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.

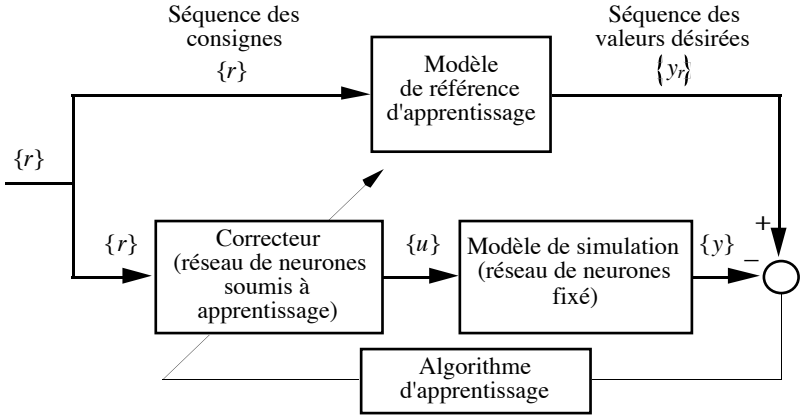


Figure 1.
Système d'apprentissage d'un correcteur neuronal.

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.

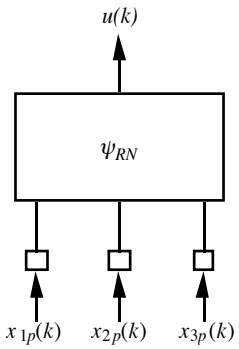


Figure 2.
Exemple de réseau régulateur non bouclé (n=3).

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*.

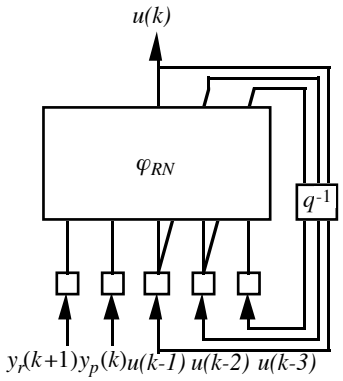


Figure 3.
Exemple de réseau correcteur bouclé (n=1, m=4).

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 (LQR). 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.

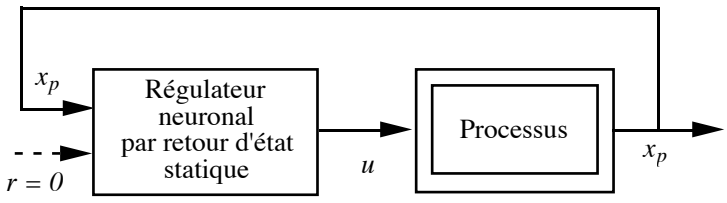


Figure 4.
Système de régulation de l'état autour d'un point d'équilibre par retour d'état statique neuronal.

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 smulation 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.

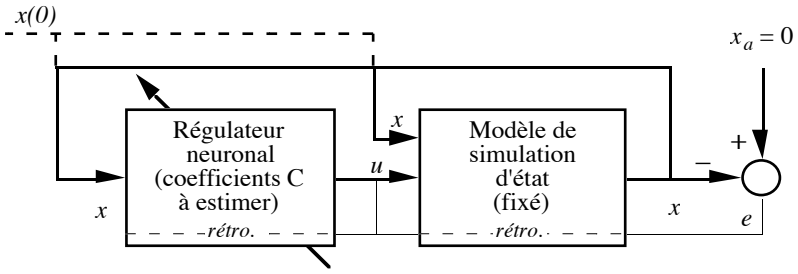


Figure 5.
Système d'apprentissage d'un régulateur optimal ab initio.

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. Le modèle et le régulateur peuvent ici être *dirigés* par le modèle de référence. Il est cependant recommandé d'effectuer un apprentissage semi-dirigé, pour lequel la performance du système d'apprentissage est plus 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). On peut éventuellement *initialiser* l'apprentissage en dirigé (non représenté).

