les résultats du tableau 1. Les prédicteurs utilisés dans la suite possèdent 5 neurones cachés.

Afin de tester la qualité des prédicteurs obtenus en tant que modèles de simulation (cf. chapitre 3 §III.2), nous avons également évalué leur performance par rapport au processus *sans bruit*, sur la séquence de test (bien entendu, ceci n'est possible que pour un processus simulé). L'erreur quadratique moyenne ainsi obtenue est appelée EQMD (*D* pour *déterministe*). Si l'identification est réalisée dans de bonnes conditions, l'EQMD doit se rapprocher de l'EQMT obtenue par apprentissage avec le processus sans bruit (cf. tableau 1) ; en pratique, sa valeur est supérieure, ce d'autant plus que le bruit est important (10 à 100 fois pour nos exemples) et perturbe l'estimation.

Pour les processus NARX et NBSX, les conséquences du choix d'une hypothèse, vraie ou fausse, sur la qualité du prédicteur a été montrée dans [NER92a] ; notre apport sera d'analyser les résultats dans le cas d'une hypothèse fausse, en nous appuyant sur la fonction d'autocorrélation de l'erreur. Nous modélisons ensuite le processus NARMAX, modélisation dont on ne trouve guère

d'exemples à l'aide de réseaux de neurones (sauf dans [BIL92], pour le cas particulier d'un processus très simple affecté d'un bruit MA, voir chapitre 2 §II.2.1).

Enfin, nous ne simulons pas une démarche " réaliste " dans les choix successifs des modèles-hypothèse : nous formulons en général d'abord l'hypothèse vraie, puis les hypothèses fausses.

I.3.1. Processus avec perturbation additive d'état (NARX).

Le processus obéit à l'équation :

$$y_p(k) = h\ \left(y_p(k\text{-}1),\ y_p(k\text{-}2), u(k\text{-}1)\right) + w(k)$$

I.3.1.1. Hypothèse vraie NARX.

On fait ici l'hypothèse vraie NARX. Le prédicteur associé à l'hypothèse n'est pas bouclé (chapitre 2 §II.2.1.1). Pour réaliser ce prédicteur, on utilise un réseau de neurones non bouclé de la forme :

$$y(k+1) = \psi_{RN}\left(y_p(k), y_p(k\text{-}1), u(k);\ C\right)$$

L'algorithme d'apprentissage est dirigé. Les résultats obtenus en fin d'apprentissage sont donnés par le tableau 2.

EQMA	EQMT	*EQMD*
$3{,}2\ 10^{-3}$	$3{,}4\ 10^{-3}$	*$3{,}4\ 10^{-4}$*

Tableau 2.
Processus NARX : résultats de l'identification avec un réseau prédicteur NARX.

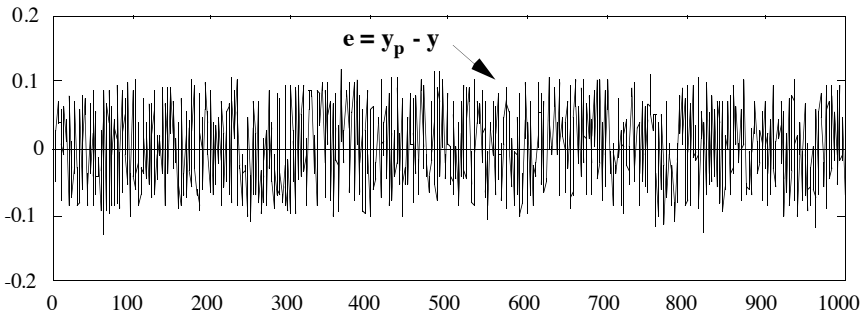


Figure 3.
Processus NARX : erreur de prédiction obtenue avec un réseau prédicteur NARX.

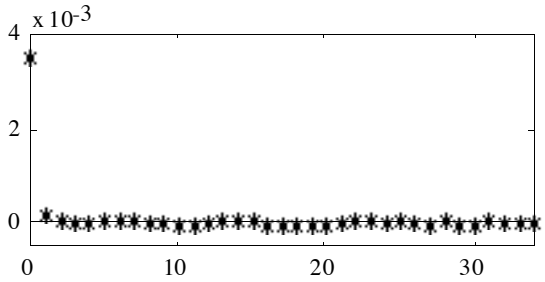


Figure 4.
Processus NARX : fonction d'autocorrélation de l'erreur de prédiction obtenue avec le réseau prédicteur NARX.

La figure 3 représente l'erreur de prédiction obtenue sur l'ensemble de test. Cette erreur présente bien les mêmes caractéristiques que la perturbation aléatoire (distribution, corrélation). En particulier, la fonction d'autocorrélation de l'erreur est bien celle d'un bruit blanc, c'est-à-dire nulle sauf en zéro, où elle prend la valeur de la variance du signal (cf. figure 4).

I.3.1.2. Hypothèse fausse NBSX.

Faisons à présent l'hypothèse fausse NBSX. Le prédicteur associé à cette hypothèse est bouclé d'ordre 2 sur la sortie du prédicteur (chapitre 2 §II.2.1.2). Pour identifier ce prédicteur, on utilise donc un réseau de neurones bouclé de la forme :

$$y(k+1) = \varphi_{RN}\left(y(k), y(k\text{-}1), u(k); C\right)$$

L'algorithme d'apprentissage est semi-dirigé. Les résultats obtenus en fin d'apprentissage sont donnés par le tableau 3 suivant.

EQMA	EQMT	*EQMD*
$7{,}4\ 10^{-3}$	$9{,}2\ 10^{-2}$	*$9{,}8\ 10^{-4}$*

Tableau 3.
Processus NARX : résultats de l'identification d'un réseau prédicteur NBSX

La variance de l'erreur de prédiction n'est que légèrement plus importante que lorsque l'hypothèse est vraie. L'EQMD montre d'ailleurs que le comportement déterministe est assez bien approché.

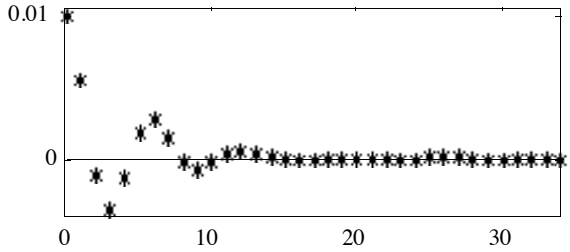


Figure 5.
Processus NARX : fonction d'autocorrélation de l'erreur de prédiction obtenue avec le réseau prédicteur NBSX

Cependant, la fonction d'autocorrélation de l'erreur est donnée sur la figure 5. Elle permet de conclure que l'erreur de prédiction n'est pas un bruit blanc et de rejeter l'hypothèse, par comparaison avec la fonction d'autocorrélation du prédicteur NARX précédent.

Interprétation de l'allure de la fonction de corrélation.

Interprétons l'allure de la fonction d'autocorrélation à l'aide d'une étude en linéaire. Soit le processus ARX suivant :

$$A(q)\ y_p(k) = B(q)\ u(k) + w(k)$$

Le prédicteur non bouclé associé au processus ARX est :

$$y(k) = \left(1 - A(q)\right) y_p(k) + B(q)\ u(k)$$

Supposons que l'on utilise le prédicteur linéaire suivant :

$$A(q)\ y(k) = B(q)\ u(k)$$

Les coefficients de ce prédicteur sont bien ceux du processus, mais il est bouclé, alors que le prédicteur associé au processus ne l'est pas. Exprimons l'erreur de prédiction en soustrayant ces deux expressions :

$$e(k) = y_p(k) - y(k) = \frac{1}{A(q)}\ w(k)$$

La fonction d'autocorrélation de l'erreur est celle d'un filtre AR dont l'ordre est celui du processus.

De même, dans le cas non linéaire, si l'identification du processus NARX est menée avec un prédicteur NBSX, mais que la fonction réalisée par le prédicteur est malgré tout proche de celle du modèle-hypothèse, alors l'erreur de prédiction sera essentiellement due au fait que le prédicteur est bouclé, et elle est analogue à celle que l'on obtient en linéaire.

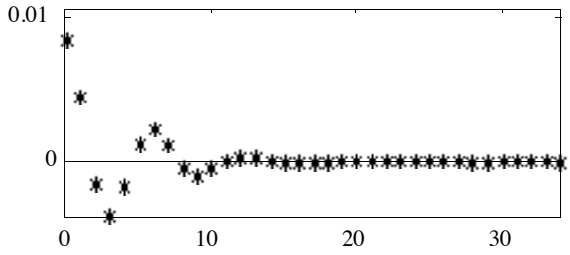


Figure 6.
Processus NARX : fonction d'autocorrélation de l'erreur de prédiction obtenue avec le réseau prédicteur NARX, mais utilisé bouclé.

La figure 6 montre justement l'autocorrélation de l'erreur obtenue avec le prédicteur NARX appris au paragraphe précédent (la partie non bouclée de ce prédicteur réalise donc la bonne fonction), mais utilisé bouclé. Cette autocorrélation ressemble beaucoup à celle d'une séquence AR du second ordre (le processus non linéaire est d'ordre 2). Elle est presque identique à celle de la figure 5.

I.3.2. Processus avec perturbation additive de sortie (NBSX).

Le processus obéit à l'équation :

$$\begin{cases} x_p(k) = h\left(x_p(k\text{-}1),\ x_p(k\text{-}2),\ u(k\text{-}1)\right) \\ y_p(k) = x_p(k) + w(k) \end{cases}$$

I.3.2.1. Hypothèse vraie NBSX.

On fait tout d'abord l'hypothèse vraie. Le prédicteur associé à cette hypothèse est bouclé d'ordre 2 sur la sortie du prédicteur (chapitre 2 §II.2.1.2). On utilise donc un réseau bouclé :

$$y(k+1) = \varphi_{RN}\left(y(k),\ y(k\text{-}1),\ u(k);\ C\right)$$

L'algorithme d'apprentissage est semi-dirigé. Les résultats obtenus sont donnés par le tableau 4.

EQMA	EQMT	*EQMD*
$3{,}3\ 10^{-3}$	$3{,}3\ 10^{-3}$	$6{,}3\ 10^{-5}$

Tableau 4.
Processus NBSX : résultats de l'identification d'un réseau prédicteur NBSX.

Le modèle de simulation associé au prédicteur identifié sera utilisé au chapitre 6 pour la commande. Nous notons ce modèle de simulation, dont les coefficients sont fixés, de la façon suivante :

$$y(k) = \varphi_{RN}^{Sim1}\left(y(k\text{-}1), y(k\text{-}2), u(k\text{-}1)\right)$$

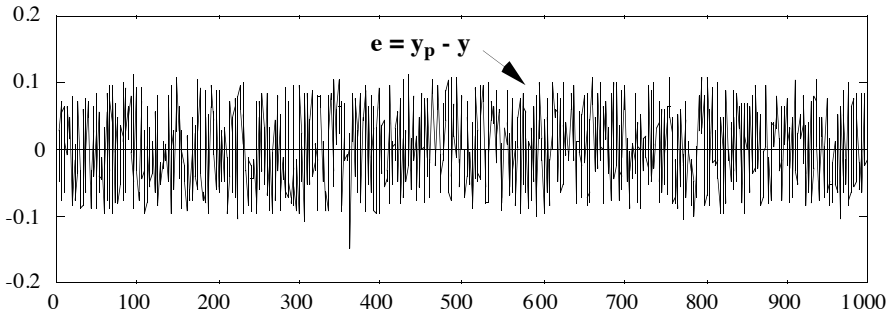


Figure 7.
Processus NBSX : erreur de prédiction obtenue avec le réseau prédicteur NBSX.

La figure 7 représente l'erreur de prédiction obtenue en fin d'apprentissage sur l'ensemble de test. Cette erreur présente bien les mêmes caractéristiques que la perturbation aléatoire de sortie (bruit blanc de valeur moyenne nulle et de distribution uniforme d'amplitude 0,2).

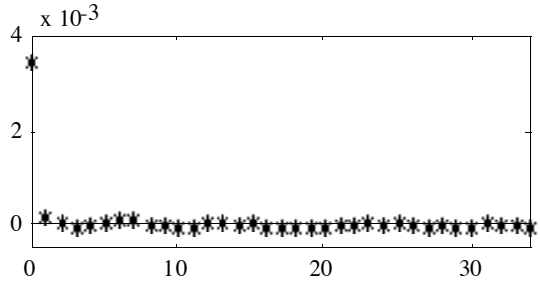


Figure 8.
Processus NBSX : fonction d'autocorrélation de l'erreur de prédiction obtenue avec le réseau prédicteur NBSX.

La fonction d'autocorrélation de l'erreur, donnée en figure 8, caractérise bien un bruit blanc.

I.3.2.2. Hypothèse fausse NARX.

Faisons maintenant l'hypothèse fausse NARX. Le prédicteur associé à l'hypothèse n'est pas bouclé (chapitre 2 §II.2.1.1). Pour identifier ce prédicteur, nous utilisons donc un réseau de neurones non bouclé de la forme :

$$y(k+1) = \psi_{RN}\left(y_p(k), y_p(k\text{-}1), u(k); C\right)$$

L'algorithme utilisé est donc dirigé. Les résultats obtenus sont donnés par le tableau 5.

EQMA	EQMT	*EQMD*
$1{,}0\ 10^{-2}$	$2{,}0\ 10^{-2}$	$1{,}2\ 10^{-2}$

Tableau 5.
Processus NBSX : résultats de l'identification d'un réseau prédicteur NARX.

La variance de l'erreur de prédiction est très importante, ce qui trahit une très mauvaise modélisation. Ceci est confirmé par une valeur importante de l'EQMD, beaucoup plus importante que celle obtenue lors de l'identification d'un processus NARX avec un prédicteur NBSX.

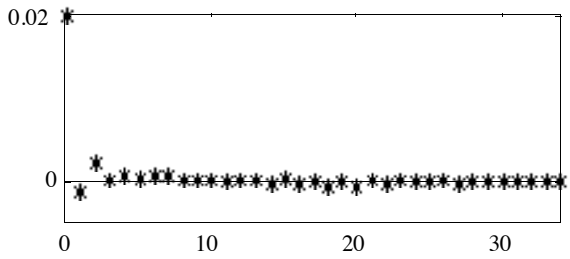


Figure 9.
Processus NBSX : fonction d'autocorrélation de l'erreur de prédiction obtenue avec le réseau prédicteur NARX.

La fonction d'autocorrélation, donnée sur la figure 9, a presque les caractéristiques de celle d'un bruit blanc. Il faut donc se fonder aussi bien sur la valeur de la variance de l'erreur de prédiction que sur son autocorrélation pour comparer les hypothèses. On observera souvent ce type de résultats lors de l'identification d'un processus NBSX avec un prédicteur NARX. Expliquons pourquoi, ici encore à l'aide des résultats que l'on peut établir en linéaire.

Interprétation de l'allure de la fonction d'autocorrélation.

Soit le processus linéaire NBSX suivant :

$$A(q)\ y_p(k) = B(q)\ u(k) + A(q)\ w(k)$$

Le prédicteur associé à ce processus est bouclé :

$$A(q)\ y(k) = B(q)\ u(k)$$

Supposons cependant que l'on utilise le prédicteur non bouclé suivant, ayant les mêmes coefficients que le processus :

$$y(k) = \big(1 - A(q)\big)\ y_p(k) + B(q)\ u(k)$$

L'erreur de prédiction est obtenue en soustrayant ces deux expressions :

$$e(k) = A(q)\ w(k)$$

La fonction d'autocorrélation est donc celle d'un filtre MA. Si le processus est d'ordre n, elle possède donc 2n+1 valeurs non nulles (donc n+1 si l'on ne considère la fonction, paire, que sur $\mathbb{N}$).

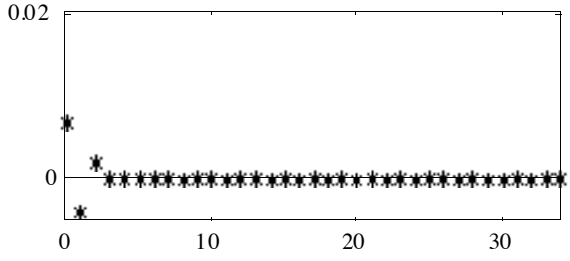


Figure 10.
Processus NBSX : fonction d'autocorrélation de l'erreur de prédiction obtenue avec le réseau prédicteur NBSX, mais utilisé non bouclé.

De même, en non linéaire, on obtient des résultats analogues. La figure 10 représente la fonction d'autocorrélation de l'erreur obtenue en utilisant le prédicteur associé NBSX obtenu au paragraphe précédent (la partie non bouclée du prédicteur réalise la bonne fonction), mais sans le boucler. Cette fonction d'autocorrélation a les mêmes caractéristiques que celles d'une séquence AR. En raison des erreurs de modélisations dues à l'utilisation d'un mauvais prédicteur, les fonctions d'autocorrélation des figures 9 et 10 sont toutefois différentes.

Ceci montre que l'identification d'un processus à l'aide d'un prédicteur non bouclé (NARX) d'un processus qui exigerait un prédicteur bouclé (NBSX) conduit souvent à une erreur très peu corrélée. Il faut donc impérativement effectuer d'autres hypothèses, et comparer les prédicteurs obtenus pour chacune d'elles, non seulement sur la base de la corrélation de l'erreur, mais aussi sur la variance de celle-ci. *Dans la réalité, il se peut aussi qu'aucune hypothèse ne soit vraie, et il est alors recommandé, comme nous y avons insisté au chapitre 3 §III.2, de comparer les prédicteurs selon les performances de leurs modèles de simulation associés, afin de ne pas favoriser le prédicteur NARX*.

I.3.3. Processus NARMAX.

Le processus obéit à l'équation :

$$y_p(k) = h\left(y_p(k\text{-}1),\ y_p(k\text{-}2), u(k\text{-}1)\right) + 0{,}5\ y_p(k\text{-}1)\ w(k\text{-}1) + w(k)$$

I.3.3.1. Hypothèse vraie NARMAX.

Faisons tout d'abord l'hypothèse vraie. Le prédicteur associé à l'hypothèse NARMAX est bouclé sur l'erreur, d'ordre 1 (chapitre 2 §III.1.3). Pour l'identifier, il faut donc utiliser un réseau bouclé de la forme :

$$y(k\text{+}1) = \varphi_{RN}\left(y_p(k), y_p(k\text{-}1), u(k), e(k); C\right)$$

Le tableau 9 donne les résultats obtenus en fin d'apprentissage :

EQMA	EQMT	*EQMD*
$3{,}9\ 10^{-3}$	$3{,}5\ 10^{-3}$	*$4{,}1\ 10^{-4}$*

Tableau 9.
Processus NARMAX : résultats de l'identification d'un réseau prédicteur NARMAX.

En toute rigueur, la fonction à approcher étant plus complexe, il aurait fallu prendre un réseau comportant plus de neurones cachés. Les résultats ci-dessus étant quasiment optimaux (EQMT de $3{,}54\ 10^{-3}$ alors que la valeur théorique est de $3{,}33\ 10^{-3}$), nous n'avons pas jugé nécessaire d'agrandir le réseau, pour cette présentation. De plus, le prédicteur utilisé comme modèle de simulation a d'excellentes performances.

La figure 12 représente l'erreur de prédiction obtenue en fin d'apprentissage sur l'ensemble de test. Cette erreur présente bien encore les mêmes caractéristiques que la perturbation aléatoire. Il est absolument nécessaire, pour obtenir un tel résultat, de disposer d'algorithmes d'optimisation efficaces, tels que ceux présentés en annexe I.

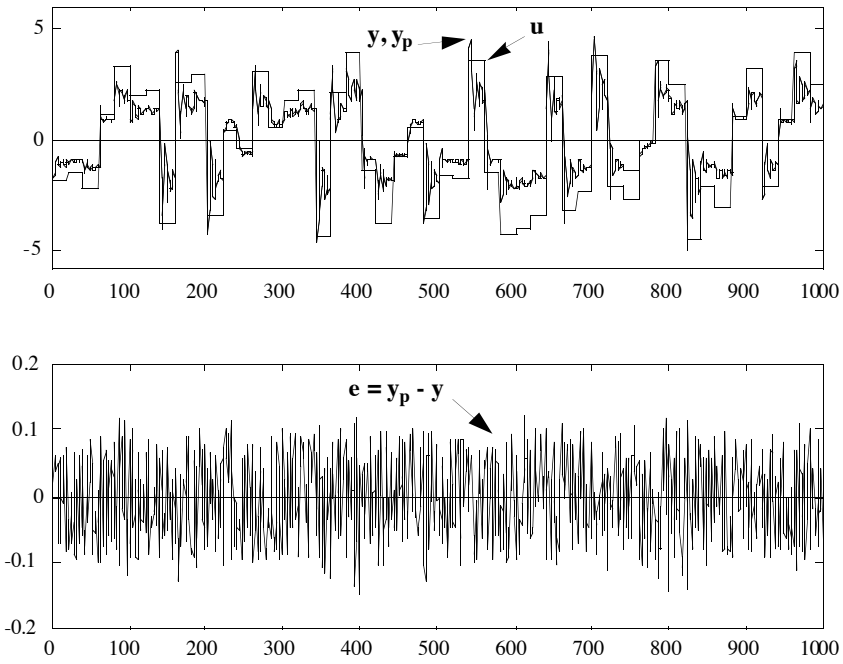


Figure 12.
Processus NARMAX : commande, sorties du processus et du réseau prédicteur, erreur de prédiction, obtenues avec le réseau prédicteur NARMAX.

I.3.3.2. Hypothèse fausse NBSX.

Si l'on fait l'hypothèse fausse qu'il s'agit d'un processus NBSX, et que l'on identifie le prédicteur associé à l'aide d'un réseau de même structure, on obtient les résultats du tableau 10.

EQMA	EQMT	*EQMD*
$1,5\ 10^{-2}$	$1,5\ 10^{-2}$	$1,3\ 10^{-3}$

Tableau 10.
Processus NARMAX : résultats de l'identification d'un réseau prédicteur NBSX.

Le prédicteur ainsi obtenu a encore une performance très mauvaise pour la prédiction (valeur de l'EQMT très supérieure à la valeur théorique $3,33\ 10^{-3}$). Comme le montre la figure 13a, l'erreur de prédiction n'a plus les caractéristiques de la perturbation aléatoire. Sa distribution n'est pas uniforme, son amplitude est beaucoup plus importante, et la corrélation est apparente (elle serait révélée par des tests statistiques). Utilisé comme modèle de simulation, ce prédicteur est également peu satisfaisant.

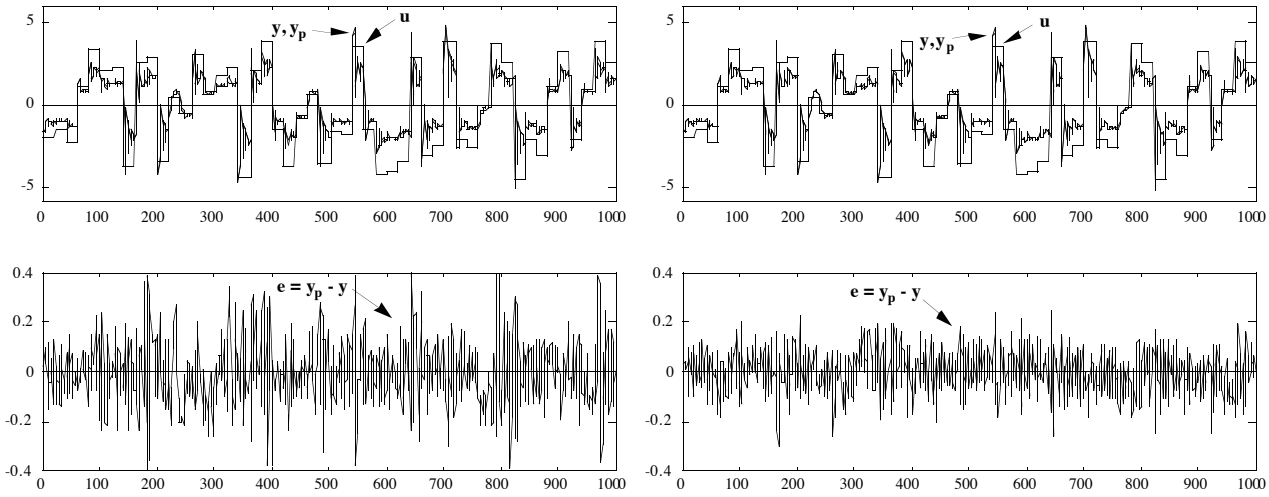
I.3.3.3. Hypothèse fausse NARX.

Si l'on fait l'hypothèse fausse qu'il s'agit d'un processus NARX, et que l'on identifie le prédicteur associé à l'aide d'un réseau de même structure, on obtient les résultats du tableau 11.

EQMA	EQMT	*EQMD*
$6,7\ 10^{-3}$	$7,0\ 10^{-2}$	*$4,2\ 10^{-3}$*

Tableau 11.
Processus NARMAX : résultats de l'identification d'un réseau prédicteur NARX.

Le réseau ainsi obtenu n'est toujours pas optimal comme prédicteur, quoique meilleur que le précédent. Ceci est normal, puisque le bruit est un bruit d'état (bien que coloré). Ici encore, comme le montre la figure 13b, l'erreur de prédiction n'a plus la distribution de la perturbation aléatoire et trahit l'erreur faite sur l'hypothèse. Utilisé comme modèle de simulation, il est particulièrement imprécis.



a) Réseau prédicteur NBSX. b) Réseau prédicteur NARX.

Figure 13.
Processus NARMAX : commande, sorties du processus et du réseau prédicteur, erreur de prédiction.

II. MODÉLISATION D'UN PROCESSUS SIMULÉ PAR UN MODÈLE D'ÉTAT.

II.1. PRÉSENTATION.

Le choix du modèle d'état pour cette présentation a été guidé par plusieurs préoccupations :
a) Tout d'abord, celle de montrer l'utilité de modèles d'état pour réaliser des prédicteurs, souvent très économes en nombre de neurones (parcimonieux).
b) Ensuite, comme pour l'exemple précédent, nous avons le souci de montrer l'influence des hypothèses concernant les perturbations aléatoires affectant le processus, que le prédicteur choisi soit de type état ou entrée-sortie.
c) Enfin, pour ne pas multiplier les exemples de processus, nous voulons réaliser la commande de ce même processus au chapitre 6, et démontrer avec lui les propriétés de robustesse des systèmes de commande qui sont présentés au chapitre 5. Pour cela, il est nécessaire de construire un processus non linéaire qui reste analysable et permette l'interprétation des résultats de la commande.

Processus sans bruit.

Les équations d'état et d'observation du processus sont les suivantes :

$$\begin{cases} x_{1p}(k+1) = a_1\, x_{1p}(k) + a_2\, x_{2p}(k) + (b_1 + 2\, b_2)\, u(k) \\ x_{2p}(k+1) = \dfrac{x_{1p}(k)}{1 + 0{,}01\, x_{2p}(k)^2} + \dfrac{(-\, b_2)}{a_2}\, u(k) \\ y_p(k) = 4\, tanh\left(\dfrac{x_{1p}(k)}{4}\right) \end{cases}$$

avec $a_1 = 1{,}145$; $a_2 = -\,0{,}549$; $b_1 = 0{,}222$; $b_2 = 0{,}181$.

Ces équations ont été construites à partir du modèle linéaire suivant :

$$y_p(k) = a_1\, y_p(k\text{-}1) + a_2\, y_p(k\text{-}2) + b_1\, u(k\text{-}1) + b_2\, u(k\text{-}2)$$

Cette équation résulte de la discrétisation exacte du second ordre linéaire de pulsation ω_n=3, d'amortissement ξ=0,4 et de gain K=1, avec une période T=0,25 s.

Le comportement du processus en réponse à des échelons de commande est représenté sur la figure 14. Autour de zéro, le processus a une dynamique linéaire oscillante, et un gain unité. Le dénominateur de la deuxième variable d'état lui donne un caractère de plus en plus amorti et plus lent pour des amplitudes supérieures à 1 ; il diminue aussi le gain, ce qui est encore accentué par la tangente hyperbolique de l'équation d'observation. Le zéro du processus linéaire a été modifié pour le rendre positif, afin d'éviter des oscillations parasites lors de la commande.

Séquences utilisées pour l'identification.

Le processus étant plus simple que le précédent, et notre but étant de nous rapprocher de conditions expérimentales qui ne permettent pas en général de réaliser une identification " parfaite ", ces séquences sont choisies moins riches que celles utilisées pour l'identification du processus du §I.

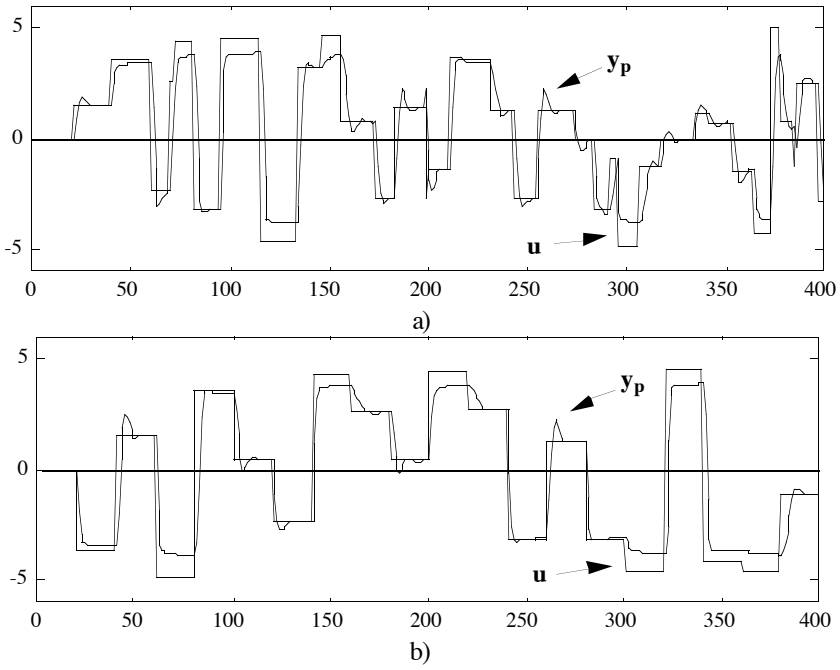


Figure 14.
a) Séquences d'apprentissage ; b) Séquences de test.

a) Séquences d'apprentissage :

La séquence de commande utilisée pour l'apprentissage des réseaux prédicteurs est constituée d'une série de créneaux d'amplitudes aléatoires entre ±5, de durées aléatoires entre 1 et 20 pas d'échantillonnage. La séquence totale comporte 400 pas d'échantillonnage : elle est représentée sur la figure 14a. EQMA est encore l'erreur quadratique moyenne obtenue sur la séquence d'apprentissage.

b) Séquences de test :

La séquence de commande utilisée pour le test des prédicteurs est constituée d'une série de créneaux d'amplitudes aléatoires entre ±5, de durée 20 pas d'échantillonnage. La séquence totale comporte 400 pas d'échantillonnage : elle est représentée sur la figure 14b. EQMT est encore l'erreur quadratique moyenne obtenue sur la séquence de test.

Prédicteurs.

Les prédicteurs utilisés sont des réseaux complètement connectés, à neurones cachés dont la fonction d'activation est la tangente hyperbolique, et à neurone de sortie linéaire.

Processus avec bruit.

Nous effectuons une étude moins exhaustive que pour l'exemple précédent, en partie en raison du fait que l'on ne mesure pas l'état. Soit encore w un bruit blanc à valeur moyenne nulle et de distribution uniforme d'amplitude 0,2 (donc de variance $3{,}33\ 10^{-3}$). Nous considérons les processus perturbés suivants :

- processus avec perturbation additive de sortie (NBSX) :

$$\begin{cases} x_{1p}(k+1) = a_1\, x_{1p}(k) + a_2\, x_{2p}(k) + (b_1 + 2\, b_2)\, u(k) \\ x_{2p}(k+1) = \dfrac{x_{1p}(k)}{1 + 0{,}01\, x_{2p}(k)^2} + \dfrac{(-\, b_2)}{a_2}\, u(k) \\ y_p(k) = 4\, tanh\left(\dfrac{x_{1p}(k)}{4}\right) + w(k) \end{cases}$$

- processus avec perturbation additive d'état (noté ici BE) :

$$\begin{cases} x_{1p}(k+1) = a_1\, x_{1p}(k) + a_2\, x_{2p}(k) + (b_1 + 2\, b_2)\, u(k) \\ x_{2p}(k+1) = \dfrac{x_{1p}(k)}{1 + 0{,}01\, x_{2p}(k)^2} + \dfrac{(-\, b_2)}{a_2}\, u(k) + w(k) \\ y_p(k) = 4\, tanh\left(\dfrac{x_{1p}(k)}{4}\right) \end{cases}$$

II.2. MODÉLISATION D'ÉTAT.

Comme on ne mesure pas l'état, que le processus soit perturbé ou non, on utilise un réseau prédicteur d'état (bouclé) de la forme :

$$\begin{cases} x(k+1) = \varphi_{RN}(x(k), u(k); C) \\ y(k+1) = \psi_{RN}(x(k), u(k); C) \end{cases}$$

II.2.1. Processus sans bruit.

Des performances excellentes sont obtenues avec de très petits réseaux (voir tableau 12).

Nombre de neurones cachés	EQMA	EQMT
1 (16 coefficients)	$5{,}5\ 10^{-4}$	$4{,}7\ 10^{-4}$
2 (24 coefficients)	$1{,}6\ 10^{-7}$	$3{,}8\ 10^{-7}$

Tableau 12.
Processus d'état sans bruit : résultats de l'identification de réseaux prédicteurs d'état.

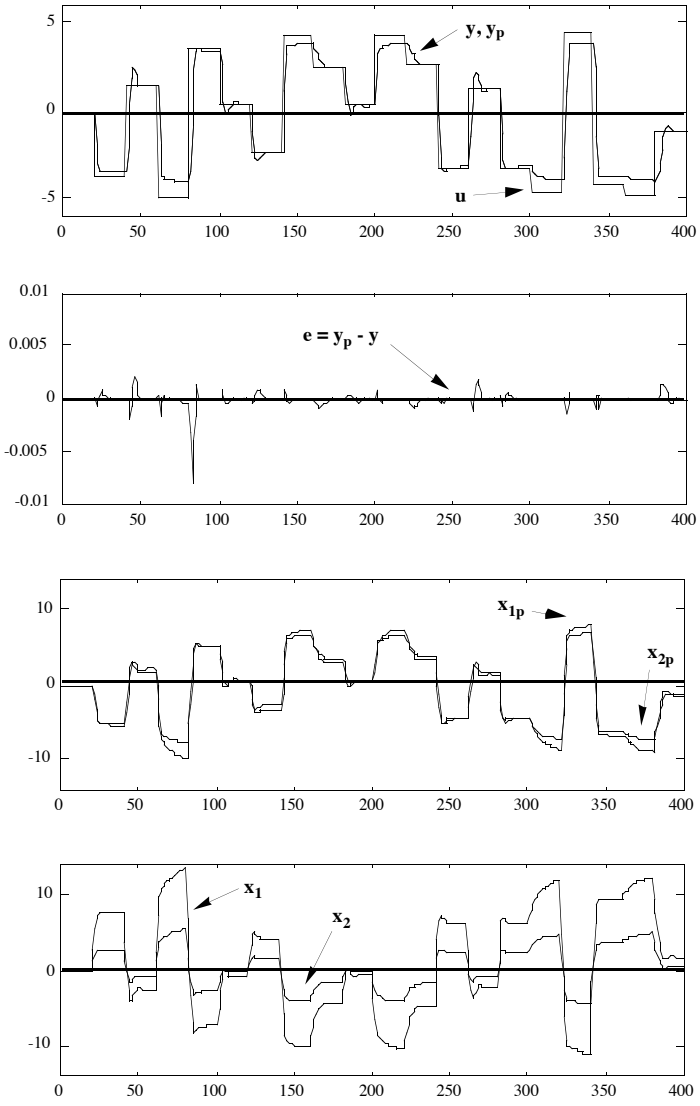


Figure 15.
Processus sans bruit : résultats obtenus avec le réseau prédicteur d'état à deux neurones cachés.

Les résultats de l'identification sont représentés sur la figure 15. Cette figure montre bien que les variables d'état utilisées par le réseau, x_1 et x_2, ne sont en général pas identiques à celles du processus x_{1p} et x_{2p}. L'ajout de neurones supplémentaires n'apporte pas d'amélioration sensible sur la séquence de test. Les identifications suivantes sont ainsi réalisées à l'aide d'un réseau prédicteur d'état possédant deux neurones cachés.

II.2.2. Processus NBSX.

En présence d'une perturbation aléatoire de sortie, le prédicteur d'état permet théoriquement d'identifier le prédicteur optimal. Les résultats du tableau 13 le confirment.

EQMA	EQMT	*EQMD*
$3{,}2\ 10^{-3}$	$3{,}5\ 10^{-3}$	*$1{,}2\ 10^{-4}$*

Tableau 13.
Processus d'état NBSX : résultats de l'identification d'un réseau prédicteur d'état.

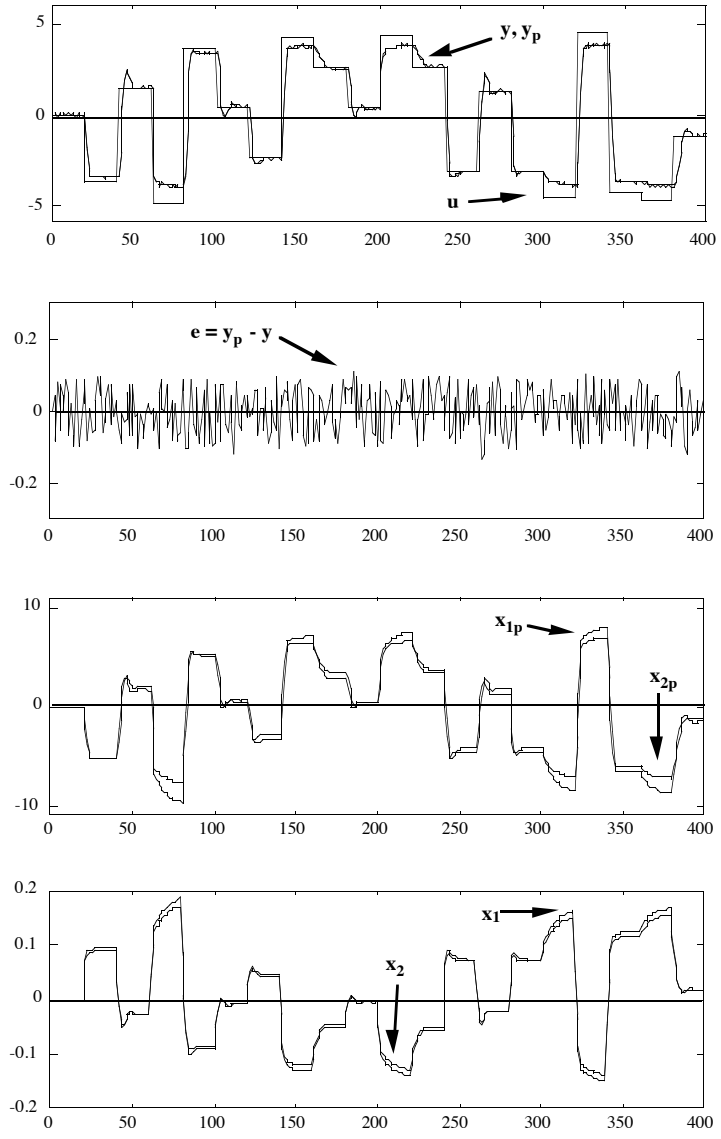


Figure 16.
Processus d'état NBSX : résultats obtenus avec un réseau prédicteur d'état (bouclé).

Comme le montre la figure 16, les variables d'état du prédicteur sont encore différentes. Très souvent, ce sont approximativement des multiples de celles du processus (cf. chapitre 2 §I.2.2.2), lorsque l'on fait la bonne hypothèse toutefois.

II.2.3. Processus BE.

En présence d'un bruit d'état, le prédicteur d'état (bouclé) ne permet théoriquement pas d'identifier le prédicteur optimal. Les résultats du tableau 14 le confirment.

EQMA	EQMT	*EQMD*
$1,2\ 10^{-3}$	$1,2\ 10^{-3}$	$6,5\ 10^{-4}$

Tableau 14.
Processus d'état BE : résultats de l'identification d'un réseau prédicteur d'état (bouclé).

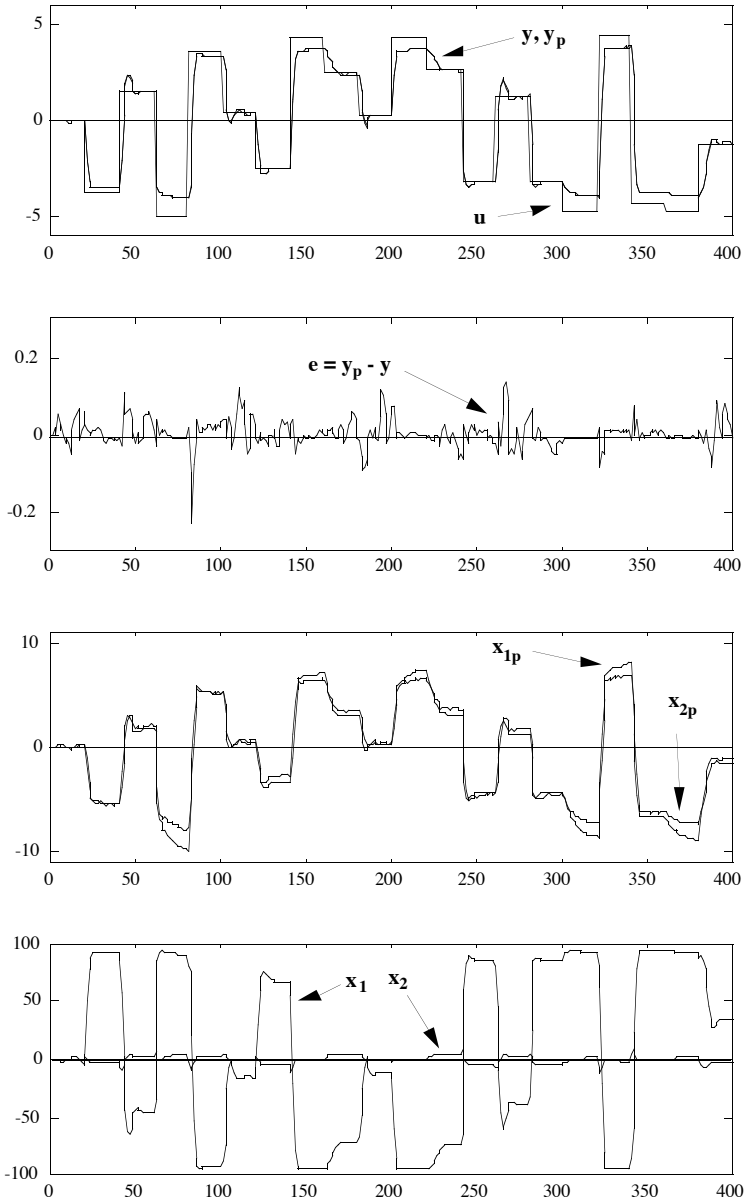


Figure 17.
Processus d'état BE : résultats obtenus avec un réseau prédicteur d'état (bouclé).

L'erreur de prédiction n'est pas un bruit blanc (cf. figure 17). On constate ici que les variables d'état du prédicteur ne sont plus des multiples de celles du processus. Cependant, la comparaison au processus sans bruit (EQMD) montre que le prédicteur identifié est un bon modèle de simulation.

II.3. MODÉLISATION ENTRÉE-SORTIE.

II.3.1. Processus sans bruit.

On peut par exemple utiliser un prédicteur entrée-sortie bouclé (algorithme semi-dirigé) :

$$y(k+1) = \varphi_{RN}(y(k), y(k-1), u(k), u(k-1); C)$$

Avec un prédicteur bouclé, on obtient les résultats donnés du tableau 12.

Nombre de neurones cachés	EQMA	EQMT
4 (35 coefficients)	$9{,}2\ 10^{-4}$	$7{,}1\ 10^{-4}$
5 (45 coefficients)	$2{,}1\ 10^{-4}$	$3{,}1\ 10^{-4}$
…	…	…
10 (110 coefficients)	$1{,}1\ 10^{-6}$	$1{,}2\ 10^{-4}$

Tableau 12.
Processus d'état sans bruit : résultats de l'identification de réseaux prédicteurs d'état.

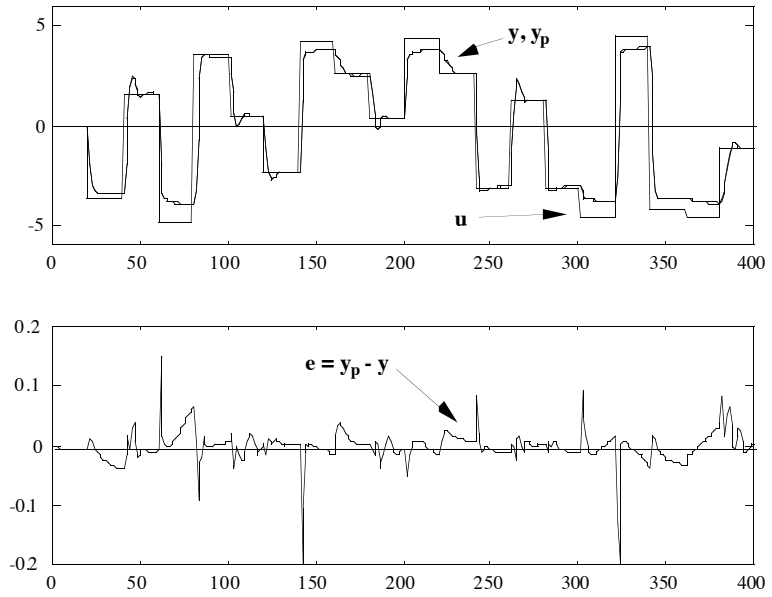


Figure 18.
Processus d'état sans bruit : résultats obtenus avec le réseau prédicteur bouclé entrée-sortie possédant 5 neurones cachés.

L'identification par un réseau prédicteur entrée-sortie se révèle peu économe en nombre de neurones. Avec un prédicteur non bouclé, les résultats sont encore moins bons. Augmenter l'ordre du réseau reste également sans effet appréciable (nous avons vu au chapitre 2 §I.2.2.3 que cela peut être nécessaire pour obtenir une représentation entrée-sortie équivalente au modèle d'état).

Les résultats correspondant au prédicteur bouclé à 5 neurones cachés sont montrés sur la figure 18. Dans un souci de réalisme, en particulier parce que les prédicteurs identifiés doivent servir au chapitre 6 pour la commande, nous nous limitons à cette taille de réseau pour la suite.

II.3.2. Processus NBSX.

Le processus est simulé par :

$$\begin{cases} x_1(k+1) = a_1\, x_1(k) + a_2\, x_2(k) + (b_1 + 2\, b_2)\, u(k) \\ x_2(k+1) = \dfrac{x_1(k)}{1 + 0{,}01\, x_2(k)^2} + \dfrac{(-\, b_2)}{a_2}\, u(k) \\ y_p(k) = 4\, tanh\left(\dfrac{x_1(k)}{4}\right) + w(k) \end{cases}$$

II.3.2.1. Hypothèse vraie NBSX.

S'il existe une représentation entrée-sortie du processus sans bruit, alors il doit exister un prédicteur entrée-sortie bouclé optimal. On peut donc espérer l'identifier à l'aide d'un réseau prédicteur bouclé, ce que confirment les résultats du tableau 17.

EQMA	EQMT	*EQMD*
$3{,}3\ 10^{-3}$	$4{,}5\ 10^{-3}$	*1,2 10-4*

Tableau 17.
Processus d'état NBSX : résultats de l'identification d'un réseau prédicteur entrée-sortie bouclé.

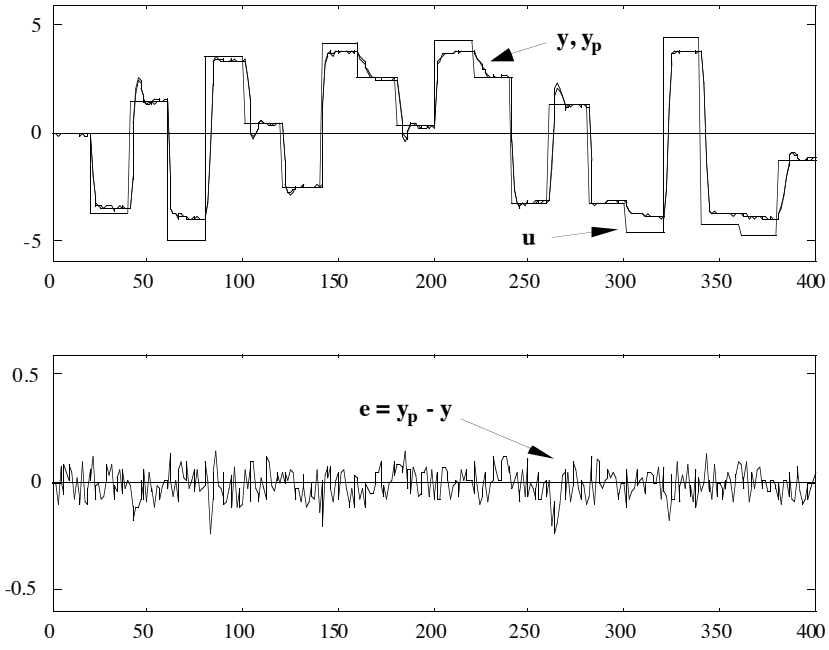


Figure 19.
Processus d'état NBSX : résultats obtenus avec un réseau prédicteur entrée-sortie bouclé.

Malgré (ou peut-être grâce à ?) la petitesse du réseau, les résultats sont quasi optimaux (valeur de l'EQMT proche de la valeur théorique de $3{,}33\ 10^{-3}$). Ceci se traduit par l'obtention d'un modèle de simulation assez satisfaisant (EQMD). Les résultats sont présentés sur la figure 19.

II.3.2.2. Hypothèse fausse.

En revanche, si l'on fait l'hypothèse d'un bruit d'état dominant et que l'on tente l'identification à l'aide d'un prédicteur entrée-sortie non bouclé (algorithme dirigé), les résultats sont mauvais (cf. tableau 18).

EQMA	EQMT	*EQMD*
$6{,}2\ 10^{-3}$	$9{,}1\ 10^{-3}$	$6{,}5\ 10^{-4}$

Tableau 18.
Processus d'état NBSX : résultats de l'identification d'un réseau prédicteur entrée-sortie non bouclé.

La figure 20 montre les résultats correspondants. Utilisé comme modèle de simulation, le prédicteur obtenu est particulièrement médiocre.

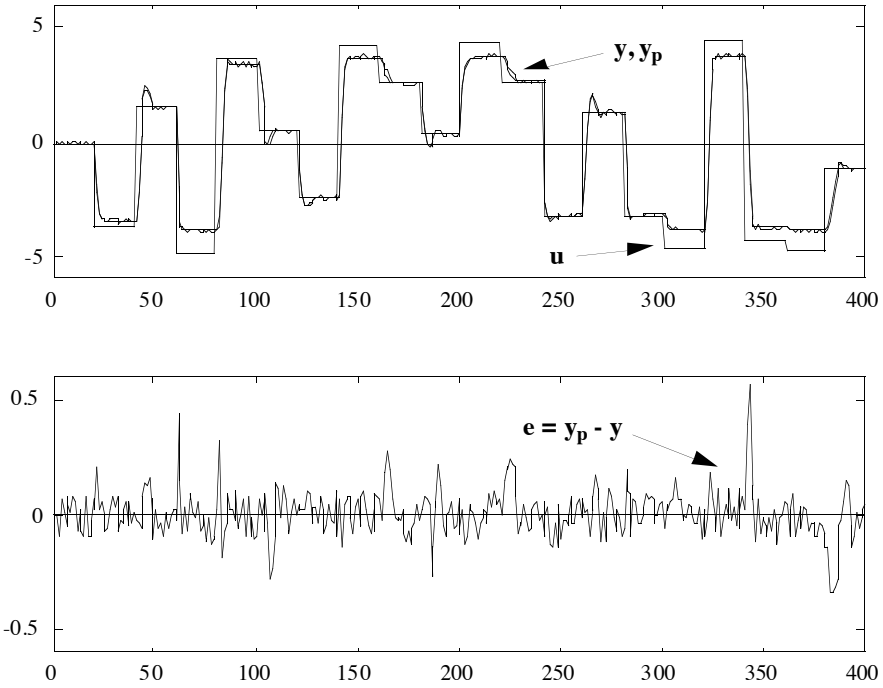


Figure 20.
Processus d'état NBSX : résultats obtenus avec un réseau prédicteur entrée-sortie non bouclé.

Cependant, c'est à partir de ce prédicteur que nous établissons un modèle de simulation pour l'apprentissage de correcteurs au chapitre 6, afin de mettre en évidence et de comparer les propriétés de robustesse des différentes méthodes de commande vis-à-vis de défauts de modélisation. Nous notons ce modèle de simulation (bouclé) :

$$y(k) = \varphi_{RN}^{Sim2}\big(y(k\text{-}1),\, y(k\text{-}2),\, u(k\text{-}1),\, u(k\text{-}2)\big)$$

où φ_{RN}^{Sim2} est la fonction réalisée par la partie non bouclée du réseau prédicteur avec les coefficients fixés aux valeurs obtenues en fin d'apprentissage.

III. MODÉLISATION D'UN ACTIONNEUR HYDRAULIQUE.

III.1. PRÉSENTATION.

La position d'un bras de robot est commandée par un actionneur hydraulique. La position du bras dépend de la pression d'huile dans l'actionneur, pression commandée par l'ouverture d'une vanne. Les variations de l'ouverture de la vanne, c'est-à-dire la séquence de commande {u(k)}, et la pression d'huile correspondante, c'est-à-dire la séquence de sortie $\{y_p(k)\}$, sont montrées sur la figure 21. Ce fichier de données[2] contient 1024 points de mesure : la première moitié d'entre eux est utilisée pour l'apprentissage, la seconde pour l'estimation de la performance

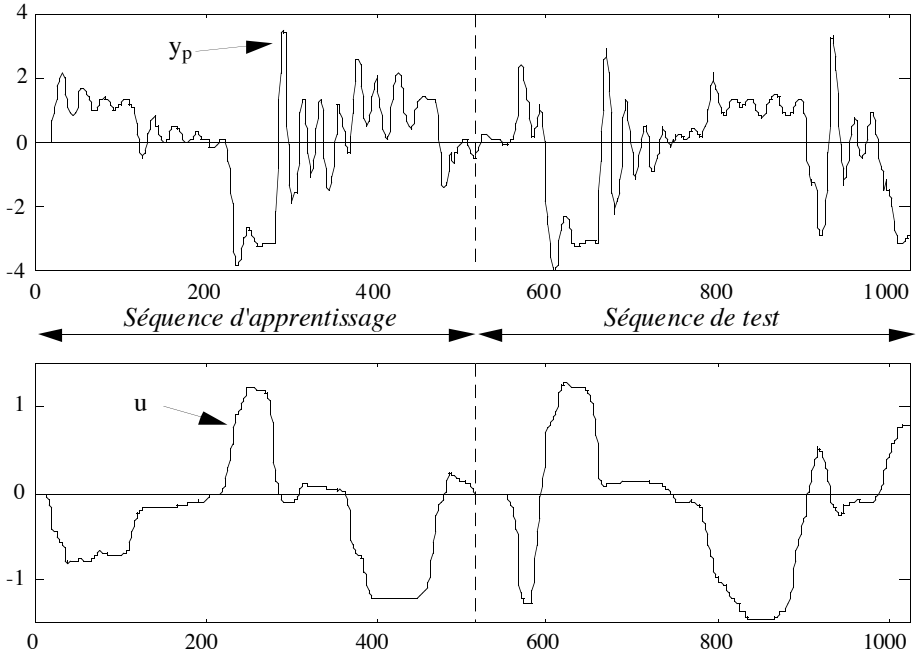


Figure 21.
Séquences d'apprentissage et de test pour la modélisation d'un bras de robot.

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

---

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

Nous avons tout d'abord envisagé des modèles entrée-sortie, puis des modèles d'état.

III.2. MODÉLISATION ENTRÉE-SORTIE.

III.2.1. Modèle-hypothèse NARX.

Nous avons considéré des modèles NARX :

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

où w est un bruit blanc, pour plusieurs valeurs de n et de m. Les modèles de simulation associés aux meilleurs prédicteurs NARX (c'est-à-dire ces prédicteurs *utilisés bouclés*) ont de très mauvaises performances, aussi bien sur la séquence d'apprentissage que sur la séquence de test (ils sont même parfois instables). Ceci signifie soit que l'hypothèse NARX n'est pas vraie, soit que la part déterministe du comportement dynamique du processus est mal modélisée, par exemple parce que l'on ne dispose pas de toutes les entrées du processus. Quoi qu'il en soit, les modèles de simulation associés aux prédicteur NARX sont inutilisables.

III.2.2. Modèle-hypothèse NBSX.

Le modèle-hypothèse NBSX est le suivant :

$$\begin{cases} x_p(k) = h\left(x_p(k\text{-}1), \ldots, x_p(k\text{-}n), u(k\text{-}1), \ldots u(k\text{-}m)\right) \\ y_p(k) = x_p(k) + w(k) \end{cases}$$

où w est un bruit blanc. Le prédicteur associé est de la forme :

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

Nous avons effectué l'apprentissage de prédicteurs neuronaux de la structure du prédicteur associé, pour plusieurs valeurs de m et de n :

$$y(k+1) = \varphi_{RN}\left(y(k), \ldots, y(k\text{-}n+1), u(k), \ldots, u(k\text{-}m+1); C\right)$$

Ces prédicteurs sont bouclés, l'apprentissage est donc semi-dirigé.

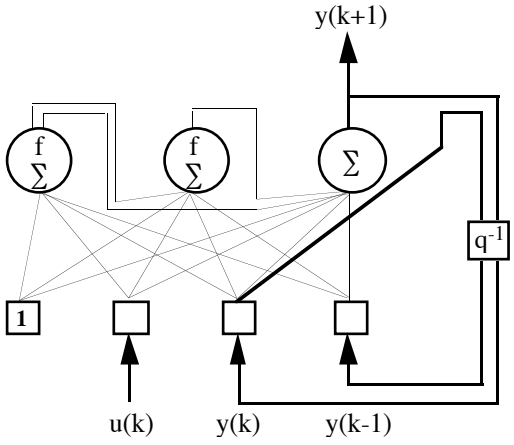
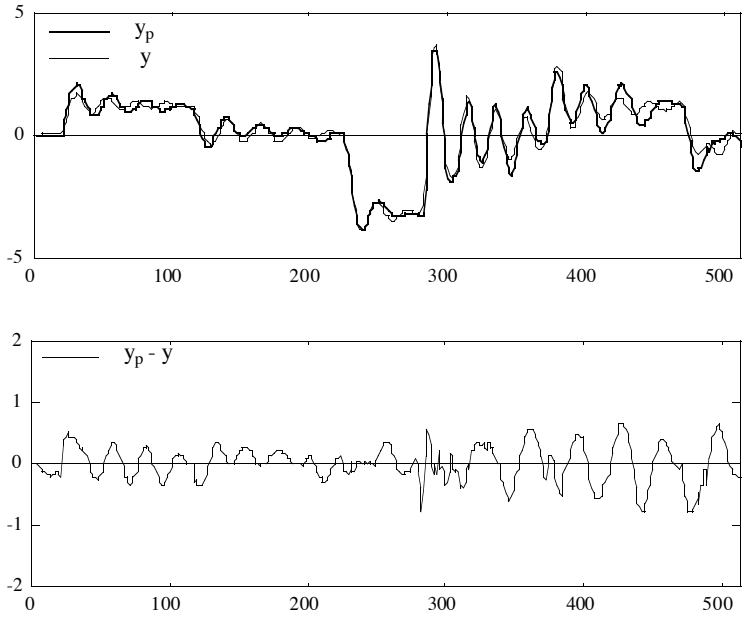
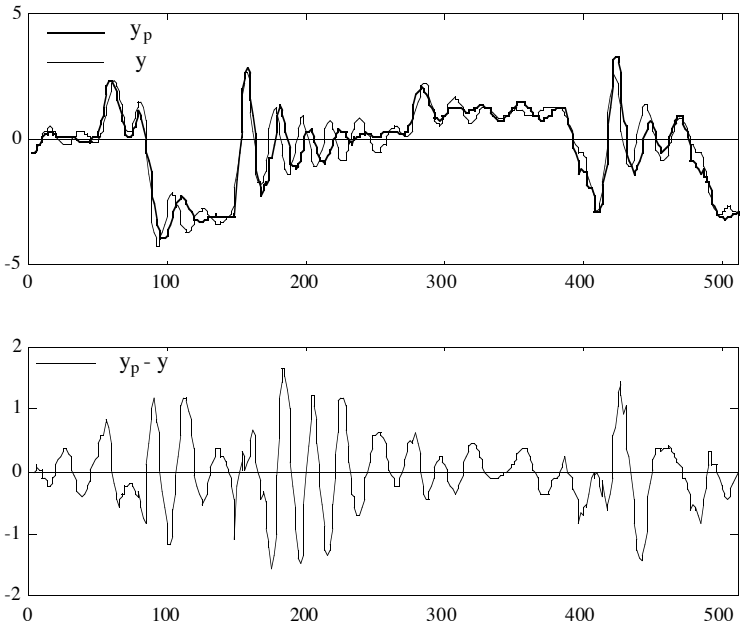


Figure 22.
Prédicteur entrée-sortie bouclé.

Les meilleurs résultats sont obtenus avec n=2 et m=1, et 2 neurones cachés (nous utilisons des réseaux complètement connectés à neurones cachés tangente hyperbolique, et neurone de sortie linéaire). Ce prédicteur est représenté sur la figure 22 (il possède 15 coefficients ajustables). Le passage à 3 neurones cachés (ou plus) diminue l'EQMA, mais augmente l'EQMT.



a) Séquence d'apprentissage : EQMA = 0,085.



b) Séquence de test : EQMT = 0,305.

Figure 23.
Performance du prédicteur entrée-sortie bouclé (NBSX).

Les résultats obtenus avec le prédicteur précédent sur les séquences d'apprentissage et de test sont représentés sur la figure 23. L'EQMA et l'EQMT sont de 0,085 et 0,305 respectivement. L'erreur n'a pas les caractéristiques d'un bruit blanc : l'hypothèse NBSX avec n=2 et m=1 n'est donc pas non plus vraie. Néanmoins, un prédicteur inexact NBSX (apprentissage semi-dirigé) est nécessairement

un meilleur modèle de simulation qu'un prédicteur inexact NARX (apprentissage dirigé) que l'on boucle pour simuler le processus.

III.3. MODÉLISATION D'ÉTAT.

Il apparaît nettement que la plus grande part de l'erreur de prédiction obtenue avec le prédicteur précédent n'est pas due à un bruit aléatoire (on distingue même une composante sinusoïdale), mais à *un défaut de modélisation déterministe*. Nous avons donc mis en œuvre des prédicteurs plus généraux, les prédicteurs d'état. Ces prédicteurs sont associés au modèle-hypothèse d'état suivant :

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

où $x \in \mathbb{R}^n$ est l'état du modèle-hypothèse. Les prédicteurs considérés sont également optimaux pour des modèles-hypothèse avec bruit de sortie. Si le bruit affecte l'état, il est impossible de mettre en œuvre le prédicteur d'état non bouclé associé, puisque les variables d'état ne sont pas mesurées (elles sont même inconnues). Mais, comme nous venons de le préciser, notre but est ici d'améliorer la modélisation de la part déterministe du comportement du processus. Les prédicteurs neuronaux utilisés sont donc de la forme :

$$\begin{cases} x(k+1) = \varphi_{RN}\left(x(k),\, u(k)\right) \\ y(k+1) = \psi_{RN}\left(x(k),\, u(k)\right) \end{cases}$$

où $x \in \mathbb{R}^n$ est l'état du prédicteur. Ces prédicteurs sont bouclés, et l'apprentissage semi-dirigé. Nous avons effectué l'apprentissage de tels prédicteurs pour n=2 et n=3. Le meilleur prédicteur obtenu est du second ordre et possède deux neurones cachés à tangente hyperbolique complètement connectés ; les neurones d'état et le neurone de sortie sont linéaires, et ne sont pas connectés entre eux, ni à l'entrée constante (le prédicteur possède 24 coefficients ajustables). Ce prédicteur est représenté sur la figure 25.

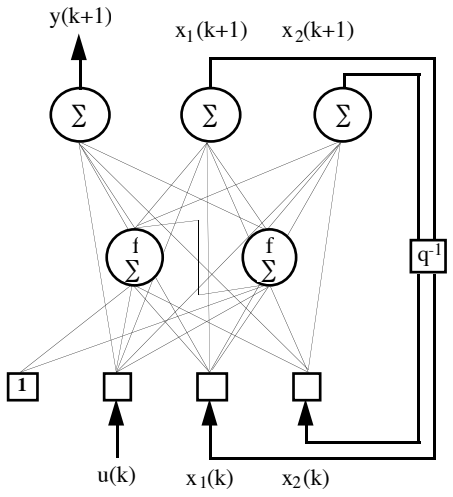
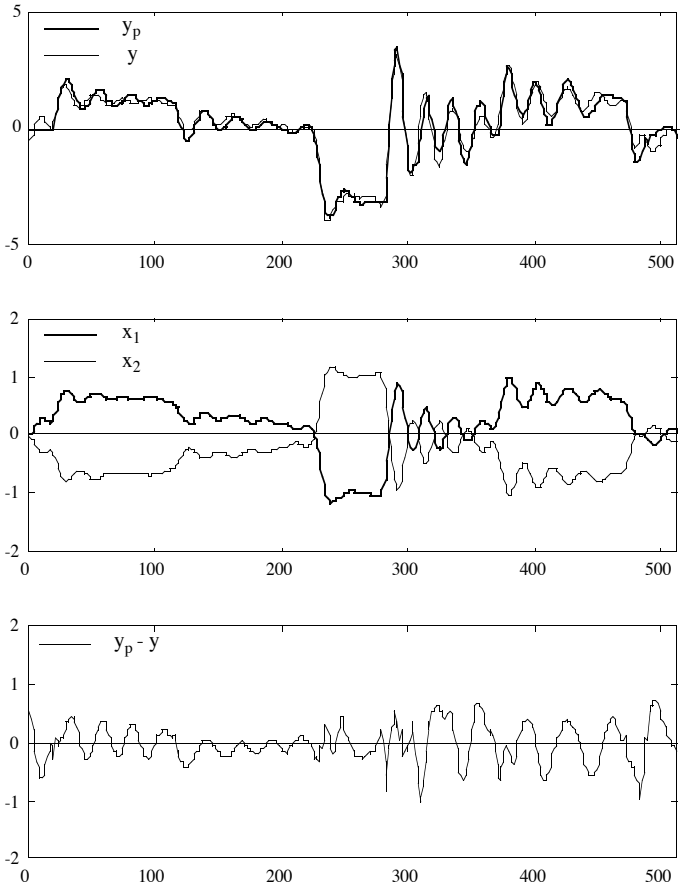
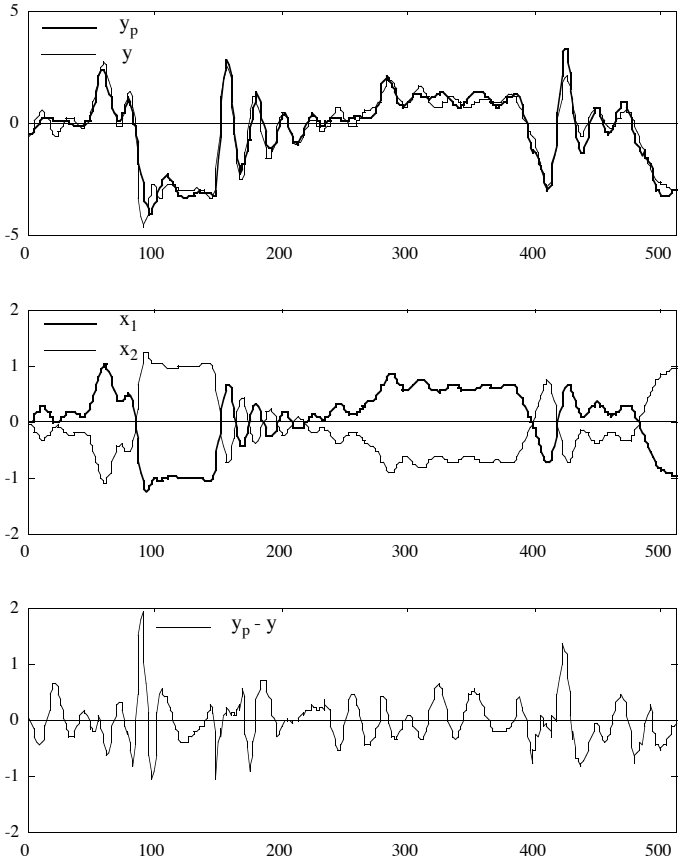


Figure 25.
Prédicteur d'état.



a) Séquence d'apprentissage : EQMA = 0,100.



b) Séquence de test : EQMT = 0,156.

Figure 26.
Performance du prédicteur d'état bouclé.

Les résultats obtenus avec le prédicteur d'état sont représentés sur la figure 26. Ils sont excellents : l'EQMA et l'EQMT sont respectivement de 0,100 et 0,156. Comme en entrée-sortie, le passage à 3 neurones cachés (ou plus) diminue l'EQMA, mais augmente l'EQMT.

La figure 27 montre l'évolution de l'EQMA et de l'EQMT au cours de l'apprentissage. L'apprentissage utilise une méthode de gradient à pas asservi pendant 200 itérations, puis une méthode quasi newtonnienne, jusqu'à ce qu'un minimum soit atteint (cf. chapitre 3 §I.2).

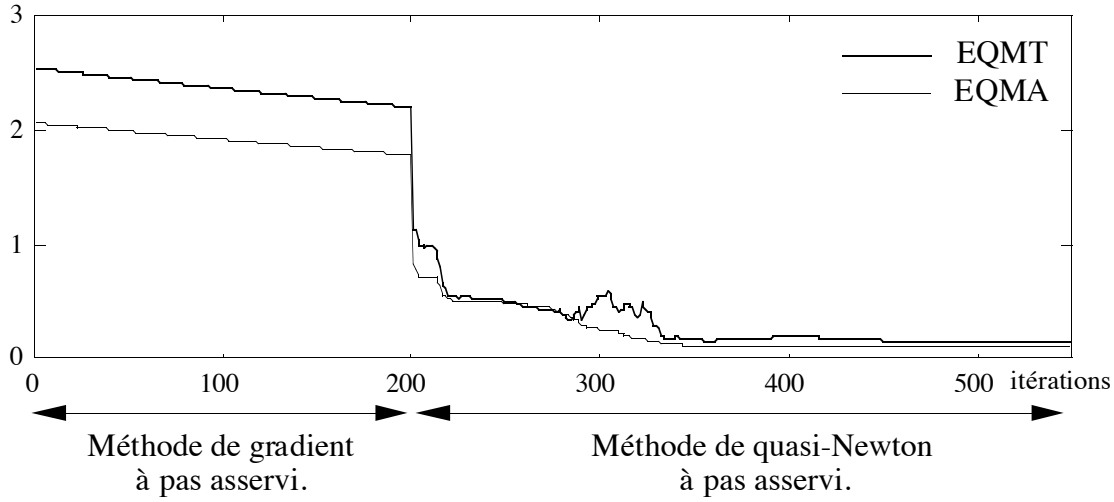


Figure 27.
Évolution de l'EQMA et de l'EQMT au cours de l'apprentissage.

On observe que l'EQMT passe par un maximum (pour 300 itérations) avant d'atteindre le minimum final (550 itérations). Cet exemple montre donc que, d'une manière générale, il n'est pas fondé d'arrêter l'apprentissage dès que l'EQMT se met à croître.

III.4. INTERPRÉTATION ET COMPARAISONS.

Les résultats obtenus à l'aide des meilleurs prédicteurs candidats sont rassemblés sur le tableau 19.

	EQMA	EQMT
Prédicteur entrée-sortie bouclé n=2, m=1, 2 neurones cachés (15 coefficients ajustables)	0,085	0,305
Prédicteur d'état bouclé n=2, 2 neurones cachés (24 coefficients ajustables)	0,100	0,156

Tableau 19.
Récapitulation des performances des meilleurs prédicteurs candidats.

Interprétation.

On observe qu'un prédicteur d'état d'ordre 2 à deux neurones cachés conduit à la meilleure performance, et qu'aucun prédicteur entrée-sortie n'approche cette performance. Tentons d'interpréter ceci.

Supposons qu'il existe un modèle d'état d'ordre 2 décrivant bien le processus. Nous avons vu au chapitre 2 §I.2.2.3, que le comportement d'un modèle d'état peut ne pas être réalisable par un modèle entrée-sortie possédant les mêmes entrées. Si l'on est dans les conditions énoncées dans ce paragraphe, pour un modèle d'état d'ordre n, 2n+1 valeurs précédentes de la sortie et 2n+1 valeurs précédentes de la commande peuvent être nécessaires (soit en tout 10 entrées pour le réseau de neurones si n=2). Or si l'on augmente le nombre d'entrées, on augmente nécessairement aussi le nombre de coefficients du réseau et donc le risque de surajustement. Ceci est d'autant plus vrai que les séquences d'apprentissage et de test contiennent peu de points (ici 512). Nous avons effectivement observé le phénomène de surajustement en ajoutant des entrées supplémentaires (EQMA continue à diminuer tandis que EQMT augmente). Ceci explique que la performance du prédicteur d'état, qui possède peu d'entrées et donc peu de coefficients ajustables, ne puisse être égalée par des réseaux prédicteurs entrée-sortie.

Rappelons nous néanmoins les observations du §III.1 au sujet des séquences d'apprentissage et de test, qui n'explorent pas le même domaine de fonctionnement : toute interprétation en termes de surajustement doit donc être nuancée.

Comparaisons avec les résultats d'autres équipes.

[SJÖ94] réalise l'apprentissage dirigé d'un prédicteur NARX avec n=3, m=2 et 10 neurones cachés. Le test est effectué avec le prédicteur obtenu bouclé (modèle de simulation associé au prédicteur), et fournit un EQMT de 0,585. Une performance analogue est obtenue dans les mêmes conditions dans [BEN94] à l'aide d'un modèle à ondelettes.

[SJÖ93] réalise un EQMT de 0,465 à l'aide du modèle de simulation associé à un prédicteur NARX avec n=2, m=2 et 8 neurones cachés.

