Contribution à l'étude des Systèmes à Fonctionnement par Morceaux : Application à l'Identification en Ligne et à la commande en Temps Réel

Afzal Chamroo

N° d'Ordre : 3827

UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE

Laboratoire d'Automatique, Génie Informatique et Signal (LAGIS)
UMR CNRS 8146

THÈSE

Présentée en vue de l'obtention du grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : Automatique et Informatique Industrielle

par

Afzal CHAMROO

CONTRIBUTION À L'ÉTUDE DES SYSTÈMES À FONCTIONNEMENT PAR MORCEAUX : APPLICATION À L'IDENTIFICATION EN LIGNE ET À LA COMMANDE EN TEMPS RÉEL

Soutenue publiquement le 29 juin 2006

JURY

Directeur	C. VASSEUR	Professeur *Université des Sciences & Technologies de Lille (USTL)*
Co-directeur	N. CHRISTOV	Professeur *Université des Sciences & Technologies de Lille (USTL)*
Rapporteurs	J. BERNUSSOU	Directeur de Recherche CNRS *Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS), Toulouse*
	A. RACHID	Professeur *Université de Picardie-Jules-Verne, Amiens*
Examinateurs	J. -P. RICHARD	Professeur *École Centrale de Lille (EC-Lille)*
	V. KONCAR	Professeur *École Nationale Supérieure des Arts et Industries Textiles (ENSAIT), Roubaix*

à Rehana, ma femme…

mes sincères remerciements s'adressent…

D'abord à tous les membres du jury, pour l'attention qu'ils ont portée à ce mémoire, particulièrement au Directeur de Recherche CNRS Jacques BERNUSSOU (LAAS, Toulouse) et au Professeur Ahmed RACHID (Université de Picardie, Amiens) pour avoir accepté de rapporter sur ce mémoire de doctorat.

Au Professeur Christian VASSEUR (LAGIS, USTL), en tant que directeur, que collaborateur et surtout à titre personnel, pour son accueil, son dévouement, ses conseils et son soutien tout au long de mon parcours de doctorant. Sa contribution m'a été très bénéfique dans le lancement de ma carrière de recherche et d'enseignement. Travailler avec lui est un réel plaisir. Je n'oublierai jamais les moments passés en sa compagnie que ce soit de nature professionnelle ou personnelle.

Au Professeur Nicolai CHRISTOV (LAGIS, USTL), qui a toujours du temps à consacrer pour partager mes problèmes et apporter des conseils.

Au Professeur Jean-Pierre RICHARD (LAGIS, ECL) pour les discussions enrichissantes lors des réunions avec les membres de l'équipe SyNeR, pour m'avoir sollicité à faire des présentations et des collaborations et, à titre personnel, pour les moments agréables passés ensemble.

Au Professeur Vladan KONCAR (GEMTEX-ENSAIT) qui a contribué énormément dans l'avancement du travail scientifique comme collaborateur et qui m'a toujours encouragé lors de cette expérience de doctorat.

À Olivier LOSSON (Maître de Conférences, LAGIS, USTL), de m'avoir accueilli et soutenu dès le début de mon doctorat et d'être toujours quelqu'un sur qui compter.

Aux doctorants du LAGIS avec qui j'ai partagé cette formidable aventure de doctorat. Je pense particulièrement à Haoping WANG et Alexandre SEURET pour les nombreuses collaborations et les bons moments passés en leur compagnie, sans oublier Habiboulaye BOUBACAR pour son soutien et ses encouragements réguliers.

À tous mes collègues permanents du LAGIS avec qui je partage une vie sociale chaleureuse. Je pense notamment à Mesdames Annick PIGNON et Patricia DEWYNTER, à Messieurs Frédéric DURAK, Philippe HENIN et Daniel FARGUE, aux responsables de la direction, particulièrement au Professeur Olivier COLOT et à tout le corps des enseignants–chercheurs.

Aux membres du conseil de laboratoire que je rencontre mensuellement depuis la deuxième année de ma thèse.

Aux étudiants et aux stagiaires qui m'ont permis d'affirmer mes compétences pédagogiques : un sujet est toujours mieux « maîtrisé » par soi quand on arrive à le transmettre à autrui avec succès.

À mes parents sans qui je n'aurais pas pu faire mes études universitaires en France.

À Rehana, qui me soutient avec amour, même quand je travaille …la nuit.

À tous mes amis qui m'ont motivé à aller jusqu'au bout de cette aventure.

Table des matières

Chapitre 1

Introduction générale 1

Chapitre 2

Les systèmes hybrides et les systèmes à fonctionnement par morceaux 9

Chapitre 3

Identification : « clonage » par les SFM 35

Chapitre 4

La poursuite échantillonnée par les SFM 61

Chapitre 5

Adaptation des contrôleurs à fonctionnement par morceaux 109

Annexe I

La plate-forme 2D à retour visuel 159

Annexe II

Réalisation des SFM sous Simulink® 163

Annexe III

Réalisation du clonage 167

Annexe IV

Conditions d'existence de la poursuite 171

Annexe V

Bibliographie personnelle 179

Références bibliographiques 181

Avant-propos

Ce mémoire représente l'aboutissement de trois ans et demi de travail de recherche effectué au Laboratoire d'Automatique, Génie Informatique & Signal, LAGIS (UMR CNRS 8146) – Université des Sciences et Technologies de Lille, USTL, sous la direction du Professeur Christian VASSEUR. Les résultats scientifiques sont le fruit d'une collaboration avec ce dernier et plusieurs autres chercheurs et ont fait, par ailleurs, l'objet de plusieurs publications.

Je suis honoré de pouvoir présenter ce manuscrit et d'apporter ainsi, une modeste contribution à la recherche dans des domaines vivants de l'automatique. En effet, même si beaucoup de concepts et de théories fondamentales ont déjà été présentés par nos prédécesseurs, il y a toujours de nouvelles problématiques à considérer et des adaptations à apporter aux théories existantes afin de répondre à des besoins d'actualité, tels que l'intégration de la nouvelle technologie dans le domaine de la commande.

Ce mémoire propose de réaliser l'identification et la commande de processus réels par une nouvelle approche : l'utilisation d'une classe de système hybride. Il n'a pas la prétention de révolutionner le monde de l'automatique, mais je serais heureux s'il pouvait satisfaire quelque besoin, et surtout ouvrir des perspectives de recherche.

D'un point de vue plus personnel, cette aventure du doctorat, comprenant la recherche, l'enseignement, les responsabilités au laboratoire et surtout la rédaction des articles et de ce mémoire, m'a apporté rigueur, méthode et organisation qui jouent positivement sur mon quotidien. J'espère pouvoir cultiver ces qualités tout au long de ma vie et les transmettre à mon entourage.

Symboles et notations

Sauf mention contraire dans le texte, nous utiliserons les notations et symboles suivants pour désigner les grandeurs correspondantes :

$u(t) \in U^r$	vecteur d'entrée
$x(t) \in \Sigma^n$	vecteur d'état
$y(t) \in Y^m$	vecteur de sortie
$w(t)$	vecteur d'observation
$\xi(t)$	vecteur d'état retardé
$\xi^*(t)$	vecteur d'état retardé et échantillonné
$z(t)$	vecteur de sortie retardée
$z^*(t)$	vecteur de sortie d'un capteur introduisant un retard et un échantillonnage
a, b, c	paramètres du METC d'un processus
A^T	transposé d'une matrice ou vecteur A
$\hat{x}(t)$	estimateur du vecteur d'état $x(t)$
$e(t)$	erreur entre l'estimée $\hat{x}(t)$ et l'état $x(t)$ lui-même
$\hat{a}$	estimée d'une variable a
S	ensemble discrétisé d'instants de commutation tel que $S = \{t_k, k = 0,1,2...\}$
T	période de commutation constante
t_e	période d'échantillonnage liée aux capteurs numériques
R	retard lié aux capteurs numériques
I_n	matrice d'identité d'ordre n
$\otimes$	produit de Kronecker
τ	constante de temps
p	variable de Laplace
$\Sigma_c(.)$	représentation symbolique d'un SLCM
$\Sigma_d(.)$	représentation symbolique d'un SLBE

$\Xi(.)$	représentation symbolique d'un *amortisseur*
R-BOZ	bloc constitué d'un Retard & d'un BOZ ($T_R = T_{BOZ}$)

Abréviations et acronymes

ARMA	AutoRegressive Moving Average
AViVA	Attelage Virtuel pour Véhicules Autonomes
BOZ	Bloqueur d'Ordre Zéro
CBE	Contrôleur Bi-Échantillonné
CCD	Charged Couple Device
CCM	Contrôleur Continu par Morceaux
CFM	Contrôleur à Fonctionnement par Morceaux
CI	Condition(s) Initiale(s)
DVS	Décomposition en Valeurs Singulières
ECL	École Centrale de Lille
ENSAIT	École Nationale Supérieure des Arts et Industries Textiles
GEMTEX	GEnie & Matériaux TEXtiles
GRAISyHM	Groupement de Recherche en Automatisation Intégrée et Systèmes Hommes-Machines
GSHF	Generalised Sample data Hold Function
ID	Ingénierie de la Décision
IR	Infra Rouge
LAAS	Laboratoire d'Analyse et d'Architecture des Systèmes
LAGIS	Laboratoire d'Automatique, Génie Informatique & Signal
LED	Light Emitting Diode
LMI	Linear Matrix Inequalities
METC	Modèle d'État à Temps Continu
MIMO	Multi Input Multi Output
PDL	Prise de Décision Logique
PID	Proportionnel, Intégrale, Dérivée
PWA	PieceWise Affine

SBE	Systèmes Bi- Échantillonnés
SCM	Systèmes Continus par Morceaux
SFC	Structure Fonctionnel de Clonage
SFM	Systèmes à Fonctionnement par Morceaux
SISO	Single Input Single Output
SLBE	Systèmes Linéaires Bi-Échantillonnés
SLCM	Système Linéaire Continu par Morceaux
SyNeR	Systèmes Non linéaires et à Retard
TAT	Technologies Avancées pour les Transports
TC	Temps Continu
TD	Temps Discret
TVP	Time Varying Parameters
UMR	Unité Mixte de Recherche
USTL	Université des Sciences et Technologies de Lille

…la plus intelligente est celle qui sait qu'elle ne sait pas…

SOCRATE

Chapitre 1

Introduction générale

Ce doctorat a été préparé au sein de l'équipe ID (Ingénierie de la Décision) du Laboratoire d'Automatique, Génie Informatique & Signal (LAGIS, UMR CNRS 8146). Le travail de recherche s'inscrit dans le cadre du projet AViVA (Attelage Virtuel pour Véhicules Autonomes), qui, lui-même, se situe dans le programme TAT[1] (Technologies Avancées pour les Transports) du contrat de plan état – région Nord – Pas de Calais 2002-2006. Le projet AViVA concerne le développement des concepts de routes et de véhicules intelligents afin d'optimiser la circulation routière, améliorer la sécurité, préserver l'environnement et proposer de nouveaux services aux usagers. Ce projet est soutenu par 10 laboratoires et organismes de recherche de la région Nord – Pas de Calais appartenant à la fédération de recherche GRAISyHM (Groupement de Recherche en Automatisation Intégrée et Systèmes Hommes-Machines).

La recherche proposée concerne le développement de nouvelles lois de commande robuste utilisant un capteur « immatériel » (par exemple un système de vision) dans la boucle de rétroaction. Elle est illustrée sur une plate-forme réelle[2] d'asservissement en position par retour visuel. Ce type d'asservissement pourra ensuite être intégré dans des projets tels que le « pendule inverse à retour visuel » qui consiste à maintenir debout un pendule en mesurant son angle de chute par un système de vision artificielle et l'« attelage virtuel » qui consiste à commander un véhicule « esclave » de sorte qu'il suive un véhicule « meneur » observé par vision artificielle.

[1] Ce programme est soutenu par la région Nord – Pas de Calais, l'état et la communauté européenne (contrat 15010/02Y0064/03-04 CAR/Presage N° 4605 Obj. 2-2004:2 - 4.1 - N° 160/4605).
[2] Description en annexe I.

1.1. Motivations

Les *systèmes* peuvent être définis comme des assemblages d'objets interconnectés. Ils apparaissent dans de nombreux domaines tels que la mécanique, la chimie, l'économie, la société, etc. L'étude des systèmes au sens large et de leur contrôle suscite la curiosité de la communauté des *automaticiens*. L'idée de *commander* ou d'*asservir* un système peut être motivée par des besoins de confort, de sécurité ou de performance. Dans cette optique, l'automaticien utilise le concept de bouclage de la sortie d'un système sur son entrée (*feedback*), qui est l'un des concepts fondamentaux de l'ingénierie. Il est à souligner que cette notion existe dans les systèmes naturels tels que les être vivants qui utilisent leurs *capteurs* (les cinq sens chez l'homme) afin de déterminer le comportement à adopter pour atteindre un objectif, compte tenu de l'environnement. Selon [Ben96], le principe de feedback date de plus de 2000 ans. L'une des premières applications connues est l'amélioration de l'horloge à eau (clepsydre), décrite par Vitruvius et attribuée à Ktesibios (env. 270 avant J.C), par un système de flotteur qui régule le débit.

Le siècle dernier a été témoin de formidables avancées technologiques. Plusieurs domaines tels que l'automobile, l'aéronautique ou encore la médecine se sont appropriés des outils de plus en plus sophistiqués pour répondre à des besoins de plus en plus exigeants. C'est ainsi que, du simple système de chasse d'eau régulant *automatiquement* le niveau d'un réservoir, on aboutit aujourd'hui au *contrôle automatique* qui permet de guider des missiles autonomes vers une cible en mouvement.

Pour bien contrôler un système, il est nécessaire de l'analyser et de comprendre son fonctionnement. Cette connaissance provient souvent d'une phase d'identification effectuée préalablement à la commande. L'enjeu consiste à définir un *modèle* mathématique et paramétrique qui décrit au mieux le comportement du système, dans le contexte de son utilisation. Ce modèle permet alors de déterminer la loi de commande la mieux adaptée en fonction de l'objectif visé.

Grâce aux développements technologiques, l'automatique dispose de moyens de plus en plus élaborés pour la réalisation des lois de commandes. Toutefois, pour prendre en compte certaines contraintes induites par les nouvelles technologies, il est nécessaire d'adapter les théories mathématiques existantes. Un premier exemple est celui des calculateurs numériques, qui ont été d'un apport considérable pour la réalisation des systèmes de commande, mais qui, en même temps, ont introduit une contrainte supplémentaire, à savoir la nécessité de raisonner directement dans le domaine du temps *discret* [AH00]. Cette contrainte a soulevé la problématique de la commande numérique nécessitant l'adaptation des théories continues classiques au temps discrétisé [Lan88].

L'évolution de la technologie a également conduit au développement de capteurs basés sur des composants numériques. Bien souvent gouvernés par des microprocesseurs, ces capteurs qui sont alors dits *intelligents* [Zay05] introduisent par nature une discrétisation des données. De plus, le temps de calcul nécessaire à l'élaboration des données (par exemple un traitement d'image) peuvent induire des retards non négligeables dans la délivrance des données [BHS96].

Dans la famille des capteurs intelligents à retard et échantillonnage, on trouve notamment les capteurs dits *immatériels*[3]. Utilisés dans des systèmes de commande de processus réels, ceux-ci permettent de reproduire artificiellement les facultés naturellement présentes chez les créatures vivantes. Dans le cadre de la robotique mobile, par exemple, il est souvent question de faire évoluer des entités mobiles de manière autonome dans un environnement encombré d'obstacles, détectables par une *vision artificielle* stéréoscopique ou par capteurs à *ultrasons*. Il convient alors de s'inspirer de la faculté de l'homme et de la chauve-souris à se repérer dans l'espace 3D pour concevoir des capteurs appropriés. Désormais, l'implantation des capteurs immatériels devient de plus en plus courante dans divers domaines. Ils permettent, en outre, de faire des mesures à distance sans encombrer les parties délicates d'un processus. Nous trouvons, par exemple, dans les chaînes de production, des mesures de distance par télémètre laser, des contrôles de fabrication par imagerie, etc. La localisation d'un mobile par GPS constitue un autre exemple de mesure sans contact.

1.2. Problématique

Dans ce mémoire, nous traitons de manière générale des processus réels qui sont supposés linéaires continus destinés à être commandés, en temps réel, par calculateur numérique. Dans ce sens, afin de généraliser l'étude à des systèmes MIMO, nous utilisons le *modèle d'état* qui est un outil de prédilection pour décrire les systèmes et leur évolution. Nous décrivons donc le processus par :

$$x'(t) = a.x(t) + b.u(t), \tag{1.1a}$$

$$y(t) = c.x(t). \tag{1.1b}$$

Ici, $a \in \Re^{n\times n}$, $b \in \Re^{n\times r}$ et $c \in \Re^{m\times n}$ représentent les matrices usuelles de la représentation d'état. Par ailleurs, $u(t) \in U^r$, $x(t) \in \Sigma^n$ et $y(t) \in Y^m$ désignent respectivement l'entrée, l'état et la sortie du processus considéré.

[3] Un capteur *immatériel* mesure une grandeur sur un objet sans rentrer en contact avec celui-ci.

Dans certain cas, nous considérerons que la sortie $y(t)$ du système réel n'est pas accessible directement, mais par un capteur numérique délivrant son information sous forme retardée (de R) et échantillonnée (de t_e). La sortie du capteur, notée $z(t)$ est défini par :

$$z(t) = y^*(t - R), \tag{1.2}$$

(*) représentant un échantillonnage de période t_e.

La problématique générale est résumée figure 1.1, où on peut distinguer schématiquement sur la figure 1.1a le processus linéaire possédant une dynamique continue et le capteur qui introduit une discrétisation en sus du retard. Les signaux de la figure 1.1b explicite l'effet du capteur. De ce fait, seule la sortie du capteur fait office de feedback dans l'architecture de commande.

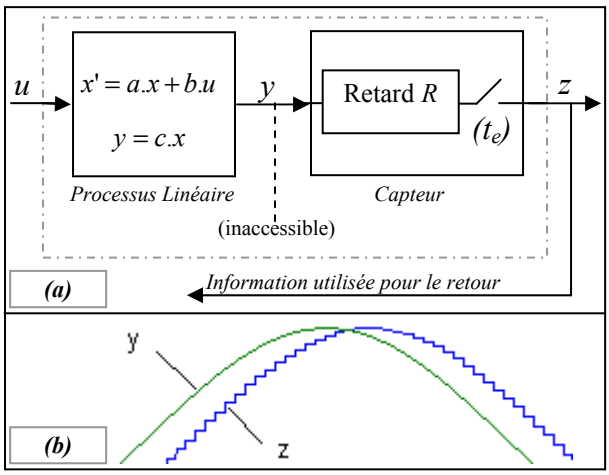


Fig. 1.1a: *L'ensemble processus–capteur*
Fig. 1.1b: *Les sorties processus/capteur*

En guise d'illustration de cette problématique, nous disposons d'une plate-forme réelle permettant l'asservissement en position d'un chariot dont la position est *observée* par un système de vision. Le montage de cet asservissement, présenté figure 1.2, découle d'une configuration particulière de la maquette détaillée en annexe I.

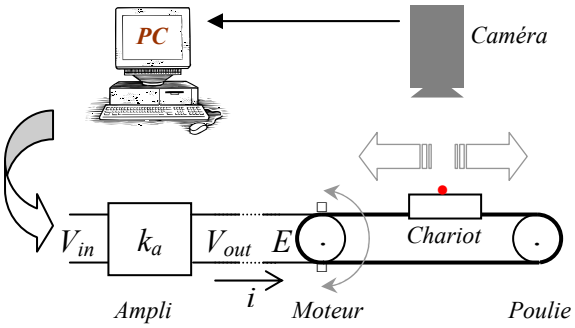


Fig. 1.2. *Asservissement visuel du chariot*

Sur la figure 1.2, l'ensemble amplificateur–moteur–chariot correspond au processus linéaire continu décrit par le système d'équations (1.1). Le système de vision (caméra–PC), qui délivre une information échantillonnée–bloquée $z(t)$ sur la sortie $y(t)$ (position) du processus, est le capteur régi par l'équation (1.2). Dans un tel cas, nous pouvons considérer que l'échantillonnage à période t_e correspond à la délivrance périodique des trames d'images avec t_e représentant la durée d'une prise d'image. Quant au retard R, il représente le temps de calcul nécessaire au traitement des images. Dans ce cas, $R = Nt_e$ avec N entier positif, en faisant l'hypothèse qu'il faut traiter N images pour obtenir l'information requise.

Nous nous proposons de développer de nouvelles techniques d'identification en ligne et de commande dans le but de réaliser, en temps réel, la poursuite échantillonnée d'une consigne par l'état du processus réel avec comme seule information la sortie du capteur numérique.