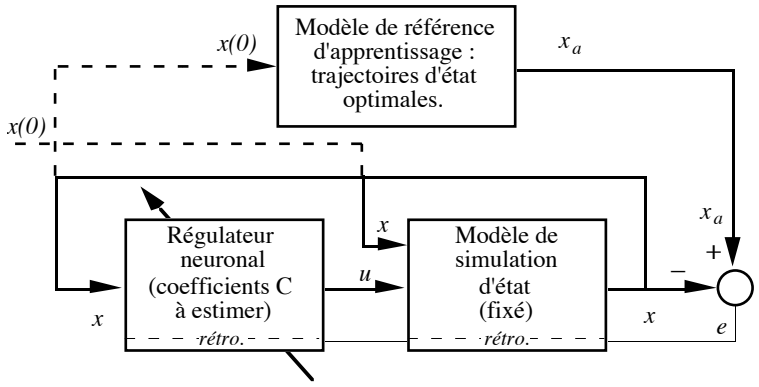


Figure 6.
Système d'apprentissage d'un régulateur optimal à partir de trajectoires optimales.

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).

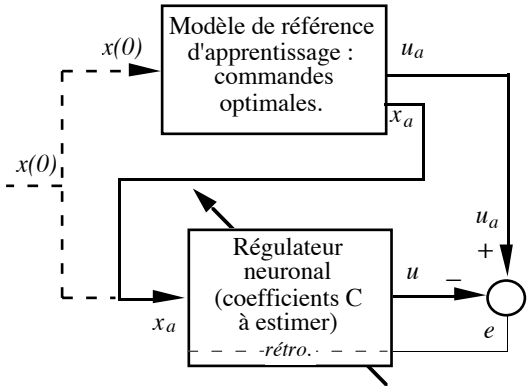


Figure 7.
Système d'apprentissage d'un régulateur optimal à partir d'un régulateur existant.

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∈R est la commande et y∈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 :

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

où u∈R est la commande, x∈$\mathbb{R}^n$ est l'état, et y∈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) :

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

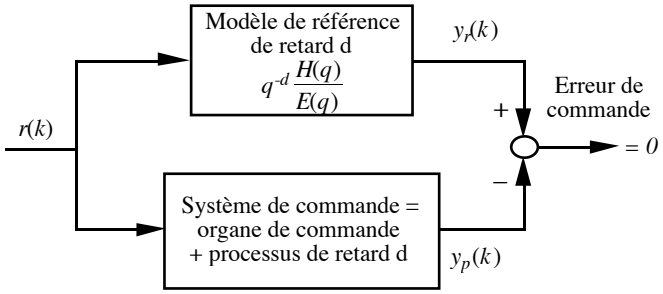


Figure 8.
Schéma de principe de la poursuite de sortie.

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 sortie en

[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].

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.

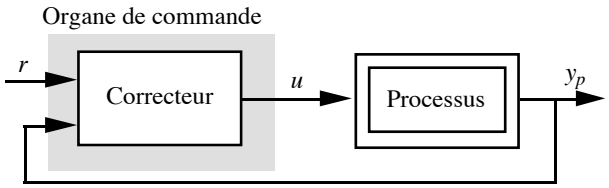


Figure 9.
Exemple de SCSB.

- ***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.

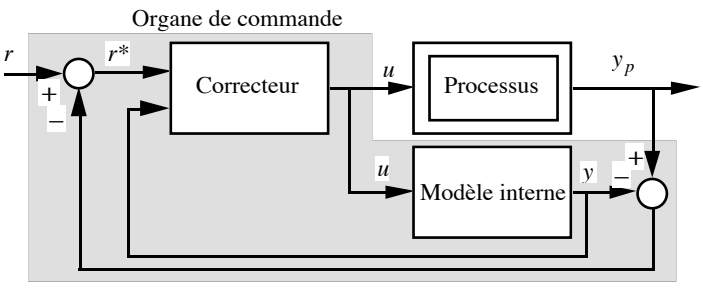


Figure 10.
Exemple de SCMI.