Enfin, P.-Y. Glorennec (communication privée) obtient un EQMT de 0,196 à l'aide d'un modèle de simulation " neuroflou " avec n=3, m=2.

Les résultats que nous avons obtenus à l'aide d'un prédicteur d'état à deux neurones cachés, bien meilleurs que les précédents (EQMT de 0,1562), démontrent donc l'intérêt de ces prédicteurs très généraux, ainsi que la faisabilité de leur apprentissage.

CONCLUSION.

Ce chapitre illustre la mise en œuvre pratique de réseaux de neurones pour la modélisation de processus, pour tous les types de modèles-hypothèse présentés au chapitre 2, avec la méthodologie (choix des séquences d'apprentissage et de test, algorithme d'apprentissage) exposée au chapitre 3.

L'exemple du processus entrée-sortie a mis en évidence l'importance du choix du prédicteur et de l'algorithme d'apprentissage pour la modélisation d'un processus avec bruit. Dans le cas où le choix effectué était correct, nous avons modélisé avec succès un processus de type NARMAX fortement non linéaire. Nous avons aussi expliqué quantitativement, dans les cas plus simples NARX et NBSX, l'influence d'une hypothèse fausse sur l'autocorrélation de l'erreur.

L'exemple du processus d'état a mis en lumière la capacité des réseaux prédicteurs d'état à modéliser de façon parcimonieuse les processus les plus généraux. En effet, même si l'existence d'une représentation entrée-sortie est assurée, un prédicteur entrée-sortie est souvent beaucoup moins économe (possède plus de coefficients) qu'un prédicteur d'état réalisant la même performance. Enfin, dans le but d'utiliser le prédicteur pour la commande du processus, nous avons réalisé la modélisation du processus d'état avec bruit à l'aide d'un prédicteur entrée-sortie.

De surcroît, la modélisation d'un processus réel, un actionneur hydraulique, avec d'excellentes performances, confirme l'intérêt pratique des prédicteurs d'état.

Chapitre 5

COMMANDE DE PROCESSUS PAR RÉSEAUX DE NEURONES

INTRODUCTION.

Ce chapitre, consacré à la commande de processus par réseaux de neurones, présente des systèmes de commande *non adaptatifs*, c'est-à-dire dont les paramètres sont fixés lors d'une phase de synthèse préalable à leur utilisation. Dans le cas d'un système de commande adaptatif, ces paramètres seraient ajustés en permanence pendant l'utilisation du système. Alors qu'un système de commande adaptatif peut être de type direct, c'est-à-dire ne pas utiliser de modèle du processus pour estimer les paramètres de l'organe de commande [MIL91] [SLO93], ou indirect [NAR92] [NER93], un système de commande *non adaptatif* est nécessairement de type *indirect* [LEV93]. Tous les systèmes de commande présentés ici utilisent ainsi un modèle du processus pour l'apprentissage du correcteur intervenant dans le système de commande, comme le montre la figure 1.

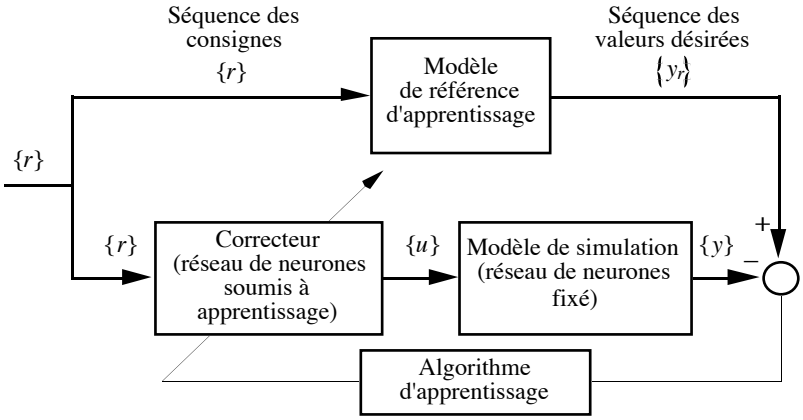


Figure 1.
Système d'apprentissage d'un correcteur neuronal.

Le modèle utilisé pour l'apprentissage est un modèle de simulation du processus non perturbé : les correcteurs ainsi synthétisés ne tiennent donc pas explicitement compte de la nature des perturbations aléatoires non mesurées qui affectent le processus pendant la phase d'utilisation. Comme nous l'avons montré, il est nécessaire pour l'identification de tenir compte du caractère éventuellement bruité du processus ; mais si les résultats obtenus à l'issue de l'identification mettent en évidence que la composante *aléatoire* de l'erreur d'identification est négligeable devant sa composante *déterministe*, ce qui est souvent le cas, il est préférable d'utiliser des méthodes de commande axées sur la robustesse des propriétés (performance et stabilité) du système de commande par rapport à des défauts de modélisation et des déterministes. Nous traitons néanmoins le problème de la commande à

variance minimale pour des processus NARX et NBSX au chapitre 6 ; les principes de ces méthodes ne peuvent être étendus dans ce mémoire à tous les modèles bruités présentés aux chapitres 2 et 3[1].

Dans le cadre qui vient d'être défini (commande non adaptative, indirecte, déterministe), nous traitons les deux problèmes de commande suivants.

* Le premier problème est celui de la *régulation de l'état* d'un processus autour d'un point d'équilibre[2], dans le cas où l'état est mesuré[3]. On disposera d'un modèle de simulation du processus de la forme :

$$x(k) = f\left(x(k\text{-}1), u(k\text{-}1)\right)$$

où $u \in \mathbb{R}^{n_u}$ est l'entrée de commande, $x \in \mathbb{R}^{n_x}$ est l'état, et f est une fonction non linéaire, par exemple réalisée par un réseau de neurones préalablement modélisé.

* Le second problème est celui de la *poursuite* et de la *régulation de la sortie* d'un processus, c'est-à-dire de son asservissement sur un signal de consigne variable dans le temps. On disposera alors :

- soit d'un modèle entrée-sortie du processus :

$$y(k) = h\left(y(k\text{-}1), \ldots, y(k\text{-}n), u(k\text{-}1), \ldots, u(k\text{-}m)\right)$$

où $u \in \mathbb{R}^{n_u}$ est la commande et $y \in \mathbb{R}^{n_y}$ est la sortie ; h est une fonction non linéaire réalisée par exemple par un réseau de neurones.

- soit d'un modèle d'état du processus :

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

où $u \in \mathbb{R}^{n_u}$ est la commande, $x \in \mathbb{R}^{n_x}$ est l'état, et $y \in \mathbb{R}^{n_y}$ est la sortie ; f et g sont des fonctions non linéaires réalisées par un ou des réseaux de neurones, par exemple. L'état du processus est mesuré.

Pour simplifier la présentation, nous prenons $n_u=n_y=1$, et $n_x=n$ (mono-entrée/mono-sortie).

Remarque 1.

Dans le cas d'un modèle non linéaire, il est parfois possible d'asservir un nombre de variables d'état supérieur à la dimension du vecteur de commande ; par exemple, on peut asservir la posture (la position [x, y] et l'orientation ψ) d'un robot mobile non holonome à l'aide de deux commandes seulement, par exemple une commande en vitesse longitudinale et une commande en vitesse angulaire [SAM90]. Mais cette possibilité est l'exception. Ainsi, dans le cas d'un modèle mono-entrée/mono-sortie linéaire avec un état de dimension n>1, seule la régulation de l'état est envisageable, et non

---

[1] Le cas d'un modèle linéaire ARMAX est traité dans [AST84] (p. 285-299), et dans [GOO84] (chapitres 7, 8, 9 et10). Un développement étendu au cas NARMAX exigerait de traiter l'apprentissage de prédicteurs optimaux à t+d, où d est le retard du système, puis la synthèse de la commande à variance minimale de l'erreur de commande à t+d.

[2] On suppose que le processus possède au moins un point d'équilibre. L'état x_0 est un point d'équilibre s'il existe u_0 telle que : $f(x_0, u_0) = x_0$. Par un changement de variables approprié, nous ramenons le point d'équilibre à l'origine $[x_0,u_0]=[0,0]$.

[3] En non linéaire, le principe de séparation n'est pas satisfait. On doit donc supposer que l'on a accès à tout l'état. En dépit des problèmes théoriques et pratiques que pose l'utilisation d'observateurs ou de filtres non linéaires avec un système de commande par retour d'état non linéaire, une approche neuronale du problème est proposée dans [LEV92].

l'asservissement de toutes ses composantes. Comme nous nous plaçons dans le cas mono-entrée, nous traitons seulement la régulation de l'état, et non son asservissement[4].

Remarque 2.

Nous ne traitons pas le problème des perturbations mesurées de façon générale. À la différence des chapitres 2 et 3, où u représente à la fois l'entrée de commande et les perturbations mesurées, u représente ici l'entrée de commande seulement. Néanmoins, pour la plupart des systèmes de commande étudiés ici, le problème des perturbations mesurées est abordé au chapitre 8 sur un exemple concret, la commande du véhicule REMI.

Notations.

Les notations utilisées pour les réseaux correcteurs ou régulateurs sont identiques à celles des réseaux prédicteurs. Ainsi :

a) La notation :

$$u(k) = \psi_{RN}\left(x_p(k);\, C\right)$$

désigne un régulateur neuronal par retour d'état statique (non bouclé) où ψ_{RN} est la fonction réalisée par le réseau muni des coefficients C.

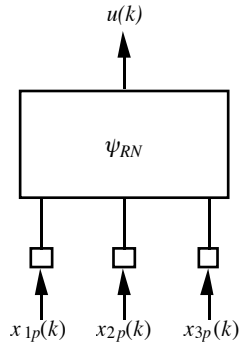


Figure 2.
Exemple de réseau régulateur non bouclé (n=3).

b) La notation :

$$u(k) = \varphi_{RN}\left(y_r(k+1),\, y_p(k),\, \ldots,\, y_p(k\text{-}n+1),\, u(k\text{-}1),\, \ldots,\, u(k\text{-}m+1);\, C\right)$$

désigne un correcteur bouclé d'ordre m-1, où y_r est la sortie de référence, et où φ_{RN} est la fonction réalisée par le réseau non bouclé de la forme canonique muni des coefficients C. Il s'agit donc d'un réseau bouclé de type entrée-sortie. Ce correcteur sera souvent noté :

$$u(k) = \varphi_{RN}\left(y_r(k+1),\, y_p\{_{k\text{-}n+1}^{k},\, u\{_{k\text{-}m+1}^{k\text{-}1};\, C\right)$$

---

[4] S'il y a autant d'entrées de commande que de variables d'état, on se ramène à un problème de poursuite de la sortie *multivariable*.

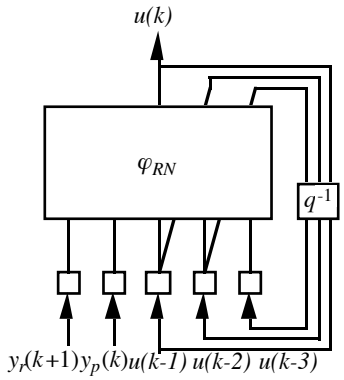


Figure 3.
Exemple de réseau correcteur bouclé (n=1, m=4).

I. RÉGULATION DE L'ÉTAT.

La fonction d'un régulateur consiste à ramener l'état du processus à commander à l'état d'équilibre désiré[5], à partir d'un état initial quelconque, qui peut avoir été imposé délibérément, ou dans lequel le processus peut se trouver à la suite d'une perturbation.

Dans le cas de la régulation, la commande optimale avec coût quadratique à horizon infini se prête bien à l'utilisation de réseaux de neurones. En effet, le but de cette méthode est de trouver une loi de commande minimisant la fonction de coût J suivante :

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

à partir d'un état initial x(0) quelconque, où Q est une matrice de pondération définie positive, et r est un réel positif ou nul. Cette méthode, étendue à des modèles non linéaires, et à l'utilisation d'un réseau de neurones pour réaliser le régulateur, est donc une généralisation *non linéaire* de la commande LQ (Linéaire Quadratique). Elle a l'intérêt de réaliser des lois de commandes stabilisantes, à l'aide de paramètres (Q et r) dont le choix a un effet sélectif sur les réponses et les commandes (même s'il est difficile de prévoir quantitativement cet effet pour un modèle quelconque). De plus, elle conduit à des systèmes de commande dont la stabilité est robuste.

Cas d'un modèle linéaire du processus.

Dans le cas d'un modèle linéaire du processus, l'état du modèle obéit à l'équation :

$$x(k+1) = A\, x(k) + B\, u(k)$$

où A est une matrice n x n et B une matrice colonne n x 1. La paire (A,B) est supposée stabilisable. Pour un tel modèle, il existe une loi de commande optimale par retour d'état linéaire de la forme :

$$u(k) = L\, x(k)$$

où L est une matrice ligne 1 x n. Ce régulateur est appelé régulateur linéaire quadratique (RLQ). Le gain de ce régulateur par retour d'état est constant. Il s'écrit :

---

5 Rappel : l'état d'équilibre désiré est ici : $[x_0,u_0]=[0,0]$.

$$L = \frac{1}{r - B^T K B} B^T K A$$

où K est une matrice n x n, solution de l'équation algébrique discrète de Riccati :

$$K = -Q + A^T K A + \frac{1}{r - B^T K B} A^T K B B^T K A$$

(voir [BOR90], par exemple). Les systèmes de commande LQR sont intéressants en raison de la robustesse de leur stabilité. Cependant, si la marge de gain d'un tel système en temps continu est infinie, elle est finie pour un système discret, et dépend des pondérations Q et r [AST84].

Cas d'un modèle non linéaire du processus.

Que le modèle soit linéaire ou non, le principe d'optimalité de Bellman stipule que la commande optimale au sens du critère ci-dessus ne dépend que de l'état, c'est-à-dire qu'il existe une fonction ρ telle que :

$$u_{opt}(k) = \rho\left(x(k)\right)$$

Ce principe ne fournit pas d'information sur la nature de la fonction ρ ; il établit seulement son existence. Il peut par exemple n'exister aucune fonction ρ continue. C'est le cas pour le problème de la régulation (stabilisation) de la posture d'un robot non holonome : la condition nécessaire de Brockett [BRO83] n'étant pas satisfaite, il n'existe pas de fonction ρ *continue* qui stabilise le robot [SAM90] (en non linéaire, commandabilité n'implique pas stabilisabilité). En revanche, on peut exhiber des retours d'état discontinus stabilisants [CAN91].

Si toutefois l'on suppose l'existence d'un retour d'état continu stabilisant, alors il existe aussi un réseau de neurones non bouclé tel que :

$$u_{opt}(k) = \psi_{RN}\left(x(k);\, C\right)$$

Le système de commande utilisant ce régulateur est représenté sur la figure 4.

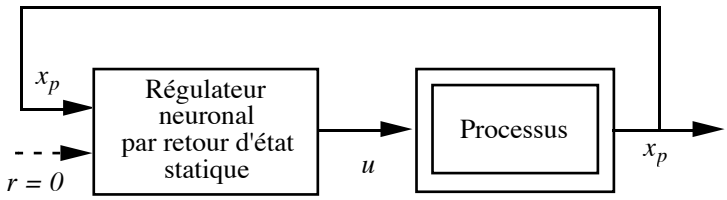


Figure 4.
Système de régulation de l'état autour d'un point d'équilibre par retour d'état statique neuronal.

Nous allons maintenant indiquer comment obtenir un tel régulateur neuronal :

- dans le cas général où aucune solution du problème n'a été établie, par exemple parce que le modèle est trop complexe : nous cherchons un régulateur neuronal optimal ab initio (§I.1).
- dans le cas particulier où le modèle du processus est sous une forme telle qu'il est facile de calculer des trajectoires optimales, mais non pas une loi de commande par retour d'état : nous utilisons ces trajectoires pour réaliser l'apprentissage d'un régulateur neuronal (§I.2).

Nous traitons la détermination complète du système d'apprentissage (modèle de simulation, modèle de référence d'apprentissage, algorithme d'apprentissage) pour les deux cas au §I.3.

I.1. Recherche d'un régulateur optimal ab initio.

Nous formulons la fonction de coût sur un ensemble représentatif de trajectoires. Son expression est la suivante :

$$J = \sum_{x(0)\in X} \sum_{k=1}^{N} x^T(k)\, Q\, x(k) + r\, u(k\text{-}1)^2$$

où x(0), l'état initial du modèle pour une trajectoire donnée, est initialisé aléatoirement dans un domaine borné X. N est un entier, choisi " grand " (N est théoriquement infini).
Le processus est simulé par le modèle, neuronal ou non, supposé continûment stabilisable :

$$x(k) = f\left(x(k\text{-}1),\, u(k\text{-}1)\right)$$

Le régulateur est un réseau de neurones non bouclé réalisant un retour d'état non linéaire (continu) :

$$u(k) = \psi_{RN}\left(x(k);\, C\right)$$

Le problème d'optimisation consiste à calculer les coefficients C de manière à minimiser la fonction de coût sous la contrainte que constitue l'équation du modèle.

Si l'on dispose d'un modèle physique satisfaisant (il n'a pas été nécessaire d'utiliser un réseau de neurones comme modèle), une solution en boucle fermée peut être obtenue par programmation dynamique. Cependant, pour des problèmes de dimension élevée, les calculs et la mémoire nécessaires sont souvents prohibitifs [WHI92] [PLU94]. De plus, le principe de la programmation dynamique exige de quantifier états et commande, quantification d'autant plus grossière que les calculs doivent être moins coûteux. Les réseaux de neurones, en revanche, fournissent une solution continue, et qui demande peu de calculs en temps réel, une fois l'apprentissage du réseau effectué.

Ajoutons que le problème de la régulation neuronale optimale au sens d'une fonction de coût faisant intervenir le temps de façon explicite (commande en temps minimal) ou implicite, avec contraintes terminales, est traité de manière très complète dans [PLU94].

I.2. Recherche d'un régulateur à partir de trajectoires optimales.

Si le modèle du processus permet de calculer facilement une loi de commande *qui ne s'exprime pas sous la forme d'un retour d'état*, par exemple par la méthode du calcul des variations, il peut être intéressant d'approcher, à l'aide d'un réseau de neurones effectuant un retour d'état, les trajectoires optimales obtenues (fonctions du temps par exemple). Dans ce cas la fonction de coût à minimiser est la suivante :

$$J = \sum_{x(0)\in X} \sum_{k=1}^{N} \left(x_a(k) - x(k)\right)^T W \left(x_a(k) - x(k)\right)$$

où les séquences de référence $\{x_a(k)\}$ sont les trajectoires des variables d'état optimales. Il n'y a donc pas lieu de mettre un terme de pondération sur u ; W est une matrice de pondération dont le

choix n'est pas crucial puisque J peut ici être annulée, si le réseau est suffisant. L'intérêt du réseau de neurones obtenu est qu'il fournit une commande en boucle fermée, donc plus robuste que la commande en boucle ouverte initiale vis-à-vis de bruits de mesure et de perturbations d'état, et que son utilisation en temps réel demande beaucoup moins de calculs.

Exemples.

Un exemple de cette démarche dans le domaine de la robotique mobile est présenté dans [RIN93]. L'auteur calcule des solutions numériques (trajectoires optimales) par le calcul des variations pour un problème de planification (trouver la trajectoire optimale pour un robot mobile non holonome reliant toute posture initiale à une posture finale donnée), et pour un problème d'asservissement sur trajectoire (trouver la séquence de commande permettant au robot de rejoindre une trajectoire définie par une ligne droite). Ces trajectoires sont ensuite utilisées pour l'apprentissage de deux réseaux de neurones réalisant un retour d'état (la posture du robot). Nous verrons au §I.3 que le problème de l'apprentissage peut être formulé de deux façons différentes.

Suivant ce principe, il est aussi possible de paramétrer des lois de commandes optimales au sens d'un coût qui n'est pas quadratique. Nous donnons ainsi l'exemple d'un régulateur neuronal paramétrant une loi de commande en temps minimal (régulation du cap d'un véhicule dans [RIV93], reproduit en annexe III du présent mémoire).

I.3. APPRENTISSAGE DES RÉGULATEURS OPTIMAUX.

Nous présentons maintenant les systèmes d'apprentissage pour la synthèse des régulateurs neuronaux des §I.1 et §I.2. Un système d'apprentissage pour la régulation (non adaptative, déterministe) d'un processus est défini par :

- un *modèle de simulation du processus* ;
- le *régulateur neuronal* soumis à apprentissage ;
- un *modèle de référence d'apprentissage* qui calcule les valeurs désirées pour le modèle (séquences d'apprentissage) ;
- un *algorithme d'apprentissage*.

Modèle de simulation.

Le modèle de simulation du processus est neuronal ou non ; il est évidemment toujours bouclé.

$$x(k) = f\left(x(k\text{-}1), u(k\text{-}1)\right)$$

Régulateur neuronal.

C'est un régulateur par retour d'état. La consigne étant nulle, ses arguments sont les variables d'état du modèle. Le régulateur n'est donc pas bouclé. Il est réalisé par un réseau de neurones non bouclé dont les coefficients C sont à estimer :

$$u(k) = \psi_{RN}\left(x(k);\, C\right)$$

Si l'identification du processus a été effectuée dans un domaine borné de valeurs de la commande, ce qui est le plus souvent le cas, la fonction d'activation du neurone de sortie doit être bornée dans les mêmes limites (au delà, le modèle de simulation n'est en effet plus valable).

Modèle de référence d'apprentissage.

Le modèle de référence d'apprentissage fournit la séquence des sorties désirées pour le modèle. Comme la consigne est toujours nulle, sa seule entrée est l'état initial du modèle. Le modèle de référence diffère selon qu'il s'agit de trouver directement un régulateur optimal (§I.1), ou de réaliser une approximation de trajectoires optimales calculées par d'autres moyens (programmation dynamique, calcul des variations), trop coûteux pour être utilisés en temps réel (§I.2).

Algorithme d'apprentissage.

Le choix de l'algorithme, dirigé ou semi-dirigé, dépend aussi du problème considéré.

I.3.1. Recherche d'un régulateur optimal ab initio.

Le modèle de référence d'apprentissage fournit une séquence d'apprentissage qui est toujours nulle, puisque l'état désiré est nul. La fonction de coût s'écrit :

$$J = \sum_{x(0)\in X} \sum_{k=1}^{N} x^T(k)\, Q\, x(k) + r\, u(k\text{-}1)^2$$

où l'entier N est choisi grand.

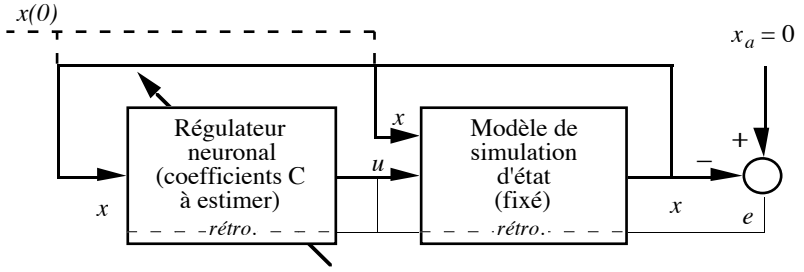


Figure 5.
Système d'apprentissage d'un régulateur optimal ab initio.

Le système d'apprentissage est représenté sur la figure 5. L'algorithme d'apprentissage est dans ce cas nécessairement *semi-dirigé*. En effet, le modèle de référence ne fournit qu'une séquence d'états nuls, qui ne peuvent donc en aucun cas servir à diriger l'état du modèle et le régulateur. Les flèches en pointillés symbolisent l'initialisation des entrées du modèle et du régulateur à l'état x(0).

Choix de la pondération.

Le choix des paramètres de pondération n'est jamais simple ; mais pour en faire une évaluation initiale raisonnable, il est toujours possible de se fonder sur un calibrage physique : si le cahier des charges spécifie les écarts admissibles pour les variables d'état et une amplitude maximale de la commande, on peut fixer les valeurs des éléments diagonaux de la matrice Q à l'inverse du carré de

ces écarts, et le scalaire *r* à l'inverse du carré de la commande maximale [AST84]. Cependant, choisir *r* non nul est surtout justifié s'il n'y a pas de contrainte sur la commande (*r* est le facteur qui confère sa robustesse au système dans le cas linéaire). Avec un réseau de neurones, il est aisé d'imposer une contrainte sur l'amplitude de la commande en choisissant pour le neurone de sortie du régulateur une *fonction d'activation bornée* à la valeur maximale de la commande (tangente hyperbolique, saturation). En ce qui concerne la pondération de l'état (matrice *Q*), on peut s'en tenir à la règle proposée, mais on peut affiner sa valeur en procédant de manière itérative : une valeur particulière de *Q* est fixée après plusieurs essais et l'estimation de la performance correspondante du système d'apprentissage (voir la régulation de la posture du véhicule REMI au chapitre 8).

I.3.2. Recherche d'un régulateur à partir de trajectoires optimales.

Le modèle de référence d'apprentissage est ici particulier : c'est un système reposant sur une méthode classique qui utilise un modèle du processus, et calcule un ensemble de trajectoires optimales, qui constituent les séquences d'apprentissage. La fonction de coût s'écrit :

$$J = \sum_{x(0)\in X} \sum_{k=1}^{N} \left(x_a(k) - x(k)\right)^T W \left(x_a(k) - x(k)\right)$$

Le système d'apprentissage est représenté sur la figure 6. Les flèches en pointillés symbolisent l'initialisation des entrées du modèle et du régulateur à l'état x(0), ainsi que celle du modèle de référence. Il est recommandé d'effectuer l'apprentissage en semi-dirigé, comme dans le cas précédent (§I.3.1). La performance du système d'apprentissage est alors représentative de la performance en phase d'utilisation, l'ensemble modèle-régulateur étant bouclé (en phase d'utilisation, le régulateur est mis en cascade avec le processus). Le modèle et le régulateur pouvant ici être *dirigés* par le modèle de référence, on peut *éventuellement initialiser* l'apprentissage en dirigé (non représenté).

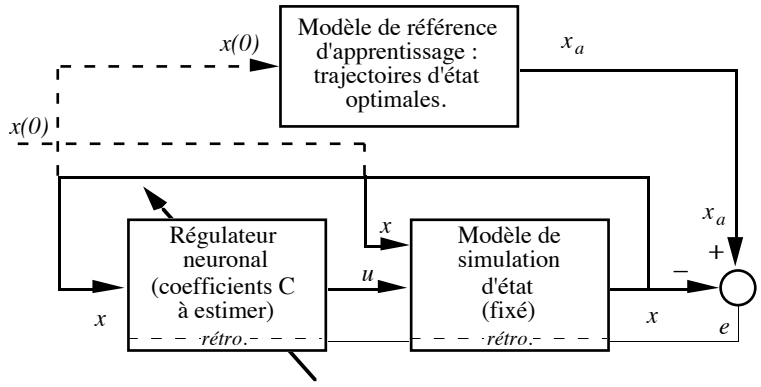


Figure 6.
Système d'apprentissage d'un régulateur optimal à partir de trajectoires optimales.

La méthode classique de commande optimale fournit évidemment, outre les trajectoires optimales, la séquence des commandes optimales $\{u_a(k)\}$: le problème peut donc aussi être formulé comme celui de l'apprentissage d'un régulateur classique existant. La fonction de coût est dans ce cas :

$$J = \sum_{x(0)\in X} \sum_{k=1}^{N} \left(u_a(k) - u(k)\right)^2$$

Le système d'apprentissage correspondant est représenté sur la figure 7. Cette méthode est utilisée par [RIN93] pour l'exemple de paramétrisation cité plus haut (au §I.2). Cependant, le système d'apprentissage de la figure 6 est préférable, car il permet un apprentissage en semi-dirigé, et par là, une meilleure estimation de la performance du système de commande (avec le processus).

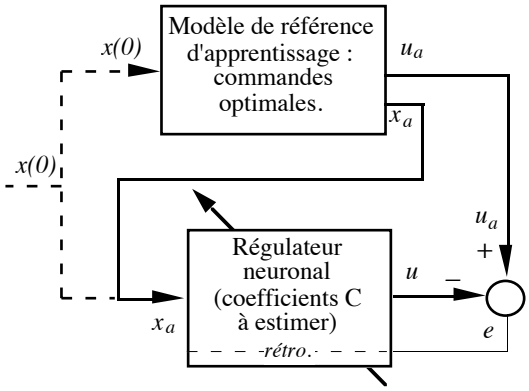


Figure 7.
Système d'apprentissage d'un régulateur optimal à partir d'un régulateur existant.

En outre, on peut, avec le système de la figure 6, utiliser des lois de commandes optimales " approchées ", c'est-à-dire calculées pour un modèle approché du modèle de simulation. Nous avons ainsi utilisé le régulateur en temps minimal conçu avec le modèle linéarisé du véhicule REMI comme modèle de référence pour l'apprentissage d'un régulateur de cap [RIV93].

II. POURSUITE ET RÉGULATION DE LA SORTIE.

Nous traitons maintenant le problème de la poursuite et de la régulation de la sortie d'un processus mono-entrée/mono-sortie, pour lequel on dispose d'un modèle entrée-sortie ou d'un modèle d'état non linéaire, neuronal ou non. Les fonctions de l'organe de commande consistent à imposer le suivi par la sortie du processus d'une consigne variable dans le temps (poursuite), et à compenser les perturbations pour une consigne constante (régulation).

Les organes de commande que nous présentons sont conçus pour imposer au système de commande une dynamique de poursuite de la consigne déterminée explicitement par un modèle de référence. Pour cela, il est nécessaire de tenir compte du retard du processus : en effet, si d est le retard du processus, la commande délivrée par l'organe de commande à l'instant k ne peut lui imposer une sortie ou une dynamique donnée qu'au bout de d pas, c'est-à-dire à l'instant k+d.

Dans le cas d'un modèle entrée-sortie du processus, on fait donc apparaître explicitement le *retard d* du modèle :

$$y(k) = h\left(y(k\text{-}1), \ldots, y(k\text{-}n), u(k\text{-}d), \ldots, u(k\text{-}m)\right)$$

où $u \in \mathbb{R}$ est la commande et $y \in \mathbb{R}$ est la sortie. Si h est une fonction réalisée par un réseau de neurones, le retard d a été mis en évidence lors de la phase de modélisation par la sélection des entrées du modèle, ou par une analyse des coefficients du réseau. On note $m' = m - d$.

Dans le cas d'un modèle d'état du processus (l'état est supposé mesuré), on a :

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

où $u \in \mathbb{R}$ est la commande, $x \in \mathbb{R}^n$ est l'état, et $y \in \mathbb{R}$ est la sortie. L'*ordre relatif* d'un modèle d'état est le nombre de pas d'échantillonnage au bout duquel l'entrée scalaire u influence la sortie scalaire y (cf. annexe II §I.2.1). L'ordre relatif d'un modèle d'état est donc son retard ; il est aussi noté d. Si f et g sont réalisées par un ou des réseaux de neurones, sa valeur, qui est comprise entre 1 et n, est évaluée par une analyse des coefficients du réseau et des simulations.

Nous considérons donc un modèle du processus de retard ou d'ordre relatif d. Le modèle de référence destiné à imposer la dynamique de poursuite de la consigne, qui est ici choisi de type entrée-sortie, linéaire, stable, et de gain statique unité, doit alors posséder le même retard d :

$$E(q)\, y_r(k) = q^{-d}\, H(q)\, r(k)$$

où q^{-1} est l'opérateur retard, et E et H sont deux polynômes d'ordre p tels que :

$$\begin{cases} E(q) = 1 + e_1\, q^{-1} + \ldots + e_p\, q^{-p} \\ H(q) = h_0 + h_1\, q^{-1} + \ldots + h_p\, q^{-p} \ ,\ h_0 \neq 0 \end{cases}$$

En général, l'ordre p du modèle de référence est choisi supérieur ou égal à l'ordre n du modèle du processus, pour ne pas engendrer de commandes d'amplitude trop importante. L'entrée du système de commande est la consigne r(k), sa sortie est celle du processus commandé $y_p(k)$, et l'organe de commande est conçu en fonction du modèle de retard d du processus pour que (voir figure 8) :

$$E(q)\, y_p(k+d) = E(q)\, y_r(k+d) = H(q)\, r(k)$$

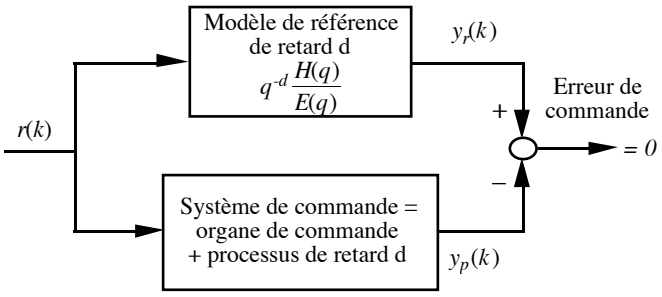


Figure 8.
Schéma de principe de la poursuite de sortie.

Les systèmes de commande que nous proposons sont dits *à un degré de liberté*, c'est-à-dire qu'ils ne permettent pas d'imposer une dynamique de poursuite et une dynamique de régulation *indépendantes*[6]. Parmi des organes de commande imposant tous la même dynamique de poursuite, nous déterminerons lesquels sont les plus performants pour la régulation, et doivent donc être retenus. Nos critères portent essentiellement sur les performances en réponse à des perturbations de

---

[6] Dans le cas contraire, un système de commande est dit à *deux degrés de liberté* [MOR89], et réalise la poursuite et la régulation *à objectifs indépendants* [LAN93].

sortie en créneaux, c'est-à-dire que les meilleurs organes de commande doivent assurer une erreur statique nulle, et une dynamique de régulation aussi satisfaisante que possible, sans à-coups ni oscillations de la sortie ou de la commande.

Fonctionnement nominal/non nominal.

Lorsqu'un système de commande est synthétisé à partir d'un modèle du processus, on parle de fonctionnement nominal si le modèle de simulation utilisé pour sa synthèse décrit exactement le processus. Si le modèle s'écarte de celui-ci, on parle de fonctionnement non nominal. La conception d'un système de commande est ainsi guidée par deux objectifs :

- garantir des propriétés données en fonctionnement nominal : stabilité, niveau de performance en poursuite et en régulation ;
- maintenir ces propriétés en fonctionnement non nominal (une condition nécessaire étant le maintien de la stabilité). Dans le cas d'un organe de commande neuronal, il faut également envisager le cas d'un défaut éventuel de cet organe, dû à l'apprentissage.

Un système de commande satisfaisant également le deuxième objectif est dit *robuste*.

Nous nous intéressons à deux familles de systèmes de commande :

- ***les systèmes de commande par simple bouclage (SCSB)*** : dans le cadre de notre travail, ces systèmes vont seulement garantir stabilité et niveau de performance *pour le système nominal*. La figure 9 montre un SCSB dont l'organe de commande est constitué d'un correcteur.

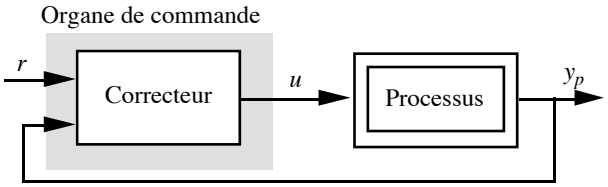


Figure 9.
Exemple de SCSB.

- ***les systèmes de commande avec modèle interne (SCMI)*** : l'organe de commande comprend un modèle interne (MI), qui est un modèle explicite de simulation du processus (le calcul de sa sortie est effectué à chaque instant pour celui de la commande). La figure 10 présente l'architecture de base d'un SCMI dont l'organe de commande est constitué d'un correcteur et du MI.

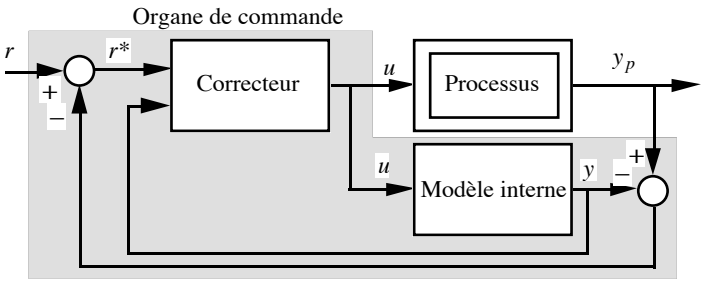


Figure 10.
Exemple de SCMI.

Comme nous le verrons, la conception d'un système de commande *robuste* est plus facile avec un modèle interne qu'en simple bouclage. Les travaux de Morari et Zafiriou [MOR89] ont suscité un vif intérêt chez les automaticiens " classiques ", qui ont développé autour du schéma de principe de la CMI de nombreuses variantes [RIC91], par exemple avec l'utilisation d'un modèle inverse comme correcteur [ABU89]. Les automaticiens utilisant des réseaux de neurones se sont à leur tour intéressés à la CMI avec de nouvelles variantes, dans le but d'apporter une contribution à l'efficacité de ces systèmes de commande, dans le cas de processus non linéaires, par l'utilisation de modèles et de correcteurs non linéaires [HUN92] [SBA93]. Nous présentons ces systèmes de commande en faisant ressortir leurs propriétés, leurs parentés entre eux et avec les SCSB. Enfin, nous déterminons ceux qui se prêtent le mieux à l'utilisation de réseaux de neurones.

Les SCSB et les SCMI que nous utilisons sont fondés sur deux correcteurs, le *correcteur-S* et le *correcteur-D*, correcteurs spécifiques des systèmes à temps discret (§II.1). Nous définissons ensuite (§II.2) les systèmes d'apprentissage des correcteurs-S et -D (modèle de simulation, modèle de référence et algorithme d'apprentissage). Nous présentons tout d'abord leur utilisation au sein de SCSB (§II.3), puis de SCMI (§II.4). Nous déterminons les systèmes de commande les plus performants en nous appuyant sur les résultats rassemblés dans l'annexe II, qui expose les propriétés des SCSB et des SCMI utilisant les correcteurs-S et -D *dans le cas linéaire*.

II.1. CORRECTEURS-S ET -D THÉORIQUES.

Ces deux correcteurs sont les briques de base des SCSB et SCMI étudiés. Nous donnons leur définition, puis leur expression théorique en fonction du modèle avec lequel ils sont conçus (modèle entrée-sortie ou modèle d'état).

II.1.1. Correcteur-S théorique.

Le correcteur-S impose une Sortie de référence au système de commande par simple bouclage. Soit une trajectoire de référence $\{y_r(k)\}$; le correcteur-S délivre à chaque instant k une commande telle que, quel que soit l'état du modèle à cet instant, et si aucune perturbation n'intervient ensuite, *la sortie du système de commande est égale à la sortie de référence à partir de l'instant k+d* :

$$y(k') = y_r(k') \qquad \forall k' \geq k+d$$

Le correcteur-S est appelé " one-step ahead controller " [GOO84]. Dans la littérature neuronale, le terme de " modèle inverse ", ou de " correcteur inverse ", est fréquemment employé, sans définition précise ; il s'agit en fait, le plus souvent, d'un correcteur-S. Le schéma de principe d'un correcteur-S est représenté sur la figure 11.

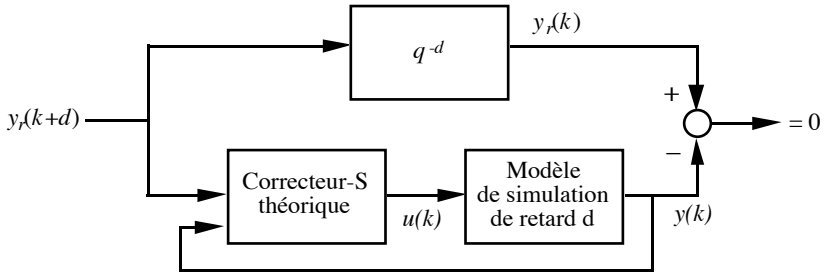


Figure 11.
Définition du correcteur-S théorique.

Exemple non linéaire.

Ce type de correcteur n'est pas particulier aux modèles linéaires, même si les modèles non linéaires soulèvent quelques difficultés supplémentaires. Reprenons l'exemple utilisé par Goodwin [GOO84] en guise d'illustration ; considérons le modèle bilinéaire mono-entrée/mono-sortie :

$$y(k+1) = a\, y(k) + b\, u(k) + n\, y(k)\, u(k)$$

L'expression du correcteur-S est :

$$u(k) = \frac{y_r(k+1) - a\, y(k)}{b + n\, y(k)}$$

Cette loi n'est bien sûr pas applicable au point singulier : y(k) = – b/n.

Le but des paragraphes suivants est, à partir des conditions d'existence d'un correcteur-S stable et de son expression dans le cas d'un modèle linéaire (établies dans l'annexe II), de déduire les arguments du réseau de neurones dont il faut réaliser l'apprentissage pour réaliser le correcteur-S théorique, s'il existe, dans le cas non linéaire. Ces arguments diffèrent suivant que le modèle est sous forme entrée-sortie ou représentation d'état.

II.1.1.1. Cas d'un modèle entrée-sortie.

Cas d'un modèle linéaire du processus[7].

L'expression du correcteur-S théorique est établie dans l'annexe II §I.1.1 à partir de l'expression du prédicteur à d pas de la sortie du modèle :

$$u(k) = y_r(k+d) - G(q)\, y(k) + \left(1 - F(q)\, B'(q)\right) u(k)$$

où F et G sont les polynômes de degrés d-1 et n-1 satisfaisant l'égalité : $1 = F(q)A(q) + q^{-d}\, G(q)$.
Si $m>1$, ce correcteur est bouclé, et il est stable si les racines du polynômes $B'(q)$ sont à l'intérieur du cercle unité, c'est-à-dire si le modèle est à inverse stable (à minimum de phase).

---

[7] Les notations du présent chapitre sont communes avec celles de l'annexe II. Nous considérons le modèle discret linéaire entrée-sortie de retard d suivant : $A(q)\, y(k) = B(q)\, u(k)$, avec :

$$\begin{cases} A(q) = 1 + a_1\, q^{-1} + \ldots + a_n\, q^{-n} \\ B(q) = q^{-d}\, B'(q) = q^{-d}\left(b_0 + b_1\, q^{-1} + \ldots + b_{m'}\, q^{-m'}\right) \end{cases} , \; d+m'=m$$

Cas d'un modèle non linéaire du processus.

D'une manière générale, il existe un prédicteur théorique à d pas de la forme :

$$y(k+d) = \varphi\left(y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-m+1)\right)$$

Pour un modèle linéaire (φ linéaire), nous venons de voir qu'il est toujours possible d'exprimer u(k) en fonction des autres arguments de la fonction φ et de la sortie de référence $y_r(k+d)$.

En non linéaire, ceci n'est pas nécessairement possible. Dans le cas particulier d'un modèle neuronal, il est difficile de savoir si la fonction φ possède une inverse (au moins) dans le domaine de validité du modèle. On peut néanmoins faire l'hypothèse qu'il existe une inverse théorique κ :

$$u(k) = \kappa\left(y_r(k+d), y\{_{k-n+1}^{k}, u\{_{k-m+1}^{k-1}\right)$$

et l'estimer par un apprentissage. Le système d'apprentissage pour la réalisation du correcteur-S théorique par un réseau de neurones est décrit au §II.2.1.

II.1.1.2. Cas d'un modèle d'état.

Cas d'un modèle linéaire du processus[8].

L'expression du correcteur-S dans le cas d'un modèle d'état linéaire du processus est établie dans l'annexe II §I.2.1 :

$$u(k) = \frac{1}{CA^{d-1}B}\left(y_r(k+d) - CA^d\, x(k)\right)$$

Ce correcteur n'est pas bouclé, sa sortie n'est fonction que du signal de référence et de l'état du modèle. Pour que la commande soit applicable (bornée), il est nécessaire que la matrice :

$$A - \frac{BCA^d}{CA^{d-1}B}$$

ait des valeurs propres de module inférieur à 1, c'est-à-dire que le modèle soit à inverse stable.

Cas d'un modèle non linéaire du processus.

Comme dans le cas du modèle entrée-sortie, nous pouvons faire l'hypothèse de l'existence d'un correcteur-S théorique, stable, ayant les mêmes arguments que le correcteur linéaire :

$$u(k) = \kappa\left(y_r(k+d), x(k)\right)$$

On peut donc obtenir une réalisation de ce correcteur à l'aide d'un réseau non bouclé de la forme :

$$u(k) = \psi_{RN}\left(y_r(k+d), x(k); C\right)$$

S'il existe effectivement une fonction κ, et si par ailleurs le système d'apprentissage est adéquat au problème, alors le réseau de neurones pourra réaliser une bonne approximation de κ. Le système d'apprentissage pour la réalisation du correcteur-S théorique par un réseau de neurones est décrit au §II.2.2.

---

[8] Comme dans l'annexe II, nous considérons le modèle d'état linéaire suivant :

$$\begin{cases} x(k+1) = A\, x(k) + B\, u(k) \\ y(k) = C\, x(k) \end{cases}$$

où A est une matrice n x n, B une matrice colonne n x 1, et C une matrice ligne 1 x n. Son ordre relatif est d.

II.1.2. Correcteur-D théorique.

Le correcteur-D impose une Dynamique de référence au système de commande par simple bouclage. Soit une trajectoire de consigne {r(k)}, et une dynamique de référence donnée par le modèle de référence linéaire de retard d : $E(q)\, y_r(k+d) = H(q)\, r(k)$, où E et H sont deux polynômes en q^{-1} ; le correcteur-D délivre à chaque instant k une commande telle que, quel que soit l'état du modèle à cet instant, et si aucune perturbation n'intervient ensuite, *la dynamique du système de commande est égale à la dynamique de référence à partir de l'instant k+d* :

$$E(q)\, y(k') = q^{-d}\, H(q)\, r(k') \qquad \forall k' \geq k+d$$

Ce correcteur est aussi appelé " model-reference controller " [GOO84]. Le schéma de principe de ce correcteur est représenté sur la figure 12.

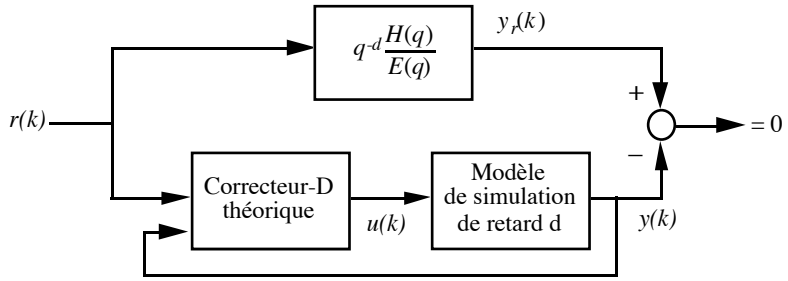


Figure 12.
Définition du correcteur-D théorique.

On constate que le correcteur-S est un cas particulier du correcteur-D, avec $E(q) = H(q) = 1$. Cependant, comme le verrons au §II.2, son utilisation au sein d'un système de commande est très différente de celle du correcteur-D : c'est pourquoi nous traitons ces deux correcteurs séparément.

Exemple non linéaire.

Reprenons l'exemple du modèle bilinéaire. Soit le modèle de référence du premier ordre et de retard 1 suivant :

$$y_r(k+1) = a_r\, y_r(k) + b_r\, r(k)$$

L'expression du correcteur-D est :

$$u(k) = \frac{(a_r - a)\, y(k) + b_r\, r(k)}{b + n\, y(k)}$$

Comme pour le correcteur-S, cette loi n'est pas applicable au point singulier : y(k) = – b/n.

II.1.2.1. Cas d'un modèle entrée-sortie.

Cas d'un modèle linéaire du processus.

L'expression du correcteur-D est établie dans l'annexe II §I.1.2 :

$$u(k) = H(q)\, r(k) - G(q)\, y(k) + \left(1 - F(q)\, B'(q)\right) u(k)$$

où F et G sont les seuls polynômes de degré d-1 et n-1 satisfaisant : $E(q) = F(q)\, A(q) + q^{-d}\, G(q)$. G

est un polynôme en q^{-1} de degré n-1 : G y(k) possède n termes ; FB' est de degré m-1 : FB'u(k) possède m termes. Il en résulte que le correcteur qui fournit la valeur de u(k) réalise une somme pondérée des valeurs y(k), ... , y(k-n+1), u(k-1), ... , u(k-m+1), r(k), ... , r(k-p) ; il est bouclé dès que $m>1$. Pour que ce correcteur soit stable, il faut que le modèle soit à inverse stable.

Cas d'un modèle non linéaire du processus.

On peut faire l'hypothèse qu'il existe un correcteur-D théorique, stable, possédant les mêmes arguments que le correcteur associé à un modèle linéaire, c'est-à-dire de la forme :

$$u(k) = \kappa\left(H(q)\ r(k),\ y\{_{k-n+1}^{k},\ u\{_{k-m+1}^{k-1}\right)$$

Si l'hypothèse est vraie, et si l'apprentissage est effectué correctement, on peut obtenir une réalisation de ce correcteur à l'aide du réseau suivant (en général bouclé) :

$$u(k) = \varphi_{RN}\left(H(q)\ r(k),\ y\{_{k-n+1}^{k},\ u\{_{k-m+1}^{k-1};\ C\right)$$

Le système d'apprentissage pour la réalisation du correcteur-S théorique par un réseau de neurones est décrit au §II.2.1.

II.1.2.2. Cas d'un modèle d'état.

Cas d'un modèle linéaire du processus.

L'expression du correcteur-D est établie dans l'annexe II §I.2.2 :

- si p > d :

$$u(k) = \frac{1}{CA^{d-1}B}\left[H(q)\ r(k) - \left(CA^{d} + e_1\ CA^{d-1} + \ldots + e_d\ C\right)\ x(k) - \left(e_{d+1}\ y(k\text{-}1) + \ldots + e_p\ y(k+d\text{-}p)\right)\right]$$

- si p ≤ d :

$$u(k) = \frac{1}{CA^{d-1}B}\left[H(q)\ r(k) - \left(CA^{d} + e_1\ CA^{d-1} + \ldots + e_p\ C\ A^{d-p}\right)\ x(k)\right]$$

Ce correcteur n'est pas bouclé. Ses arguments sont p valeurs successives du signal de référence (pondérées par le polynôme H), l'état du modèle et, si p>d, p-d valeurs de ses sorties antérieures. Pour que le système de commande soit stable, il faut que le modèle soit à inverse stable.

Cas d'un modèle non linéaire du processus.

On peut encore faire l'hypothèse de l'existence d'un correcteur-D théorique, stable, possédant les mêmes arguments que le correcteur linéaire, soit (si p > d) :

$$u(k) = \kappa\left(H(q)\ r(k),\ x(k),\ y\{_{k+d-p}^{k-1}\right)$$

Le réseau de neurones du système d'apprentissage est donc non bouclé de la forme :

$$u(k) = \psi_{RN}\left(H(q)\ r(k),\ x(k),\ y\{_{k+d-p}^{k-1};\ C\right)$$

Le système d'apprentissage pour la réalisation du correcteur-S théorique par un réseau de neurones est décrit au §II.2.2 suivant.

II.2. APPRENTISSAGE DES CORRECTEURS-S ET -D.

L'objet de ce paragraphe est de présenter les systèmes d'apprentissage des correcteurs-S et -D. Ils sont définis par le modèle de simulation du processus, un correcteur neuronal, un modèle de référence d'apprentissage, et par un algorithme d'apprentissage (voir figures 13 et 14 ci-dessous).

Modèle de référence d'apprentissage.

C'est un modèle de poursuite dont la sortie y_a obéit dans le cas général à :

$$E_a(q)\, y_a(k+d) = H_a(q)\, r(k)$$

- dans le cas de la synthèse d'un correcteur-S, $E_a(q) = H_a(q) = 1$: le modèle de référence d'apprentissage est un retard de d pas ;
- dans le cas de la synthèse d'un correcteur-D, $E_a(q) = E(q)$; $H_a(q) = H(q)$, où E(q) et H(q) sont les polynômes définissant la dynamique de poursuite désirée pour le système de commande.

La fonction de coût à minimiser est donc dans tous les cas (correcteur-S et -D) :

$$J = \sum_{k=1}^{N} \left(y_a(k) - y(k)\right)^2 = \sum_{k=1}^{N} e(k)^2$$

où N est la taille de la séquence d'apprentissage choisie.

Le problème du choix des séquences d'apprentissage et de test ne se pose pas de la même façon que dans le cas de l'identification. En effet, dans la mesure où l'on a la possibilité de choisir des séquences d'apprentissage infiniment riches (puisque l'on travaille avec le modèle et non avec le processus), il n'est pas nécessaire de mettre au point des séquences de test. Cependant, il est recommandé de vérifier que le comportement du système d'apprentissage en réponse à des perturbations du type de celles que l'on souhaite rejeter est correct.

Comme nous l'avons établi aux paragraphes précédents, les arguments du correcteur théorique diffèrent selon que le modèle disponible est de type entrée-sortie ou représentation d'état. Par conséquent, les systèmes d'apprentissage correspondants diffèrent également.

II.2.1. Cas d'un modèle entrée-sortie.

Modèle de simulation.

Le modèle de simulation disponible est de la forme :

$$y(k) = h\left(y(k\text{-}1), \ldots, y(k\text{-}n), u(k\text{-}d), \ldots, u(k\text{-}m)\right)$$

Notons que si, en linéaire, on utilise l'expression du prédicteur à d pas de la sortie du processus pour calculer l'expression du correcteur-S (cf. annexe II §I), le système d'apprentissage neuronal utilise le modèle de simulation qui est un prédicteur à 1 pas pour trouver le correcteur-S. Cet apprentissage ne demande donc pas l'identification du prédicteur à d pas.

Corrrecteur neuronal.

Le système d'apprentissage doit utiliser un correcteur bouclé de la forme :

$$u(k) = \varphi_{RN}\left(H_a(q)\ r(k), y\{_{k-n+1}^{k}, u\{_{k-m+1}^{k-1}; C\right)$$

S'il existe une contrainte sur l'amplitude de la commande, ou pour ne pas sortir du domaine de fonctionnement décrit pendant l'identification du modèle de simulation, la fonction d'activation du neurone de sortie doit être bornée.

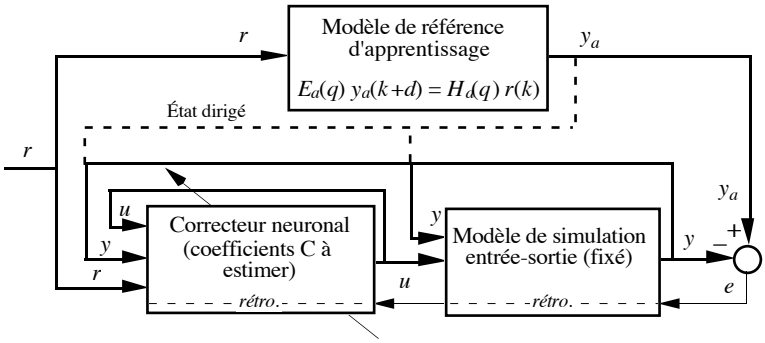


Figure 13.
Système d'apprentissage d'un correcteur-D à l'aide d'un modèle entrée-sortie du processus
(pour un correcteur-S $E_a(q) = H_a(q) = 1$).

Le système d'apprentissage est représenté sur la figure 13. Le caractère bouclé du correcteur impose un algorithme *semi-dirigé* (sauf si m=d). Il est cependant possible de réinitialiser à chaque instant les sorties du modèle et les entrées correspondantes du correcteur avec celles du modèle de référence (flèches pointillées). Il est recommandé de commencer l'apprentissage en dirigeant ainsi le modèle et le correcteur, puis de le poursuivre sans les diriger pour mieux évaluer la performance du futur système de commande. L'apprentissage semi-dirigé est aussi recommandé pour éviter d'obtenir un correcteur instable dans le cas où l'inverse du modèle est instable.

II.2.2. Cas d'un modèle d'état.

Modèle de simulation.

Il est de la forme :

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

Son ordre relatif d a été déterminé par une analyse des poids du réseau et/ou des simulations. De même qu'en entrée-sortie, il n'est pas nécessaire d'utiliser le prédicteur à d pas.

Correcteur neuronal.

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

$$u(k) = \psi_{RN}\left(H_a(q)\ r(k), x(k), y\{_{k+d-p}^{k-1}; C\right)$$

On retrouve le cas particulier du correcteur-D avec p≤d (§II.1.2.2), et celui du correcteur-S pour p=0 (§II.1.1.2). Comme en entrée-sortie, la fonction d'activation du neurone de sortie est souvent bornée.

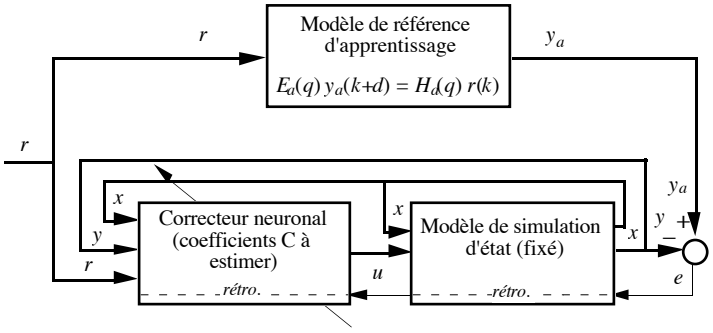


Figure 14.
Système d'apprentissage d'un correcteur -D à l'aide d'un modèle d'état du processus
(pour un correcteur-S $E_a(q) = H_a(q) = 1$).

Le système d'apprentissage est représenté sur la figure 14. L'algorithme d'apprentissage est toujours semi-dirigé, le modèle de référence ne donnant pas de valeurs désirées pour les variables de l'état x.

II.2.3. Conclusion.

Si le processus est à inverse stable, si les séquences d'apprentissage sont bien choisies et si le réseau est de taille suffisante, alors l'apprentissage conduira à un correcteur ayant bien les propriétés du correcteur-S ou -D théorique dans le domaine de fonctionnement exploré pendant l'apprentissage. Bien entendu, dans le cas d'une limitation de l'amplitude de la commande, le correcteur obtenu ne pourra pas être identique au correcteur théorique au voisinage de cette limite. Nous dirons d'un correcteur obtenu par apprentissage qu'il est *parfait* dans un domaine donné, si le correcteur théorique existe dans ce domaine, et si le correcteur obtenu en est une bonne estimation.

Si le processus n'est pas à inverse stable, le correcteur ne peut pas imposer au modèle le suivi du modèle de référence d'apprentissage. Ceci est facilement décelé en analysant les résultats obtenus en fin d'apprentissage. Si néanmoins le correcteur obtenu est stable, et qu'il donne satisfaction en fin d'apprentissage (donc en simple bouclage avec le modèle), on peut tenter de l'utiliser en simple bouclage avec le processus (§II.3). Mais nous verrons qu'il ne peut pas, en règle générale, être utilisé au sein d'un système de commande avec modèle interne (§II.4).

II.3. Systèmes de commande par simple bouclage (SCSB).

Dans ce paragraphe, nous présentons les systèmes de commande par simple bouclage utilisant les correcteurs-S et -D. Nous faisons fréquemment appel aux résultats établis en annexe II en linéaire.

II.3.1. SCSB utilisant un correcteur-S.

La première propriété du système de commande à garantir est, rappelons-le, une dynamique de poursuite donnée par le modèle de référence :

$$E(q)\, y_r(k+d) = H(q)\, r(k)$$

C'est-à-dire que le correcteur doit imposer au système nominal, s'il n'y a pas de perturbation :

$$E(q)\, y_p(k+d) = H(q)\, r(k)$$

Or, par définition, le correcteur-S n'impose pas une dynamique mais une sortie (par opposition au correcteur-D). Par conséquent, sa mise en œuvre nécessite que l'organe de commande qui l'utilise comprenne, en outre, un modèle de référence pour le calcul de la trajectoire de référence à partir de la trajectoire de consigne. La trajectoire de référence peut être calculée au moyen de deux modèles de référence différents (figure 15) :

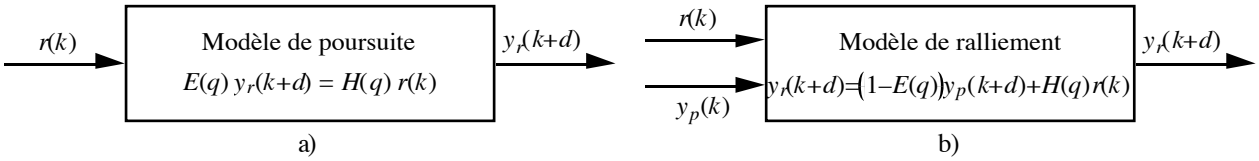


Figure 15.
Modèles de référence.

- *un modèle de poursuite* : un modèle de poursuite pour un signal de consigne est un modèle bouclé (récursif) calculant à l'instant k la sortie de référence $y_r(k+d)$ *à partir de la consigne uniquement*. Son expression est (voir figure 15a) :

$$E(q)\, y_r(k+d) = H(q)\, r(k)$$

Si le correcteur est parfait et s'il n'y a pas de perturbations à partir de l'instant k, le correcteur-S garantit $y_p(k)=y_r(k)$; on a bien dans ces conditions : $E(q)\, y_p(k+d) = H(q)\, r(k)$.

- *un modèle de ralliement* : un modèle de ralliement calcule une trajectoire de ralliement de la consigne *pour un système*, qui peut être un modèle ou le processus par exemple, *à partir de la consigne et des valeurs successives de la sortie de ce système*. C'est un modèle non bouclé (non récursif). Un modèle de ralliement pour le processus de sortie y_p (comme sur la figure 16b), définissant la même dynamique que le modèle de poursuite précédent, calcule à l'instant k la sortie de référence $y_r(k+d)$ de la manière suivante (voir figure 15b) :

$$y_r(k+d) = (1 - E(q))\, y_p(k+d) + H(q)\, r(k)$$

De la même façon, si le correcteur-S est parfait et s'il n'y a pas de perturbation, celui-ci garantit $y_p(k)=y_r(k)$, donc le modèle de ralliement s'écrit : $y_r(k+d) = (1 - E(q))\, y_r(k+d) + H(q)\, r(k)$. Cette expression est équivalente à celle du modèle de poursuite : on a bien $E(q)\, y_p(k+d) = H(q)\, r(k)$.

Cependant, si y_p est perturbé entre k et k+d, seul le modèle de ralliement prend en considération la sortie du processus ; de même, si le correcteur n'est pas parfait, la sortie d'un modèle de ralliement tient compte de cette imperfection, contrairement à celle d'un modèle de poursuite. Nous montrons dans l'annexe II qu'il est en effet beaucoup plus avantageux d'utiliser un modèle de ralliement qu'un modèle de poursuite, tant du point de vue de la stabilité, que du point de vue du comportement de la commande en réponse à une perturbation de sortie en échelon (annexe II §II.1.1.1 et §II.1.1.2). Cependant, l'utilisation d'un modèle de ralliement n'est possible que si le retard d du processus vaut 1, sinon certaines sorties *futures* du processus contenues dans $(1 - E(q))\, y_p(k+d)$ sont nécessaires au calcul, à l'instant k, de $y_r(k+d)$. Il faudrait donc prédire leurs valeurs, ce qui compliquerait considérablement le système de commande.

II.3.1.1. Cas d'un modèle entrée-sortie.

Un correcteur-S obtenu par apprentissage, noté $\varphi_{RN}^{S,\,e\text{-}s}$, est mis en cascade avec le processus :

$$u(k) = \varphi_{RN}^{S,\,e\text{-}s}\left(y_r(k+d),\, y_p\{_{k-n+1}^{k},\, u\{_{k-m+1}^{k-1}\right)$$

a) SCSB utilisant un modèle de poursuite.

Ce SCSB est représenté sur la figure 16a. Nous supposons le correcteur-S parfait. Nous soulignons en annexe II §II.1.1.1 les inconvénients de ce système dans le cas linéaire qui, même en nominal, impose une dynamique de régulation beaucoup trop rapide (pour une perturbation constante, le régime permanent est atteint en n pas, où n est l'ordre du modèle linéaire). En effet, la sortie du modèle de poursuite est la même, que la sortie du processus suive ou non (dans le cas d'une perturbation) la sortie de référence. Ceci peut conduire à un effort excessif sur la commande. De plus, il se produira des oscillations à chaque période d'échantillonnage si le processus possède des " zéros " négatifs. Nous présentons néanmoins ce SCSB, car on le rencontre souvent dans la littérature neuronale, à propos de l'utilisation d'un " modèle inverse ", sans que ces inconvénients soient soulignés [LEV92] [HUN92] [SBA93].

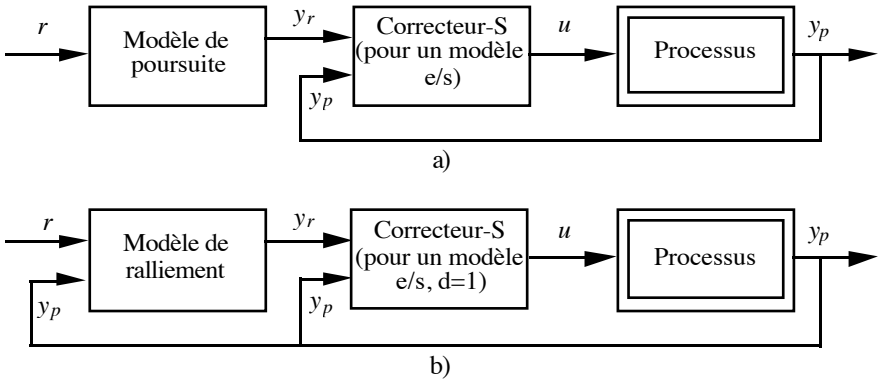


Figure 16.
SCSB avec correcteur-S (pour un modèle entrée-sortie du processus).

b) SCSB utilisant un modèle de ralliement.

Comme indiqué plus haut, ce SCSB n'est concevable que si d=1. Si le correcteur-S est parfait, la réponse du système à une perturbation de sortie est meilleure que celle d'un SCSB avec modèle de poursuite : en effet, la dynamique de régulation imposée par cet organe de commande est la même que la dynamique de poursuite. Le modèle de ralliement filtre les oscillations possibles de la commande dues à une perturbation, et atténue l'effet d'éventuels " zéros " négatifs. Les calculs effectués en annexe II §II.1.1.2 montrent que, dans le cas linéaire, la stabilité de ce système est plus robuste que celle du système précédent. Ce SCSB est représenté sur la figure 16b. Nous verrons au §II.3.2.1 que le SCSB avec correcteur-D possède les mêmes propriétés, quel que soit le retard du modèle.

II.3.1.2. Cas d'un modèle d'état.

Un correcteur-S obtenu par apprentissage, noté $\psi_{RN}^{S,\,état}$, est mis en cascade avec le processus :

$$u(k) = \psi_{RN}^{S,\,état}\left(y_r(k+d),\, x_p(k)\right)$$

Comme dans le cas entrée-sortie, le système de commande utilise soit un modèle de poursuite (figure 17a), soit, si l'ordre relatif d est égal à 1, un modèle de ralliement (figure 17b).

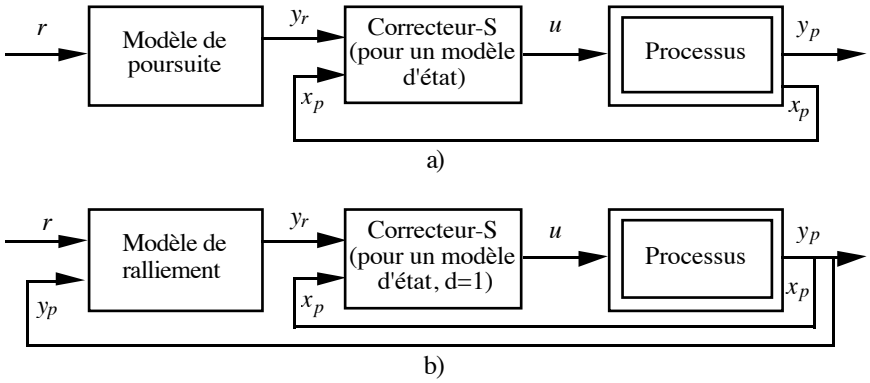


Figure 17.
SCSB avec correcteur-S (pour un modèle d'état du processus).

Si le correcteur est parfait, les inconvénients du modèle de poursuite sont les mêmes que dans le cas entrée-sortie. Comme en entrée-sortie, nous montrons plus loin (§II.3.2.2) que le problème de la dynamique de régulation du système en réponse à une perturbation est résolu avec un correcteur-D quel que soit le retard.

II.3.2. SCSB utilisant un correcteur-D.

Puisque, par définition, un correcteur-D impose une dynamique au processus, un organe de CSB utilisant un correcteur-D ne comprend que le correcteur-D lui-même : il n'est pas nécessaire de lui adjoindre un modèle de poursuite ou un modèle de ralliement.

II.3.2.1. Cas d'un modèle entrée-sortie.

Un correcteur-D obtenu par apprentissage, noté $\varphi_{RN}^{D,\,e\text{-}s}$, est mis en cascade avec le processus :

$$u(k) = \varphi_{RN}^{D,\,e\text{-}s}\left(H(q)\; r(k),\, y_p\{_{k-n+1}^{k},\, u\{_{k-m+1}^{k-1}\right)$$

Le système de commande est représenté sur la figure 18.

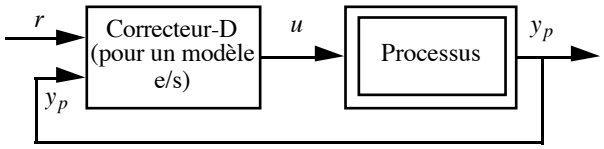


Figure 18.
SCSB avec correcteur-D (pour un modèle entrée-sortie du processus).

Nous montrons dans l'annexe II §II.1.2 que, pour un modèle linéaire et un correcteur parfait, ce système possède les mêmes caractéristiques dynamiques que le système utilisant un correcteur-S et un

modèle de ralliement : il impose une dynamique de régulation identique à la dynamique de poursuite, et sa stabilité est d'autant plus robuste que le modèle de poursuite est lent. Le système avec correcteur-D a l'avantage sur le système avec correcteur-S de pouvoir être utilisé quel que soit le retard du processus. Cependant, dans le cas où d=1, on a intérêt à mettre en œuvre un système utilisant un correcteur-S avec un modèle de ralliement : il est en effet possible de modifier le modèle de ralliement pendant le fonctionnement du système de commande, pour le stabiliser par exemple. En revanche, si l'on utilise un correcteur-D, et si le système de commande ne donne pas satisfaction, il faut procéder à un nouvel apprentissage du correcteur-D.

II.3.2.2. Cas d'un modèle d'état.

Un correcteur-D obtenu par apprentissage, noté $\psi_{RN}^{D,\,état}$, est mis en cascade avec le processus :

$$u(k) = \psi_{RN}^{D,\,état}\left(H(q)\; r(k),\, x_p(k),\, y_p\{_{k+d-p}^{k-1}\right)$$

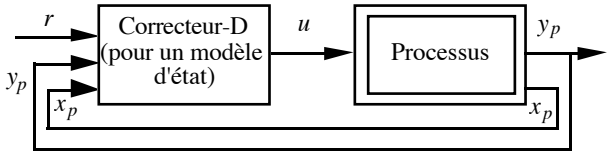


Figure 19.
SCSB avec correcteur-D (pour un modèle d'état du processus).

Si le correcteur-D est parfait, les propriétés de ce système de commande sont les mêmes que dans le cas entrée-sortie.

II.3.3. Conclusions et remarques.

Récapitulation.

Si le modèle est à inverse stable, et que l'apprentissage est réalisé dans de bonnes conditions, le correcteur obtenu est une bonne approximation du correcteur théorique sur le domaine d'apprentissage. Dans le cas nominal (modèle parfait), tous les systèmes de commande par simple bouclage proposés possèdent alors la dynamique de référence comme dynamique de poursuite. Cependant, seuls les systèmes utilisant un correcteur-S et un modèle de ralliement, ou bien un correcteur-D, imposent que la dynamique de régulation soit identique à la dynamique de poursuite, et ont donc un comportement satisfaisant en régulation. De plus, dans le cas non nominal (modèle imparfait), leur stabilité est plus robuste que celle du système utilisant un correcteur-S avec un modèle de poursuite. Enfin, dans le cas où le retard (ou l'ordre relatif) est égal à 1, le système utilisant un correcteur-S et un modèle de ralliement, modifiable sans que cela nécessite de nouvel apprentissage, est préférable au système avec correcteur-D.

Si le processus est à inverse instable, les conclusions précédentes ne sont bien entendu pas valables. Ceci doit être mis en évidence dès l'apprentissage du correcteur. Néanmoins, si le correcteur obtenu est stable et suffisamment performant, il peut être intégré dans un des systèmes de commande

par simple bouclage présentés (cf. §II.2). Sinon, il faut utiliser des systèmes mieux adaptés à des processus à inverse instable, tels que les systèmes de commande prédictive, qui commencent à faire l'objet de nombreux travaux dans le domaine des réseaux de neurones [PSI91] [HUN92] [SBA93] [GRO94].

Inconvénients.

Tous ces systèmes de commande par simple bouclage présentent l'inconvénient de ne pas garantir une erreur statique nulle, même si le modèle est parfait (cas nominal), à inverse stable, et si le correcteur neuronal est aussi parfait : une perturbation constante de sortie n'est pas rejetée (ceci est clairement montré dans le cas linéaire dans l'annexe II §II.1). La performance de ces systèmes n'est donc pas robuste. De plus, il est difficile d'évaluer et d'accroître la robustesse de leur stabilité. Nous allons maintenant présenter des systèmes de commande qui pallient ces deux inconvénients : les systèmes de commande avec modèle interne (SCMI).

Remarque : correcteur " PID " neuronal.

Dans le but d'éliminer l'erreur statique, il est possible d'utiliser un SCSB utilisant un correcteur neuronal non linéaire avec un terme intégral et éventuellement un terme dérivé, ou " NID ". Le NID le plus simple est constitué d'un seul neurone de la forme :

$$u(k) = F_{act}\left(u(k\text{-}1) + c_I\, e\,(k) + c_P\, \Delta e\,(k\text{-}1) + c_D\, \Delta^2 e\,(k\text{-}2)\right)$$

où $e(k) = r(k) - y_p(k)$; $\Delta e(k) = e(k) - e(k\text{-}1)$; $\Delta^2 e(k) = e(k) - 2\, e(k\text{-}1) + e(k\text{-}2)$. ; c_I, c_P et c_D sont les coefficients ajustables du neurone ; F_{act} désigne la fonction d'activation du neurone, typiquement une tangente hyperbolique ou une saturation. Une commande par NID de ce type est proposée dans [SCO92], mais elle n'est comparée qu'à un PID ordinaire, et à un correcteur neuronal par retour d'état utilisé dans un SCSB. Or, pour être vraiment performant, un NID doit avoir une structure telle que la valeur des coefficients proportionnel, intégral et dérivé puisse varier en fonction du point de fonctionnement. Il faut donc que chacun de ces termes soit la sortie d'un réseau de neurones : ceci complique beaucoup la conception (connectivité) et l'apprentissage du réseau. Nous avons mis en œuvre un tel NID pour la commande de la vitesse du véhicule REMI. Il s'est avéré moins performant qu'un SCMI mis au point suivant les principes qui vont être exposés maintenant.