1.3. Approche

Il est évident qu'il existe déjà de nombreuses méthodes qui traitent de l'identification et de la commande des procédés linéaires. Notre problématique, qui fait apparaître une dynamique continue et une mesure discrète, suggère de l'aborder par une approche *échantillonnée*. Au-delà, nous proposons de l'aborder par le biais d'une classe particulière de systèmes hybrides : les systèmes à fonctionnement par morceaux (SFM). Ces systèmes, qui font appel à des espaces de temps et d'entrées multiples discrets et continus, sont caractérisés par des sauts de leur état à des instants discrets dits *instants de commutation* et par une évolution continue entre ces instants. Les SFM sont, de par leur nature, particulièrement bien adaptés à la technologie numérique, surtout quand il s'agit de traiter des données discrétisées comme dans le cas des capteurs présentés précédemment.

1.4. Contributions

Dans le sens d'une approche par les SFM, nous proposons, dans ce mémoire :

- un état de l'art des systèmes hybrides et particulièrement de la classe des SFM,
- une méthode hybride d'identification rapide en ligne,
- une méthode hybride de commande temps réel des processus linéaires MIMO,
- l'adaptation de la commande précédente en vue de traiter le cas où la seule observation disponible sur l'évolution du processus provient d'un capteur délivrant une information retardée et échantillonnée,

Il est important de préciser que ce mémoire ne vise pas à réaliser l'identification ou le contrôle des systèmes hybrides eux-mêmes, mais bien à utiliser les SFM pour identifier et commander des processus linéaires continus ou échantillonnés et intégrant éventuellement un retard. De ce fait, même si nous faisons mention de systèmes naturellement hybrides dans le texte, les systèmes hybrides que nous utilisons sont purement artificiels. Ils sont conçus pour être facilement implantés en temps réel sur calculateurs numériques, selon une procédure systématique, que nous avons développée.

1.5. Organisation du document

Sachant que les SFM constituent la base des méthodes développées, nous proposons au ***chapitre 2*** un état de l'art des systèmes hybrides, l'accent étant mis sur les SFM.

Les ***chapitres 3*** à ***5*** seront consacrés, quant à eux, aux développements des méthodes d'identification et de commande utilisant sur les SFM. Traitant des domaines vivants de l'automatique, chacun de ces chapitres offre un état de l'art pour situer la méthode par rapport l'existant. Les chapitres incluent une formalisation mathématique par l'approche des SFM et des exemples de simulation et d'implantation temps réel à l'aide de la technologie Matlab®-Simulink/dSpace®. La figure 1.3 donne le schéma d'articulation des différents chapitres.

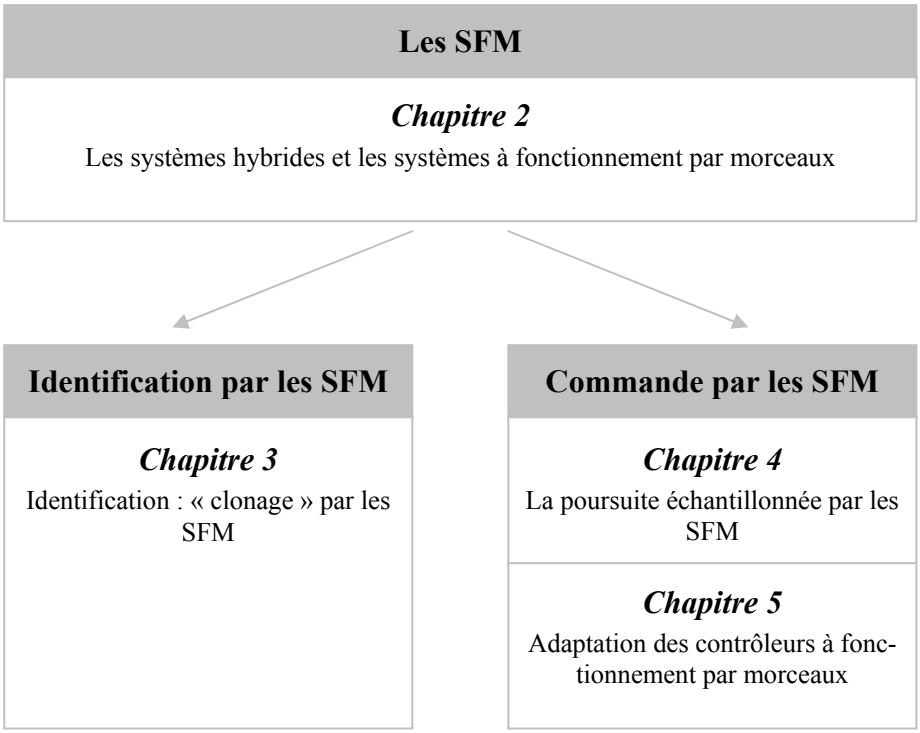


Fig. 1.3. *Organisation du document*

Enfin, le ***chapitre 6*** fournit une conclusion générale ainsi que quelques pistes de développement.

Chapitre 2

Les systèmes hybrides et les systèmes à fonctionnement par morceaux

La problématique présentée dans l'introduction générale n'a rien d'original en elle-même. En effet, de nombreux travaux ont déjà été publiés sur l'identification des processus linéaires MIMO continus et sur la commande de systèmes à retard. Par contre, l'approche que nous proposons pour répondre à cette problématique est assez particulière, car elle fait appel à des systèmes possédant des propriétés hybrides. Dans ce sens, ce chapitre est consacré à un état de l'art sur les systèmes hybrides et à la présentation d'une classe particulière de tels systèmes : les systèmes à fonctionnement par morceaux (SFM). Les SFM sont utilisés pour développer les méthodes d'identification en ligne et de commande présentées dans les chapitres 3 à 5.

2.1. Systèmes hybrides : généralités

2.1.1. Les systèmes à commutation

Les méthodes proposées dans ce mémoire se basent sur une approche par systèmes à fonctionnement par morceaux définis en détail à la fin de ce chapitre. Ces systèmes sont caractérisés par une *commutation* dans leur fonctionnement interne. Selon [Sun04], un système à commutation peut être considéré comme un système *hybride* formé de plusieurs sous-systèmes munis d'une règle qui gère la commutation de l'un à l'autre. L'auteur précise d'une part que la dynamique continue des sous-systèmes régit le comportement local du système, et que, d'autre part, le mécanisme de commutation détermine sa performance globale. En ce sens, les systèmes à commutation fournissent une formalisation générique des systèmes dynamiques conventionnels, des systèmes intelligents, etc. Plusieurs systèmes dans l'ingénierie, tels que les réseaux d'ordinateurs, les systèmes de routage automatique ou d'électronique de puissance peuvent être représentés par des systèmes à commutation. Récemment, plusieurs ouvrages [EH89, TE98, BT99, LM99, DBPL00, XW03] ont été consacrés à ce type de systèmes, en y apportant des approches et des analyses personnalisées. Dans la suite de la section 2.1, nous tentons de préciser la notion de système hybride.

2.1.2. Qu'est-ce qu'un système hybride ?

En théorie des systèmes, le qualificatif *dynamique* sous-tend l'idée d'activité et de changement en fonction du temps. Selon que l'on se réfère à un espace temps continu ou à un espace temps discrétisé, on parle de systèmes dynamiques continus ou de systèmes dynamiques discrets ou échantillonnés. Les systèmes dynamiques hybrides sont, quant à eux, caractérisés par l'interaction entre une dynamique (continue ou échantillonnée) et des événements discrets [SAL96, Tho04]. Ces systèmes contiennent des variables ou signaux prenant des valeurs continues (ou échantillonnées) et également d'autres variables associées aux événements discrets prenant leurs valeurs dans un espace discret fini [AKZ98].

L'évolution des variables de la dynamique continue ou échantillonnée est régie par des équations différentielles ou aux différences. Les événements discrets sont gouvernés, quand à eux, par des lois logiques (décrites par exemple par logique séquentielle, automate, lois si-alors-sinon, etc.) ou encore par des composants discrets (boutons poussoirs, valves, sélecteurs de vitesse, etc.) [BBBM05]. Ainsi, un système hybride présente plusieurs modes de fonctionnement avec des règles spécifiques pour commuter d'un mode à l'autre. Selon [Ant00], la transition entre les modes est provoquée par des variables pas-

sant par des seuils spécifiques (événements d'état), par des variables proportionnelles au temps (événements temporels) ou encore par des entrées externes (événements d'entrée).

Il existe de nombreux exemples de systèmes présentant une nature hybride : la boîte de vitesse d'une voiture, les chaînes de production contrôlées par les automates programmables industriels, les réseaux de communication. Un autre exemple plus *ludique* est celui de la balle de tennis (ou de football), au cours d'un match. Dans ce cas, la position de la balle et sa vitesse constituent l'état d'un système à impulsions, qui subit des sauts aux instants de frappe (ou de rebond) et qui évolue selon une dynamique continue ailleurs. Il est évident que l'approche hybride fournit des outils adaptés pour la modélisation et/ou la commande de tels systèmes.

Même si les systèmes hybrides vus sous cet angle existent à l'état naturel, les automaticiens s'intéressent de plus en plus au formalisme hybride, non seulement pour modéliser des systèmes hybrides existants, mais aussi pour concevoir et construire des systèmes hybrides artificiels pouvant eux-mêmes être utilisés pour identifier et/ou commander d'autres systèmes.

Par exemple, un système non linéaire peut être modélisé par un ensemble de systèmes linéaires évoluant chacun dans une zone de fonctionnement spécifique. Il suffit alors de *commuter* judicieusement entre ces modèles linéaires pour atteindre un modèle global pouvant décrire la dynamique du système non linéaire [AKZ98]. Ce principe a déjà fait l'objet d'investigations chez les automaticiens. Dans [RDI05], les auteurs introduisent la notion de *phase* pour décrire l'évolution d'un système hybride. Selon eux, des trajectoires hybrides peuvent être considérées comme un assemblage de phases. De ce point de vue, une phase correspond à une portion de trajectoire où les équations différentielles restent inchangées. Ils considèrent alors que la commutation entre différentes phases est régie par des règles à définir suivant des contraintes établies.

Il existe de nombreux systèmes de commande basés sur le contrôle de variables continues et de tests logiques permettant de cerner le comportement du processus à piloter et de déterminer les algorithmes de commande en conséquence. Cette combinaison donne lieu à une commande hybride. Dans ce sens, l'intégration croissante des calculateurs numériques dans le domaine de la commande a donné lieu à plusieurs ouvrages [KJN+95, KNR96] dédiés à une approche hybride en ce qui concerne la commande de processus réels (souvent continus ou continus par morceaux) par ordinateur (domaine discrétisé). Les auteurs de [HCCS02] soulignent que lors du contrôle de procédés industriels, la commande et le filtrage numériques associés au traitement de signaux continus rendent l'ensemble du système bouclé hybride, de telle sorte que son état subit des impulsions successives à des instants particuliers.

Toutefois, même si le développement de lois de commande par une approche hybride a été motivé par la technologie numérique, cette stratégie apporte des propriétés nouvelles, comme par exemple une meilleure stabilisation, pour améliorer la qualité des systèmes de contrôle/commande. Ainsi, les automaticiens cherchent à développer des contrôleurs hybrides [KV02, KV03] même pour commander des procédés continus.

Les deux premiers modèles hybrides proposés sont ceux de Witsenhausen [Wit66] et de Tavernini [Tav87]. Depuis, la modélisation des systèmes hybrides a connu une évolution notoire avec la contribution de [PD88, Pel92, ACHH93, GNRR93, ACH$^+$95, Bra95, LTEP96, SAL96, BBM98, Pet99]. Dans ses travaux, Witsenhausen [Wit66] définit les systèmes hybrides comme une classe de systèmes possédant un état partiellement continu et partiellement discret. Cet état est décrit par des équations différentielles combinées avec des éléments « multistables ». Selon lui, la transition des éléments entre les états discrets est conditionnée par des triggers provenant de la partie continue de l'état et non pas directement par les entrées du système.

Plusieurs autres types de systèmes hybrides ont été conçus plus récemment [BGM93, NK93, BBM94]. Dans la plupart des cas, les systèmes hybrides incorporent des blocs à temps continu (TC) et/ou à temps discret (TD) associés à des procédés logique ou de décision introduits par Decarlo [BM99a, BM99b, DBPL00]. Les composants TC/TD peuvent correspondre à des équations différentielles ou aux différences ou alors à des modèles continus ou discrets. Les blocs de prise de décision logique (PDL) peuvent quant à eux se référer à des automates finis ou alors à des systèmes à événements discret au sens large [Bra98, DBPL00]. Les procédés TC/TD influent sur la transition des états des PDL et dans l'autre sens, les PDL jouent sur la dynamique des procédés TC/TD [YMH98, MH99].

Le paragraphe suivant (2.2) fournit une classification des systèmes hybrides, selon Branicky [BBM94, BBM98] qui a proposé une formulation unitaire du concept hybride.

2.2. Classification de Branicky

De manière générale, la dynamique d'un système hybride peut être représentée par une équation différentielle dépendant d'un phénomène discret :

$$x'(t) = \zeta(t), \quad t \geq 0 \tag{2.1}$$

Dans cette équation, $x(t)$ représente la composante continue de l'état et prend ses valeurs dans un sous-espace de l'espace Euclidien. Par ailleurs, $\zeta(t)$ est un champ de vecteurs qui dépend généralement :

- de $x(t)$,
- de la composante continue $u(t)$ de la commande,
- du phénomène discret mentionné précédemment.

La taxonomie proposée par Branicky [Bra95, Bra98] en fonction des phénomènes discrets est représentée par la figure 2.1.

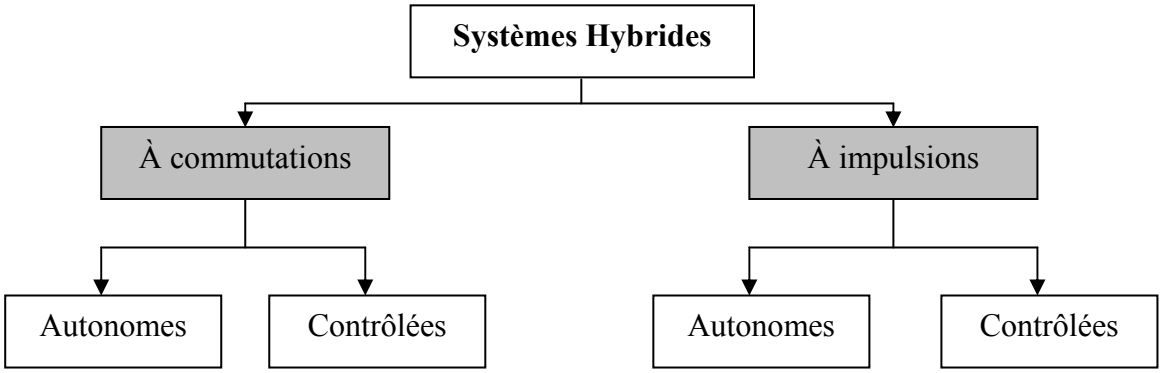


Fig. 2.1. *Taxonomie des systèmes hybrides*

Les principales caractéristiques des différents cas envisagés sont résumées dans le tableau 2.1 ci-dessous :

Commutations autonomes (endogènes)	Le champ de vecteurs $\zeta(t)$ commute quand l'état $x(.)$ traverse des frontières prédéfinies de l'espace d'état [BBM94].
Commutations contrôlées (exogènes)	Le champ de vecteurs $\zeta(t)$ commute en réponse à une loi de commande [Zab73].
Impulsions autonomes (endogènes)	L'état continu $x(.)$ change impulsivement (saut) lorsqu'il atteint certaines zones prédéfinies de l'espace d'état [BGM93, BS89].
Impulsions contrôlées (exogènes)	L'état continu $x(.)$ change impulsivement (saut) en réponse à une loi de commande [BL84].

Tab. 2.1. *Les différents types de systèmes hybrides (taxonomie de Branicky)*

Cette taxonomie conduit aux modèles mathématiques définis ci-après :

Définition 2.1. *Un système hybride à* ***commutation autonome*** *est défini par :*

$$x'(t) = \zeta(x(t), q(t)), \tag{2.2a}$$

$$q^+(t) = v(x(t), q(t)). \tag{2.2b}$$

Avec $x(t) \in \Re^n$ *et* $q(t) \in Q = \{1,...,N\}$. *Chaque* $\zeta(.,q(t)) : \Re^n \to \Re^n$ *,globalement Lipschitz-continu, représente la dynamique continue de l'équation (2.2a) pour la valeur de q(t) définie par l'équation (2.2b)* $v : \Re^n \times Q \to Q$ *, où* $q^+(t)$ *est le successeur de* $q(t)$.

Définition 2.2. *Un système hybride à* ***commutation contrôlée*** *est défini par :*

$$x'(t) = \zeta(x(t), q(t), u(t)), \tag{2.3a}$$

$$q^+(t) = v(x(t), q(t), u(t)). \tag{2.3b}$$

Avec les mêmes notations qu'en définition 2.1, sauf que, ici, $u(t) \in \Re^r$ *représente la commande.* ζ *et* v *sont modifiés en conséquence.*

Le modèle de Witsenhausen [Wit66] en est un exemple.

Définition 2.3. *Un système hybride à* ***impulsion autonome*** *est défini par :*

$$x'(t) = f(x(t)),\ x(t) \notin M \tag{2.4a}$$

$$x^+(t) = J(x(t)),\ x(t) \in M \tag{2.4b}$$

Avec $x(t) \in \Re^n$, $J : \Re^n \to \Re^n$ *et* $M \subset \Re^n$.

Définition 2.4. *Un système hybride à* ***impulsion contrôlée*** *est défini par :*

$$x'(t) = f(x(t), u(t)),\ x(t) \notin M \tag{2.5a}$$

$$x^+(t) = J(x(t), u(t)),\ x(t) \in M \tag{2.5b}$$

Avec les mêmes notations qu'en 2.3, sauf que, ici, $u(t) \in \Re^r$. *f et J sont modifiées convenablement.*

Remarque 2.1. *Ces quatre types peuvent être combinés pour donner lieu à des systèmes hybrides plus complexes.*

2.3. Quelques modèles hybrides spécifiques

2.3.1. Prise en compte d'un retard (Hu et al., 2002)

Les systèmes hybrides proposés dans [HCCS02] sont du type à *impulsions autonomes*. Les auteurs proposent en outre d'adapter la modélisation pour tenir compte d'un retard Ils prennent comme exemple les réseaux de communication, les circuits électriques, etc. Leur définition découle d'une mise en équations proposée par [Hal77] :

$$x'(t) = f(t, x(t-\theta)),\ t \in]t_k, t_{k+1}], \tag{2.6a}$$

$$x(t_k^+) = I_k(x(t)),\ k = 0,1,2,... \tag{2.6b}$$

où θ est un retard tel que $0 < \theta < r$.

Dans le système d'équations (2.6), r est un réel positif connu. Enfin, f est une fonction bornée et continue à droite, définie sur $]t_k, t_{k+1}]$ et I_k est telle que $I_k : \Re^n \to \Re^n$, $k = 0,1,2,....$ Il est à souligner que les instants $\{t_k, k = 0,1,2,...\}$ correspondent aux commutations de l'état.

2.3.2. Systèmes à impulsions (Haddad et al., 1999)

Les travaux présentés dans [HCK99, HCK01, Kab03a] décrivent une classe de systèmes hybrides (dits systèmes à impulsions) caractérisés par des impulsions singulières ou généralisées. Un système à impulsions est décrit par :

- une équation différentielle continue qui gouverne son évolution entre les événements responsables de la commutation,
- une équation aux différences qui définit le changement instantané de l'état au moment de l'événement,
- un critère déterminant la condition de commutation.

Soit, en formalisant :

$$x'(t) = f_c.x(t) + G_c(x(t))u_c(t),\ (t, x(t), u_c(t)) \notin S \tag{2.7a}$$

$$\Delta x(t) = f_d.x(t) + G_d(x(t))u_d(t),\ (t, x(t), u_d(t)) \in S, \tag{2.7b}$$

$$y_c(t) = h_c.x(t) + J_c(x(t))u_c(t),\ (t, x(t), u_c(t)) \notin S, \tag{2.7c}$$

$$y_d(t) = h_d.x(t) + J_d(x(t))u_d(t),\ (t, x(t), u_d(t)) \in S. \tag{2.7d}$$

L'ensemble des commutations est représenté par $S \subset [0,\infty[\times \Re^n \times U_c$.

L'équation (2.7a) correspond à la dynamique continue du système, avec :

- $f_c : D \to \Re^n$, Lipschitz-continue vérifiant $f_c(0) = 0$,
- $u_c(t) \in U_c \subseteq \Re^{m_c}$, commande continue,
- $G_c : D \to \Re^{n \times m_c}$, matrice de commande continue.

L'équation (2.7b) définit la loi régissant le saut de l'état aux instants de commutation, avec :

- $f_d : D \to \Re^n$,
- $u_d(t) \in U_d \subseteq \Re^{m_d}$, commande discrète,
- $G_d : D \to \Re^{n \times m_d}$, matrice de commande discrète.