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 au modèle 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 après l'instant k, *la sortie du modèle 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.

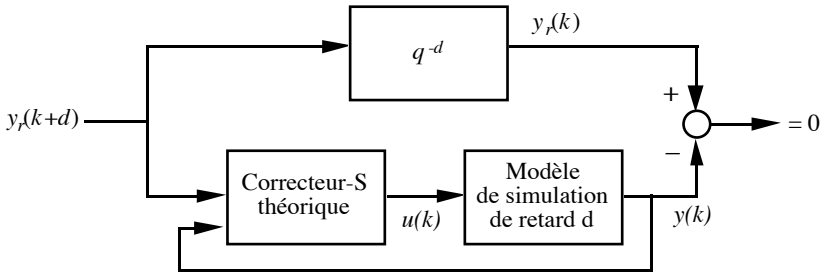


Figure 11.
Définition du correcteur-S théorique.

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 pour l'ensemble des points singuliers tels que : $b + n\, y(k) = 0$.

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 au modèle une Dynamique de référence. 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 modèle 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.

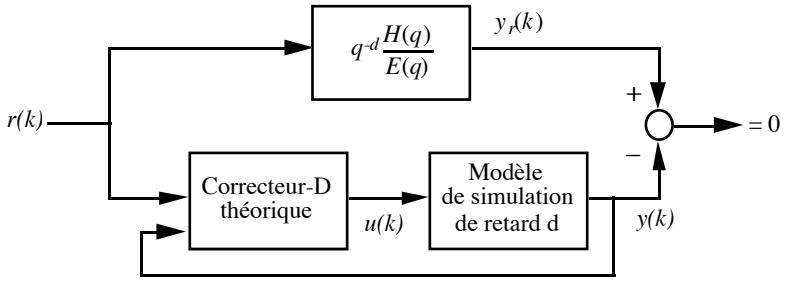


Figure 12.
Définition du correcteur-D théorique.

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 :

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

Comme pour le correcteur-S, cette loi n'est pas applicable pour l'ensemble des points singuliers tels que : $b + n\, y(k) = 0$.

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 :

$$u(k) = \frac{1}{CA^{d-1}B}\left[H(q)\ r(k) - \left(CA^{d} + e_1\ CA^{d-1} + \ldots + e_d\ C\right)x(k) - \left(e_{d+1}\ y(k-1) + \ldots + e_p\ y(k+d-p)\right)\right]$$

- 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 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 :