II.4. Systèmes de commande avec modèle interne (SCMI).

II.4.1. Propriétés.

Nous commençons par présenter ces propriétés dans le cas d'un modèle entrée-sortie linéaire du processus. Ces propriétés sont conservées dans le cas d'un modèle non linéaire, mais le choix d'un système de commande neuronal pose des problèmes spécifiques, dus à l'apprentissage, qui sont abordés au §II.4.1.5.

Soient C(z), M(z) et Π(z) les fonctions de transfert en z du correcteur, du MI et du processus. Le schéma de principe d'un SCMI linéaire est représenté sur la figure 20.

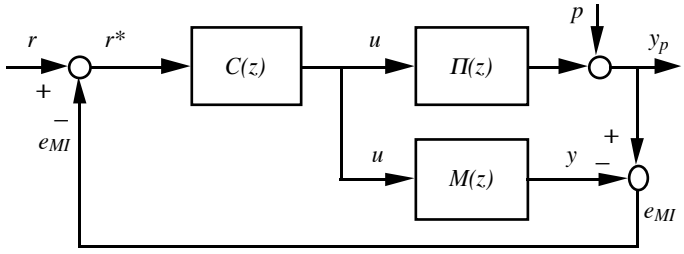


Figure 20.
SCMI linéaire.

La transformée en z du signal de rétroaction e_{MI} a pour expression :

$$E_{MI}(z) = \left(\Pi(z) - M(z)\right) U(z) + P(z)$$

Dans le cas nominal, c'est-à-dire si le modèle est parfait (Π=M), et s'il n'y a pas de perturbation (P=0), alors le signal de rétroaction e_{MI} est nul. Ce dernier exprime le défaut de modélisation et/ou l'effet des perturbations non mesurées.

II.4.1.1. Conception du correcteur dans le cas linéaire.

Dans le cas nominal, et s'il n'y a pas de perturbation, la fonction de transfert du SCMI est égale au produit C(z) M(z). Ceci rend simple la conception du correcteur C avec le modèle, puisque le système n'est alors pas bouclé.

Considérons le SCSB équivalent au SCMI précédent, mettant en œuvre le correcteur C'(z) :

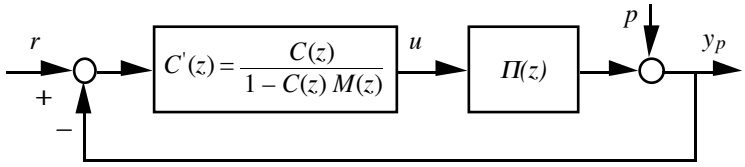


Figure 21.
SCSB équivalent au SCMI de la figure 20.

La synthèse directe de C'(z), qui met en jeu le système bouclé complet, est donc plus complexe que celle de C.

Pour un système de commande neuronal, l'avantage du SCMI se retrouve dans la facilité d'apprentissage du correcteur C, par rapport à celle d'un SCSB offrant les mêmes performances. Si C' réalise une intégration, ce qui sera le plus souvent le cas, nous avons vu que son apprentissage par un réseau de neurones est problématique (cf. NID §II.3.3).

II.4.1.2. Stabilité dans le cas linéaire.

Condition de stabilité du système nominal.

Le système nominal (M=Π) est stable si et seulement si le correcteur *et* le processus sont stables. La structure de CMI présentée ne peut donc être appliquée que si le processus est stable (pour une présentation de SCMI de processus instables, voir [MOR89]). Dans la suite, nous nous plaçons dans cette hypothèse.

Robustesse de la stabilité.

La robustesse de la stabilité est une condition nécessaire. La sortie du processus de la figure 20 (ou 21) a pour expression :

$$Y_p(z) = \frac{C(z)\prod(z)}{1 + C(z)\left(\prod(z) - M(z)\right)} R(z) + \frac{1 - C(z)\,M(z)}{1 + C(z)\left(\prod(z) - M(z)\right)} P(z)$$

Le système reste stable tant que les racines du dénominateur sont à l'intérieur du cercle unité. L'idée de base, largement développée dans [MOR89], consiste à choisir le correcteur C(z) de manière à réduire l'influence de l'écart modèle-processus $\prod(z) - M(z)$ dans le domaine de fréquences où cet écart est le plus important, c'est-à-dire typiquement aux hautes fréquences. Cela signifie que le correcteur C(z) doit contenir un filtre passe-bas.

II.4.1.3. Performance dans le cas linéaire.

La commande s'écrit :

$$U(z) = \frac{C(z)}{1 - C(z)\,M(z)}\left(R(z) - Y_p(z)\right)$$

De cette expression, on déduit que, si le gain statique du correcteur est l'inverse de celui du modèle, il n'y aura *pas d'erreur statique* pour une perturbation ou une consigne constante. En effet, cela signifie que 1 est racine de $1 - C(z)\,M(z)$, et que la fonction de transfert du correcteur peut s'écrire :

$$U(z) = \frac{1}{\left(1 - z^{-1}\right)} \frac{C(z)}{N(z)}\left(R(z) - Y_p(z)\right)$$

avec : $1 - C(z)\,M(z) = \left(1 - z^{-1}\right) N(z)$. L'organe de commande réalise implicitement un intégrateur. De même, un choix adéquat du correcteur permet d'*annuler l'erreur de vitesse* [MOR89].

II.4.1.4. Compromis stabilité-performance dans le cas linéaire.

Le meilleur système de commande possible est celui pour lequel la sortie du modèle est égale à la consigne retardée du retard du modèle, qui utilise donc le correcteur-S (" Perfect Control " [MOR89]). Ce correcteur ne peut donc être utilisé que si le modèle est à inverse stable[9]. Soit le modèle de retard 1 :

---

[9] Dans le cas où le modèle n'est pas à minimum de phase, Morari propose les correcteurs stables donnant une erreur de commande à variance minimale en fonction du type de consignes prévues. Nous verrons que, dans le cas de l'utilisation de réseaux de neurones, il faut supposer que l'inverse du processus est stable.

$$M(z) = \frac{B(z)}{A(z)} = \frac{z^{-1}\, B'(z)}{A(z)}$$

Le correcteur-S a pour expression : $B'(z)\ U(z) = R(z) + z\left(A(z) - 1\right) Y(z)$. Si les valeurs initiales de y sont égales aux valeurs correspondantes de r, on a: $B'(z)\ U(z) = A(z)\ R(z)$. Soit C_S la fonction de transfert du correcteur-S :

$$C_S(z) = \frac{U(z)}{R(z)} = \frac{A(z)}{B'(z)} = \frac{z^{-1}}{M(z)}$$

Le module du produit $C_S(z)\ M(z)$ est toujours égal à 1. Or pour que la stabilité soit robuste, C(z) doit contenir un filtre passe-bas (cf. §II.4.1.2). Comme le processus est en général aussi un filtre passe-bas, il est impossible que, simultanément, le produit des modules soit égal à 1 pour tout z, et que C(z) soit un filtre passe-bas.

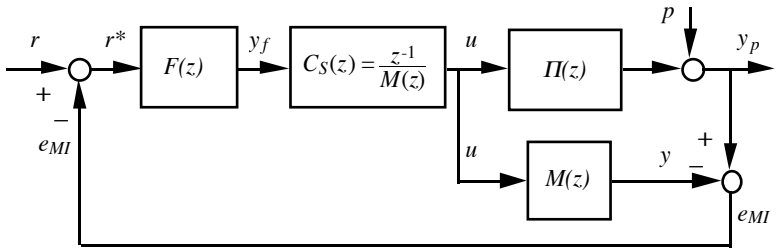


Figure 22.
Compromis stabilité-performance.

On réalise un compromis entre les deux exigences en introduisant un filtre passe-bas de gain statique unité dans l'organe de commande (figure 22). En fonctionnement nominal, sans perturbation, la dynamique du système de commande est celle du filtre :

$$Y_P(z) = z^{-1}\ F(z)\ R(z)$$

Morari montre que, pour assurer la stabilité robuste, quel que soit l'ordre du modèle, il est toujours possible d'utiliser un filtre du premier ordre [MOR89].

II.4.1.5. Application aux réseaux de neurones.

Il est tentant de réaliser des SCMI utilisant des réseaux de neurones, afin d'obtenir une meilleure robustesse de la performance et de la stabilité qu'avec les SCSB. Même si " les méthodes telles que la CMI ont le grand mérite d'offrir un guide méthodologique qui engendre nécessairement la robustesse " [LAR89], la démarche guidant la conception des SCMI ne peut être rigoureusement identique dans les cas neuronal et linéaire. Nous comparons ci-dessous ces deux démarches.*On supposera dans la suite que le modèle est à inverse stable*.

Cas d'un modèle linéaire du processus.

a) Tout d'abord, le correcteur-S $C_S(z)$ est calculé pour assurer une poursuite parfaite dans le cas nominal.

b) Considérant ensuite les écarts possibles entre le modèle et le processus, qu'on peut exprimer en termes d'incertitudes structurées ou non (ces dernières concernent typiquement les réponses fréquentielles), on conçoit le filtre F(z) pour garantir la stabilité robuste.

Cas d'un modèle non linéaire neuronal du processus.

a) L'étape de *calcul* du correcteur-S est remplacée, dans le cas neuronal, par l'*apprentissage* de ce correcteur.

b) Le filtre F(z) de la figure 22 peut être interprété comme le modèle de référence que nous avons utilisé dans les SCSB. Nous montrons dans l'annexe II §II.2.1 que ce modèle de référence doit être un modèle de ralliement pour le MI. Comme dans les SCSB, *l'ordre du filtre doit être choisi supérieur ou égal à celui du modèle*, en linéaire comme en non linéaire. En effet, la stabilité du SCMI peut être garantie par un filtre du premier ordre [MOR89], mais son utilisation risque de provoquer des commandes d'amplitude trop importante si l'ordre du modèle est supérieur à 1. Alors qu'en linéaire, le filtre peut être conçu en fonction des incertitudes sur le modèle, il est difficile d'exprimer des incertitudes sur un modèle neuronal, et plus encore de les exploiter pour la conception du modèle de ralliement. Toutefois, les caractéristiques du modèle, les limitations d'amplitude de la commande, et le cahier des charges qui précise la dynamique de poursuite désirée, orientent le concepteur pour le choix du modèle de ralliement. Mais il peut être nécessaire de restreindre les exigences en poursuite, afin d'assurer la *robustesse de la stabilité*, par exemple en ralentissant le modèle de ralliement. Le meilleur modèle de ralliement est donc obtenu au terme d'une procédure itérative avec le processus. Notons que ceci ne demande pas de nouvel apprentissage (on conserve le correcteur-S pendant toute la procédure).

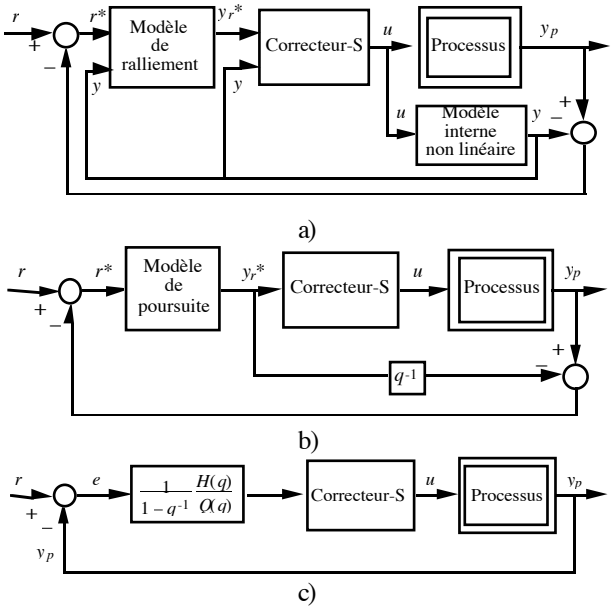


Figure 23.
Performance robuste du SCMI non linéaire avec correcteur-S (équivalences a-b et a-c valables *localement*).

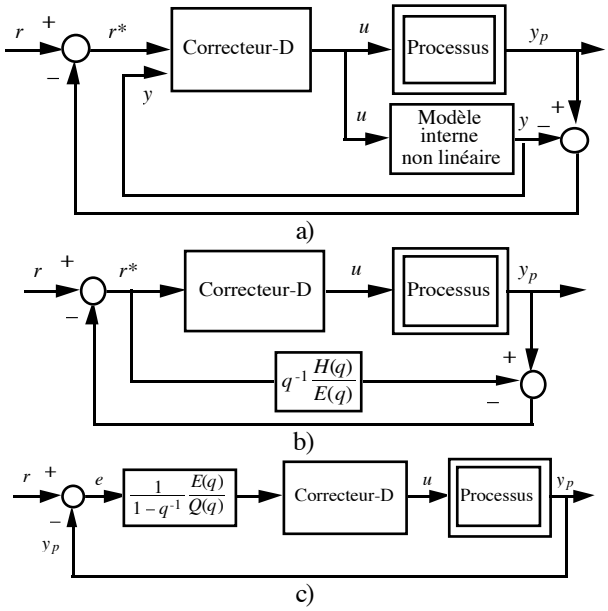


Figure 24.
Performance robuste du SCMI non linéaire avec correcteur-D (équivalences a-b et a-c valables *localement*).

Pour résumer, les SCMI neuronaux possèdent les propriétés suivantes :

- Dans tout le domaine où le correcteur-S théorique existe, et est bien approché par le correcteur neuronal, *la performance est robuste*. En effet, dans ce domaine, les schémas 23a et 23b sont équivalents. Dans le cas nominal, la dynamique de poursuite est égale à la dynamique du modèle de ralliement (le modèle de poursuite de la figure 23b est défini par les mêmes polynômes E(q) et H(q) que le modèle de ralliement de la figure 23a). Comme l'organe de commande réalise implicitement

un intégrateur (Q(q) est le polynôme tel que : $E(q) - q^{-1} H(q) = (1 - q^{-1}) Q(q)$), l'erreur statique est nulle même dans le cas non nominal, et des perturbations de sortie additives constantes sont rejetées. Les schémas 23b et 23c sont toujours équivalents.
- Le modèle de ralliement peut toujours être ajusté (même en fonctionnement) de manière à assurer *la robustesse de la stabilité*.

Nous avons vu qu'il est équivalent d'utiliser un modèle de ralliement avec un correcteur-S, et un correcteur-D imposant la dynamique définie par le modèle de ralliement (§II.3.2). Nous verrons que cette dernière solution doit être utilisée dans certains cas (§II.4.2.2 et §II.4.3.2). Les équivalences de la figure 24 sont alors valables dans le domaine où le correcteur-D est parfait.

Suppression du modèle.

Dans le domaine ou le correcteur neuronal est l'inverse parfait du modèle, les systèmes de commande des figures 23b et 23c sont équivalents au schéma 23a, mais seulement dans ce domaine. Soit par exemple une consigne constante r supérieure à la valeur maximale que peut atteindre la sortie y_p du processus (supposée limitée) : alors que dans le schéma 23a la rétroaction est nulle, il y a dans le schémas 23b une rétroaction non nulle. Le schéma 23c montre que l'erreur $e=r-y_p$ est intégrée par l'organe de commande, ce qui risque de déstabiliser le système. Il est aussi bien connu que, en cas de limitation de l'amplitude de la commande, alors que les SCSB contenant un intégrateur nécessitent des dispositifs de conditionnement de l'intégrateur, dits " anti-wind-up " [AST84], les SCMI sont naturellement stables [MOR93]. Les systèmes simplifiés 23b ou 23c, qui demandent moins de calcul pendant l'utilisation, et qui ne souffrent pas d'éventuels défauts du correcteur dus à l'apprentissage, sont parfois recommandés [PSI91] [GRO94]. Mais l'analyse précédente montre que leur comportement peut être dangereux dès que le correcteur n'est plus l'inverse parfait du modèle, en particulier dès que la commande entre en saturation : ils sont donc *à proscrire*. Il en est de même pour les systèmes des figures 24b et 24c utilisant un correcteur-D. Nous montrons leurs inconvénients au chapitre 6 §I.5.3 à l'aide d'un processus simulé.

Choix des séquences d'apprentissage et domaine de validité du correcteur.

Dans le cas d'un modèle linéaire, ou d'un modèle non linéaire analytique (modèles bilinéaires considérés dans [ABU89] et [CHA90] par exemple), le domaine d'existence du correcteur-S (ou-D) théorique est connu. Le correcteur est valable globalement pour un modèle linéaire ; il en est de même pour les modèles bilinéaires à l'exception de quelques points singuliers (cf. l'exemple utilisé au §II.1). Dans le cas neuronal, *le correcteur appris ne peut approcher le correcteur théorique (s'il existe) que dans le domaine défini par les séquences d'apprentissage*. La séquence de consigne utilisée pour l'apprentissage doit donc être représentative de la consigne du correcteur pendant l'utilisation du SCMI. Pendant le fonctionnement du système de commande, l'entrée de consigne du correcteur n'est plus la consigne r (ou y_r), mais r* (ou y_r*), décalées par rapport à r (ou y_r) de l'écart entre les sorties du processus et celles du modèle, décalage dû aux défauts de modélisation et aux perturbations

éventuelles. Il faut donc choisir une séquence de consigne d'apprentissage dans un domaine d'amplitudes et de fréquences plus vaste que celui de la consigne prévue pour le processus.

II.4.2. SCMI utilisant un correcteur-S.

Comme pour les SCSB, nous faisons appel aux résultats établis en annexe II.

II.4.2.1. Cas d'un modèle entrée-sortie.

Nous montrons en annexe II §II.2.2 que le correcteur-S doit être utilisé avec un *modèle de ralliement*, et non un modèle de poursuite. Sa sortie y_r* est l'entrée du correcteur. La sortie du modèle de ralliement est :

$$y_r^*(k+d) = \big(1 - E(q)\big)\, y(k+d) + H(q)\, r^*(k)$$

où $r^*(k) = r(k) - \big(y_p(k) - y(k)\big)$. Pour pouvoir utiliser ce modèle de ralliement quel que soit le retard d, il faut effectuer la prédiction à d-1 pas de la sortie du modèle[10]. Pour cela, on peut donc utiliser comme prédicteur le modèle interne qui, à l'instant k, calcule la prédiction y(k+d-1) :

$$y(k+d\text{-}1) = h\left(y\{_{k+d-n-1}^{k+d-2},\ u\{_{k-m'-1}^{k-1}\right)$$

Le correcteur-S $\varphi_{RN}^{S,\,e\text{-}s}$ obtenu par apprentissage, associé à ce modèle interne, et qui à l'instant k calcule u(k) telle que y(k+d)=y_r*(k+d) est utilisé avec les arguments suivants :

$$u(k) = \varphi_{RN}^{S,\,e\text{-}s}\left(y_r^*(k+d),\ y\{_{k+d-n}^{k+d-1},\ u\{_{k-m'}^{k-1}\right)$$

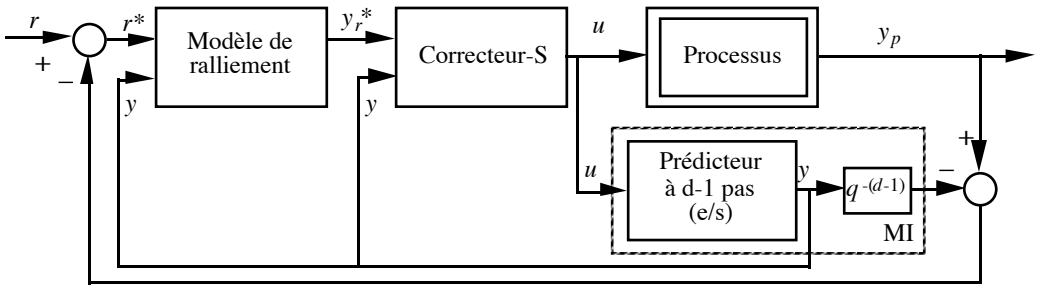


Figure 25.
SCMI avec correcteur-S et modèle de ralliement (modèle entrée-sortie).

Ce SCMI est représenté sur la figure 25. Si le correcteur-S est parfait, le système a la propriété d'imposer la dynamique définie par le modèle de ralliement en poursuite *et* en régulation, et d'assurer une erreur statique nulle par rapport à des perturbations de sortie.

Remarque importante pour l'apprentissage.

y(k+d-1) est la prédiction à k+d-1 de la sortie du processus, calculée à l'instant k. Notons y_{pred}(k)=y(k+d-1) cette prédiction. Le modèle interne prédictif s'écrit :

$$y_{pred}(k) = h\left(y_{pred}\{_{k-n}^{k-1},\ u\{_{k-m'-1}^{k-1}\right)$$

---

[10] En simple bouclage, nous avons vu qu'un modèle de ralliement pour le processus ne peut être utilisé que si d=1, car le calcul de la sortie du modèle de ralliement y_r(k+d) effectué à l'instant k nécessite que l'on dispose de y_p(k+d-1).

C'est donc sous cette forme (avec un retard de 1) que le système d'apprentissage du correcteur-S doit utiliser le modèle de simulation ; le modèle de référence d'apprentissage doit être un retard 1.

II.4.2.2. Cas d'un modèle d'état.

La présentation des propriétés des SCMI et de leur conception a été effectuée dans le cas d'un modèle entrée-sortie en vue d'un maximum de clarté, ainsi que pour faire le lien avec les résultats de Morari, qui sont les plus connus. Dans le cas d'un modèle d'état, seuls changent les arguments des correcteurs-S et -D. Une présentation d'un SCMI utilisant correcteur-S et modèle de ralliement pour le cas de *modèles d'état non linéaires (bilinéaires)* est donnée par l'ADERSA[11], par exemple dans [ABU89]. Dans le cas d'un modèle d'état, il n'est pas possible d'" avancer le modèle ", et donc d'utiliser un modèle de ralliement. Le SCMI utilise donc en général un modèle de poursuite :

$$E(q)\, y_r^*(k+d) = H(q)\, r^*(k)$$

Le MI est le modèle de simulation utilisé pour l'apprentissage :

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

Le correcteur-S obtenu par apprentissage, $\psi_{RN}^{S,\,état}$, est bouclé sur le modèle :

$$u(k) = \psi_{RN}^{S,\,état}\left(y_r^*(k+d),\, x(k)\right)$$

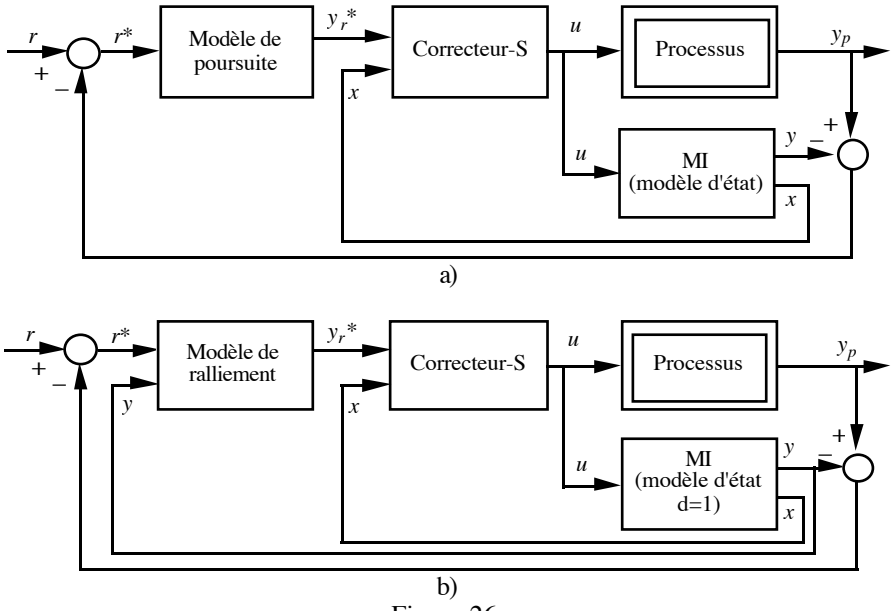


Figure 26.
SCMI avec correcteur-S (modèle d'état).

Le système de commande correspondant est représenté sur la figure 26a). Si d est égal à 1, on utilise le système de commande de la figure 26b). Nous montrons dans l'annexe II §II.2.1.2 que le système de la figure 26a) est peu robuste vis-à-vis d'imperfections du correcteur par rapport au modèle, pour un système linéaire ; nous le montrerons également au chapitre 6 pour un processus (simulé) non linéaire (§I.5.3). Si l'ordre relatif est supérieur à 1, il vaut mieux utiliser un correcteur-D (§II.4.3.2).

---

[11] Association pour le Développement de l'Enseignement et de la Recherche en Systématique Appliquée (7, Bd du Maréchal Juin, 91371 Verrières-le-Buisson Cedex).

II.4.3. SCMI utilisant un correcteur-D.

II.4.3.1. Cas d'un modèle entrée-sortie.

Ici encore, la solution la plus élégante consiste à apprendre le correcteur-D à l'aide du modèle de simulation avec un retard de 1 pas, qui est utilisé comme MI avec les arguments :

$$y(k+d\text{-}1) = h\left(y\{_{k+d-n-1}^{k+d-2}, u\{_{k-m'-1}^{k-1}\right)$$

Le correcteur-D obtenu par apprentissage, toujours noté $\varphi_{RN}^{D,\,e\text{-}s}$, est mis en cascade avec le modèle :

$$u(k) = \varphi_{RN}^{D,e\text{-}s}\left(H(q)\ r^*(k), y\{_{k+d-n}^{k+d-1}, u\{_{k-m'}^{k-1}\right)$$

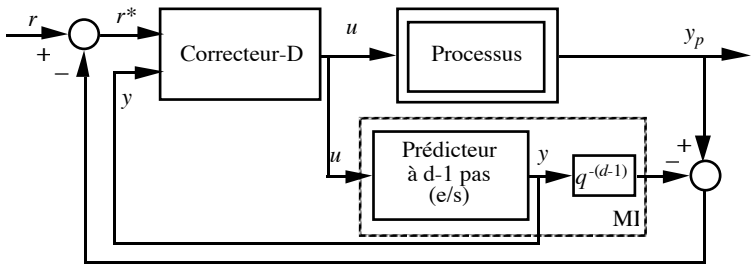


Figure 27.
SCMI avec correcteur-D (pour un modèle entrée-sortie).

Le SCMI est représenté sur la figure 27. Ses propriétés sont les mêmes que celles du SCMI utilisant un correcteur-S (supposé parfait) et un modèle de ralliement, si le correcteur-D est parfait. Sinon, ce système est également assez robuste vis-à-vis d'une imperfection du correcteur.

II.4.3.2. Cas d'un modèle d'état.

Le correcteur neuronal obtenu par apprentissage est utilisé dans le SCMI avec les arguments :

$$u(k) = \psi_{RN}^{D,\,état}\left(H(q)\ r^*(k), x(k), y\{_{k+d-p}^{k-1}\right)$$

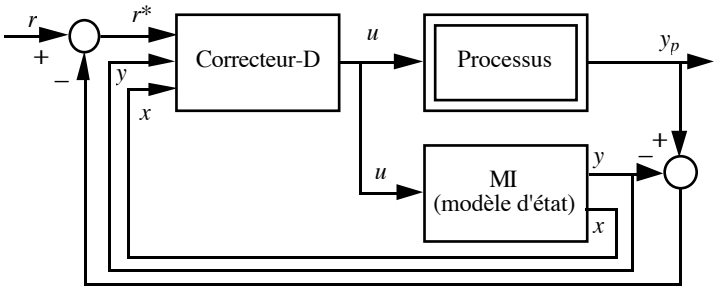


Figure 28.
SCMI avec correcteur-D (pour un modèle d'état).

Le SCMI est représenté sur la figure 28. Il possède bien sûr les mêmes propriétés que le précédent, si le correcteur est parfait.

II.4.4. Conclusions.

S'il existe un correcteur-S et un correcteur-D théoriques stables, que l'on peut donc estimer par un apprentissage, concluons sur les différents systèmes de commande avec modèle interne présentés :

- *Si l'on dispose d'un modèle entrée-sortie du processus, quel que soit son retard*, il faut utiliser un correcteur-S et un modèle de ralliement, en utilisant comme modèle interne le modèle avancé de d-1 pas, et le correcteur-S adéquat à ce modèle sans retard. Ce système de commande présente une plus grande souplesse d'utilisation que le système de commande utilisant un correcteur-D, qui par ailleurs possède les mêmes propriétés.
- *Si l'on dispose d'un modèle d'état du processus, et que l'ordre relatif du modèle est égal à 1*, il faut utiliser un correcteur-S et un modèle de ralliement. Ce système de commande possède les mêmes propriétés que le système de commande utilisant un correcteur-D, mais ici encore présente une plus grande souplesse d'utilisation.
- *Si l'on dispose d'un modèle d'état du processus, et que l'ordre relatif est plus grand que 1*, il n'est plus possible d'utiliser un modèle de ralliement. Dans ce cas, il vaut mieux utiliser un système de commande avec correcteur-D, quitte à effectuer un nouvel apprentissage avec un modèle de référence différent si la performance n'est pas satisfaisante.

Si le modèle est à inverse instable, ce qui est mis en évidence dès l'apprentissage, les correcteurs obtenus ne peuvent être mis en œuvre au sein de systèmes de commande avec modèle interne. Ils peuvent néanmoins être utilisés dans des systèmes de commande par simple bouclage, d'où l'intérêt de ces systèmes.

CONCLUSION.

Pour le problème de la régulation de l'état, l'intérêt des réseau de neurones réside dans leur capacité à réaliser des lois de commande par retour d'état non linéaire, optimales au sens d'un coût quadratique. Ceci est illustré dans la deuxième partie de cette thèse par la commande latérale du véhicule REMI, qui pose le problème de régulation suivant : ramener à zéro l'erreur latérale et l'erreur de cap du véhicule par rapport à une trajectoire de consigne, à l'aide de la commande du volant du véhicule.

Pour la poursuite de la sortie, les réseaux de neurones sont un outil efficace pour réaliser des systèmes de commande imprimant au processus (non linéaire) un comportement linéaire désiré, fondés sur les correcteurs-S et -D. Nous avons montré que ces correcteurs gagnent à être employés au sein de systèmes de commande avec modèle interne, si le modèle du processus est à inverse stable. Ces systèmes ont fait l'objet de beaucoup moins de travaux que les régulateurs optimaux, aussi leurs propriétés sont-elles largement illustrées au chapitre 6, à l'aide d'un processus simulé, et au chapitre 8, par l'asservissement de vitesse du véhicule REMI.

Chapitre 6

EXEMPLES DE COMMANDE DE PROCESSUS

INTRODUCTION.

Le but de ce chapitre est d'illustrer la mise en œuvre et les propriétés des divers systèmes de commande présentés au chapitre 5 précédent. Le problème de la régulation de l'état d'un processus étant bien illustré par la commande latérale du véhicule REMI, nous ne l'abordons pas ici. Nous portons nos efforts sur la comparaison des systèmes de commande de poursuite et de régulation de sortie, et nous illustrons les propriétés de ces systèmes avec un processus non linéaire. Ce dernier est le processus simulé par le modèle d'état identifié au chapitre 4 §II.3.2.2.

Par ailleurs, les méthodes du chapitre 5 n'étant pas optimales par rapport au bruit, nous montrons rapidement au §II qu'il est néanmoins possible de construire des systèmes de commande à variance minimale en utilisant le correcteur-S, dans les cas NARX et NBSX. Ceci est illustré à l'aide du processus simulé par le modèle entrée-sortie identifié au chapitre 4 §I.3.2.1.

I. POURSUITE ET RÉGULATION DE LA SORTIE.

I.1. PRÉSENTATION.

Nous étudions les performances des correcteurs-S et -D au sein de systèmes de commande par simple bouclage (SCSB), et avec modèle interne (SCMI). Notre but est ici de mettre en évidence les propriétés de robustesse de ces systèmes de commande par rapport à des perturbations déterministes non mesurées, ainsi que par rapport à des défauts de modélisation ou d'apprentissage des correcteurs.

Processus.

Le processus simulé est le suivant :

$$\begin{cases} x_{1p}(k+1) = a_1\, x_{1p}(k) + a_2\, x_{2p}(k) + (b_1 + 2\, b_2)\, u(k) \\ x_{2p}(k+1) = \dfrac{x_{1p}(k)}{1 + 0{,}01\, x_{2p}(k)^2} + \dfrac{(-\, b_2)}{a_2}\, u(k) \\ y_p(k) = 4\, tanh\left(\dfrac{x_{1p}(k)}{4}\right) \end{cases}$$

avec $a_1 = 1{,}145$; $a_2 = -\,0{,}549$; $b_1 = 0{,}222$; $b_2 = 0{,}181$. Ces coefficients sont obtenus à partir de la discrétisation du filtre linéaire du second ordre de pulsation ω_n=3, d'amortissement ξ=0,4 et de gain 1 (voir chapitre 4 §II.1). Le modèle est stable et à inverse stable dans le domaine de fonctionnement décrit par le processus lors de l'identification (la commande varie entre ±5). L'apprentissage des

correcteurs-S et -D, et leur utilisation au sein de SCSB et SCMI ne pose donc pas de problème théorique.

Objectifs de commande.

La dynamique de poursuite désirée pour le système de commande, ou dynamique de référence, est définie par le modèle obtenu en discrétisant le second ordre linéaire de pulsation $\omega_n=3$, d'amortissement $\xi=0,7$ et de gain 1, que nous notons :

$$E(q)\, y_r(k+1) = H(q)\, r(k)$$

avec :

$E(q) = 1 + e_1\, q^{-1} + e_2\, q^{-2} = 1 - 1,017\, q^{-1} + 0,350\, q^{-2}$; $H(q) = h_0 + h_1\, q^{-1} = 0,195 + 0,137\, q^{-1}$.

Ce modèle est plus rapide que le processus non asservi ; il est beaucoup plus amorti que le processus pour les petites amplitudes ; son gain statique est constant, toujours supérieur à celui du processus, qui lui décroît avec l'amplitude de la commande (voir chapitre 4 §II.1). D'autre part, nous nous intéressons à la dynamique de régulation par rapport à des perturbations additives de sortie, et à la compensation de ces perturbations, comme cela a été fait au chapitre 5 §II, et dans l'annexe II pour des systèmes linéaires.

I.2. APPRENTISSAGE DES CORRECTEURS-S ET -D.

Modèle de référence d'apprentissage.

Sa sortie y_a obéit à : $E_a(q)\, y_a(k+1) = H_a(q)\, r(k)$, avec (cf. chapitre 5 §II.2.1) :

- dans le cas d'un correcteur-S, $E_a(q) = 1$; $H_a(q) = 1$,
- dans le cas d'un correcteur-D, $E_a(q) = E(q)$; $H_a(q) = H(q)$.

Modèle de simulation.

Nous avons conservé le modèle de simulation neuronal établi à partir du prédicteur entrée-sortie non bouclé à cinq neurones cachés identifié au chapitre 4 §II.3.2.2 :

$$y(k) = \varphi_{RN}^{Sim2}\big(y(k\text{-}1),\, y(k\text{-}2),\, u(k\text{-}1),\, u(k\text{-}2)\big)$$

où φ_{RN}^{Sim2} est la fonction réalisée par la partie non bouclée du réseau obtenu en fin d'identification.

Correcteurs-S et -D.

Ces correcteurs sont bouclés. Il faut pour les réaliser des réseaux de neurones bouclés de la forme (cf. chapitre 5 §II.2.1) :

$$u(k) = \varphi_{RN}\big(H_a(q)\, r(k),\, y(k),\, y(k\text{-}1),\, u(k\text{-}1);\, C\big)$$

Nous utilisons des réseaux de même complexité que le modèle, c'est-à-dire possédant 5 neurones cachés complètement connectés. La fonction d'activation de leur neurone de sortie est une sigmoïde d'amplitude 5, ainsi choisie pour ne pas sortir du domaine de fonctionnement dans lequel le modèle de simulation est valable.

Algorithme d'apprentissage.

Nous choisissons de réaliser un apprentissage semi-dirigé, où ni les entrées d'état du modèle, ni celles du correcteur, ne sont dirigées par les sorties du modèle de référence (l'algorithme est nécessairement semi-dirigé puisque le correcteur est bouclé).

Le système d'apprentissage complet est représenté sur la figure 1.

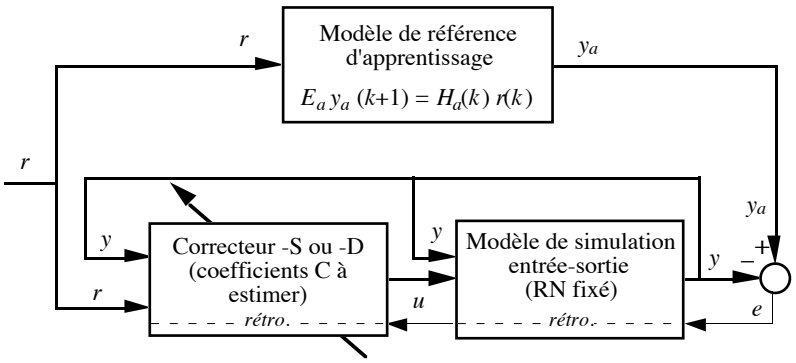


Figure 1.
Système d'apprentissage des correcteurs-S et -D en semi-dirigé.

Séquences d'apprentissage.

La sortie du modèle se stabilise à environ ±3,5 pour la commande d'amplitude maximale ±5. Afin que les performances respectives des correcteurs-S et -D soient comparables, nous avons choisi dans chaque cas :

- correcteur-S : la séquence de consigne d'apprentissage {r(k)} est calculée à l'aide du modèle de référence définissant la dynamique de poursuite désirée (polynômes E et H), à partir d'une suite de créneaux d'amplitudes aléatoires dans l'intervalle [-3,5 ; 3,5], et de durée 10 pas d'échantillonnage. La séquence totale comporte 1000 pas d'échantillonnage. La séquence de référence d'apprentissage $\{y_a(k)\}$, avec $y_a(k+1) = r(k)$, est représentée sur la figure 2 suivante.
- correcteur-D : la séquence de consigne d'apprentissage {r(k)} est la séquence de créneaux précédente. $\{y_a(k)\}$ est calculée à l'aide du modèle de référence d'apprentissage ($E_a(q) = E(q)$; $H_a(q) = H(q)$), et coïncide avec la séquence de référence d'apprentissage du correcteur-S. Ces séquences sont représentées sur la figure 3.

Séquences d'estimation de la performance.

Nous présentons les résultats obtenus en fin d'apprentissage sur une séquence de test destinée à discriminer les différents systèmes de commande, en poursuite, et en particulier en régulation. Il s'agit d'une série de 5 paliers de consigne de durée 50 pas d'échantillonnage. Une perturbation déterministe additive en sortie d'amplitude +0,5 est appliquée entre les instants 110 et 140 (voir les figures 6a et 6b par exemple). La perturbation de sortie y est signalée par une barre grisée sur l'axe des abscisses.

I.2.1. Correcteur-S.

La performance du correcteur-S en fin d'apprentissage est représentée sur la figure 2.

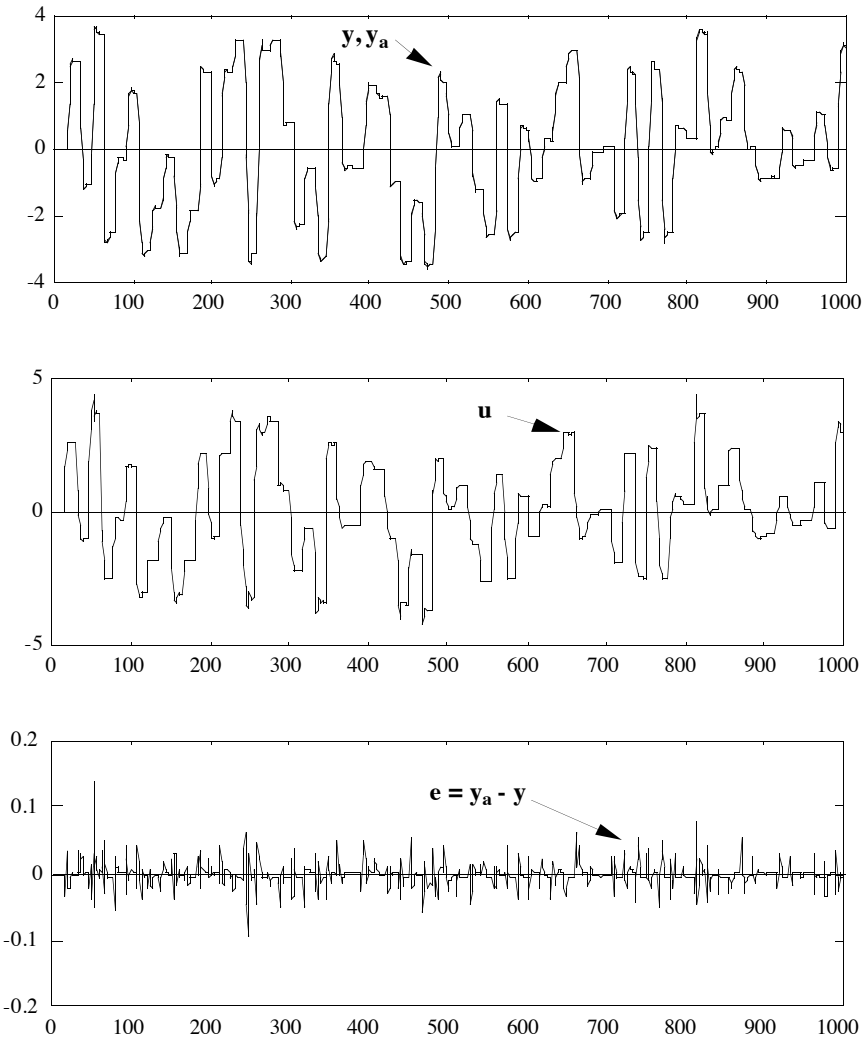


Figure 2.
Performance du correcteur-S sur la séquence d'apprentissage.

Ces résultats montrent que le correcteur obtenu est assez précis pour être utilisé dans un SCMI : on n'observe de grandes erreurs que pour des amplitudes ou des variations de consigne importantes, c'est-à-dire lorsque l'inverse théorique du modèle n'est plus réalisable à cause de la saturation en ±5 du correcteur. Le comportement général de la commande est bien celui attendu : amortir le processus aux petites amplitudes, où il est oscillant, et augmenter son gain aux grandes. Nous notons :

$$u(k) = \varphi_{RN}^{S}\left(r(k), y(k), y(k\text{-}1), u(k\text{-}1)\right)$$

le correcteur-S obtenu en fin d'apprentissage

I.2.2. Correcteur-D.

Les résultats obtenus avec le correcteur-D en fin d'apprentissage sont représentés sur la figure 3. La qualité de l'apprentissage est comparable à celle du correcteur-S, et les remarques générales sont les mêmes que pour ce dernier. Nous notons :

$$u(k) = \varphi_{RN}^{D}\left(H(q)\ r(k), y(k), y(k\text{-}1), u(k\text{-}1)\right)$$

le correcteur-D obtenu en fin d'apprentissage.

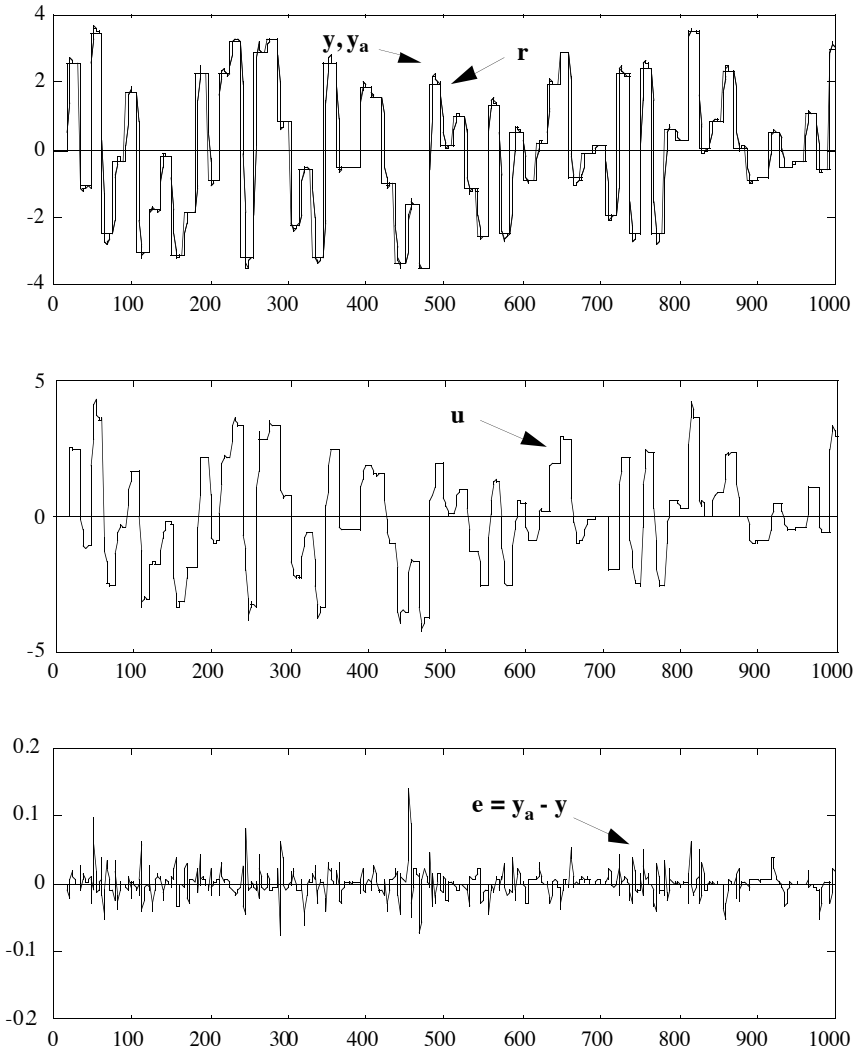


Figure 3.
Performance du correcteur-D sur la séquence d'apprentissage.

I.3. TEST DES CORRECTEURS AVEC LE MODÈLE PERTURBÉ.

Afin d'observer le comportement en régulation du système nominal en réponse à une perturbation de sortie en créneau, nous soumettons les deux correcteurs en simple bouclage avec le modèle (comme pendant l'apprentissage) à la séquence de test.

I.3.1. Correcteur-S.

La figure 4 représente le schéma bloc du système. Le correcteur-S est utilisé avec les arguments :

$$u(k) = \varphi_{RN}^{S}\left(y_r(k+1),\, y(k),\, y(k\text{-}1),\, u(k\text{-}1)\right)$$

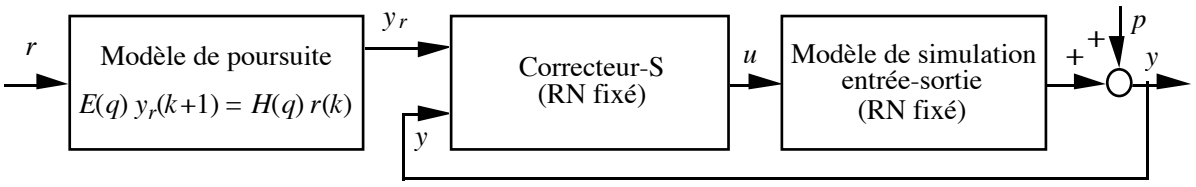
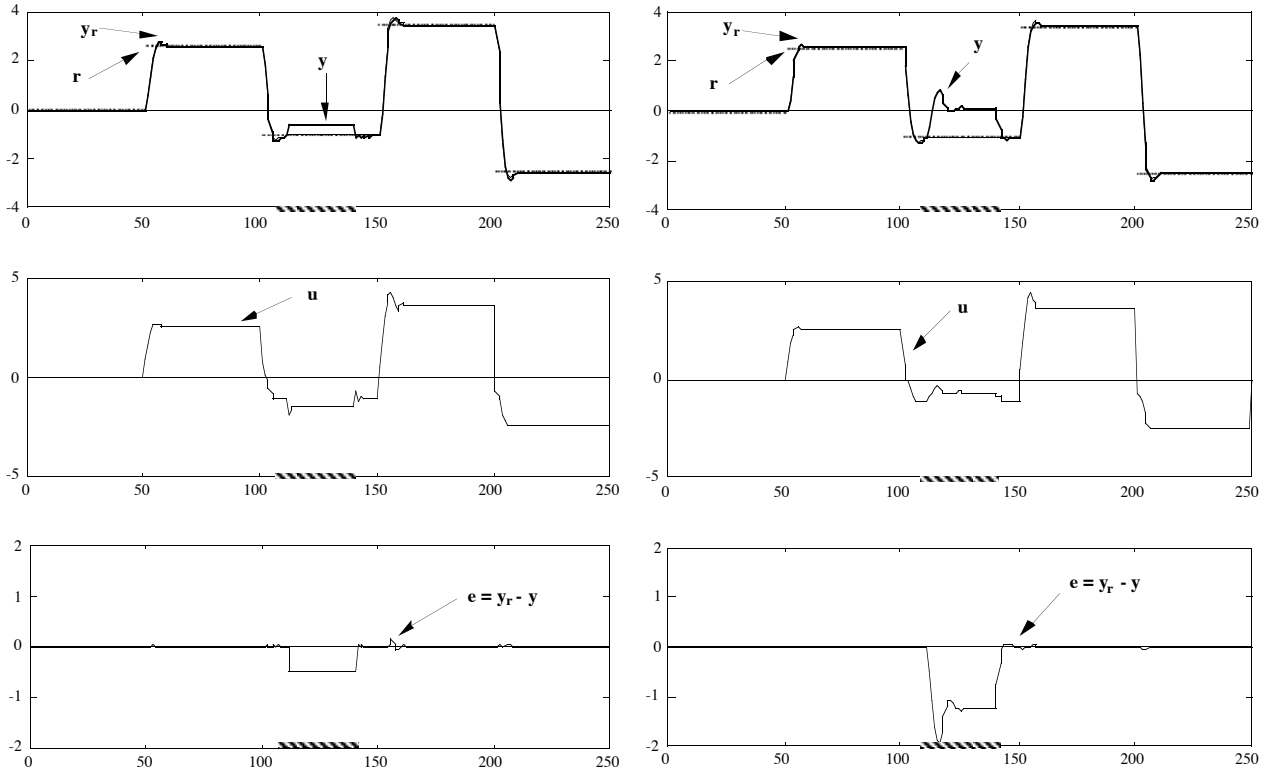


Figure 4.
Test avec le modèle perturbé du correcteur-S.

Les résultats correspondants sont rassemblés sur la figure 5a. Comme on pouvait le prévoir d'après l'annexe II §II.1.1.1 et le chapitre 5 §II.3.1.1, la perturbation provoque une erreur statique non négligeable, et de brusques variations de la commande, de faible amplitude toutefois.



a) Correcteur-S. b) Correcteur-D.

Figure 5.
Performances sur la séquence de test avec le modèle perturbé.

I.3.2. Correcteur-D.

La figure 6 représente le schéma bloc du système. Le correcteur est utilisé avec les arguments :

$$u(k) = \varphi_{RN}^{D}\left(H(q)\ r(k),\ y(k),\ y(k\text{-}1),\ u(k\text{-}1)\right)$$

La séquence $\{y_r(k)\}$, qui sert à évaluer la performance des systèmes de commande, est calculée, comme dans le cas précédent, par le modèle de référence donnant la dynamique désirée, c'est-à-dire $E(q)\ y_r(k+1) = H(q)\ r(k)$.

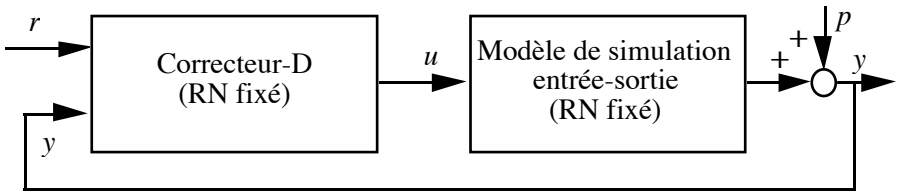


Figure 6.
Test avec le modèle perturbé du correcteur-D.

Les résultats correspondants sont rassemblés sur la figure 5b. Comme il était prévisible (annexe II §II.1.2 et chapitre 5 §II.3.2.1), l'erreur statique provoquée par la perturbation de sortie est plus importante que pour le correcteur-S avec un modèle de poursuite, mais la dynamique de régulation est moins brusque, et ne provoque pas d'à-coups sur la commande : la dynamique de régulation est la même que la dynamique de poursuite.

I.4. SYSTÈMES DE COMMANDE PAR SIMPLE BOUCLAGE (SCSB).

I.4.1. SCSB utilisant un correcteur-S.

Le correcteur-S peut être intégré au sein de deux SCSB différents : un système utilisant un modèle de poursuite, et un système utilisant un modèle de ralliement.

I.4.1.1. Avec un modèle de poursuite.

Le système est représenté sur la figure 7 ; le correcteur est utilisé avec les arguments :

$$u(k) = \varphi^{S}_{RN}\left(y_r(k+1),\, y_p(k),\, y_p(k\text{-}1),\, u(k\text{-}1)\right)$$

où la séquence $\{y_r(k)\}$ est calculée par le modèle de poursuite.

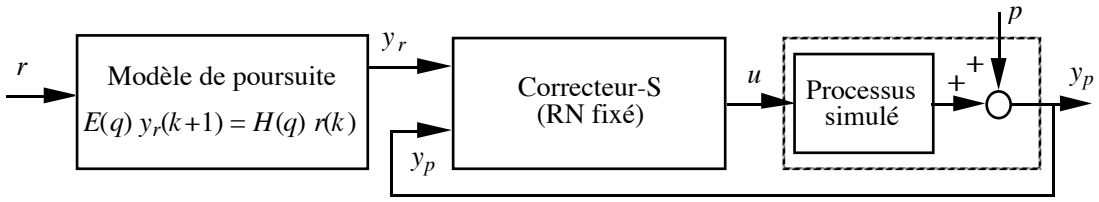
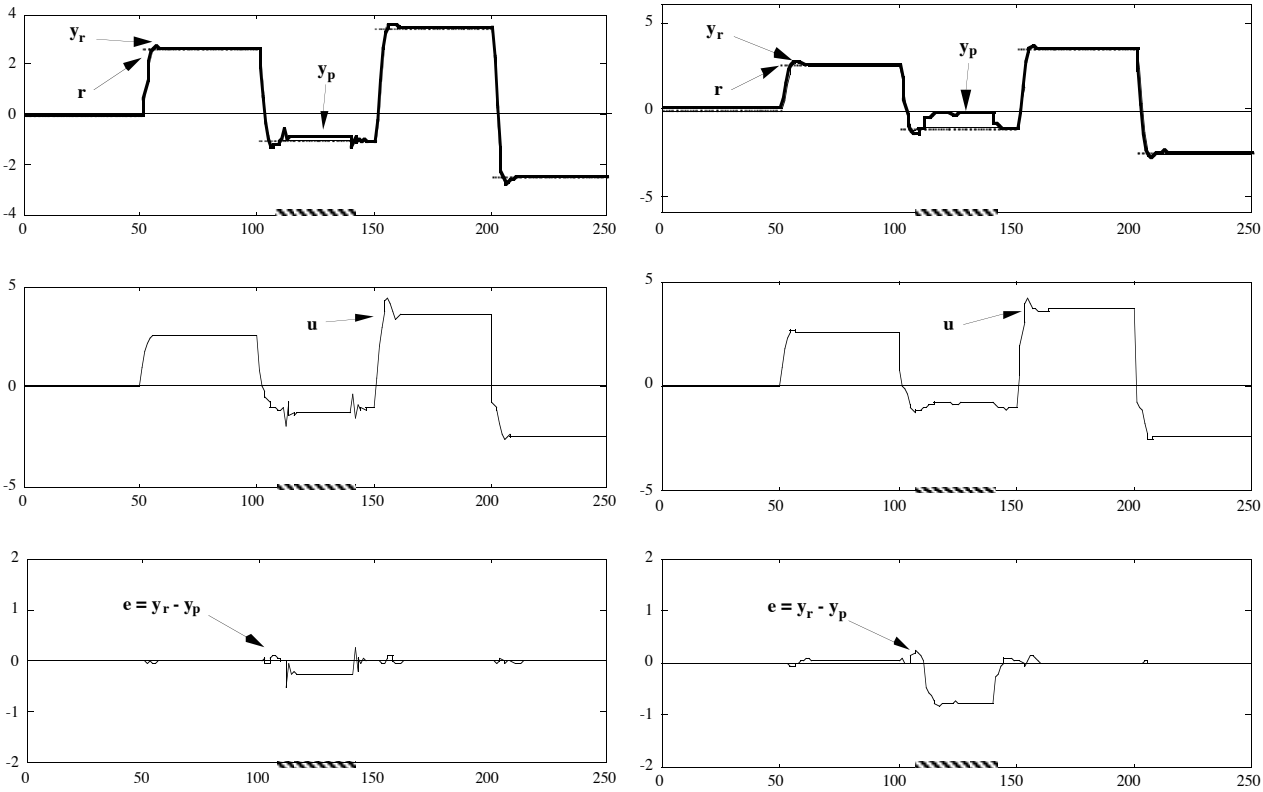


Figure 7.
SCSB utilisant un correcteur-S et un modèle de poursuite.

Les résultats du test sont rassemblés sur la figure 8a. Les défauts de modélisation accentuent les oscillations de la commande en réponse à la perturbation, par rapport au système où le correcteur est adapté au système corrigé, c'est-à-dire le modèle utilisé pour l'apprentissage (comparer à la figure 5a).



a) Avec un modèle de poursuite. a) Avec un modèle de ralliement.

Figure 8
Performance des SCSB utilisant un correcteur-S.

I.4.1.2. Avec un modèle de ralliement.

Le système est représenté sur la figure 9 ; le correcteur-S est utilisé avec les mêmes arguments :

$$u(k) = \varphi^{S}_{RN}\left(y_r(k+1),\, y_p(k),\, y_p(k\text{-}1),\, u(k\text{-}1)\right)$$

mais la séquence $\{y_r(k)\}$ est calculée par le modèle de ralliement.

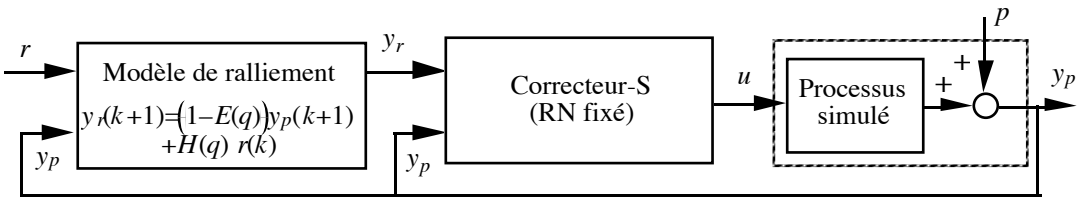


Figure 9.
SCSB utilisant un correcteur-S et un modèle de ralliement.

Les résultats sont rassemblés sur la figure 8b. Le comportement de la commande en réponse à une perturbation est satisfaisant, mais l'erreur statique reste importante (voir annexe II §II.1.1.2). De plus, il existe une erreur statique même en l'absence de perturbation (comparer à la figure 5b).

I.4.2. SCSB utilisant un correcteur-D.

Le système est représenté sur la figure 10 ; le correcteur-D est utilisé avec les arguments :

$$u(k) = \varphi^{D}_{RN}\left(H(q)\ r(k),\, y_p(k),\, y_p(k\text{-}1),\, u(k\text{-}1)\right)$$

La séquence $\{y_r(k)\}$ est calculée par le modèle de référence $E(q)\ y_r(k+1) = H(q)\ r(k)$, et c'est sa sortie qui est utilisée pour le calcul de l'erreur, représentée en bas de la figure 11.

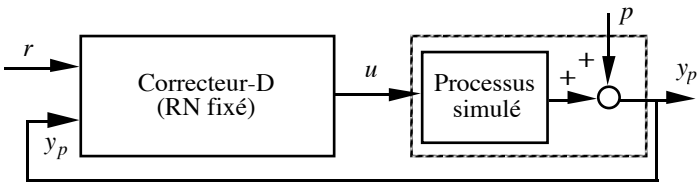


Figure 10.
SCSB utilisant un correcteur-D.

Alors que, si tout est parfait, le SCSB avec correcteur-D, et le SCSB utilisant un correcteur-S et un modèle de ralliement sont équivalents, les défauts de modélisation se répercutent de façon plus importante sur le premier. En effet, la dynamique est très modifiée dans certains domaines du fonctionnement (après la perturbation notamment), et l'erreur statique en l'absence de perturbation est plus importante (dernier créneau de consigne). Ces différences sont peut-être dues à une moins bonne qualité de l'apprentissage du correcteur-D.

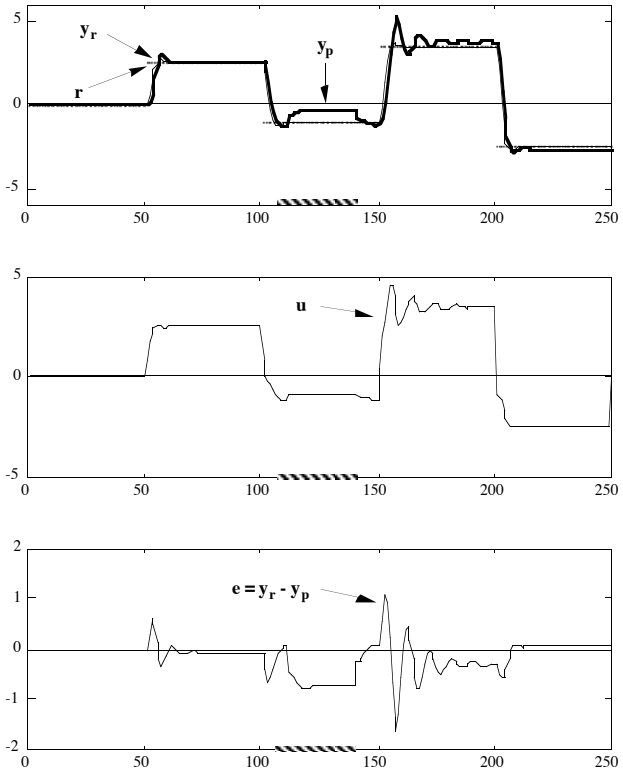


Figure 11.
Performance du SCSB utilisant un correcteur-D.

I.5. Systèmes de commande avec modèle interne (SCMI).

I.5.1. SCMI utilisant un correcteur-S.

Trois SCMI sont possibles : avec modèle de poursuite extérieur, avec modèle de poursuite intérieur, et avec modèle de ralliement. Si le correcteur est parfait (ce qui est presque le cas), les deux

derniers SCMI sont équivalents, et donnent effectivement des résultats identiques. Nous ne présentons donc que le premier et le troisième de ces systèmes.

I.5.1.1. Avec un modèle de poursuite extérieur.

La figure 12 représente le SCMI ; le correcteur-S est utilisé avec les arguments :

$$u(k) = \varphi_{RN}^{S}\left(y_r^*(k+1),\, y(k),\, y(k\text{-}1),\, u(k\text{-}1)\right)$$

où y est la sortie du modèle de simulation φ_{RN}^{Sim2}, utilisé comme modèle interne.

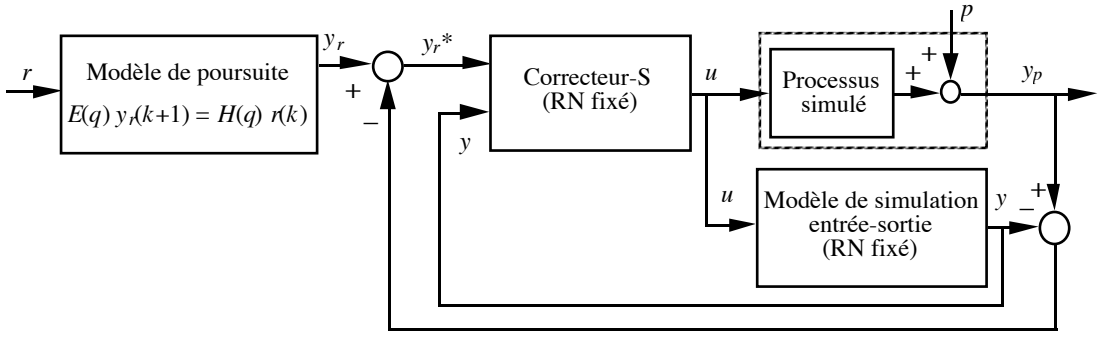


Figure 12.
SCMI utilisant un correcteur-S et un modèle de poursuite extérieur.

Les résultats sont rassemblés sur la figure 14a. Ils sont très bons puisque l'erreur statique est annulée partout, en particulier en réponse à la perturbation. Seul le comportement en régulation de la commande laisse à désirer : des oscillations se produisent.

I.5.1.2. Avec un modèle de ralliement.

Le système de commande est représenté sur la figure 13.

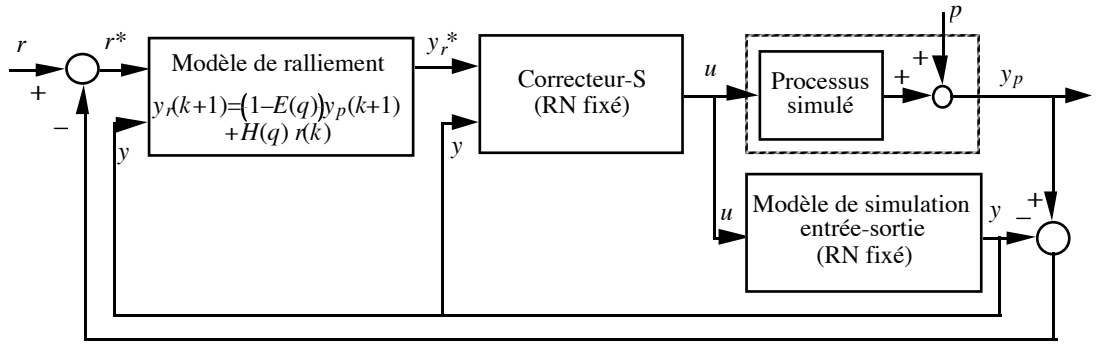
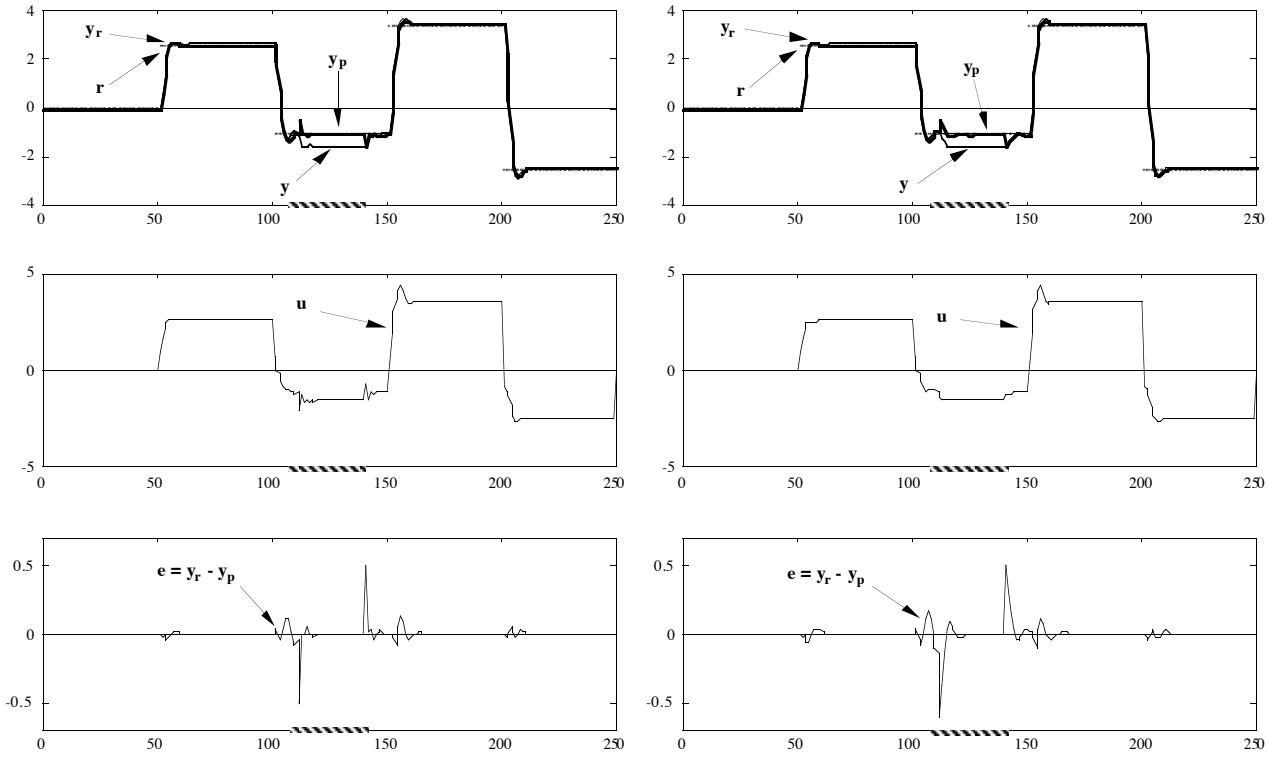


Figure 13.
SCMI utilisant un correcteur-S et un modèle de ralliement.

Le correcteur est utilisé avec les mêmes arguments que dans le SCMI précédent, mais la séquence $\{y_r^*(k)\}$ est calculée par un modèle de ralliement. La séquence $\{y_r(k)\}$ pour estimer l'erreur est calculée à partir de la consigne par le modèle de référence $E(q)\, y_r(k+1) = H(q)\, r(k)$ (cf. §I.4.2). Les résultats sont rassemblés sur la figure 14b. La performance est meilleure que celle du système précédent, car les oscillations de la commande en réponse à la perturbation sont filtrées par le modèle

de ralliement. Ces résultats sont analogues aux résultats établis pour un modèle linéaire dans l'annexe II §II.2.1.1 et §II.2.1.3, et généralisés aux modèles non linéaires au chapitre 5 §II.4.



a) Avec modèle de poursuite extérieur. b) Avec modèle de ralliement.

Figure 14.
Performances des SCMI avec correcteur-S.

I.5.2. SCMI utilisant un correcteur-D.

La figure 15 représente le système de commande ; le correcteur-D est utilisé avec les arguments :

$$u(k) = \varphi_{RN}^{D}\left(H(q)\, r^*(k), y(k), y(k\text{-}1), u(k\text{-}1)\right)$$

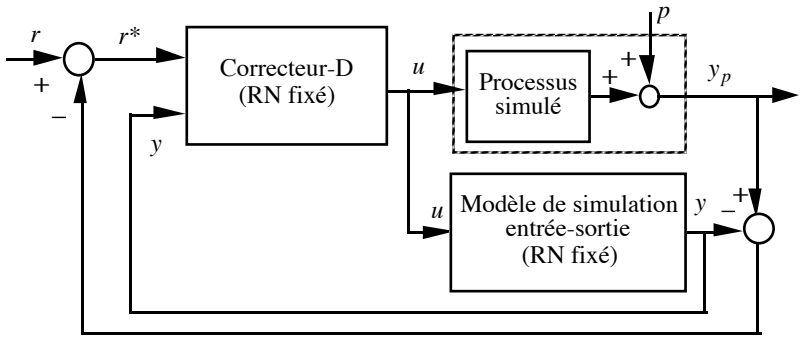


Figure 15.
SCMI utilisant un correcteur-D.

Les résultats, présentés sur la figure 16, sont tout à fait comparables à ceux de la figure 14b, conformément aux résultats de l'annexe II §II.2.1 et §II.2.2. On note cependant qu'il subsiste une petite erreur statique pendant la perturbation : ceci provient donc d'un défaut d'apprentissage du correcteur-D.

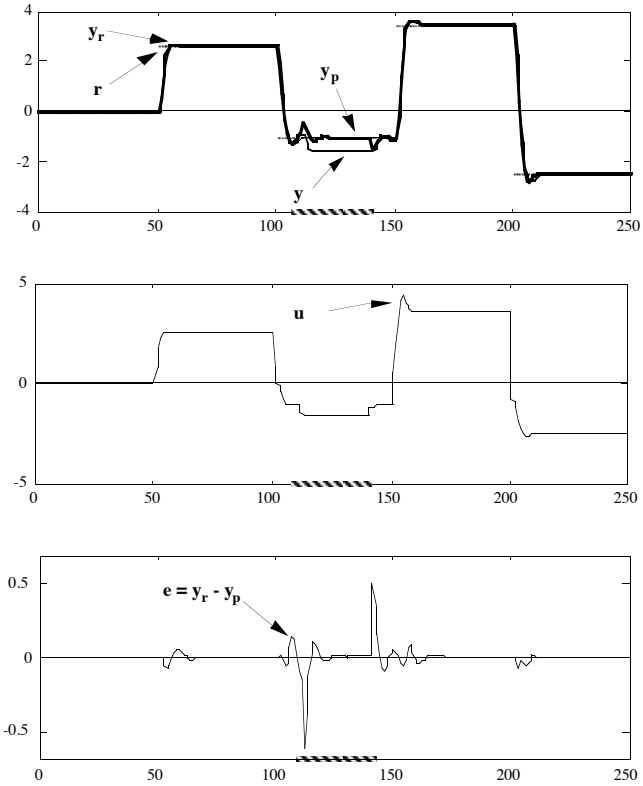


Figure 16.
Performance du SCMI utilisant un correcteur-D.

I.5.3. Suppression du modèle interne.

Au chapitre 5 §II.4.1.5, nous avons insisté sur la nécessité de conserver le modèle interne dans les SCMI, en mettant en évidence les conséquences de sa suppression, en particulier lorsque le correcteur entre en saturation.