Les équations (2.7c) et (2.7d) sont respectivement les équations de sorties continue et discrète.

À partir de ces définitions, les auteurs ont développé des méthodes de contrôle des systèmes à impulsions [Kab05a], même dans le cas non linéaire [HKCN05]. Par ailleurs, ils ont proposé une étude de stabilité des systèmes à impulsions non linéaires dans [Kab05b, Kab03b].

Kablar [Kab03a] utilise ce même formalisme pour définir des systèmes à commutation exogène, en admettant que les commutations se font à des instants discrets. Elle définit alors l'ensemble de commutation par $S = \{t_k, k = 1,2,...\}$, $t_{k+1} > t_k$ de sorte que les commutations se font à chaque t_k.

Nakura et Ichikawa [NI02] avaient introduit un formalisme similaire en 2002 pour décrire un système libre à sauts commandés, selon les équations :

$$x'(t) = f(x(t)), \tag{2.8a}$$

$$\Delta x(t_k) = f_d(x(t_k), u_d(t_k)), \tag{2.8b}$$

$$y(t_k) = h_d(x(t_k)). \tag{2.8c}$$

Ce formalisme peut se déduire de celui proposé par le système d'équations (2.7) avec $S = \{t_k, k = 1,2,...\}$ et $u_c(t) = 0$.

2.3.3. Systèmes multi modèles (Riedinger et al., 2005)

Les auteurs de [RDI05] associent à un ensemble fini d'états discret ($\overline{Q} = \{1,...,Q\}$), une collection de dynamiques continues définies par :

$$x'(t) = f_q(x(t),u(t),t),\ q \in \overline{Q}\ . \tag{2.9}$$

Dans l'équation (2.9), l'état continu $x(.) \in \Re^n$ $(n \in N)$, la commande continu $u(.) \in \Re^m$ $(m \in N)$ et le champ de vecteur f_q sont supposés définis et dérivables continûment sur $\Re^n \times \Re^m \times [t_1,t_2],\ \forall q \in \overline{Q}$, $[t_1,t_2]$ étant un intervalle de temps fini.

La dynamique discrète est définie grâce à une fonction de transition $\upsilon : \Re^n \times \overline{Q} \times \overline{D} \times [t_1,t_2] \to \overline{Q}$ de la forme :

$$q(t^+) = \upsilon(x(t^-),q(t^-),d(t),t), \tag{2.10}$$

$q(.)$ étant l'état discret ($q(t) \in \overline{Q}$) et $d(.)$ représentant la commande discrète ($d : [t_1,t_2] \to \overline{D}$ où $\overline{D} = \{1,...,D\}$ désigne un ensemble fini).

La variable discrète $q(.)$ est une fonction constante par morceaux. La valeur de la transition υ dépend de deux types de phénomènes qui peuvent altérer l'évolution de $q(.)$: des changements de la commande $d(.)$ et les conditions frontalières sur (x,t) de la forme $C_{(q,q')}(x,t) = 0$ qui modifient l'ensemble des états discrets atteignables.

Le modèle de système hybride présenté peut prendre en compte des événements autonomes et/ou contrôlés. Les auteurs donnent également une description d'un système à commutations contrôlées en exprimant sa dynamique par un champ de vecteur :

$$x'(t) = F(x,u,\alpha) = \sum_{q=1}^{Q} \alpha_q(t).f_q.(x(t),u(t)), \tag{2.11}$$

Ici, $\alpha(t)$ est un vecteur booléen ($\alpha(t) \in \{0,1\}^Q$) et $\alpha_q(t)$ représente la $q^{ème}$ composante de $\alpha(t)$ de sorte qu'une et une seule composante de $\alpha(t)$ soit égale à 1. Ainsi, $\alpha(t)$ joue le rôle de commande discrète.

Borrelli et al. [BBBM05] utilisent également une approche multi modèles pour modéliser les systèmes affines par morceaux (PWA). Pour cela, ils partitionnent l'espace d'état en régions polyédriques et associent à chaque région une dynamique différente [Son81, Son96]. Un système PWA est alors représenté par :

$$x(t+1) = A^i.x(t) + B^i.u(t) + f^i \text{ si } \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in \wp^i, \quad i = \{1,...,s\} \tag{2.12}$$

Dans cette équation, $x \in \Re^{n_c} \times \{0,1\}^{n_l}$, $u \in \Re^{m_c} \times \{0,1\}^{m_l}$ et $\{\wp^i\}_{i=1}^{s}$ est une répartition polyédrique de l'ensemble $\wp \subset \Re^{n+m}$ qui regroupe l'espace d'état et celui de l'entrée, avec $n = n_c + n_l$, $m = m_c + m_l$. Les composantes réelles de l'état et de l'entrée sont représentées par $x_c \in \Re^{n_c}$ et $u_c \in \Re^{m_c}$ respectivement.

Les systèmes PWA sont équivalents à l'interconnexion de systèmes linéaires avec des automates finis. Plusieurs ouvrages sont consacrés à ce type de système hybride [HSB01, Bem04, Imu04].

2.4. Systèmes à fonctionnement par morceaux

Depuis plusieurs années, des chercheurs du Laboratoire d'Automatique, Génie Informatique & Signal (LAGIS) de Lille [KV00, KV01, KV02, KV03] s'intéressent à une classe particulière de systèmes hybrides appelés systèmes à fonctionnement par morceaux (SFM).

Un SFM est un système strictement causal à dimension finie caractérisé par une commutation de son état en réponse à des impulsions contrôlées. Un tel système possède deux espaces d'entrées et se réfère à deux espaces temps.

Le premier espace temps constitue la référence de la dynamique du système en réponse au premier espace d'entrée. Le second espace temps, nécessairement discret, constitue la référence des événements discrets choisis pour forcer l'état du SFM aux valeurs imposées par la seconde entrée. L'espace temps discret détermine donc l'ensemble des instants de commutation de l'état.

Selon que la dynamique du SFM est continue ou échantillonnée entre les instants de commutation, on parle de systèmes continu par morceaux (SCM) [KV01, KV03] ou de systèmes bi-échantillonnés (SBE) [KV02].

La figure 2.2 résume les caractéristiques de ces deux différents types de SFM.

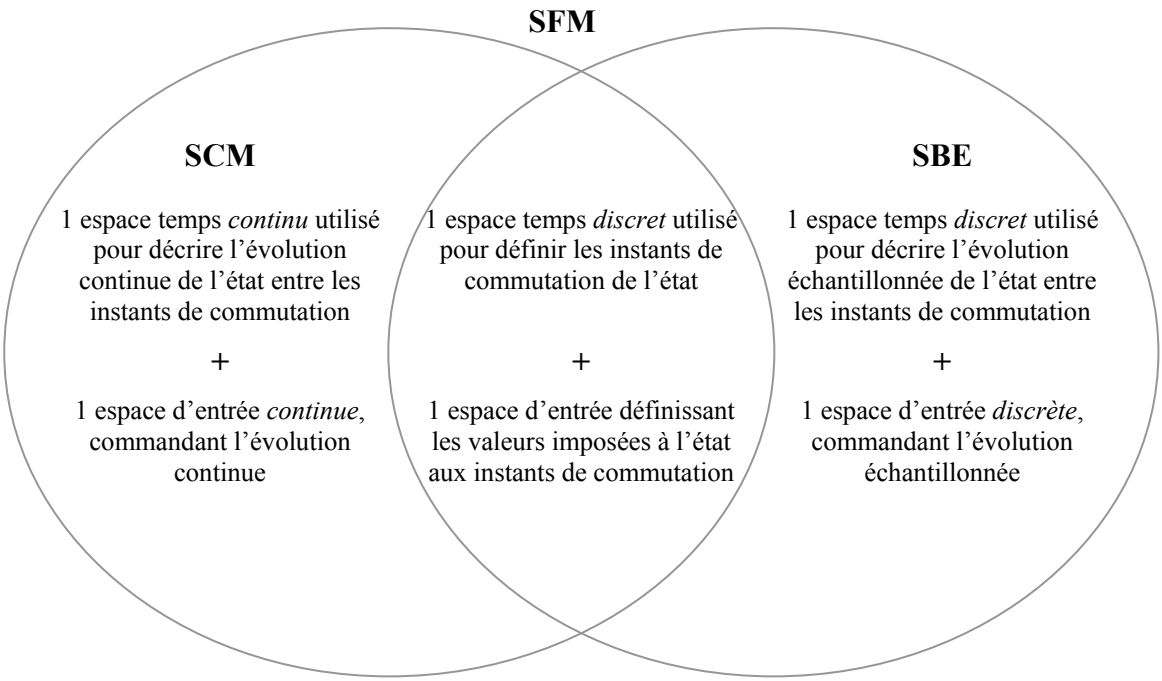


Fig. 2.2. *Les SCM et les SBE : deux types de SFM*

2.4.1. Formalisme des SFM

Dans le cas habituel, la représentation d'état des systèmes dynamiques utilise trois espaces vectoriels : l'espace d'état de dimension n, noté Σ^n, l'espace d'entrée de dimension r, noté U^r et l'espace de sortie de dimension m, noté Y^m. Si l'on note $\Im$ l'espace temps (continu ou discret) et respectivement $x(t) \in \Sigma^n$, $u(t) \in U^r$ et $y(t) \in Y^m$ l'état, l'entrée et la sortie en fonction du temps $t \in \Im$, alors la dynamique du système s'exprime par des transformations de la forme :

$$\Im \times \Im \times \Sigma^n \times U^r \to \Sigma^n : \quad x(t_l) = g(t_k, t_l, x(t_k), u_{]t_k, t_l]}), \tag{2.13a}$$

avec : $t_l > t_k$ et $u_{]t_k, t_l]}$ désignant l'entrée sur $]t_k, t_l]$,

et

$$\Im \times \Sigma^n \times U^r \to Y^m : \quad y(t_l) = h(t_l, x(t_l), u(t_l)). \tag{2.13b}$$

L'équation (2.13a), appelée *équation de transition d'état*, traduit la causalité et constitue la solution d'une équation différentielle ou d'une équation aux différences, selon que l'espace de temps utilisé est continu ou discret. L'équation (2.13b), appelée *équation de sortie* est, quant à elle, une équation sans mémoire. Ces équations font apparaître le premier espace temps $\Im$.

Pour décrire les phénomènes de commutation, les auteurs introduisent un second espace temps, noté S. De nature discrète, cet espace temps qui est un sous-espace de $\Im$ est appelé *espace de commutation*. Il est défini par :

$$S = \{t_k, k = 0,1,2,...\} \quad | \quad S \subset \Im \tag{2.14}$$

La valeur imposée à l'état aux instants de commutation est introduite par le biais du second espace d'entrée V^σ de dimension σ.

Si nous notons $v(t) \in V^\sigma$ la seconde entrée, il est possible de définir une transformation de la forme $S \times V^\sigma \to \Sigma^n$ afin de décrire le phénomène de saut :

$$x(t_k^+) = s(t_k, v(t_k)), \quad \forall t_k \in S, \tag{2.15}$$

avec $x(t_k^+) = \lim_{t \to t_k, t > t_k} \{x(t)\}$.

En rassemblant le système d'équation (2.13) et l'équation (2.15), nous pouvons définir un nouveau fonctionnement par le système d'équations (2.16) :

$$x(t_l) = g(t_k, t_l, x(t_k^+), u_{]t_k, t_l]}), \quad \forall t_l \in]t_k, t_{k+1}] \text{ et } \forall t_k \in S, \tag{2.16a}$$

$$x(t_k^+) = s(t_k, v(t_k)), \quad \forall t_k \in S, \tag{2.16b}$$

$$y(t) = h(t, x(t), u(t)), \quad \forall t \in \Im \setminus S. \tag{2.16c}$$

Ces équations définissent un fonctionnement par morceaux, chaque morceau étant un intervalle $]t_k, t_{k+1}]$ défini par deux instants de commutation successifs de S.

Remarque 2.2. *Il est aussi possible d'envisager un changement de dynamique à chaque instant de commutation, selon une représentation multimodèle. Dans ce cas, les fonctions $g(.)$, $s(.)$ et $h(.)$ deviennent $g_k(.)$, $s_k(.)$ et $h_k(.)$ où l'indice k signifie que les fonctions sont redéfinies à chaque commutation.*

Remarque 2.3. *Par nature, les SFM font appel à des fonctions pouvant être discontinues aux instants de commutation $t_k \in S$. De ce fait, nous définissons, pour une fonction du temps $f(t)$: $f(t_k^-) = f_k^-$ et $f(t_k^+) = f_k^+$. En général, quand une fonction f est discontinue aux instants de commutation, nous considérons sa valeur juste après la commutation et simplifions l'écriture f_k^+ par f_k. Bien entendu, en cas de continuité, $f_k^- = f_k^+ = f_k$. De plus, dans le cas où nous sommes amenés à définir une commutation à période T constante ($S = \{k.T, k = 0,1,2,...\}$), nous comprendrons que $f_k = f(k.T)$. Dans tous les cas, l'intervalle de temps $]t_k, t_{k+1}]$ ou $]k.T, (k+1).T]$ est appelé morceau k du SFM.*

2.4.2. Systèmes continus par morceaux

2.4.2.1. Définition

Pour faciliter la compréhension, nous présentons le cas linéaire à modèle dynamique invariant. On parle, dans ce cas, de système linéaire continu par morceaux (SLCM), qui est représenté par le quintuplet $\Sigma_c(S,\alpha,\beta_c,\beta_d,\gamma)$ défini par le système d'équations suivant :

$$x'(t)=\alpha.x(t)+\beta_c.u(t) \quad \forall t\in]t_k,t_{k+1}], \tag{2.17a}$$

$$x(t_k^+)=\beta_d.v(t_k) \quad \forall t_k\in S, \tag{2.17b}$$

$$y(t)=\gamma.x(t) \quad \forall t, \tag{2.17c}$$

dans lequel :

- $x(t)\in\Sigma^n$ est le vecteur d'état,
- $u(t)\in U^r$ est le vecteur d'entrées continues et bornées,
- $v(t)\in V^\sigma$ est le vecteur d'entrées bornées à discrétiser aux instants de commutation,
- $y(t)\in Y^m$ est le vecteur de sortie,
- $\alpha\in\Re^{n\times n}$, $\beta_c\in\Re^{n\times r}$, $\beta_d\in\Re^{n\times\sigma}$, $\gamma\in\Re^{m\times n}$: matrices réelles,
- $S=\{t_k,k=0,1,2,... \mid t_{k+1}>t_k\}, t_k\in\Im$: espace de commutation, avec $\Im$ espace temps continu.

L'équation (2.17a) décrit le fonctionnement continu du SCM sur le morceau $]t_k,t_{k+1}]$, $\forall t_k\in S$. L'équation (2.17b) donne la valeur de commutation de l'état juste après la l'instant de commutation. L'équation (2.17c) représente, quant à elle, l'équation de sortie.

Remarque 2.4. *Le système d'équation (2.17) montre clairement le rôle des deux entrées du SCM. L'entrée discrète régit les sauts de l'état aux instants de commutation (2.17b), alors que l'entrée continue agit sur son comportement entre ces instants (2.17a).*

Remarque 2.5. *Le cas non linéaire peut être décrit selon le même schéma, en remplaçant les équations (2.17a), (2.17b) et (2.17c) respectivement par :*

$$x'(t)=f_c(x(t),u(t)) \quad \forall t\in]t_k,t_{k+1}],$$

$$x(t_k^+)=f_d(v(t_k)) \quad \forall t_k\in S,$$

$$y(t)=h(x(t),u(t)) \quad \forall t.$$

Remarque 2.6. *Il est possible d'envisager un changement de modèle à chaque commutation. Dans ce cas, il convient de modifier les équations précédentes en conséquence :*

$$x'(t) = f_c^k(x(t),u(t)) \quad \forall t \in]t_k, t_{k+1}],$$

$$x(t_k^+) = f_d^k(v(t_k)) \quad \forall t_k \in S,$$

$$y(t) = h^k(x(t),u(t)) \quad \forall t,$$

où l'exposant k identifie le morceau k.

Ceci définit un formalisme multimodèle.

2.4.2.2. Fonctionnement

Considérant l'équation (2.17a) sur un morceau *k*, nous avons :

$$x(t) = e^{\alpha(t-t_k)}.x(t_k^+) + \int_{t_k}^{t} e^{\alpha(t-\tau)}.\beta_c.u(\tau).d\tau \quad \forall t \in]t_k, t_{k+1}] \tag{2.18}$$

En remplaçant la valeur de $x(t_k^+)$ selon (2.17b) dans l'équation (2.18), nous avons :

$$x(t) = e^{\alpha(t-t_k)}.\beta_d.v(t_k) + \int_{t_k}^{t} e^{\alpha(t-\tau)}.\beta_c.u(\tau).d\tau \quad \forall t \in]t_k, t_{k+1}] \tag{2.19}$$

La valeur de l'état $x(t)$ à gauche de t_k, notée $x(t_k^-)$, est obtenue en écrivant l'équation (2.19) sur le morceau $k-1$ et en faisant tendre t vers t_k par valeurs inférieures, soit :

$$x(t_k^-) = e^{\alpha(t_k-t_{k-1})}.\beta_d.v(t_{k-1}) + \int_{t_{k-1}}^{t_k} e^{\alpha(t_k-\tau)}.\beta_c.u(\tau).d\tau \tag{2.20}$$

À partir des équations de fonctionnement, nous pouvons illustrer le comportement d'un SLCM comme en figure 2.3 où son état évolue de manière continue en réponse à la première entrée continu $u(t)$ sur un morceau *k* à partir d'une condition initiale imposée par la deuxième entrée $v(t)$ à l'instant t_k.

Remarque 2.7. *En général, $x(t_k^-) \neq x(t_k^+)$, ce qui traduit une discontinuité de fonctionnement à la transition d'un morceau à l'autre.*

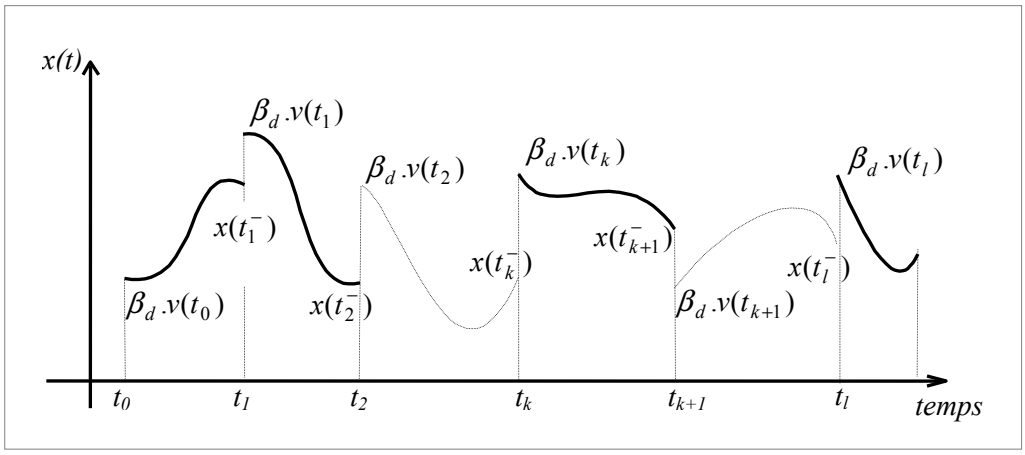


Fig. 2.3. *Évolution de l'état d'un SLCM*

2.4.2.3. Schéma fonctionnel

Le mode de description donné par le système d'équations (2.17) est directement interprétable sous la forme d'un schéma fonctionnel, ainsi que l'illustre la figure 2.4. Dans la figure 2.4a, nous représentons un schéma détaillé basé sur un intégrateur continu délivrant l'état $x(t)$. La sortie de cet intégrateur peut être réinitialisée à une valeur fournie par une deuxième entrée aux instants de commutation.