$$u(k) = \psi_{RN}\left(H(q)\ r(k),\ x(k),\ y\{_{k+d-p}^{k-1};\ C\right)$$

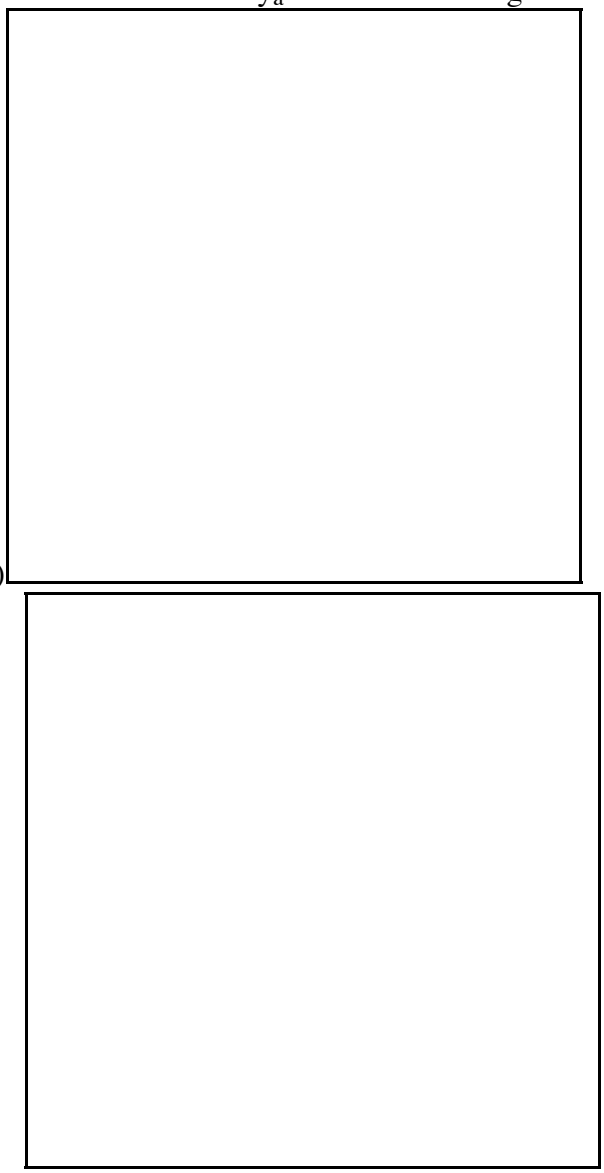
Le système d'apprentissage pour la réalisation du correcteur-S théorique par un réseau de neurones est décrit au §II.2.2 suivant.

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) :

$$J = \sum_{k=1}^{N} (y_a(k) - y(k))^2 = \sum_{k=1}^{N} e(k)^2$$

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 :

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

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.

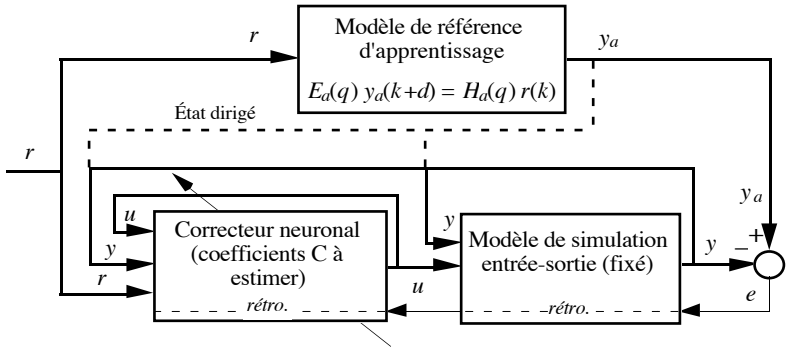


Figure 13.
Système d'apprentissage d'un correcteur-D à l'aide d'un modèle entrée-sortie du processus
(pour un correcteur-S $E_a(q) = H_a(q) = 1$).

Le système d'apprentissage est représenté sur la figure 13. Le caractère bouclé du correcteur impose un algorithme *semi-dirigé* (sauf si d=1 et m=1). 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 :

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

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 :

$$u(k) = \psi_{RN}\left(H(q)\ r(k),\ x(k),\ y\{_{k+d-p}^{k-1};\ C\right)$$

On retrouve le cas particulier du correcteur-D avec p≤d (§II.1.2.2), et celui du correcteur-S pour lequel p=0 (§II.1.1.2). Comme en entrée-sortie, la fonction d'activation de son neurone de sortie est souvent bornée.

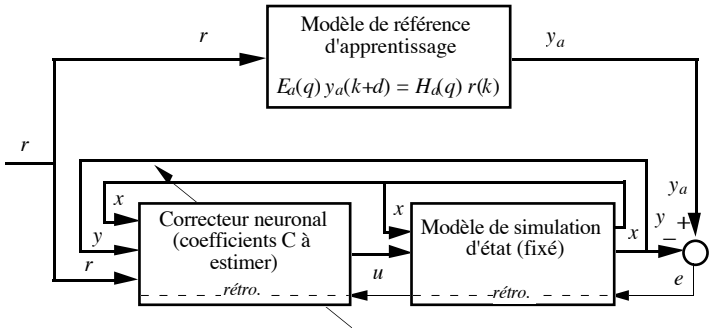


Figure 14.
Système d'apprentissage d'un correcteur -D à l'aide d'un modèle d'état du processus
(pour un correcteur-S $E_a(q) = H_a(q) = 1$).