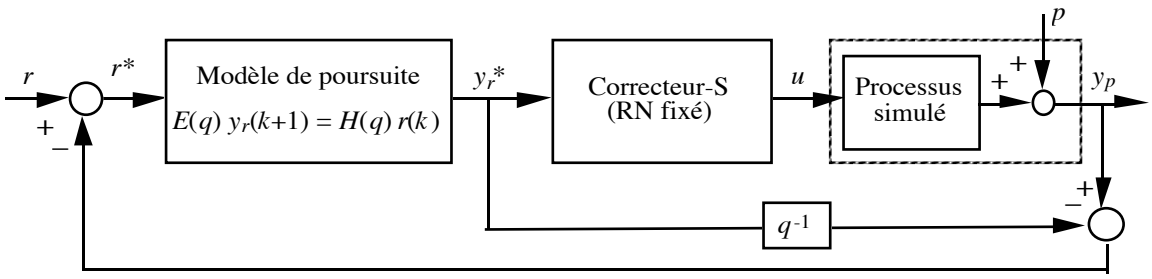
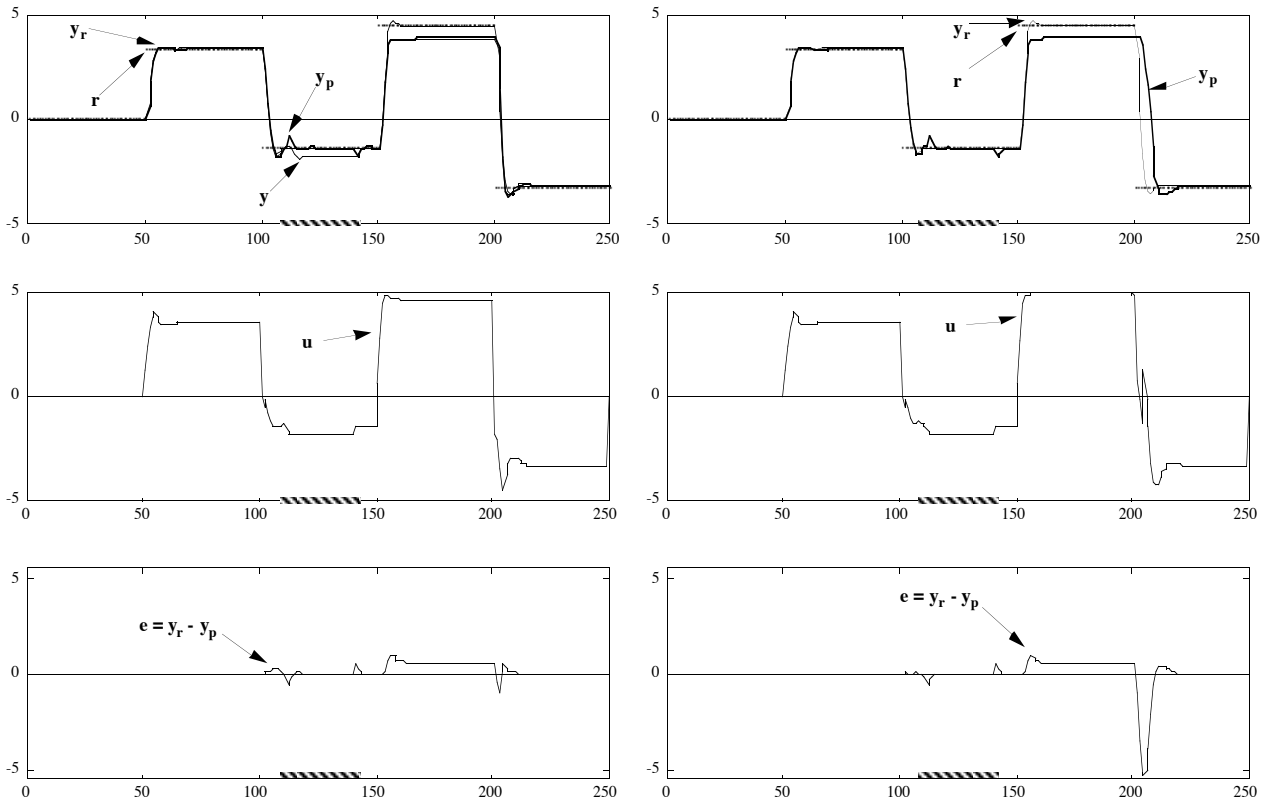


Figure 17.
“ SCMI ” utilisant un correcteur-S et un modèle de poursuite extérieur, sans MI.

Nous illustrons ceci en présentant, à l’aide d’une séquence de test analogue à la précédente, mais pour laquelle le correcteur sature (la consigne maximale est de 4,5 au lieu de 3,5), la performance du SCMI utilisant un correcteur-S et un modèle de ralliement (SCMI de la figure 13), et celle du système de la figure 17 ; le correcteur-S y est utilisé avec les arguments :

$$u(k) = \varphi_{RN}^{S}\left(y_r^*(k+1),\, y_r^*(k),\, y_r^*(k-1),\, u(k-1)\right)$$

Le modèle étant supprimé, le modèle de ralliement doit être remplacé par un modèle de poursuite intérieur.



a) Avec modèle de ralliement. b) Avec modèle de poursuite intérieur, sans MI.

Figure 18.
Performances en saturation de SCMI utilisant un correcteur-S.

Les résultats obtenus avec les deux systèmes sont représentés sur la figure 18. Alors que le SCMI classique se comporte comme le modèle de référence dès que le correcteur ne sature plus, le système sans MI voit sa dynamique très affectée par la saturation : la boucle avec le retard unité réalise une intégration de l'erreur.

I.5.4. Cas d'un correcteur imparfait.

Dans le cas où le correcteur est parfait, les SCMI utilisant un correcteur-S avec un modèle de poursuite intérieur ou un modèle de ralliement sont équivalents. Nous avons cependant établi dans l'annexe II §II.2.1.2 que, pour un modèle linéaire, la stabilité du second système est plus robuste en cas d'une imperfection du correcteur. Afin de vérifier ceci dans un cas non linéaire, nous avons procédé à l'apprentissage d'un correcteur pour un modèle *plus lent* (la même structure de modèle, les coefficients a_1, a_2, b_1 et b_2 étant calculés à partir du second ordre linéaire de pulsation ω_n=2, au lieu de 3). Puis nous avons utilisé ce correcteur dans les deux SCMI, avec le *même modèle de simulation* que dans les paragraphes précédents. Il s'agit donc du cas nominal (processus~modèle), avec correcteur imparfait. Le SCMI avec modèle de ralliement est celui de la figure 13, et le SCMI avec modèle de poursuite intérieur est représenté sur la figure 19 suivante.

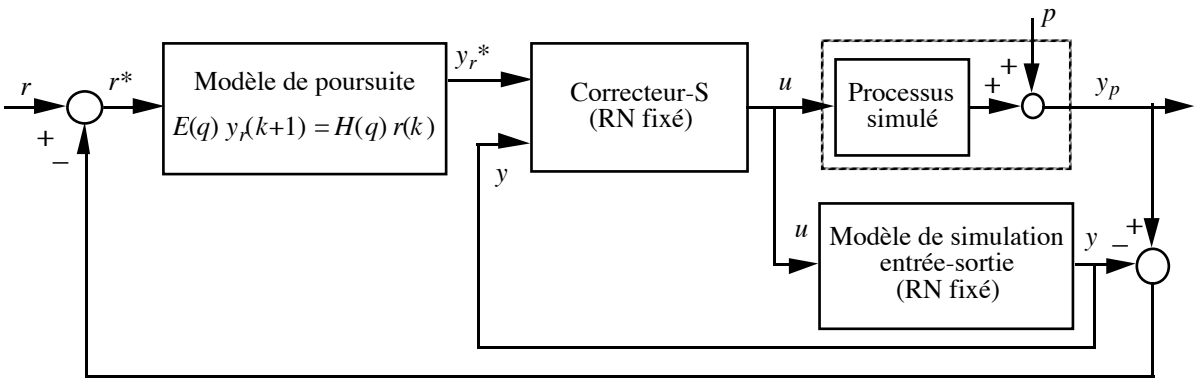
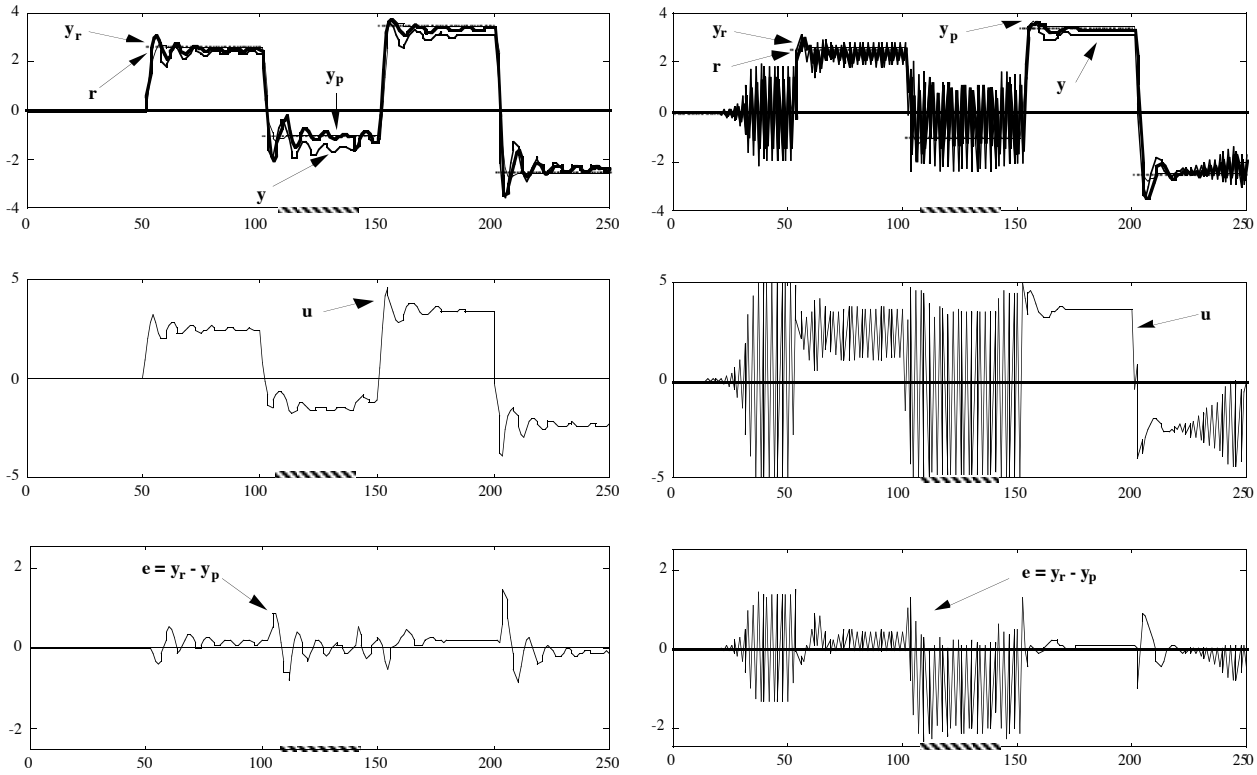


Figure 19.
SCMI utilisant un correcteur-S (***imparfait*** ici) et un modèle de poursuite intérieur.



a) Avec modèle de ralliement. b) Avec modèle de poursuite intérieur.

Figure 20.
SCMI utilisant un correcteur-S ***imparfait***.

Les performances des deux systèmes sont rassemblées sur la figure 20. Le système avec modèle de ralliement est perturbé, mais il n'est pas déstabilisé (figure 20a). Le système avec modèle de poursuite intérieur est instable dans pratiquement tout le domaine de fonctionnement (sa sortie ne devient pas infinie car la sortie du processus est limitée par une tangente hyperbolique). Ces résultats corroborent donc la généralisation à certains systèmes non linéaires des résultats établis dans l'annexe II pour les systèmes linéaires : un SCMI neuronal doit impérativement utiliser un modèle de ralliement, en raison de l'éventuelle imperfection du correcteur due à l'apprentissage.

La performance du SCMI utilisant un correcteur-D imparfait (c'est-à-dire adapté au modèle plus lent), est tout à fait analogue à celle du SCMI utilisant un correcteur-S imparfait et un modèle de ralliement.

I.6. PROCESSUS AVEC BRUIT.

Parmi les systèmes de commande que nous avons présentés, tous ceux dont la dynamique de régulation est satisfaisante, car égale à la dynamique de poursuite, filtrent également le bruit qui peut perturber le processus (en particulier un bruit de sortie), dont nous n'avons pas tenu compte ici. Ces systèmes sont : les SCSB utilisant un correcteur-S et un modèle de ralliement, ou un correcteur-D, et les SCMI utilisant un correcteur-S et un modèle de ralliement ou un modèle de poursuite, ou un correcteur-D. Le §II a pour objet des systèmes de commande optimaux par rapport au bruit.

II. COMMANDE À VARIANCE MINIMALE.

II.1. PRÉSENTATION.

Alors que dans la première partie, nous avons privilégié la robustesse du système de commande vis-à-vis de perturbations *déterministes* non mesurées, l'objet de ce paragraphe est la synthèse de systèmes de commande optimaux vis-à-vis du bruit, c'est-à-dire des perturbations *aléatoires* non mesurées. Les systèmes de commande que nous allons présenter ont donc un intérêt pour la commande de processus pour lesquels les perturbations déterministes sont négligeables devant les perturbations aléatoires. Nous reprenons l'exemple du processus simulé par le modèle entrée-sortie du chapitre 4, processus dont la partie déterministe est :

$$y_p(k) = h\left(y_p(k\text{-}1), y_p(k\text{-}2), u(k\text{-}1)\right) = 50\ tanh\left[2\ 10^{-3}\left(\frac{\left(24+y_p(k\text{-}1)\right)}{3}y_p(k\text{-}1) - 8\ \frac{u(k\text{-}1)^2}{1+u(k\text{-}1)^2}\ y_p(k\text{-}2)\right)\right] + 0{,}5\ u(k\text{-}1)$$

Les systèmes de commande seront conçus pour :

* le processus avec perturbation additive d'état (NARX) :

$$y_p(k) = h\ \left(y_p(k\text{-}1),\ y_p(k\text{-}2), u(k\text{-}1)\right) + w(k)$$

* le processus avec perturbation additive de sortie (NBSX) :

$$\begin{cases} x_p(k) = h\left(x_p(k\text{-}1), x_p(k\text{-}2), u(k\text{-}1)\right) \\ y_p(k) = x_p(k) + w(k) \end{cases}$$

où w est un bruit blanc à valeur moyenne nulle et de distribution uniforme d'amplitude 0,2 (donc de variance $3{,}33\ 10^{-3}$).

II.2. SYSTÈMES DE COMMANDE À VARIANCE MINIMALE.

Pour un processus sans bruit, le correcteur qui minimise la variance de l'erreur de commande est justement le correcteur-S : il annule même l'erreur de commande. Dans le cas d'un processus avec bruit, nous allons montrer que, dans les cas NARX et NBSX, le système de commande minimisant la variance de l'erreur s'obtient facilement à partir du correcteur-S associé au modèle sans bruit. Étudions préalablement le correcteur à variance minimale d'un modèle avec bruit linéaire.

II.2.1. Cas linéaire ARMAX.

Pour cette présentation rapide, nous confondons le modèle du processus et le processus lui-même.

Correcteur à variance minimale (processus avec bruit).

Soit un processus ARMAX :

$$A(q)\, y_p(k) = B(q)\, u(k) + C(q)\, w(k) = q^{-d}\, B'(q)\, u(k) + C(q)\, w(k)$$

où w(k) est un bruit blanc, et C un polynôme monique de degré p. On montre [GOO84] que la commande à variance minimale est :

$$u(k) = y_r(k+d) - G_b(q)\, y_p(k) + \left(1 - F_b(q)\, B'(q)\right) u(k) + \left(C(q) - 1\right) y(k+d) \qquad \text{(I)}$$

où y(k+d) est la prédiction optimale à l'instant k de y_p(k+d) (dont l'expression est établie au chapitre 2 §II.2), et où F_b, monique, et G_b sont les uniques polynômes de degrés respectifs d-1 et n-1 satisfaisant l'équation : $C(q) = F_b(q)\, A(q) + q^{-d}\, G_b(q)$. Le système obéit alors à :

$$e(k+d) = y_p(k+d) - y_r(k+d) = F_b(q)\, w(k+d)$$

soit, si la commande est appliquée à partir de l'instant 0, $e(k) = F_b(q)\, w(k)$ quel que soit k ≥d.

Correcteur-S (processus sans bruit).

Rappelons l'expression du correcteur pour le processus déterministe (annexe II §I.1.1) :

$$A(q)\, y_p(k) = B(q)\, u(k) = q^{-d}\, B'(q)\, u(k)$$

L'expression du correcteur-S est :

$$u(k) = y_r(k+d) - G(q)\, y_p(k) + \left(1 - F(q)\, B'(q)\right) u(k) \qquad \text{(II)}$$

où F, monique, et G sont les uniques polynômes de degrés respectifs d-1 et n-1 satisfaisant l'égalité polynômiale : $1 = F(q)\, A(q) + q^{-d}\, G(q)$. Dans le cas où d=1, $F(q) = 1$; $G(q) = q\left(1 - A(q)\right)$, et (II) devient :

$$u(k) = y_r(k+1) + \left(A(q) - 1\right) y_p(k+1) + \left(1 - B'(q)\right) u(k)$$

On constate que :

- dans le cas particulier ARX, la commande calculée par le correcteur-S est le correcteur à variance minimale. En effet, $C(q) = 1$ et les lois I et II sont donc identiques. Ceci est vrai quel que soit le retard d.
- dans le cas particulier BSX, $C(q) = A(q)$ et donc $F_b(q) = 1$; $G_b(q) = 0$. La commande I s'écrit :

$$u(k) = y_r(k+d) + \left(1 - B'(q)\right) u(k) + \left(A(q) - 1\right) y(k+d)$$

qui, si d=1, est aussi l'expression du correcteur-S, mais appliquée à la sortie y(k+1) du prédicteur optimal et non à celle du processus.

II.2.2. Cas non linéaires NARX (d quelconque) et NBSX (d=1).

De même, dans le cas non linéaire :

- le correcteur à variance minimale pour un processus NARX de retard quelconque est le correcteur-S du processus sans bruit.
- le correcteur à variance minimale pour un processus NBSX avec d=1 est constitué du correcteur-S du processus sans bruit, et du prédicteur optimal NBSX.

Le cas NARMAX dans le cas particulier d'un bruit MA est également facile à traiter, et exposé rapidement dans [GOO84].

II.3. MISE EN ŒUVRE.

II.3.1. Apprentissage du correcteur-S.

Les systèmes de commande à variance minimale des processus NARX et NBSX avec d=1 ne nécessitent donc que l'apprentissage du correcteur-S pour processus sans bruit.

Modèle de simulation.

Le modèle de simulation est le modèle φ_{RN}^{Sim1} obtenu à partir du prédicteur optimal NBSX (identifié au chapitre 4 §I.3.2.1) :

$$y(k) = \varphi_{RN}^{Sim1}\left(y(k\text{-}1), y(k\text{-}2), u(k\text{-}1)\right)$$

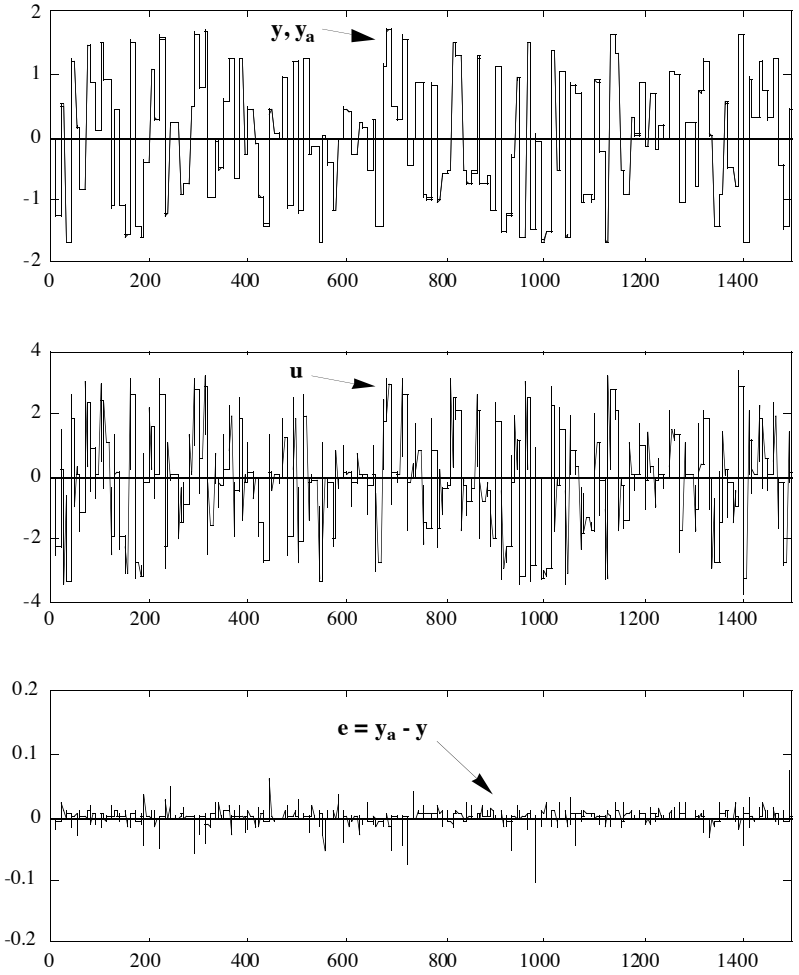


Figure 21.
Performances sur la séquence d'apprentissage.

Correcteur-S.

Nous prenons un réseau de neurones de même complexité que celui du modèle de simulation, possédant 5 neurones cachés complètement connectés, et un neurone de sortie à sigmoïde d'amplitude

5, qui est la valeur maximale de la commande utilisée pour l'identification, valeur qui ne doit donc pas être dépassée pendant l'apprentissage du correcteur. Le correcteur-S est un réseau non bouclé :

$$u(k) = \psi_{RN}\left(r(k), y(k), y(k\text{-}1);\, C\right)$$

Modèle de référence d'apprentissage.

C'est un retard unité. Sa sortie y_a obéit à : $y_a(k+1) = r(k)$

Algorithme d'apprentissage.

L'algorithme est semi-dirigé.

Séquences d'apprentissage.

La séquence de consigne est constituée de créneaux d'amplitudes aléatoires entre ±1,75 (sorties maximales du processus pour une commande entre ±5), et de durée 10 pas d'échantillonnage. La séquence totale comporte 1500 pas d'échantillonnage. Les résultats de l'apprentissage sont représentés sur la figure 21.

Séquences d'estimation de la performance des systèmes de commande.

La séquence de consigne est constituée de créneaux d'amplitudes aléatoires comprises entre ±1,75, et de durée 20 pas d'échantillonnage. La séquence totale comporte 1000 pas d'échantillonnage (cf. figures 23 et 25).

II.3.2. Système de commande du processus NARX.

Le processus NARX est simulé par l'équation :

$$y_p(k) = h\left(y_p(k\text{-}1), y_p(k\text{-}2), u(k\text{-}1)\right) + w(k)$$

où w est un bruit blanc à valeur moyenne nulle, de distribution uniforme d'amplitude 0,2. Le correcteur-S obtenu en fin d'apprentissage, noté ψ^S_{RN}, est mis en cascade avec le processus :

$$u(k) = \varphi^S_{RN}\left(y_r(k+1), y_p(k), y_p(k\text{-}1)\right)$$

Le système de commande à variance minimale pour le processus NARX est représenté sur la figure 22. Le comportement désiré en poursuite est donné par un modèle de référence qui est la discrétisation du filtre passe-bas du second ordre de pulsation ω_n=0,5, de coefficient d'amortissement ξ=0,7, et de gain unité. Ce filtre passe-bas est toujours plus rapide que le processus (dans tout l'espace d'état). Il est noté :

$$E(q)\, y_r(k+1) = H(q)\, r(k)$$

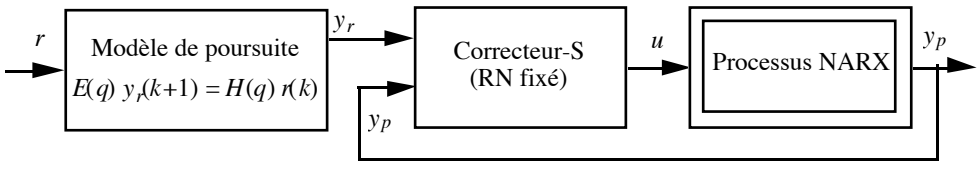


Figure 22.
Système de commande à variance minimale pour le processus NARX.

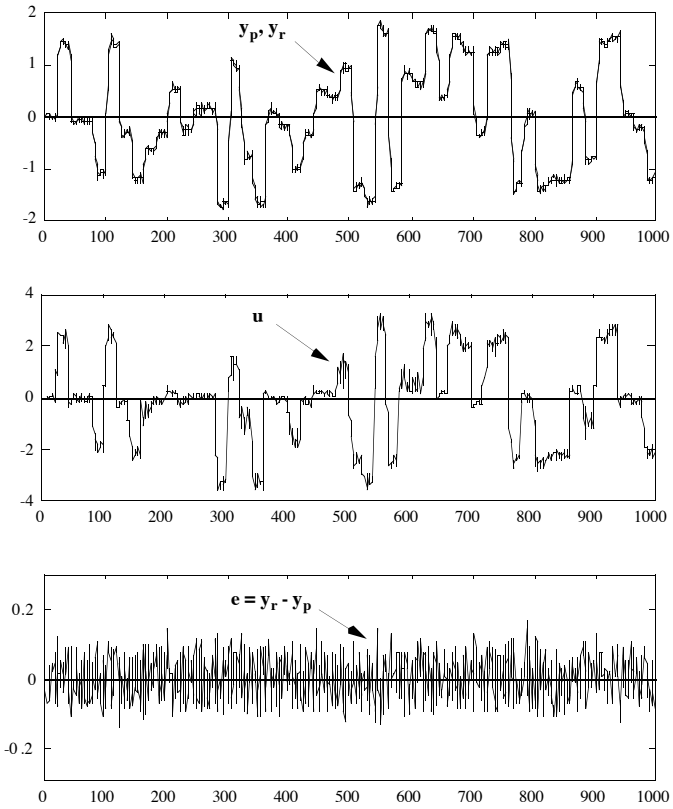


Figure 23.
Performance du système de commande à variance minimale du processus NARX.

Les résultats obtenus sont représentés sur la figure 23. L'erreur de commande a bien les mêmes caractéristiques que la perturbation aléatoire w, c'est-à-dire qu'elle n'est pas corrélée et que sa distribution est semblable à celle de la perturbation (uniforme d'amplitude 0,2).

II.3.3. Systèmes de commande du processus NBSX.

Le processus NBSX est simulé par l'équation :

$$\begin{cases} x_p(k) = h\left(x_p(k\text{-}1),\, x_p(k\text{-}2),\, u(k\text{-}1)\right) \\ y_p(k) = x_p(k) + w(k) \end{cases}$$

où w est un bruit blanc à valeur moyenne nulle, de distribution uniforme d'amplitude 0,2. Le correcteur-S obtenu en fin d'apprentissage est mis en cascade avec le prédicteur optimal du processus NBSX, qui est le prédicteur bouclé :

$$y(k+1) = \varphi_{RN}^{Sim1}\left(y(k),\, y(k\text{-}1),\, u(k)\right)$$

Le correcteur est donc utilisé avec les arguments :

$$u(k) = \varphi_{RN}^{S}\left(y_r(k+1),\, y(k),\, y(k\text{-}1)\right)$$

Le système de commande est représenté sur la figure 24.

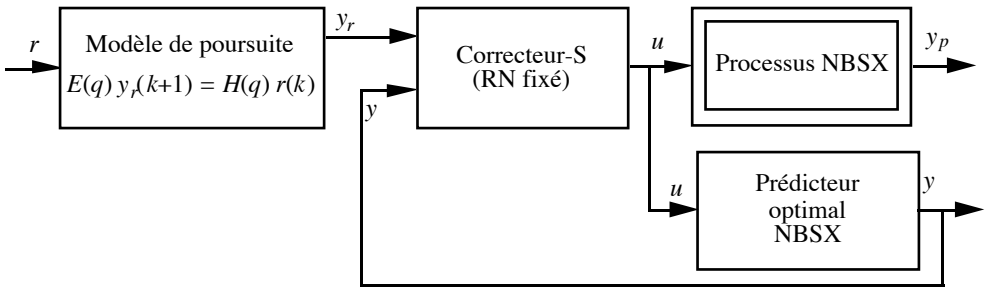
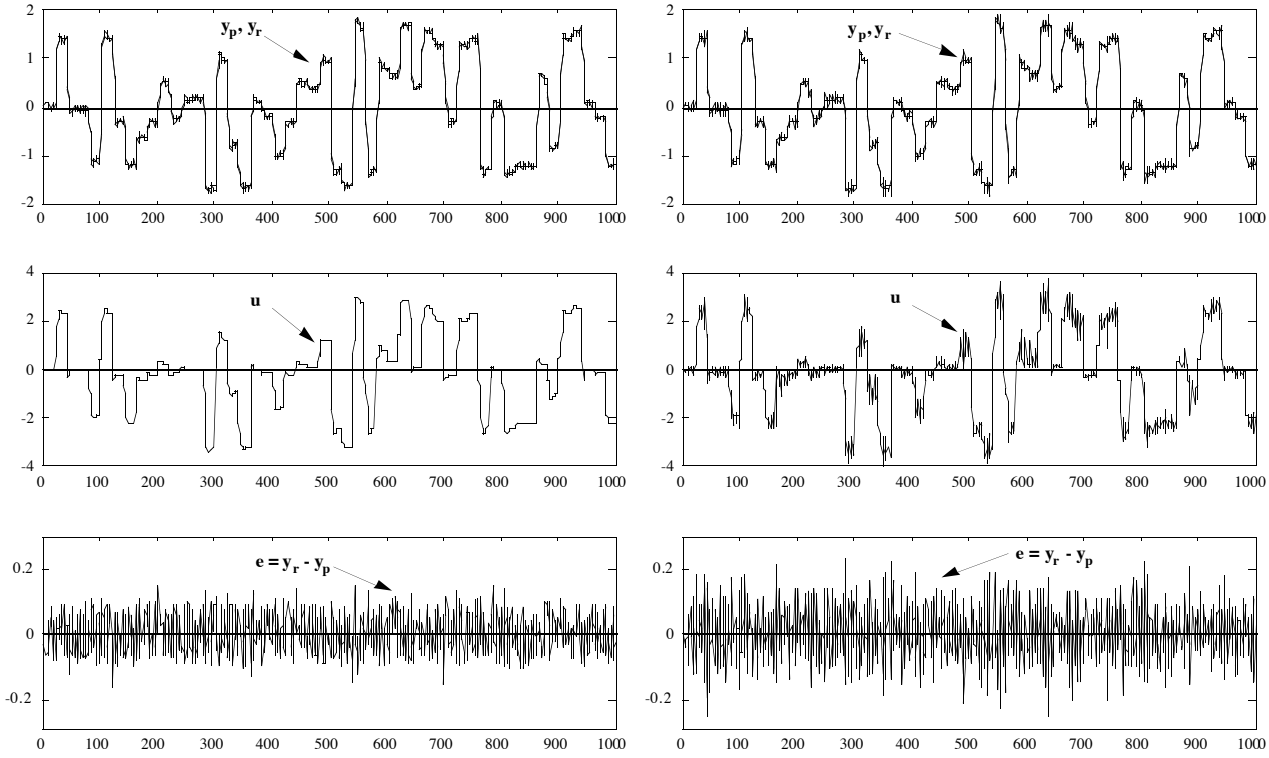


Figure 24.
Système de commande à variance minimale pour le processus NBSX.

Les résultats obtenus sont représentés sur la figure 25a. Ici encore, l'erreur de commande a bien les mêmes caractéristiques que la perturbation aléatoire (distribution, corrélation). On constate que la commande n'est pas bruitée, puisque calculée à partir des sorties du *prédicteur*, qui est déterministe.



a) Système de commande à variance minimale. b) Système de commande par simple bouclage.

Figure 25.
Performances des systèmes de commande du processus NBSX .

À titre de comparaison, nous avons également utilisé le correcteur-S en simple bouclage avec le processus (même système de commande que celui de la figure 22, adéquat à la commande du processus NARX). Les résultats sont représentés sur la figure 25b. La variance de l'erreur de commande est plus importante que celle de la perturbation aléatoire qui affecte le processus, et cette erreur n'a plus les caractéristiques d'un bruit blanc.

CONCLUSION.

La première partie de ce chapitre nous a permis d'illustrer, sur l'exemple d'un processus non linéaire, les propriétés des systèmes de commande par simple bouclage et avec modèle interne démontrées dans le cas linéaire en annexe II. Elle fait notamment apparaître les propriétés de robustesse des systèmes préconisés, c'est-à-dire les systèmes avec modèle interne utilisant un correcteur-S avec un modèle de ralliement, ou bien un correcteur-D, vis-à-vis :

- de perturbations déterministes de sortie en créneaux ;
- de défauts de modélisation (le modèle de simulation utilisé est identifié dans de mauvaises conditions au chapitre 4 §II.3.2.2) ;
- de défauts du correcteur dus à l'apprentissage, problème particulier à l'utilisation de réseaux de neurones.

De plus, nous mettons en évidence le manque de robustesse de systèmes de commande souvent rencontrés dans la littérature.

Dans la seconde partie, nous jetons les bases du principe de la commande à variance minimale par réseaux de neurones pour des processus NARX (de retard quelconque) et NBSX (de retard unité). Un axe de nos recherches futures porte sur la généralisation de ce type de commande aux processus NARMAX.

Deuxième partie :

APPLICATION AU PILOTAGE D'UN VÉHICULE AUTONOME.

INTRODUCTION AU PILOTAGE D'UN VÉHICULE AUTONOME

I. OBJET ET ENJEUX DE L'ÉTUDE.

La conception des véhicules automatisés, ou robots mobiles, à roues est un domaine de recherche en pleine expansion. Ces véhicules sont utilisés dans l'industrie comme moyen de transport, d'inspection, ou d'opération, et sont particulièrement adaptés à des interventions en environnement hostile. La conception mécanique, les systèmes de vision et de localisation, la planification ainsi que la commande de ces véhicules, ont suscité de nombreux travaux. Ainsi, cette étude a pour objet la mise en œuvre de réseaux de neurones pour la commande d'un tel véhicule. Ce dernier est un système intrinsèquement non linéaire de par sa cinématique et ses caractéristiques dynamiques (actionneurs, moteur thermique). La commande d'un tel système est donc un problème qui, pour être résolu de façon satisfaisante, doit prendre ces non-linéarités en considération. Les réseaux de neurones sont de bons candidats : d'une part, ils permettent d'identifier le comportement dynamique du véhicule, souvent complexe, et d'autant plus difficile à modéliser physiquement que les constructeurs fournissent rarement toutes les informations nécessaires ; d'autre part, l'apprentissage de correcteurs utilisant un modèle neuronal du véhicule prend en considération toutes les non-linéarités identifiées. Enfin, des perturbations non modélisées et non mesurées telles que des changements d'adhérence du terrain, ou des perturbations mesurées telles que sa pente, doivent être compensées par l'organe de commande, besoin que les méthodes de commande neuronales par modèle interne présentées dans la première partie de ce mémoire sont à même de satisfaire.

L'objet de cette partie applicative est de donner un exemple des performances effectives des réseaux de neurones pour une tâche concrète, tout en complétant l'illustration des aspects théoriques développés dans ce mémoire, illustration déjà esquissée dans les chapitres 4 et 6 sur des exemples mettant en œuvre des processus simulés, ainsi que l'actionneur d'un bras de robot. Nous montrons également comment l'utilisation des techniques neuronales s'imbrique avec celle des techniques classiques de l'ingénieur automaticien ; nous verrons en particulier comment la modélisation " boîte noire " neuronale se conjugue avec l'analyse physique des phénomènes mis en jeu par le processus, et de quelle manière les préoccupations qui guident la conception d'un système de commande " classique " interviennent dans celle de son homologue " neuronal ".

Dans cette introduction, nous présentons l'*architecture de mobilité classique* d'un véhicule autonome, ainsi que les choix effectués pour sa réalisation au moyen de réseaux de neurones. Enfin nous décrivons le véhicule, REMI, qui a servi de banc d'essais pour ces recherches.

II. ARCHITECTURE DE MOBILITÉ D'UN VÉHICULE AUTONOME.

Véhicules automatisés.

Pour intervenir dans des milieux dangereux, pollués, impropres à la vie humaine, ou pour remplacer l'homme dans l'exécution de tâches répétitives, l'utilisation de véhicules automatisés se généralise. Un véhicule automatisé est un véhicule capable d'exécuter sa tâche sans opérateur humain à bord.

Véhicules autonomes.

Un véhicule autonome est un véhicule automatisé susceptible de remplir sa tâche *sans intervention* aucune d'un opérateur humain, même à distance. Pour de nombreuses applications industrielles, il n'est pas encore envisageable, pour des raisons de sécurité et d'efficacité, de laisser un véhicule automatisé évoluer en autonomie complète dans un environnement peu structuré. Comme nous le verrons au §II.1, il existe donc en général pour des véhicules dits autonomes plusieurs niveaux possibles pour l'intervention à distance d'un opérateur humain (par liaison radio par exemple), afin de compenser l'incapacité du véhicule à s'adapter à des changements trop radicaux de son environnement.

Architecture de mobilité d'un véhicule automatisé.

Un véhicule automatisé remplit le plus souvent d'autres tâches qu'un simple déplacement. Par exemple, un véhicule forestier doit non seulement pouvoir se déplacer en forêt, mais aussi abattre les arbres ; un véhicule minier doit à la fois se déplacer dans les carrières, et effectuer des forages. Nous nous intéressons ici uniquement à la tâche de déplacement, et donc à l'architecture du système qui organise et commande le déplacement d'un véhicule automatisé, ou *architecture de mobilité*.

II.1. ARCHITECTURE DE MOBILITÉ CLASSIQUE.

Architecture fonctionnelle.

La chaîne fonctionnelle d'un véhicule automatisé, représentée sur la figure 1, fait traditionnellement intervenir trois modules de commande organisés hiérachiquement [FRA93] :

- le module de **planification** calcule un itinéraire local sous forme de points de passage à partir d'une destination définie par un opérateur humain (déplacement associé à la tâche du véhicule). Il calcule également une vitesse moyenne ou un profil de vitesse de consigne associé à cet itinéraire, en fonction du délai spécifié par l'opérateur pour la réalisation de la tâche.
- le module de **guidage** calcule les consignes de vitesse et de cap qui permettent au centre de gravité du véhicule (ou à tout autre point du véhicule, appelé *point de commande*) de suivre une trajectoire interpolant les points calculés par le module de planification.

- le module de **pilotage** a pour fonction d'asservir la vitesse et le cap du point de commande sur les valeurs de consigne déterminées par le module de guidage, c'est-à-dire d'élaborer les commandes qu'il faut délivrer aux actionneurs de la vitesse et de la direction.

Un module de *localisation,* qui ne fait pas à proprement parler partie de l'architecture de mobilité, est nécessairement présent à bord du véhicule. Il fournit aux différents étages hiérarchiques les informations de position, d'attitude (cap, roulis et tangage) et de vitesse du véhicule. Nous revenons sur les éléments constitutifs de ce module au paragraphe III. L'architecture de mobilité est représentée sur la figure 1.

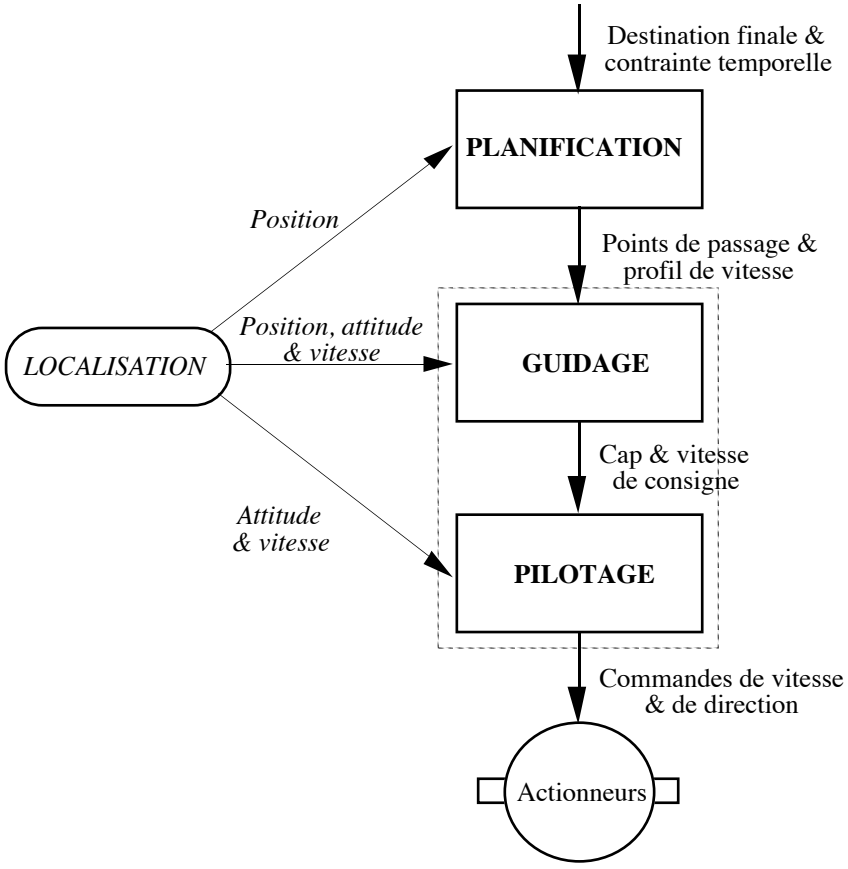


Figure 1.
Architecture de mobilité classique d'un véhicule automatisé.

Niveaux d'intervention.

Dans le cas d'une autonomie partielle, selon le niveau où l'opérateur humain intervient, on donne une dénomination différente au mode de commande du véhicule automatisé :

- on parle de **télépilotage** (ou téléconduite) lorsque les ordres, ou consignes, qui sont transmis au véhicule par un opérateur humain, sont les commandes des actionneurs. Ce mode de commande suppose l'opérateur à un poste de commande reproduisant le véhicule et ses actionneurs, et qu'il dispose d'informations sensorielles (visuelles surtout) représentatives de celles qu'il aurait s'il se trouvait à bord du véhicule. C'est le mode de commande où l'autonomie du véhicule est la plus limitée. Il est relativement sûr, puisque voisin d'une conduite humaine, mais n'est véritablement

praticable qu'à basse vitesse, et en terrain peu accidenté. En effet, il n'est plus possible à haute vitesse de négliger le temps de transfert des ordres de l'opérateur et des informations en provenance du véhicule dans le réseau de communication : au delà de 30 km/h, un système télépiloté devient en général instable. De plus, les informations sensorielles communiquées à l'opérateur sont le plus souvent insuffisantes pour une conduite correcte en tout terrain (champ de vision limité, insuffisance du retour d'informations concernant d'autres modalités sensorielles, telles que les sons, vibrations, accélérations…).

- il s'agit de **téléguidage**, lorsque les consignes communiquées au véhicule sont des consignes de vitesse et de cap (et/ou de position). La fonction de pilotage est dans ce cas assurée par le véhicule. Un tel mode de commande ne souffre pas des inconvénients du télépilotage. L'autonomie du véhicule y est cependant encore restreinte, puisque le choix de la manière de suivre la trajectoire incombe à l'opérateur. Pour que ce choix soit optimal, il est nécessaire que l'opérateur ait une bonne connaissance de la dynamique du véhicule.
- on parle de **téléplanification**, lorsque le véhicule reçoit comme consignes une liste de points de passage assortie d'une contrainte temporelle, qui peut être modifiée par l'opérateur dans le cas de la présence inopinée d'obstacles. Les fonctions de pilotage et de guidage sont assurées au niveau du véhicule. Ce mode de commande est particulièrement sûr, puisque le guidage est bien adapté à la trajectoire à suivre.
- l'**autonomie** est effective, si toutes les fonctions de planification, guidage et pilotage sont assurées par le véhicule. En présence d'obstacles (localisés par un système de détection d'obstacles), le module de planification embarqué redéfinira lui-même les points de passage et les contraintes à respecter, en s'appuyant sur des cartes numérisées du terrain où il évolue.

Enjeux industriels.

Comme nous l'avons déjà observé, le mode de totale autonomie est à l'heure actuelle irréaliste pour la plupart des véhicules automatisés industriels ou militaires. En revanche, le stade de la téléplanification est parfaitement réalisable tout en offrant de meilleures garanties de sécurité que le télépilotage ou le téléguidage, et en ayant l'avantage de moins solliciter l'opérateur, qui doit souvent veiller au déroulement correct d'autres opérations que celle du seul déplacement du véhicule (comme pour les véhicules forestiers ou miniers).

Pour ces raisons, nous avons choisi de réaliser à l'aide de réseaux de neurones les **fonctions réalisées par les modules de guidage et de pilotage** (encadrés en pointillés sur la figure 1), pour un fonctionnement aussi bien en mode téléplanifié qu'en mode autonome, lorsque celui-ci sera industriellement envisageable.

Dans une optique moins roboticienne, on peut aussi envisager l'utilisation pour des véhicules de série d'une partie des fonctions du guidage-pilotage, pour la réalisation de systèmes de " cruise-control " par exemple (ces systèmes permettent de maintenir la vitesse d'un véhicule autour d'une valeur nominale).

II.2. RÉALISATION NEURONALE.

Fonction assurée par le système de commande neuronal.

Le système de commande neuronal doit donc assurer les fonctions de l'ensemble guidage-pilotage. Ceci consiste, à partir des points de passage et du profil de vitesse de consigne déterminés par le module de planification (mode autonome) ou par l'opérateur (mode téléplanifié), à calculer les commandes à délivrer aux actionneurs de manière à respecter au mieux ces consignes.

Réalisation neuronale du guidage-pilotage.

Traditionnellement, le module de guidage, après avoir calculé une trajectoire continue à partir des points de passages, sélectionne à chaque instant un *point-cible* sur cette trajectoire. La position de ce point, et la vitesse de consigne qui lui est associée, sont les données utilisées par le module de guidage pour calculer le cap et la vitesse à imposer au véhicule de manière à atteindre l'objectif de poursuite. Ce *cap* et cette *vitesse* constituent les consignes transmises au module de pilotage. Nous avons conçu un premier système de commande selon ce principe [RIV93] [RIV94]. Dans le système retenu, les consignes transmises au module de pilotage neuronal sont des consignes de *cap*, de *vitesse*, *et de position*. Une comparaison entre les deux approches est présentée dans [RIV93]. La réalisation neuronale des modules de guidage et de pilotage, dont nous allons maintenant décrire les modules constitutifs, est représentée schématiquement sur la figure 2.

Le ***module de guidage*** est constitué des deux éléments suivants :

Le générateur de trajectoire.

À chaque réception d'une mission, ce module interpole les points de passage, fournissant ainsi une trajectoire continue (interpolation linéaire si les points sont suffisamment rapprochés, à l'aide de B-splines sinon). À chaque point de la trajectoire est associée une vitesse correspondant à la contrainte temporelle de la mission (profil de vitesse ou vitesse moyenne constante). Cette opération est donc habituellement aussi réalisée par le module de guidage.

Le sélecteur du point-cible.

À chaque instant, ce module détermine le point-cible, qui est donc le point de la trajectoire dont il faut se donner la vitesse, la position et le cap pour objectif. Intuitivement, ce point doit être choisi à une certaine distance en avant du véhicule. La détermination de la loi gouvernant la distance au point-cible ne peut donc être effectuée qu'en fonction des modules de commande intervenant en aval du sélecteur du point-cible, détermination que nous décrivons au chapitre 8.

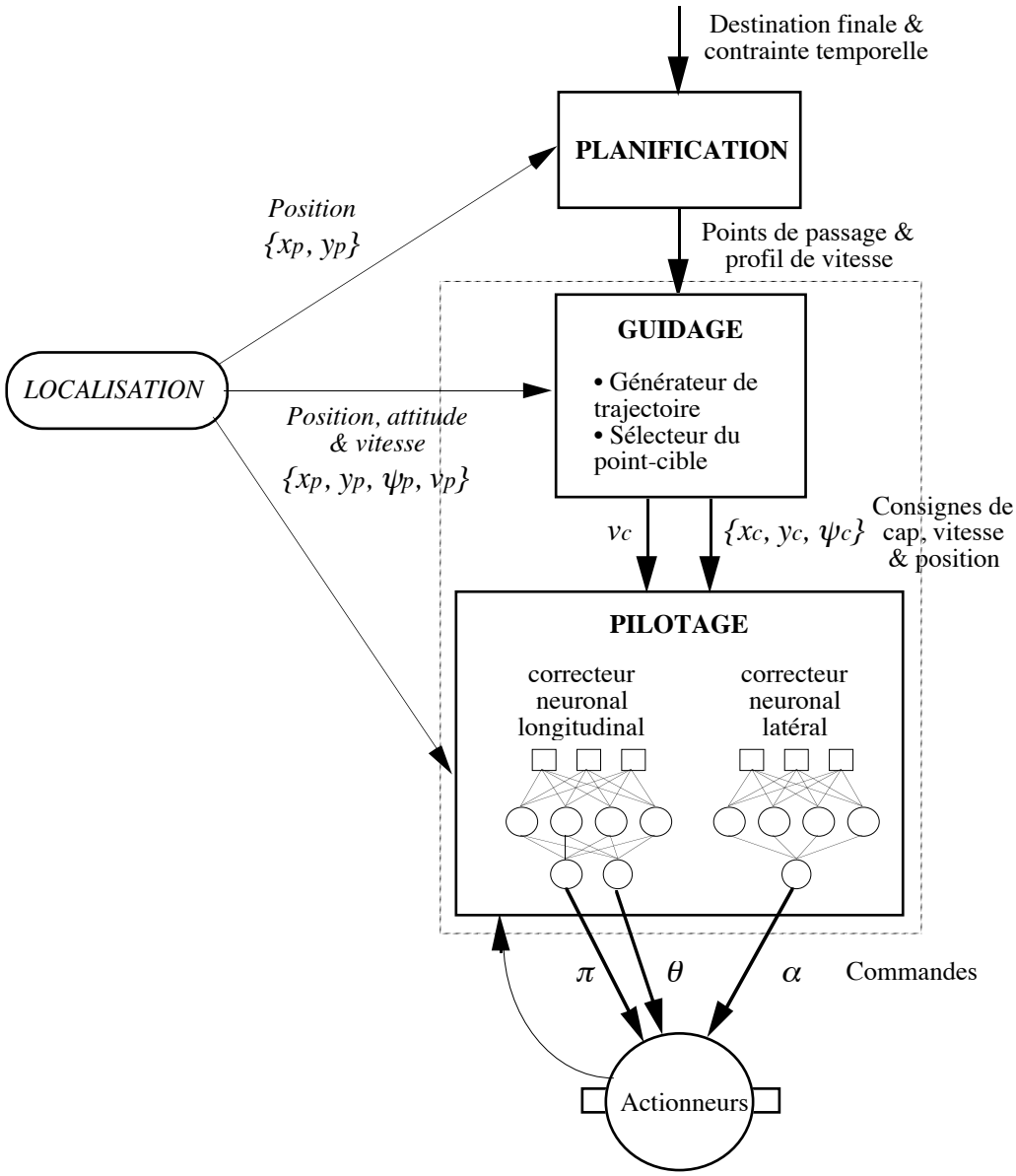


Figure 2.
Architecture de mobilité neuronale.

Le ***module de pilotage*** est constitué de deux correcteurs neuronaux :

D'après des résultats biens connus dans le domaine de la commande de robots mobiles à roues du type de REMI, c'est-à-dire soumis à des contraintes non holonomes, il est possible d'asservir la position du véhicule sur une trajectoire, indépendamment de sa vitesse, par retour d'état statique continu stationnaire [SAM92]. Nous rappelons ces résultats au chapitre 8. C'est cette solution que nous avons choisie, ou solution de " path-following " (opposée à la solution, également envisageable, de " path-tracking ", qui désigne l'asservissement de posture, c'est-à-dire à la fois de la position et du cap). Cette solution a l'avantage de demander la synthèse de deux correcteurs *indépendants*, un correcteur latéral pour l'asservissement sur trajectoire, et un correcteur longitudinal pour l'asservissement de la vitesse.

a) Correcteur latéral.

Ce correcteur a pour consignes la position du point-cible sélectionné, et l'orientation de la vitesse de consigne qui lui est associée, ou cap de consigne. En fonction de l'état du véhicule, il calcule la commande du volant pour l'asservissement du véhicule sur la doite passant par le point-cible et orientée selon le cap de consigne (cette droite est donc la tangente à la trajectoire de consigne passant par le point-cible).

b) Correcteur longitudinal.

Ce correcteur a pour consigne la vitesse du point-cible sélectionné. Il calcule, en fonction de cette consigne et de l'état du véhicule, les commandes des freins, de l'accélérateur, et du sélecteur des vitesses.

Identification des sous-systèmes correspondant aux deux correcteurs.

Les techniques de commande neuronale que nous mettons en œuvre sont de type indirect, c'est-à-dire utilisent un modèle du véhicule. Il est donc nécessaire de réaliser préalablement la modélisation de la dynamique du véhicule pour les comportements latéral et longitudinal. D'autres techniques de commande de véhicules par réseaux de neurones n'utilisent pas de modèle. Par exemple, Pomerleau propose un système qui réalise l'apprentissage d'un réseau de neurones dont l'entrée est une image de la route, et la sortie la commande du volant, à l'aide d'un " professeur " humain [POM91] [POM94].

Nous allons maintenant présenter REMI, le véhicule automatisé qui a servi de banc d'essais à nos recherches ; nous décrivons la nature et le fonctionnement des actionneurs et des capteurs nécessaires à la commande et à la localisation, éléments propres à chaque véhicule automatisé.

III. PRÉSENTATION DU VÉHICULE REMI.

REMI (Robot Evaluator for Mobile Investigations) est un véhicule de série Mercedes à essence à quatre roues motrices. Il est entièrement équipé par la société SAGEM des actionneurs et des capteurs nécessaires à la commande et à la localisation du véhicule. Le véhicule est représenté sur la figure 3.

Actionneurs.

a) Actionneur utilisé pour l'asservissement latéral : c'est un moteur à courant continu sur la colonne de direction, avec mesure de position angulaire.

b) Actionneurs utilisés pour l'asservissement longitudinal :

* accélération : l'action sur le papillon des gaz est effectuée par un moteur à courant continu.

* freinage : un groupe hydraulique fournit l'énergie, et une servo-valve permet de modifier la pression dans le circuit de freinage, avec mesure de la pression.
* passage des rapports de vitesse : il est réalisé par un moteur à courant continu agissant sur le manche du sélecteur des vitesses, avec mesure de position angulaire.

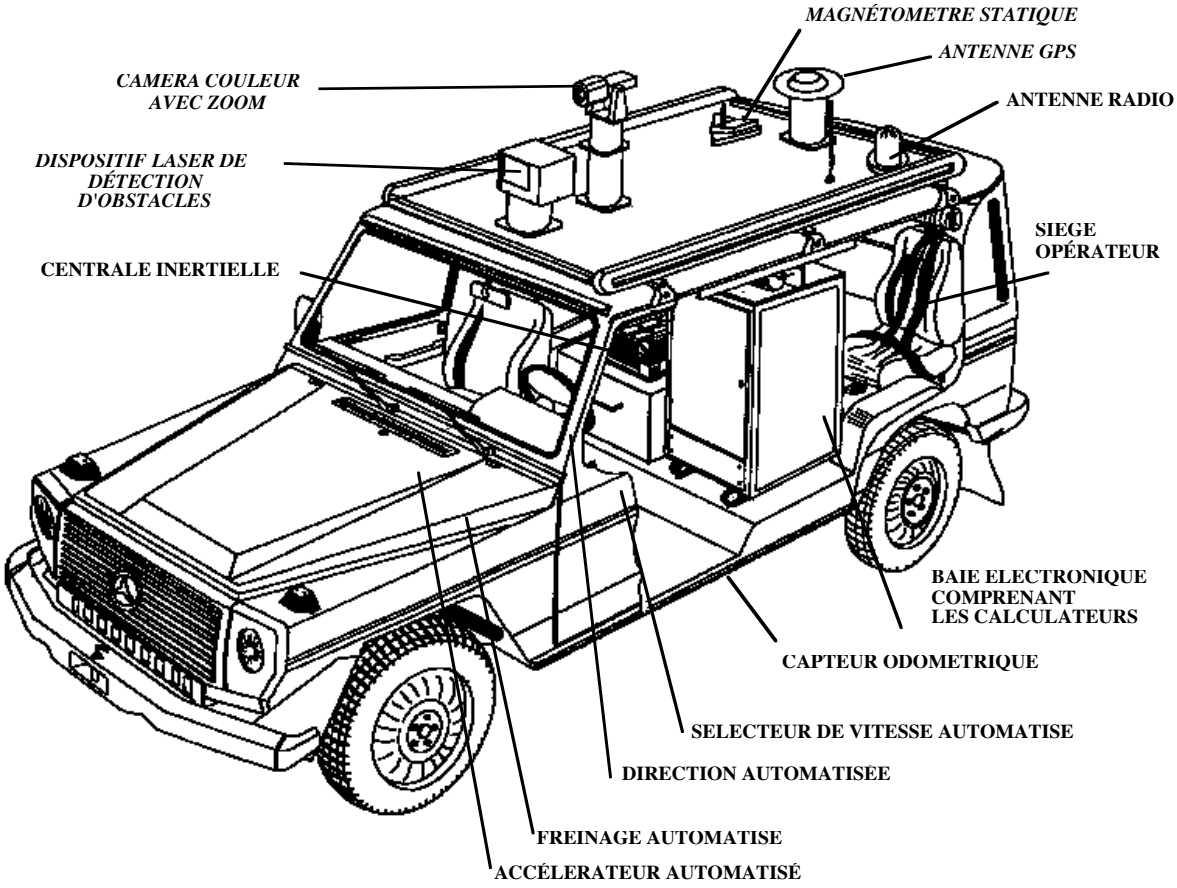


Figure 3.
Le véhicule REMI.

Capteurs.

La localisation est la fonction qui consiste à estimer, dans un repère de travail donné, certains paramètres de position et/ou d'attitude du véhicule nécessaires aux asservissements du pilotage-guidage. La méthode de localisation utilisée sur le véhicule REMI est dite " *à l'estime* ". Par les méthodes de localisation à l'estime, la position courante est déterminée par intégration de déplacements successifs orientés depuis la position de départ, à partir d'informations fournies par des capteurs généralement proprioceptifs qui peuvent être :
- de nature odométrique (mesure du déplacement relatif par rapport au sol),
- de nature inertielle (mesure des accélérations et/ou des rotations par rapport à un repère galiléen),
- de nature gravitationnelle (inclinomètres) ou magnétique (compas).

REMI dispose ainsi d'une *centrale inertielle*, et le système de transmission est équipé d'un *odomètre*. En raison de son principe même d'intégration simple ou double, et de l'existence inévitable d'erreurs de la part des capteurs, la localisation à l'estime dérive en fonction du temps et/ou de la distance parcourue. Elle exige donc d'être recalée périodiquement par diverses

méthodes, présentées dans [FAR93] par exemple. Nous n'entrons pas dans ces détails techniques, car le mode de fonctionnement utilisé pour l'expérimentation des systèmes de commande ne nécessite pas de recalage, comme nous allons le voir au paragraphe suivant.

Mode de commande pour l'expérimentation du guidage-pilotage : pseudo-autonome.

Nous effectuons l'expérimentation du système de commande avec REMI dans les conditions suivantes :

- avant le départ, une mission (c'est-à-dire la liste de points de passage et la contrainte temporelle associée) est communiquée au véhicule. Le véhicule n'utilisant pas de système de détection d'obstacles, cette mission ne sera plus modifiée ;
- la mission est d'une longueur telle que la dérive du système de localisation est négligeable.

Ce mode de fonctionnement est équivalent au mode téléplanifié. Les éléments qui n'interviennent pas en mode d'expérimentation du guidage-pilotage sont indiqués en italique sur la figure 3 (essentiellement les éléments utilisés pour la détection d'obstacles, dont une description est donnée dans [VDB93]). Comme nous l'avons vu, les asservissements latéral et longitudinal peuvent, dans une large mesure, être réalisés indépendamment. Pour faciliter encore l'expérimentation, ils sont aussi souvent testés séparément, c'est-à-dire qu'un opérateur à bord du véhicule commandera éventuellement freins et accélérateur si le but de l'expérience est de tester l'asservissement latéral, et inversement, cet opérateur pourra commander le volant si l'expérience porte sur l'asservissement longitudinal.

Chapitre 7

MODÉLISATION DU VÉHICULE REMI

INTRODUCTION.

Il est possible de dissocier les comportements dynamiques latéral et longitudinal par une approche statico-dynamique, qui consiste à considérer le comportement latéral du véhicule à vitesse stabilisée, puis son comportement longitudinal [BLO]. Nous présentons donc successivement et indépendamment les modélisations latérale et longitudinale. Nous montrons que ces modélisations bénéficient beaucoup d'une analyse préalable des phénomènes physiques mis en jeu. Cette démarche est en effet indispensable à l'élaboration d'un modèle-hypothèse (entrées, sorties, variables d'état du modèle, ordre…), même si la plupart des paramètres du modèle final n'ont pas de signification physique. Nous mettons également en évidence que, si certains de ces phénomènes sont suffisamment bien modélisés, il est aisé d'intégrer cette connaissance dans le réseau modèle.

I. MODÉLISATION LATÉRALE.

Un modèle du comportement latéral du véhicule, c'est-à-dire du comportement dynamique de son cap en réponse à la commande du volant, est nécessaire à l'élaboration du système de commande pour l'asservissement sur trajectoire du véhicule. Les caractéristiques de cette dynamique latérale sont essentiellement dues à :
- l'actionneur du volant,
- à la cinématique (donc à la géométrie) du véhicule,
- à l'inertie en lacet du véhicule.

La position du volant étant mesurée, nous avons pu modéliser séparément l'actionneur.

I.1. MODÉLISATION DE L'ACTIONNEUR DU VOLANT.

Les variables intervenant dans la description de l'actionneur du volant sont : en entrée, l'angle commandé α, et en sortie, l'angle du volant mesuré β_p. Cet actionneur est constitué d'un moteur à courant continu entraînant une courroie solidaire de la colonne de direction. Aux petits signaux, un modèle entrée-sortie du premier ordre linéaire décrit correctement le comportement de l'actionneur. Cependant, le dépouillement des essais fait également apparaître :
- une saturation de l'amplitude de l'angle du volant de l'ordre de : $\beta_{max} \approx 0{,}5\ rd$,

- une saturation de la vitesse de rotation de l'angle du volant de l'ordre de : $d\beta/dt_{max} \approx 0{,}2\ rd/s$.
(Ces valeurs des saturations en amplitude et en vitesse du *volant* correspondent en fait à celles des *roues*). Ces caractéristiques confèrent à l'actionneur une dynamique fortement non linéaire, qu'il importe de prendre en considération pour la conception d'un système de commande.

Réseau prédicteur.

L'hypothèse du premier ordre, et la prise en considération des saturations conduit à la définition d'un réseau prédicteur de la forme :
- pour une hypothèse NARX :

$$\beta(k+1) = \psi_{RN}\left(\beta_p(k), \alpha(k); C\right)$$

- pour une hypothèse NBSX :

$$\beta(k+1) = \varphi_{RN}\left(\beta(k), \alpha(k); C\right)$$

- pour une hypothèse NARMAX du premier ordre enfin :

$$\beta(k+1) = \varphi_{RN}\left(\beta_p(k), \alpha(k), e(k); C\right)$$

avec $e(k) = \beta_p(k) - \beta(k)$.

Séquences d'apprentissage et de test.

Ces séquences de mesures ont été effectuées pendant un asservissement du cap du processus sur un cap de consigne à l'aide d'un correcteur proportionnel à gain constant[1]. On dispose d'une dizaine d'enregistrements d'une durée moyenne de 30 secondes. La fréquence d'échantillonnage est de 25 Hz (pas d'échantillonnage ΔT=0,04 s), pour l'asservissement et l'enregistrement. La moitié des enregistrements environ correspond à des consignes en créneaux d'amplitude $\pm \pi/2$ rd, les autres sont relatifs à des consignes en créneaux de faible amplitude (entre 0,05 et 0,1 rd). Nous utilisons tous les enregistrements pour l'apprentissage, à l'exception de deux d'entre eux, un de chaque type, destinés à l'estimation de la performance.

Nous avons effectué l'identification à l'aide de deux types de réseaux, pour des motifs que nous allons exposer maintenant.

1) Réseau à neurones cachés avec fonction d'activation sigmoïdale.

L'identification à l'aide de prédicteurs bouclés et non bouclés conduit à une performance équivalente, pour le prédicteur utilisé bouclé. Ceci indique à la fois que le processus est assez peu bruité, qu'il n'est pas affecté par des perturbations déterministes non mesurées, et que l'identification est de bonne qualité. Les meilleurs résultats sont atteints avec un réseau prédicteur

---

[1] Nous avons justifié au chapitre 3 l'intérêt de l'identification en boucle fermée, qui permet l'exploration du domaine de fonctionnement désiré pour le système commandé ; nous négligeons le fait que l'entrée de commande ne doit plus être considérée comme une entrée exogène, mais comme la réalisation d'une variable aléatoire, pour l'identification.

NARX (non bouclé) possédant deux neurones cachés. Les figures 1a) et 1b) montrent la performance de ce réseau prédicteur, *utilisé bouclé*, sur les deux séquences de test.

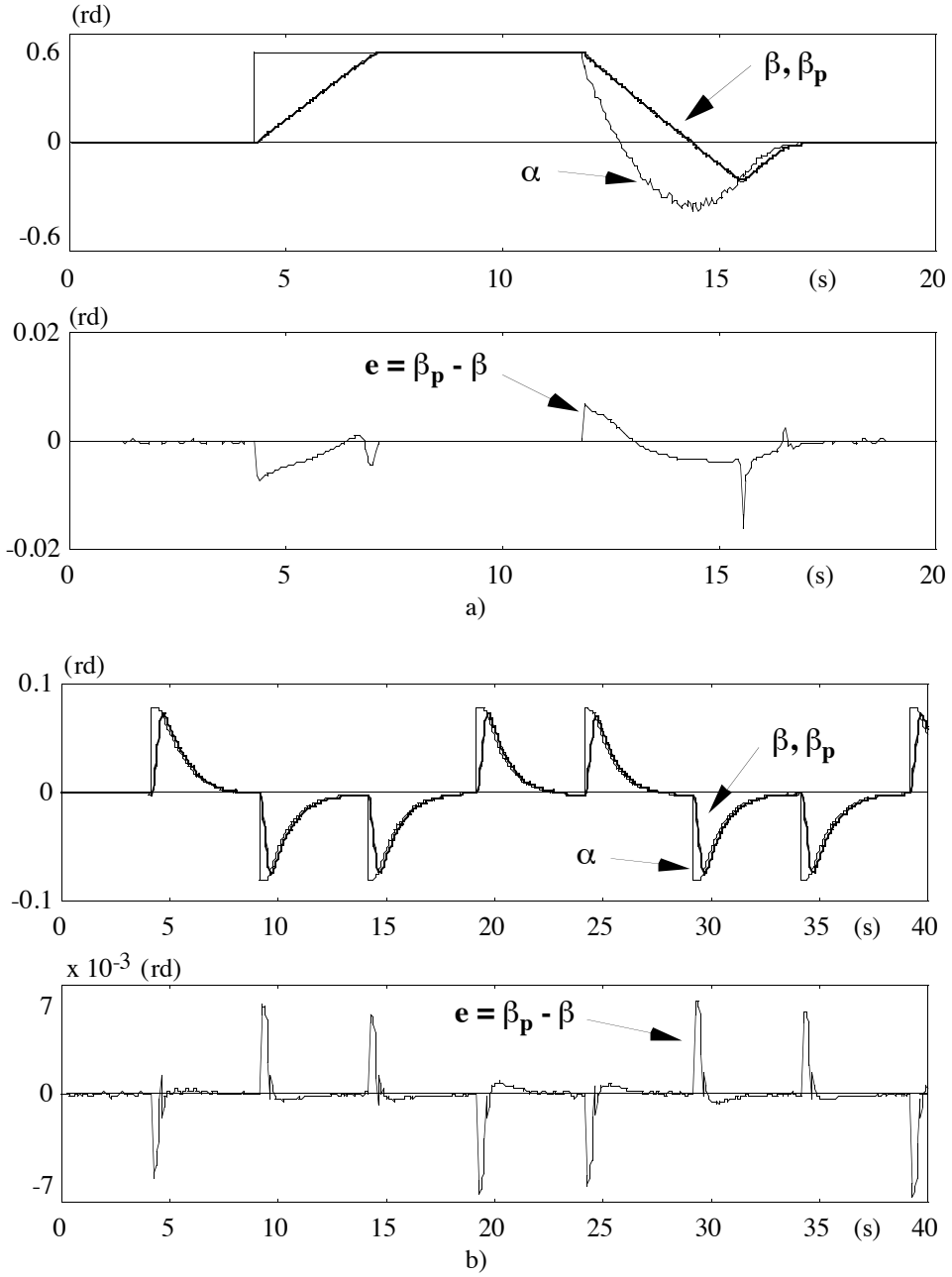


Figure 1.
Modélisation de l'actionneur du volant par un réseau prédicteur NARX :
test du prédicteur utilisé bouclé sur les séquences de test.

La figure 1a) correspond à la séquence de test obtenue pour de grandes amplitudes de consigne, et fait donc apparaître les deux saturations de l'actionneur, saturation en amplitude de l'angle du volant, et saturation de sa vitesse angulaire. La commande est bruitée, car calculée par un correcteur proportionnel ayant pour argument la vitesse du véhicule, dont l'estimation est entachée de bruit à haute vitesse.

La figure 1b) correspond à la séquence de test obtenue pour de petites amplitudes de consigne, et fait apparaître la saturation angulaire. On peut observer que la partie aléatoire de l'erreur (le bruit) est très faible relativement aux erreurs de modélisation déterministes.

À cause de la double saturation, il est intéressant, ici, d'analyser les coefficients du réseau obtenu. L'apprentissage fournit le réseau présenté en figure 2. Les valeurs des coefficients du réseau sont d'ordres de grandeur très divers : ceci est nécessaire pour représenter les saturations de l'actionneur. En effet, la valeur de la plus petite de ces saturations est d'environ $\dot{\beta}_{max} \times \Delta T \approx 0.2 \times 0.04 = 8\ 10^{-3}$, ordre de grandeur à comparer à l'amplitude des sigmoïdes (à valeurs dans]-1; 1[) et aux valeurs de β_p, dont les valeurs appartiennent à [-0,5; 0.5]. Sans connaissance *a priori* du système à identifier, il n'est donc pas conseillé de pénaliser l'amplitude des coefficients d'un réseau.

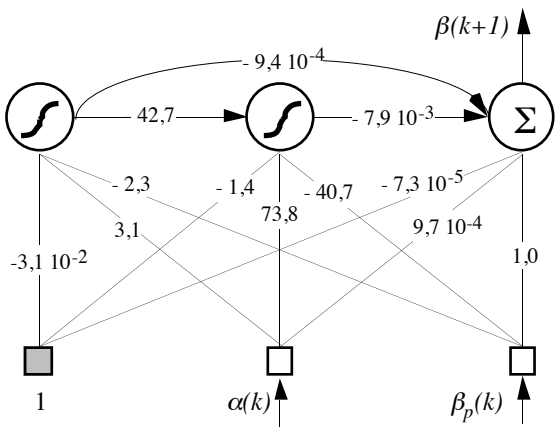


Figure 2.
Réseau prédicteur non bouclé à sigmoïdes pour la modélisation de l'actionneur du volant.

2) Réseau à neurones cachés avec fonction d'activation " saturation ".

Les premiers essais d'asservissement avec le véhicule ont nécessité plusieurs modifications de la période d'asservissement. Une transposition, même approchée, à un autre pas d'échantillonnage du réseau précédent étant pratiquement impossible (inconvénient de la modélisation à temps discret avec des modèles non linéaires signalé au chapitre 3), nous avons identifié un réseau rediscrétisable à volonté : ceci était possible en raison de la nature du processus, très bien approché par un premier ordre linéaire affecté de deux saturations. Nous avons effectué l'identification à l'aide d'un prédicteur constitué de deux neurones avec fonction d'activation de type saturation, dont les amplitudes ont été fixées d'après l'analyse des données. Le modèle obtenu est présenté sur la figure 3. À partir des coefficients obtenus, au même pas d'échantillonnage ΔT=0,04 s, on estime la valeur de la constante de temps à τ=0,1 s. La performance de ce nouveau réseau est presque aussi bonne que celle du réseau à sigmoïdes. L'objectif principal étant de prendre en considération les saturations de façon explicite pour la commande, cette performance est jugée suffisante. Ces caractéristiques sont en effet soit ignorées (voir par exemple [SHI90], où les auteurs prennent seulement la constante de temps de l'actionneur en considération, et non ses saturations), soit

considérées indirectement (par exemple avec un terme de pondération sur la grandeur de commande, ou de ses variations, dans la fonction de coût d'une commande LQ).

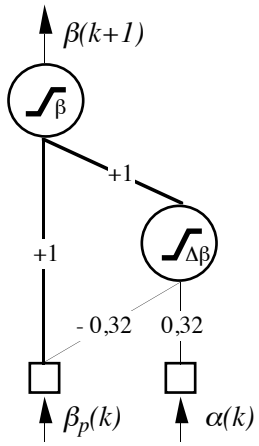


Figure 3.
Réseau prédicteur non bouclé à saturations (rediscrétisable) pour la modélisation de l'actionneur du volant avec $\Delta\beta_{max} = 0{,}008$; $\beta_{max} = 0{,}5$.

I.2. MODÉLISATION DU COMPORTEMENT LATÉRAL DE REMI.

Nous nous sommes intéressés à la modélisation dynamique latérale du véhicule sur terrain plat (les systèmes de commande synthétisés ont toutefois pu être utilisés en terrain accidenté, dans des conditions de pente et de dévers importants, comme nous le verrons au chapitre 8). Les variables choisies pour décrire le processus sont l'angle du volant β_p, le cap du véhicule ψ_p, et la position (x_p, y_p) du point situé au milieu de l'essieu arrière du véhicule, point choisi pour être asservi sur la trajectoire de consigne, ou *point de commande*. Justifions le choix de ce point, qui est moins manœuvrable que les autres points de l'axe longitudinal du véhicule[2]. Il présente entre autres les avantages suivants :
- il se situe à l'emplacement du point de localisation (la vitesse odométrique est celle de ce point),
- la direction de la vitesse en ce point est, à la dérive des pneus arrières près, celle de l'axe longitudinal du véhicule, et donc du cap de celui-ci.

I.2.1. Modèle cinématique d'un tricycle.