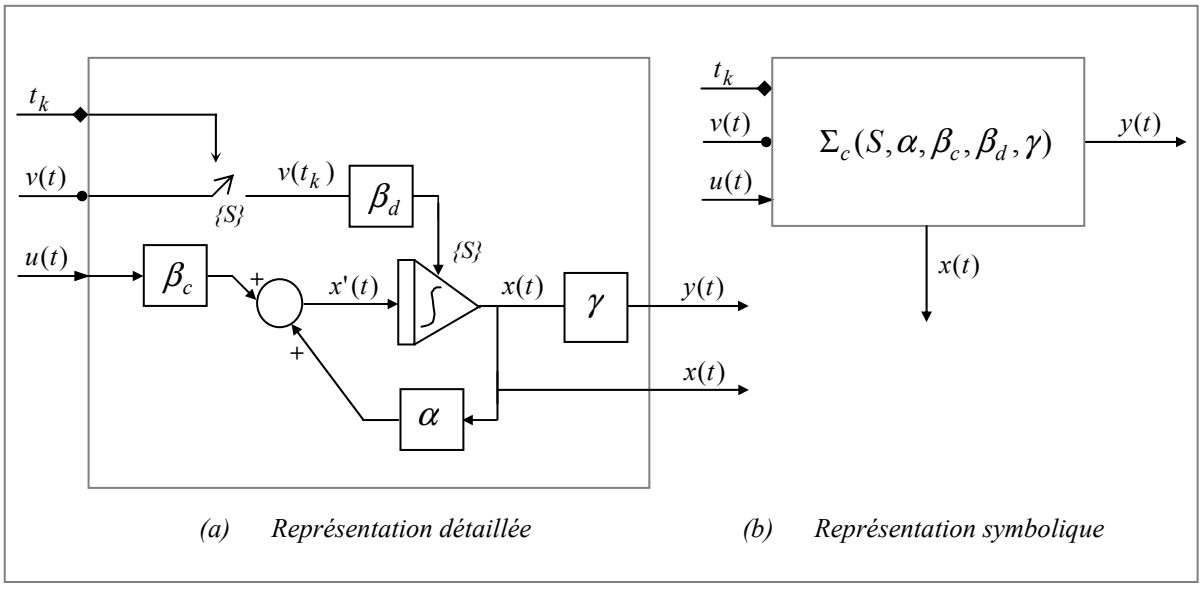


Fig. 2.4. *Construction d'un SLCM*

À partir du schéma détaillé de la figure 2.4a, nous proposons la représentation symbolique de la figure 2.4b, pour définir un SLCM.

Remarque 2.8. *Le SLCM possède donc trois entrées :*

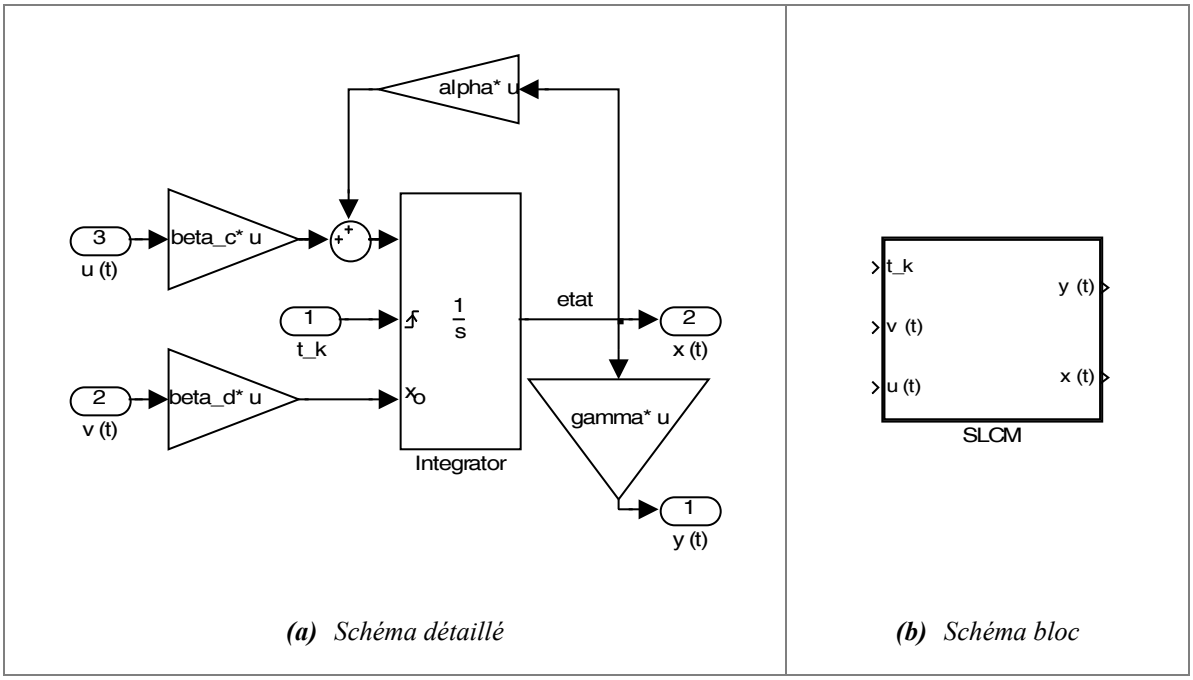
- *l'entrée continue* $u(t)$ *repérée par une flèche,*

- *l'entrée* $v(t)$ *repérée par un point marqueur peut être continue. Elle est, de toute façon, discrétisée selon* S *afin d'obtenir* $v(t_k)$,
- *l'entrée* t_k *repérée par un losange définit les instants de communication.*

Le fonctionnement n'est possible que si $u(t)$ *est bornée* $\forall t \in \Im$ *et* $v(t)$ *définie et bornée aux instants de communication. Pratiquement, cette condition est obtenue en imposant* $v(t)$ *bornée et continue aux instants de commutation. Dans ces conditions, à chaque instant* t_k, *on commute la sortie de l'intégrateur à la valeur* $x(t_k^+) = \beta_d . v(t_k)$, *puis le système évolue à partir de* $x(t_k^+)$, *sous l'influence de* $u(t)$, *selon (2.19). En* t_{k+1}, *on commute à une nouvelle valeur* $x(t_{k+1}^+) = \beta_d . v(t_{k+1})$, *et ainsi de suite.*

2.4.2.4. Réalisation technologique

Le schéma fonctionnel précédent (figure 2.4) est immédiatement implantable sur calculateur, par exemple à l'aide du logiciel Matlab/Simulink®. La figure 2.5 donne le schéma de réalisation d'une *boîte* SLCM sous ce logiciel.



(a) *Schéma détaillé* ***(b)*** *Schéma bloc*

Fig. 2.5. *Schéma de réalisation d'un SLCM sous Matlab/Simulink®*

L'élément clé du SLCM est bien évidemment l'intégrateur qui doit être paramétré convenablement en mode *reset* pour que sa sortie puisse être réinitialisée à une valeur fournie par une entrée supplémentaire. Le détail du paramétrage est donné en annexe II.

2.4.2.5. Exemples de fonctionnement

Afin d'apprécier le fonctionnement par morceaux, considérons un SLCM d'ordre 1 décrit par $\Sigma_c(S,-0.5,2,1,1)$ avec $S = \{k.T,\ k = 0,1,2,...\}$. L'évolution de la sortie (égale à l'état car $\gamma = 1$), ainsi que l'entrée $v(t)$ sont représentée en figure 2.6 pour une entrée continue

$u(t)=8\sin(2t)$, une entrée $v(t)=10\sin(3t)$ à discrétiser selon la période de commutation $T=1s$.

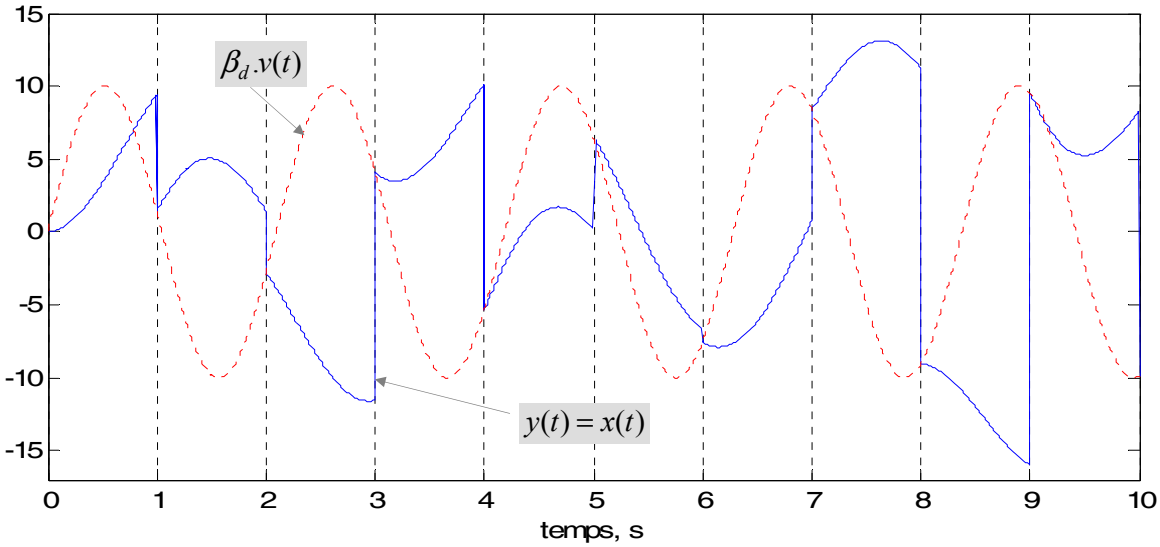


Fig. 2.6. *Évolution de l'état du SLCM d'ordre* 1 *avec* $T = 1s$

Nous pouvons constater dans la figure 2.6 que l'état du SLCM évolue de manière continue suivant $x'(t)=-0.5x(t)+16\sin(2t)$ selon (2.17a) entre les instants de commutation, à partir d'un condition initiale imposé par $v(t)$ à chaque $k.T, k=0,1,2,...$.

De la même manière, la figure 2.7 illustre l'évolution de l'état d'un SLCM d'ordre 2 défini par $\Sigma_c(S,\alpha,\beta_c,I_2,I_2)$ avec $\alpha=$[-0.5 -0.4; -0.7 0.8], $\beta_c=[2;-4]$ et $S=\{k.T, k=0,1,2,...\}$ où $T=0.5s$. Dans ce cas, $u(t)=8\sin(2t)$ et le signal $v(t)$ est tel que les deux composantes de $\beta_d.v(t)$ sont représentées en pointillé. Nous y voyons les deux composantes de l'état subissant une commutation à chaque $k.T, k=0,1,2,...$ et évo-

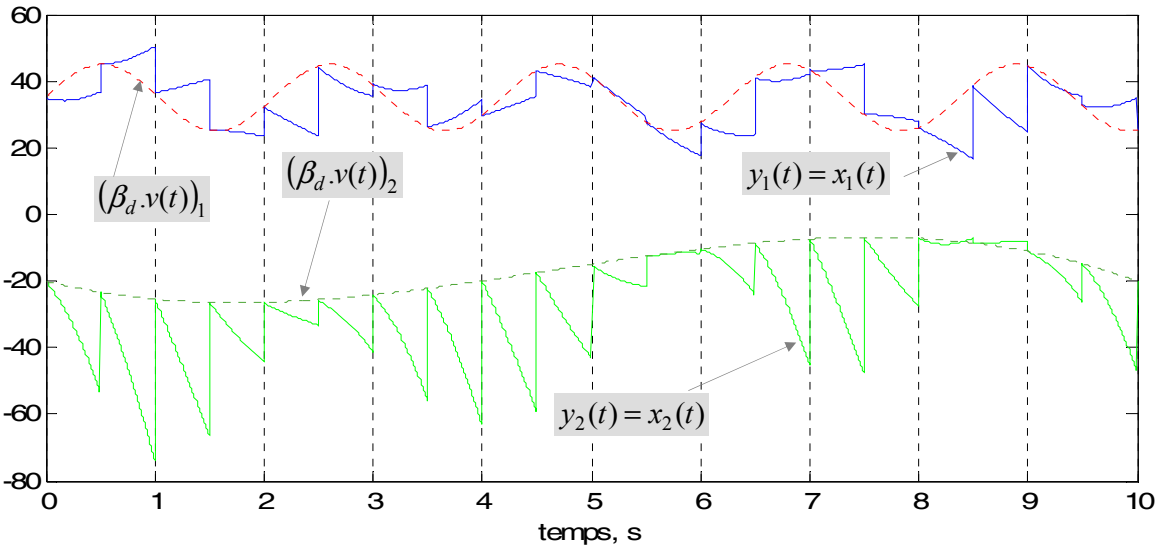


Fig. 2.7. *Évolution de l'état du SLCM d'ordre 2 avec* $T = 0.5s$

luant de manière continue selon (2.17a) sur un morceau k.

2.4.2.6. Cas remarquables

Le paramétrage du SLCM dépend de l'utilisation souhaitée. Nous verrons chapitre 4 le cas où les paramètres $S, \alpha, \beta_c, \beta_d$ et γ d'un SFM sont définis pour réaliser un contrôleur. Par ailleurs, les paramètres de $\Sigma_c(.)$ peuvent être définis pour réaliser des systèmes bien connus. Quelques exemples sont donnés ci-après.

2.4.2.6.1. Le Bloqueur d'Ordre Zéro (BOZ)

Pour réaliser un BOZ, nous choisissons $\alpha = 0$, $\beta_c = 0$, $\beta_d = I_n$, $\gamma = I_n$, ($\sigma = m = n$), et nous n'utilisons pas d'entée continue $u(t)$. Ainsi, selon les équations (2.17c) et (2.19), $\Sigma_c(.) = \Sigma_c(S,0,0,I_n,I_n)$ réalise l'échantillonnage-blocage de $v(t)$ selon S.

Si de plus $S = \{k.T, k = 0,1,2,...\}$, où T est la période d'échantillonnage, alors $\Sigma_c(S,0,0,I_n,I_n)$ réalise un BOZ à période constante.

2.4.2.6.2. L'échantillonneur généralisé

Ce type d'échantillonneur correspond au GSHF (Generalised Sample data Hold Function) défini par Kabamba [Kab87].

Dans ce cas, on choisit $S = \{k.T, k = 0,1,2,...\}$, où T est la période d'échantillonnage. Ici aussi, l'entrée continue n'est pas utilisée et on a : $\Sigma_c(.) = \Sigma_c(S,\alpha,0,\beta_d,\gamma)$.

En utilisant les équations (2.17c) et (2.19), nous pouvons donc écrire l'équation de sortie de $\Sigma_c(S,\alpha,0,\beta_d,\gamma)$ sur un morceau k :

$$y(t) = \gamma.e^{\alpha(t-k.T)}.\beta_d.v_k \quad \forall t \in]k.T,(k+1).T] \tag{2.21}$$

Afin d'interpréter ce résultat, posons :

$$F(t) = \sum_{k=0}^{\infty} \Gamma(t,k) \quad \forall t\,, \tag{2.22}$$

avec $$\Gamma(t,k) = \begin{cases} \gamma.e^{\alpha(t-k.T)}.\beta_d & \forall t \in]k.T,(k+1).T] \\ 0 & \text{ailleurs} \end{cases}$$

Il apparaît alors simplement que $F(t) = F(t+T)$ et que $F(t)$ est intégrable. Ainsi, $F(t) = \Gamma(t,k)\ \forall t \in]k.T,(k+1).T]$ et $y(t) = F(t).v_k$. Donc $F(t) \in \Re^{m\times\sigma}$ est une matrice intégrable et périodique (de période T). Ceci réalise un GSHF tel que défini par Kabamba.

2.4.2.6.3. Système linéaire continu

Bien que destinée à modéliser un système à commutation, la formalisation générique d'un SFM permet également de décrire un système linéaire continu ordinaire. Deux cas peuvent être envisagés :

- système dont l'état initial est imposé, ce qui signifie qu'à chaque démarrage du système, son état est amené à une valeur connue,
- système dont l'état initial n'est pas imposé, ce qui signifie qu'à chaque démarrage du système, son état initial est celui dans lequel on l'a laissé précédemment.

Système continu à état initial imposé

Nous adoptons, dans ce cas :

- $S = \{t_0\}$, impliquant que l'état initial est imposé au démarrage t_0
- $\beta_d = I_n$
- $\sigma = n$
- $\alpha = a$, $\beta_c = b$ et $\gamma = c$

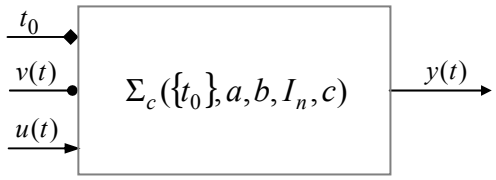


Fig. 2.8. *Système linéaire continu à état initial imposé*

Ainsi, $\Sigma_c(.) = \Sigma_c(\{t_0\}, a, b, I_n, c)$, qui est représenté en figure 2.8, décrit le fonctionnement d'un système linéaire continu ordinaire évoluant à partir d'un état initial imposé via l'entrée $v(t)$: $x(t_0^+) = v(t_0)$. Nous avons donc :

$$\Sigma_c(\{t_0\}, a, b, I_n, c) \quad \Leftrightarrow \quad \left\{ \begin{array}{c} x'(t) = a.x(t) + b.u(t) \\ y(t) = c(t) \end{array} \right\}_{x_0 = v(t_0)} \tag{2.23}$$

L'équation (2.19) montre d'ailleurs que la dynamique de $\Sigma_c(\{t_0\}, a, b, I_n, c)$ est donnée par :

$$x(t) = e^{a(t-t_0)}.v(t_0) + \int_{t_0}^{t} e^{a(t-\tau)}.b.u(\tau).d\tau \,, \ \forall t > t_0 \tag{2.24}$$

Système continu à état initial propre

Dans ce cas, nous adoptons le paramétrage suivant : $\Sigma_c(.) = \Sigma_c(\{t_k\}, a, b, I_n, c)$ et nous imposons : $v(t) = x(t)$ de sorte qu'à l'instant t_0, l'état conserve la valeur atteinte à l'issue du dernier fonctionnement. La figure 2.9 illustre ce cas.

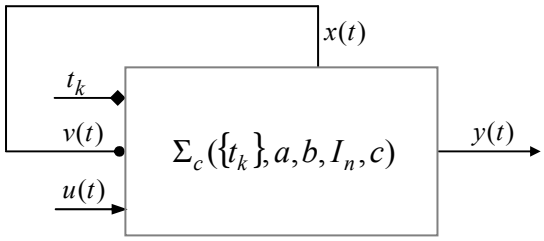


Fig. 2.9. *Système linéaire continu à état initial propre*

Remarque 2.9. *Le paramétrage* $\Sigma_c(.) = \Sigma_c(\{t_0\}, a, b, I_n, c)$, *où* t_k *est remplacé par* t_0, *réalise la même fonctionnalité.*

2.4.3. Systèmes bi-échantillonnés

2.4.3.1. Introduction

Les auteurs de [KV02] ont repris le formalisme général des SFM pour mettre en œuvre un second type de SFM : les systèmes bi-échantillonnés (SBE). Contrairement aux SCM qui présentent une évolution *continue* de leur état entre les instants de commutation, l'état d'un SBE évolue de manière échantillonnée entre ces instants. Ces systèmes possèdent donc deux espaces temps discrets représentés par deux échelles de temps (figure 2.10).

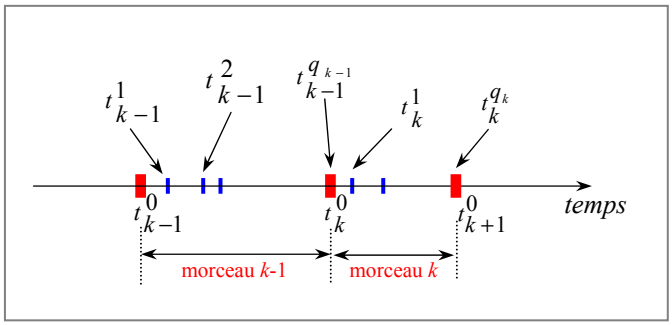


Fig. 2.10. *Les deux échelles de temps discrètes du SBE*

Les instants discrets sont notés t_k^i, où k désigne l'échelle de temps relative aux commutations et i l'échelle de temps relative au fonctionnement du système entre deux instants de commutation. Ainsi, l'espace de commutation est donné par $S_k = \{t_k^0, k = 0,1,2,...\}$ et le morceau k est délimité par deux instants de commutation successifs t_k^0 et t_{k+1}^0 conformément à la remarque 2.3. Par ailleurs l'espace temps discret utilisé pour décrire le fonc-

tionnement dans un morceau k est donné par $S_i = \{t_k^i, i = 0,...,q_k - 1\}$. A l'intérieur du morceau k l'indice reste constant, tandis que l'exposant croît de 0 (instant initial du morceau) à $q_k - 1$. Il existe, par conséquent, q_k échantillons de l'état dans un morceau *k*. Dans le cas général, la durée des morceaux n'est pas constante, de même que la période d'échantillonnage et le nombre d'instants d'échantillonnage à l'intérieur d'un morceau. Enfin, l'instant final du morceau $k-1$ est confondu avec l'instant initial du morceau k. Ceci se traduit par la relation $t_{k-1}^{q_{k-1}} = t_k^0$.

Les SLBE mettent en jeu deux espaces d'entrée. Le premier espace U^r, nécessairement échantillonné selon S_i, détermine l'évolution de l'état sur un morceau *k*. Le second espace V^σ sert à définir la valeur à laquelle l'état est commuté $\forall t \in S_k$.