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) :

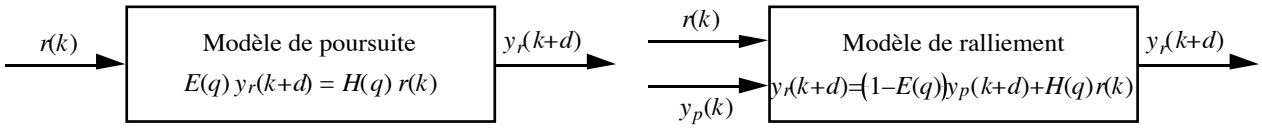


Figure 15.
Modèles de référence.

- *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) :

$$y_r(k+d) = \big(1 - E(q)\big)\, y_p(k+d) + H(q)\, r(k)$$

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 : $y_r(k+d) = \big(1 - E(q)\big)\, y_r(k+d) + H(q)\, r(k)$. 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 $\big(1 - E(q)\big)\, y_p(k+d)$ 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. En effet, même le système nominal a l'inconvénient majeur d'imposer une dynamique de régulation beaucoup trop rapide. En effet, la sortie du modèle de poursuite pour une consigne donnée est la même, que le processus suive ou non le modèle de référence (qu'il soit perturbé ou non). Ceci peut se traduire par un effort excessif sur la commande en cas de perturbation de sortie, et par 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].

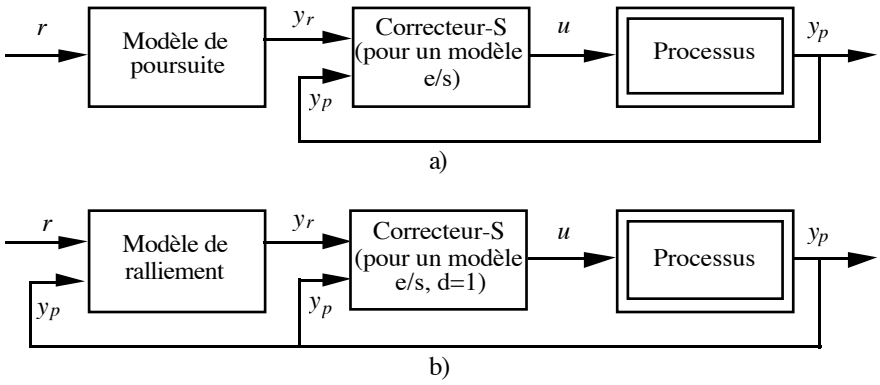


Figure 16.
SCSB avec correcteur-S (pour un modèle entrée-sortie du processus).

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).

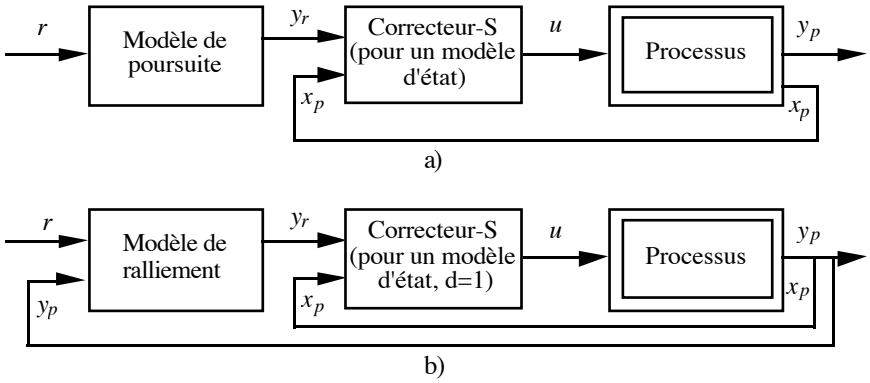


Figure 17.
SCSB avec correcteur-S (pour un modèle d'état du processus).