La modélisation cinématique d'un véhicule à quatre roues est souvent effectuée comme celle du tricycle de la figure 4, avec roue avant directrice et roues arrières motrices. Dans des conditions d'adhérence parfaite (roulement sans glissement), cette cinématique est régie par les équations à temps continu suivantes :

---

[2] Alors que l'orientation de la vitesse des autres points de l'axe dépend de l'angle des roues avant, et peut donc être commandée, la vitesse du point situé au niveau de l'essieu arrière est toujours orientée selon l'axe longitudinal.

$$\begin{cases} \dot{x} = v \cos \psi \\ \dot{y} = v \sin \psi \\ \dot{\psi} = \frac{v}{L} tg\ \beta \end{cases}$$

où x et y sont les coordonnées du milieu de l'essieu arrière du modèle, ψ est son cap (c'est-à-dire l'orientation de la vitesse du point de commande), β est l'angle de la roue directrice et v est la vitesse du milieu de l'essieu arrière. L est l'empattement du véhicule, c'est-à-dire la distance entre essieux.

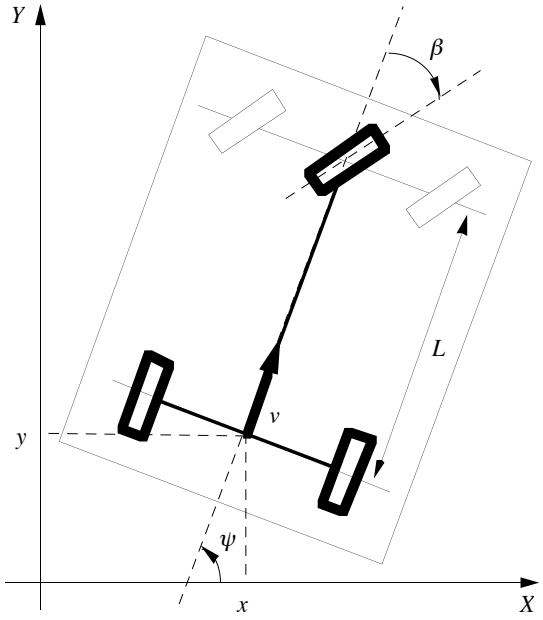


Figure 4.
Modèle tricycle : notations.

I.2.2. Modélisation neuronale.

En supposant une adhérence parfaite, les deux premières équations sont toujours vérifiées. En revanche, la variation de cap du véhicule en fonction de l'angle du volant dépend du mécanisme de direction. On se contente souvent du modèle cinématique du tricycle linéarisé :

$$\dot{\psi} = \frac{v}{L} \beta$$

Afin de disposer d'un modèle aussi précis que possible, et d'éviter toute démarche linéarisante, nous avons réalisé la modélisation de la relation reliant l'angle des roues β_p et le cap ψ_p.

Prédicteur neuronal.

Nous avons supposé que la variation de cap était effectivement proportionnelle à la vitesse du véhicule et inversement proportionnelle à son empattement (comme pour le tricycle). Nous avons donc réalisé l'identification avec un prédicteur non bouclé de la forme :

$$\psi(k+1) = \psi_p(k) + \frac{v_p(k)}{L} \psi_{RN}\left(\beta_p(k+1); C\right)$$

et un prédicteur bouclé :

$$\psi(k+1) = \psi(k) + \frac{v_p(k)}{L} \psi_{RN}\left(\beta_p(k+1); C\right)$$

Dans les deux cas, le sous-réseau réalisant la fonction ψ_{RN} soumis à apprentissage est statique.

Séquences d'apprentissage et de test.

Elles sont obtenues dans les mêmes conditions que les précédentes. La vitesse n'y varie qu'entre 0 et 5 m/s, mais le fait d'avoir imposé une dépendance en la vitesse multiplicative permet de se satisfaire de ces données (ψ_{RN} n'est pas fonction de la vitesse v_p).

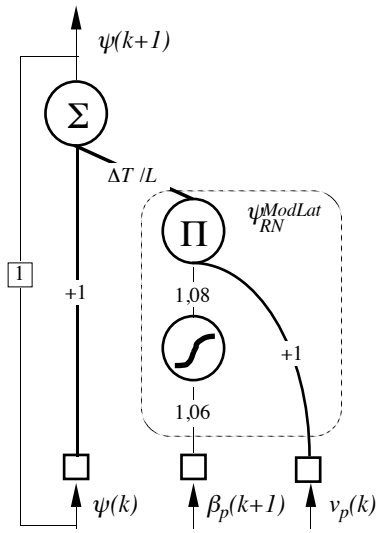


Figure 5.
Réseau prédicteur bouclé du comportement latéral de REMI.

L'identification à l'aide d'un prédicteur bouclé donne des résultats significativement meilleurs qu'un prédicteur non bouclé, ce qui s'explique par le caractère bruité de la mesure du cap ψ_p. Le processus s'écarte assez peu du comportement linéaire et est très correctement identifié avec un prédicteur possédant un seul neurone caché sigmoïdal. Le prédicteur est représenté sur la figure 5 : le réseau statique obtenu, appelé ψ_{RN}^{ModLat}, comprend en fait un neurone linéaire ; le " neurone Π " effectue un produit de ses entrées. Les connexions dont les coefficients ont été maintenus fixes pendant l'apprentissage sont représentées en traits gras.

Afin de comparer le modèle neuronal au modèle tricycle et à son linéarisé, nous avons représenté sur la figure 6 la fonction réalisée par le réseau statique, accompagnée des points d'apprentissage (la moitié d'entre eux). Pour chaque point d'apprentissage, on a calculé (et reporté sur le graphe en fonction de $\beta_p(k)$) la valeur :

$$\frac{L}{v_p(k)} \frac{\psi_p(k+1) - \psi_p(k)}{\Delta T}$$

Le modèle tricycle réalise la fonction tangente, et le réseau de neurones la fonction ψ_{RN}^{ModLat}.

On constate, d'après la figure 6, que les trois modèles sont assez proches, mais que le modèle neuronal épouse plus fidèlement les données. Cette modélisation permet également de s'assurer de l'absence d'erreurs d'échelle (nombreux rapports de transmission entre l'angle du volant et celui de la roue).

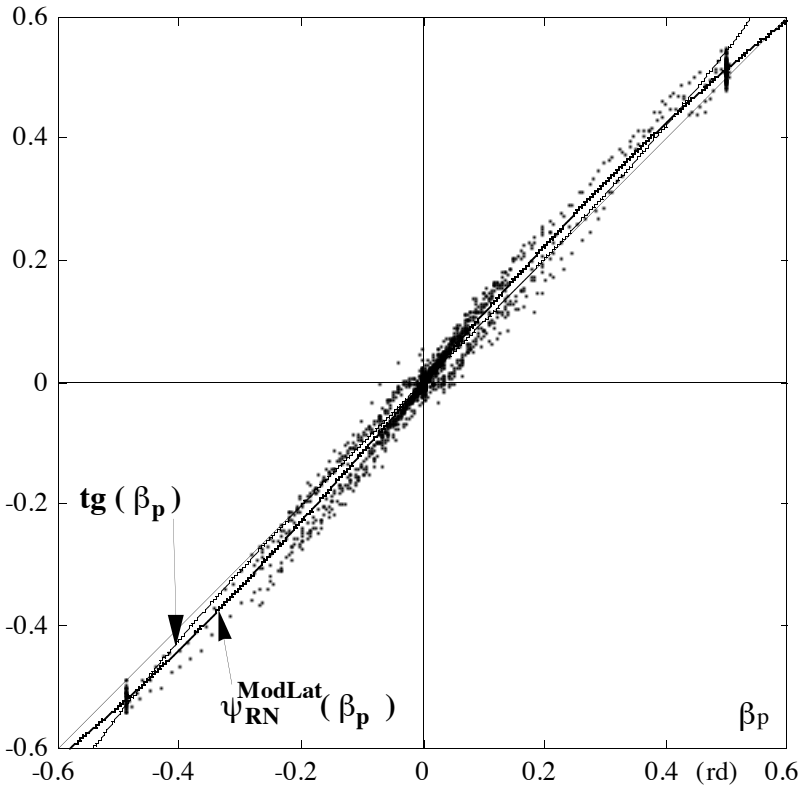


Figure 6.
Fonction réalisée par le réseau de neurones du prédicteur du comportement latéral de REMI.

Limites du modèle latéral neuronal.

Une modélisation complète eût demandé de considérer l'inertie en lacet du véhicule, c'est-à-dire le moment du véhicule par rapport à l'axe de lacet (l'axe vertical passant par le centre de gravité du véhicule). Si le véhicule est neutre, celui-ci se situe à égale distance des deux essieux. Dans ce cas, on obtient un modèle dynamique de la forme (modèle linéarisé aux petits angles β) :

$$J_z \frac{d^2\psi}{dt^2} = \frac{D\,L}{2}\,\beta - \frac{D\,L^2}{2\,v}\,\frac{d\psi}{dt}$$

où J_z est l'inertie en lacet, et D est la rigidité de dérive au niveau d'un essieu[3]. Si J_z=0, on retrouve bien le modèle linéarisé cinématique. Cette dynamique latérale ne sera prise en considération qu'au niveau de la commande, en limitant la vitesse pour éviter des accélérations latérales trop importantes.

---

[3] La force latérale $\overrightarrow{F_{lat}}$ s'exerçant au niveau des pneus est de la forme :

$$\left|\overrightarrow{F_{lat}}\right| = D\,\delta$$

où δ est l'angle de dérive des pneus.

I.3. MODÈLE LATÉRAL GLOBAL.

Le modèle global de la dynamique latérale du véhicule est représenté sur la figure 7. Nous notons les opérations réalisées par le réseau de simulation comme suit :

$$\begin{cases} \beta(k) = sat_{\beta_{max}}\left(\beta(k\text{-}1) + sat_{\Delta\beta_{max}}\left(a_{lat}\left(\alpha(k\text{-}1) - \beta(k\text{-}1)\right)\right)\right) \\ \psi(k) = \psi(k\text{-}1) + \dfrac{\Delta T}{L}\, v_p(k\text{-}1)\, \psi_{RN}^{ModLat}\left(\beta(k\text{-}1)\right) \end{cases}$$

où ψ_{RN}^{ModLat} est la fonction réalisée par le réseau non bouclé de la figure 5, et a_{lat}=0,32 (voir figure 3 représentant le modèle de l'actionneur).

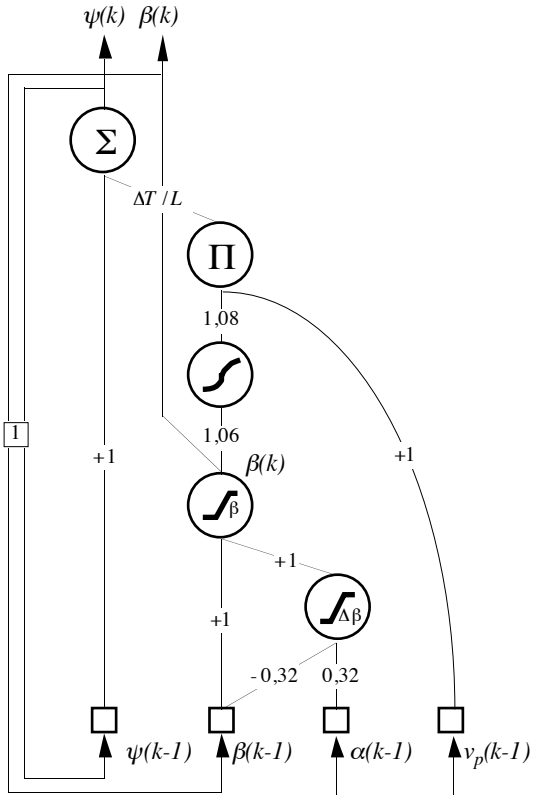


Figure 7.
Modèle dynamique latéral global de REMI.

Le modèle à temps discret du déplacement du point de commande est obtenu à l'aide de la méthode d'Euler (ces deux dernières équations ne sont pas représentées sur la figure 7) :

$$\begin{cases} x(k) = x(k\text{-}1) + v_p(k\text{-}1)\, \Delta T\ cos\ \left(\psi(k\text{-}1)\right) \\ y(k) = y(k\text{-}1) + v_p(k\text{-}1)\ \Delta T\ sin\ \left(\psi(k\text{-}1)\right) \end{cases}$$

Ce modèle a l'intérêt de prendre en considération explicite la dynamique de l'actionneur de la direction, d'être économe en nombre de neurones, et d'être plus précis que les modèles classiques utilisés auparavant (modèle cinématique du tricycle, ou son linéarisé). Comme nous l'avons signalé, ses limites sont dues au fait que l'inertie de lacet a été négligée. Pour la simulation, la vitesse v_p mesurée du véhicule sera remplacée par la vitesse simulée notée v.

Un exemple de résultat de modélisation complète (ψ en réponse à la commande α) par le modèle latéral bouclé est donné dans [RIV93b] et [RIV94].

II. MODÉLISATION LONGITUDINALE.

Pour concevoir un asservissement de la vitesse du processus, il est nécessaire de disposer de son modèle longitudinal, qui est un modèle de son comportement dynamique longitudinal en réponse aux commandes des actionneurs mis en jeu. Rappelons que l'asservissement de la vitesse utilise trois actionneurs :

1) L'action sur le papillon des gaz est réalisée à l'aide d'un moteur à courant continu, avec mesure de la position.

2) L'action sur le circuit hydraulique de freinage est réalisée à l'aide d'une servo-valve qui permet de modifier la pression dans le circuit hydraulique, avec mesure de cette pression.

3) Le changement de vitesse s'effectue à l'aide d'un moteur à courant continu agissant sur le sélecteur, avec mesure de la position angulaire du levier. Ajoutons que REMI possède une boîte automatique, et donc que l'on ne peut pas commander le passage des rapports avec le sélecteur.

Le comportement longitudinal d'un véhicule à moteur thermique met en jeu des phénomènes complexes au niveau de la chaîne propulsive (moteur thermique et système de transmission de la puissance). La modélisation de ce comportement dynamique vise surtout à la mise au point d'un modèle de simulation destiné à l'apprentissage d'un correcteur, et éventuellement aussi à servir de modèle interne au sein du système de commande. Il importe donc que ce modèle soit suffisamment représentatif, mais aussi suffisamment simple (qu'il soit décrit par un nombre raisonnable de variables d'état) pour rendre praticables un apprentissage ainsi qu'une utilisation en temps réel (comme modèle interne). Il est ainsi admis dans la littérature qu'un bon simulateur possède au moins neuf états, mais qu'un modèle à trois états suffit en général à l'élaboration d'une loi de commande satisfaisante [SHL78].

Nous allons maintenant présenter les éléments de modélisation physique qui ont permis d'établir partiellement la structure du modèle-hypothèse, puis la modélisation neuronale elle-même.

II.1. ÉLÉMENTS DE MODÉLISATION PHYSIQUE.

L'équation de la dynamique longitudinale du véhicule s'écrit, en projection selon l'axe longitudinal du véhicule :

$$M\,\gamma = M\,\frac{dv}{dt} = F_{TR} + F_{PERT}$$

où M est la masse du véhicule, γ est l'accélération longitudinale du véhicule, v est sa vitesse. F_{TR} est la force de traction appliquée au niveau des roues. F_{PERT} sont des forces perturbatrices. Notre modélisation physique est très inspirée de l'article de Hedrick [HED91] pour les aspects généraux, ainsi que de [MIN93b] pour ce qui concerne le véhicule REMI plus particulièrement.

II.1.1. Force transmise au niveau des roues.

Il s'agit de la force transmise par l'intermédiaire des pneus. Dans un large domaine de fonctionnement, cette force est proportionnelle au pseudo-glissement :

$$F_{TR} = K_r\, i$$

où K_r est la rigidité longitudinale des pneus, et i est le pseudo-glissement. Ce dernier s'exprime de la façon suivante :

$$i = \frac{R\,\omega_r - v}{R\,\omega_r}$$

où R est le rayon des roues, et ω_r est la vitesse angulaire des roues. La valeur du pseudo-glissement dépend du terrain et des pneus, mais on estime en général que sa valeur maximale est de 0,15.

Il faut, pour calculer ω_r, ou du moins déteminer de quels facteurs elle dépend, modéliser la chaîne propulsive et le freinage. La chaîne propulsive comprend le moteur (à combustion interne), le coupleur hydraulique, la boîte de vitesse, la boîte de transfert, les ponts et les roues, comme le montre la figure 8.

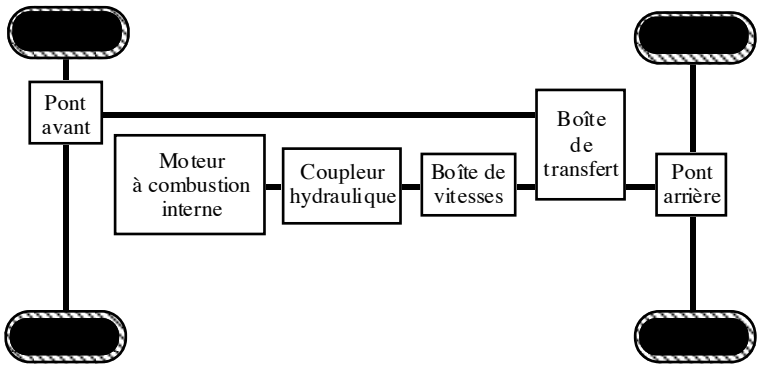


Figure 8.
Chaîne de propulsion.

Le couple moteur s'écrit :

$$J(r)\,\frac{d\omega_r}{dt} = \mathcal{M}_m - R\;F_{TR} - \mathcal{M}_f$$

où $\mathcal{M}_m$ est le couple dû au moteur en sortie de boîte. $\mathcal{M}_f$ est le couple de freinage dû aux plaquettes et tambours arrières. J est l'inertie des pièces tournantes, y compris celles du moteur, ramenées sur l'arbre des roues. J dépend donc du rapport engagé r. Son expression est :

$$J(r) = J_r + J_m\, r_{pont}\, r^2$$

où J_r est l'inertie des pièces tournant à la vitesse angulaire des roues, et J_m est l'inertie des pièces tournant à la vitesse de rotation du moteur ; r_{pont} est le rapport de transmission des ponts avant et arrière.

D'après l'expression du pseudo-glissement, la vitesse ω_r peut être reliée à la vitesse longitudinale du véhicule par la relation :

$$\omega_r = \frac{v}{R\left(1 - i\right)}$$

On considère que l'on a de façon approchée (comme $i \leq 0{,}15$) :

$$\omega_r = \frac{v}{R}$$

Exprimons maintenant la force F_{TR} :

$$F_{TR} = -\frac{J(r)}{R}\frac{d\omega_r}{dt} + \frac{\mathcal{M}_m - \mathcal{M}_f}{R}$$

En remplaçant ω_r par son expression en fonction de v :

$$F_{TR} = -\frac{J(r)}{R^2}\frac{dv}{dt} + \frac{\mathcal{M}_m - \mathcal{M}_f}{R}$$

II.1.2. Forces perturbatrices.

Les forces perturbatrices considérées ont trois origines : les résistances de l'air (F_{AIR}), au roulement (F_{ROU}), et en pente (F_{PEN}) :

$$F_{PERT} = F_{AIR} + F_{ROU} + F_{PEN}$$

II.1.2.1. Résistance de l'air.

L'expression de la résistance aérodynamique est la suivante :

$$F_{AIR} = -0{,}5\ \rho\ S\ C_x\ v_{air}^2$$

où ρ désigne la masse volumique de l'air, S la surface frontale du véhicule, C_x le coefficient de pénétration dans l'air, et v_{air} la vitesse du véhicule par rapport à la masse d'air. En l'absence de vent, on peut considérer cette force comme proportionnelle au carré de la vitesse du véhicule. Pour des vitesses inférieures à 60 km/h, la décélération qu'elle provoque est négligeable.

II.1.2.2. Résistance au roulement.

Cette résistance provient du travail de déformation des roues et de la chaussée. Cette résistance augmente théoriquement avec la vitesse. Cependant, elle varie très peu pour des vitesses inférieures à 60 km/h. Des essais en " roue libre " effectués avec REMI l'ont confirmé [MIN93b]. Ces essais consistent à mesurer la décélération du véhicule γ_{ROU} en ayant débrayé, et, connaissant la décélération due à la résistance de l'air, à en déduire la résistance au roulement par la relation :

$$F_{ROU} = M\ \gamma_{ROU}$$

Dans la plage des basses vitesses, la décélération observée est effectivement constante.

II.1.2.3. Résistance en pente.

La résistance en pente s'exprime en fonction de la pente p du terrain suivant la relation :

$$F_{PEN} = -M\ g\ \sin(p)$$

où g est l'accélération de la pesanteur. La pente p est estimée à l'aide de la centrale inertielle présente à bord du véhicule (qui fournit en fait le tangage de celui-ci, et non directement la pente du terrain).

II.1.3. Modèle global.

Réécrivons la relation fondamentale de la dynamique. Il vient :

$$\left(M + \frac{J(r)}{R^2}\right)\frac{dv}{dt} = \frac{\mathcal{M}_m - \mathcal{M}_f}{R} + F_{PERT}$$

Or nous avons vu que J(r) s'exprimait comme la somme d'une constante et du carré du rapport engagé : J(r) a une valeur plus importante en première qu'en quatrième. On peut donc remplaçer J(r) par un terme correctif K sur la masse du véhicule selon l'équation :

$$M + \frac{J(r)}{R^2} = M(1 + K)$$

En remplaçant également les forces perturbatrices par leurs expressions, on obtient finalement :

$$\frac{dv}{dt} = \frac{\mathcal{M}_m - \mathcal{M}_f}{R\,M(1+K)} - \frac{0{,}5\,r\,S\,C_x\,v^2}{M(1+K)} - \frac{F_{ROU}}{M(1+K)} - \frac{g\,sin(p)}{1+K}$$

Il reste à établir, pour obtenir le modèle global, les expressions du couple moteur $\mathcal{M}_m$ et du couple de freinage $\mathcal{M}_f$.

II.1.3.1. Couple moteur.

Le moteur de REMI est un moteur à essence équipé d'un système d'injection. On cherche T, le couple fourni par le moteur. Idéalement, la modélisation complète requiert la prise en considération de l'ouverture du papillon des gaz, de la vitesse de rotation du moteur, de l'avance à l'allumage, de la composition du mélange air-essence, et de la recirculation des gaz d'échappement. Pratiquement, seules les deux premières variables peuvent être facilement mesurées. Pour les autres, on est amené à faire les hypothèses suivantes :

- l'avance à l'allumage est optimale, quelles que soient les conditions de charge du moteur,
- le mélange air-essence est effectué aux conditions stœchiométriques, quelles que soient les conditions de fonctionnement,
- la recirculation des gaz d'échappement est négligée.

Sous ces hypothèses, le couple T n'est fonction que du régime moteur ω_m et de la masse de gaz d'échappement. On peut même ramener cette dépendance à une dépendance en θ, ouverture du papillon :

$$T = T(\omega_m, \theta)$$

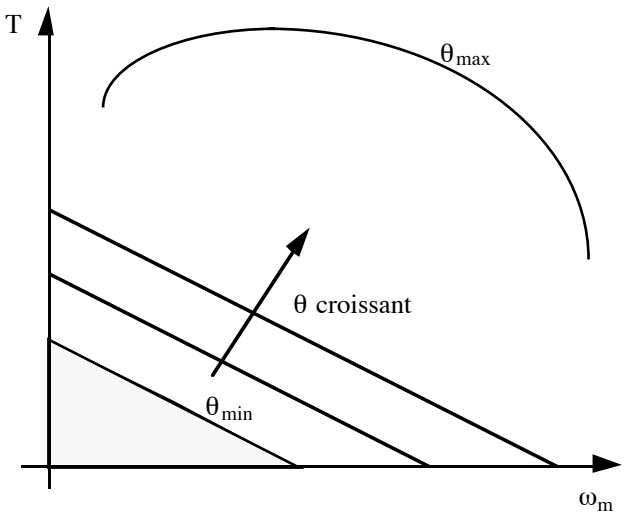


Figure 9.
Allure générale de la relation entre le couple moteur T, le régime moteur ω_m et l'ouverture papillon θ.

Pratiquement, même sous ces hypothèses restrictives, il est très difficile d'identifier cette relation approchée. Une solution consiste à faire effectuer l'identification du moteur par l'I. F. P. (Institut Français du Pétrole), qui dispose à cet effet de bancs d'essais. Les caractéristiques $T(\omega_m, \theta)$ que l'on peut obtenir par ce moyen ont en général, pour des moteurs à combustion interne du type de celui de REMI, la forme indiquée sur la figure 9. On observe une relation à peu près linéaire aux faibles ouvertures papillon et faible régime moteur [MIN93b], et une forme " en cloche " aux grandes ouvertures papillon (la seule courbe fournie par les constructeurs est en général la courbe à ouverture papillon maximale θ_{max}).

Finalement, le couple moteur en sortie de boîte, en considérant la boîte de transfert non engagée, a pour expression :

$$\mathcal{M}_m = \mu\, r_{pont}\, r\, \eta_{boîte}\, T\left(\omega_m, \theta\right)$$

où μ est un facteur de conversion entre le couple moteur et le couple transmis à la boîte, r_{pont} est la valeur du rapport du pont, r est celle du rapport de la boîte de vitesses, et $\eta_{boîte}$ est le rendement au niveau de la boîte de vitesses. Pour une boîte de vitesses automatique, la connaissance de r peut être obtenue en calculant le rapport entre le régime moteur et la vitesse du véhicule, celle-ci étant directement liée à la vitesse de rotation des roues, et donc à celle du moteur.

II.1.3.2. Couple de freinage.

On adopte en général pour un système de freinage du type de celui de REMI un modèle linéaire du premier ordre de l'action par les freins [HED91] :

$$\tau_{FR}\ \dot{\mathcal{M}}_f + \mathcal{M}_f = k_\pi\, \pi$$

où π est la pression dans le circuit de freinage, et k_π est une constante fonction des paramètres du système de freinage. La constante de temps de l'actionneur τ_{FR} est en général assez faible, et négligée : $\mathcal{M}_f = k_\pi\, \pi$

Des essais en freinage ont montré que cette hypothèse était justifiée pour REMI [MIN93b]. L'actionneur du freinage de REMI est en effet une servo-valve commandée en courant et constituée d'un moteur à courant continu très faible puissance.

II.1.3.3. Expression finale.

L'expression générale d'un modèle physique simplifié est donc :

$$\frac{dv}{dt} = \frac{\mu\, r_{pont}\, r\, \eta_{boîte}\, T\left(\omega_m, \theta\right)}{R\, M\left(1+K\right)} - \frac{k_\pi\, \pi}{R\, M\left(1+K\right)} - \frac{0{,}5\, r\, S\, C_x\, v^2}{M\left(1+K\right)} - \frac{F_{ROU}}{M\left(1+K\right)} - \frac{g\, \sin\left(p\right)}{1+K}$$

où ω_m s'exprime en fonction de la vitesse :

$$\omega_m = r_{pont}\, r\, \omega_r = r_{pont}\, v$$

À titre de comparaison, citons le modèle longitudinal du véhicule DARDS [FAR93], véhicule " jumeau " de REMI, modèle également destiné à servir pour la commande du véhicule (non pour la simulation). L'équation globale retenue est de la forme :

$$M\frac{dv}{dt} = K_1\, C_{mot} - K_2\, v - 0{,}5\, r\, S\, C_x\, v^2 - R_{rou} - M\ g\, \sin\left(p\right) - F_{frein}$$

où les constantes dépendent du rapport engagé, et où la modélisation du moteur (C_{mot}) est réalisée à l'aide d'une cartographie de puissance moyenne en fonction de l'ouverture papillon d'admission et de la vitesse de rotation du moteur. L'identification est effectuée à l'aide d'une série d'essais effectués dans des conditions connues de pente, de vent, etc.

Une modélisation analogue est également fournie dans [LIU93].

II.2. MODÉLISATION NEURONALE.

Cette première approche par une modélisation physique du véhicule nous a permis de formuler le modèle-hypothèse entrée-sortie non linéaire discret du premier ordre suivant :

$$v_p(k) = f\Big(v_p(k\text{-}1), \theta(k\text{-}1), \pi(k\text{-}1), r_p(k\text{-}1)\Big) - \frac{g\,\Delta T}{1 + K\left(r_p(k\text{-}1)\right)}\, sin\, p_p(k\text{-}1)$$

où v_p est la vitesse mesurée du véhicule (par l'odomètre), θ est la commande de la position angulaire du papillon, π est la commande de la pression dans le circuit de freinage ; r_p est le rapport transmission : il est estimé à partir du rapport de la vitesse du véhicule et de la vitesse de rotation du moteur, également mesurée en temps réel. $K(r_p)$ est le facteur d'inertie (connu) fonction du rapport engagé ; p_p est l'estimation de la pente du terrain donnée par la centrale inertielle placée à bord du véhicule. Pente et rapport engagé sont considérés comme des entrées perturbatrices mesurées.

Normalisation des entrées.

Afin de ne pas surestimer l'influence de certaines entrées au détriment d'autres pour des raisons purement numériques, il est toujours souhaitable de normaliser les ordres de grandeur des entrées s'ils sont très disparates.

- Le rapport r_p prend des valeurs, théoriquement quantifiées, comprises entre 1 et 4. Afin de faciliter l'apprentissage, nous avons normalisé cette valeur entre 1 (en première) et 0,25 (en quatrième). La variable utilisée est désignée par ρ_p, et s'exprime en fonction de r_p suivant la relation :

$$\rho_p(k) = 1 - r_p(k)$$

  Les valeurs qui sont fournies au réseau sont des valeurs quantifiées, correspondant aux valeurs des quatre rapports de vitesses soit :

$$\rho_p\left(1^{ère}\right) = 0{,}00\ ;\ \rho_p\left(2^{nde}\right) = 0{,}40\ ;\ \rho_p\left(3^{ème}\right) = 0{,}64\ ;\ \rho_p\left(4^{ème}\right) = 0{,}75\ .$$

- Les entrées de commande θ et π sont normalisées entre 0 et 1 et -1 et 0 respectivement. Comme nous avons imposé que l'asservissement de vitesse ne conduise jamais à une utilisation simultanée du frein et de l'accélérateur, nous aurions pu n'utiliser qu'une entrée de commande. Cependant, conserver deux entrées distinctes permet d'économiser beaucoup de neurones cachés, car les caractéristiques des deux entrées sont très différentes. Les commandes θ et π seront néanmoins représentées sur le même graphe.
- Enfin, la vitesse est divisée par dix, et prend ainsi des valeurs comprises entre 0 et 2 dans la plage de vitesses considérée (0-20 m/s soit 0-72 km/h).

Prédicteur neuronal.

Nous avons utilisé des prédicteurs neuronaux bouclés (NBSX) de la forme :

$$v(k+1) = \varphi_{RN}\left(v(k), \theta(k), \pi(k), \rho_p(k); C\right) - \frac{g\,\Delta T}{1 + K\left(\rho_p(k)\right)} sin\left(p_p(k)\right)$$

et des prédicteurs non bouclés (NARX) :

$$v(k+1) = \psi_{RN}\left(v_p(k), \theta(k), \pi(k), \rho_p(k); C\right) - \frac{g\,\Delta T}{1 + K\left(\rho_p(k)\right)} sin\left(p_p(k)\right)$$

Les prédicteurs sont de type complètement connecté, à neurone de sortie à fonction d'activation particulière (que nous notons Id+) :

$$\begin{cases} Id+(v) = v & v \geq 0 \\ Id+(v) = 0 & v < 0 \end{cases}$$ avec, pour l'apprentissage, la dérivée : $$\begin{cases} Id+'(v) = 1 & v \geq 0 \\ Id+'(v) = 0 & v < 0 \end{cases}$$

Cette fonction d'activation impose au véhicule de ne pas se mouvoir en arrière, ce qui correspond bien à une réalité mécanique. L'utilisation de cette fonction d'activation nous a permis de réaliser l'identification du processus à l'aide de réseaux beaucoup plus petits que ceux possédant un neurone de sortie à fonction d'activation linéaire[4].

Séquences d'apprentissage et de test.

Nous avons disposé, pour l'identification, d'une dizaine d'enregistrements effectués pendant l'asservissement longitudinal du véhicule sur une vitesse de consigne variant de 0 à 15 m/s, avec divers correcteurs (PID linéaire, NID[5] neuronal, commande avec modèle interne…). La figure 10 présente un exemple typique de ces enregistrements. D'autres enregistrements, à basse vitesse uniquement, ont aussi été utilisés pour l'apprentissage. La pente du terrain est nulle pour toutes les séquences d'apprentissage.

Résultats d'identification.

Un prédicteur bouclé donne une meilleure performance qu'un modèle de simulation obtenu à partir d'un prédicteur non bouclé. La figure 10 représente une séquence de test. La vitesse v du modèle est représentée en trait fin, et celle du processus v_p en trait gras. L'erreur est relativement importante, ce qui explique la meilleure performance de modèles de simulation obtenus à partir de prédicteurs bouclés que de prédicteurs non bouclés. En effet, étant donnée la simplicité du modèle, la plus grande part de la variance de l'erreur est due à une insuffisance de la modélisation de la partie déterministe du processus, et non aux bruits qui l'affectent. Or un réseau non bouclé ne doit être préféré à un réseau bouclé que si la modélisation déterministe peut être parfaite, et que le bruit est de type bruit d'état. Ceci n'étant pas le cas, une identification avec un prédicteur non bouclé peut conduire à un modèle de simulation dont la sortie diverge totalement de celle du processus.

---

[4] Si le neurone de sortie est linéaire, il faut des neurones cachés supplémentaires pour interdire à la sortie de prendre des valeurs négatives.

[5] NID : correcteur Non linéaire Intégral et Dérivé (cf. chapitre 5).

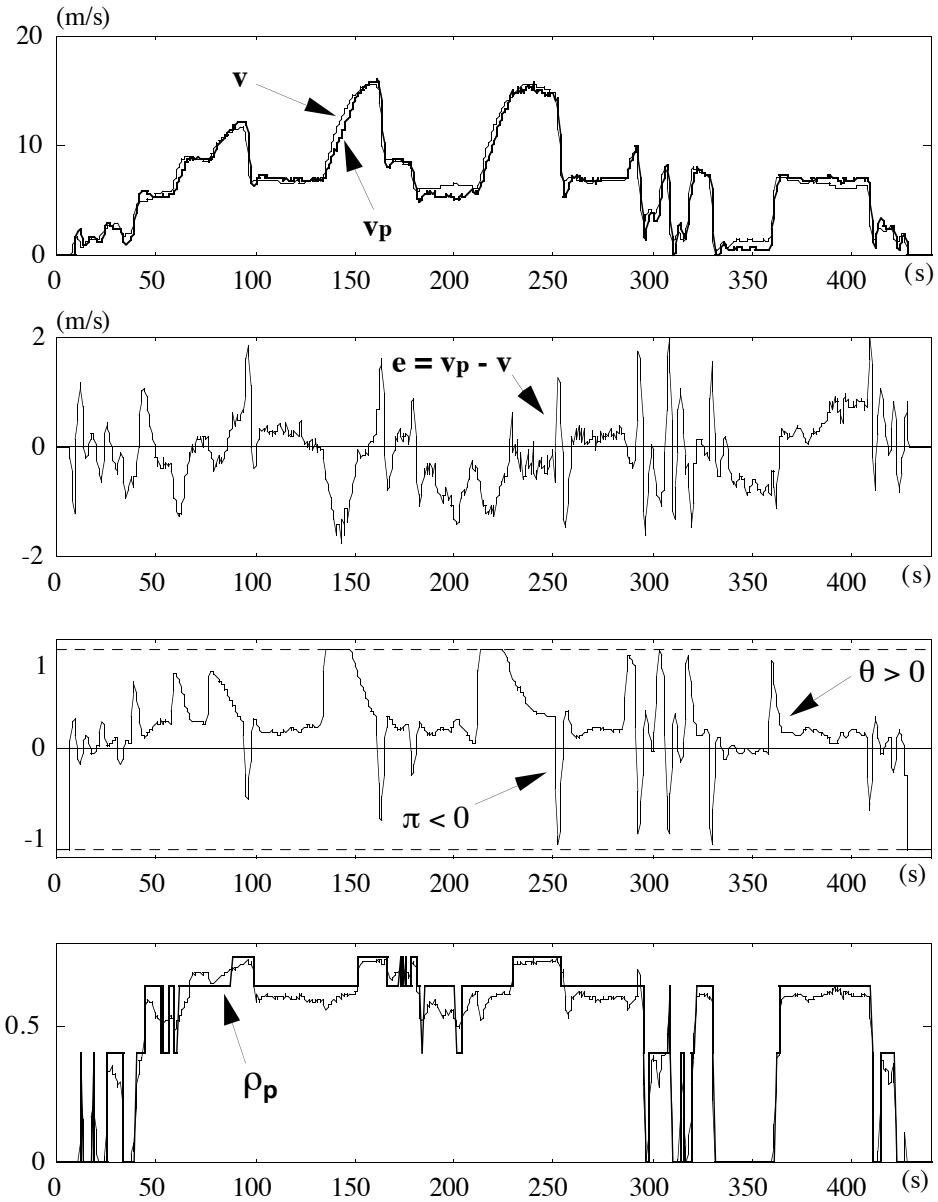


Figure 10.
Modélisation du comportement longitudinal de REMI : performance sur une séquence de test d'un réseau bouclé complètement connecté à trois neurones cachés et à neurone de sortie à fonction d'activation Id+.

Les résultats de la figure 10 sont obtenus avec un réseau possédant 3 neurones cachés, parcimonie qui a donc été atteinte grâce aux choix particuliers effectués (choix et normalisation des entrées, fonction d'activation du neurone de sortie), sans lesquels le nombre de neurones cachés nécessaire à l'obtention de la même performance est de 7 environ. L'erreur quadratique moyenne est de 0,47 sur la séquence de test présentée, et de 0,45 sur l'ensemble d'apprentissage (l'unité de vitesse est le m/s). L'ajout de neurones cachés améliore un peu la performance sur l'ensemble d'apprentissage, mais reste sans effet sur le test.

Nous n'avons pas identifié spécifiquement le comportement en marche arrière. Celui-ci est simulé à l'aide du même modèle, avec le rapport de la première, en inversant le signe de la vitesse obtenue (cette inversion de signe est réalisée dès que la position du sélecteur impose la marche arrière). Cette solution s'est révélée satisfaisante du point de vue de la performance en commande obtenue en utilisant le même modèle pour les simulations des déplacements en marche avant et

arrière. De même, l'influence de la pente, telle que le modèle physique l'impose, a été conservée sans procéder à une identification spécifique.

Globalement, le modèle obtenu procure une performance similaire à celle du modèle établi précédemment pour REMI. Il offre cependant l'avantage d'une plus grande simplicité et compacité que ce dernier, et il est ainsi préféré pour la simulation en laboratoire. Au chapitre 8 consacré à la commande de REMI, le modèle de simulation obtenu sera noté :

$$v(k) = \varphi_{RN}^{ModLon}\Big(\quad v(k\text{-}1), \theta(k\text{-}1), \pi(k\text{-}1), \rho(k\text{-}1)\Big) - \frac{g\,\Delta T}{1 + K\left(\rho(k\text{-}1)\right)}\, sin\left(p(k\text{-}1)\right)$$

où φ_{RN}^{ModLon} est la fonction réalisée par la partie non bouclée du réseau de neurones, ses coefficients étant maintenant fixés.

Limites du modèle longitudinal neuronal.

Le comportement du modèle devient assez approximatif aux grandes ouvertures papillon et à grande vitesse (au dessus de 15 m/s), bien que le gain statique soit correct : le modèle est plus rapide que le processus. Ceci est notamment dû au fait que les séquences d'apprentissage contiennent trop peu de points dans ce domaine de fonctionnement. Pour améliorer la performance, il faudrait réaliser des expériences beaucoup plus systématiques. L'idéal serait évidemment de faire réaliser par des spécialistes l'identification de la relation exprimant le couple moteur en fonction du régime moteur ω_m et de l'ouverture papillon θ.

CONCLUSION.

La modélisation du comportement latéral et longitudinal d'un véhicule qui vient d'être présentée est un bon exemple d'utilisation des réseaux de neurones en tant que modèles dynamiques. Elle a permis d'élaborer un modèle fidèle du véhicule, meilleur que les modèles classiques existants (dans le cas latéral), ou comparable à ceux-ci (dans le cas longitudinal), et d'une remarquable compacité.

Cet exemple pratique de modélisation met surtout en lumière les faits suivants :

- la démarche neuronale peut tirer le même profit d'une étude physique des phénomènes mis en jeu que les approches classiques ;
- la modélisation physique conduit, selon les cas, à un modèle boîte noire (longitudinal) ou à un modèle dont la majeure partie des paramètres gardent une signification physique (latéral) ;
- les connaissances *a priori* peuvent facilement être intégrées dans un modèle neuronal en segmentant celui-ci (latéral), en faisant usage de neurones à fonction d'activation *ad-hoc*, par exemple une saturation ou un produit, et éventuellement en fixant la valeur de certains coefficients.

Chapitre 8

COMMANDE DU VÉHICULE REMI

INTRODUCTION.

Comme nous l'avons annoncé dans l'introduction, nous avons choisi comme mode de commande un asservissement latéral sur trajectoire, ou " path-following ", avec asservissement longitudinal indépendant. Justifions rapidement le choix de ce mode de commande. On l'oppose en général à l'asservissement sur la posture d'un robot de consigne virtuel, ou " path-tracking ", mode qui demande d'agir de manière coordonnée sur la vitesse et sur la direction du robot. D'un point de vue théorique, les deux modes de commande sont équivalents, au sens où l'on peut garantir la stabilisation du robot sur la trajectoire ou sur la posture de consigne, tant que la vitesse de consigne ne devient pas asymptotiquement nulle. Néanmoins, il est beaucoup plus simple de concevoir des asservissements latéral et longitudinal indépendants, qu'un asservissement sur une posture en mouvement le long de la trajectoire de consigne à la vitesse imposée, ou robot virtuel. Un autre argument en faveur de ce choix est que le suivi de la trajectoire est évidemment impératif, et prime devant le suivi de la vitesse. Or, dans le cas de l'asservissement sur posture, une perturbation provoquant un retard du processus sur le robot virtuel se traduirait inévitablement par un " raccourci " inacceptable par rapport à la trajectoire de celui-ci.

I. ASSERVISSEMENT LATÉRAL.

Enjeux de l'asservissement latéral.

Le problème de l'asservissement latéral a été largement étudié, en particulier par Claude Samson [SAM90] [SAM91a, b, &c] [SAM92]. L'auteur a établi qu'il est possible de stabiliser sur une trajectoire de consigne un robot mobile de type tricycle par retour d'état statique continu, tant que la vitesse du robot ne tend pas vers zéro, et donne dans [SAM92] un exemple d'une telle commande. Cependant, les lois de commande mises au point sont conçues pour des systèmes idéaux, c'est-à-dire purement cinématiques, comme le tricycle présenté au chapitre 7 (où v et β sont les entrées de commande). Elles ne peuvent donc être utilisées telles quelles pour la commande de systèmes réels, dont les actionneurs ont une dynamique non négligeable. Ainsi, une commande non linéaire par retour d'état statique issue de [SAM92] a été testée sur REMI sans qu'il y ait d'amélioration sensible par rapport à une commande linéaire, car elle tient compte de la non-linéarité (négligeable) de la cinématique, mais non de celle des actionneurs [MIN93a]. L'enjeu de la présente étude réside dans la mise en œuvre d'une loi de commande stabilisante, prenant en considération les non-linéarités du

véhicule réel, en effectuant l'apprentissage du correcteur latéral à l'aide du modèle de simulation latéral neuronal présenté au chapitre 7. Nous verrons plus loin que le problème terminal (qui se pose lorsque la vitesse de consigne s'annule) est résolu dans la pratique. En effet, que ce soit par une stratégie de path-following ou par une stratégie de path-tracking, la stabilisation sur une posture n'est effective à l'aide d'un retour d'état statique que si celui-ci est instationnaire, ou bien discontinu. Une solution instationnaire est proposée dans [SAM91c] [SAM92], une solution discontinue l'est dans [CAN91] ou [SOR92]. Une solution hybride (discontinue, et instationnaire au voisinage de la posture de consigne) est donnée dans [POM92].

Position du problème.

Soit une trajectoire de consigne (C) quelconque, respectant toutefois les contraintes non holonomes[1] du véhicule, en particulier la courbure maximale admissible de la trajectoire. Le problème de l'asservissement sur trajectoire consiste à déterminer une loi de commande pour l'angle du volant afin d'imposer au véhicule (i) de rejoindre et de suivre la trajectoire de consigne, et (ii) d'avoir son cap tangent à cette trajectoire quand il la suit. Nous avons choisi de réaliser l'apprentissage d'un régulateur stabilisant le véhicule sur une trajectoire rectiligne. Cette trajectoire rectiligne est ensuite choisie comme une tangente à la trajectoire réelle de consigne en un point appelé *point-cible*. Cette approche, fondée sur l'asservissement de la posture du véhicule, a été développée dans [RIV93a] et [RIV93b] (" posture-based approach "). Elle y est comparée à une approche fondée sur l'asservissement du cap du véhicule, respectant la séparation traditionnelle guidage-pilotage (" heading-based approach "), qui s'est révélée moins performante. Nous présentons maintenant l'apprentissage du régulateur neuronal correspondant à l'asservissement sur trajectoire ; nous considérons ensuite le choix du point-cible.

---

[1] Une *contrainte non holonome de type égalité* s'exprime par une contrainte scalaire non intégrable sur les paramètres de configuration (x, y et ψ pour la cinématique d'un robot se déplaçant dans un plan) et leurs dérivées ; elle n'affecte pas la dimension de l'espace des configurations, mais réduit la dimension de l'espace des vitesses. Par exemple, un véhicule tel que REMI peut adopter n'importe quelle configuration puisqu'il est commandable, mais ne peut se mouvoir parallèlement à l'axe de ses roues arrière. En supposant qu'il est parfaitement décrit par le modèle tricycle, ceci est dû à la contrainte non holonome de roulement pur qui s'écrit : $- x' \sin\psi + y' \cos\psi = 0$, et qui limite l'espace des vitesses au plan ainsi défini.

Une *contrainte non holonome de type inégalité* limite la vitesse à une partie de l'espace des vitesses. Par exemple, la limitation du rayon de courbure d'un chariot par la saturation en amplitude de l'angle de la roue restreint l'ensemble des vitesses possibles à un cône dans l'espace des vitesses [LAT91].

I.1. APPRENTISSAGE D'UN RÉGULATEUR NEURONAL DE STABILISATION SUR TRAJECTOIRE.

La trajectoire de consigne est la droite du plan d'équation y=0 (l'axe des abscisses), orientée vers les x positifs. Le but de l'asservissement est de stabiliser le véhicule sur cette trajectoire, c'est-à-dire de réguler les variables d'état ψ_p, le cap du véhicule, et y_p, son ordonnée. Puisque les consignes sont nulles, ces variables représentent aussi l'erreur de cap et l'erreur latérale. Nous traitons ce problème de régulation par la méthode de commande présentée au chapitre 5 §I, qui est donc une extension de la régulation LQ à horizon infini à des modèles non linéaires.

Modèle de simulation.

Les variables d'état du modèle sont le cap ψ, l'angle du volant β, l'abscisse x et l'ordonnée y du point de commande. La commande de l'angle du volant est notée α. Nous rappelons le modèle de simulation obtenu à l'issue de l'identification latérale :

$$\begin{cases} \beta(k) = sat_{\beta_{max}}\left(\beta(k\text{-}1) + sat_{\Delta\beta_{max}}\left(a_{lat}\left(\alpha(k\text{-}1) - \beta(k\text{-}1)\right)\right)\right) \\ \psi(k) = \psi(k\text{-}1) + \dfrac{\Delta T}{L}\, v(k\text{-}1)\, \psi_{RN}^{ModLat}\left(\beta(k\text{-}1)\right) \\ x(k) = x(k\text{-}1) + v(k\text{-}1)\, \Delta T\ cos\ \left(\psi(k\text{-}1)\right) \\ y(k) = y(k\text{-}1) + v(k\text{-}1)\ \Delta T\ sin\ \left(\psi(k\text{-}1)\right) \end{cases}$$

La vitesse v est considérée comme une perturbation mesurée. Le modèle et ses variables d'état sont représentés sur la figure 1. On note z=[β ψ y] les variables d'état d'intérêt pour la régulation.

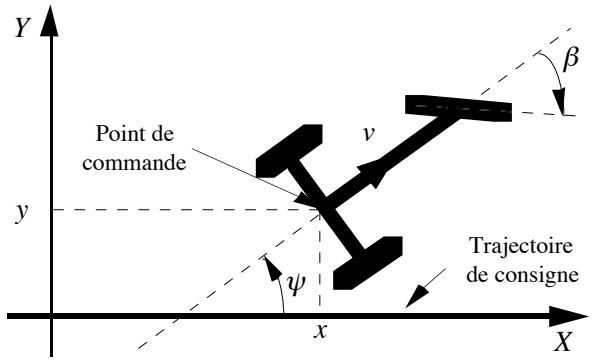


Figure 1.
Stabilisation sur la trajectoire de consigne y=0.

Régulateur neuronal.

Le réseau de neurones correcteur réalise un retour d'état statique, aussi fonction de la perturbation de vitesse :

$$\alpha(k) = \psi_{RN}\left(\beta(k),\, \psi(k),\, y(k),\, v(k);\, C\right)$$

Il n'y a pas de retour sur l'abscisse x, puisque l'asservissement longitudinal est indépendant. La non-linéarité du neurone de sortie est choisie sigmoïdale, à valeurs dans l'intervalle [$-\beta_{max}$; β_{max}], où β_{max} est l'amplitude maximale de l'angle du volant.

Séquences d'apprentissage.

Les séquences d'apprentissage sont constituées de N_Z trajectoires de N pas d'échantillonnage correspondant à une initialisation aléatoire (distribution uniforme) de l'état z(0) dans les intervalles [$-\beta_{max}$; β_{max}] pour l'angle du volant β (rd), [$-\pi$; π] pour le cap ψ (rd), et [-10; 10] pour l'ordonnée y (m) (l'abscisse initiale est toujours prise nulle, puisqu'elle n'intervient pas dans le problème). La vitesse est constante sur chaque trajectoire, initialisée aléatoirement dans l'intervalle [0; 20] (m/s).

Fonction de coût.

Il n'est pas nécessaire de faire intervenir la commande α dans la fonction de coût, puisque la sortie du régulateur est limitée (par une sigmoïde) :

$$J = \frac{1}{N_Z} \sum_{z(0)\in Z} \left(\sum_{k=1}^{N} z^T(k)\ W\ z(k) \right) = \frac{1}{N_Z} \sum_{z(0)\in Z} \left(\sum_{k=1}^{N} w_\beta\ \beta(k)^2 + w_\psi\ \psi(k)^2 + w_y\ y(k)^2 \right)$$

Faire intervenir l'angle du volant β reviendrait à pondérer la commande : nous avons donc choisi $w_\beta=0$. Pour déterminer la meilleure pondération du cap ψ et de l'ordonnée y, nous avons procédé de manière itérative, en imposant $w_y=1$ et en faisant varier w_ψ. Comme nous le verrons (figure 3), ce choix a une influence très importante sur l'allure des trajectoires.

Algorithme d'apprentissage.

L'algorithme est nécessairement semi-dirigé (cf chapitre 5 §I.3.1). Le système d'apprentissage est représenté sur la figure 2.

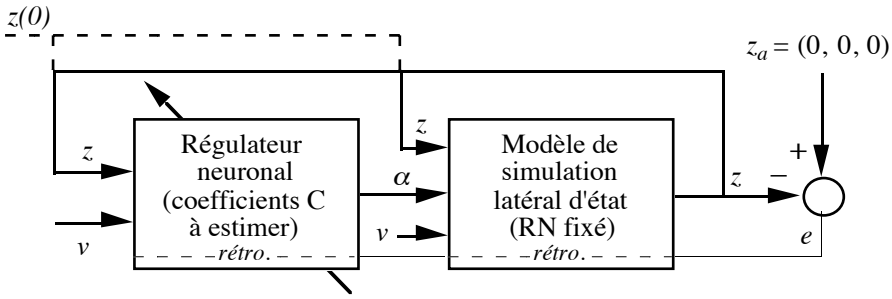


Figure 2.
Système d'apprentissage pour l'asservissement sur trajectoire rectiligne (y=0).

Plusieurs apprentissages ont donc été effectués pour déterminer les coefficients de pondération. La figure 3 présente le comportement du modèle commandé résultant d'apprentissages du régulateur pour trois valeurs différentes de w_ψ : 1, 10 et 50. Ces trajectoires de test, représentatives de la dynamique de ralliement, ont été obtenues en initialisant le modèle neuronal du véhicule sur la trajectoire avec un cap variant entre $-\pi$ et $+\pi$ et un angle du volant de 0 rd, à une vitesse constante de 10 m/s (la forme des trajectoires varie assez peu en fonction de la vitesse). Ces simulations nous ont conduit au choix de la pondération suivante : $w_\beta = 0$; $w_\psi = 10$; $w_y = 1$. La performance ne s'améliore plus à partir de six neurones cachés.

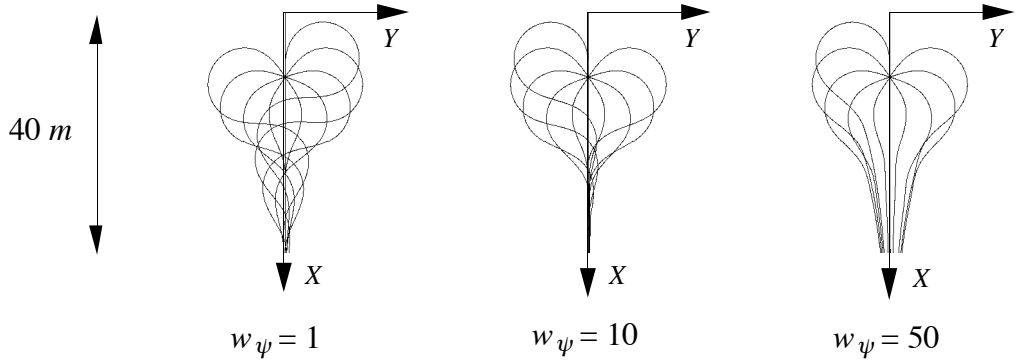


Figure 3.
Influence de la pondération des variables d'état dans la fonction de coût sur la dynamique de ralliement de la trajectoire.

I.2. CHOIX D'UN POINT-CIBLE.

Comme cela a été dit, le point-cible est le point de la trajectoire (C) choisi à chaque instant pour déterminer la tangente à la trajectoire sur laquelle s'asservir. Si l'on choisit comme point-cible le point de (C) le plus proche du véhicule, il faut tenir compte de la courbure de la trajectoire en ce point. La solution la plus simple consiste à ajouter un terme de boucle ouverte de la forme[2] :

$$\alpha_c(k) = L\ \kappa(k)$$

où κ(k) est la courbure de (C) au point-cible défini à l'instant k. Cette solution est utilisée par exemple par [RIN93] [VDM93]. Elle se révèle cependant peu performante dans le cas de courbures importantes de la trajectoire (nous l'avons expérimentée). En effet, la relation ci-dessus suppose que le robot se trouve déjà sur la trajectoire, donc dès que ce n'est plus le cas, typiquement dans le cas de virages importants, la loi de commande est inadéquate[3].

Sans remettre en question l'apprentissage sur trajectoire de courbure nulle effectué (il est possible de réaliser un apprentissage sur des cercles de courbure variant entre 0 et κ_{max}, en fournissant la courbure en entrée du réseau), il est un autre moyen de tenir compte de la courbure, cette fois de façon implicite, tout en évitant les inconvénients de l'approche fondée sur la courbure. Ce moyen consiste à choisir le point-cible non pas au plus près sur la trajectoire, mais à une certaine distance D en avant du véhicule, afin d'anticiper les virages éventuels. Ce choix du point-cible et de la tangente à la trajectoire en ce point est représenté sur la figure 4.

La distance D est choisie proportionnelle à la vitesse : $D = F_D\ v_p$. Le facteur de proportionnalité F_D est fixé en simulation, en faisant évoluer le véhicule simulé sur des trajectoires réelles cartographiées. Les simulations permettent de fixer $D = v$ (soit F_D=1), où v est la vitesse du modèle.

---

[2] Cette relation est obtenue à partir de la variation de cap du modèle linéarisé. Si l'on suppose que ce modèle suit la trajectoire, on a : dψ/dt = ds/dt 1/L α, soit α = L dψ/ds = L κ (où s est l'abscisse curviligne le long de la trajectoire).

[3] Pour éviter cet inconvénient, la loi de commande donnée dans [SAM92] comporte un terme sur la courbure qui s'écrit : $\dot{\alpha_c} = v\ \kappa\ cos\left(\psi\right) / \left(1 - \kappa\ y\right)$. Ce terme varie donc en fonction de l'erreur latérale.

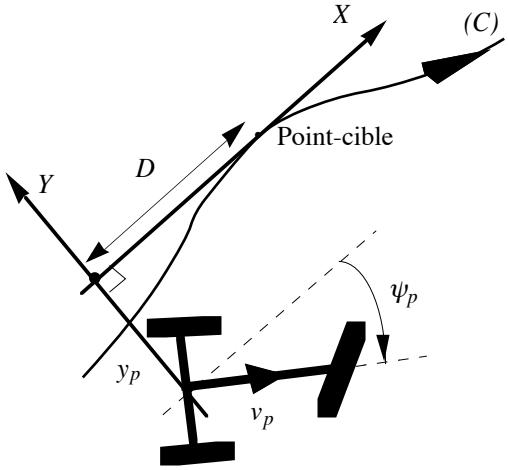


Figure 4.
Choix du point-cible et détermination de la tangente sur laquelle asservir le point de commande.

Justification du choix $D = F_D\, v_p$.

Le choix d'une distance au point-cible proportionnelle à la vitesse mérite une justification. Nous allons montrer qu'aux petites erreurs (erreur latérale y et erreur de cap ψ), l'application d'un retour d'état conçu pour rallier une trajectoire rectiligne pour le choix du point-cible comme point le plus proche de (C) avec addition d'un terme proportionnel à la courbure (approche " courbure "), est équivalente à l'application du même retour d'état pour le choix d'un point-cible évoluant à une distance D proportionnelle à la vitesse en avant du point de commande (approche " point-cible ").

Pour de petites erreurs, on peut raisonner avec le modèle linéarisé suivant (nous nous plaçons en temps continu pour simplifier la présentation) :

$$\begin{cases} \dot{\psi} = \frac{v}{L}\, tg\,(\alpha) \approx \frac{v}{L}\,\alpha \\ \dot{y} = v\ \ sin\,(\psi) \approx v\ \ \psi \end{cases}$$

On montre facilement que pour imposer une dynamique de ralliement de la trajectoire y=0 indépendamment de la vitesse, il faut utiliser un retour d'état de la forme :

$$\alpha = K_y\, y + K_\psi\, \psi \qquad avec \qquad \begin{cases} K_y = \frac{k_y}{v^2} & k_y < 0 \\ K_\psi = \frac{k_\psi}{v} & k_\psi < 0 \end{cases}$$

(Le réseau de neurones trouve donc nécessairement une solution de ce type aux petites erreurs.)

Notons α_{cou} et α_{pc} les commandes utilisées respectivement dans les approches courbure et point-cible. Ces commandes ont pour expressions :

$$\begin{cases} \alpha_{cou} = K_y\, y_{cou} + K_\psi\, \psi_{cou} + L\, \kappa \\ \alpha_{pc} = K_y\, y_{pc} + K_\psi\, \psi_{pc} \end{cases}$$

où y_{cou} et ψ_{cou} sont les variables d'état de l'approche courbure, et y_{pc} et ψ_{pc} les variables d'état de l'approche point-cible ; κ est la courbure de la trajectoire entre le point le plus proche et le point-cible, supposée constante. Ces variables sont représentées sur la figure 5.

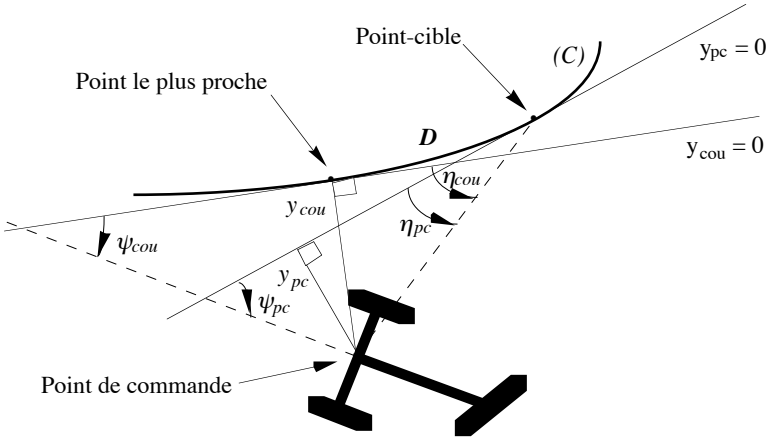


Figure 5.
Comparaison des approches fondées sur la courbure et sur le point-cible.

Aux petits angles, on a (en utilisant la relation angulaire entre les angles η_{cou} et η_{pc}) :

$$\begin{cases} y_{pc} = y_{cou} - \dfrac{D^2}{2}\,\kappa \\ \psi_{pc} = \psi_{cou} + D\,\kappa \end{cases}$$

Récrivons α_{pc} :

$$\alpha_{pc} = K_y\, y_{cou} + K_\psi\, y_{cou} + \left(K_\psi\, D - K_y\, \frac{D^2}{2}\right)\,\kappa$$

Comme K_y et K_ψ sont inversement proportionnels à la vitesse et à son carré respectivement, cette expression de α_{pc} est égale à celle de α_{cou} si l'on prend $D = F_D\, v$, avec F_D tel que :

$$k_\psi f - k_y \frac{f^2}{2} = L$$

Ceci est toujours possible, l'équation du second degré :

$$f^2 - 2\frac{k_\psi}{k_y} f + 2\frac{L}{k_y} = 0$$

ayant toujours deux racines réelles dont une positive, puisque $k_y<0$.

Nous avons donc prouvé que, aux petites erreurs, et pour un régulateur donné imposant une dynamique de ralliement indépendante de la vitesse, une stratégie utilisant un point-cible situé à une distance proportionnelle à la vitesse du point le plus proche de (C) est fondée. Aux grandes erreurs, il est intuitif que cette solution ne possède pas les défauts de l'approche fondée sur la courbure, mais il est moins simple de calculer, comme nous venons de le faire, la loi de variation de la distance au point-cible.

I.3. RÉSULTATS EXPÉRIMENTAUX.

Nous montrons ici quelques résultats expérimentaux obtenus avec le véhicule sur terrain sec non accidenté, puis en dévers. Des résultats obtenus en tout terrain sont présentés au §III de ce chapitre (avec l'asservissement de vitesse correspondant).

Système de commande.

Il s'agit d'un système de commande par simple bouclage. Le régulateur latéral obtenu en fin d'apprentissage, que nous appelons " RégLat " est utilisé avec les arguments :

$$\alpha(k) = \psi_{RN}^{RégLat}\left(\beta_p(k),\, \psi_p(k),\, y_p(k),\, v_p(k)\right)$$

où y_p et ψ_p sont déterminés en fonction du point-cible sélectionné (voir figure 4). Le système de commande de l'asservissement latéral est représenté sur la figure 6.

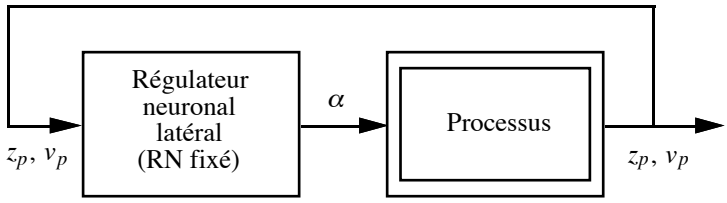


Figure 6.
Système d'asservissement latéral.

Asservissement latéral sur trajectoire.

La première expérience a pour but l'asservissement du véhicule sur une trajectoire de consigne d'environ 600 m de long (il s'agit du tour du centre de SAGEM Éragny). La courbure maximale de la trajectoire est de 0,1 m^{-1}. Cette trajectoire est représentée sur la figure 7.

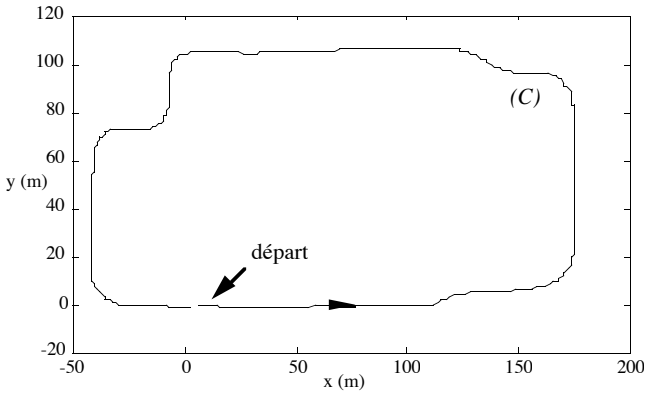


Figure 7.
Trajectoire de consigne.

Les résultats de l'asservissement sont rassemblés sur la figure 8. La vitesse, imposée par un conducteur, est assez importante compte tenu des virages serrés de la trajectoire (voir figure 8d). La figure 8c permet de visualiser la saturation en vitesse de l'angle du volant, qui se manifeste en particulier dans les virages situés en haut à gauche sur la trajectoire de consigne. L'erreur latérale indiquée est la distance du véhicule au point le plus proche de la trajectoire de consigne, et l'erreur de cap, l'angle de l'axe du véhicule par rapport à la tangente à la trajectoire en ce point. Ces erreurs, qui correspondent aux erreurs y_{cou} et ψ_{cou} de la figure 5, définissent la performance, et ne sont pas celles qui sont fournies au réseau pendant le fonctionnement du système de commande, y_{pc} et ψ_{pc}. Les performances sont bonnes, les deux erreurs étant inférieures à 40 cm et 50 mrd respectivement.

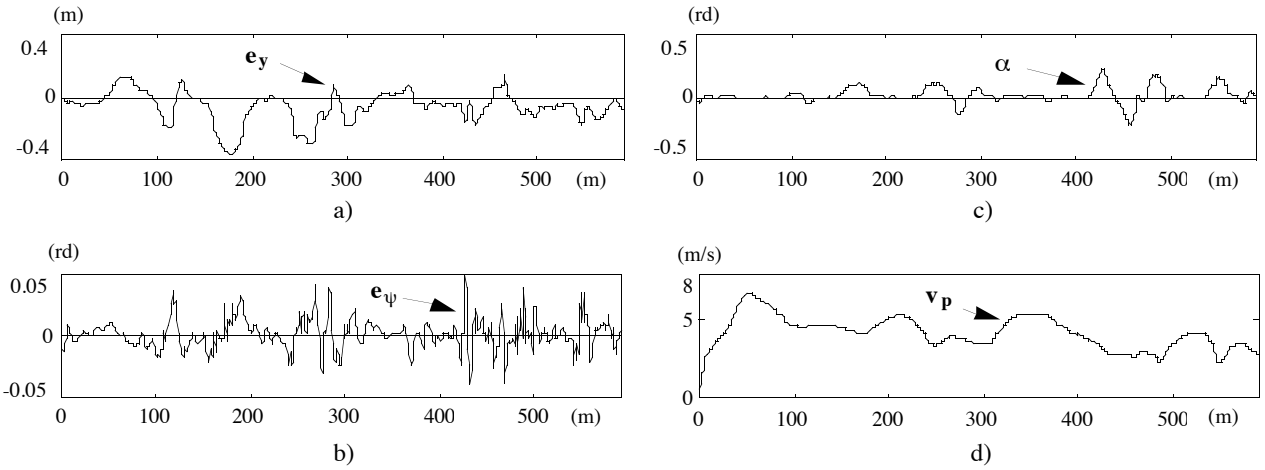


Figure 8.
Asservissement latéral sur trajectoire : a) erreur latérale, b) erreur de cap, c) commande du volant, d) vitesse, en fonction de l'abscisse curviligne le long de la trajectoire de consigne.

Ralliement de la trajectoire de consigne.

La deuxième expérience a été effectuée en vue de montrer la capacité du système de commande à rallier la trajectoire après que le véhicule en a été écarté de quelques mètres, c'est-à-dire hors du domaine des petites erreur de cap et erreur latérale. Nous présentons sur la figure 9 la trajectoire de consigne, et celle du point de commande du véhicule (milieu de l'essieu arrière).

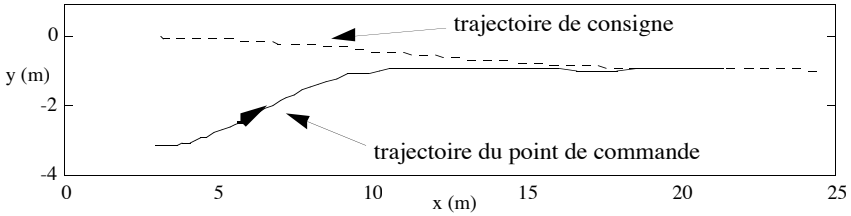


Figure 9.
Ralliement de la trajectoire de consigne.

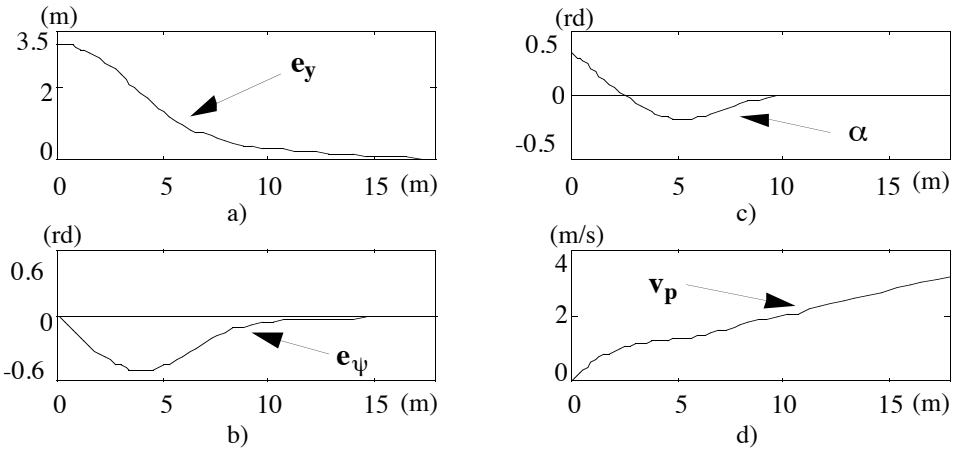


Figure 10.
Ralliement de la trajectoire de consigne : a) erreur latérale, b) erreur de cap, c) commande du volant, d) vitesse, en fonction de l'abscisse curviligne le long de la trajectoire de consigne.

L'évolution des variables d'intérêt le long de cette trajectoire est représentée sur la figure 10. Ici encore, la vitesse est imposée par un conducteur. À l'instant initial, le véhicule est maintenu à l'arrêt : le volant est alors complètement braqué au démarrage ($\alpha(0)=\beta_p(0)=0,5$ rd). Néanmoins, l'erreur latérale, initialement de 3,5 m, est annulée sans dépassement.

Asservissement latéral en dévers.

Le système de commande utilisé n'impose pas une erreur latérale statique nulle, puisque le correcteur ne contient pas d'intégrateur (explicite ou implicite). Il faut donc s'attendre à une erreur non nulle en cas de dévers important. Nous avons réalisé des essais sur des pistes en dévers calibrées sur les terrains d'essais de l'ETAS (Etablissement Technique d'AngerS). La trajectoire de consigne et celle du point de commande du véhicule sont représentées sur la figure 11.

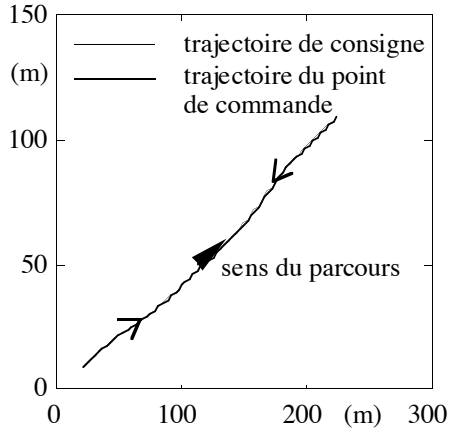


Figure 11.
Asservissement latéral en dévers (20%) à 15 km/h : trajectoires de consigne et du point de commande.

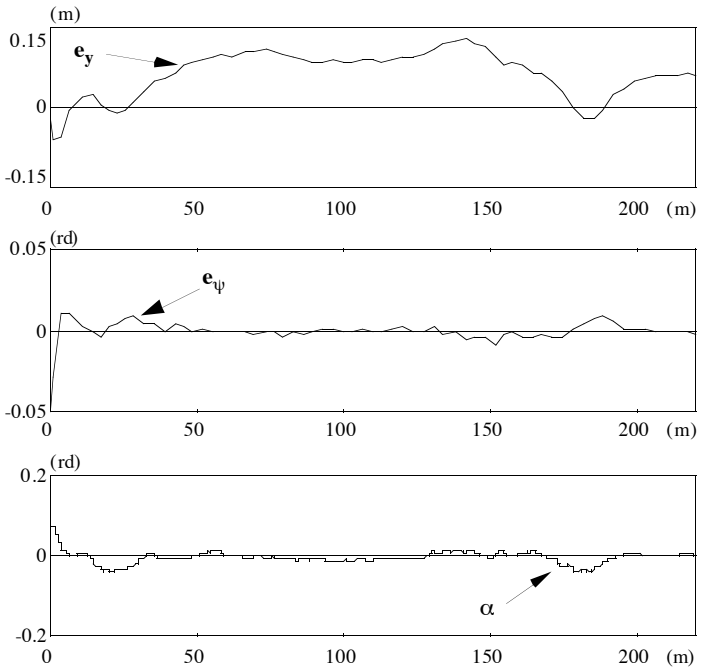


Figure 12.
Résultats d'asservissement latéral en dévers (20%) à la vitesse de 4,2 m/s (15 km/h).

Les résultats de la figure 12 correspondent à une piste en dévers de 20%, parcourue à une vitesse stabilisée de 15 km/h. L'erreur statique observée est de l'ordre de 10 cm, performance tout à fait satisfaisante. Les deux créneaux importants sur le volant (vers 25 et 175 m d'abscisse curviligne) correspondent aux endroits, indiqués par des chevrons sur la figure 11, où le véhicule aborde et quitte le dévers. À 6 km/h, toujours sur un dévers de 20%, l'erreur latérale n'est que de 4 cm environ ; à 6 km/h, sur un dévers de 30%, elle est de l'ordre de 7 cm [ETA94]. Il est préférable de se contenter de ces performances, très satisfaisantes, plutôt que d'introduire un terme intégral dans la loi de commande, qui risquerait de diminuer la stabilité du système.