2.4.3.2. Définition

Comme précédemment, nous présentons le cas linéaire à modèle dynamique invariant. On parle, dans ce cas, de système linéaire bi-échantillonné (SLBE), qui est représenté par le sextuplet $\Sigma_d(S_i, S_k, A, B, \beta_d, C)$ défini par le système d'équations suivant :

$$x_k^{i+1} = A.x_k^i + B.u_k^i \text{ pour } i = 0,...,q_k - 1 \text{ sur chaque morceau } k, \quad (2.25a)$$

$$x_k^0 = \beta_d.v_k^0 \qquad \forall t_k^0 \in S_k, \quad (2.25b)$$

$$y_k^i = C.x_k^i \qquad \forall t_k^i \quad (2.25c)$$

dans lequel :

- $x_k^i \in \Re^n$ est le vecteur d'état à l'intérieur du morceau *k*,
- $u_k^i \in U^r$ est le vecteur de commande à l'intérieur du morceau *k*,
- $v(t) \in V^\sigma$ est le vecteur de commande qui n'est pris en compte qu'aux instants (t_k^0) de commutation pour définir l'état initial x_k^0 pour le morceau considéré,
- $y_k^i \in \Re^m$ est le vecteur de sortie,
- $A \in \Re^{n\times n}$, $B \in \Re^{n\times r}$, $\beta_d \in \Re^{n\times\sigma}$, $C \in \Re^{m\times n}$: matrices réelles,

À partir des équations de fonctionnement (2.25), nous pouvons illustrer le comportement d'un SLBE comme en figure 2.11 où son état évolue selon (2.25a) sur un morceau *k* à partir d'une condition initiale imposée par l'entrée $v(t)$ à l'instant t_k^0.

Remarque 2.10. *En général,* $x_k^0 \neq x_{k-1}^{q_{k-1}}$, *ce qui traduit une discontinuité de fonctionnement à la transition d'un morceau à l'autre.*

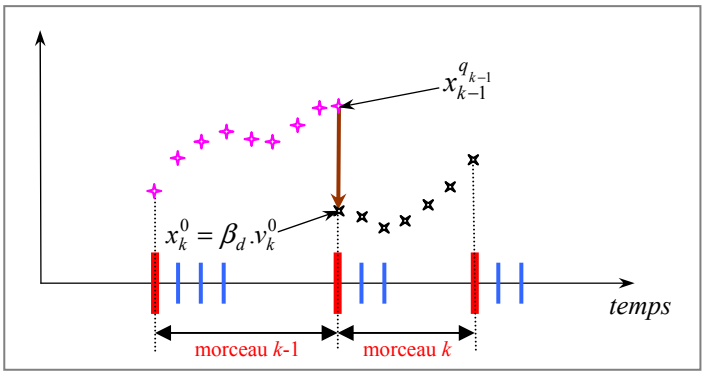


Fig. 2.11. *Évolution de l'état d'un SLBE*

Remarque 2.11. *Comme pour les SCM, on peut étendre ce formalisme au cas non linéaire et/ou multimodèle.*

2.4.3.3. Schéma fonctionnel

La définition d'un SLBE (2.25) est directement interprétable sous la forme d'un schéma fonctionnel, ainsi que l'illustre la figure 2.12. Dans la figure 2.12a, nous introduisons un bloc « R-BOZ » à sortie « réinitialisable » représentant un retard suivi de d'un BOZ. Le R-BOZ, cadencé à des intervalles de temps selon l'échelle de temps i, est réinitialisé à une valeur fournie par une deuxième entrée aux instants de commutation.

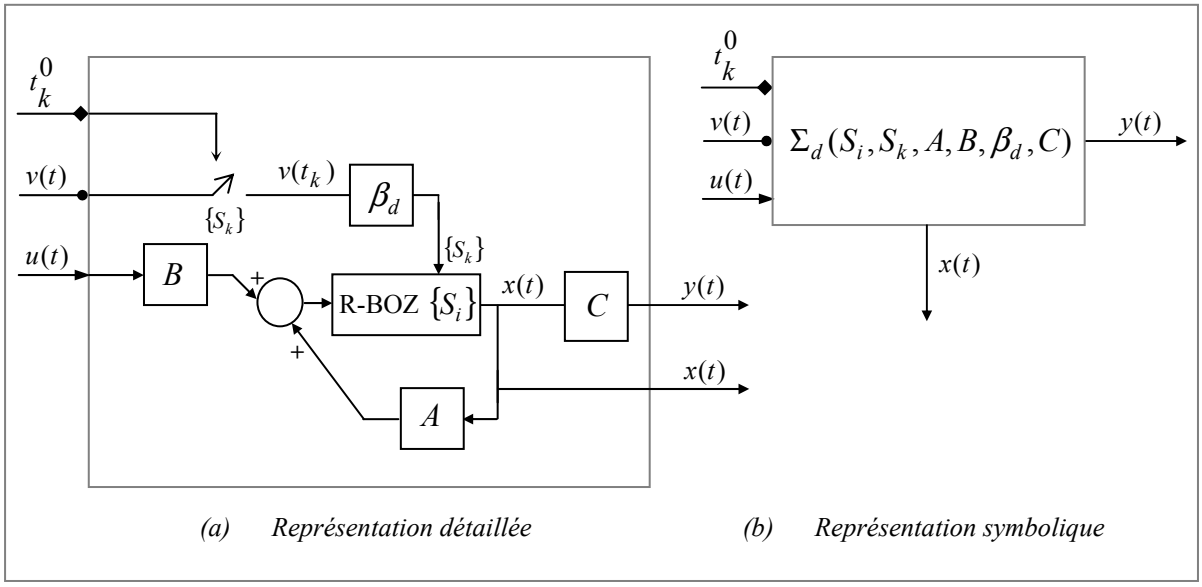


Fig. 2.12. *Construction d'un SLBE*

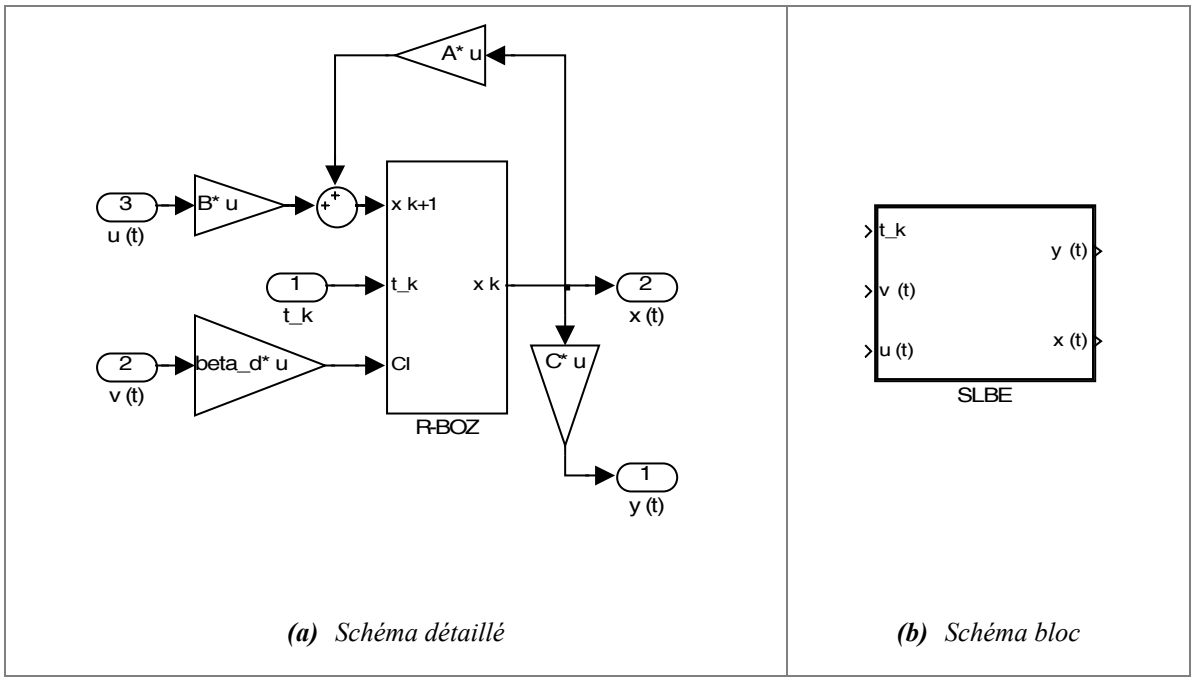
À partir du schéma détaillé de la figure 2.12a, nous proposons la représentation symbolique de la figure 2.12b, pour définir un SLBE.

Remarque 2.12. *Le R-BOZ peut être remplacé simplement par un bloc retard à sortie « réinitialisable » si u(t) est garantie échantillonnée-bloquée selon l'échelle de temps i.*

2.4.3.4. Réalisation technologique

Comme pour les SLCM, l'implantation des SLBE sur calculateur numérique est immédiate. En pratique, le schéma global d'un SLBE est quasi-identique à celui du SLCM présenté dans la figure 2.5a, sauf que l'intégrateur continu est remplacé par un bloc R-BOZ.

Le schéma de réalisation sous Matlab/Simulink® est donné en figure 2.13.



(a) *Schéma détaillé* ***(b)*** *Schéma bloc*

Fig. 2.13. *Schéma de réalisation d'un SLBE sous Matlab/Simulink®*

La bibliothèque de Simulink® n'offrant pas directement de bloc de retard intégrant le mode *reset* avec condition initiale externe, nous montrons comment réaliser un R-BOZ avec les fonctionnalités du logiciel en annexe II.

2.4.3.5. Exemple de fonctionnement

Soit $\Sigma_d(S_i, S_k, A, B, \beta_d, C)$ un SLBE d'ordre 1 défini par $S_k = \{k.T, k = 0,1,2,...\}$, $T = 2s$, $S_i = \{i.t_e, i = 0,...,q-1\}$, $q = 20$ (q étant constant pour chaque morceau k ($q_k = q \; \forall k$)), $t_e = T/q = 0.1s$, $A = 0.8, B = C = 1$ et $\beta_d = 1$.

La figure 2.14 montre l'évolution de la sortie (égale à son état car $C = 1$) de ce SLBE pour des entrées $u(t) = 8\sin(t)$ et $v(t) = 10\sin(2t)$. Nous constatons que l'état évolue de manière échantillonnée selon (2.25a) sur un morceau k à partir d'une condition initiale imposée par $\beta_d.v(t)$ à chaque instant de commutation selon (2.25b). La figure illustre bien les instants de commutation relatifs à $S_k = \{k.T, k = 0,1,2,...\}$, montrant ainsi une discontinuité dans le fonctionnement échantillonné de l'évolution de l'état sur le morceau k. Elle montre également qu'il existe 20 pas sur chaque morceau.

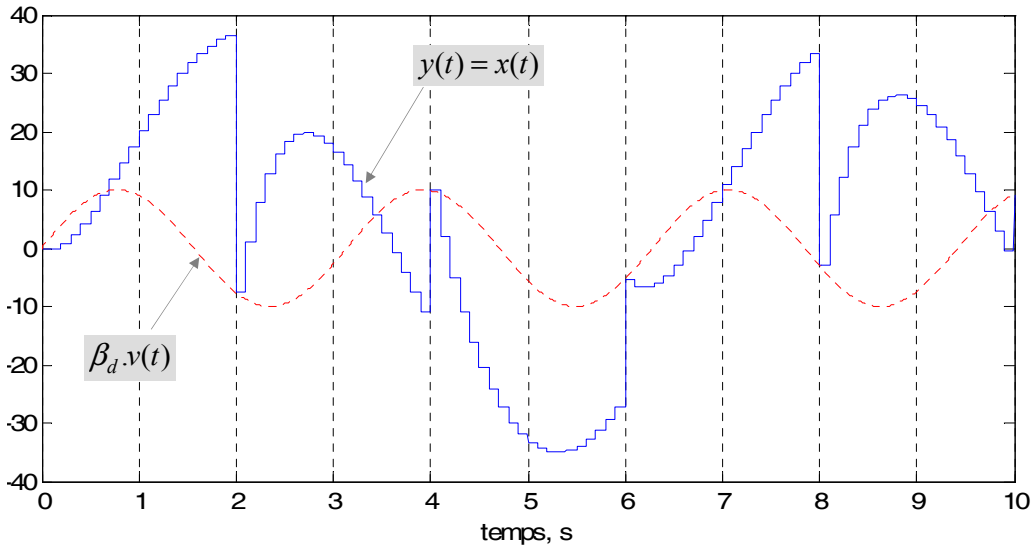


Fig. 2.14. *Évolution de l'état du SLBE d'ordre* 1 *avec* $T = 2s$

2.4.3.6. Comparaison des deux types de SFM linéaires

2.4.3.6.1. Structure propre du SLBE

Afin de réaliser le fonctionnement d'un SLCM par un SLBE donné en figure 2.13a, il faut ajuster les matrices A et B ce dernier pour qu'il réalise la fonction d'intégration sur un morceau k. En effet, la fonction intégrale (1/s) de l'intégrateur continu du SLCM peut être approximée, en discret, par plusieurs fonctions de transfert pour définir un intégrateur discret. Si on choisit de remplacer l'intégrateur continu (dans la figure 2.4a) par la méthode « Forward Euler », représentée par la fonction $t_e/(z-1)$ avec t_e étant la période d'échantillonnage de l'intégrateur discret, il convient de décrire le fonctionnement d'un SLCM de manière échantillonnée sur un morceau k par :

$$x_k^{i+1} = (I_n + t_e.\alpha).x_k^i + t_e.\beta_c.u_k^i .$$

Nous pouvons donc identifier un SLBE équivalent paramétré par $S_i = \{i.t_e, i = 0,...,q-1\}$, $A = I_n + t_e.\alpha$ et $B = t_e.\beta_c$ en ce qui concerne l'évolution de l'état. Les matrices de sortie et de gain sur $v(t)$ restent les mêmes.

2.4.3.6.2. Autre structure bi-échantillonnée

Toutefois, si l'on désire faire l'équivalent discret (au sens des SLBE) d'un SLCM, on peut simplement remplacer l'intégrateur continu de la figure 2.13a par une approximation échantillonnée adéquate de la fonction intégrale continu « 1/s ». Dans ce cas, on conserve les matrices α et β relatives au SLCM dans le nouveau montage afin de réaliser l'équivalent discret du SLCM. La démonstration de la remarque suivante utilise ce principe pour faire la comparaison des deux types de SFM.

Remarque 2.13. *L'évolution discrète de l'état d'un SLBE entre les instants de commutation dépendant de la valeur de q, nous comprenons bien que cette évolution tend vers celle d'un SLCM si* $q \to \infty$. *Cette tendance est illustrée par la figure 2.15 où q augmente de (a) à (c) pour un SLBE équivalent à un SLCM dont l'évolution est représentée par (d).*

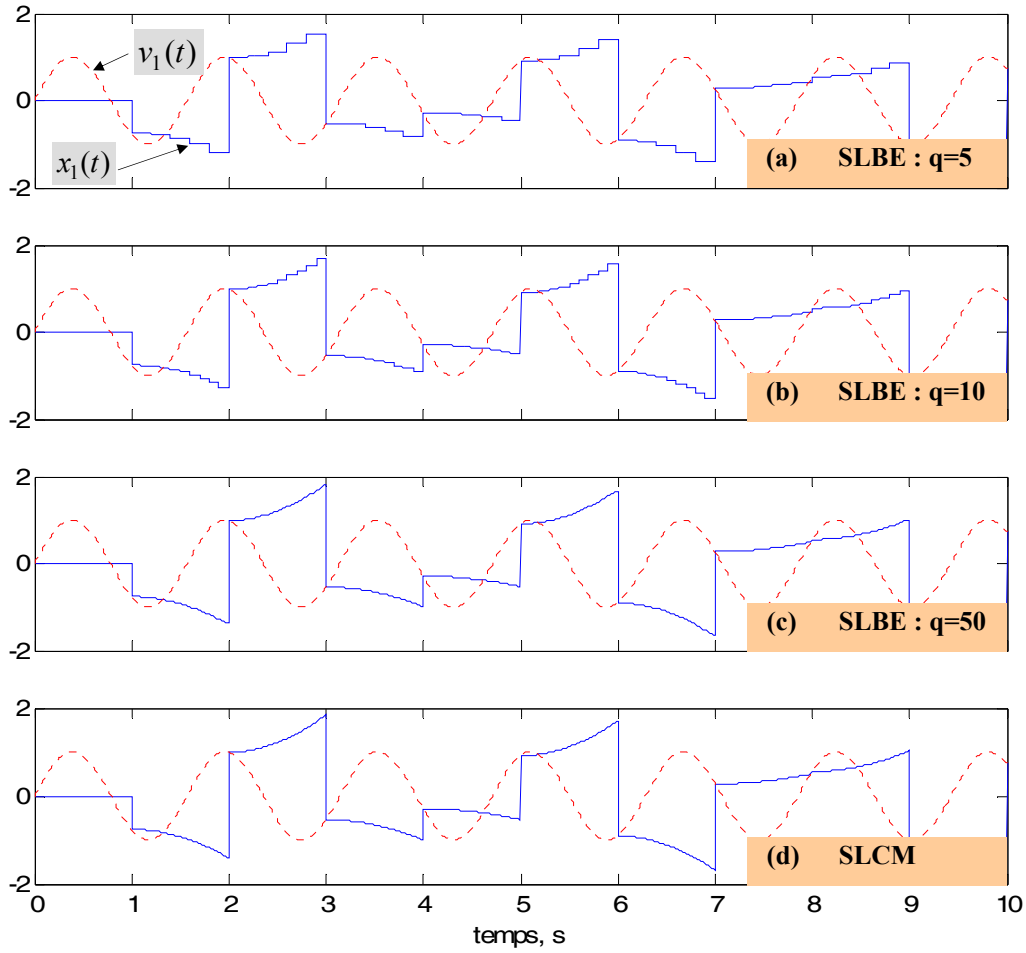


Fig. 2.15. *Évolution de l'état d'un SLBE tendant vers celle d'un SLCM quand* $q \to \infty$

À noter que seule la première de des deux composantes de l'état est illustrée. Le SLCM est caractérisé par $\alpha = [-0\ 1; 2\ -1]$ *et la période de commutation des SFM est d'une seconde.*

2.5. Conclusion

L'étude des systèmes hybrides a récemment pris une ampleur considérable en automatique. Comme nous l'avons montré, il est possible de donner de multiples éclairages à la modélisation et à l'utilisation de ces systèmes. Dans ce sens, les différents travaux cités ont apporté des contributions variées selon les problématiques envisagées. Dans tous les cas, les auteurs proposent des formalismes qui satisfont la taxonomie de Branicky.

Notre étude se situe dans la continuité des travaux entamés au LAGIS [KV01] et [KV02]. Dans ce sens, elle propose des modèles hybrides de la classe des SFM, selon un formalisme se rapprochant de celui de Haddad et al. [HCK99, HCK01, Kab03a], qui considèrent des systèmes dits à impulsions (*impulsive systems*). Toutefois, par rapport aux travaux de Haddad et al., le formalisme SFM offre l'avantage supplémentaire d'une mise en œuvre immédiate offrant de multiples possibilités d'exploitation temps réel. Dans la suite de l'étude nous développons l'approche SFM pour construire des systèmes d'identification en ligne et de commande.

Le chapitre 3 propose un modèle SFM dont l'état est forcé à chaque instant de commutation à la valeur de celui d'un processus réel *continu* à identifier. Ce mode de *synchronisation* permet d'adapter les paramètres du modèle hybride à ceux du processus.

Les chapitres 4 et 5 seront consacrés, quant à eux, à la mise en œuvre des SFM pour réaliser des contrôleurs utilisant comme retour soit l'état échantillonné (chapitre 4) soit la sortie du processus sous forme échantillonnée et retardée (chapitre 5).

Chapitre 3

Identification : « clonage » par les SFM

Nous décrivons, dans ce chapitre, une méthode d'identification récursive applicable aux processus MIMO linéaires continus dont l'état est disponible. Cette méthode, qui a été développée pour une identification en ligne, est adaptable à des processus contrôlés par calculateur numérique. L'enjeu est d'identifier les paramètres d'un modèle d'état linéaire du processus. L'identificateur, considéré comme un clone du processus, est un SFM multimodèle dont les paramètres varient selon un algorithme récursif et adaptatif. Le principe est de faire varier itérativement les paramètres du SFM clone afin de rendre son état identique à celui du processus. Ainsi, le comportement de celui-ci est reproduit par son clone à la fin du clonage. Nous donnons, dans ce chapitre, une description formelle de la méthode ainsi que la démonstration de convergence de l'algorithme. Nous fournissons également des exemples qui illustrent la méthode en simulation et en temps réel.