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.

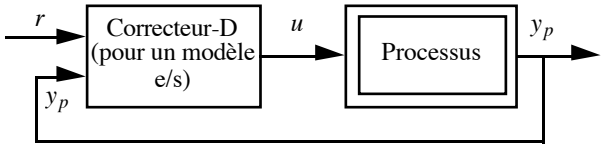


Figure 18.
SCSB avec correcteur-D (pour un modèle entrée-sortie du processus).

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. Il a l'avantage sur ce dernier de pouvoir être utilisé quel que soit le retard du processus. Dans le cas où d=1 cependant, 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 :

$$u(k) = \psi_{RN}^{D,\,état}\left(H(q)\ r(k),\ x_p(k),\ y_p\{_{k+d-p}^{k-1}\right)$$

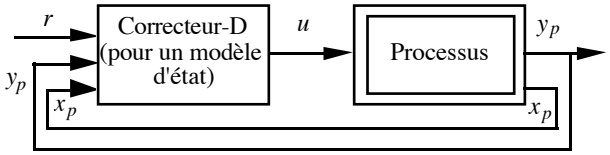


Figure 19.
SCSB avec correcteur-D (pour un modèle d'état du processus).

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.

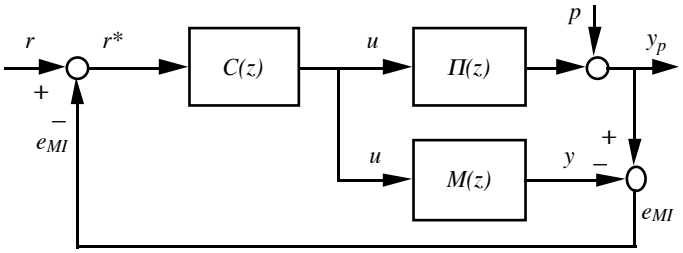


Figure 20.
SCMI linéaire.

La transformée en z du signal de rétroaction e_{MI} a pour expression :

$$E_{MI}(z) = \left(\prod(z) - M(z)\right) U(z) + P(z)$$

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) :

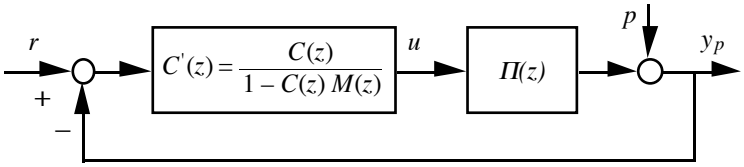


Figure 21.
SCSB équivalent au SCMI de la figure 20.

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. Ceci 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, ceci 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.

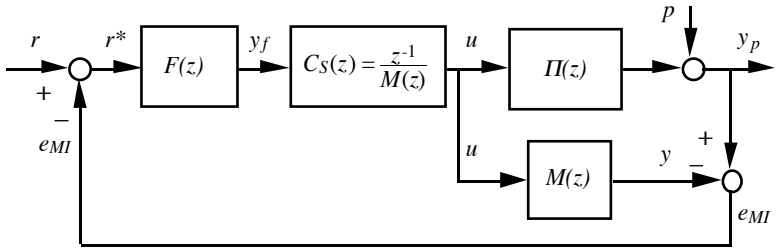


Figure 22.
Compromis stabilité-performance.

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).