II. ASSERVISSEMENT LONGITUDINAL.

Enjeux de l'asservissement longitudinal.

L'asservissement longitudinal de la vitesse s'effectue par l'action sur le papillon des gaz, sur la pression dans le circuit de freinage, et également sur le sélecteur des vitesses. Il n'existe pas de modèle universel de la dynamique longitudinale de robots mobiles à roues, car celle-ci dépend étroitement de leur mode de propulsion et de leurs actionneurs particuliers. Par exemple, on ne peut comparer la dynamique d'un véhicule électrique, dont un modèle standard est donné par exemple dans [SHL78], et celle d'un véhicule à moteur thermique tel que REMI. Plus encore que pour l'asservissement latéral, il est donc nécessaire de tenir compte de la dynamique particulière du véhicule. De plus, la dynamique de REMI est complexe et non linéaire, et l'expérience a montré qu'un simple PID était insuffisant pour une commande performante [MIN93b].

Position du problème.

Un profil de vitesse de consigne est associé à la trajectoire de consigne. Le problème de l'asservissement longitudinal est le problème de la poursuite de cette consigne de vitesse, problème que nous avons choisi de résoudre à l'aide d'un système de commande avec modèle interne (SCMI). Nous avons fait ce choix en raison de la moindre précision du modèle longitudinal dans certains domaines de fonctionnement (par comparaison avec le modèle latéral), et en fonction des résultats obtenus avec des systèmes de commande qui, comme on pouvait s'y attendre, se sont révélés moins performants : un système de commande par simple bouclage (SCSB) utilisant un correcteur-S et un modèle de ralliement, et un " PID " non linéaire (ou NID, voir chapitre 5 §II.3.3).

II.1. APPRENTISSAGE D'UN CORRECTEUR-S.

Le problème est donc celui de la poursuite et de la régulation de la vitesse du véhicule. On agit toujours séparément sur les freins et sur l'accélérateur, qui sont des commandes de même nature : le problème peut donc être considéré comme un problème à une seule entrée. Comme le modèle du processus est de type entrée-sortie de retard unité, nous avons vu que le SCMI le plus performant et

souple d'utilisation, dans ce cas, est le système utilisant un correcteur-S et un modèle de ralliement (chapitre 5 §II.4.2.1). Nous avons donc procédé à l'apprentissage d'un correcteur-S.

Modèle de référence d'apprentissage.

Le modèle de référence d'apprentissage, de sortie v_a, est un simple retard :

$$v_a(k+1) = v_r(k)$$

où v_r est la séquence de consigne.

Modèle de simulation.

La sortie du modèle est la vitesse v. Les entrées de commande du modèle sont l'angle papillon commandé θ, et la pression de freinage commandée π. Les entrées perturbatrices simulées sont d'une part le rapport des vitesses r, converti en la grandeur quantifiée ρ, et d'autre part la pente p. K est le facteur d'inertie qui est fonction du rapport engagé. Rappelons l'expression de ce modèle de simulation établi à partir du prédicteur identifié :

$$v(k) = \varphi_{RN}^{ModLon}\left(v(k\text{-}1),\, \theta(k\text{-}1),\, \pi(k\text{-}1),\, \rho(k\text{-}1)\right) - \frac{g\,\Delta T}{1 + K\left(\rho(k\text{-}1)\right)}\, sin\left(p(k\text{-}1)\right)$$

Correcteur-S.

Le réseau de neurones correcteur est non bouclé de la forme :

$$u(k) = \psi_{RN}\left(v_r(k), v(k), \rho(k), p(k);\, C\right)$$

où la commande u est appliquée à l'accélérateur si elle est positive, aux freins sinon :

$$\begin{cases} si\ u(k) \geq 0,\ \theta(k) = u(k)\ et\ \pi(k) = 0. \\ si\ u(k) < 0,\ \theta(k) = 0\ et\ \pi(k) = u(k). \end{cases}$$

Les valeurs autorisées de la commande u étant confinées dans l'intervalle [-1; 1], on choisit comme fonction d'activation du neurone de sortie une sigmoïde à valeurs dans [-1; 1].

Séquences d'apprentissage.

Les séquences d'apprentissage sont constituées de trajectoires à rapport des vitesses ρ et pente p constants. Ces trajectoires $\{v_r\}$ sont calculées par un modèle de référence à partir de créneaux de consigne $\{v_c\}$: ce modèle est le même quels que soient ρ et p ; il correspond à un filtre à temps continu du second ordre de pulsation ω_n=1 et d'amortissement ζ=1. Ce modèle est plus rapide que tous les modèles de ralliement qui seront utilisés au sein du SCMI. Il est noté :

$$E(q)\, v_r(k+1) = H(q)\, v_c(k)$$

où $\{v_c\}$ est une séquence de consigne constituée de 10 créneaux de durée 35 secondes. L'amplitude des créneaux de consigne dépend des valeurs de ρ et de p.

* Ainsi, pour une pente nulle, l'amplitude maximale v_{max} des créneaux est la suivante :

- en 1ère (ρ=0,00) : v_{max} = 10 m/s ;
- en 2nde (ρ=0,40) : v_{max} = 15 m/s ;
- en 3ème (ρ=0,64) : v_{max} = 20 m/s ;
- en 4ème (ρ=0,75) : v_{max} = 25 m/s.

La vitesse maximale choisie pour les trois premières vitesses est supérieure à celle que le modèle peut effectivement atteindre. En effet l'utilisation future du réseau dans un SCMI exige un apprentissage dans un domaine plus large que le domaine de fonctionnement réel du processus, même si le correcteur sature (voir chapitre 5 §II.4.1.5).

* Nous avons choisi de ne compenser la pente à l'aide du correcteur qu'en première : c'est en effet le seul rapport de vitesses qui sera utilisé pour gravir ou descendre des pentes importantes. Si, en fonctionnement, la première n'est pas imposée au moyen du levier des vitesses, ce qui ne sera le cas que pour des pentes faibles, le SCMI devra compenser. En première, on prend pour la pente p, pendant l'apprentissage, des valeurs comprises entre ±40%.

Fonction de coût.

La fonction de coût est la somme des carrés des erreurs de vitesse sur toutes les trajectoires :

$$J = \frac{1}{N} \sum_{k=1}^{N} e(k)^2 = \frac{1}{N} \sum_{k=1}^{N} \left(v_a(k) - v(k)\right)^2$$

où N est le nombre total d'instants de l'ensemble d'apprentissage.

Algorithme d'apprentissage.

L'algorithme d'apprentissage est semi-dirigé. Le système d'apprentissage est représenté sur la figure 13.

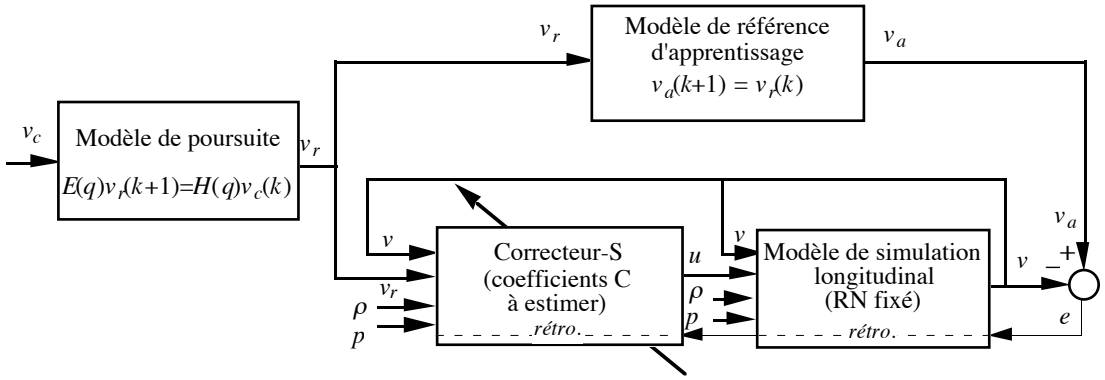


Figure 13.
Système d'apprentissage du correcteur-S.

Résultats d'apprentissage.

Nous allons maintenant présenter les résultats de cet apprentissage sur des séquences de test. Pour alléger la présentation, nous la restreignons à des simulations en pente nulle, la seule perturbation mesurée à compenser étant donc le rapport des vitesses.

Sur les figures 14a et 14b sont représentées les réponses du système d'apprentissage à des séquences de test dans le domaine de fonctionnement *souhaité pour le processus* : la consigne y est telle que le modèle peut l'atteindre en régime statique (la commande peut toutefois se saturer si l'accélération demandée est trop importante). Les limites de saturation de la commande à ±1 sont indiquées en traits fins pointillés sur les figures.

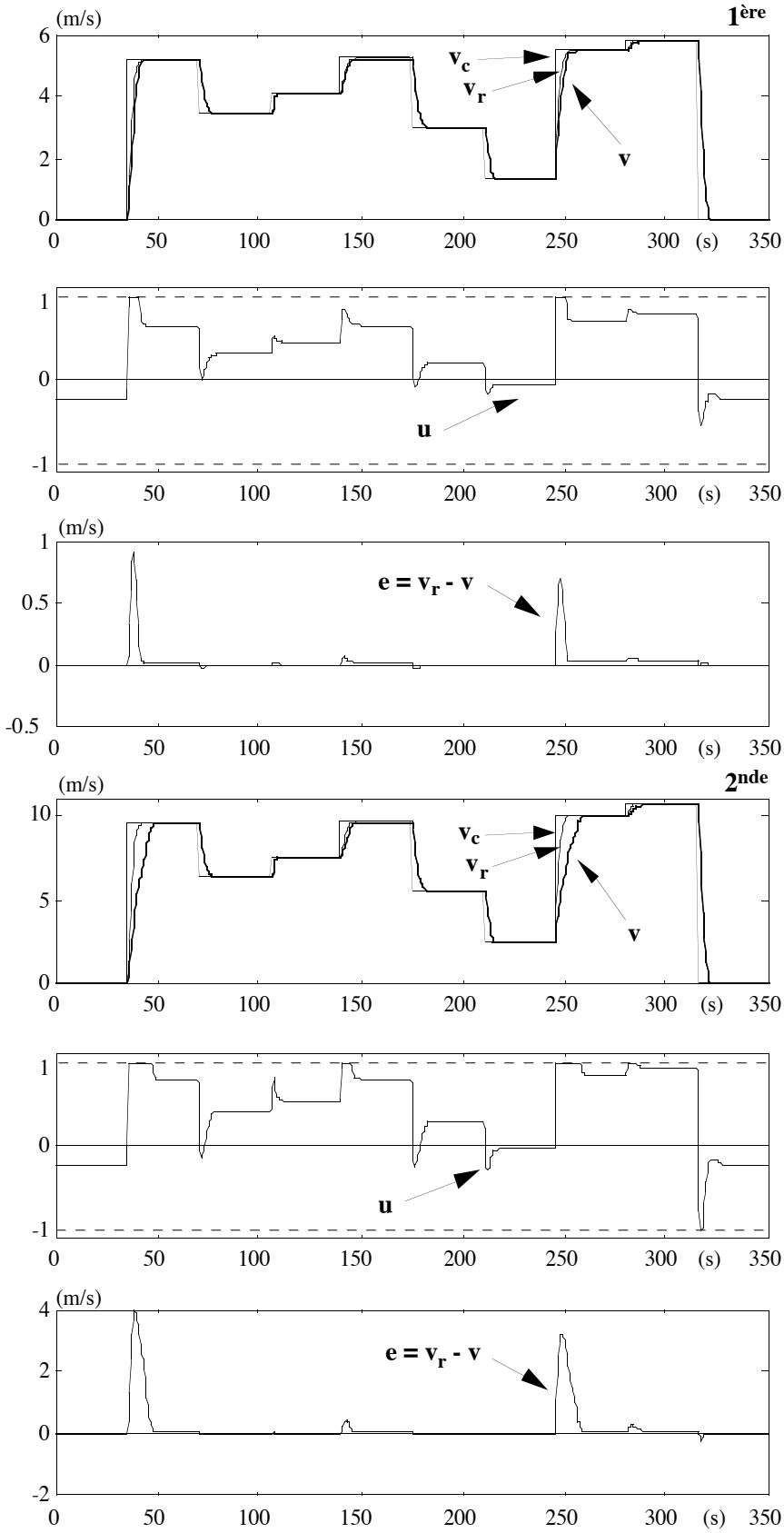


Figure 14a.
Test du correcteur-S en simple bouclage avec le modèle, en 1ère et en 2nde, dans le domaine de fonctionnement souhaité pour le processus, avec un modèle de poursuite rapide (ω_n=1).

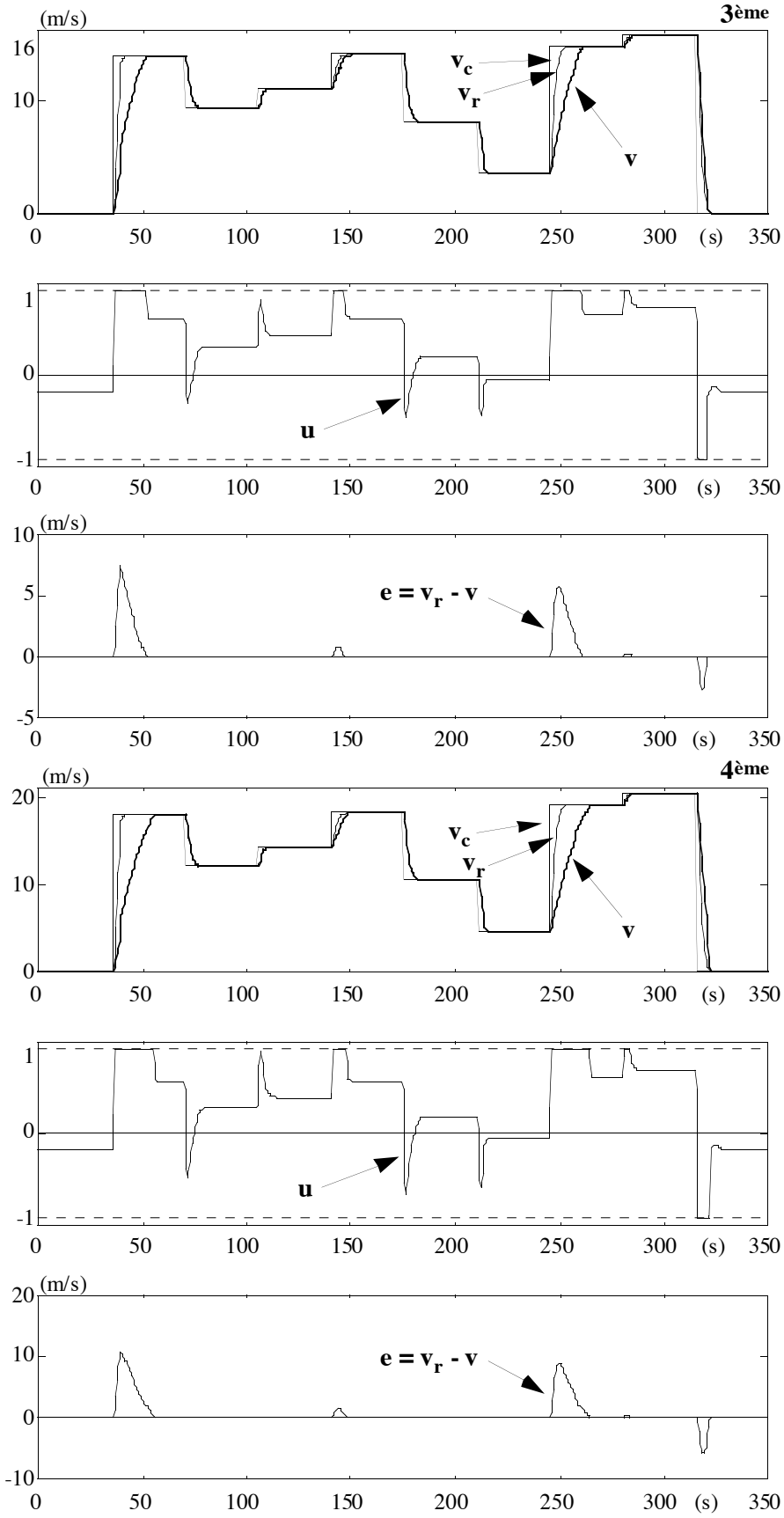


Figure 14b.
Test du correcteur-S en simple bouclage avec le modèle, en 3ème et en 4ème, dans le domaine de fonctionnement souhaité pour le processus, avec un modèle de poursuite rapide (ω_n=1).

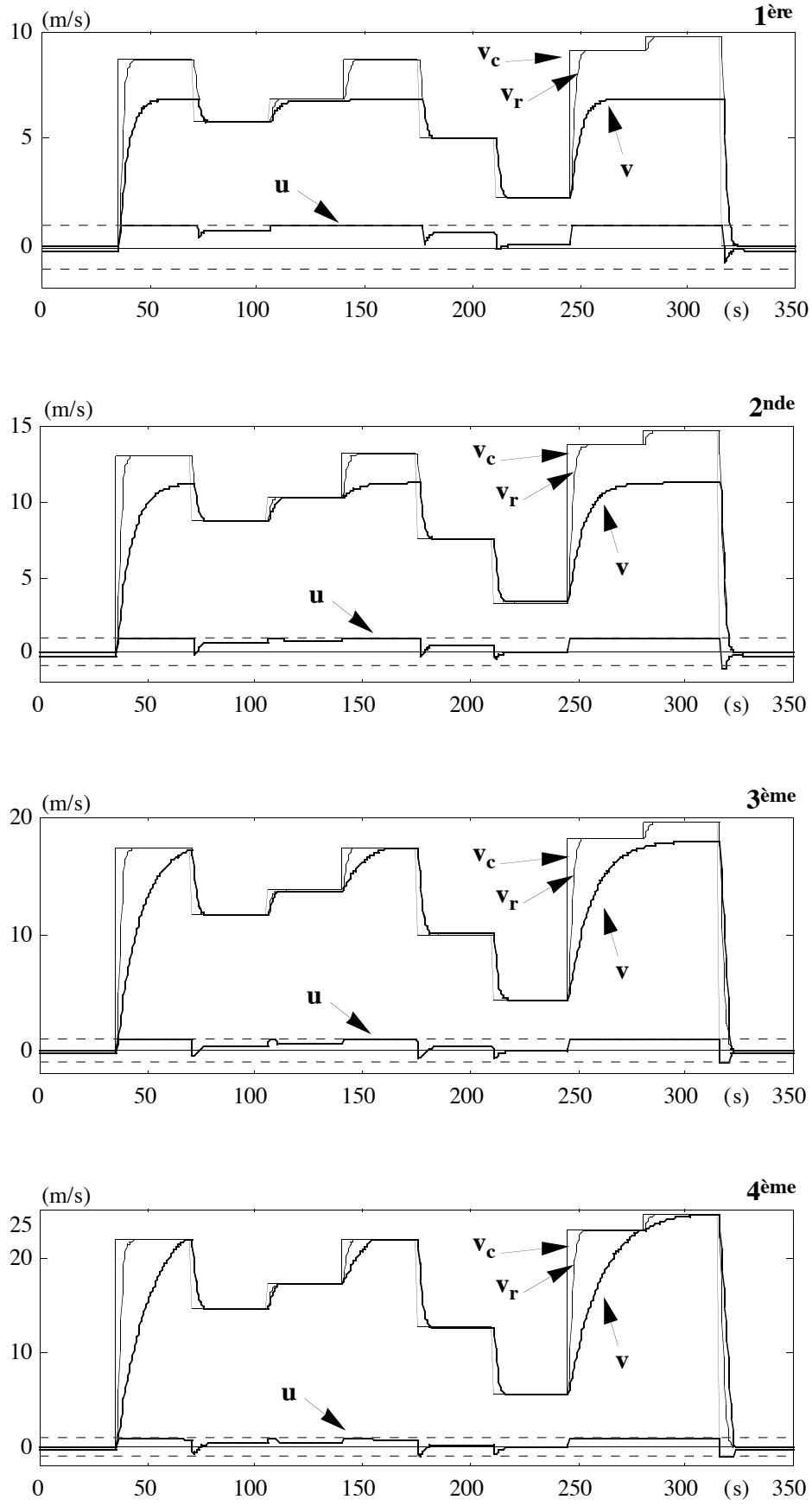


Figure 15.
Test du correcteur-S en simple bouclage avec le modèle, dans le domaine d'apprentissage entier du modèle, avec un modèle de poursuite rapide (ω_n=1) : le correcteur est presque toujours **saturé** (u=±1).

Sur la figure 15, nous avons rassemblé des résultats illustrant le comportement du modèle commandé dans tout le domaine d'apprentissage, qui est le domaine de fonctionnement *souhaité pour le modèle* : les consignes maximales y sont telles que, étant donnés la saturation en ±1 du correcteur et le gain statique du modèle, celui-ci ne peut les atteindre. Il est capital que le comportement du réseau correcteur soit correct pour de telles consignes, qui ne seraient pas demandées dans un SCSB, mais qui peuvent l'être dans un SCMI (car la consigne y est décalée de la valeur de l'écart de vitesse entre le processus et le modèle). Les courbes de la figure 15 montrent que le comportement du correcteur est effectivement correct dans tout le domaine d'apprentissage.

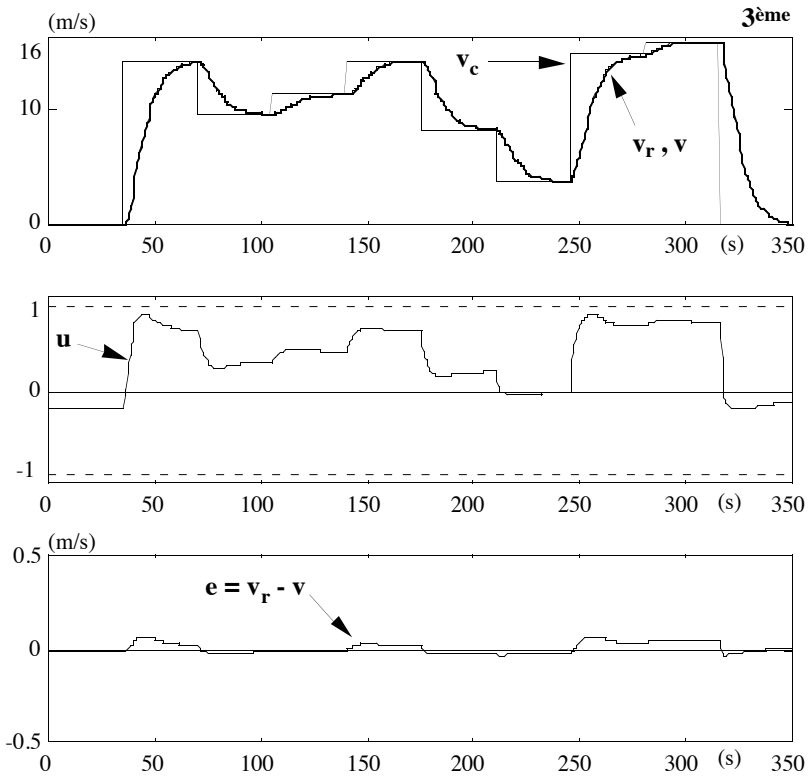


Figure 16.
Test du correcteur-S en simple bouclage avec le modèle, dans le domaine de fonctionnement souhaité pour le processus, en 3ème, avec un modèle de poursuite plus lent (ω_n=0,2).

La figure 16 enfin illustre le comportement du système d'apprentissage pour des séquences de consignes obtenues avec un modèle de référence plus lent (correspondant à la discrétisation d'un filtre continu de pulsation ω_n=0,2 et ξ=1) ; elle est donc représentative des performances que l'on peut obtenir pour des entrées de consigne moins rapides que des échelons : dans ces conditions, le correcteur-S est presque parfait (l'erreur de commande est inférieure à 5 10^{-2}, lorsque la commande ne se sature pas). Nous montrons les résultats obtenus en 3ème, dans le domaine de fonctionnement souhaité pour le processus. Ces résultats sont à comparer à ceux de la figure 14, en 3ème également.

Ces résultats sont obtenus à l'aide d'un correcteur possédant 5 neurones cachés complètement connectés. Le correcteur est donc plus complexe que le modèle (3 neurones cachés). Ceci est dû au

fait que la commande est appliquée, suivant son signe, à deux entrées distinctes du modèle, de caractéristiques très différentes (celle des freins et celle de l'accélérateur).

II.2. RÉSULTATS EXPÉRIMENTAUX (COMMANDE AVEC MODÈLE INTERNE).

Système de commande.

Comme nous l'avons annoncé, le correcteur-S est intégré dans un SCMI, avec un modèle de ralliement. Ce modèle de ralliement définit une dynamique identique à celle du modèle de poursuite utilisé pendant l'apprentissage pour calculer la séquence de consigne (ou éventuellement plus lente, en fonction de résultats d'essais avec le véhicule). L'expression de sa sortie v_r* est donnée par :

$$v_r^*(k+1) = \left(1 - E(q)\right) v(k+1) + H(q)\, v_c^*(k)$$

Le correcteur-S obtenu, que nous appellons ψ_{RN}^{CorLon}, a pour arguments :

$$u(k) = \psi_{RN}^{CorLon}\left(v_r^*(k+1),\, v(k),\, \rho_p(k),\, p_p(k)\right)$$

où ρ_p et p_p sont les estimations du rapport des vitesses du véhicule et de la pente du terrain.
Le modèle interne est bouclé, et défini par la même fonction que le modèle de simulation entrée-sortie de l'apprentissage, mais avec les arguments :

$$v(k) = \varphi_{RN}^{ModLon}\left(v(k\text{-}1),\, \theta(k\text{-}1),\, \pi(k\text{-}1),\, \rho_p(k\text{-}1)\right) - \frac{g\,\Delta T}{1 + K\left(\rho_p(k\text{-}1)\right)} \sin\left(p_p(k\text{-}1)\right)$$

Le modèle de ralliement correspond au filtre du second ordre avec ω_n=0,3 et ξ=1 (ce choix résulte d'expérimentations avec le véhicule). Le système de commande est représenté sur la figure 17.

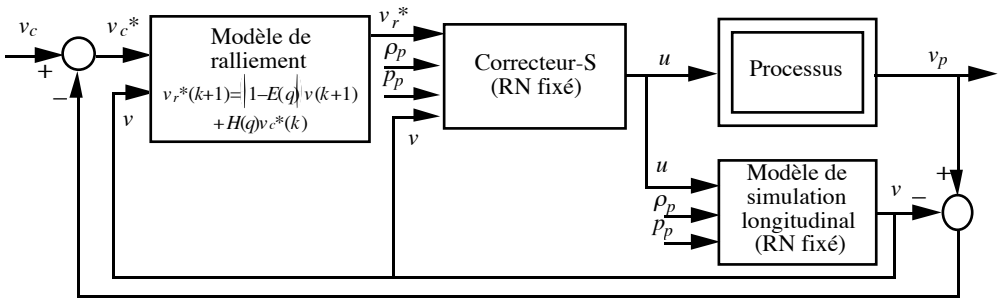


Figure 17.
SCMI avec correcteur-S et modèle de ralliement pour l'asservissement longitudinal.

Modalités d'expérimentation.

La figure 18 montre ainsi un exemple des résultats expérimentaux obtenus sur terrain plat (route). Avant de commenter ces résultats, nous donnons quelques détails sur les modalités de l'expérimentation.

- Comme pour l'asservissement latéral, l'asservissement longitudinal est ici mis en œuvre sans asservissement latéral, un conducteur assurant le pilotage de la direction du véhicule ;
- La position du levier des vitesses est telle que le véhicule peut passer tous les rapports, ce que l'on peut voir sur la figure 18 (la valeur de $\rho_p = 1 - 1/r_p$ atteint sa valeur maximale 0,75). La valeur du

rapport donnée en entrée du réseau de neurones ρ_p est quantifiée selon les quatre valeurs possibles du rapport (à partir de l'estimation faite en fonction de la vitesse et du régime moteur) ;

- Lorsque la vitesse de consigne est nulle, et que le véhicule est presque à l'arrêt, la commande maximale de freinage u=-1 est appliquée, par sécurité (instants 280 et 380 sur la figure 18).

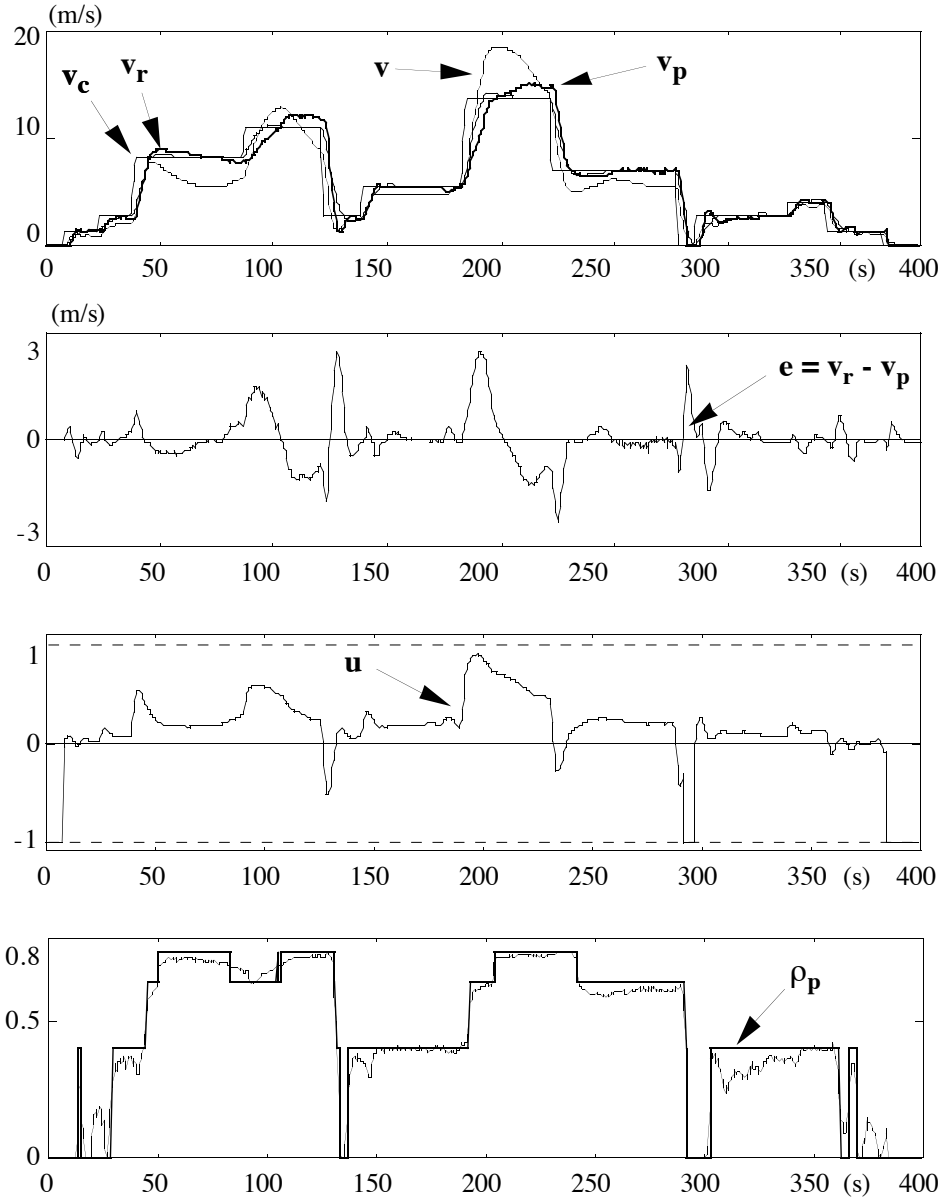


Figure 18.
Résultats expérimentaux obtenus avec le SCMI.

Les résultats de la figure 18 font apparaître un bon comportement du système de commande dans le domaine de vitesses 0-10 m/s. L'erreur statique est nulle, et la réponse est à la fois rapide et amortie. Au delà, la dynamique est assez considérablement ralentie. Ceci s'explique par une moins bonne modélisation à haute vitesse. Nous avons en effet observé, au chapitre 7, que la dynamique du modèle y est plus rapide que celle du processus. Cette observation se trouve confirmée par les résultats de la figure 18 : la vitesse du modèle s'écarte beaucoup plus de celle du processus à haute vitesse, et sa réponse est plus rapide que celle de ce dernier.

Améliorations à apporter.

Le meilleur moyen d'atténuer ces défauts est bien sûr de réaliser une identification plus précise du véhicule à haute vitesse. Cependant, une amélioration du système de commande actuel pourrait consister à définir un modèle de ralliement pour chaque rapport de vitesse. Le modèle de ralliement actuel a en effet été réglé de manière à conférer au système de commande un comportement satisfaisant en moyenne. Une telle modification permettrait de compenser les écarts de modélisation dans chaque zone du domaine de fonctionnement.

III. ASSERVISSEMENT COMPLET EN TOUT-TERRAIN.

Nous montrons à présent des résultats expérimentaux obtenus en asservissement complet, latéral et longitudinal, sans conducteur à bord (mode pseudo-autonome). La trajectoire de consigne est représentée sur la figure 19. Le terrain est inégal (boue et herbe), et présente deux pentes de 30% (indiquées par des chevrons sur la figure 19). La vitesse de consigne a une valeur constante de 1,67 m/s (6 km/h).

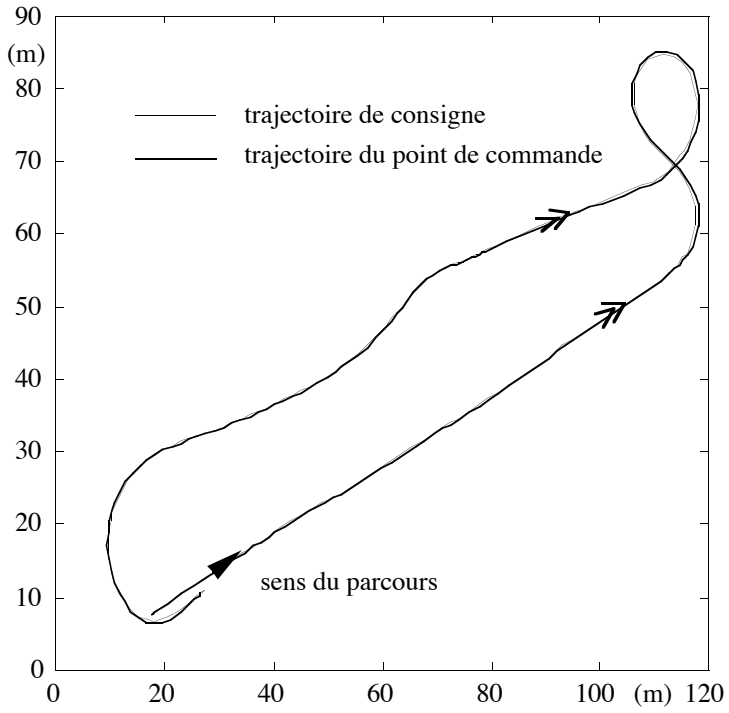


Figure 19.
Résultats expérimentaux d'asservissement complet : trajectoires de consigne et du point de commande.

Les figures 20 et 21 suivantes présentent les résultats des deux asservissements. Toutes les grandeurs sont données en fonction de l'abscisse curviligne le long de la trajectoire de consigne.

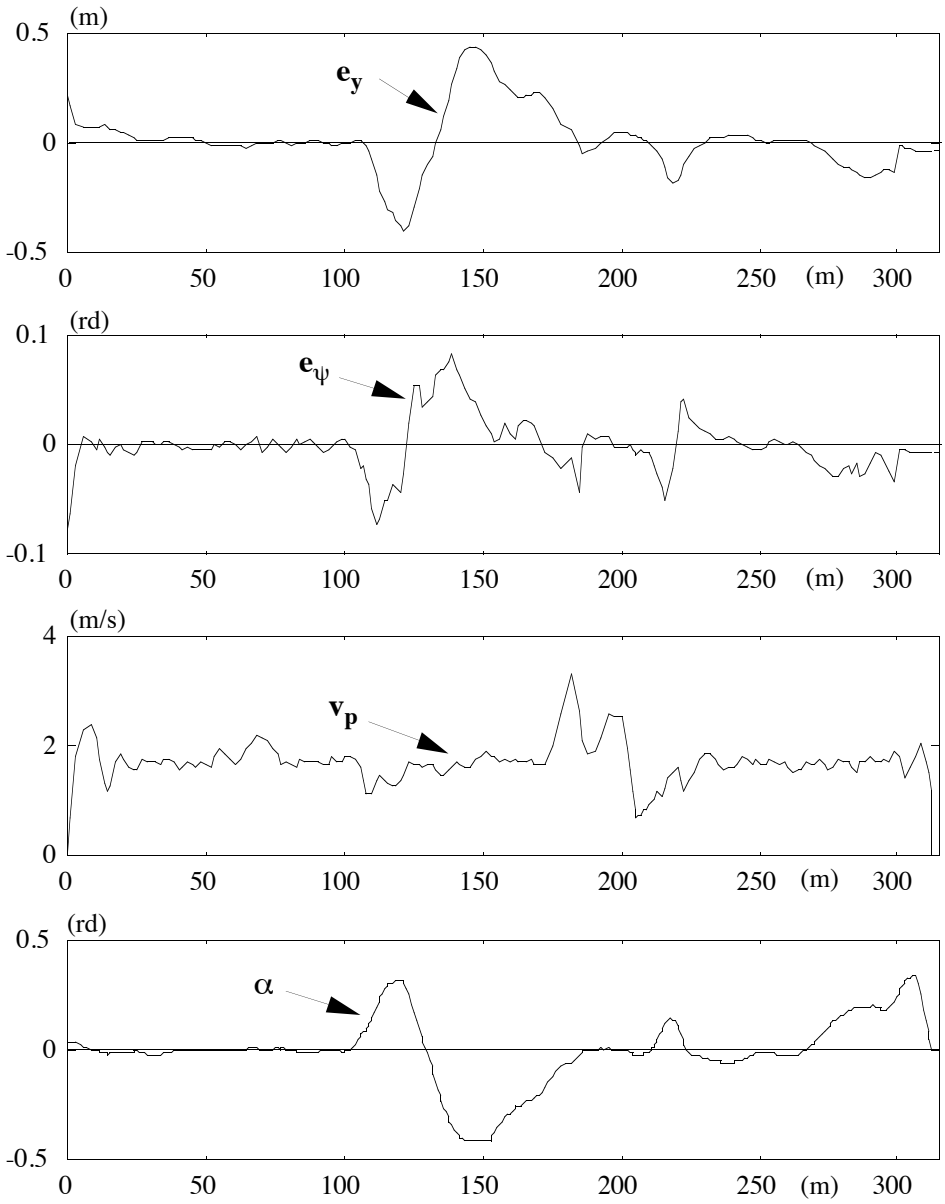


Figure 20.
Résultats expérimentaux d'asservissement complet : asservissement latéral.

Les résultats de la figure 20 font apparaître le maintien des performances du système de commande latéral en tout-terrain : l'erreur latérale ne dépasse pas non plus 40 cm, et ce pour un angle de braquage presque maximal des roues (0.4 rd).

La figure 21 présente les résultats de l'asservissement longitudinal. Les discontinuités de la pente, indiquée sur la courbe du bas, sont bien compensées en montée (instants 50 et 100 s). Pendant la montée (instants 50 à 100 s), l'erreur statique est annulée, pour une valeur de 30% de la pente. Les résultats sont moins bons à la descente (instants 175 à 210 s). Ce défaut pourrait être atténué par une meilleure modélisation du véhicule aux basses vitesses d'une part, et d'autre part par une meilleure prise en compte de la pente par le modèle, qui surestime un peu l'accélération qui lui est due (comme on peut le voir sur l'évolution de la vitesse v du modèle interne).

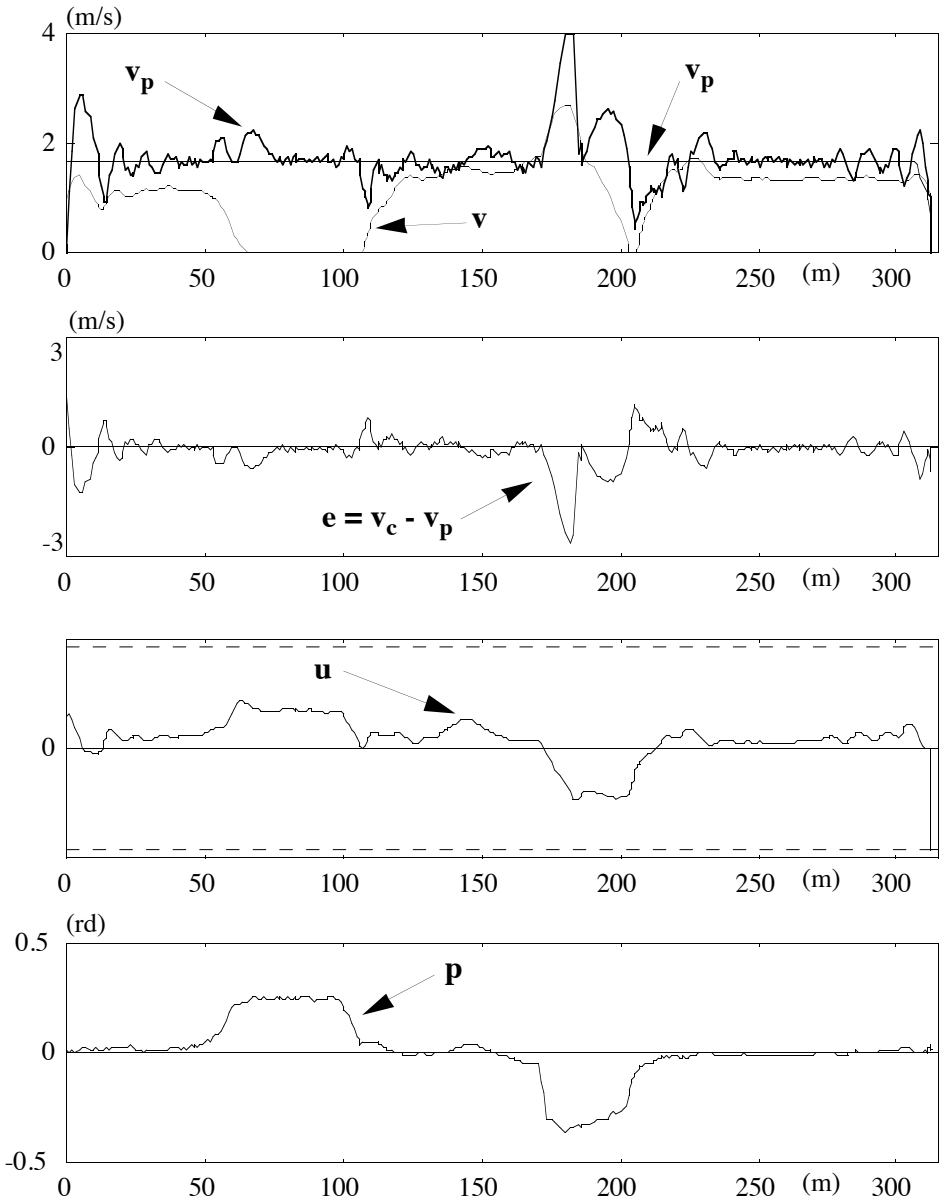


Figure 21.
Résultats expérimentaux d'asservissement complet : asservissement longitudinal.

Notons que le deuxième pic d'erreur de vitesse après le début de la descente (vers 190 s) est dû aux inégalités du terrain à cet endroit (ornières profondes), que révèle d'ailleurs la vitesse du modèle interne, laquelle diminue brutalement au lieu d'augmenter.

Résolution pratique du problème terminal.

Nous avons laissé en suspens le problème de l'arrêt en fin de trajectoire. En effet, aucun système de commande utilisant un correcteur latéral par retour d'état statique invariant ne peut garantir une erreur latérale finale nulle. Dans la pratique, la trajectoire de consigne est prolongée au delà de son point final de manière à pouvoir continuer à définir le point-cible en avant du véhicule selon la règle du §I.2. Lorsque la vitesse de consigne devient nulle (le profil de consigne de la mission en asservissement complet - en mode pseudo-autonome - est conçu en fonction de la décélération maximale du véhicule, d'environ 7 m/s^2), le véhicule s'arrête en pratique très vite, et les erreurs

latérales finales ne dépassent pas la dizaine de centimètres, comme on l'observe sur les figures 8 et 20. Seules des valeurs beaucoup plus importantes de l'erreur justifieraient l'utilisation d'un correcteur d'asservissement sur posture par retour d'état discontinu ou instationnaire à l'approche de la destination finale (correcteurs proposés par [CAN91] [SAM92] [POM92]) comme nous le suggérions dans l'introduction ; de plus ces correcteurs demanderaient à être réglés pour le véhicule.

CONCLUSION.

Le régulateur latéral est un bon exemple de la mise en œuvre de la recherche *ab initio* d'une loi de commande optimale à l'aide d'un réseau de neurones, méthode décrite dans la première partie de ce travail. Les résultats obtenus, aussi bien sur route à fortes variations de courbure (courbure maximale de l'ordre de 0,1 m^{-1}), que sur des terrains accidentés, en dévers (jusqu'à 30%), en pente (jusqu`à 40%), et en tout-terrain avec une pente de 30%, montrent les capacités du système de commande à maintenir d'excellentes performances (erreur latérale inférieure à 40 cm) dans le domaine de fonctionnement non linéaire du véhicule (braquage et vitesse angulaire du volant importants). Les conditions expérimentales extrêmes dans lesquelles ces résultats ont été obtenus rendent difficile la comparaison avec d'autres véhicules automatisés. Ces derniers sont en général testés sur route plane et à faible courbure (0,01 à 0,05 m^{-1}), mais généralement à plus haute vitesse (50-110 km/h), comme par exemple les véhicules décrits dans [JUR93] et [PEN92]. Notre cahier des charges comporte des vitesses de consigne plus faibles (au maximum 72 km/h), mais a le mérite de mettre l'accent sur le comportement non linéaire du véhicule, et de plus en tout-terrain.

De même, l'asservissement longitudinal avec modèle interne et correcteur-S montre la faisabilité de cette approche à l'aide de réseaux de neurones. Malgré les défauts du modèle, le système garantit une erreur statique nulle, même en présence de perturbations importantes (pente de 30%). L'affinement du modèle, et le choix de plusieurs modèles de ralliement pour les différents régimes de vitesse, choix réalisable sans effectuer de nouvel apprentissage, devrait permettre d'améliorer encore ces performances. De plus, l'expérimentation à la SAGEM d'approches classiques apparentées à la commande avec modèle interne et correcteur-S [MIN93b], tend à montrer que ces systèmes de commande sont particulièrement bien adaptés au problème de la conduite en terrain perturbé. Ces considérations nous encouragent à persévérer dans l'investigation et l'amélioration de ces méthodes par l'utilisation de réseaux de neurones.

CONCLUSION GÉNÉRALE

Tout au long du présent travail, nous avons tiré parti de deux caractéristiques fondamentales des réseaux de neurones :

- *la propriété d'approximation universelle parcimonieuse*, dont l'apport est manifeste pour la modélisation et la commande de processus non linéaires ;
- *l'existence d'algorithmes d'apprentissage également universels*, au sens où leur mise en œuvre ne dépend pas de l'application considérée ni de la complexité du réseau soumis à un apprentissage.

Dans un premier temps, nous avons complété le cadre de l'apprentissage des réseaux de neurones élaboré par O. Nerrand pour des réseaux de type entrée-sortie, en l'étendant aux réseaux les plus généraux que sont les réseaux d'état ; nous avons ainsi disposé d'outils algorithmiques complets et puissants pour la modélisation et la commande de processus dynamiques.

C'est en nous appuyant sur ces deux caractéristiques fondamentales des réseaux de neurones que nous avons :

- déterminé les arguments de la fonction théorique que doit réaliser le réseau de neurones selon la tâche considérée (le prédicteur théorique associé à un modèle-hypothèse donné pour une tâche de modélisation, le correcteur théorique associé au modèle d'un processus pour une tâche de commande) : la propriété d'approximation universelle affirme alors que, si cette fonction existe, elle est réalisable par un réseau de neurones ;
- construit un système d'apprentissage du réseau de neurones (éléments constitutifs, algorithme) conduisant à une réalisation de la fonction théorique, si elle existe : l'universalité des algorithmes d'apprentissage garantit que cette réalisation est possible en pratique, quelle que soit la complexité de la fonction.

Dans le domaine de la commande de processus, cette démarche nous a conduit à définir deux familles de systèmes de commande fondés soit sur le correcteur-S, qui impose à un système une sortie de référence, soit sur le correcteur-D, qui lui impose une dynamique de référence. Ces deux familles recouvrent une grande partie des systèmes de commande utilisés en Automatique, et notamment en commande " neuronale ". Nous avons étudié les avantages et les inconvénients respectifs de ces systèmes, en particulier du point de vue de leur robustesse vis-à-vis de défauts du modèle d'une part, et de défauts du correcteur dus à son apprentissage d'autre part. C'est ainsi que nous avons établi les modalités de la mise en œuvre de systèmes de commande neuronaux avec modèle interne.

Par leur conception même, ces systèmes de commande neuronaux avec modèle interne présentent une bonne tolérance aux défauts de modélisation et aux perturbations ; comme il s'agit de systèmes non adaptatifs, le problème de la stabilité des systèmes d'apprentissage ne se pose pas. Néanmoins, ces systèmes de commande ne sont réalisables, avec des réseaux de neurones, que pour des processus dont le modèle est à inverse stable, alors qu'avec des méthodes classiques, il est possible de modifier la synthèse des correcteurs pour éviter cette restriction. Un développement souhaitable consisterait à élaborer les modifications à apporter au système d'apprentissage lorsque le modèle du processus est à inverse instable (comme cela a été fait pour les méthodes de commande avec modèle interne classiques). La commande prédictive neuronale constitue un autre axe de recherche pour la commande de processus à inverse instable.

Tous les exemples illustratifs choisis présentent une dynamique nettement non linéaire excluant une approche linéaire classique, et mettent en lumière l'apport des réseaux de neurones pour une approche non linéaire. L'application industrielle, le pilotage d'un véhicule autonome en tout-terrain, souligne également cet apport. Pour le pilotage latéral, auparavant réalisé à la SAGEM avec des techniques linéaires, la prise en considération des non-linéarités par le correcteur neuronal a permis d'améliorer les performances, sans qu'il soit nécessaire de procéder à aucun réglage après l'apprentissage et la mise en œuvre sur le véhicule. Pour le pilotage longitudinal, aucune technique linéaire ne s'est avérée satisfaisante, et le système de commande neuronal avec modèle interne a obtenu des performances comparables à celle du système non linéaire mis au point à la SAGEM. Ces performances sont très encourageantes dans la mesure où le système de commande n'a fait, ici non plus, l'objet d'aucun réglage supplémentaire. Nous avons ainsi pu vérifier les propriétés de la régulation et de la commande avec modèle interne neuronales, et démontrer la pertinence de l'approche neuronale dans son ensemble sur un problème réel complexe.

L'un des aspects originaux de notre démarche est l'utilisation systématique, pour la conception de modèles et de correcteurs " neuronaux ", des connaissances mathématiques du domaine d'application ; nous avons ainsi tiré parti de l'introduction d'éléments ad-hoc dans les réseaux, éléments déterminés par une analyse physique des phénomènes mis en jeu (modélisation), ou du type de performance souhaité (commande). D'un point de vue conceptuel, nous avons décrit et illustré une approche qui est fondamentalement très voisine de celle de l'automaticien classique. Loin de participer de démarches inconciliables, les synthèses classiques et neuronales de modèles ou de systèmes de commande de processus utilisent des concepts et des méthodes communs, et bénéficient mutuellement de leurs contributions originales respectives.

RÉFÉRENCES BIBLIOGRAPHIQUES

[ABU88] ABU EL ATA S. & COIC A. (1988) " Commande prédictive par inversion : application aux systèmes non linéaires ", Contrat DRET n°87/1226 du 6 avril 1988, ADERSA.

[AST84] ASTRÖM K. J. & WITTENMARK B. (1984) Computer controlled Systems, Prentice Hall, Englewood Cliffs (New Jersey).

[AYE81] AYELES D. (1981) " Generic observability of differentiable systems ", SIAM Journal of Control and Optimization, Vol.5 No.19, pp. 595-603.

[BAR93] BARRON A. (1991) " Universal approximation bounds for superposition of a sigmoidal function ", IEEE Trans. Information theory IT-39, pp. 930-945.

[BEN94] BENVENISTE A., JUDITSKY A., DELYON B., ZHANG Q. & GLORENNEC P.-Y. (1994) " Wavelets in identification ", Preprints of the 10th IFAC Symposium on Identification, Copenhague, 4-6 juillet 1994.

[BIL92] BILLINGS S. & CHEN S. (1992) " Neural networks and system identification ", in [HUS92], pp. 181-205.

[BLO] MM. BLONDELEAU, SEZNEC & PIERRE. Architecture et dynamique automobile, Service Formation de la DRD Renault.

[BOR90] BORNE P., DAUPHIN-TANGUY G., RICHARD J. P., F. ROTELLA & ZAMBETTAKIS I. (1990) Commande et optimisation des processus, Éditions Technip, Paris.

[BRO83] BROCKETT R. W. (1983) " Asymptotic stability and feedback stabilization ", in Differential geometric control theory, Proceedings of the conference held at Michigan Technological University, June 28-July 2, 1982, Progress in Mathematics Vol.27, Birkhäuser, pp. 181-191.

[CAN91] CANUDAS DE WIT C. & SORDALEN O. J. (1991) " Exponential stabilization of mobile robots with nonholonomic constraints ", IEEE Conference on Decision and Control, pp. 692-697.

[CHA90] CHANTRE P., CANAS D. & DARDENNES J. (1990) " Control by inversion of non-linear non-minimum-phase systems ", Proceedings of the IASTED'90, Autriche, pp. 1-6.

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

[CHE90a] CHEN S., BILLINGS S. A. & GRANT P. M. (1990) " Non linear system identification using neural networks ", Int. J. Control Vol.51 No.6, pp. 1191-1214.

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

[CYB89] CYBENKO G. (1989) " Approximation by superpositions of a sigmoïdal function ", Mathematics of Control, Signals and Systems 2, pp. 303-314.

[ETA94] Analyse des essais de guidage-pilotage du véhicule REMI, ETAS 24-30 mars 1994, document Confidentiel Industrie SAGEM SE 96/22/94.

[FAR93] FARGEON C. & QUIN J.-P. (1993) Robotique mobile, Teknea.

[FRA93] FRAPPIER G. (1993) " MINERVE : navigation - guidage - pilotage de véhicules autonomes ", Journée thématique DRET : " Vers une plus grande autonomie des robots mobiles ", janvier 1993, Paris.

[FUN89] FUNAHASHI K. (1989) " On the approximate realization of continous mappings by neural networks ", Neural Networks Vol.2, pp. 183-192.

[GOO84] GOODWIN G. C. & SIN K. S. (1984) Adaptive filtering prediction and control, Prentice-Hall, New Jersey.

[GRO94] GRONDIN B. (1994) " Les réseaux de neurones pour la modélisation et la conduite des réacteurs chimiques : simulations et expérimentations ", Thèse de doctorat de l'Université de Bordeaux I.

[HED91] HEDRICK J. K., McMAHON D., NARENDRAN V. & SWAROOP D. (1991) " Longitudinal vehicle controller design for IVHS systems ", Proceedings of 1991 American Control Conference, Boston, MA, June 1991, pp. 3107-3112.

[HOR89] HORNIK K., STINCHCOMBE M. & WHITE H. (1989) " Multilayer feedforward networks are universal approximators ", Neural Networks 2, pp. 359-366.

[HUN92] HUNT K. J . & SBARBARO D. (1992) " Studies in neural network based control ", in [HUS92], pp. 94-122.

[HUS92] HUNT K. J . & SBARBARO D. (1992) Neural networks for control and systems, Peter Peregrinus Ltd., London.

[JOR85] JORDAN M. I. (1985) " The learning of representations for sequential performance ", Doctoral Dissertation, University of California, San Diego.

[JUR93] JURIE F., RIVES P., GALLICE J. & BRAME J. L. (1993) " High speed vehicle guidance based on vision ", Preprints of the 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 205-210.

[KOS92] KOSKO B. (1992) Neural networks and fuzzy systems, Prentice Hall, Englewood Cliffs, New Jersey.

[LAN93] LANDAU I. D. (1993) Identification et commande des systèmes, 2ème édition, Hermès, Paris.

[LAR89] LARMINAT P. de (1989) " La commande robuste : un tour d'horizon ", Journées nationales du G. R. " Automatique ", 26-28 avril 1989, Biarritz.

[LAR93] LARMINAT P. de (1993) Automatique ; Commande des systèmes linéaires, Hermes.

[LAT91] LATOMBE J. C. (1991) Robot motion planning, Kluwer Academic Publishers, Boston.

[LEO85a] LEONTARITIS I. J. & BILLINGS S. A. (1985) " Input-output parametric models for non-linear systems. Part I : deterministic non-linear systems ", Int. J. Control Vol.41 No.2, pp. 303-328.

[LEO85b] LEONTARITIS I. J. & BILLINGS S. A. (1985) " Input-output parametric models for non-linear systems. Part II : stochastic non-linear systems ", Int. J. Control Vol.41 No.2, pp. 329-344.

[LEO87] LEONTARITIS I. J. & BILLINGS S. A. (1987) " Model selection and validation methods for non-linear systems ", Int. J. Control Vol.45 No.1, pp. 311-341.

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

[LEV93] LEVIN A. U. & NARENDRA K. S. (1993) " Control of nonlinear dynamical systems using neural networks : Controllability and stabilization ", IEEE Trans. on Neural Networks Vol.4 N°2, pp. 1011-1020.

[LIU93] LIUBAKKA M. K., RHODE D. S., WINKELMAN J. R. & KOKOTOVIC P. V. (1993) " Adaptive automotive speed control ", IEEE Trans. on Robotics and Automation Vol.38 No.7, pp. 1-9.

[LJU87] LJUNG L. (1987) System identification ; theory for the user, Prentice Hall, Englewood Cliffs, New Jersey.

[LJU91] LJUNG L. (1991) " Issues in system identification ", IEEE Control Systems Magazine Vol.11 N°1, pp. 25-29.

[MCU43] McCULLOCH W. & PITTS W. (1943) " A logical calculus of the ideas immanent in the brain ", Bull. Math. Biophysics No.5.

[MIL91] MILLER W. T., SUTTON R. S. & WERBOS P. J. (1991) Neural networks for control, MIT Press, Cambridge MA.

[MIN83] MINOUX M. (1983) Programmation mathématique ; théorie et algorithmes, Dunod.

[MIN93a] MINERVE.3 (1993) Contrôle de la direction. Rapport intermédiaire. Contrat DRET N°91.426. Mars 1993.

[MIN93b] MINERVE.3 (1993) Contrôle de la vitesse. Rapport intermédiaire. Contrat DRET N°91.426. Octobre 1993.

[MIN94] MINERVE.3 (1994) Rapport de synthèse. Contrat DRET N°91.426. Février 1994.

[MOR89] MORARI M. & ZAFIRIOU E. (1989) Robust process control, Prentice-Hall International Editions.

[MOR93] MORARI M. (1993) " Some control problems in the process industries ", in [TRE93], pp. 55-78.

[NAR90] NARENDRA K. S. & PARTHASARATHY K. (1990) " Identification and control of dynamical systems using neural networks ", IEEE Trans. on Neural Networks vol.1 No.1, pp. 4-27.

[NAR91] NARENDRA K. S. & PARTHASARATHY K. (1991) " Gradient methods for the optimization of dynamical systems containing neural networks ", IEEE Trans. on Neural Networks Vol.2 No.2, pp. 252-262.

[NAR92] NARENDRA K. S. (1992) " Adaptive control of dynamical systems using neural networks ", in Handbook of intelligent control : neural, fuzzy and adaptive approaches, D. A. White & D. A. Sofge eds., Van Nostrand Reinhold, New York.

[NAS90] NASH J. C. (1990) Compact numerical methods for computers : linear algebra and function minimisation, Adam Hilger.

[NER92a] NERRAND O., ROUSSEL-RAGOT P., PERSONNAZ L. & DREYFUS G. (1992) " Neural networks and non-linear adaptive filtering : unifying concepts and new algorithms ", Neural Computation Vol.5 No.2, pp. 165-199.

[NER92b] NERRAND O. (1992) " Réseaux de neurones pour le filtrage adaptatif, l'identification et la commande de processus ", Thèse de doctorat de l'Université Paris VI.

[NER93] NERRAND O, PERSONNAZ L. & DREYFUS G. (1993) "Non-linear recursive identification and control by neural networks : a general framework", European Control Conference 1993.

[NOR88] NORTON J. P. (1988) An introduction to identification, Academic Press, London.

[PEN92] PENG H., HESSGURG T., TOMIZUKA M., ZHANG W., LIN Y., DEVLIN P. & SHLADOVER S. E. (1992) " A theoretical and experimental study on vehicle lateral control ", Proceedings of the 1992 American Control Conference, Chicago, June 1992, pp. 1738-1742.

[PLO94] PLOIX J.-L., DREYFUS G., CORRIOU J.-P., PASCAL D. (1994) " From knowledge-based models to recurrent networks : an application to an industrial distillation process ", NeuroNîmes'94.

[PLU94] PLUMER E. S. (1994) " Optimal control of terminal processes using neural networks ", soumis à IEEE Transactions on Neural Networks.

[POI02] POINCARÉ H. (1902) La science et l'hypothèse, Flammarion.

[POM91] POMERLEAU D. A. (1991) " Efficient training of artificial neural networks for autonomous navigation ", Neural ComputationVol.1 No.3, pp. 88-97.

[POM92] POMET J.-B., THUILOT B., BASTIN G. & CAMPION G. (1992) " A hybrid strategy for the feedback stabilization of nonholonomic mobile robots ", Proc. of the 1992 International Conference on Robotics and Automation, Nice, France, mai 1992, pp. 129-134.

[POM94] POMERLEAU D. A. (1994) " Reliability estimation for neural network based autonomous driving ", Robotics and Autonomous Systems 12, pp. 113-119.

[PSI91] PSICHOGIOS D. C. & UNGAR L. H. (1991) " Direct and indirect model based control using artificial networks ", Ind. Eng. Chem. Res. Vol.30 N°12, pp. 2564-2573.

[RIC91] RICHALET J. (1991) " Model based predictive control ", Cours de l'École supérieure d'Électricité.

[RIN93] RINTANEN K. T. (1993) " Curvature-optimal path planning and servoing for autonomous vehicles : a neural net implementation ", 1st IFAC International Workshop on Intelligent Autonomous Vehicles, University of Southampton, 18-21 april 1993.

[RIV93a] RIVALS I. (1993) " Application des réseaux de neurones au pilotage automatique de véhicules ", document Confidentiel Industrie SAGEM SE 96/01/93.

[RIV93b] RIVALS I., PERSONNAZ L., DREYFUS G. & CANAS D. (1993) " Real-time control of an autonomous vehicle : a neural network approach to the path-following problem ", 5th International Conference on Neural Networks and their Applications, pp. 219-229 (NeuroNîmes'93).

[RIV94] RIVALS I., CANAS D., PERSONNAZ L. & DREYFUS G. (1994) " Modeling and control of mobile robots and intelligent vehicles by neural networks ", IEEE Conference on Intelligent Vehicles, 24-26 octobre 1994, Paris, pp. 137-142.

[RUM86] RUMELHART D. E., HINTON G. E. & WILLIAMS R. J. (1986) " Learning internal representations by error back-propagation ", In Parallel Distributed Processing : explorations in the microstructure of cognition. Vol.1 : Foundations, D. E. Rumelhart, J. L. McLelland and the PDP Research Group eds., MIT Press, Cambridge MA, pp. 318-362.

[SAM90] SAMSON C. & AIT-ABDERRAHIM K. (1990) " Feedback control of a non-holonomic wheeled cart in cartesian space ", rapport de recherche INRIA N°1288, octobre 1990. Cité comme : IEEE Int. Conf. on Robotics and Automation, Sacramento, California, April 1991.

[SAM91a] SAMSON C. & AIT-ABDERRAHIM K. (1991) " Feedback stabilization of a non-holonomic wheeled mobile robot ", Int. Workshop on Intelligent Robots and Systems (IROS'91), Osaka, Japan.

[SAM91b] SAMSON C. (1991) " Velocity and torque feedback control of a non-holonomic cart ", in Advanced Robot Control, Proc. of the Int. Workshop on non-linear and adaptive control : Issues in Robotics, Grenoble, Nov. 21-23, 1990, vol.162, C. Canudas de Wit ed., Springer-Verlag.

[SAM91c] SAMSON C. (1991) " Time-varying feedback stabilization of non-holonomic car-like mobile robots ", rapport de recherche INRIA N°1515, Septembre 1991.

[SAM92] SAMSON C. (1992) " Path following and time-varying feedback stabilisation of a wheeled mobile robot ", Proc. Conf. ICARCV'92, Singapore, September 1992.

[SBA93] SBARBARO-HOFER D., NEUMERKEL D. & HUNT K. (1993) " Neural control of a steel rolling mill ", IEEE Control Systems Vol.13 No.3, June 1993, pp. 69-75.

[SCO92] SCOTT G. M., SHAVLIK J. W. & RAY W. H. (1992) " Refining PID controllers ", Neural Computation 4, pp. 746-757.

[SHI90] SHIN D. H. & SINGH S. (1990) " Vehicle and path models for autonomous navigation ", in Vision and navigation : the Carnegie Mellon Navlab, Thorpe C. E. ed., Kluwer Academic Publishers, Boston/Dordrecht/London, pp. 283-307.

[SHL78] SHLADOVER S. E. (1978) " Longitudinal control of automated guideway transit vehicle within platoons ", Journal of dynamic systems, measurement and control, Vol.100, December 1978, pp. 302-310.

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

[SJÖ94] SJÖBERG J., HJALMARSSON H. & LJUNG L. (1994) " Neural networks in system identification ", Report LiTH-ISY-R-1622.

[SLO93] SLOTINE J.-J. E. & SANNER R. M. (1993) " Neural networks for adaptive control and recursive identification ", in [TRE93], pp. 381-436.

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

[SOR92] SORDALEN O. J. & CANUDAS DE WIT C. (1992) " Evaluation of an exponential control law for a mobile robot : from regulation to tracking ", Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France, mai 1992.

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

[URB94] URBANI D., ROUSSEL-RAGOT P., PERSONNAZ L. & DREYFUS G. (1994) " The selection of neural models of non-linear dynamical systems by statistical tests ", Neural Networks for Signal Processing, Proceedings of the 1994 IEEE Workshop, pp. 229-237.

[VDB93] Van den BOGAERT T., LEMOINE P., VACHERAND F. & DO S. (1993) " Obstacle avoidance in PANORAMA ESPRIT II project ", 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 48-53.

[VDM93] VAN DER MOLEN G. M. (1993) " Modelling and control of a wheeled mobile robot ", Preprints of the 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 289-294.

[WAI89] WAIBEL A., HANAZAWA T., HINTON G., SHIKANO K. & LANG K. (1989) " Phoneme recognition using Time-Delay Neural Networks ", IEEE Trans. on Acoustics, Speech, and Signal Processing Vol.37, pp. 328-339.

[WHI92] WHITE D. A. & JORDAN M. I. (1992) " Optimal control : a foundation for intelligent control ", in [WHS92], pp. 185-214.

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

Annexe I

APPRENTISSAGE DES RÉSEAUX DE NEURONES

Position du problème.

Nous considérons ici le problème pratique de l'apprentissage d'un réseau de neurones bouclé décrit par la représentation d'état générale (forme canonique, chapitre 1 §I.2) :

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

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

La tâche du réseau est définie par (chapitre 1 §III.2) :

- des séquences d'apprentissage constituées d'une séquence appliquée aux entrées externes {I(k)}, et une séquence de valeurs désirées correspondantes {D(k)} pour les sorties du réseau ;
- une fonction de coût, définie à l'itération i sur une fenêtre fixe englobant toute la longueur N de la séquence d'apprentissage :

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

où C représente les coefficients du réseau C(i-1) disponibles à l'itération i, $E^i(k)$ le vecteur des erreurs à l'instant k et à l'itération i, W une matrice définie positive (souvent diagonale), D(k) le vecteur des sorties désirées à l'instant k, et $Y^i(k)$ le vecteur des sorties du réseau à l'instant k et à l'itération i.

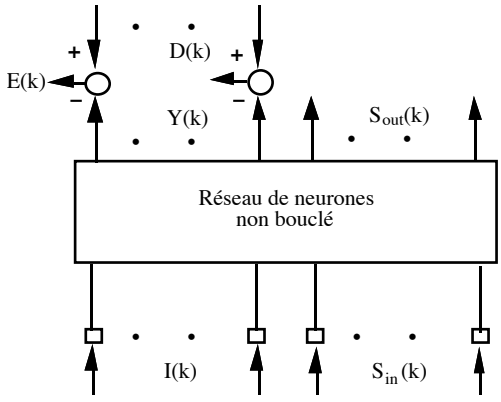


Figure 1.
Copie n°k utilisée pour l'apprentissage.

Notations.

Si l'on développe " spatialement " le comportement temporel du réseau [NER92b], on obtient un grand réseau non bouclé, dans lequel le réseau non bouclé de la forme canonique intervient N fois en cascade avec les coefficients disponibles à l'itération i. Les multiples interventions de ce même réseau sont appelées " *copies* ". Seules les valeurs des entrées et des sorties des neurones sont différentes d'une copie à l'autre. La copie numéro k est représentée sur la figure 1 (en omettant les indices i se référant à l'itération i).

Les entrées de la copie k sont :

- les entrées externes à l'instant k, I(k) $\mathbb{R}^{NI}$ (séquence d'apprentissage),
- les variables d'état d'entrée à l'instant k, $S^{in}(k) \in \mathbb{R}^{NS}$, avec $S^{in}(k)=S^{out}(k-1)$.

Les sorties de la copie k sont :

- les sorties à l'instant k, $Y(k) \in \mathbb{R}^{NY}$, qui sont les activités de NY neurones de sortie,
- les variables d'état de sortie à l'instant k, $S^{out}(k) \in \mathbb{R}^{NS}$, qui sont les activités de NS neurones d'état.

$D(k) \in \mathbb{R}^{NY}$ est le vecteur des sorties désirées à l'instant k (séquence d'apprentissage).

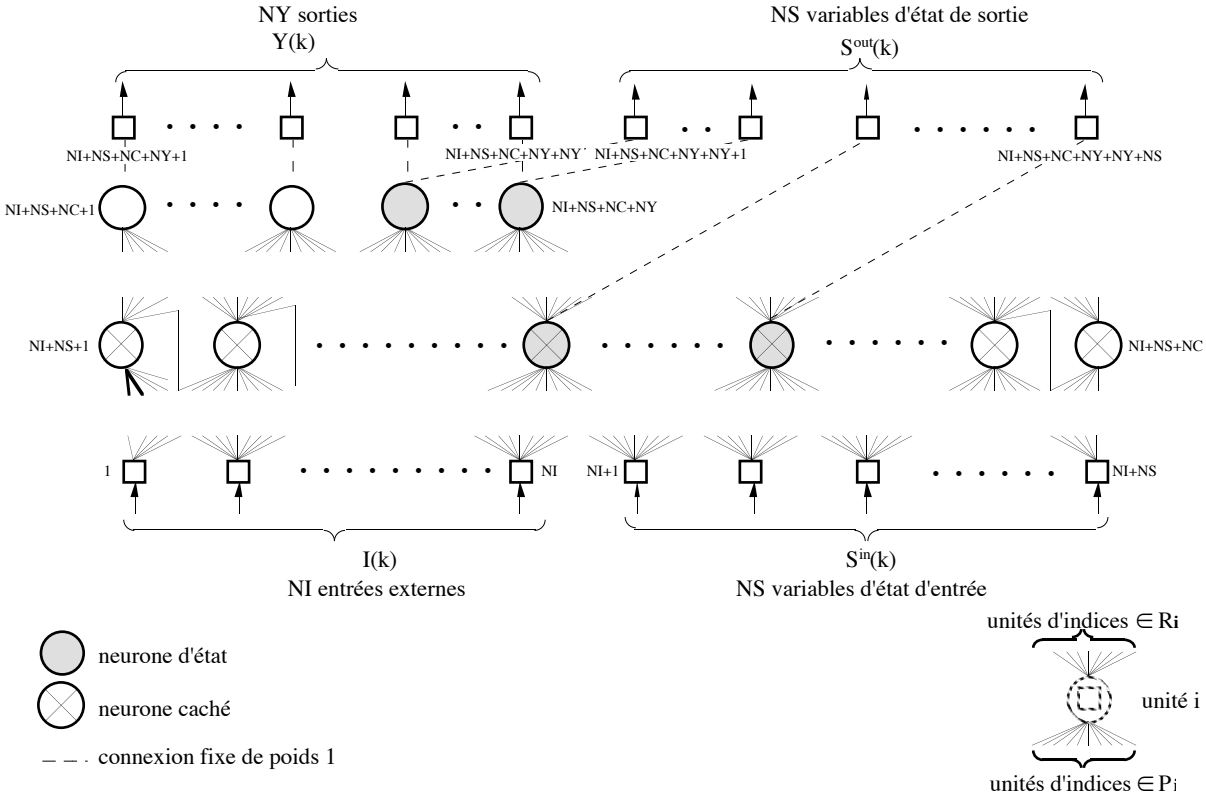


Figure 2.
Détail de la copie n°k utilisée pour l'apprentissage.

Pour faciliter la présentation modulaire du calcul du gradient, nous distinguons les neurones d'unités particulières dont la fonction d'activation est l'identité, appelés cellules (représentées par des carrés sur la figure 2). Les entrées sont ainsi des cellules d'entrée. Les valeurs des sorties de la copie sont les valeurs des activités de NY cellules de sortie, connectées aux NY neurones de sortie avec un coefficient 1. De même, les valeurs des variables d'état de sortie sont les valeurs des activités de NS cellules d'état de sortie, connectées aux NS neurones d'état avec un coefficient 1 (un neurone d'état

est un neurone caché ou un neurone de sortie). Chaque copie est ainsi composée de $NI+NS+N_C+NY+NY+NS$ unités :

- NI cellules d'entrées externes (numérotées de 1 à NI) ;
- NS cellules d'état d'entrée (numérotées de NI+1 à NI+NS) ;
- NC neurones cachés (ordonnés de NI+NS+1 à NI+NS+NC) ;
- NY neurones de sortie (ordonnés de NI+NS+NC+1 à NI+NS+NC+NY) ;
- NY cellules de sortie (numérotées de NI+NS+NC+NY+1 à NI+NS+NC+NY+NY) ;
- NS cellules d'état de sortie (numérotées de NI+NS+NC+NY+NY+1 à NI+NS+NC+NY+NY+NS).