3.1. Motivations

L'identification d'un processus est une phase préliminaire très importante dans l'optique de faire de la commande. En effet, afin de mieux contrôler un système, il est préférable d'en avoir une *connaissance* a priori. Par exemple un conducteur pourra mieux maîtriser son véhicule s'il *connaît* le comportement de celui-ci dans l'environnement où il est amené à évoluer. Formellement, la construction d'un *modèle* (modélisation) permet de décrire le comportement d'un processus à l'aide d'une représentation mathématique à partir de laquelle il est possible de comprendre, commander et/ou améliorer le fonctionnement du processus. Définir un modèle consiste à lui attribuer une structure : c'est l'étape de *caractérisation* [WP94]. Cette étape est très délicate car l'intuition peut y jouer un grand rôle. Le choix d'une structure définit une classe de comportements possibles et un ensemble admissible, auquel le vecteur des paramètres doit appartenir pour que le modèle soit considéré comme acceptable. Selon les auteurs de [WP94], les points suivants sont à tenir en compte dans le choix d'une classe de modèle :

- l'objectif de la modélisation (analyse de phénomènes pour offrir une explication, détermination de grandeurs inaccessibles par capteur, test d'hypothèse, commande de processus, …),
- les conditions d'utilisation du modèle,
- le coût de construction du modèle,
- les possibilités d'investigation (disponibilité de données détaillées).

Dans notre cas, nous admettons, pour le processus, un modèle d'état continu destiné au développement de lois de commande.

Le modèle étant choisi, il reste à déterminer la valeur de ses paramètres, c'est la phase d'identification [SSR96, ADG+98]. La communauté des automaticiens s'est beaucoup penchée sur cette question [UR87, UR98, DVS93, LAMK95, ZBG95] et de nombreuses méthodes d'identification ont déjà été développées. Nous présentons ci-après une classification des méthodes existantes, selon les critères suivants :

- approche *boîte noire* et approche *boîte blanche*,
- méthodes *directes* et *indirectes*,
- approche *représentation d'état* et approche *fonction de transfert*,
- méthodes *en ligne* et *hors ligne*,

- approche *récursive*.

3.1.1. Approche *boîte noire* et approche *boîte blanche*

L'identification *boîte noire* vise à déterminer les paramètres d'un modèle générique d'un processus sans aucune connaissance *a priori* sur la structure interne de celui-ci. Le processus est alors vu comme une *boîte noire* et l'identification se fait en analysant son entrée et sa sortie. En revanche, dans certains cas il est possible de faire appel aux connaissances des spécialistes du domaine considéré pour regrouper, généralement sous forme de systèmes différentiels ou algébriques, les lois et relations de la physique qui décrivent le comportement du processus : c'est l'approche *boîte blanche*. Entre ces deux approches, il existe une approche intermédiaire dite *boîte grise* qui se base à la fois sur certaines relations connues et sur l'analyse des entrées – sorties du processus. La méthode présentée dans ce document est une approche *boîte noire*.

3.1.2. Méthodes *directes* et *indirectes*

Dépendant de la stratégie d'identification, nous pouvons distinguer les approches suivantes selon [FL99] :

L'approche directe	L'identification se fait en utilisant les mesures de l'entrée et de la sortie du système en omettant un éventuel asservissement [SK01]. Cette méthode donne *directement* les paramètres d'un modèle du système.
L'approche indirecte	Le système asservi est identifié et les paramètres du système sont déterminés en utilisant ceux du contrôleur qui sont connus [FL99].

Tab. 3.1. *Les approches directes et indirectes*

Alors que certains ouvrages sont dédiés spécialement à l'identification en boucle fermée [QL03], la méthode présentée dans ce chapitre est conçue comme une approche directe, tout en admettant que l'identificateur puisse être utilisé en boucle fermée, pour des raisons de stabilisation ou de commande adaptative. Par contre, contrairement aux méthodes indirectes, les seules informations requises par l'identificateur sont l'entrée et l'état du processus.

Il est à souligner que la notion d'identification directe/indirecte a une autre interprétation chez d'autres auteurs. En effet, plusieurs publications présentent la possibilité d'identifier

soit un modèle continu soit un modèle discret, à partir de données discrétisées. Dans ce sens, [GY04] adopte l'appellation *directe* pour définir l'identification d'un modèle continu *directement* à partir des données discrètes et l'appellation *indirecte* lorsque le modèle continu est déduit de l'identification d'un modèle discret. Même si la méthode directe semble plus intuitive, les auteurs de [GY04] soulignent la difficulté de reconstruire les dérivées dans le modèle continu, notamment lorsque les signaux ne sont pas mesurables. Ils indiquent toutefois que, compte tenu de l'intérêt croissant de l'approche directe [SR83, SR93], plusieurs ouvrages [You81, GMR03] ont été consacrés au développement de méthodes de calcul de ces dérivées dans le cadre de l'identification.

3.1.3. Approche *représentation d'état* et approche *fonction de transfert*

L'approche par la fonction de transfert est couramment utilisée dans le domaine de l'identification [SSR96, SM65, TD96]. Elle permet de déterminer les paramètres d'un modèle représenté par une fonction de transfert. Bien que particulièrement efficace pour des systèmes monovariables, l'identification d'un modèle entrée-sortie se révèle être plus difficile lorsque le procédé comporte plusieurs entrées et plusieurs sorties. Elle demande en effet de connaître *a priori* l'ordre des polynômes composant les matrices polynomiales du modèle. De plus, lorsque le nombre d'entrées et de sorties augmente, la gestion des matrices de transfert devient plus délicate. Déjà en identification hors ligne, ces difficultés ont motivé le développement de techniques estimant directement un modèle d'état[1] du système sans phase transitoire d'identification de représentations d'entrée-sortie. En même temps, les lois de commande se basant très souvent sur la représentation d'état, il est intéressant d'envisager cette identification du modèle d'état, évitant ainsi une série de calculs supplémentaire pour retrouver les paramètres de la représentation d'état à partir de ceux d'une fonction de transfert, surtout si on envisage une commande adaptative faisant intervenir l'identificateur et le contrôleur. La méthode des sous-espaces [Vib95, VD96, BSR98, BRS98, Mer04], par exemple, utilise l'approche d'état.

Avec l'avènement des calculateurs numériques, on peut être tenté de mettre en avant la modélisation d'un procédé par une représentation d'état *discrète*. Toutefois, il est préférable de conserver un modèle d'état à temps continu (METC) pour décrire les processus physiques dont l'état est souvent continu [UR90, SR91]. La méthode, que nous présentons concerne l'identification des systèmes linéaires MIMO continus, par l'approche de la représentation d'état.

[1] La représentation d'état constitue un outil de prédilection pour décrire les systèmes et leur évolution.

3.1.4. Méthodes *en ligne* et *hors ligne*

Parmi les méthodes disponibles, il existe des méthodes (comme l'analyse harmonique) qui fournissent une estimation des paramètres par un traitement *hors ligne* des mesures collectées sur le système. D'autres méthodes réalisent l'identification *en ligne*, ce qui permet de détecter les changements éventuels dans le fonctionnement pour faire de la commande adaptative [TD96, CSS03]. Certaines d'entre elles identifient directement le modèle recherché en temps réel. La méthode que nous proposons vise à construire un modèle de simulation, qui est exécuté en parallèle avec le processus (càd *en ligne*), afin d'en reproduire au mieux le comportement.

3.1.5. Approche *récursive*

La récursivité est particulièrement intéressante pour envisager l'identification d'un modèle, dans le cas où des changements (légères variations) dans le comportement du processus peuvent survenir. Plusieurs auteurs se sont penchés sur les méthodes adaptatives depuis les années 70 [LS83, You84, Lju99] pour fournir des modèles dont les paramètres varient pour prendre en compte les changements du processus. La majorité des méthodes proposées reposent sur des représentations de type entrée-sortie. Ces dernières se sont initialement développées en raison de la disponibilité d'algorithmes hors ligne fondés sur ce type d'écriture [SS89]. L'identification récursive, considérée comme une spécialité de l'automatique a connu un développement considérable à la fin du $XX^{ème}$ siècle tant sur le plan théorique [LS83, Lju96, Lju99, JKDW01, LBB+01] que pratique [ZVDL94, GSMR96, LLM97, CR01].

Classiquement, les méthodes comme ARMA, qui se basent sur les moindres carrés récursifs servent à sélectionner le vecteur de paramètres qui permet le meilleur ajustement de la sortie du modèle à celle du système. Néanmoins, ces méthodes travaillent sur une fenêtre glissante des valeurs d'entrée/sortie du système, ce qui requiert une sauvegarde de tous les échantillons de données, rendant la méthode coûteuse en mémoire lors de l'implantation temps réel. De plus, elles nécessitent souvent une inversion matricielle à chaque pas d'échantillonnage qui ne peut être évitée que par une série d'opérations qui risque d'augmenter le temps de calcul. Les ouvrages récents ont repris les méthodes classiques hors ligne pour développer des techniques moins coûteuses afin d'envisager l'identification en ligne dans les domaines tels que l'aéronautique [KKKN01, UB03], la robotique [ÖG02], le filtrage [GBT99, Gus00] ou encore la surveillance de systèmes [Abi88, DBL+01, Öst02, Oku03]. Les techniques récursives récentes ont pu offrir une alternative aux méthodes robustes telles que la décomposition en valeurs singulières (DVS) [Ric01] qui demeure irréalisable en ligne de par sa charge calculatoire.

3.2. Contexte de travail

La méthode que nous proposons réalise l'identification *récursive directe* et *en ligne* d'un processus MIMO en utilisant, en temps réel, la valeur bornée[2] de l'entrée $u(t) \in U^r$ du système et son état $x(t) \in \Sigma^n$ qui est supposé entièrement disponible[3].

Hypothèse 3.1. *Nous faisons l'hypothèse que l'état reste borné, soit parce que le processus est naturellement stable, soit parce qu'il est stabilisé par un contrôleur.*

La méthode, appelée *clonage*, permet d'obtenir un modèle représentatif (*clone*) du processus sans aucune connaissance *a priori* de ses équations physiques internes. La seule information utilisée pour bâtir le *clone* est la dimension n du vecteur d'état $x(t) \in \Sigma^n$ du processus supposé entièrement disponible. Dans ce contexte, la méthode peut être considérée comme une approche *boîte noire*.

Le principe de la méthode est comparable à celui de [HV69, LN73, ADG$^+$98], où un algorithme adaptatif *continu* identifie le modèle. De la même façon, nous établissons un modèle de simulation et rendons l'erreur d'état entre le processus et son modèle nulle afin d'en déduire les paramètres d'un METC du processus, le tout étant régi par une approche adaptative.

L'originalité de notre stratégie réside dans la nature hybride du modèle de simulation qui identifie le METC du processus. Ce modèle conceptuel que nous considérons comme étant le *clone* du processus possède des paramètres modifiables, et est conçu pour reproduire le comportement du processus à l'aide d'une procédure adaptative qui modifie ses paramètres de manière itérative. Une fois le *clonage* terminé, les paramètres du *clone* correspondent aux paramètres du METC du processus. Le *clone* est un SLCM[4] à paramètres changeants dont l'état est synchronisé à celui du processus à chaque instant de commutation.

Ce chapitre décrit le principe du clonage et donne les définitions des termes utilisés ainsi que le formalisme mathématique correspondant. Aussi, nous proposons un schéma de construction du clone facilement implantable sur simulateur ou sur application temps réel. Enfin, des exemples de simulation et d'application temps réel sont donnés à la fin du chapitre. Ces exemples prennent en compte les bruits de mesure d'état et les variations temporelles (sous certaines conditions) des paramètres du processus à identifier.

[2] Car on suppose que la commande *u(t)* du processus provient d'un calculateur numérique.
[3] Ce qui implique donc que la dimension *n* est supposée connue.
[4] Défini au chapitre 2.

3.3. Le principe de clonage

3.3.1. Modèle mathématique

Le processus à identifier est décrit par un modèle d'état linéaire d'ordre n :

$$x'(t) = a.x(t) + b.u(t) \tag{3.1}$$

L'état $x(t) \in \Sigma^n$ du processus est supposé entièrement accessible et $u(t) \in U^r$ représente l'entrée. Les matrices constantes $a \in \Re^{n \times n}$ et $b \in \Re^{n \times r}$ représentent les paramètres du modèle à identifier.

Définition 3.1. $\varphi \in \Re^{(n+r) \times n}$ *telle que* $\varphi^T = [a,b]$ *désigne l'ensemble des paramètres du METC du processus à identifier.*

3.3.2. Identification par clonage

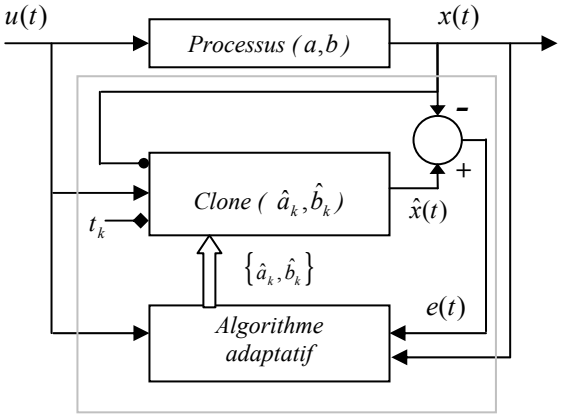


Fig. 3.1. *Structure de clonage*

La figure 3.1 illustre le principe de la méthode d'identification semblable à celui proposé dans [Knu06, ADG+98]. Deux blocs opératifs sont nécessaires pour réaliser le clonage : d'abord :

- le *clone* qui fonctionne en parallèle avec le processus et s'exécute en temps réel,
- le bloc adaptatif qui permet de régler itérativement les paramètres du *clone*.

Définition 3.2. *Le clone, qui est du même ordre que le processus, a pour fonction d'en reproduire le comportement. Son état est représenté par* $\hat{x}(t) \in \Sigma^n$ *. Il possède trois caractéristiques principales :*

- *Il a la même entrée u(t) que le processus et s'exécute en parallèle.*

- *Il est défini par ses matrices d'état* $\hat{a}_k \in \Re^{n\times n}$ *et* $\hat{b}_k \in \Re^{n\times r}$*, qui sont constantes sur chaque morceau* k *et rafraîchies à chaque* t_k *par le bloc adaptatif. Nous désignons la matrice des paramètres du clone par* $\hat{\varphi}_k \in \Re^{(n+r)\times n}$ *telle que* $\hat{\varphi}_k^T = \left[\hat{a}_k, \hat{b}_k\right]$.

- *Il possède une nature hybride : son état* $\hat{x}(t)$ *est continu sur chaque morceau* k *et est commuté à la valeur de l'état* $x(t_k)$ *du processus à chaque* t_k.

Définition 3.3. *On appelle matrice d'erreur paramétrique la matrice* $\Phi_k \in \Re^{(n+r)\times n}$ *telle que* $\Phi_k^T = \hat{\varphi}_k^T - \varphi^T$.

Définition 3.4. *On appelle erreur d'état la différence* $e(t)$ *entre l'état du clone et celui du processus :* $e(t) = \hat{x}(t) - x(t)$.

Afin de réaliser le clonage, nous devons annuler Φ_k lorsque $k \to \infty$.

3.4. Structure du clone

L'idée étant d'utiliser un modèle de référence, nous construisons un clone qui est un SLCM à paramètres changeants tel que défini dans le chapitre 2. Nous décrivons son fonctionnement par :

$$\hat{x}'(t) = \hat{a}_k.\hat{x}(t) + \hat{b}_k.u(t) - \hat{a}_k.e(t) \quad \forall t \notin S \tag{3.2a}$$

$$\hat{x}(t_k^+) = x(t_k) \quad \forall t_k \in S \tag{3.2b}$$

La figure 3.2 représente l'architecture détaillée du clone, correspondant au système d'équations (3.2). Les flèches verticales marquées de t_k en dessous des paramètres $\hat{a}_k$ et $\hat{b}_k$ du clone montrent que ces derniers sont modifiés à chaque t_k.

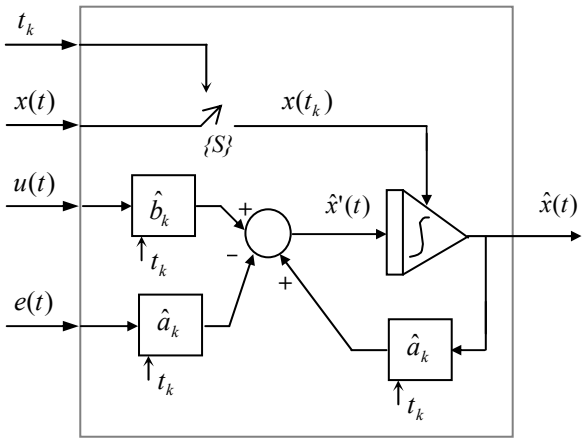


Fig. 3.2. *Architecture détaillée du clone*

En reprenant le formalisme des SLCM, cette description peut être synthétisée par le quintuplet $\Sigma_c(S, \hat{a}_k, \hat{\varphi}_k^T, I_n, I_n)$, en prenant pour entrée continue le vecteur $\left[-e^T(t) \quad u^T(t)\right]^T$ et pour entrée à discrétiser l'état du processus $x(t)$.

Propriété 3.1. *(Propriété de synchronisation du clone) La propriété fondamentale du clone est que son vecteur d'état, $\hat{x}(t)$, est forcé à la valeur de celui du processus à chaque instant de commutation par synchronisation (3.2b). Par conséquent, l'erreur d'état s'annule à chacun de ces instants ($e(t_k^+) = 0 \ \forall t_k$) comme le montre la figure 3.3.*

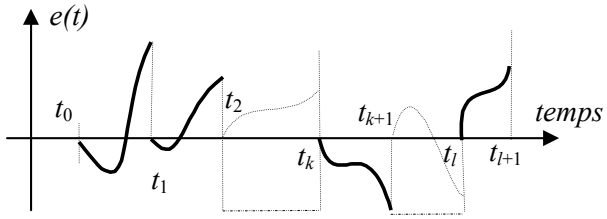


Fig. 3.3. *Erreur d'état*

En remplaçant $e(t)$ par sa valeur (définition 3.4), le système (3.2) devient :

$$\hat{x}'(t) = \hat{a}_k.x(t) + \hat{b}_k.u(t) \quad \forall t \notin S \tag{3.3a}$$

$$\hat{x}(t_k^+) = x(t_k) \quad \forall t_k \in S \tag{3.3b}$$

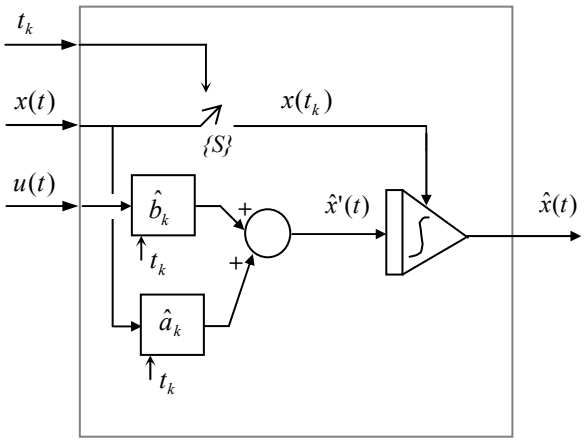


Fig. 3.4. *Structure fonctionnelle du clone*

Nous pouvons donc proposer une construction plus simple du clone en figure 3.4

Cette nouvelle écriture montre que l'état du processus intervient sur l'évolution de celui de son clone :

- de manière continue sur le morceau k (3.3a),
- par synchronisation à chaque instant de commutation (3.3b).

3.5. Approche mathématique

Rappelons que les états du processus et de son clone sont continus, bornés et dérivables entre les instants de commutation, ce qui implique que l'erreur d'état $e(t)$ y est également continue, bornée et dérivable.

Théorème 3.1. *Si $u(t)$ présente des discontinuités à l'intérieur d'un morceau k, le clonage et l'identification sont réalisables ssi $e(t)$ est rendue identiquement nulle $\forall t \to \infty$.*

Démonstration

En ce qui concerne le clonage, il est évident que si $e(t) \equiv 0 \ \ \forall t \to \infty$, alors, le clone reproduit exactement l'état du processus. Quant à l'identification, nous procédons comme suit pour montrer comment elle est obtenue en partant de la condition $e(t) \equiv 0 \ \ \forall t \to \infty$:

Les équations (3.1) et (3.3a) mènent à :

$$e'(t) = [\hat{a}_k - a]x(t) + [\hat{b}_k - b]u(t) \ \ \forall t \notin S \tag{3.4}$$

Réaliser $e(t) \equiv 0 \ \ \forall t \to \infty$ implique que $e'(t) \equiv 0 \ \forall t \to \infty$. Par conséquent :

$$[\hat{a}_k - a]x(t) + [\hat{b}_k - b]u(t) \equiv 0 \ \ \forall t \notin S \mid k \to \infty \tag{3.5}$$

L'algorithme adaptatif modifie les paramètres $\hat{a}_k$ et $\hat{b}_k$ de manière à satisfaire (3.5) pour toute paire de valeurs prises par $u(t)$ et $x(t)$ et tel que $\hat{a}_k$ et $\hat{b}_k$ soient constantes sur un morceau k. Sachant que $u(t)$ est choisie comme présentant des discontinuités sur chaque morceau k et que $x(t)$ est forcément *continu* (selon (3.1)), les seules valeurs de $\hat{a}_k$ et $\hat{b}_k$ qui peuvent satisfaire (3.5) sont nécessairement celles qui correspondent aux paramètres a et b du système.