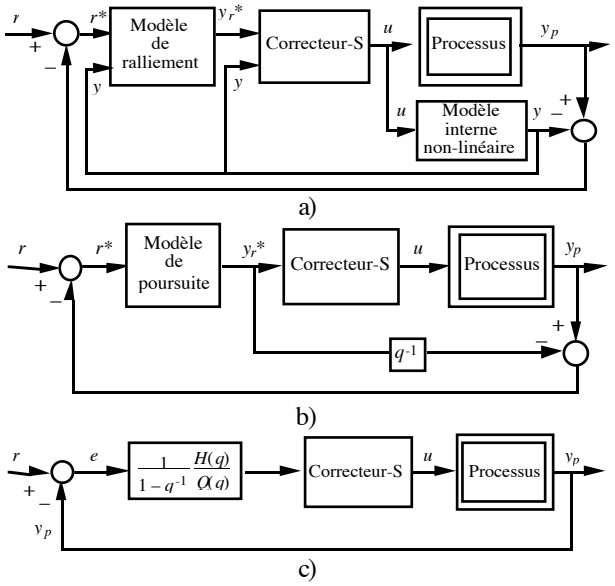


Figure 23.
Performance robuste du SCMI non linéaire avec correcteur-S (équivalences a-b et a-c valables *localement*).

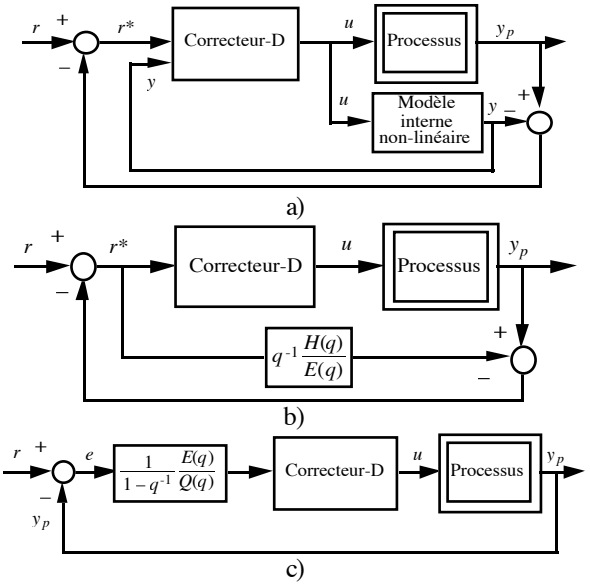


Figure 24.
Performance robuste du SCMI non linéaire avec correcteur-D (équivalences a-b et a-c valables *localement*).

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 :

$$u(k) = \varphi_{RN}^{S,\ e\text{-}s}\left(y_r^*(k+d),\ y\{_{k+d-n}^{k+d-1},\ u\{_{k-m'}^{k-1}\right)$$

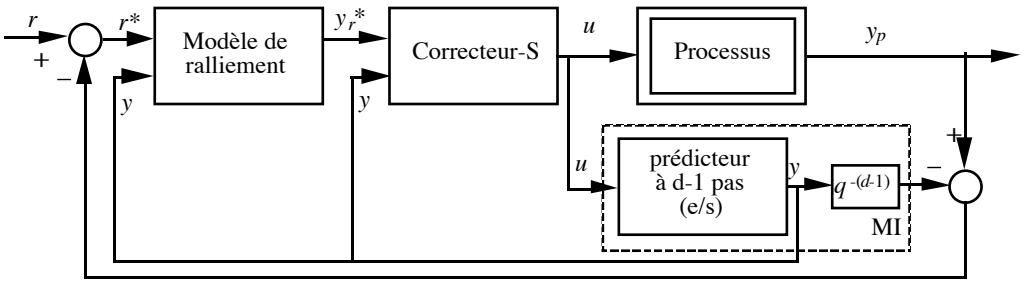


Figure 25.
SCMI avec correcteur-S et modèle de ralliement (modèle e/s).