Il est commode de définir la connectivité du réseau par les P_i, ensemble des indices des unités *p*ropageant leur activité vers l'unité i, et les R_i, ensemble des unités *r*ecevant l'activité du neurone i. En particulier, l'ensemble P_i d'une cellule d'état de sortie est constitué de l'indice du neurone d'état auquel elle correspond, et l'ensemble P_i d'une cellule de sortie est constitué de l'indice du neurone de sortie auquel elle correspond. De même, l'ensemble R_i d'un neurone de sortie est constitué de l'indice de la cellule de sortie à laquelle il correspond, et, si c'est aussi un neurone d'état, de l'indice de la cellule d'état de sortie à laquelle il correspond.

Exemple.

À titre illustratif, la forme canonique associée à un prédicteur bouclé entrée-sortie de la forme :

$$y(k+1) = \varphi_{RN}\left(y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-m+1); C\right)$$

est représentée sur la figure 3, avec les mêmes notations que celles de la figure 2.

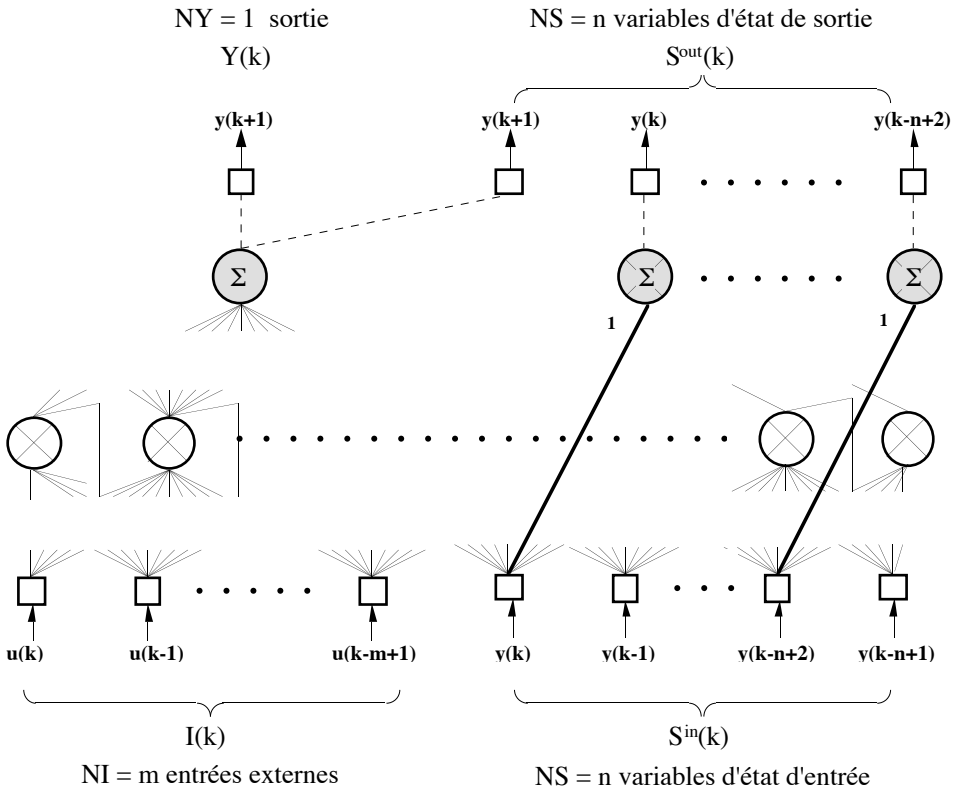


Figure 3.
Copie n°k utilisée pour l'apprentissage d'un prédicteur bouclé entrée-sortie.

Le neurone de sortie est linéaire, ainsi que les neurones d'état. Ils ne possèdent qu'une connexion de poids 1 fixé sur les variables d'état d'entrée (valeurs passées de la sortie) auxquelles ils correspondent. Dans la suite, nous omettons ces neurones sur les schémas (voir les notations du chapitre 2 §I.1, le chapitre 3, et les notations du chapitre 5).

I. CALCUL DE LA FONCTION DE COÛT ET DE SON GRADIENT.

Calcul de la fonction de coût par propagation.

La propagation consiste à calculer les valeurs des potentiels $\{v_i(k)\}$ et des activités $\{z_i(k)\}$ de toutes les unités du réseau ainsi que les erreurs de sortie $\{E_i(k)\}$ à chaque instant k, donc pour chaque copie (k est le numéro de la copie), de l'instant 1 à l'instant N. On peut ainsi calculer la fonction de coût définie pour une itération, dont nous rappelons l'expression en omettant l'indice de l'itération :

$$J(C) = \frac{1}{2} \sum_{k=1}^{N} \sum_{i=1}^{NY} w_{ii}\, E_i^2(k)$$

Calcul du gradient de la fonction de coût par rétro-propagation.

La rétro-propagation consiste à calculer à chaque itération de l'algorithme les valeurs $\{\partial J/\partial v_i(k)\}$ des dérivées partielles de la fonction de coût par rapport au potentiel de chaque unité du réseau à tout instant k, de la dernière unité de chaque copie à la première, de l'instant N à l'instant 1, d'où la dénomination de rétro-propagation[1]. Ces dérivées permettent ensuite le calcul des dérivées de la fonction de coût par rapport aux coefficients. En effet, écrivons la différentielle de J(C) :

$$dJ(C) = \sum_{i,\,j} \sum_{k=1}^{N} \frac{\partial J}{\partial c_{ij}(k)}\; dc_{ij}(k) \quad \text{avec } dc_{ij}(k) = dc_{ij}\ \forall k, \text{ soit} \quad dJ(C) = \sum_{i,\,j} \sum_{k=1}^{N} \frac{\partial J}{\partial c_{ij}(k)}\; dc_{ij}$$

où $c_{ij}(k)$ représente le coefficient c_{ij} dans la copie numéro k. $c_{ij}(k)$ n'est qu'une écriture, puisque les coefficients sont les mêmes pour chaque copie de la fenêtre de la fonction de coût, donc pour chaque instant k. La composante du gradient de J relative au coefficient c_{ij} s'écrit :

$$\left(\frac{\partial J}{\partial c_{ij}}\right)_{\substack{\forall p,q \neq i,j \\ c_{pq}\,constant}} = \sum_{k=1}^{N} \left(\frac{\partial J}{\partial c_{ij}(k)}\right)_{\substack{\forall p,q \neq i,j \\ c_{pq}\,constant}}$$

La rétro-propagation consiste à poser le calcul de la manière suivante :

$$\frac{\partial J}{\partial c_{ij}(k)} = \frac{\partial J}{\partial v_i(k)} \frac{\partial v_i(k)}{\partial c_{ij}(k)} = \frac{\partial J}{\partial v_i(k)}\, z_j(k)$$

---

[1] Le calcul par rétro-propagation suppose implicitement la nullité des dérivées de la fonction de coût par rapport aux entrées d'état de la première copie. Pour leur imposer des valeurs non nulles, il faut calculer le gradient par une méthode directe [NER92a]. Or imposer des valeurs non nulles n'est justifié que pour un apprentissage récursif, ce qui ne sera pas le cas dans ce mémoire, où nous traitons l'apprentissage de systèmes non adaptatifs. Nous utilisons donc exclusivement le calcul par rétropropagation, beaucoup plus économe que le calcul direct en nombre d'opérations.

où P_i est l'ensemble des indices des unités propageant leur activité au neurone i. Les valeurs des activités $\{z_j(k)\}$ ont toutes été calculées lors de la propagation. La connaissance des $\{\partial J/\partial v_i(k)\}$ et des $\{z_j(k)\}$ permet donc le calcul du gradient.

Si le réseau n'est pas bouclé (NS=0), chaque vecteur d'erreur E(k) est calculé indépendamment des autres puisque les copies ne se transmettent aucune valeur. De même, la contribution de chaque erreur au gradient est calculée de façon indépendante. La contribution de la copie k à la valeur du gradient dépend des entrées externes I(k) et des valeurs désirées D(k) (N rétro-propagations).

Si le réseau est bouclé, les valeurs des variables d'état sont transmises d'une copie à la suivante :

$$S_i^{in}(k) = S_i^{out}(k\text{-}1)$$

La contribution de chaque erreur au gradient ne peut plus être calculée de façon indépendante. En effet, on a :

$$\frac{\partial J}{\partial S_i^{out}(k)} = \frac{\partial J}{\partial S_i^{in}(k+1)}$$

On effectue dans ce cas une seule rétro-propagation.

I.1. CALCUL DE LA FONCTION DE COÛT, OU PROPAGATION.

* Valeurs des entrées externes : *par définition*, pour i=1 à NI

$$z_i(k) = v_i(k) = I_i(k)$$

* Valeurs des activités des cellules d'état d'entrée : *par définition*, pour i=1 à NS

$$k = 1 : \quad z_{NI+i}(1) = v_{NI+i}(1) = S_i^{in}(1) = \text{valeur imposée}$$

$$k > 1 : \quad z_{NI+i}(k) = v_{NI+i}(k) = S_i^{in}(k) = S_i^{out}(k\text{-}1)$$

La valeur imposée est choisie en fonction du problème particulier considéré, cf chapitres 3 et 5.

* Valeurs des activités des neurones cachés, des neurones de sortie, des cellules de sortie, et des cellules d'état de sortie : *calcul*, pour i variant de NI+NS+1 à NI+NS+NC+NY+NY+NS

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

où P_i est l'ensemble des indices des unités propageant leur activité au neurone i.

* Valeurs des sorties : *par définition,* pour i=1 à NY

$$Y_i(k) = z_{NI+NS+NC+NY+i}(k)$$

* Valeurs des variables d'état de sortie : *par définition*, pour i=1 à NS

$$S_i^{out}(k) = z_{NI+NS+NC+NY+NY+i}(k)$$

* Valeurs des erreurs : *calcul*, pour i=1 à NY de

$$E_i(k) = D_i(k) - Y_i(k)$$

I.2. CALCUL DU GRADIENT DE LA FONCTION DE COÛT, OU RÉTRO-PROPAGATION.

* Valeurs des $\partial J/\partial v_i(k)$ des cellules d'état de sortie : *par définition*, pour i=1 à NS

$$k = N : \quad \frac{\partial J}{\partial v_{NI+NS+NC+NY+NY+i}(N)} = \frac{\partial J}{\partial S_i^{out}(N)} = 0$$

$$k < N : \quad \frac{\partial J}{\partial v_{NI+NS+NC+NY+NY+i}(k)} = \frac{\partial J}{\partial S_i^{out}(k)} = \frac{\partial J}{\partial S_i^{in}(k+1)}$$

* Valeurs des $\partial J/\partial v_i(k)$ des cellules de sortie : *calcul*, pour i=1 à NY

$$\frac{\partial J}{\partial v_{NI+NS+NC+NY+i}(k)} = - w_{ii} E_i(k)$$

* Valeurs des $\partial J/\partial v_i(k)$ des neurones de sortie, des neurones cachés et des cellules d'état d'entrée : *calcul*, pour i décroissant de NI+NS+NC+NY à NI+1

$$\frac{\partial J}{\partial v_i(k)} = f_i^{'}\left(v_i(k)\right) \sum_{h \in R_i} c_{hi} \frac{\partial J}{\partial v_h(k)}$$

où R_i est l'ensemble des indices des unités recevant l'activité du neurone i.

* Valeur de $\partial J/\partial S^{in}$: *par définition*, pour i=1 à NS

$$\frac{\partial J}{\partial S_i^{in}(k)} = \frac{\partial J}{\partial v_{NI+i}(k)}$$

L'algorithme de calcul de la fonction de coût et de son gradient est dit " dirigé " si le réseau est non bouclé (NS=0), " semi-dirigé " sinon. Ceci signifie, dans le premier cas, que le réseau n'a pas d'état, et qu'il est donc entièrement " dirigé " par les entrées externes, et, dans le second cas, que l'état du réseau ne lui est imposé qu'au début de la fenêtre de la fonction d'apprentissage (pour la première copie)[2]. Les notions de dirigé et de semi-dirigé prendront tout leur sens dans les chapitres consacrés à l'apprentissage de prédicteurs et de correcteurs.

I.3. RÉSUMÉ DES CALCULS.

La figure 4 résume les calculs à effectuer pour chaque copie k, k variant de 1 à N, horizon sur lequel la fonction de coût est définie. La propagation, ou calcul de la fonction de coût par le réseau de propagation FF, doit être effectuée dans tous les blocs avant la rétro-propagation, ou calcul du gradient de la fonction de coût par le réseau BP, sauf si le réseau n'est pas bouclé (les flèches grisées disparaissent).

---

[2] Dans le cas récursif, un troisième algorithme, dit " non-dirigé ", est également utilisé [NER92a].

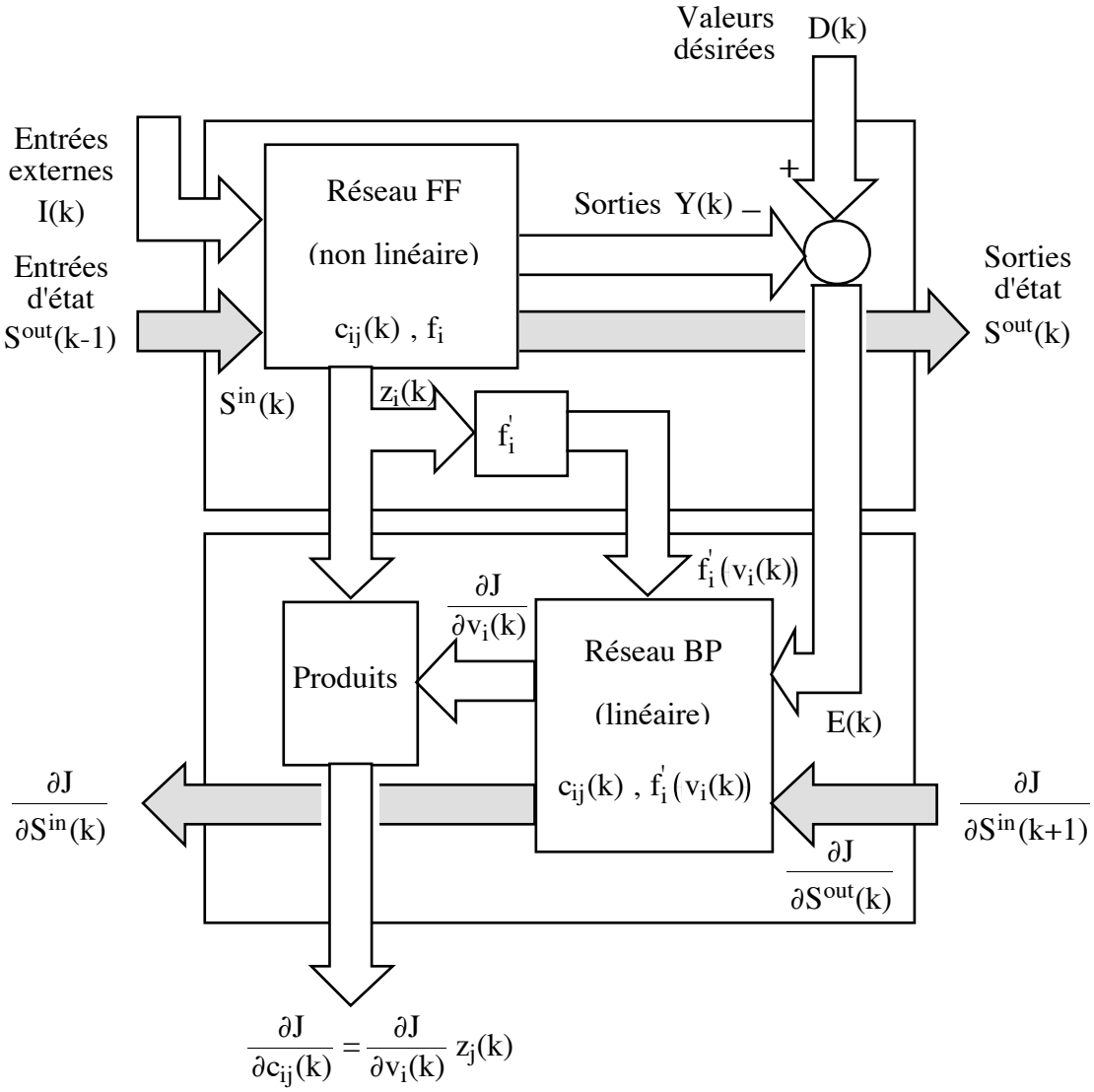


Figure 4.
Calculs correspondants à la copie k.

II. MODIFICATION DES COEFFICIENTS.

Dans le cas général, le calcul de la modification des coefficients (dont la valeur n'est pas fixée[3]) à l'itération i s'écrit :

$$\Delta C = + \mu D$$

où D est une direction de descente,
$\mu>0$ est le pas de descente.

[3] Dans le cas de l'apprentissage d'un correcteur en partculier, le système d'apprentissage utilise un modèle de simulation, dont les coefficients ne doivent pas être modifiés.

II.1. MÉTHODE DE GRADIENT À PAS CONSTANT.

La méthode la plus couramment utilisée est celle où D est l'opposé du gradient et μ est constant. Les modifications des coefficients sont données par l'expression :

$$\Delta c_{ij} = - \mu \frac{\partial J}{\partial c_{ij}} = - \mu \sum_{k=1}^{N} \frac{\partial J}{\partial c_{ij}}(k) = - \mu \sum_{k=1}^{N} \frac{\partial J}{\partial v_i(k)} z_j(k)$$

Cette méthode présente l'inconvénient d'avoir une vitesse de convergence très lente lorsque l'on s'approche du minimum. Nous utilisons les méthodes suivantes qui permettent d'accélérer la convergence de façon appréciable.

II.2. MÉTHODES À PAS VARIABLE.

Quelle que soit la direction de descente utilisée, il est possible d'asservir le pas μ de telle sorte que la fonction de coût diminue à chaque modification des coefficients. Nous avons utilisé deux méthodes de minimisation unidimensionnelle " économiques ", les méthodes de Nash [NAS90] et de Wolfe et Powell [MIN83]. Ces méthodes permettent d'obtenir un pas convenable avec un nombre très limité d'évaluations de la fonction de coût et du gradient J. Notons C_i les coefficients, D_i la direction de descente, et μ_i le pas de descente à l'itération i. Soit la fonction g définie par : $g\ (\mu_i) = J(C_i + \mu_i D_i)$. Sa dérivée s'écrit : $g'(\mu_i) = D_i^T \nabla J(C_i + \mu_i D_i)$. g est donc la fonction dont on cherche le minimum en fonction du pas μ_i.

a) Règle de Nash.

Le principe de cette méthode est que le pas de descente ne doit pas être choisi trop grand sinon l'algorithme risque d'avoir un comportement oscillatoire. Le pas de descente μ_i doit être choisi de façon telle que :

$$g(\mu_i) \leq g(0) + m_1 \mu_i g'(0)$$

soit :

$$J(C_i + \mu_i D_i) \leq J(C_i) + m_1 \mu_i D_i^T \nabla J(C_i)$$

Cette condition assure la propriété de descente ; m_1, le facteur de tolérance, est un nombre petit devant 1 (par exemple 10^{-3}). Si la condition ci-dessus n'est pas satisfaite, le pas μ_i est multiplié par un facteur de réduction (Nash propose 0,2). En pratique, avec les valeurs précédentes, on se limite à une vingtaine d'évaluations de J, après quoi les modifications effectuées sont de l'ordre de grandeur des erreurs d'arrondis.

b) Règle de Wolfe et Powell.

(a) Le pas de descente ne doit pas être choisi trop grand sinon l'algorithme risque d'avoir un comportement oscillatoire.

(b) Le pas de descente ne doit pas être choisi trop petit pour permettre une convergence rapide de l'algorithme.

μ_i doit être choisi de façon telle que :

(1) $g(\mu_i) \leq g(0) + m_1\, \mu_i\, g'(0)$ avec $m_1 \in]0,1[$

Cette première règle assure la propriété de descente.

(2) $g'(\mu_i) \geq m_2\, g'(0)$ avec $m_2 \in]m_1,1[$

La seconde règle permet de s'assurer que le pas n'est pas non plus choisi trop petit, comme le montre la figure 5. Les deux règles s'écrivent, en fonction du critère et de son gradient :

(1) $J(C_i + \mu_i D_i) \leq J(C_i) + m_1\, \mu_i\, D_i^T \nabla J(C_i)$ avec $m_1 \in]0,1[$

(2) $D_i^T \nabla J(C_i + \mu_i D_i) \geq m_2\, D_i^T \nabla J(C_i)$ avec $m_2 \in]m_1,1[$

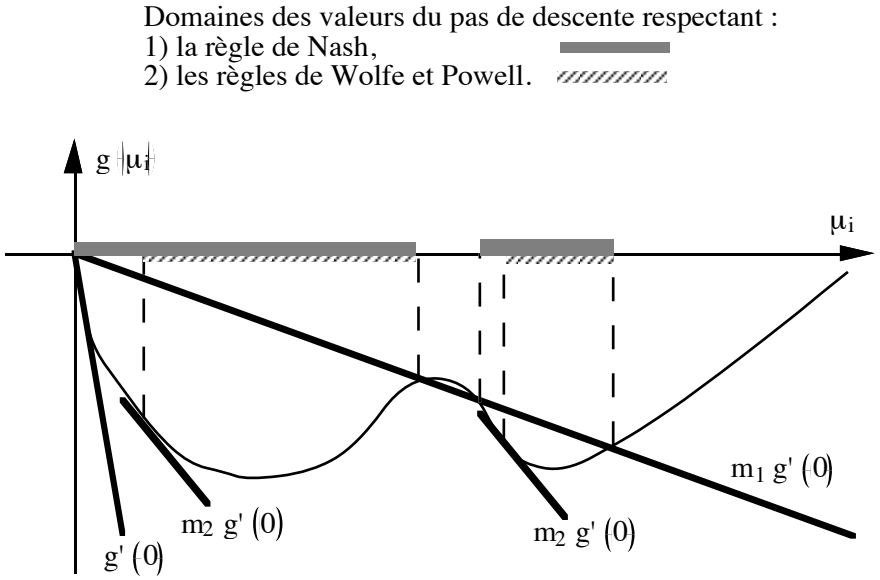


Figure 5.
Optimisation unidimensionnelle : règles de Nash et de de Wolfe et Powell (coupe dans la direction de descente D_i).

Pratiquement, la procédure est la suivante :

Initialiser les valeurs : $\mu=1$; $\mu_{min}=\mu^0_{min}$, ; $\mu_{max}=\mu^0_{max}$.

a) Calculer $J(C_i + \mu D_i)$.

b) Si la condition (1) est vérifiée, aller en c); sinon $\mu_{max}=\mu$ (μ est trop grand) et aller en d).

c) Si la condition (2) est vérifiée $\mu_i=\mu$ est accepté ; sinon $\mu_{min}=\mu$ (μ est trop petit) et aller en d).

d) Modifier le pas μ : $\mu = \frac{\mu_{min} + \mu_{max}}{2}$ et aller en a).

On prend par exemple : $m_1 = 0{,}1$; $m_2 = 0{,}7$; $\mu_{min}=0$; $\mu_{max}=10$ (ce choix n'est pas très critique).

La mise en œuvre de cette méthode est donc un peu plus lourde que la précédente.

II.3. MÉTHODE QUASI NEWTONIENNE À PAS VARIABLE.

Nous avons également employé une méthode utilisant une autre direction de descente que le gradient, une méthode quasi newtonienne (ou méthode à métrique variable) utilisant l'algorithme de Broyden, Fletcher, Goldfarb et Shanno (BFGS) dont la vitesse de convergence est beaucoup plus grande que celle du gradient [MIN83].

Méthode de Newton.

La méthode de Newton consiste à remplacer la fonction de coût par son approximation quadratique au voisinage du point courant (la quadrique osculatrice) :

$$q(C) = J(C_i) + \nabla J^T(C_i)(C - C_i) + \frac{1}{2}(C - C_i)^T \nabla^2 J(C_i)(C - C_i)$$

et à choisir les coefficients C_{i+1} de manière à minimiser q(C). Le minimum existe si le Hessien $\nabla^2 J(C_i)$ est défini positif. Ce qui conduit à la formule itérative :

$$C_{i+1} = C_i - \left[\nabla^2 J(C_i)\right]^{-1} \nabla J(C_i)$$

La méthode de Newton demande donc de calculer puis d'inverser le Hessien à chaque itération. Notons que le pas de descente est fixé (μ=1). Appliquée à une fonction J des coefficients C quadratique strictement convexe, cette méthode converge en une seule itération, mais dans le cas général d'une fonction non linéaire quelconque, elle ne possède pas la propriété de convergence globale. Si les coefficients de départ sont trop éloignés du minimum, la méthode peut ne pas converger.

Méthode de quasi-Newton.

Le principe des méthodes quasi newtoniennes consiste en une généralisation de la formule itérative de Newton :

$$C_{i+1} = C_i + \mu_i D_i \text{ avec } D_i = - H_i \nabla J(C_i)$$

où H_i est une matrice définie positive donnant la direction de descente à partir du gradient $\nabla J(C_i)$, et μ_i est choisi tel que $J(C_i + \mu_i D_i) < J(C_i)$, donc par exemple par les méthodes de Nash ou de Wolfe et Powell. La matrice H_i est une approximation de l'inverse du Hessien. En effet, elle est modifiée à chaque itération de telle manière que, pour une fonction quadratique, elle converge vers l'inverse du Hessien $\left[\nabla^2 J(C_i)\right]^{-1}$. Différentes formules du type :

$$H_{i+1} = H_i + \Delta_i$$

ont été proposées. L'une d'elle est la formule BFGS suivante :

$$H_{i+1} = H_i + \left[1 + \left(\frac{\gamma_i^T H_i \gamma_i}{\Delta C_i^T \gamma_i}\right)\right] \frac{\Delta C_i^T \Delta C_i}{\Delta C_i^T \gamma_i} - \frac{\Delta C_i \gamma_i^T H_i + H_i \gamma_i \Delta C_i^T}{\Delta C_i^T \gamma_i}$$

avec $\gamma_i = \nabla J(C_{i+1}) - \nabla J(C_i)$ et $\Delta C_i = C_{i+1} - C_i$.

Si la matrice H_{i+1} ainsi calculée n'est pas définie positive, elle est réinitialisée à la matrice identité. On repart alors avec $D_i = - \nabla J(C_i)$, c'est-à-dire dans la direction opposée à celle du gradient. Un des principaux attraits de la méthode BFGS est qu'elle est relativement insensible au choix du pas. Comme nous l'avons précisé ci-dessus, il suffit que μ_i soit tel que $J(C_i + \mu_i D_i) < J(C_i)$. On peut donc se contenter de la méthode de Nash, la plus simple des méthodes présentées.

Annexe II

SYSTÈMES DE COMMANDE LINÉAIRE PAR SIMPLE BOUCLAGE ET AVEC MODÈLE INTERNE

INTRODUCTION.

Cette annexe établit l'expression des correcteurs-S et -D dans le cas de modèles discrets linéaires mono-entrée/mono-sortie du processus. Puis elle étudie les propriétés de systèmes de commande utilisant ces correcteurs, des systèmes de commande par simple bouclage (SCSB), ou avec modèle interne (SCMI). Le but de cette étude est, outre une récapitulation utile des résultats en linéaire, leur généralisation partielle au cas de modèles et de correcteurs non linéaires, par exemple des réseaux de neurones, généralisation mise à profit au chapitre 5.

I. CORRECTEURS-S ET -D.

Les correcteurs-S et -D sont particuliers aux modèles à temps discret. Soit un système de commande constitué d'un tel modèle de retard d, en simple bouclage avec le correcteur-S ou -D. Les définitions des correcteurs sont les suivantes :

Correcteur-S : il impose au système de commande une sortie de référence.

Soit une trajectoire de référence $\{y_r(k)\}$. Le correcteur-S délivre à chaque instant k une commande telle que, quel que soit l'état du modèle à cet instant, et si aucune perturbation n'intervient ensuite, *la sortie du système de commande est égale à la sortie de référence* à partir de l'instant k+d :

$$y(k') = y_r(k') \qquad \forall k' \geq k+d$$

Correcteur-D : il impose au système de commande une dynamique de référence.

Soit une trajectoire de consigne {r(k)}, et une dynamique donnée par un modèle de référence linéaire de retard d de sortie y_r telle que $E(q)\, y_r(k+d) = H(q)\, r(k)$, ou E et H sont deux polynômes en q^{-1}. Le correcteur-D délivre à chaque instant k une commande telle que, quel que soit l'état du modèle à cet instant, et si aucune perturbation n'intervient ensuite, *la dynamique du système de commande est égale à la dynamique de référence* à partir de l'instant k+d :

$$E(q)\, y(k') = q^{-d}\, H(q)\, r(k') \qquad \forall k' \geq k+d$$

Notons que le correcteur-S est un cas particulier du correcteur-D, avec $E(q) = H(q) = 1$. Cependant, sa mise en œuvre est suffisamment particulière pour que nous traitions son cas séparément du cas du correcteur-D : en effet, elle nécessite un modèle de référence au sein des systèmes de commande qui l'utilisent pour calculer la trajectoire de référence $\{y_r(k)\}$ à partir de la consigne (voir chapitre 5).

Dans le cas linéaire, il n'est pas nécessaire de traiter le cas des modèles d'état, que l'on peut toujours écrire sous une forme entrée-sortie adéquate [GOO84]. Mais le le but de cette présentation est une généralisation au cas non linéaire : or le modèle non linéaire disponible peut être du type représentation d'état. La mise sous forme entrée-sortie de ce modèle demanderait l'apprentissage d'un modèle entrée-sortie, apprentissage dont les difficultés ont été soulevées au chapitre 2 §I.2.2.3 (ordre du modèle entrée-sortie…). Nous étudions donc aussi l'expression des correcteurs dans le cas où l'on dispose d'un modèle d'état. Les expressions des correcteurs dans le cas d'un modèle linéaire entrée-sortie sont données dans [GOO84].

I.1. EXPRESSIONS DES CORRECTEURS DANS LE CAS D'UN MODÈLE ENTRÉE-SORTIE.

Nous considérons le modèle discret linéaire entrée-sortie de retard d :

$$A(q)\, y(k) = B(q)\, u(k)$$

avec :

$A(q) = 1 + a_1\, q^{-1} + \ldots + a_n\, q^{-n}$

$B(q) = q^{-d}\, B'(q) = q^{-d}\left(b_0 + b_1\, q^{-1} + \ldots + b_{m'}\, q^{-m'}\right)$

où d est le retard du modèle. Le degré du polynôme B est $d + m' = m$.

I.1.1. Correcteur-S.

Comme le modèle possède un retard d, la commande u(k) affecte la sortie à l'instant k+d. Cherchons les arguments en fonction desquels s'exprime la sortie y(k+d) :

- pour calculer y(k+1), on a besoin de y(k), … , y(k-n+1), u(k-d+1), … , u(k-d-m'+1);
- pour calculer y(k+2), on a besoin, en plus, de u(k-d+2);

…

- pour calculer y(k+d), on a besoin, en plus, de u(k), d'où:

$$\begin{aligned} y(k+d) &= \varphi\left(y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-d-m'+1)\right) \\ &= \varphi\left(y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-m+1)\right) \end{aligned}$$

En linéaire, il est facile de calculer la fonction φ, ou prédicteur à d pas de la sortie. Le raisonnement précédent montre qu'il faut effectuer la division 1/A limitée aux d premiers termes :

$\frac{1}{A(q)} = F(q) + q^{-d}\frac{G(q)}{A(q)}$ soit : $1 = F(q)A(q) + q^{-d}\, G(q)$

où F, monique, et G sont les uniques polynômes de degrés d-1 et n-1 satisfaisant cette égalité :

$F(q) = 1 + f_1\, q^{-1} + \ldots + f_{d-1}\, q^{-d+1}$
$G(q) = g_0 + g_1\, q^{-1} + \ldots + g_{n-1}\, q^{-n+1}$
Les coefficients de F et de G sont :
$f_0 = 1$; $f_i = -\sum_{j=0}^{i-1} f_j\, a_{i-j}$ pour i=1 à d-1 ; $g_i = -\sum_{j=0}^{d-1} f_j\, a_{i+d-j}$ pour i=0 à n-1 .

Pour obtenir l'expression du prédicteur, on multiplie celle du modèle par le polynôme F :
$F(q)\, A(q)\, y(k) = F(q)\, q^{-d}\, B'(q)\, u(k)$; comme $F(q)A(q) = 1 - q^{-d}\, G(q)$:
$y(k) = q^{-d}\, G(q)\, y(k) + q^{-d}\, F(q)\, B'(q)\, u(k)$ soit :

$$y(k+d) = G(q)\, y(k) + F(q)\, B'(q)\, u(k)$$

qui est l'expression du prédicteur cherché, encore appelé prédicteur à d pas.

Le correcteur-S est le correcteur qui permet d'obtenir $y(k+d)=y_r(k+d)$; il s'écrit donc :

$$\boxed{u(k) = y_r(k+d) - G(q)\, y(k) + \left(1 - F(q)\, B'(q)\right) u(k)}$$

où G est de degré n-1 et 1–FB' de degré $d-1+m' = m-1$. Ce correcteur est en général bouclé (dès que m>1). Si cette commande est appliquée à l'instant k, et si aucune perturbation n'intervient ensuite, le système bouclé obéit à :

$$B(q)\, u(k) = A(q)\, y_r(k) \qquad \forall k' \geq k+d$$

Le correcteur-S est donc stable si les racines du polynôme B sont à l'intérieur du cercle unité, c'est-à-dire si le modèle est *à minimum de phase*.

Remarque.
Tous les polynômes utilisés dans les expressions de prédicteurs et de correcteurs ont un terme constant (en q^0).

I.1.2. Correcteur-D.

Soit la dymamique de référence donnée par le modèle :

$$E(q)\, y_r(k+d) = H(q)\, r(k)$$

avec :
$E(q) = 1 + e_1\, q^{-1} + \ldots + e_p\, q^{-p}$
$H(q) = h_0 + h_1\, q^{-1} + \ldots + h_p\, q^{-p}$
Le prédicteur nécessaire à la synthèse du correcteur-D s'obtient en effectuant la division de E par A limitée aux d premiers termes :

$$\frac{E(q)}{A(q)} = F(q) + q^{-d}\frac{G(q)}{A(q)} \quad soit \quad E(q) = F(q)\, A(q) + q^{-d}\, G(q)$$

où F, monique, et G sont les deux seuls polynômes de degrés d-1 et n-1 satisfaisant l'équation ci-dessus (ceci suppose p≤d+n-1). Les coefficients de F et de G sont :
$f_0 = 1$; $f_i = -\sum_{j=0}^{i-1} f_j\, a_{i-j} - e_i$ pour i=1 à d-1 ; $g_i = -\sum_{j=0}^{d-1} f_j\, a_{i+d-j} - e_{i+d}$ pour i=0 à n-1 .

Pour obtenir l'expression du prédicteur associé au problème, on multiplie comme précédemment celle du modèle par le polynôme F :
$F(q)\, A(q)\, y(k) = F(q)\, q^{-d}\, B'(q)\, u(k)$; comme $F(q)\, A(q) = E(q) - q^{-d}\, G(q)$:

$E(q)\ y(k) = q^{-d}\ G(q)\ y(k) + q^{-d}\ F(q)\ B'(q)\ u(k)$ soit :

$$E(q)\ y(k+d) = G(q)\ y(k) + F(q)\ B'(q)\ u(k)$$

qui est l'expression du prédicteur cherché.

Le correcteur-D est le correcteur qui permet d'obtenir $E(q)\ y(k+d) = H(q)\ r(k)$: il s'écrit donc :

$$\boxed{u(k) = H(q)\ r(k) - G(q)\ y(k) + \left(1 - F(q)\ B'(q)\right) u(k)}$$

G est un polynôme en q^{-1} de degré n-1et 1–FB' est de degré m–1. Le correcteur est en général bouclé (dès que m>1). Si cette commande est appliquée à l'instant k, et si aucune perturbation n'intervient ensuite, le système bouclé obéit à :

$$E(q)\ B(q)\ u(k) = E(q)\ A(q)\ y_r(k) \qquad \forall k' \geq k+d$$

Pour que le correcteur-D soit stable, il faut donc aussi que le modèle soit *à minimum de phase*.

I.2. Expressions des correcteurs dans le cas d'un modèle d'état.

Comme nous l'avons précisé dans l'introduction, nous traitons le problème dans le cas d'une représentation d'état uniquement en vue de la généralisation aux modèles non linéaires. Nous considérons le modèle discret linéaire d'état suivant :

$$\begin{cases} x(k+1) = A\ x(k) + B\ u(k) \\ y(k) = C\ x(k) \end{cases}$$

où A est une matrice n x n, B une matrice colonne n x 1, et C une matrice ligne 1 x n. Son ordre relatif est d.

I.2.1. Correcteur-S.

Comme dans le cas entrée-sortie, cherchons le prédicteur de la sortie du modèle à d pas, ou d est l'ordre relatif du modèle. Pour un modèle mono-entrée/mono-sortie linéaire ou non, l'ordre relatif est défini comme le retard existant entre l'entrée et la sortie du modèle, c'est-à-dire l'entier d tel que, pour x(k) et u(k) donnés, l'effet de l'entrée u(k) apparaît à la sortie après d pas :

$y(k+1) = CA\ x(k) + CB\ u(k)$

$y(k+2) = CA^2\ x(k) + CAB\ u(k) + CB\ u(k+1)$

…

$$y(k+i) = CA^i\ x(k) + \sum_{j=1}^{i} CA^{j-1}B\ u(k+i-j)$$

L'ordre relatif d est donc la plus petite valeur entière de j telle que : $CA^{j-1}B \neq 0$. Pour un modèle mono-entrée/mono-sortie, $CA^{d-1}B$ est un scalaire. On a :

$$y(k+d) = CA^d\ x(k) + CA^{d-1}B\ u(k)$$

qui est l'expression du prédicteur à d pas pour un modèle d'état.

Le correcteur-S est le correcteur qui permet d'obtenir $y(k+d)=y_r(k+d)$. Son expression est donc :

$$\boxed{u(k) = \frac{1}{CA^{d-1}B}\left(y_r(k+d) - CA^d\ x(k)\right)}$$

Ce correcteur n'est pas bouclé, sa sortie n'est fonction que du signal de référence et de l'état du modèle. La dynamique du système {correcteur+modèle} est :

$$x(k+1) = \left(A - \frac{BCA^d}{CA^{d-1}B}\right) x(k) + \frac{B}{CA^{d-1}B}\, y_r(k+d)$$

Pour que la commande soit applicable (bornée), il est nécessaire que la matrice :

$$A - \frac{BCA^d}{CA^{d-1}B}$$

ait des valeurs propres de module inférieur à 1. Cela signifie en particulier que le modèle doit être *à minimum de phase*.

I.2.2. Correcteur-D.

Par définition du correcteur-D, on désire obtenir :

$$E(q)\, y(k+d) = H(q)\, r(k)$$

soit :

$y(k+d) + e_1\, y(k+d-1) + \ldots + e_p\, y(k+d-p) = H(q)\, r(k)$

Calculons chacun des y en fonction de l'état à l'instant k, quand c'est possible. On a :

$y(k) = C\, x(k)$

$y(k+1) = CA\, x(k)$

…

$y(k+d) = CA^d\, x(k) + CA^{d-1}B\, u(k)$

Il faut donc que :

$$CA^d\, x(k) + CA^{d-1}B\, u(k) + \ldots + e_d\, C\, x(k) + e_{d+1}\, y(k-1) + \ldots + e_p\, y(k+d-p) = H(q)\, r(k)$$

L'expression du correcteur est :

- si p > d :

$$u(k) = \frac{1}{CA^{d-1}B}\left[H(q)\, r(k) - \left(CA^d + e_1\, CA^{d-1} + \ldots + e_d\, C\right) x(k) - \left(e_{d+1}\, y(k-1) + \ldots + e_p\, y(k+d-p)\right)\right]$$

- si p ≤ d :

$$u(k) = \frac{1}{CA^{d-1}B}\left[H(q)\, r(k) - \left(CA^d + e_1\, CA^{d-1} + \ldots + e_p\, C\, A^{d-p}\right) x(k)\right]$$

Ce correcteur n'est pas bouclé : ses arguments sont des valeurs successives du signal de référence (pondérées par le polynôme H), l'état du modèle, et un certain nombre de valeurs de ses sorties antérieures. Si p > d, il est impossible d'exprimer u(k) uniquement en fonction de la consigne et de l'état[4].

---

[4] Pour une expression du correcteur-D formulée comme un retour d'état statique, voir [GOO84], p. 148 et suivantes : l'état choisi s'exprime comme une combinaison linéaire des sorties et des commandes passées. Nous préférons la solution proposée ci-dessus, qui fournit dans tous les cas un correcteur non bouclé, et donc plus facile à réaliser par apprentissage avec un réseau de neurones (on peut utiliser un algorithme dirigé).

II. SYSTÈMES DE COMMANDE POUR LA POURSUITE ET LA RÉGULATION.

Nous étudions les propriétés des systèmes de commande par simple bouclage (SCSB) et avec modèle interne (SCMI) proposés au chapitre 5, dans le cas d'un processus linéaire, sous forme entrée-sortie, *de retard d=1* :

$$A_p(q)\, y_p\,(k) = B_p(q)\, u\,(k) + A_p(q)\, p(k) = q^{-1}\, B_p'(q)\, u\,(k) + A_p(q)\, p(k)$$

où A_p est un polynôme monique de degré n, et B_p un polynôme de degré m. y_p est la sortie du processus, u la commande, et p est une perturbation additive en sortie du processus. Le modèle est stable et à inverse stable, c'est-à-dire que les zéros de A_p et de B_p sont à l'intérieur du cercle unité.

Les systèmes de commandes étudiés sont tous conçus pour que leur dynamique de poursuite, c'est-à-dire la dynamique de suivi de la consigne r(k), soit la dynamique de référence définie par le modèle :

$$E(q)\, y_r(k+1) = H(q)\, r(k)$$

où E et H sont deux polynômes de degré p≥n :

$E(q) = 1 + e_1\, q^{-1} + \ldots + e_p\, q^{-p}$

$H(q) = h_0 + h_1\, q^{-1} + \ldots + h_p\, q^{-p}$

Nous supposons qu'une identification du processus a conduit au modèle linéaire :

$$A(q)\, y\,(k) = B(q)\, u\,(k) = q^{-1}B'(q)\, u\,(k)$$

où A est un polynôme monique de degré n, et B un polynôme de degré m. Les zéros de B sont supposés à l'intérieur du cercle unité.

Enfin, pour rendre compte de l'influence des défauts d'apprentissage du correcteur sur les SCMI, nous considérons que le correcteur n'est généralement pas parfait (parfaitement adapté au modèle).

- D'après le §I.1.1 de cette annexe, l'expression du correcteur-S parfait est (d=1, F=1, B'= q B) :

$$B'(q)\, u(k) = y_r(k+1) + q\left(A(q) - 1\right) y(k)$$

Nous considérons le correcteur suivant :

$$B_c'(q)\, u(k) = y_r(k+1) + q\left(A_c(q) - 1\right) y(k)$$

où A_c est un polynôme monique ; les zéros de B_c sont supposés à l'intérieur du cercle unité (condition nécessaire à la stabilité interne du système de commande).

- D'après le §I.1.2 de cette annexe, le correcteur-D parfait est :

$$B'(q)\, u(k) = H(q)\, r(k) + q\left(A(q) - E(q)\right) y(k).$$

Nous considérons le correcteur suivant :

$$B_c'(q)\, u(k) = H(q)\, r(k) + q\left(A_c(q) - E(q)\right) y(k)$$

Nous étudions les propriétés des systèmes de commande en matière de stabilité, et de performance en poursuite et en régulation :

- pour les SCSB : dans le cas nominal (A_c=A_p, B_c=B_p), et non nominal (A_c≠A_p, B_c≠B_p).
- pour les SCMI : dans le cas nominal (A_p=A, B_p=B), nous étudions l'influence d'une imperfection éventuelle du correcteur (A_c≠A, B_c≠B) ; nous étudions aussi les propriétés du système non nominal, lorsque le correcteur est parfait (A_p≠A, B_p≠B et A_c=A, B_c=B).

II.1. SYSTÈMES DE COMMANDE PAR SIMPLE BOUCLAGE (SCSB).

II.1.1. SCSB utilisant un correcteur-S.

Dans un SCSB, le correcteur-S est mis en cascade avec le processus :

$$B_c'(q)\, u(k) = y_r(k+1) + q\left(A_c(q) - 1\right) y_p(k)$$

où y_p est la sortie mesurée du processus. La sortie de référence y_r est calculée par un modèle de référence qui peut être soit un *modèle de poursuite*, soit un *modèle de ralliement*.

II.1.1.1. Avec un modèle de poursuite.

Le modèle de poursuite s'écrit :

$$E(q)\, y_r(k+1) = H(q)\, r(k)$$

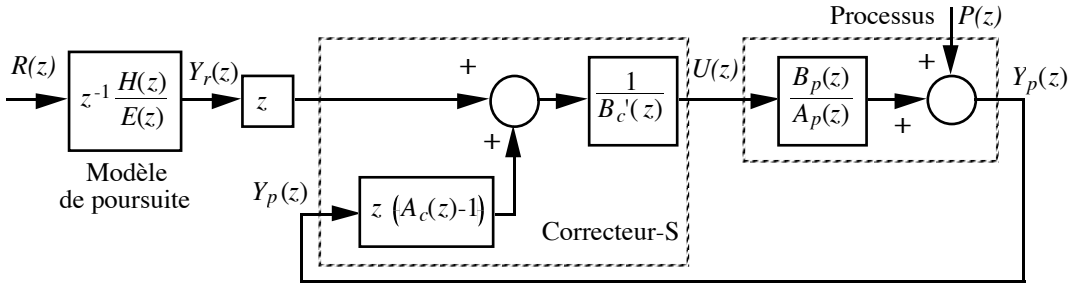


Figure 1.
SCSB avec correcteur-S et modèle de poursuite.

La transformée en z de la commande u de la figure 1, en fonction de la consigne et de la perturbation, est la suivante :

$$U(z) = \frac{A_p(z)}{B_p'(z) + \left(A_p(z)\, B_c'(z) - A_c(z)\, B_p'(z)\right)} \left(\frac{H(z)}{E(z)} R(z) + z\left(A_c(z) - 1\right) P(z)\right)$$

a) Fonctionnement nominal ($A_c = A_p$, $B_c = B_p$) .

$$U(z) = \frac{A_p(z)}{B_p'(z)} \left(\frac{H(z)}{E(z)} R(z) + z\left(A_p(z) - 1\right) P(z)\right)$$

La sortie du processus est :

$$Y_p(z) = z^{-1} \frac{H(z)}{E(z)} R(z) + A_p(z)\, P(z)$$

La dynamique de poursuite du système est évidemment la dynamique du modèle de poursuite. La dynamique de régulation est donnée par le polynôme A_p : le régime permanent est atteint en n pas.
Par définition du correcteur-S, s'il n'y pas de perturbation, il n'y a pas d'erreur statique. Mais soit une perturbation additive de sortie en échelon :

$$P(z) = \frac{P_0}{1 - z^{-1}}$$

Cette perturbation n'est pas rejetée ; elle provoque une erreur statique :

$$E_0 = P_O \sum_{i=0}^{n} a_{ip}$$

De plus, comme l'indique la fonction de transfert de U par rapport à P, si le polynôme B_p possède des zéros négatifs, en réponse à une perturbation la commande oscille à chaque instant. Par rapport à la consigne R, la commande est filtrée par E, et l'effet des éventuels zéros négatifs de B_p est très atténué par le facteur E.

b) Fonctionnement non nominal ($A_c \neq A_p$, $B_c \neq B_p$).

$$Y_p(z) = \frac{1}{B_p'(z) + \left(A_p(z)\ B_c'(z) - A_c(z)\ B_p'(z)\right)} \left(B_p'(z)\ z^{-1}\ \frac{H(z)}{E(z)}\ R(z) + A_p(z)\ B_c'(z)\ P(z) \right)$$

La stabilité du système de commande est fonction de l'inadéquation du correcteur par rapport au processus. Nous allons comparer l'influence de cette inadéquation sur la stabilité de ce système et sur celle du système de commande du paragraphe II.1.1.2 suivant, pour A_p, B_p, A_c et B_c donnés.

II.1.1.2. Avec un modèle de ralliement.

Le modèle de ralliement s'écrit :

$$y_r(k+1) = \left(1 - E(q)\right) y_p(k+1) + H(q)\ r(k)$$

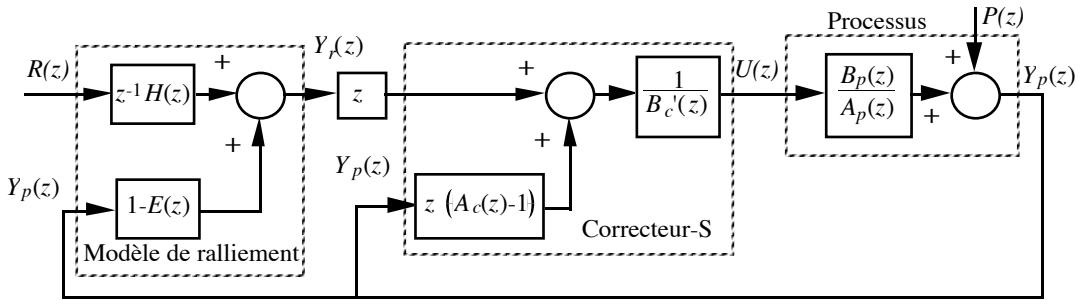


Figure 2.
SCSB avec correcteur-S et modèle de ralliement.

La sortie du correcteur-S de la figure 2 est la suivante :

$$U(z) = \frac{A_p(z)}{E(z)\ B_p'(z) + \left(A_p(z)\ B_c'(z) - A_c(z)\ B_p'(z)\right)} \left(H(z)\ R(z) + z\left(A_c - E(z)\right) P(z) \right)$$

a) Fonctionnement nominal ($A_c = A_p$, $B_c = B_p$) .

$$U(z) = \frac{A_p(z)}{B_p'(z)} \left(\frac{H(z)}{E(z)}\ R(z) + z \left(\frac{A_p(z)}{E(z)} - 1 \right) P(z) \right)$$

La sortie du processus est :

$$Y_p(z) = z^{-1}\ \frac{H(z)}{E(z)}\ R(z) + \frac{A_p(z)}{E(z)}\ P(z)$$

La dynamique de poursuite du système de commande est évidemment encore la dynamique de référence. Grâce au modèle de ralliement, qui possède comme argument la sortie du processus et prend ainsi d'éventuelles perturbations de sortie en considération, la dynamique de régulation est donnée par le polynôme A_p/E : le régime permanent est atteint asymptotiquement. Il existe une erreur statique dont l'amplitude est :

$$E_0 = P_O \frac{\sum_{i=0}^{n} a_{pi}}{\sum_{i=0}^{n} e_i}$$

L'erreur statique est ainsi généralement plus importante que dans le cas précédent[5]. En revanche, la commande est filtrée par E, donc son comportement est satisfaisant en réponse à une perturbation de sortie quel que soit B_p (la commande n'oscille que faiblement si B_p possède des zéros négatifs), ce qui constitue un avantage important de ce système de commande sur le précédent.

b) Fonctionnement non nominal ($A_c \neq A_p$, $B_c \neq B_p$).

$$Y_p(z) = \frac{1}{E(z)B_p'(z) + \left(A_p(z)\ B_c'(z) - A_c(z)\ B_p'(z)\right)} \left(B_p'(z)\ z^{-1}\ H(z)\ R(z) + A_p(z)\ B_c'(z)\ P(z)\right)$$

La stabilité du système de commande est déterminée par les racines du polynôme au dénominateur.

Étude simplifiée de la stabilité.

Nous allons voir que la stabilité de la plupart des systèmes de commande étudiés dépend des racines d'un polynôme D(z) de la forme :

$$D(z) = \varepsilon(z)\ B_1(z) + A_1(z)\ B_2(z) - A_2(z)\ B_1(z)$$

avec, dans le cas du SCSB avec correcteur-S et modèle de poursuite :

$\varepsilon(z) = 1$; $A_1(z) = A_p(z)$; $B_1(z) = B_p'(z)$; $A_2(z) = A_c(z)$; $B_2(z) = B_c'(z)$.

et dans le cas du SCSB avec correcteur-S et modèle de ralliement :

$\varepsilon(z) = E(z)$; $A_1(z) = A_p(z)$; $B_1(z) = B_p'(z)$; $A_2(z) = A_c(z)$; $B_2(z) = B_c'(z)$.

Nous allons montrer sur un exemple que, si $\varepsilon(z)=E(z)\neq 1$, la stabilité du système est plus robuste vis-à-vis d'un écart entre les polynômes A_1 et A_2 et les polynômes B_1 et B_2, que si $\varepsilon(z) = 1$.

Considérons le cas particulier d'un processus du premier ordre, dont le correcteur n'est pas parfait, mais possède un gain statique égal à l'inverse de celui du processus, 1 par exemple :

$$\begin{cases} A_1(q) = 1 + a_1\ q^{-1} \\ B_1(q) = (1 + a_1)\ q^{-1} \end{cases}, \begin{cases} A_2(q) = 1 + a_2\ q^{-1} \\ B_2(q) = (1 + a_2)\ q^{-1} \end{cases}, \varepsilon(q) = 1 + e\ \ q^{-1}$$

où a_1, a_2 et e sont des réels négatifs appartenant à]-1; 0[. La racine du polynôme D(z) s'écrit :

$$\frac{a_2 - a_1 - e\ (1 + a_1)}{1 + a_2} = 1 - \frac{(1 + e\ )(1 + a_1)}{1 + a_2}$$

La valeur de la racine n'est jamais supérieure à 1 ; la condition pour qu'elle soit inférieure à -1 (et donc que le système de commande soit instable) est :

$$2\ \frac{1 + a_2}{1 + a_1} < 1 + e$$

Plus e est petit ($e \leq 0$), moins le système risque d'être instable. En conclusion, pour A_1 et A_2 donnés, le système de commande est d'autant plus stable que E impose une dynamique lente (filtre passe-bas).

---

[5] En effet, la somme des e_i est aussi le produit des racines du polynôme E (monique), toutes de module inférieur à 1 puisque E est stable : cette somme est donc inférieure à 1.

Revenons à la comparaison des SCSB avec modèle de poursuite et modèle de ralliement. Leur stabilité est donc déterminée par D(z), où $A_1=A_p$, $B_1=B_p$', $A_2=A_c$, $B_2=B_c$' dans les deux cas ; mais pour le système avec modèle de poursuite ε(z)=1, donc e=0, alors que pour le système avec modèle de ralliement e<0 : il vaut donc mieux utiliser un modèle de ralliement du point de vue de la stabilité.
Application numérique : $a_p = a_1 =$ - 0,8 ; $a_c = a_2 =$ - 0,95 .
- SCSB avec modèle de poursuite : le système est instable.

$$2\,\frac{1 - 0{,}95}{1 - 0{,}8} = 0{,}5 < 1$$

- SCSB avec modèle de ralliement : le système est stable dès que $e <$ - 0,5, ce qui permet encore d'accélérer le processus de façon appréciable.

En conclusion, si l'on utilise un correcteur-S dans un SCSB, le modèle de référence du système doit être un modèle de ralliement, qui garantit une meilleure stabilité et un meilleur comportement en réponse à des perturbations de sortie qu'un modèle de poursuite.

II.1.2. SCSB utilisant un correcteur-D.

Dans un SCSB, le correcteur-D est mis en cascade avec le processus :

$$B_c'(q)\,u(k) = H(q)\,r(k) + q\left(A_c(q) - E(q)\right) y_p(k)$$

où y_p est la sortie mesurée du processus. Comme le montre la comparaison de la figure 3 avec la figure 2 (correcteur-S avec modèle de ralliement du §II.1.1.2), les deux systèmes de commande sont rigoureusement équivalents.

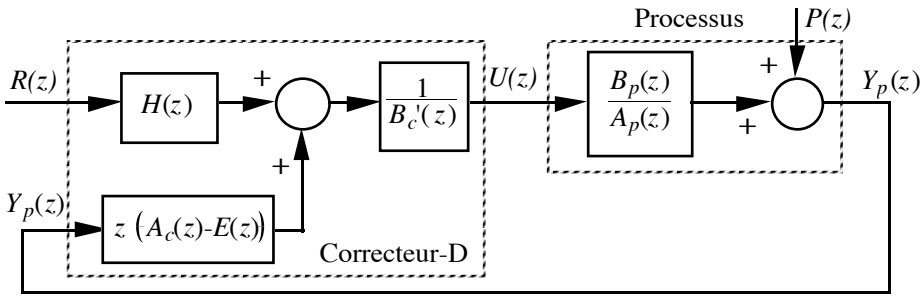


Figure 3.
SCSB avec correcteur-D.

II.2. Systèmes de commande avec modèle interne (SCMI).

Pour les SCMI, nous montrons également l'influence d'un correcteur imparfait ; cette imperfection n'a pas lieu d'être envisagée si les coefficients du correcteur sont calculés à partir de ceux du modèle, mais doit l'être si l'on apprend un réseau correcteur à partir du modèle.

II.2.1. SCMI utilisant un correcteur-S.

Dans un SCMI, le correcteur-S est mis en cascade avec le modèle interne (MI) :

$$B_c'(q)\, u(k) = y_r^*(k+1) + q\left(A_c(q) - 1\right) y(k)$$

où y est la sortie du MI, et y_r^* est la sortie de référence pour le MI. Trois modèles de référence sont *a priori* possibles pour le calcul de la sortie de référence : un *modèle de poursuite à l'extérieur* de la boucle du MI, un *modèle de poursuite à l'intérieur* de celle-ci, ou encore un *modèle de ralliement* pour le MI.

II.2.1.1. Avec un modèle de poursuite extérieur.

L'expression du modèle de poursuite à l'extérieur de la boucle du MI est :

$$E(q)\, y_r(k+1) = H(q)\, r(k)$$

L'entrée de référence du correcteur-S est :

$$y_r^*(k+1) = y_r(k+1) - \left(y_p(k) - y(k)\right)$$

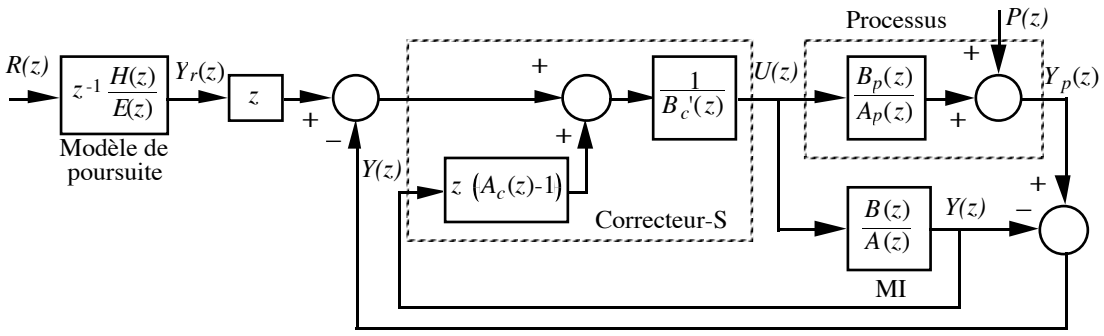


Figure 4.
SCMI avec correcteur-S et modèle de poursuite extérieur.

La commande U(z) de la figure 4 s'écrit, en fonction de la consigne et de la perturbation :

$$U(z) = \frac{A(z)\,A_p(z)}{A_p(z)B'(z) + A_p(z)\left(A(z)B_c'(z) - A_c(z)B'(z)\right) + z^{-1}\left(A(z)B_p'(z) - A_p(z)B'(z)\right)}\left(\frac{H(z)}{E(z)}R(z) - P(z)\right)$$

L'expression ci-dessus fait apparaître au dénominateur l'influence de la désadaptation modèle-processus et de l'imperfection du correcteur.

a) Fonctionnement nominal ($A = A_p$, $B = B_p$) .

Correcteur parfait ($A_c = A$, $B_c = B$) .

$$U(z) = \frac{A_p(z)}{B_p'(z)}\left(\frac{H(z)}{E(z)}R(z) - P(z)\right)$$

Faisons apparaître l'intégrateur que réalise implicitement l'organe de commande en exprimant la perturbation P en fonction de Y_p et de U :

$$U(z) = \frac{1}{1 - z^{-1}}\frac{A_p(z)}{B_p'(z)}\left(\frac{H(z)}{E(z)}R(z) - Y_p(z)\right)$$

Il n'y aura donc pas d'erreur statique. La sortie du processus est :

$$Y_p(z) = z^{-1}\frac{H(z)}{E(z)}R(z) + \left(1 - z^{-1}\right)P(z)$$

La dynamique de régulation est infiniment rapide : en effet, la perturbation est éliminée au bout d'un pas s'il s'agit d'un échelon. La commande oscille fortement à chaque pas d'échantillonnage si le polynôme B_p possède des zéros négatifs.

Correcteur imparfait ($A_c \neq A$, $B_c \neq B$) .

Nous étudions ce cas pour simuler un défaut d'apprentissage du correcteur. La commande est :

$$U(z) = \frac{A_p(z)}{B_p'(z) + \left(A_p(z)\,B_c'(z) - A_c(z)\,B_p'(z)\right)}\left(\frac{H(z)}{E(z)}R(z) - P(z)\right)$$

La stabilité du système de commande dépend du polynôme au dénominateur dont les racines sont étudiées au §II.1.1.2. Dans le cas présent, avec les notations du §II.1.1.2, ε(z)=1 et donc le système de commande risque fort d'être instable si le correcteur n'est pas parfait, et en particulier s'il est adapté à un modèle plus lent (dans l'exemple du §II.1.1.2, un correcteur adapté à un modèle plus lent correspond à $a_2<a_1$).

b) Fonctionnement non nominal ($A \neq A_p$, $B \neq B_p$) ; correcteur parfait ($A_c=A$, $B_c=B$) .

Dans ce cas, la commande s'écrit :

$$U(z) = \frac{A(z)\,A_p(z)}{A_p(z)\,B'(z) + z^{-1}\left(A(z)\,B_p'(z) - A_p(z)\,B'(z)\right)}\left(\frac{H(z)}{E(z)}R(z) - P(z)\right)$$

Même dans le cas non nominal, pourvu que le correcteur soit parfait, l'organe de commande utilisant un MI réalise encore implicitement un intégrateur :

$$U(z) = \frac{1}{1 - z^{-1}}\frac{A(z)}{B'(z)}\left(\frac{H(z)}{E(z)}R(z) - Y_p(z)\right)$$

La performance du système est donc robuste. Le SCSB équivalent est représenté sur la figure 5.

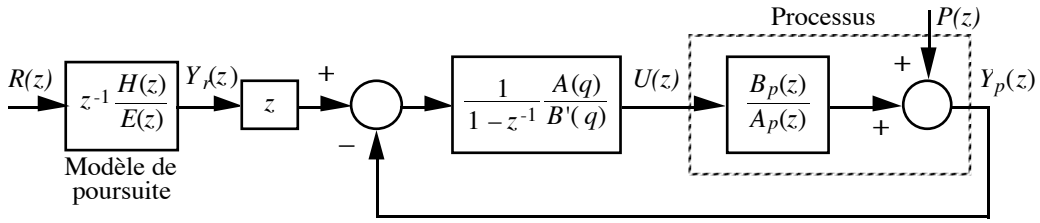


Figure 5.
SCSB équivalent au SCMI avec correcteur-S et modèle de poursuite extérieur
(fonctionnement non nominal, correcteur parfait).

La sortie du processus est :

$$Y_p(z) = \frac{1}{A_p(z)B'(z) + z^{-1}\left(A(z\,)B_p'(z) - A_p(z)B'(z)\right)}\left(z^{-1}A(z)B_p'(z)\frac{H(z)}{E(z)}R(z) + \left(1 - z^{-1}\right)A(z)B_p'(zP(z)\right)$$

On vérifie ainsi qu'il n'y a pas d'erreur statique pour une perturbation de sortie constante.

II.2.1.2. Avec un modèle de poursuite intérieur.

L'expression du modèle de poursuite à l'intérieur de la boucle du MI est :

$$E(q)\, y_r^*(k+1) = H(q)\, r^*(k)$$

où :

$$r^*(k) = r(k) - \left(y_p(k) - y(k)\right)$$

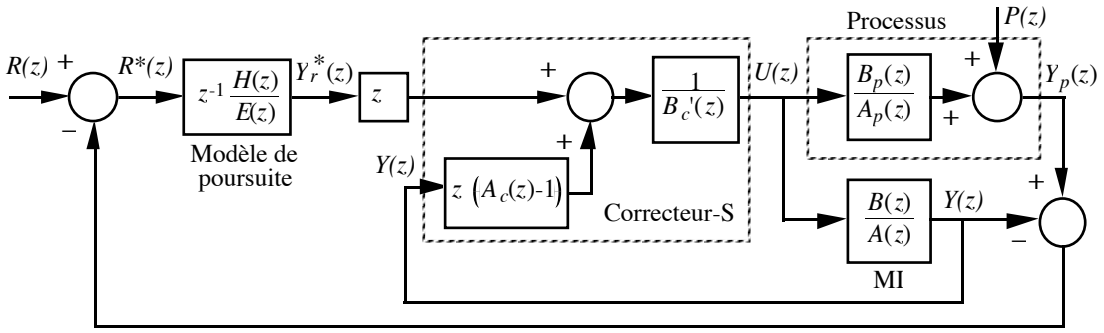


Figure 6.
SCMI avec correcteur-S et modèle de poursuite intérieur.

La sortie du correcteur-S de la figure 6 est la suivante :

$$U(z) = \frac{A(z)\,A_p(z)\,H(z)}{E(z)A_p(z)\left(B'(z) + \left(A(z)B_c'(z) - A_c(z)B'(z)\right)\right) + z^{-1}H(z)\left(A(z)B_p'(z) - A_p(z)B'(z)\right)}\left(R(z) - P(z)\right)$$

a) Fonctionnement nominal (A= A_p, B= B_p) .

Correcteur parfait (A_c=A, B_c=B) .

$$U(z) = \frac{A_p(z)}{B_p'(z)}\,\frac{H(z)}{E(z)}\left(R(z) - P(z)\right)$$

La dynamique de régulation est identique à la dynamique de poursuite, et donc à la dynamique de référence. Le dénominateur commun de la commande est E B_p, alors que dans le cas du SCMI avec correcteur-S et modèle de poursuite extérieur (§II.2.1.1), le dénominateur de la fonction de transfert par rapport à R est E B_p, mais le dénominateur de la fonction de transfert par rapport à P est B_p seulement. Si le polynôme B_p possède des zéros négatifs, l'effet en sera atténué pour le présent système.

Exprimons la commande en fonction de l'erreur :

$$U(z) = \frac{H(z)}{E(z) - z^{-1}\,H(z)}\,\frac{A_p(z)}{B_p'(z)}\left(R(z) - Y_p(z)\right)$$

Le premier facteur du dénominateur s'écrit :

$$E(z) - z^{-1}\,H(z) = \sum_{i=0}^{p} e_i\, z^{-i} - \sum_{i=0}^{p} h_i\, z^{-(i+1)}$$

Or le modèle de référence étant choisi de gain statique unité, on a l'égalité :

$$\sum_{i=0}^{p} e_i = \sum_{i=0}^{p} h_i$$

1 est donc racine du polynôme, ce qui nous permet d'écrire le polynôme sous la forme :

$$E(z) - z^{-1} H(z) = \left(1 - z^{-1}\right) Q(z)$$

où Q(z) est un polynôme monique de degré p-1. L'organe de commande réalise encore implicitement un intégrateur :

$$U(z) = \frac{1}{1 - z^{-1}} \frac{H(z)}{Q(z)} \frac{A_p(z)}{B_p'(z)} \left(R(z) - Y_p(z)\right)$$

La sortie du processus est :

$$Y_p(z) = z^{-1} \frac{H(z)}{E(z)} R(z) + \left(1 - z^{-1} \frac{H(z)}{E(z)}\right) P(z)$$

Correcteur imparfait ($A_c \neq A$, $B_c \neq B$) .

Cette éventualité est envisagée pour simuler un défaut d'apprentissage du correcteur. L'expression de la commande est la suivante :

$$U(z) = \frac{A_p(z)}{B_p'(z) + \left(A_p(z)\, B_c'(z) - A_c(z)\, B_p'(z)\right)} \frac{H(z)}{E(z)} \left(R(z) - P(z)\right)$$

Le polynôme apparaissant au dénominateur est le même que pour le système de commande précédent : la stabilité du présent système de commande n'est donc pas non plus robuste vis-à-vis d'une imperfection du correcteur.

b) Fonctionnement non nominal ($A \neq A_p$, $B \neq B_p$) ; correcteur parfait ($A_c = A$, $B_c = B$) .

La commande s'écrit :

$$U(z) = \frac{A(z)\, A_p(z)\, H(z)}{E(z)\, A_p(z)\, B'(z) + z^{-1}\, H(z) \left(A(z)\, B_p'(z) - A_p(z)\, B'(z)\right)} \left(R(z) - P(z)\right)$$

Pourvu que le correcteur soit adapté au modèle, l'organe de commande comprenant un MI réalise encore implicitement un intégrateur, comme le montre la figure 7 :

$$U(z) = \frac{1}{1 - z^{-1}} \frac{H(z)}{Q(z)} \frac{A(z)}{B'(z)} \left(R(z) - Y_p(z)\right)$$

La performance du système est donc robuste.

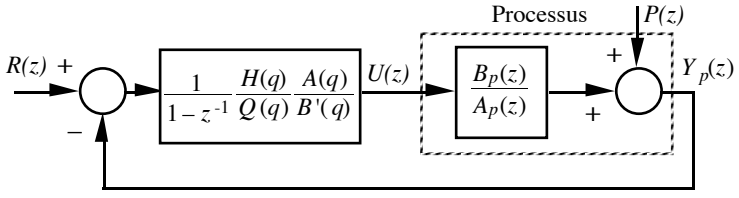


Figure 7.
SCSB équivalent au SCMI avec correcteur-S et modèle de poursuite intérieur
(fonctionnement non nominal, correcteur parfait).

La sortie du processus est :

$$Y_p(z) = \frac{\left(A(z)B_p'(z)z^{-1}H(z)\; R(z) + \left(1 - z^{-1}\right)A_p(z)B'(z)Q(z)\; P(z)\right)}{E(z)A_p(z)B'(z) + z^{-1}\; H(z) \left(A(z)B_p'(z) - A_p(z)B'(z)\right)}$$

On vérifie qu'il n'y a pas d'erreur statique pour une perturbation de sortie constante.

II.2.1.3. Avec un modèle de ralliement.

Le modèle de ralliement rallie la trajectoire du modèle à celle de la consigne décalée. L'expression du modèle de ralliement est :

$$y_r^*(k+1) = \big(1 - E(q)\big)\, y(k+1) + H(q)\, r^*(k)$$

où :

$$r^*(k) = r(k) - \big(y_p(k) - y(k)\big)$$

La sortie du correcteur-S de la figure 8 est la suivante :

$$U(z) = \frac{A(z)\, A_p(z)\, H(z)}{E(z)A_p(z)B'(z) + A_p(z)\big(A(z)B_c'(z) - A_c(z)B'(z)\big) + z^{-1}H(z)\big(A(z)B_p'(z) - A_p(z)B'(z)\big)} \big(R(z) - P(z)\big)$$

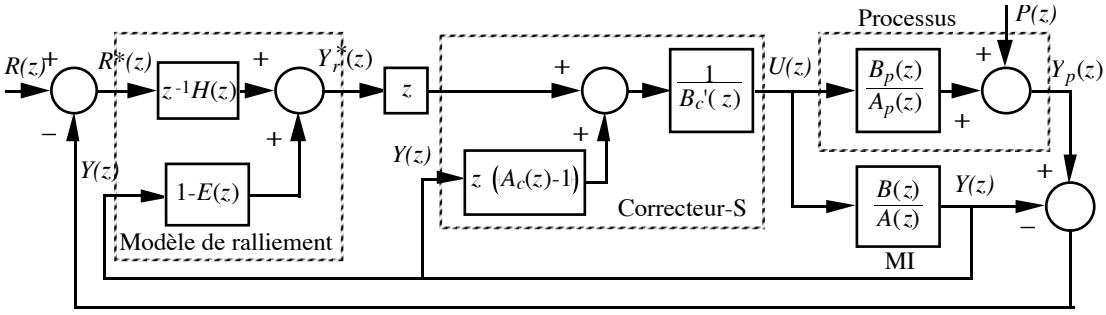


Figure 8
SCMI avec correcteur-S et modèle de ralliement.

a) Fonctionnement nominal ($A = A_p$, $B = B_p$) .

Correcteur parfait ($A_c = A$, $B_c = B$) .

Le système est identique au SCMI avec correcteur-S et modèle de poursuite intérieur (§II.2.1.2).

Correcteur imparfait ($A_c \neq A$, $B_c \neq B$) .

L'expression de la commande est la suivante :

$$U(z) = \frac{A_p(z)\, H(z)}{E(z)\, B_p'(z) + \big(A_p(z)\, B_c'(z) - A_c(z)\, B_p'(z)\big)} \big(R(z) - P(z)\big)$$

Le polynôme apparaissant au dénominateur fait intervenir E(z) (soit $\varepsilon(z) \neq 1$ avec les notations du §II.1.1.2) : la stabilité du présent système de commande est donc plus robuste vis-à-vis d'une imperfection du correcteur, que celle des deux systèmes précédents (où $\varepsilon(z)=1$).

b) Fonctionnement non nominal ($A \neq A_p$, $B \neq B_p$); correcteur parfait ($A_c = A$, $B_c = B$) .

Le fonctionnement du système est identique à celui du SCMI avec correcteur-S et modèle de poursuite intérieur (§II.2.1.2). Ses propriétés de performance robuste sont les mêmes.

II.2.2. SCMI utilisant un correcteur-D.