Nous pouvons donc, dans les conditions du théorème 3.1, affirmer que : $e(t) \equiv 0 \Leftrightarrow \hat{a}_k \equiv a$ et $\hat{b}_k \equiv b$, garantissant ainsi l'identification du processus.

Remarque 3.1. *En pratique, pour l'identification, les méthodes classiques* [ADG+98] *nécessitent une entrée dite « riche » pour garantir un résultat satisfaisant. Cela se comprend par analogie avec un conducteur qui, pour mieux « connaître » le comportement de son véhicule, le teste dans des courbures et des pentes (trajectoires « riches »), plutôt qu'à vitesse constante en ligne droite. Dans notre cas, la condition de discontinuité de $u(t)$ à l'intérieur de chaque morceau k dans le théorème 3.1 peut être réalisée en pratique en choisissant un $u(t)$ généré à partir d'un SBE ou d'un SCM à cadencement plus rapide que celle du clone.*

3.5.1. Annulation de l'erreur d'état

La propriété 3.1 garantit seulement l'annulation de l'erreur d'état aux instants de commutation ($e_k = 0$). L'enjeu est donc de la rendre identiquement nulle à *l'intérieur* d'un morceau k pour aboutir à une identification suivant le théorème 3.1. Pour ce faire, nous utilisons la loi adaptative qui joue sur les paramètres du clone jusqu'à ce que $e(t)$ devienne identiquement nulle quand $t \to \infty$.

Stratégie. *Afin d'annuler l'erreur d'état, nous envisageons une méthode qui annule chacune de ses composantes indépendamment. Pour cela, nous introduisons, pour chaque morceau k, la grandeur* ε_k^i*, définie comme suit :*

Définition 3.5. $\varepsilon_k^i = \int_{t_k}^{t_{k+1}} \left| e^i(\theta) \right| .d\theta, \ \forall i = 1..., n$, $e^i(t)$ *étant la* $i^{ème}$ *composante de* $e(t)$.

ε_k^i est donc un scalaire.

Propriété 3.2. *Si l'intégrale d'une fonction du temps* $f(t)$ *bornée et positive sur un intervalle de temps (morceau* k*) est nulle, alors* $f(t)$ *est identiquement nulle sur cet intervalle de temps.*

Dans notre cas, $f(t) = \left| e^i(t) \right|$. Donc, si nous parvenons à réaliser $\varepsilon_k^i = 0 \ (k \to \infty) \ \forall i$, alors $e^i(t) \equiv 0 \ \forall i, \forall t \notin S$ suivant la propriété 3.2. De plus, $e(t)$ étant également nulle aux instants de commutation, nous avons :

$$\varepsilon_k^i = 0 \ (k \to \infty) \ \forall i \Leftrightarrow e^i(t) \equiv 0 \ \forall i, \forall t$$

3.5.2. Procédure adaptative

En suivant cette stratégie, nous allons définir une loi adaptative (semblable à celle utilisée dans [NA89]) permettant d'annuler ε_k^i de sorte à réaliser *simultanément* le clonage ($\hat{x}(t) \xrightarrow[k \to \infty]{} x(t)$) et l'identification ($\hat{\varphi}_k \xrightarrow[k \to \infty]{} \varphi$). Dans ce paragraphe, nous exprimons d'abord ε_k^i dans une forme qui convienne à l'utilisation de la loi adaptative, et ensuite nous définissons celle-ci.

Définition 3.6. *Les* $i^{ème}$ *colonnes de* φ, $\hat{\varphi}_k$ *et* Φ_k *sont désignées respectivement par* φ^i, $\hat{\varphi}_k^i$ *et* Φ_k^i. *Ainsi,* ${\Phi_k}^i = \hat{\varphi}_k^i - \varphi^i$.

3.5.2.1. Expression de ε_k^i

Soit le vecteur $W(t) \in \Re^{(n+r)}$ tel que $W^T(t) = [x^T(t), u^T(t)]$. Dans la littérature [Lan88], ce vecteur est appelé *vecteur d'observation*. En se basant sur les définitions de φ, $\hat{\varphi}_k$ et Φ_k données dans la section 3.2, nous pouvons exprimer (3.4) comme suit :

$$e'(t) = (\hat{\varphi}_k^T - \varphi^T).W(t) = \Phi_k^T.W(t) \quad \forall t \in]t_k, t_{k+1}] \tag{3.6}$$

L'intégration de (3.6) sur le morceau k donne :

$$e(t) = e(t_k^+) + \int_{t_k}^{t} \Phi_k^T.W(\theta).d\theta \quad \forall t \in]t_k, t_{k+1}] \tag{3.7}$$

En tenant en compte du fait que $e(t_k^+) = 0$ (propriété 3.1), et que la matrice Φ_k^T est constante sur un morceau k, nous pouvons écrire $e(t)$ sur un morceau k :

$$e(t) = \Phi_k^T.\int_{t_k}^{t} W(\theta).d\theta \quad \forall t \in]t_k, t_{k+1}] \tag{3.8}$$

Donc, l'erreur d'état est de la forme :

$$e(t) = \Phi_k^T.\widetilde{W}(t) \text{ avec } \widetilde{W}(t) = \int_{t_k}^{t} W(\theta).d\theta\ ,\ \forall t \in]t_k, t_{k+1}] \tag{3.9}$$

Afin d'évaluer $\varepsilon_k^i = \int_{t_k}^{t_{k+1}} \left| e^i(\theta) \right|.d\theta$, nous exprimons la $i^{\text{ème}}$ composante du vecteur $e(t)$:

$$e^i(t) = \widetilde{W}^T(t).\Phi_k^i \quad \forall t \in]t_k, t_{k+1}] \tag{3.10}$$

À l'intérieur d'un morceau k, nous pouvons donc écrire ε_k^i comme :

$$\varepsilon_k^i = \int_{t_k}^{t_{k+1}} \left\{ signe[e^i(\theta)] \right\}.e^i(\theta).d\theta \text{ pour } i = 1, \cdots, n \tag{3.11}$$

avec $$signe(f) = \begin{cases} 1 & \text{si } f \geq 0 \\ -1 & \text{autrement} \end{cases}$$

En remplaçant l'équation (3.10) dans la dernière, nous obtenons :

$$\varepsilon_k^i = \left\{ \int_{t_k}^{t_{k+1}} \left\{ sign[e^i(\theta)] \right\}.\widetilde{W}^T(\theta).d\theta \right\}.\Phi_k^i \text{ pour } i = 1, \cdots, n \tag{3.12}$$

Enfin, en posant $V_k^i = \int_{t_k}^{t_{k+1}} \left\{ sign[e^i(\theta)] \right\}.\widetilde{W}(\theta).d\theta$ pour $i = 1, \cdots, n$, nous pouvons définir :

$$\varepsilon_k^i = V_k^{iT}.\Phi_k{}^i \tag{3.13}$$

3.5.2.2. La loi adaptative

Avec la nouvelle expression de ε_k^i, nous définissons la loi adaptative pour chacun de ses composantes par la récurrence suivante :

$$\hat{\varphi}_{k+1}^i = \hat{\varphi}_k^i - \frac{g_k^i . V_k^i . \varepsilon_k^i}{1 + g_k^i . {V_k^i}^T . V_k^i} \text{ pour } i = 1, \cdots, n \tag{3.14}$$

où g_k^i est un gain scalaire réel et positif qui influence la convergence de la récurrence (3.14).

Les expressions (3.13) et (3.14) constituent la procédure récursive utilisée en pratique pour réaliser le clonage. Nous allons montrer ci-après que la récurrence (3.14) permet d'obtenir simultanément :

- $\varepsilon_k^i \xrightarrow[k \to \infty]{} 0$ et
- $\hat{\varphi}_k^i \xrightarrow[k \to \infty]{} \varphi^i$,

garantissant respectivement le clonage et l'identification du processus.

Remarque 3.2. *Le fait d'annuler le vecteur d'erreur d'état $e(t)$ composante par composante revient à construire n blocs adaptatifs distincts permettant de régler les paramètres de la matrice $\hat{\varphi}_k$ colonne par colonne, dans le cas d'une identification boîte noire. Bien entendu, si une ou plusieurs colonnes de φ sont déjà connue(s) (identification boîte grise), alors la structure de clonage peut être simplifiée en utilisant un nombre réduit de blocs récurrents. Par ailleurs, le fait d'utiliser des blocs récurrents séparés nous permet d'accélérer le procédé de clonage en faisant du parallélisme.*

3.5.3. Démonstration de la convergence de l'algorithme de clonage

Le but de ce paragraphe est de montrer que $\varepsilon_k^i \xrightarrow[k \to \infty]{} 0$. Pour ce faire, nous considérons la suite ${\Phi_k^i}^T . \Phi_k^i$, dont les termes sont nécessairement positifs et scalaires (${\Phi_k^i}^T . \Phi_k^i$ est la norme Euclidienne de Φ_k^i).

Remplaçons tout d'abord dans la récurrence (4.14), l'expression de ε_k^i donnée en (3.13) :

$$\hat{\varphi}_{k+1}^i = \hat{\varphi}_k^i - \frac{g_k^i . V_k^i . {V_k^i}^T}{1 + g_k^i . {V_k^i}^T . V_k^i} \Phi_k^i \text{ pour } i = 1, \cdots, n \tag{3.15}$$

Avec l'hypothèse que le processus est à paramètres invariants (au moins pendant la durée du clonage : voir ci après §3.6.1.3), nous pouvons soustraire φ^i des deux côtés de (3.15), ce qui donne :

$$\Phi_{k+1}^i = \left[I - \frac{g_k^i . V_k^i . V_k^{i\,T}}{1 + g_k^i . V_k^{i\,T} . V_k^i} \right] \Phi_k^i \text{ pour } i = 1, \cdots, n \tag{3.16}$$

La relation (3.16) est de la forme :

$$\Phi_{k+1}^i = [I - S_k^i] . \Phi_k^i \text{ avec } S_k^i = \frac{g_k^i . V_k^i . V_k^{i\,T}}{1 + g_k^i . V_k^{i\,T} . V_k^i} \text{ pour } i = 1, \cdots, n \tag{3.17}$$

Il est clair que $S_k^{i\,T} = S_k^i$ et de plus :

$$S_k^{i\,T} . S_k^i = \frac{g_k^i . V_k^{i\,T} . V_k^i}{1 + g_k^i . V_k^{i\,T} . V_k^i} S_k^i = \mu_k^i . S_k^i \text{ pour } i = 1, \cdots, n \tag{3.18}$$

avec $$\mu_k^i = \frac{g_k^i . V_k^{i\,T} . V_k^i}{1 + g_k^i . V_k^{i\,T} . V_k^i} \tag{3.19}$$

Nous pouvons constater d'après (3.19) que $0 \le \mu_k^i < 1$.

Analysons maintenant l'évolution du terme scalaire $\Phi_k^{i\,T} . \Phi_k^i$:

$$\Phi_{k+1}^{i\,T} . \Phi_{k+1}^i = \Phi_k^{i\,T} [I - S_k^i]^T . [I - S_k^i] \Phi_{k+1}^i \text{ pour } i = 1, \cdots, n \tag{3.20}$$

Selon (3.17) et (3.18) :

$$\Phi_{k+1}^{i\,T} . \Phi_{k+1}^i = \Phi_k^{i\,T} . \Phi_k^i - (2 - \mu_k^i) . \Phi_k^{i\,T} . S_k^i . \Phi_k^i \text{ pour } i = 1, \cdots, n \tag{3.21}$$

Ce qui implique que :

$$\Phi_{k+1}^{i\,T} . \Phi_{k+1}^i = \Phi_k^{i\,T} . \Phi_k^i - (2 - \mu_k^i) . \Phi_k^{i\,T} \frac{g_k^i . V_k^i . V_k^{i\,T}}{1 + g_k^i . V_k^{i\,T} . V_k^i} \Phi_k^i \text{ pour } i = 1, \cdots, n \tag{3.22}$$

En se référant à l'expression (3.13), nous pouvons faire apparaître le scalaire ε_k^i :

$$\Phi_{k+1}^{i\,T}.\Phi_{k+1}^{i} = \Phi_{k}^{i\,T}.\Phi_{k}^{i} - (2-\mu_k^i)\frac{g_k^i.\varepsilon_k^{i\,2}}{1+g_k^i.V_k^{i\,T}.V_k^i} \text{ pour } i=1,\cdots,n \quad (3.23)$$

Ceci nous donne :

$$\Phi_{k+1}^{i\,T}.\Phi_{k+1}^{i} = \Phi_{k}^{i\,T}.\Phi_{k}^{i} - g_k^i.(2-\mu_k^i).(1-\mu_k^i).\varepsilon_k^{i\,2} \text{ pour } i=1,\cdots,n \quad (3.24)$$

Nous pouvons ainsi dire que la différence entre deux termes successifs de la suite est de la forme :

$$\Phi_{k+1}^{i\,T}.\Phi_{k+1}^{i} - \Phi_{k}^{i\,T}.\Phi_{k}^{i} = -\gamma_k^i.\varepsilon_k^{i\,2} \text{ pour } i=1,\cdots,n \quad (3.25)$$

avec $\gamma_k^i = g_k^i(2-\mu_k^i)(1-\mu_k^i)$.

Par définition, $g_k^i > 0$ et $0 \le \mu_k^i < 1$ (d'après (3.19)). Il est donc évident que :

$$\gamma_k^i = g_k^i(2-\mu_k^i)(1-\mu_k^i) > 0 \text{ pour } i=1,\cdots,n \quad (3.26)$$

Par conséquent : $\Phi_{k+1}^{i\,T}.\Phi_{k+1}^{i} - \Phi_{k}^{i\,T}.\Phi_{k}^{i} \le 0$ et $\Phi_{k}^{i\,T}.\Phi_{k}^{i} \xrightarrow[k\to\infty]{} cst$.

Interprétation. *Sachant que les termes* $\Phi_{k}^{i\,T}.\Phi_{k}^{i}$ *sont des scalaires positifs, et que* $\Phi_{k+1}^{i\,T}.\Phi_{k+1}^{i} - \Phi_{k}^{i\,T}.\Phi_{k}^{i} \le 0$, *nous pouvons dire que* $\Phi_{k}^{i\,T}.\Phi_{k}^{i}$ *est une suite positive et décroissante. Sa limite quand k tend vers l'infini est donc une constante positive ou nulle. Quand cette limite est atteinte,* $\Phi_{k+1}^{i\,T}.\Phi_{k+1}^{i} - \Phi_{k}^{i\,T}.\Phi_{k}^{i} \xrightarrow[k\to\infty]{} 0$. *Par conséquent, selon (3.25),* $-\gamma_k^i.\varepsilon_k^{i\,2} \xrightarrow[k\to\infty]{} 0$. *De plus, comme* $\gamma_k^i \neq 0$ *d'après (3.26), nous pouvons affirmer que* $\varepsilon_k^i \xrightarrow[k\to\infty]{} 0$.

Comme nous l'avons vu précédemment, ce résultat garantit que $e(t)$ tend à être identiquement nul et que donc, d'après le théorème 3.1, le clonage et l'identification sont réalisés.

3.5.4. Implantation de la méthode

La méthode est implantable directement sur simulateur numérique, en utilisant le formalisme SFM défini au chapitre 2. Les blocs fonctionnels et leurs équations ont été réalisés dans Simulink® pour tester les performances de la méthode (cf. paragraphe suivant).

L'algorithme relatif à cette méthode est donné en annexe III.

3.6. Validation expérimentale

Plusieurs essais réalisés en simulation ont permis d'une part de valider la mise en œuvre de la méthode de clonage, et d'autre part d'en tester l'efficacité. À partir de la figure 3.1 et des équations propres à la méthode, nous avons construit une structure fonctionnel de clonage (SFC) en Simulink®, dont le détail figure en annexe III.

Dans cette section, nous avons sélectionné certains exemples qui démontrent l'efficacité de la méthode, non seulement par rapport au cahier des charges, mais également dans des situations où les hypothèses théoriques sont transgressées[5]. Nous apportons des explications au fait que la méthode est applicable pour certaines de ces transgressions. Aussi, une comparaison avec une méthode existante est proposée. Enfin, nous avons pu cloner un moteur réel par le biais de la technologie temps réel Matlab®RTW/Simulink/dSpace® et nous donnons les résultats de l'identification.

Remarque 3.3. *Dans tous les exemples, le gain adaptatif* g_k^i *est suffisamment grand* $\forall i$ *pour assurer une convergence rapide de l'algorithme. En occurrence, nous avons choisi* $g_k^i = g^i = 1\times 10^{20} \quad \forall i$. *Même si la converge s'accélère avec de grandes valeurs de ce gain, il faut néanmoins s'assurer que le simulateur numérique ne sature pas.*

Remarque 3.4. *Dans les exemples, nous avons défini les instants de commutation du clone par* $S = \{k.T, k = 0,1,2...\}$, *c'est-à-dire à période constante.*

Remarque 3.5. *Afin de réaliser le clonage, nous avons appliqué, dans chaque cas, une entrée* $u(t)$ *présentant des discontinuités dans l'intervalle* $]kT,(k+1)T]$, *comme le requiert le théorème 3.1.*

Il est évident que si $u(t)$ *reste constante sur un morceau* k, *il existera d'autres solutions de* $\hat{a}_k$ *et* $\hat{b}_k$ *qui satisfassent (3.5) hors* $\hat{a}_k \equiv a$ *et* $\hat{b}_k \equiv b$. *En pratique, nous constatons alors que la méthode a tendance à faire stagner les estimateurs des paramètres sur des valeurs erronées. D'où la nécessité de la condition du théorème 3.1.*

Par ailleurs, nous avons constaté en pratique que plus il y a de discontinuités dans cet intervalle, plus vite l'algorithme converge vers les résultats.

3.6.1. Le clonage en simulation

Les essais en simulations ont permis de cloner des processus linéaires invariant de n'importe quel ordre et de faire une identification *boîte noire*. Nous présentons, ici, l'identification de divers types de systèmes, en appliquant, là où c'est possible, les simpli-

[5] Pour se rapprocher des cas réels.

fications correspondant à l'approche *boîte grise*. Certains exemples comportent des transgressions par rapport aux suppositions théoriques.

3.6.1.1. Processus invariant d'ordre 2

Cet exemple illustre l'identification *boîte noire* d'un système linéaire. Toutes les hypothèses théoriques y sont respectées. Le processus simulé est un système d'ordre 2. Son modèle (3.1) à identifier est défini par :

$$a = \begin{bmatrix} -3 & 1 \\ 5 & 4 \end{bmatrix} \text{ et } b = \begin{bmatrix} 30 \\ 90 \end{bmatrix}, \text{ avec } n = 2 \text{ et } r = 1.$$

Pour cette simulation, la condition initiale de la SFC est $\hat{\varphi}_0^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ et la période de commutation du clone est $T = 100\text{ms}$.

Les résultats du clonage en figure 3.7 montrent que les estimateurs convergent rapidement (en moins de 5s) vers les paramètres du processus. Les composantes du vecteur d'erreur d'état tendent simultanément vers zéro.