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 :

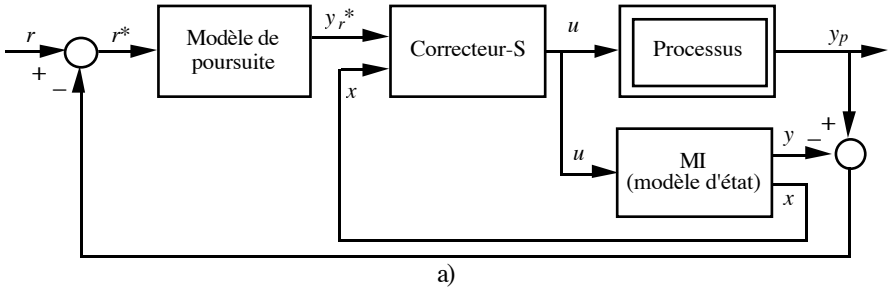
$$E(q)\, y_r^*(k+d) = H(q)\, r^*(k)$$

Le MI est le modèle de simulation utilisé pour l'apprentissage :

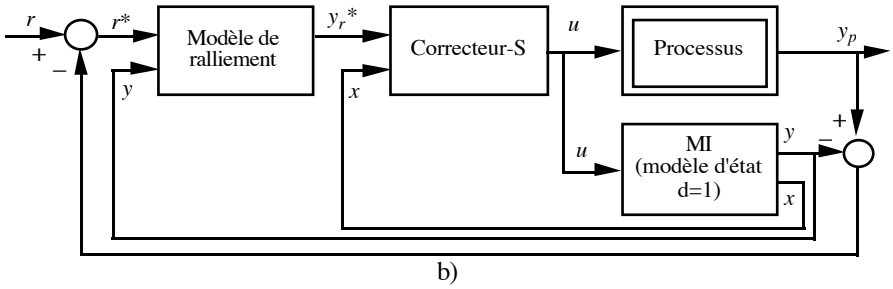
$$\begin{cases} x(k+1) = f\left(x(k),\, u(k)\right) \\ y(k) = g\left(x(k)\right) \end{cases}$$

Le correcteur-S obtenu par apprentissage, $\psi_{RN}^{S,\,état}$, est bouclé sur le modèle :

$$u(k) = \psi_{RN}^{S,\,état}\left(y_r^*(k+d),\, x(k)\right)$$



a)



b)

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 :

$$u(k) = \varphi_{RN}^{D,\,e\text{-}s}\left(H(q)\ r^*(k),\ y\{_{k+d-n}^{k+d-1},\ u\{_{k-m'}^{k-1}\right)$$

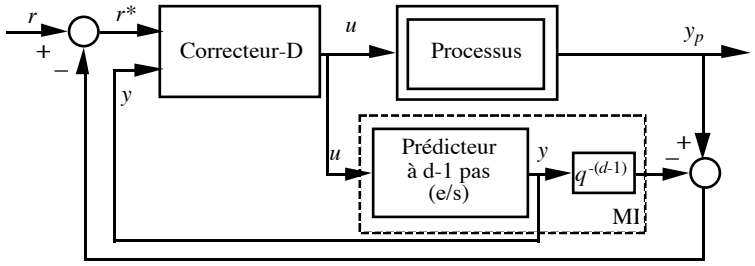


Figure 27.
SCMI avec correcteur-D (pour un modèle entrée-sortie).

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 :

$$u(k) = \psi_{RN}^{D,\,état}\left(H(q)\ r^*(k),\ x(k),\ y\{_{k+d-p}^{k-1}\right)$$

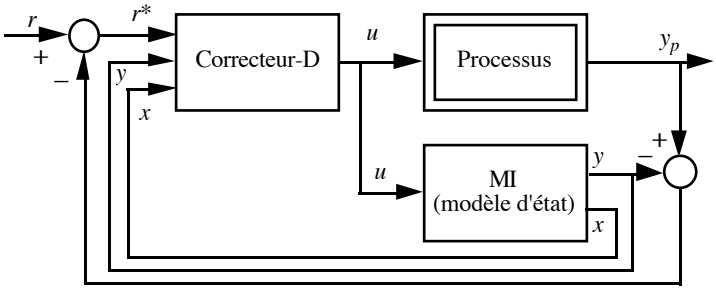


Figure 28.
SCMI avec correcteur-D (pour un modèle d'état).

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 est 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.