L'entrée de consigne du correcteur-D est :

$$r^*(k) = r(k) - \big(y_p(k) - y(k)\big)$$

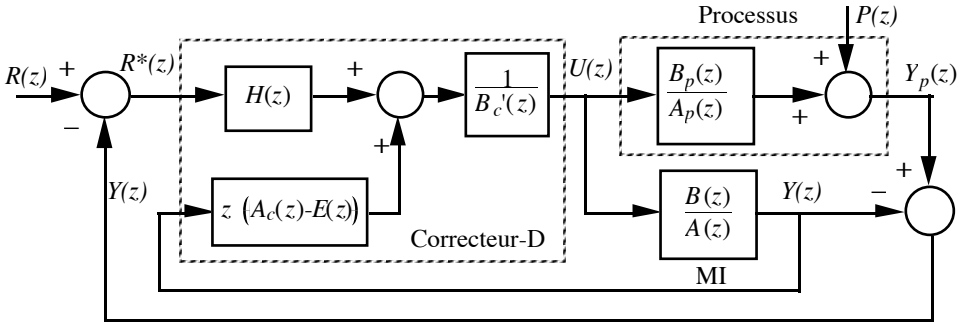


Figure 9.
SCMI avec correcteur-D.

La sortie du correcteur-D du schéma de la figure 9 est la suivante :

$$U(z) = \frac{A(z)\,A_p(z)\,H(z)}{E(z\,)A_p(z)B'(z) + A_p(z)\big(A(z)B_c'(z) - A_c(z)B'(z)\big) + z^{-1}H(z)\big(A_p(z)B'(z) - A(z)B_p'(z)\big)}\big(R(z) - P(z)\big)$$

a) Fonctionnement nominal (A= A_p, B= B_p).

Correcteur parfait (A_c=A, B_c=B) .

$$U(z) = \frac{A_p(z)}{B_p'(z)}\,\frac{H(z)}{E(z)}\big(R(z) - P(z)\big)$$

En faisant apparaître l'intégrateur implicite :

$$U(z) = \frac{1}{1 - z^{-1}}\,\frac{H(z)}{Q(z)}\,\frac{A_p(z)}{B_p'(z)}\big(R(z) - Y_p(z)\big)$$

La sortie du processus est :

$$Y_p(z) = z^{-1}\,\frac{H(z)}{E(z)}\,R(z) + \left(1 - z^{-1}\frac{H(z)}{E(z)}\right)P(z)$$

La dynamique de régulation est la même que la dynamique de poursuite : elle donnée par le polynôme de référence 1/E. Il n'y a pas d'erreur statique pour une perturbation de sortie constante puisque le modèle de référence est de gain unité. Le comportement de la commande est toujours satisfaisant quels que soient les zéros du processus (la commande est filtrée par E).

Correcteur imparfait (A_c≠A, B_c≠B) .

L'expression de la commande est la suivante :

$$U(z) = \frac{A_p(z)\,H(z)}{E(z)\,B_p'(z) + \big(A_p(z)\,B_c'(z) - A_c(z)\,B_p'(z)\big)}\big(R(z) - P(z)\big)$$

Dans ce cas, le système de commande est équivalent au système avec correcteur-S et modèle de ralliement. Il est donc aussi relativement robuste vis-à-vis d'une imperfection du correcteur.

b) Fonctionnement non nominal (A≠ A_p, B≠ B_p) ; correcteur parfait (A_c=A, B_c=B) .

Dans ce cas, la commande s'écrit :

$$U(z) = \frac{A(z)\,A_p(z)\,H(z)}{E(z)\,A_p(z)\,B'(z) + z^{-1}\,H(z)\,\big(A(z)B_p'(z) - A_p(z)\,B'(z)\big)}\big(R(z) - P(z)\big)$$

C'est la même expression que dans le cas précédent : on retrouve les propriétés de la commande avec correcteur-S et modèle de ralliement.

CONCLUSION.

Presque tous les systèmes de commande étudiés dans cette annexe sont proposés dans la littérature, mais rarement comparés objectivement. L'intérêt de ce travail est : d'une part, de les comparer sur la base de critères qui sont stabilité et performance du système nominal, et robustesse de cette stabilité et de cette performance ; d'autre part, d'indiquer lesquels de ces systèmes doivent être évités, et lesquels doivent être utilisés pour une commande neuronale. Récapitulons les conclusions de cette étude :

- Les systèmes de commande par simple bouclage (SCSB) ou avec modèle interne (SCMI) utilisant un correcteur-S et un modèle de poursuite extérieur à la boucle de retour d'état et, le cas échéant, extérieur à la boucle du modèle interne, ont la caractéristique d'une dynamique de régulation non maîtrisée, et d'une stabilité peu robuste. Les systèmes des paragraphes II.1.1.1 et II.2.1.1 utilisant un tel modèle de poursuite doivent donc être évités.
- Les SCMI présentent l'avantage sur les SCSB d'éliminer les erreurs statiques dues à toute perturbation de sortie en fonctionnement nominal, et de présenter une grande robustesse du niveau de performance vis-à-vis d'écarts par rapport au système nominal (erreur statique nulle pour le système non nominal).
- Un SCMI utilisant un correcteur-S doit comprendre un modèle de ralliement (et non un modèle de poursuite intérieur à la boucle du modèle interne), qui rend la stabilité du système plus robuste vis-à-vis d'une éventuelle imperfection du correcteur.
- Les SCMI utilisant un correcteur-D, ou un correcteur-S et un modèle de ralliement, ont des propriétés voisines. Cependant, le second doit être préféré si l'on utilise des réseaux de neurones, car changer de dynamique de poursuite si nécessaire (par exemple pour stabiliser le processus) ne demande pas de nouvel apprentissage, mais seulement de modifier le modèle de ralliement.

5th International Conference on Neural Networks and their Applications (NeuroNîmes'93)

REAL-TIME CONTROL OF AN AUTONOMOUS VEHICLE : A NEURAL NETWORK APPROACH TO THE PATH FOLLOWING PROBLEM

Isabelle Rivals*, Léon Personnaz**, Gérard Dreyfus** and Daniel Canas*.

* SAGEM Eragny, Unité R&D, Avenue du Gros Chêne, 95 610 Eragny, France.
Phone : 33 1 34 30 51 19 ; Fax : 33 1 34 30 50 96.

** ESPCI, Laboratoire d'Électronique, 10 rue Vauquelin, 75 005 Paris, France.
Phone : 33 1 40 79 45 45 ; Fax : 33 1 40 79 44 25.

Abstract : A neural-network based approach to the control of non-linear dynamical systems such as wheeled mobile robots is presented. A general framework for the training of neural controllers is outlined, and applied to the lateral control of a vehicle for the path following and trajectory servoing problems. Simulation as well as experimental results on a four-wheel drive vehicle equipped with actuators and sensors are shown.

Key-words : autonomously guided vehicles (AGVs), mobile robots, model reference control, non-linear control, optimal control, path following, recurrent neural networks, trajectory servoing.

Résumé : Nous décrivons une approche neuronale de la commande de processus dynamiques non-linéaires tels que les robots mobiles à roues. Nous esquissons les grands traits d'un cadre général pour l'apprentissage de correcteurs neuronaux, et appliquons nos méthodes à la commande de la direction d'un véhicule pour son asservissement sur trajectoire. Nous illustrons notre propos à l'aide de simulations, et présentons les résultats expérimentaux obtenus sur un véhicule tout-terrain équipé des capteurs de navigation et des actionneurs nécessaires au pilotage.

Mots-Clés : asservissement sur trajectoire, commande avec modèle de référence, commande non-linéaire, commande optimale, robots mobiles, réseaux neuronaux bouclés, suivi de trajectoire, véhicules autonomes.

1. INTRODUCTION

We address the lateral control of an autonomous vehicle along a predefined trajectory using neural networks. Experimental results on a full-scale outdoor robot, a standard four-wheel drive car equipped with the sensors and actuators needed for navigation and control, demonstrate that neural techniques can be applied to real-world problems in robotics.

Classical control theory provides many design techniques to achieve performances specified in terms of rise and settling-time, gain and phase margin, bandwidth… These methodologies are well suited to the design of linear controllers for linear systems, with guaranteed stability and robustness. They are, however, of restricted use for control problems involving non-linear dynamic processes with inequality constraints on state and control variables, such as wheeled mobile robots with actuator limitations.

Optimal control theory has been widely used to solve such non-linear, constrained problems. But the conventional scheme of optimal control has its own drawbacks. Finding the control trajectory that minimizes the performance measure often requires the solution of non-linear differential equations. This can be achieved by using iterative numerical methods (quasi-linearization, steepest descent…) which are time consuming, or by dynamic programming ; both miss closed-form expressions for the feedback control laws.

Neural networks offer an alternative to the usual formulations and solutions of constrained optimization problems. Their approximation capabilities make them of possible use as models of the process to be controlled, as well as suitable controllers parameterizing non-linear optimal feedback control laws. In addition, the performance measure which, when minimized, corresponds to the optimal behaviour, can be defined with respect to a reference model. Finally, generic algorithms using a gradient-based approach to achieve the minimization of the performance measure - regardless of model and controller complexity - have been established.

In the second part of this paper, we present a general framework for the training of neural networks for control purposes. Part 3 is devoted to our application : the lateral control of a vehicle for the path following problem. We first present our test-vehicle and its neural model. We subsequently apply the control scheme developed in part 2 using two different approaches of the path following problem. Simulation and experimental results are shown in both cases. They are discussed in part 4.

2. TRAINING OF RECURRENT NEURAL NETWORKS FOR NON-LINEAR CONTROL

We assume that the process to be controlled is described by the following discrete-time model :

$$\begin{cases} S(k+1) = f(S(k), U(k)) \\ Y(k) = g(S(k)) \end{cases}$$

where $S(k)$, $Y(k)$ and $U(k)$ are the state, output and control vectors at time k respectively, and f and g are unknown non-linear functions.

The control system consists of the following components :

- a *neural network predictor model* with state S_m and output Y_m, is first trained :

$$\begin{cases} S_m(k+1) = f_{NN}(X_m(k), U(k)) \\ Y_m(k) = g_{NN}(S_m(k)) \end{cases}$$

where f_{NN} and g_{NN} are non-linear functions implemented by neural networks, and where the state input X_m may take different values depending on the particular predictor choice. For a recursive predictor, $X_m(k) = S_m(k)$; for a non-recursive predictor, $X_m(k)$ is often taken to be equal to the process state $X_p(k)$. The rationale of this choice has been discussed in [NER92a].

- a *reference model* (possibly a simple delay) is designed, which generates the desired output sequence $\{Y_r(k)\}$ from the setpoint sequence $\{R(k)\}$:

$$\begin{cases} S_r(k+1) = f_r(S_r(k),R(k)) \\ Y_r(k) = g_r(S_r(k)) \end{cases}$$

where $S_r(k)$ is the state of the reference model at time k, $Y_r(k)$ its output, $R(k)$ the setpoint vector, and f_r and g_r are known (possibly linear) functions.

- the *neural controller*, with weight matrix C, computes the control sequence $\{U(k)\}$ from the setpoint sequence and the model state input, and can therefore implement any suitable non-linear state-feedback control law $U(k) = \psi(X_m(k), R(k))$.

The aim of the training is to compute the weights of the neural controller, either adaptatively or non adaptively, so that the output of the process becomes as close as possible to the output of the reference model. We restrict the scope of our presentation to a non-adaptive context in which the process itself is not taken into account during the training of the controller (for a general presentation, including adaptive control schemes, see [NER 93a]). This approach of neural control is well suited to the off-line validation of controller structures, provided the neural model of the process is accurate enough, and is often encountered in the literature (e. g. [NAR 91]).

2.1 Training phase

The training algorithms aim at minimizing a cost-function J by a gradient-based technique. J involves the squared difference E_m between the output of the model and the output of the reference model over a time horizon of length N_c :

$$J = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} \|E_m(k)\|^2 = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} [Y_r(k) - Y_m(k)]^T W [Y_r(k) - Y_m(k)]$$

where W is a weighting matrix, $N_c \leq N_t$ (for example $N_c=1$ is chosen if a desired value exists for the final output only) and N_t is the number of time steps used for the evaluation of the gradient of the cost-function ([NER92b][NER93b]).
The weights will be modified iteratively in the direction opposite to that of the gradient :

$$\Delta Cij = - \mu \frac{\partial J}{\partial Cij} = - \mu \frac{\partial}{\partial Cij} (\frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} \|E_m(k)\|^2)$$

where μ is the gradient step.
The value of the cost-function J, hence of its gradient, depends on the NN predictor model used for the computation of Y_m. The NN predictor model is either recursive or non-recursive, each case leading to a specific training algorithm.

a) The UD control algorithm (" UnDirected " algorithm, see figure 1)

If the NN predictor model is *recursive*, the state input $X_m(k)$ takes the values :
$X_m(0)$ *arbitrary, and* $X_m(k) = S_m(k)$ $\forall$ $k>0$.

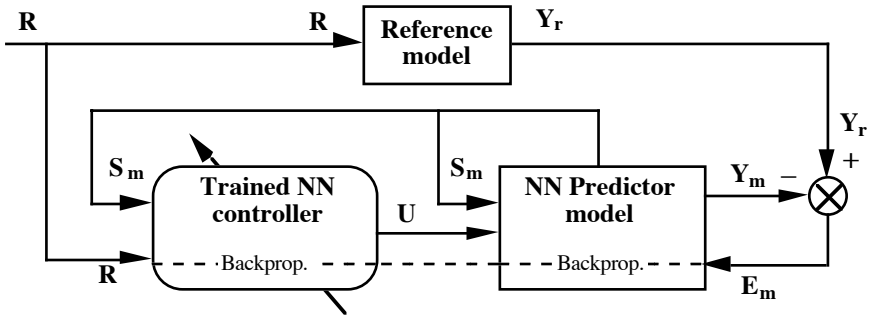


FIGURE 1
Controller training architecture associated to an UD control algorithm.

At $k=0$ only, the state inputs of the model are initialized to arbitrary values. The controller and the predictor build up a *recurrent network*, and the computation of the gradient - with respect to the controller weights - is achieved using *dynamic back-propagation* over the time-horizon N_t ([NER92b], [NAR90], [NAR91]).

b) The D control algorithm (" Directed " algorithm, see figure 2)

If the NN predictor model is *non-recursive*, the state input $X_m(k)$ takes the values :
$X_m(k) = S_r(k)$ $\forall$ k.

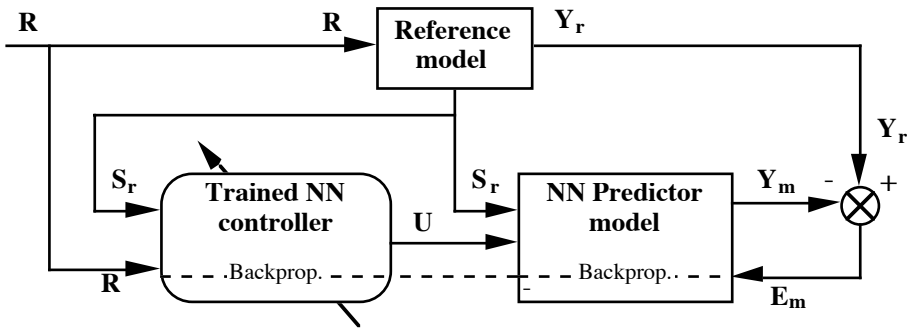


FIGURE 2
Controller training architecture associated to a D control algorithm.

Thus, at every time step, the state inputs of the model are taken equal to the values of the state variables of the reference model. The controller and the predictor now build up a *feedforward*

network, and the computation of the gradient can be achieved using *static back-propagation*. For this algorithm to be applicable, it is necessary that the reference model specifies the whole state (not only the desired output).

2.2 Operating phase

During the operating phase, which follows the training phase, the weights are frozen and the controller is used as shown in figure 3. The reference model being *implicit* (that is, its output is not fed to the neural controller during training [LAN79]), it is not part of the operating control system itself.

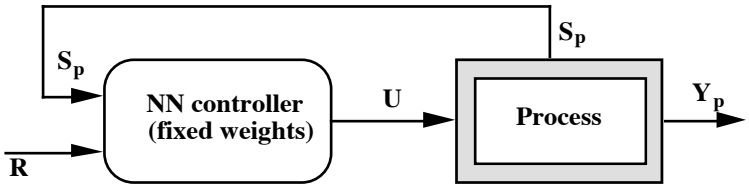


FIGURE 3
Control system during the operating phase.

3. APPLICATION TO THE LATERAL CONTROL OF AN AUTONOMOUS VEHICLE

3.1 The vehicle REMI and its model

Our testbed is the SAGEM autonomous navigation test vehicle REMI (Robot Evaluator for Mobile Investigations), a four-wheel drive vehicle equipped with actuators and sensors. Our aim being here the lateral control, we are concerned with its steering system only. The steering actuator monitors the angular position of the steering wheel, and the velocity is controlled by a human operator using the brake and throttle pedals. Thanks to an inertial dead-reckoning unit and an odometric sensor, a navigation module computes position, orientation and velocity of the vehicle. The navigation and piloting modules are implemented on a 68030 board running under the operating system OS9. The interfaces with the hardware are achieved with specific boards. More details about REMI can be found in [VDB93][FRA93]. As pointed out in part 2, an accurate model of the vehicle is required for the training, since the controller is non-adaptive. Thus, both the kinematics of the vehicle and its dynamics were identified, using two neural networks.

For the kinematic part, the classically adopted " bicycle " model [JUR93][SIN90] proving to be unsatisfactory, we performed a (non-adaptive) identification of the non-linear relationship (3) - given below - between the heading ψ of the vehicle and the position β of the steering wheel (Figure 4). We used a model based on the bicycle model, including a neural network which gives a better estimate of the geometrical non-linearity (ϕ_{NN}) than the *tan(.)* relationship of the bicycle model ($\psi(k+1) = \psi(k) + v(k)\ (\Delta T/L)\ \tan(\beta(k))$).

The vehicle dynamics can be separated into *actuator dynamics* and v*ehicle-ground interactions* . The latter are due to the flexing and slipping in the contact between the vehicle and the ground, and are negligible at low speed (*<4 m/s*). But we assume that, for a given trajectory, the upper limit of speed for which the vehicle will not slip will not be reached. Moreover, we consider that the ground is flat (nevertheless, we achieved good performances on uneven ground). Thus, the dynamics reduces to the steering actuator dynamics. We identified the relationship between steering command α and steering wheel position β with a second neural net consisting of two neurons with linear saturated activation function, neglecting other dynamical effects than a dead-time and the angle and velocity saturations of the actuator. Measurement noise being dominant, both identifications were performed with recursive predictors [NER93b].

The overall model, with state $S(k) = [x(k), y(k), \psi(k), \beta(k), v(k)]$ is described by the following state equations :

(1) $x(k+1) = x(k) + v(k)\ \Delta T \cos \psi(k)$

(2) $y(k+1) = y(k) + v(k)\ \Delta T \sin \psi(k)$

(3) $\psi(k+1) = \psi(k) + \frac{v(k)\ \Delta T}{L}\ \phi_{NN}(\beta(k))$

(4) $\beta(k+1) = sat_{\beta max}\ (\ \beta(k) + sat_{\Delta\beta max}\ (\ \alpha(k-N) - \beta(k)\ )\ )$

(5) $v(k+1) = v(k) + f_u(S(k), \Theta(k), B(k))$

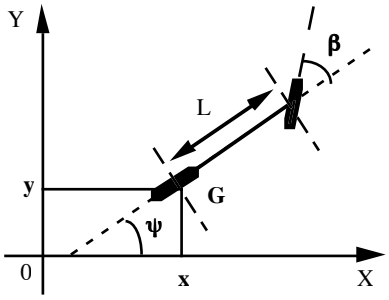
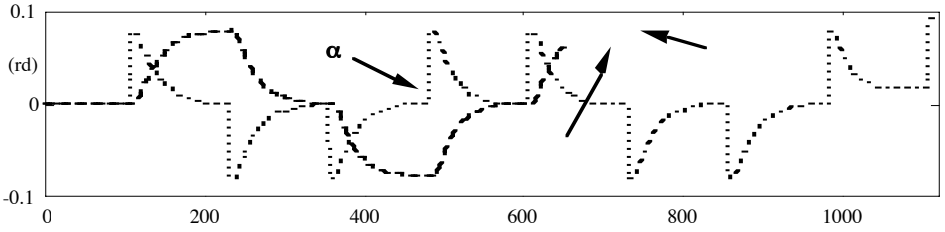
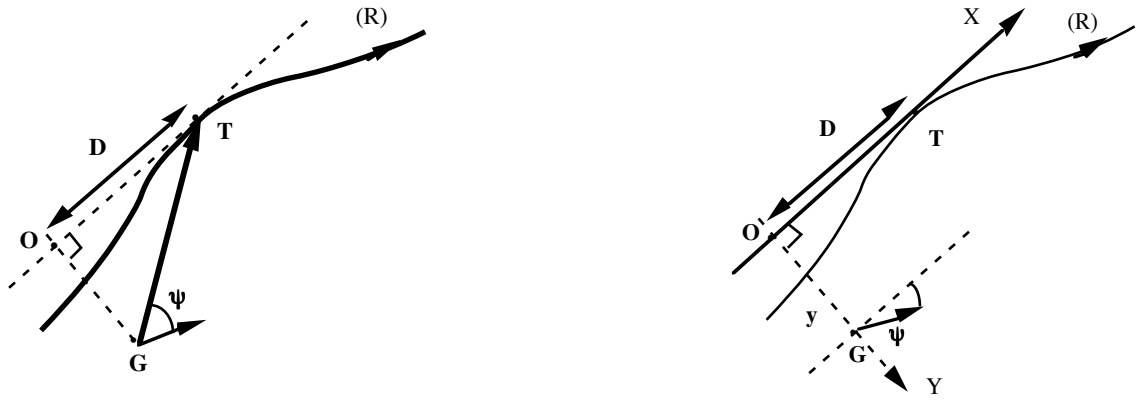


FIGURE 4
The variables of the model.

where :
- x, y are the coordinates of the guide-point G located at the center of the rear axle (for the justification of this choice, see [FRA93][SIN90]) in (X,Y) ;
- ψ is the heading of the vehicle ;
- β is the steering wheel angle ;
- α is the steering command ;
- v is the vehicle velocity measured at the guide-point G ;
- $L=2.85\ m$ is the distance between axles ;
- $sat_X(.)$ is the saturation function between $-X$ and $+X$; β_{max} and $\Delta\beta_{max}$ are the values of the angle and velocity saturations, identified at respectively $0.5rd$ and $0.2\ rd/s$ by the neural network representing the dynamics of the vehicle ;
- $\Delta T=0.04\ s$ is the sampling period ;
- N is the actuator dead-time, experimentally identified as being ~ 4 ($4\ \Delta T = 160\ ms$) ;
- ϕ_{NN} is the non-linear function implemented by the neural network representing t
he kinematics of the vehicle.
- the state variable v is governed by a relationship involving the control inputs Θ (throttle) and B (brakes) characterized by the unknown function f_u (its knowledge is not necessary for the lateral control of the vehicle).



orientation error, and y the transversal error. Part 3.2.2 is devoted to this approach, termed " posture-based " approach.



a) Heading-based approach. **b) Posture-based approach.**

FIGURE 6

Target-point and relative coordinates definition during the operating phase.
(T = target-point ; O = orthogonal projection on the tangent to (R) through P ; G = guide-point of the vehicle)

3.2.1 The heading-based approach

a) Training phase and simulation results

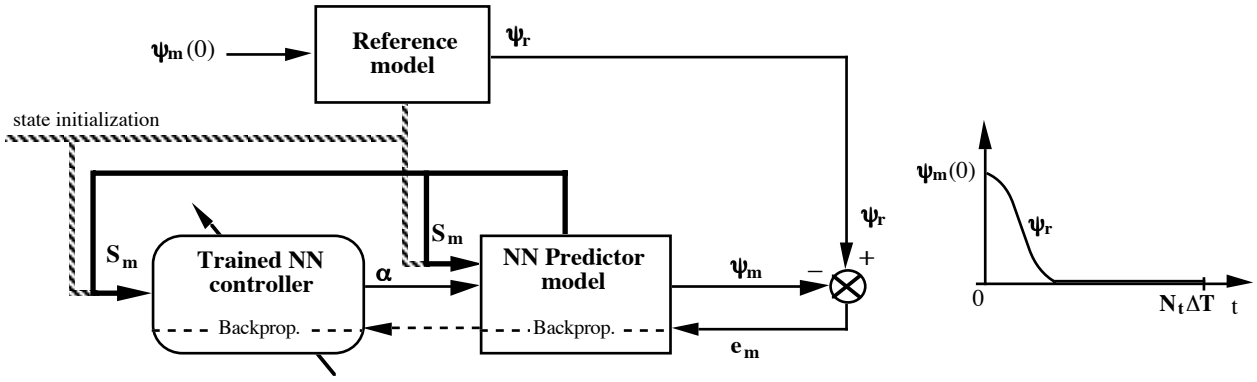
The neural controller task is to make the model output ψ_m as close as possible to the output of a reference model ψ_r which defines the desired way of bringing the heading to zero. Feedback is thus required only from the partial state :

$$S_m(k) = [\psi_m(k), \beta_m(k), v_m(k)]$$

The cost-function to minimize over a trajectory of time-length N_t involves only the heading ψ_m :

$$J = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} (\psi_r(k) - \psi_m(k))^2 = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} e_m^2(k)$$

The reference output ψ_r is defined as the output of the linearized model ($\psi_r(k+1) = \psi_r(k) + v(k)\ (\Delta T/L)\ \alpha_r(k-N)$) controlled by a minimum-time controller with inequality constraints on α_r and $\Delta\alpha_r$ (corresponding to the angle and velocity saturations β_{max} and $\Delta\beta_{max}$ of the actuator). The parameters of the reference controller are computed for various initial conditions, with constant speed on each trajectory, in which case the computation is very simple. Since the reference model specifies only the desired output (and not the whole state), the training has to be performed with an UD algorithm. We choose $N_c=N_t$, N_t being sufficiently long for the reference model to reach $\psi=0$ (figure 7b). The initial conditions for the vehicle model and reference model are taken in the range $\psi_r(0)=\psi_m(0) \in [-\pi/2; +\pi/2]$ *rd* and $v \in [0;10]$ *m/s.*



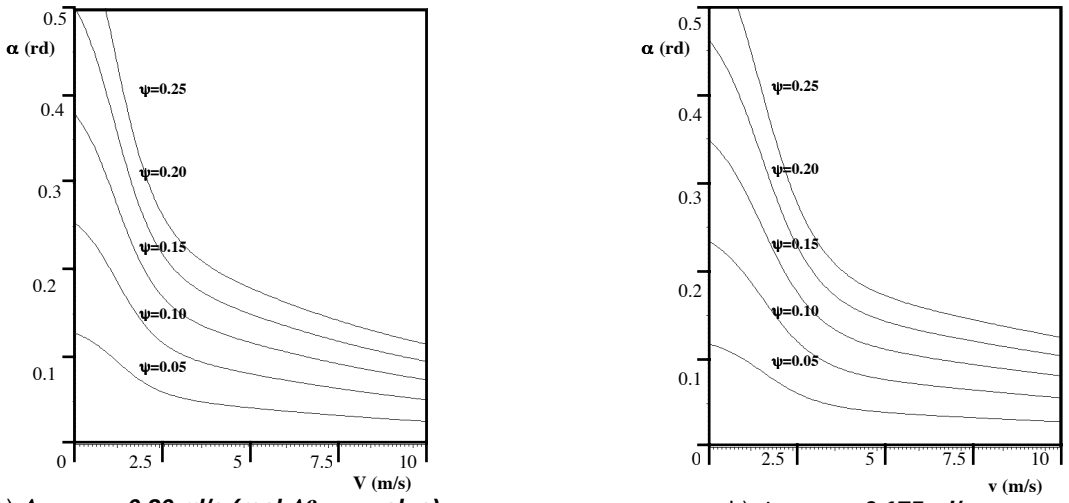
a) Controller training architecture using an UD algorithm. **b) Reference heading.**

FIGURE 7

Heading-based approach : training of the steering controller with minimum-time control reference model.

To avoid the drawbacks of a bang-bang type controller, we used reference models with a lower constraint on $\Delta\alpha_r$ than the real value of $\Delta\beta_{max}$ ($\Delta\beta_{max}$ = *0.20 rd/s*). The best trade-off

beetwen performance and driving comfort was determined experimentally on REMI, leading to the choice of $\Delta\alpha_{max}$ = *0.175 rd/s*. Figure 8 shows the output α of the neural controller as a function of *v* and ψ , for these two values of $\Delta\alpha_{max}$.



a) $\Delta\alpha_{max}$ = *0.20 rd/s (real* $\Delta\beta_{max}$ *value)*

b) $\Delta\alpha_{max}$ = *0.175 rd/s*

FIGURE 8

Output of the neural network controller trained with two different reference models.

The neural controller is a MLP (2,3,3,1), with inputs ψ, *v* and output α (feeding β to the network did not improve the performance).

b) Operating phase and experimental results

The distance *D* to the target-point is chosen to depend linearly on vehicle velocity :

$$D = d_0 + F_v \, . \, v \, .$$

The values of the parameters d_0 and F_v were determined by computer simulations.

The controller has been used successfully on various trajectories. Figure 10a shows the experimental path following performance obtained along the trajectory of figure 9. The transversal error e_t (distance to the closest point of the reference trajectory) is kept very small (< *60 cm*, the curvature reaching *0.1* m^{-1}) with speeds up to *25 km/h*. The heading error e_ψ (defined with respect to the tangent to (R) at the closest point of the reference trajectory) is kept smaller than *0.1 rd*. Figure 10b shows the behaviour of the vehicle for the trajectory servoing problem, the vehicle being initialized far away from the reference trajectory.

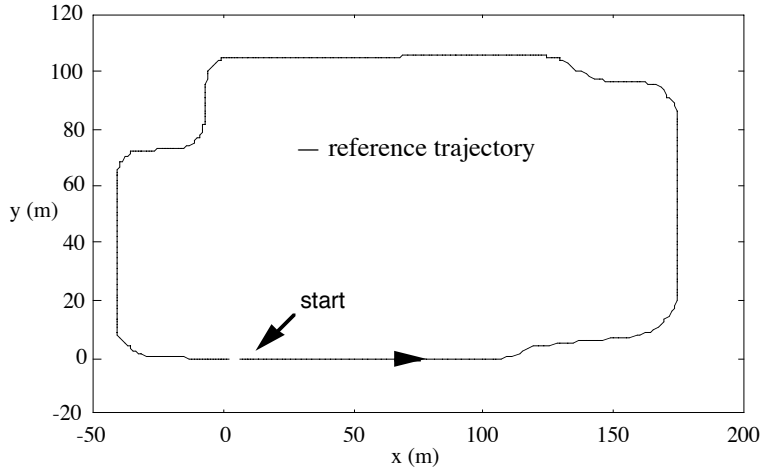
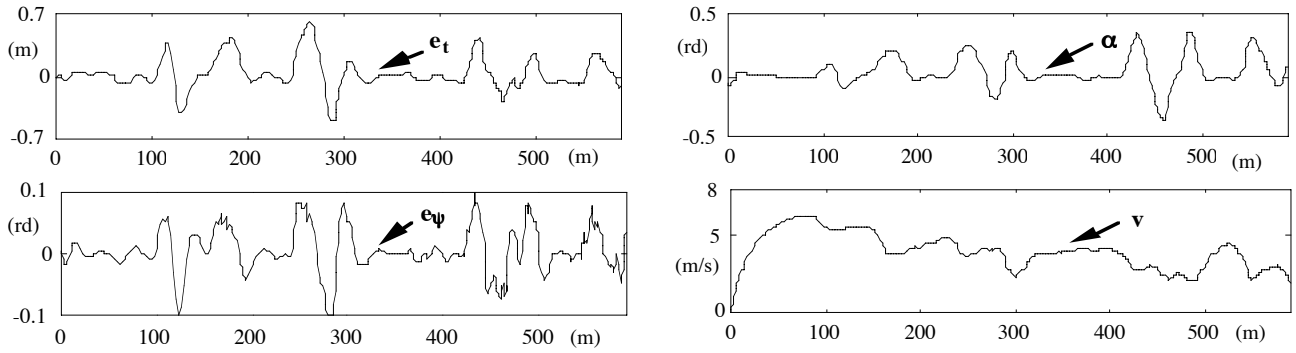
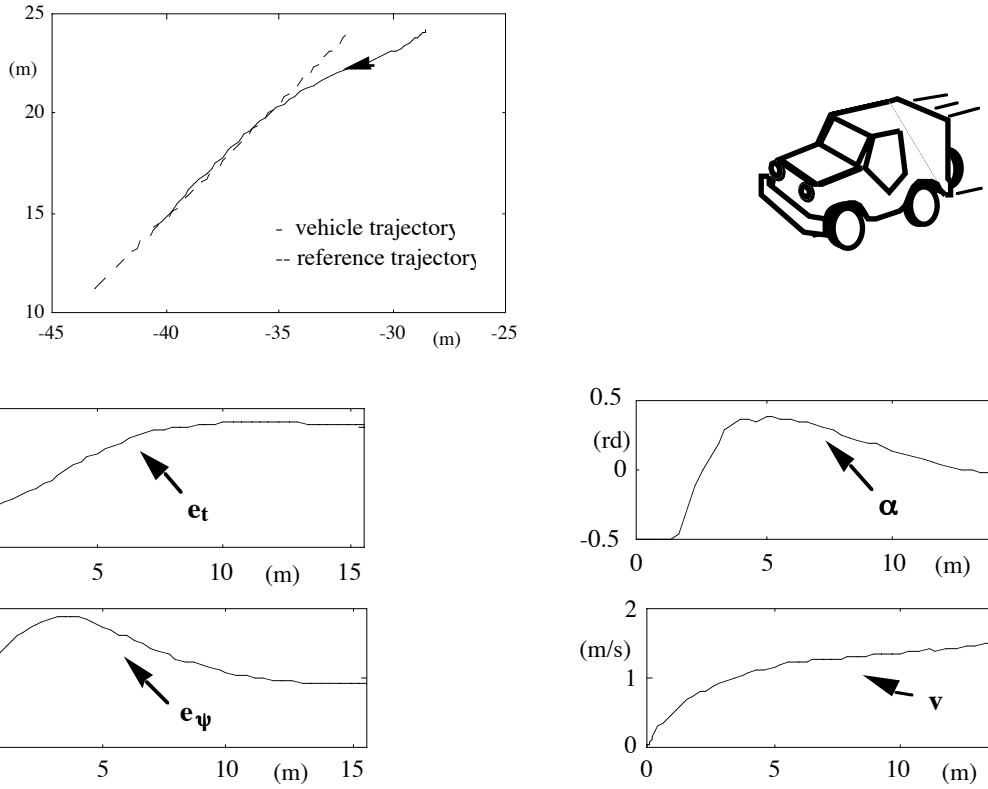


FIGURE 9

Map of the reference trajectory (around the SAGEM research center, with total length ~ 600 m) used for the path following and trajectory servoing experiments of figures 10 and 13.



a) Path following : experimental results.



b) Servoing the reference trajectory with an initial transversal error of *2,5 m* : experimental results.

FIGURE 10
Experimental results on the vehicle REMI using the heading-based approach
(e_t = transversal error, e_ψ = heading error, α = steering command (neural controller output), v = vehicle velocity).

3.2.2 The posture-based approach

a) Training phase and simulation results

The neural controller task is now to bring the model output ξ_m as close as possible to the *posture* ξ_r defined by a reference model. Feedback is now required from the state :

$$S_m(k) = [y_m(k),\ \psi_m(k),\ \beta_m(k),\ v_m(k)]$$

The cost-function to minimize over a trajectory of time-length N_t involves the posture, thus ψ_m and y_m :

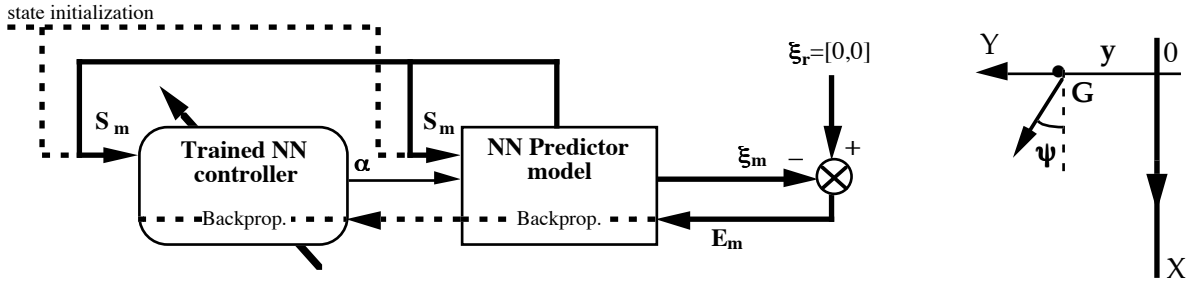
$$J = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} (y_r(k)-y_m(k))^2 + F_\psi . (\psi_r(k)-\psi_m(k))^2 = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} \|E_m(k)\|^2$$

where F_ψ is a weighting factor which has to be determined.

In this case, the computation of a minimum-time controlled reference model is quite difficult ; without taking the velocity saturation of the actuator into account, it is already a non-trivial task and requires some tedious algebra (see [RIN93]). Another choice could be to fix a distance depending on vehicle speed within which the robot should be brought back on the reference trajectory, and to define the path (polynomial (e. g. [SHI90]), splines…) satisfying the boundary conditions and the constraints on curvature and curvature variation. We found it more tractable to use a simple " identity " reference model :

$$\xi_r(t) = [0, 0]$$

and to let the cost-function, hence the weighting factor F_ψ, determine entirely the desired behaviour.

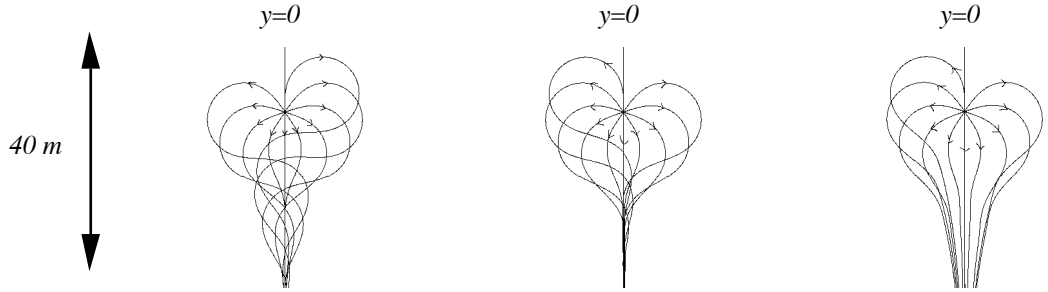


a) Controller training architecture using an UD algorithm. **b) Reference posture.**

FIGURE 11

Posture-based approach : training of the steering controller with identity reference model.

We performed the training using an UD algorithm with $N_c=N_t$, N_t being chosen sufficiently long for the model to stabilize on the *y=0* trajectory. The state of the model was randomly initialized in the range $\psi(0) \in [-\pi;+\pi]$ *rd*, $y(0) \in [0;10]$ *m*, and with constant speeds $v \in [0;10]$ *m/s*. As expected, the choice of the weighting factor in the cost-function proves to be decisive. Figure 12 shows simulation results obtained with three different values for F_ψ , which led us to the choice of $F_\psi = 10$.



a) $F_\psi=1$ **b)** $F_\psi=10$ **c)** $F_\psi=50$

FIGURE 12

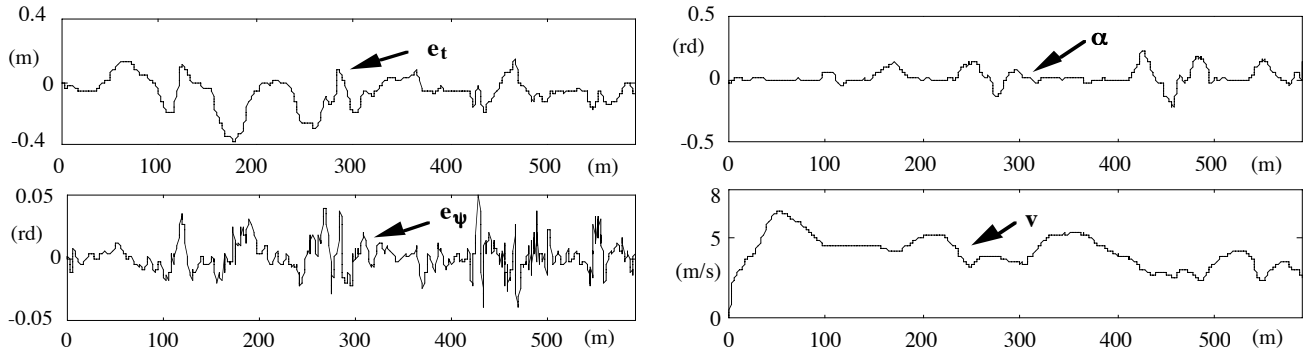
Simulation of the model behaviour depending on the weighting factor F_ψ in the cost-function used for training.

The neural controller is a MLP (3,3,3,1), with inputs *y*, ψ, *v* and output α (as in the heading-based approach, β is not used).

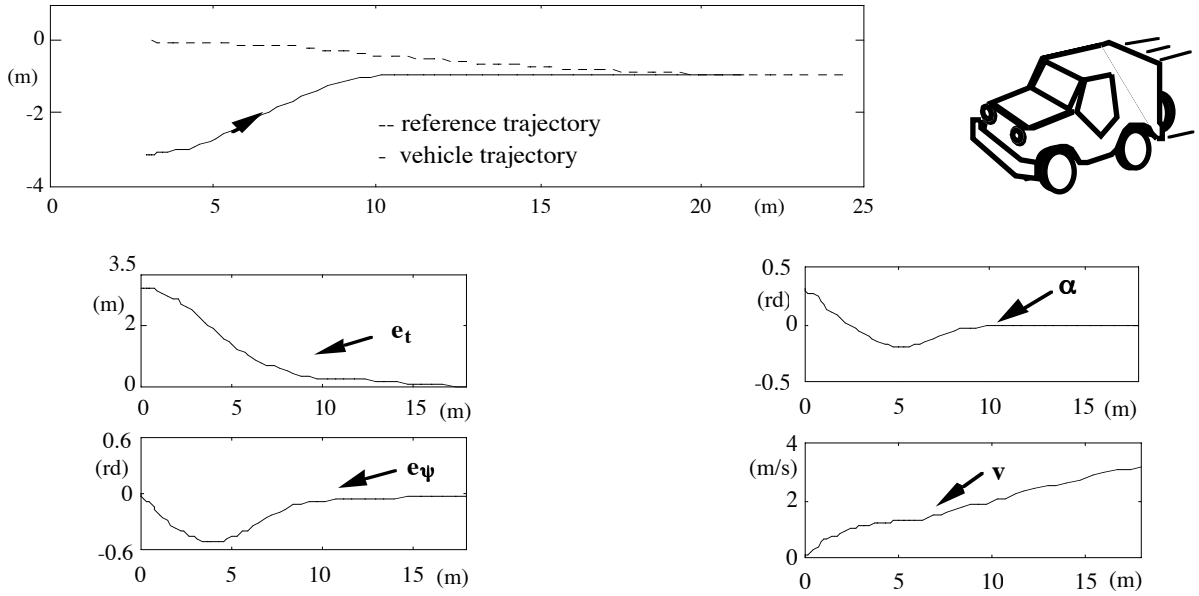
b) Operating phase and experimental results

During the operating phase, the distance *D* to the target-point was also chosen to depend linearly on vehicle speed, and the appropriate values of d_0 and F_v for this case were also determined by computer simulations.

Figure 13a shows the experimental path following performance of the controller on the same trajectory (the trajectory shown in figure 9). With an equivalent speed profile, both the transversal error ($e_t<35$ *cm*) and the heading error ($e_\psi<0{,}05$ *rd*) are smaller than in the heading-based approach. Figure 13b shows the behaviour for the trajectory servoing task : the trajectory is reached without overshoot.



a) Path following : experimental results.



b) Servoing the reference trajectory with an initial transversal error of 3 m : experimental results.

FIGURE 13
Experimental results on the vehicle REMI using the posture-based approach
(e_t = transversal error, e_ψ = heading error, α = steering command (neural controller output), v = vehicle velocity).

4. DISCUSSION

Both approaches, especially at high speed, lead to better results than the classical approach used before, consisting in LQ control (with fixed gains), based on the linearized model, without taking the actuator dynamics into account. The posture-based approach leads to better behaviour for the trajectory servoing task. This is an important property, which plays a role when a relocalization procedure is activated for example. The choice of the target-point is also extremely important : we compared our strategy to the commonly used geometric approach where the target-point is simply the closest point on the trajectory, and where a *feed-forward* curvature term is added to the command output by the feedback controller [RIN93][VDM93]. Our approach proved to be much more robust with respect to curvature variations, and does not necessitate the estimation of the curvature, which is often a problem (be it by image processing [JUR93], or with a trajectory defined by a series of setpoints - which is our case).

We would further like to stress the following advantages of the neural control scheme developed in part 2, of both theoretical and practical nature, as they are illustrated by our application :`

- The use of a *reference model* allows roughly two types of control policies : (i) In the case where a reference model can be defined, the right choice of the learning algorithm and of its parameters (N_C, N_t) makes it possible to assign *explicitly* its dynamics to the closed-loop system ; this was shown with the heading-based approach, where an optimal controller was learned by the neural network ; not only do we obtain optimal solutions in the sense of a chosen criterion (and to a given accuracy) but, in addition to their optimality, these solutions take a *closed-form expression*, as pointed out in [RIN93]. (ii) In the case where it is difficult to formulate the desired dynamics explicitly, the optimal behaviour is defined *implicitly* by the cost-function, much as in optimal control ; this was illustrated with the posture-based approach.

- Any *non-linear model of the process* can be used, be it a neural network, or a physical model, provided its jacobian can be evaluated and used for the computation of the gradient. Known non-linearities such as saturations introduced by the actuators can be simply incorporated in the model using saturation functions instead of standard sigmoids. The algorithms can then be applied regardless of the complexity of the model (and of the controller).
- The neural controllers are of small size (at most ten neurons), making their implementation easy. There is *no need for special purpose hardware* to operate in real time, even at high frequency control rates (up to *12.5 Hz* for the heading-based approach, *5 Hz* for the posture-based). Neural networks are well suited to real time operation, in contrast to traditional, iterative methods of solving optimal control problems.

5. CONCLUSION

The neural control scheme outlined in this paper applies successfully to non-linear systems such as wheeled mobile robots with actuator limitations. As a demonstration, the design of a neural controller for the trajectory following and servoing problem using two different approaches has been presented, already showing the promising performance and flexibility of the neural control framework. Future studies will deal with the extension of this framework to more difficult operating conditions (higher speed, rough terrain) requiring adaptive identification and control schemes. This work is also being currently extended to the velocity control using brakes, throttle and gear.

Acknowledgements

The authors are grateful to Philippe Lemoine and Stéphane Sallé (SAGEM) for numerous discussions and their help with the experiments, and wish to thank Michel de Cremiers (SAGEM) for initiating and efficiently supporting the project.

6. REFERENCES

[FRA93] Frappier G. (1993) " MINERVE : navigation - guidage - pilotage de véhicules autonomes ", Journée thématique DRET : " Vers une plus grande autonomie des robots mobiles ", janvier 1993, Paris.

[JUR93] Jurie F., Rives P., Gallice J. & Brame J. L. (1993) "High speed vehicle guidance based on vision", 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 205-210.

[LAN79] Landau Y. D. (1979) *Adaptive Control : the Model Reference Approch*, Marcel Dekker.

[NAR90] Narendra K. S. & Parthasarathy K. (1990) "Identification and control of dynamical systems using neural networks", IEEE Trans. on Neural Networks vol.1 No.1, pp. 4-27.

[NAR91] Narendra K. S. & Parthasarathy K. (1991) " Gradient methods for the optimization of dynamical systems containing neural networks ", IEEE Trans. on Neural Networks **2**, 252-262.

[NER92a] Nerrand O. (1992) "Réseaux de neurones pour le filtrage adaptatif, l'identification et la commande de processus", Thèse de doctorat de l'Université Paris VI.

[NER92b] Nerrand O., Roussel-Ragot P., Personnaz L., Dreyfus G. & Marcos S. (1992) " Neural networks and non-linear adaptive filtering : unifying concepts and new algorithms ", Neural Computation **5**, 165-199.

[NER93a] Nerrand O., Personnaz L. & Dreyfus G. (1993) " Non-linear recursive identification and control by neural networks : a general framework ", European Control Conference 1993.

[NER93b] Nerrand O., Roussel-Ragot P., Personnaz L. & Dreyfus G. (1993) " Training recurrent neural networks : why and how ? An illustration in process modeling ", IEEE Trans. on Neural Networks, in press.

[RIN93] Rintanen K. T. (1993) " Curvature-optimal path planning and servoing for autonomous vehicles : a neural net implementation ", 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 475-480.

[SAM92] SAMSON C. (1992) " Path following and time-varying feedback stabilisation of a wheeled mobile robot ", Proc. Conf. ICARCV'92, Singapore, September 1992.

[SHI90] Shin D. H. & Singh S. (1990) " Vehicle and path models for autonomous navigation ", in *Vision and navigation : the Carnegie Mellon Navlab*, Thorpe C. E. ed., Kluwer Academic Publishers, Boston, pp. 283-307.

[VDB93] Van den Bogaert T., Lemoine P., Vacherand F. & DO S. (1993) " Obstacle avoidance in PANORAMA ESPRIT II project ", 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 48-53.

[VDM93] Van der Molen G. M. (1993) " Modelling and control of a wheeled mobile robot ", 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 289-294.

IEEE Conference on Intelligent Vehicles, october 24-26 1994, Paris, France.

MODELING AND CONTROL OF MOBILE ROBOTS AND INTELLIGENT VEHICLES BY NEURAL NETWORKS

Isabelle RIVALS*, Daniel CANAS*, Léon PERSONNAZ** and Gérard DREYFUS**.

* SAGEM Eragny, Unité R&D, Avenue du Gros Chêne, 95 610 Eragny, France.
Phone : 33 1 34 30 52 07 ; Fax : 33 1 34 30 50 96 ; E-mail : rivals@sagem.fr.

** ESPCI, Laboratoire d'Électronique, 10 rue Vauquelin, 75 005 Paris, France.
Phone : 33 1 40 79 44 62 ; Fax : 33 1 40 79 44 25 ; E-mail : personna@neurones.espci.fr.

Abstract : This paper introduces the four-wheel-drive vehicle REMI, a testbed developed by SAGEM for research purposes in mobile robotics and intelligent car systems. The motion control architecture of the robot is presented, with an emphasis on the guidance and piloting modules. The latter relies on neural network techniques, and the principles underlying its design are outlined. A robust neural control scheme using an internal model of the process is developed. Experimental results are presented.

Keywords : intelligent cruise control, internal model control, neural networks, nonlinear identification and control, path following, wheeled mobile robots.

1. Introduction.

This paper describes the main features of an autonomous outdoor robot with neural network control. This robot is a fully automated standard 4WD vehicle called REMI (Robot Evaluator for Mobile Investigations) that is used as a testbed for motion control. The main components of the motion control architecture are path planning, guidance and piloting modules. Their hierarchical organization will be presented in section 2. In section 3, we will focus on the guidance module. Classical, robust, and neural piloting control techniques have been successfully experimented on REMI. We will however devote section 4 to a piloting module that is entirely designed using neural network techniques, which have achieved very good performances and offer new perspectives for intelligent vehicle control.

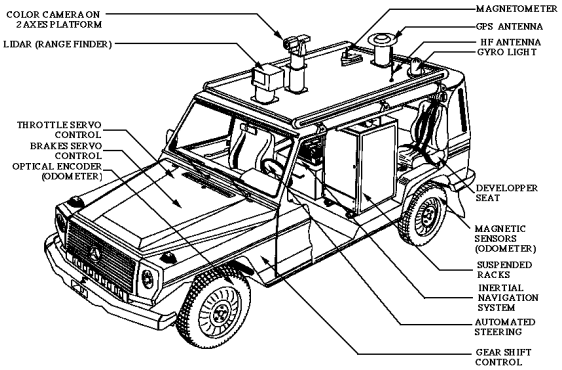


Figure 1.
REMI : the motion control testbed.

Typical automotive applications range from fully autonomous robots (open mine industry, construction, forest exploitation, agriculture [VDB93]) to automated functions on standard vehicles, such as intelligent cruise control systems, controlling the accelerator and braking systems with neural controllers.

2. Functional architecture.

The functional motion control architecture is divided in four separate modules, which are hierarchicaly organized, as shown in figure 2.

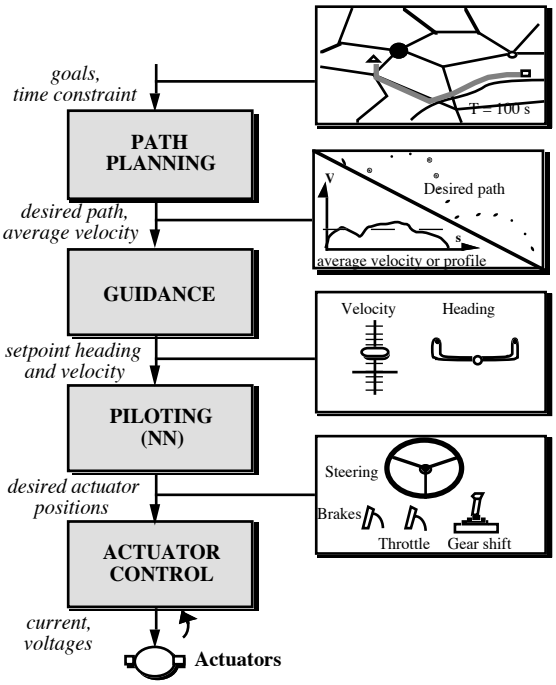


Figure 2.
Motion control architecture.

These modules achieve the following tasks : the path planning, the guidance, the neural network piloting, and the actuator control.

The path planning module.

A final goal position being given by the operator, the path planning module computes a path as a set of points satisfying this endpoint constraint. This path is associated to a time constraint specified by the operator in terms of an average velocity. A " teaching by doing " procedure can replace this module. In this case, path and velocity are " tought " by driving the vehicle manually while its position and velocity are recorded. The set of points and the average velocity are sent to the guidance module.

The guidance module.

The guidance module first generates a continuous reference trajectory and a velocity profile. Then, at each cycle of operation, this module chooses a target-point on the reference trajectory and computes a setpoint heading ψ_c from vehicle and target-point postures (i. e. position and heading). This setpoint heading and the velocity associated to the target-point, the setpoint velocity v_c, are sent to the piloting module.

The neural network piloting module.

The piloting module consists of two separate neural controllers. The heading controller computes a desired steering wheel position α. The velocity controller generates a desired position for the throttle of the engine θ and a desired pressure for the braking system π. These control intputs, or desired actuators positions, are sent to the next module at 20 Hz.

The actuator control module.

This module is interfaced with the actuators through power boards. Three digital-analog servo-loops running at 100 Hz monitor the angular position of the steering wheel, the throttle valve angular position, and the pressure in the braking system.

A localisation module computes the position, the attitude and the velocity of the vehicle, using an inertial dead-reckoning unit and an odometric sensor. Localisation and motion control modules are implemented on a 68030 board running under the real-time operating system OS9 [VDB93].

3. Guidance module.

First a continuous reference trajectory with an associated velocity profile is computed from the set of points and average velocity determined by the path planning module.

Then, at each sampling time, a setpoint heading and a setpoint velocity are generated for the piloting module. Our control strategy is based on a target-point approach. A target-point is defined for a point of the vehicle body, the control-point, which is chosen at the center of the rear axle. This location is of course weakly controllable, due to the nonholonomy constraint, but presents the following advantages :

- it coincides with the localisation point (the position computed by the dead-reckoning unit),
- steering and velocity control of this point can be decoupled,
- its turning radius is the smallest,
- the heading ψ of the vehicle at this point is tangent to the path.

The following section is devoted to the trajectory generation, section 3.2 to the choice of the target-point, and section 3.3 to the computation of the setpoint heading ψ_c.

3.1. Trajectory generator.

The aim of this function is to generate a smooth, feasible reference trajectory, interpolating the set of points sent by the path planning module. Previous work established criteria for trajectories to be suited to tracking [FRA 93]. B-splines and clothoïds both show continuity in position, heading and curvature. But B-splines are based on polynomial functions which are easely computed, integrated and differentiated, and are therefore the best candidates for real-time implementation. The trajectory generator also generates a velocity profile respecting a maximum allowable lateral acceleration and the kinematic constraints of the vehicle.

3.2. Choice of the target-point.

It is chosen at a so called " lookahead " distance D in front of the vehicle. D is a function of vehicle velocity, which must be chosen so as to guarantee that :

- D is not too small : otherwise, the vehicle might reach the target point between two computations, or oscillations might appear.
- D is not too big : if this condition is not fulfilled, the vehicle might cut corners.

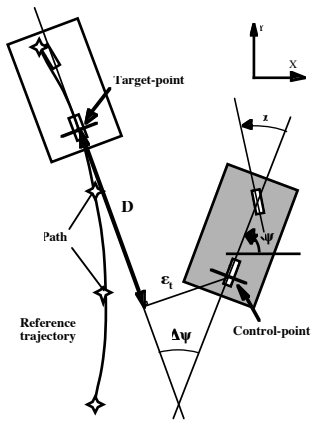


Figure 3.
Target-point and control-point postures.

Reference trajectory, lookahead distance D, control-point and target-point are shown in figure 3. $\Delta\psi$ denotes the orientation error, and ε_t the transversal error.

3.3. Setpoint heading generator.

The aim is to define a setpoint heading ψ_c for the heading controller of the piloting module. Thus ψ_c must be defined in order that if ψ converges to ψ_c, then ε_t and $\Delta\psi$ both converge to 0. Different choices of ψ_c are described in [FRA93] and [RIV93].

4. Neural network piloting module.

We address the lateral and longitudinal piloting of the vehicle, that is how to have the vehicle follow the setpoint heading and velocity determined by the guidance module. Why is a neural network approach advantageous, and how to achieve the modeling and control of the vehicle using recurrent neural networks ?

4.1. Why?

Wheeled mobile robots are nonlinear dynamical systems ; their kinematics involves geometrical nonlinearities, and their actuators introduce dynamical nonlinearities, e. g. the saturations of the steering wheel actuator and the nonlinear dynamics of the thermal engine. Thus, the identification and control of these processes require nonlinear models and controllers.

Optimal control theory has been widely used to solve such nonlinear, constrained problems. But the conventional scheme of optimal control has its own drawbacks. Very often, finding the optimal control trajectory is time consuming, and the solution obtained consists of a sequence of open-loop control vectors.

Neural networks offer interesting solutions of nonlinear control problems. Their approximation properties make them a useful tool for modeling nonlinear processes, and they provide a solution to the problem of getting a closed-form expression for optimal control laws. Besides, robustness considerations led us to develop an internal model based approach that was successfully applied to the longitudinal control of the vehicle, as will be shown in part 5. Finally, generic training algorithms were established, regardless of model and controller complexity. They will be presented in the next section.

4.2. How?

A general framework for training recurrent networks for nonlinear modeling and control is now outlined.

4.2.1. Training the neural model.

We assume that the process to be controlled can be described by the following model :

$$(1)\qquad \begin{cases} S_p(n+1) = f\left[S_p(n),\, D_p(n),\, U(n)\right] \\ y_p(n+1) = g\left[S_p(n+1)\right] + w(n+1) \end{cases}$$

where y_p is the output of the process, S_p its state, D_p are measured disturbances, and U is the control input. f and g are unknown functions. w is an output additive white noise modeling a measurement noise : it has been shown [NER94] that the optimal predictor for such a process is recursive. Thus, the model must be trained as the following feedback predictor :

$$(2)\qquad \begin{cases} S(n+1) = f_{NN}\left[S(n),\, D(n),\, U(n);\, C_m\right] \\ y(n+1) = g_{NN}\left[S(n+1);\, C_m\right] \end{cases}$$

where y is the output of the model, S its state, and f_{NN} and g_{NN} are the functions implemented by the static part of the neural net with weights C_m, which must be estimated.

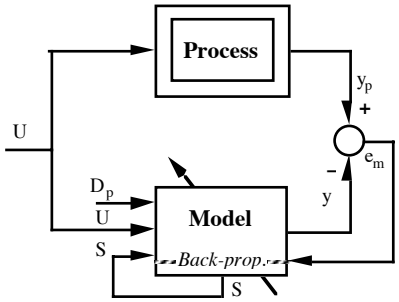


Figure 4.1.
Model training architecture.

The training of the model (see figure 4.1), i.e. the estimation of its weights C_m, is performed by minimizing the cost function J_m :

$$J_m = \frac{1}{2}\sum_{n=1}^{N}\left(e_m(n)\right)^2 = \frac{1}{2}\sum_{n=1}^{N}\left(y_p(n) - y(n)\right)^2$$

The minimization is carried out by a gradient-based technique. The gradient is computed by back-propagation on a fixed window of size N (non recursive, iterative training) using a semi-directed algorithm [NER92]. The size N of the window corresponds to the amount of training data available from the process.

4.2.2. Training the neural controller.

We perform the training of the controller using the neural model and a reference model, as shown in figure 4.2. The neural net controller with weights C_c implements the nonlinear state-feedback :

$$(3)\qquad U(n) = h_{NN}\left[U(n\text{-}1),\, S(n),\, D(n),\, R(n);\, C_c\right]$$

where D are simulated disturbances and {R(n)} is the setpoint sequence.

The weights of the neural model (2) are set to the values computed during its training.

A reference model is designed to generate the desired output sequence, or reference sequence, $\{y_r(n)\}$:

$$(4) \qquad \begin{cases} S_r(n+1) = f_r\left[S_r(n),\, R(n)\right] \\ y_r(n+1) = g_r\left[S_r(n+1)\right] \end{cases}$$

where S_r is the state of the reference model ; f_r and g_r are chosen by the designer.

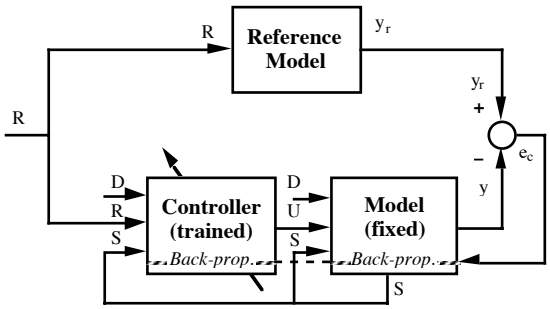


Figure 4.2.
Controller training architecture.

The weights of the controller C_c are computed so as to minimize the cost-function J_c :

$$J_c = \frac{1}{2}\sum_{n=1}^{N}\left(e_c(n)\right)^2 = \frac{1}{2}\sum_{n=1}^{N}\left(y_r(n)-y(n)\right)^2$$

Again, back-propagation on the fixed window of size N, fixed by the designer, is used to compute the gradient of J_c.

4.2.3. Using the neural controller.

a) Simple state-feedback control (SFC).

The operating control system is shown in figure 4.3. The controller, with fixed weights, computes its output from the process state, measured disturbance and setpoint :

$$(5) \qquad U(n) = h_{NN}\left[U(n\text{-}1),\, S_p(n),\, D_p(n),\, R(n)\right]$$

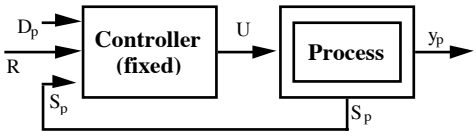


Figure 4.3.
Operating phase : SFC system.

For this control system to be stable and show good tracking and regulation properties, the neural model used for the training of the controller has to be quite accurate.

b) Internal model control (IMC).

The IMC structure incorporates a model of the process to be controlled that is simulated on-line in the control computer. This recursive model is only fed with the process inputs (controls and disturbances). Using neural networks, the model must be the feedback neural model the controller was trained with. The basic IMC structure is shown in figure 4.4. The controller computes its output as follows :

$$(6) \qquad U(n) = h_{NN}\left[U(n\text{-}1),\, S(n),\, D_p(n),\, R^*(n)\right]$$

where S is the state of the model, $R^*=R-e_{im}$, and $e_{im}=y_p-y$. Thanks to the second loop, model uncertainty and measured output disturbances are taken into account.

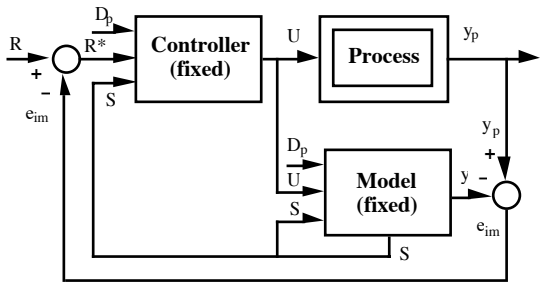


Figure 4.4.
Operating phase : IMC system.

IMC has many desirable properties, which were extensively analyzed for linear systems in [MOR89], and can be generalized to nonlinear systems [ECO86]. Thus, our neural IMC system has the two following characteristics :

(a) If the plant and the controller are input-output stable, and if the model is perfectly accurate, then the closed loop system is also input-output stable.

(b) If the controller steady-state gain is equal to the inverse of the model steady-state gain (assuming its existence), and if the closed-loop system is stable with this controller, offset-free control is obtained for constants inputs.

If the neural controller is trained using a reference model with unity steady-state gain, then the first condition of (b) is met.

One must be aware that the validity domain of the neural controller is restricted to the region of the state space it was trained in. This training region must be larger than the operating region desired for the process. As a matter of fact, during the operating phase, the controller controls the model (its input is the state of the model), which might evolve in a larger region of the state space than the process, due to the mismatch between model and process and to perturbations. In the case the model should leave the validity domain of the neural controller during the operating phase, the model state must be reset to the process state.

One can also train the controller to implement the inverse of the operator describing the plant model (if it exists) by using a reference model consisting of a simple delay for the training. In this case, a filter must be introduced between setpoint and controller in order to achieve the desirable robustness against perturbations, and the desired

tracking response [MOR89]. If the model is perfectly accurate, perfect tracking can be achieved.

5. Experimental results.

The piloting problem is split into two separate control problems (see part 2), a lateral or heading control problem, and a longitudinal or velocity control problem. The lateral problem has been presented more extensively in [RIV93].

5.1. Lateral modeling and control.

The usually adopted "bicycle model " proved to be only a rough approximation of REMI's lateral behaviour. We thus performed an identification of the nonlinear relationship between the vehicle heading ψ_p and the steering wheel control input α. Incorporating *a priori* knowledge (the basic structure of the bicycle model) into the neural model led to a good generalization in regions where training data was not sufficient (at high velocity). In addition, we could model the nonlinear first order dynamics with two saturations (in angle and velocity) of the steering wheel actuator. The neural model consists of two subnetworks NNβ and NNψ :

$$\begin{cases} \beta(n+1) = f_{NN\beta}\left[\beta(n),\, v_p(n),\, \alpha(n)\right] \\ \psi(n+1) = \psi(n) + v_p(n)\, f_{NN\psi}\left[\beta(n)\right] \end{cases}$$

where $\beta(n)$ is the model steering wheel angle. The velocity of the vehicle v_p is considered as a measured disturbance.

The heading controller was designed as a static state-feedback regulator :

$$\alpha(n) = \rho_{NN}\left[e_\psi(n),\, \beta(n),\, v(n)\right]$$

$e_\psi(n)=\psi_c(n)-\psi(n)$, $\psi_c(n)$ being the heading setpoint given by the guidance module.
The reference model used for the training of the controller (figure 4.2) was designed to compute minimal-time rallying sequences $\{\psi_r(n)\}$ rallying ψ to ψ_c for the linearized model, at various velocities (0-80 km/h) [RIV93]. Thus the neural controller was trained to be a minimal-time controller (see [RIV93] part 3.2.1 : the heading-based approach).

The model being very accurate, the heading controller could be used with very good performances in a SFC system (figure 4.3). Since the process itself is an integrator, zero steady state error could be achieved.

Experimental lateral control results (path following) are shown in figure 5.1. As described in section 3.3, the setpoint heading was computed by the guidance module so as to bring the vehicle onto the reference trajectory. The transversal error shown is the distance to the closest point of the reference trajectory, and the heading error is defined with respect to the tangent to the trajectory at this point. The velocity was monitored by a human operator, with a mean value of 4,5 m/s. The transversal error did not exceed 40 cm, with curvature values of 0.1 m^{-1} in the sharp curves.

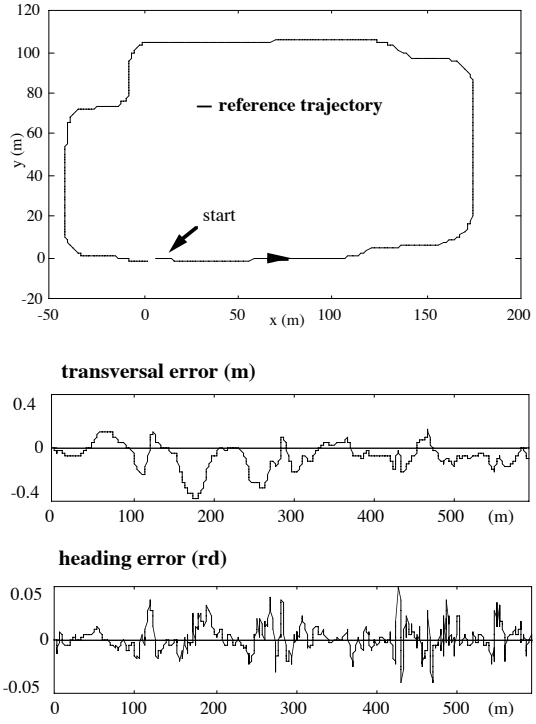


Figure 5.1.
Lateral control result (SFC system).

5.2. Longitudinal modeling and control.

The model of the longitudinal dynamical behaviour of the vehicle is a second order neural model :

$$v(n+1) = g_{NN}\left[v(n),\, v(n\text{-}1),\, \rho_p(n),\, \theta(n),\, \pi(n)\right] - G\, sin\left[s_p(n)\right]\; k\left[\rho_p(n)\right]$$

where v is the velocity, θ the throttle angular position and π the brake pressure. The (automatic) gear transmission ratio ρ_p is considered as a measured disturbance. G is the gravity constant, k a known inertia factor, and s_p the measured slope of the terrain.

The velocity controller was designed as a dynamic state-feedback controller implementing the inverse of the model (i. e. it was trained with a reference model consisting of a unit delay, within the training architecture shown in figure 4.2) :

$$U(n)=\kappa_{NN}\left[U(n\text{-}1),v_c(n),v_c(n\text{-}1),\rho(n),s(n),v(n),v(n\text{-}1)\right]$$

where $\{v_c(n)\}$ is the velocity setpoint sequence. The following switching logic was used to control throttle and brakes alternatively :
if $u(n)>0$ then $\theta(n) = u(n)$ (throttle), and $\pi(n) = 0$.
if $u(n)<0$ then $\pi(n) = u(n)$ (brakes), and $\theta(n) = 0$.

The longitudinal model being less accurate than the lateral one, we chose to use the controller trained with this model as a part of an IMC system (figure 4.4). We chose a damped second order low-pass linear filter with faster dynamics than the vehicle :

$v_r(n+1) = a_1\, v_r(n) + a_2\, v_r(n\text{-}1) + b_1\, v_c(n) + b_2\, v_c(n\text{-}1)$

where {vr(n)} is the reference sequence.

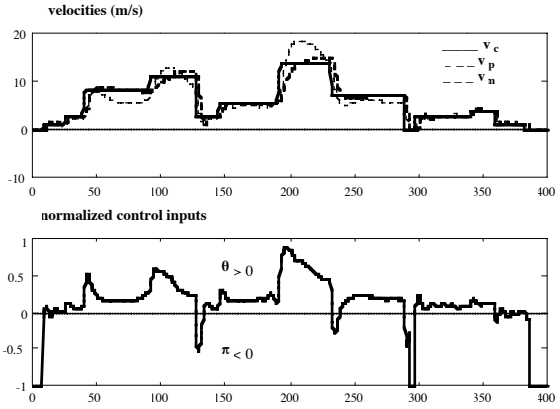


Figure 5.2.
Longitudinal control results (IMC system).

Experimental results are shown in figure 5.2. v_m, the thin dotted line, denotes the internal model velocity. These graphs show (i) the mismatch between model and process (i. e. between v_m and v_p, the thick dotted line), and (ii) that, despite this, zero steady-state error and good tracking dynamics are achieved.

6. Conclusion.

On the example of the piloting control system of a 4WD vehicle, we show the applicability of the neural network techniques to a complex control problem in automotive industry. The design of the control system was achieved using dynamical neural network models of the vehicle, and neural controllers implementing an optimal control law (lateral control) and a control law related to the inverse of the neural model in an internal model control scheme (longitudinal control). One of the salient features of our methodology is the use of *a priori* knowledge in the design of the model and controller networks [PLO94].

This piloting system was integrated in a fully automated robot ; systematic tests in rough terrain as well as comparisons to other approaches are in progress.

Acknowledgments.

This work was financed by SAGEM in the frame of a PhD study at ESPCI. We wish to thank Mrs Catherine Fargeon from DRET for her financial support to the MINERVE program, which was the starting base for the neural network project.

References.

[ECO86] Economou C. G., Morari M. & Palsson B. O. (1986) " Internal model control. 5. Extension to nonlinear systems ", Ind. Eng. Chem. Process Des. Dev., vol.25, pp. 403-411.

[FRA93] Frappier G. (1993) " MINERVE : navigation - guidage - pilotage de véhicules autonomes ", Journée thématique DRET : " Vers une plus grande autonomie des robots mobiles ", janvier 1993, Paris.

[MOR89] Morari M. & Zafiriou E. (1989) Robust process control, Prentice-Hall International Editions.

[NER92] Nerrand O., Roussel-Ragot P., Personnaz L., Dreyfus G. & Marcos S. (1992) " Neural networks and nonlinear adaptive filtering : unifying concepts and new algorithms ", Neural Computation Vol.5, pp. 165-199.

[NER94] Nerrand O., Roussel-Ragot P., Personnaz L. & Dreyfus G. (1994) " Training recurrent neural networks : why and how ? An illustration in process modeling ", IEEE Trans. on Neural Networks Vol.5, pp. 178-184.

The same methodology has been used in :

[PLO94] Ploix J.-L., Dreyfus G., Corriou J.-P., Pascal D. (1994) " From knowledge-based models to recurrent networks : an application to an industrial distillation process ", submitted to " Neural Information Processing Systems 94 ".

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

[VDB93] Van den Bogaert T., Lemoine P., Vacherand F. & Do S. (1993) " Obstacle avoidance in PANORAMA ESPRIT II project ", 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 48-53.