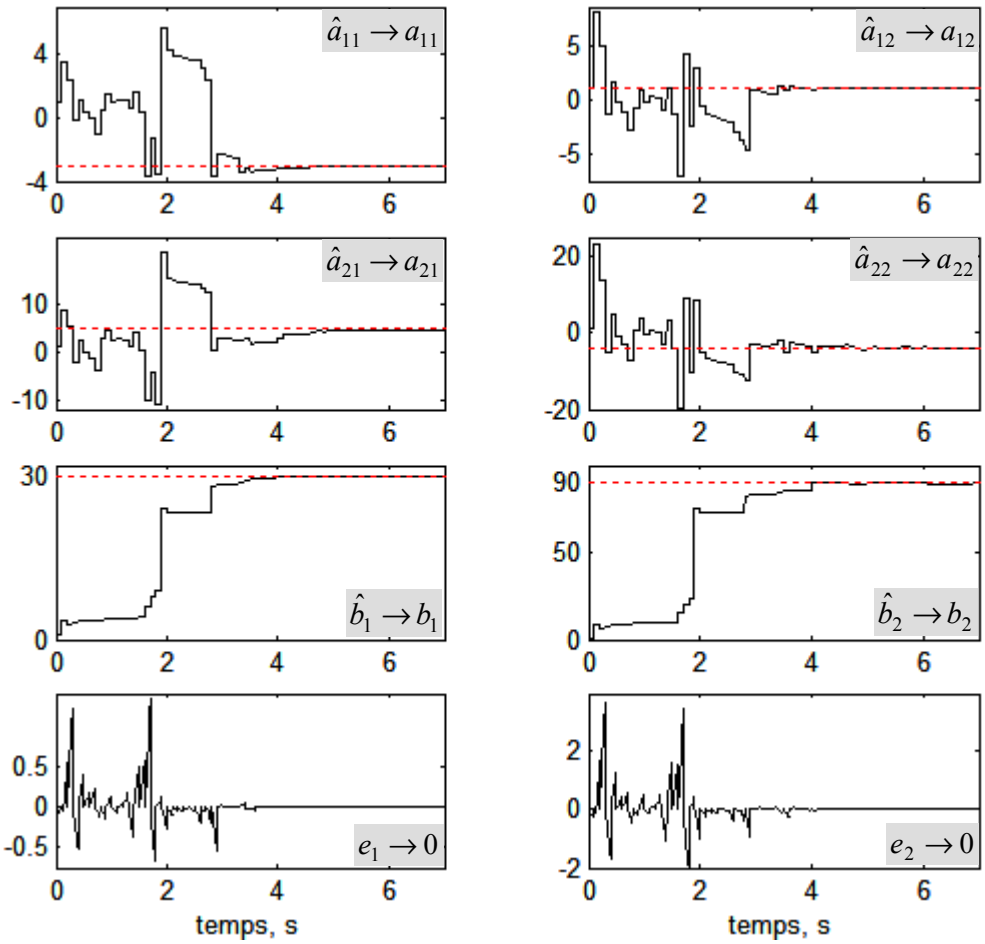


Fig. 3.7. *Clonage du système d'ordre 2*

Remarque 3.6. *Nous avons effectué le clonage de plusieurs systèmes en simulation. La méthode converge vers les « bonnes » valeurs* $\forall n$ *et quelque soit la condition initiale du clone.*

3.6.1.2. Processus invariant d'ordre 3 avec perturbations

Dans cet exemple, nous identifions le système considéré dans [HV69]. Le modèle (3.1) à identifier est dans ce cas, défini par :

$$a = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -4.5 & -10.5 & -3.5 \end{bmatrix} \text{ et } b = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}, \text{ avec } n = 3 \text{ et } r = 1.$$

Nous avons utilisé ces valeurs pour simuler le processus, et avons réaliser son clonage en présence d'un bruit blanc, proportionnel à l'état, rajouté à l'état lui-même, de sorte à tester la robustesse de la méthode face à des perturbations (figure 3.8).

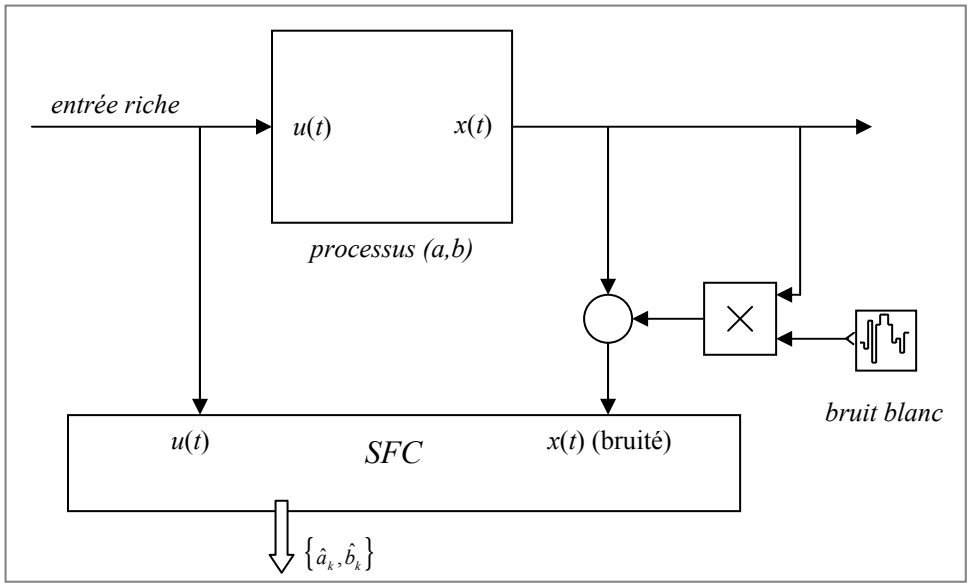


Fig. 3.8. *Schéma de réalisation du clonage avec l'état bruité*

Pour cet exemple, compte tenu des structures des matrices *a* et *b*, il n'y a pas lieu d'identifier toutes les lignes de φ^T, puisque nous avons une connaissance *a priori* de la structure interne du METC du processus. Nous pouvons donc procéder à une identification *boîte grise*, nécessitant seulement un bloc adaptatif qui joue sur la troisième ligne de $\hat{\varphi}_k^T$.

Pour cette simulation, la condition initiale de la SFC est $\hat{\varphi}_0^{3^T} = [1 \quad 1 \quad 1 \quad 1]$ et la période de commutation du clone est $T = 100\text{ms}$.

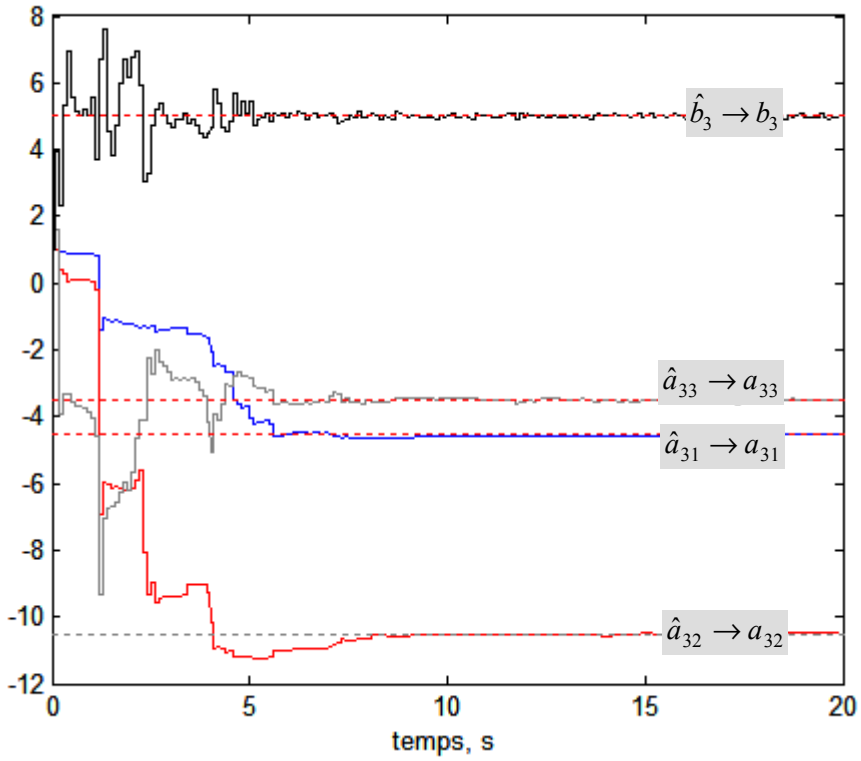


Fig. 3.9. *Identification du système d'ordre 3*

La figure 3.9 montre comment les estimateurs évoluent depuis leur condition initiale pour tendre vers la valeur du paramètre correspondant (en pointillé) à la fin du clonage. Nous pouvons également constater (figure 3.10) que parallèlement, le clonage fait tendre la troisième composante du vecteur d'erreur d'état vers zéro, même si l'état est bruité.

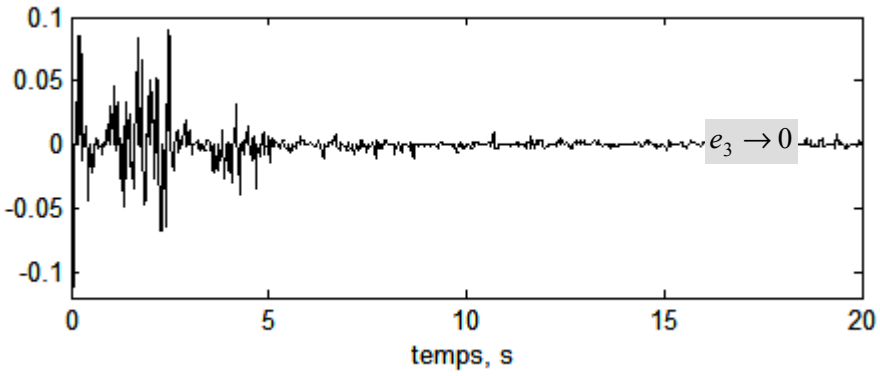


Fig. 3.10. *Annulation de la 3ème composante de l'erreur d'état*

Nous constatons qu'ici, l'algorithme converge en moins de 10s alors que la méthode proposée en [HV69] demande plus de 150s. Celle en [ADG$^+$98] converge au bout de 20s pour un système équivalent. En fait, ces deux méthodes sont semblables à la notre, sauf que la méthode du clonage permet une accélération du processus grâce à la propriété 3.1 (commutation du clone), alors que les techniques dans [HV69, ADG$^+$98] réalisent une adaptation des paramètres en continu.

3.6.1.3. Processus à paramètres variants

Même si la méthode du clonage suppose théoriquement que les processus à identifier soient invariants dans le temps, nous observons expérimentalement qu'elle peut aussi bien identifier des paramètres changeants (TVP).

3.6.1.3.1. Paramètres constants par morceaux

Considérons un processus dont les paramètres, variants temporellement, sont constants par morceaux, et encourent une discontinuité après chaque morceau (la figure 3.11 illustre ce cas pour un paramètre du système).

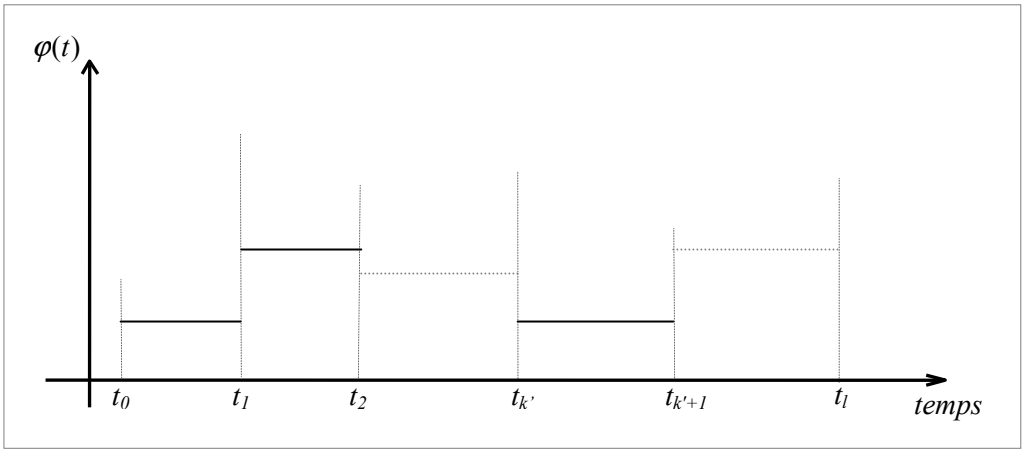


Fig. 3.11. *Paramètre constant par morceaux*

Dans ce cas, nous pouvons dire que le clonage est possible si chaque morceau $]t_{k'}, t_{k'+1}]$ est suffisamment long pour que l'algorithme converge. En effet, nous pouvons considérer qu'à chaque instant $t_{k'}$, une nouvelle procédure de clonage démarre avec une nouvelle condition initiale qui est en fait la valeur de l'estimateur, et donc du paramètre en cas de réussite du clonage, sur le morceau précédent.

Nous justifions cette hypothèse expérimentalement en clonant, en simulation, un processus d'ordre 2, dont le paramètre a_{11} du METC varie de manière constante par morceaux. Nous définissons ce paramètre par : $a_{11}(t) = -3 + f(t)$, $f(t)$ étant un signal carré d'amplitude 3 et de période 0.5s.

Les autres paramètres étant considérés constant et connus, nous avons réalisé une identification *boîte grise* avec un bloc adaptatif. La condition initiale du clone a été placée loin de l'intervalle de valeurs prises par $a_{11}(t)$: $(\hat{a}_{11})_0 = -6{,}5$. Afin de rendre le clonage plus « réactif », nous avons choisi une commutation du clone à $T = 5\,\text{ms}$.

Les résultats de la figure 3.12 montrent que l'estimateur suit la variation discontinue du paramètre correspondant, après une courte phase transitoire (100ms), même avec une condition initiale éloignée. Nous constatons également que le suivi qui se fait à chaque

changement discontinu équivaut à un redémarrage d'un nouveau processus de clonage avec une condition initiale égale à la valeur identifié sur le morceau précédent (comme décrit précédemment).

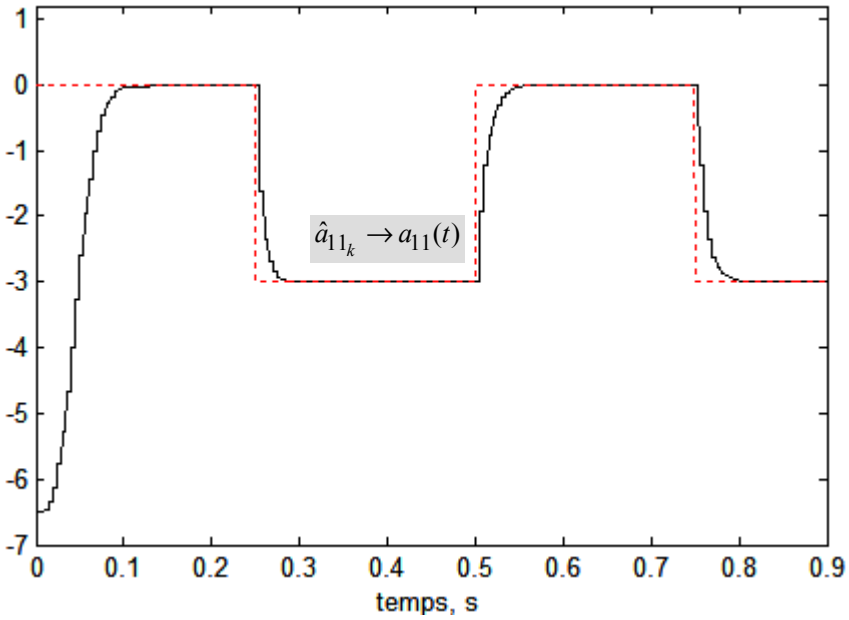


Fig. 3.12. *Identification d'un paramètre variant discontinûment*

La courte phase transitoire à chaque changement met en valeur la puissance de convergence de l'algorithme, surtout avec une valeur de T relativement faible, comme ici.

3.6.1.3.2. Paramètres variant continûment

Le modèle que nous voulons identifier dans cet exemple est défini par $x'(t) = a(t).x(t) + b.u(t)$ avec :

$$a(t) = \begin{bmatrix} -3+\sin(2\pi.t) & 1 \\ 5 & -4+2\sin(4\pi.t) \end{bmatrix} \text{ et } b = \begin{bmatrix} 30 \\ 90 \end{bmatrix}.$$

Dans ce cas, nous avons procédé à une identification *boîte grise*, utilisant deux blocs adaptatifs pour identifier a_{11} et a_{22} (les paramètres variants temporellement). Les conditions initiales des estimateurs sont très éloignées des domaines de variation des paramètres : $(\hat{a}_{11})_0 = (\hat{a}_{22})_0 = 1$. Par ailleurs, la commutation du clone se fait à période $T = 5$ ms, pour accélérer le processus de clonage, comme dans le cas précédent.

Les résultats de la figure 3.13 montrent que les estimateurs suivent la variation des paramètres correspondant, après une phase de transition très courte (inférieure à 200ms), liée aux conditions initiales du clone.

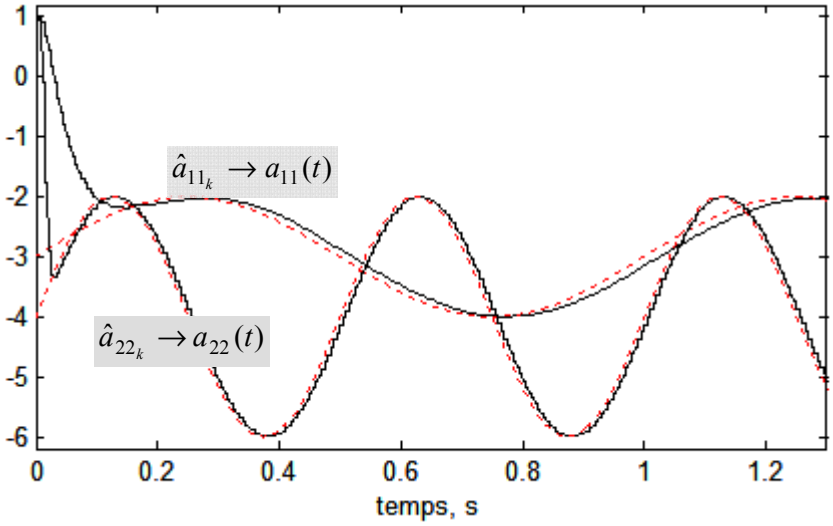


Fig. 3.13. *Identification de paramètres variant continûment*

On constate qu'à partir du moment où $\hat{\varphi}_k$ a rejoint la valeur de $\varphi(t)$ après une phase transitoire, les nouvelles procédures de clonage se déroulent avec des conditions initiales du clone très proches des valeurs courantes des paramètres à identifier. Toutefois, la dynamique de changement des paramètres doit être lente par rapport à celle du clone pour que l'identification demeure satisfaisante.

Remarque 3.7. *Nous avons observé que l'identification de paramètres variant conduit à une perte de la faculté boîte noire de la méthode, car seulement un paramètre par ligne de* φ^T *doit être identifié pour de meilleurs résultats. Les autres paramètres doivent être fixés dans* $\hat{\varphi}_k$ *afin que l'algorithme ne les identifie pas comme des paramètres changeants de sorte à satisfaire globalement* $\hat{\varphi}_k^i = \varphi^i$. *Toutefois, la performance boîte noire reste valable dans le cas où les paramètres du processus présentent des changements discontinus synchronisés (comme par exemple, un système multimodèle).*

3.6.2. Le clonage d'un système réel

Les performances temps réel ont été testées sur la maquette décrite en annexe I. La partie qui nous intéresse correspond au moteur électrique qui sert à déplacer un chariot sur un axe horizontal et rectiligne. Ce chariot, relié au moteur par le biais d'une courroie crantée, parcours un tronçon fini de l'axe. Le montage est représenté en figure 3.14.

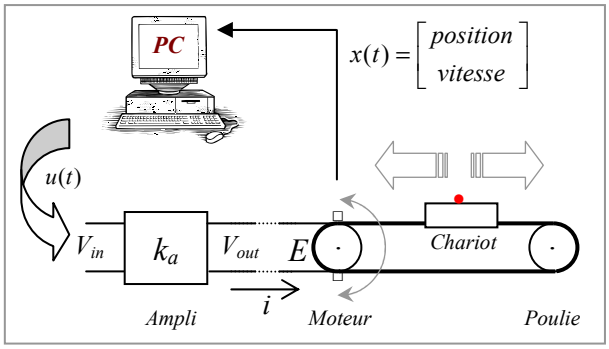


Fig. 3.14. *Montage temps réel*

Dans cet exemple, nous réalisons le clonage de l'ensemble amplificateur–moteur–chariot, afin d'en déterminer les paramètres d'un METC associé. Cet ensemble a déjà subi une phase d'identification hors ligne par des méthodes classiques qui ont révélé qu'il se comporte plus ou moins comme un système linéaire du second ordre par rapport à la position. Lors de son clonage, nous lui avons donc associé un modèle (selon (3.1)) défini par :

$$x(t) = \begin{bmatrix} position \\ vitesse \end{bmatrix},\ a = \begin{bmatrix} 0 & 1 \\ 0 & -1/\tau \end{bmatrix},\ b = \begin{bmatrix} 0 \\ K/\tau \end{bmatrix},$$

où τ est la constante de temps et K le gain statique de l'ensemble amplificateur-moteur-chariot.

Ayant cette connaissance *a priori* de la structure du processus, il est évident qu'il faut identifier seulement deux composantes de φ. Nous avons donc procédé à une identification *boîte grise* qui nécessite un seul bloc adaptatif pour régler les paramètres de la deuxième ligne.

La condition initiale adoptée pour le clone est $\hat{\varphi}_0^{2\,T} = [0 \quad 1 \quad 1]$ et les commutations se font à période $T = 100\text{ms}$.

La figure 3.15 montre comment chaque estimateur convergent vers sa valeur préalablement identifiée par des méthodes classiques hors ligne (analyse harmonique). De plus, l'erreur d'état sur la composante correspondante (vitesse) tend rapidement vers un voisi-

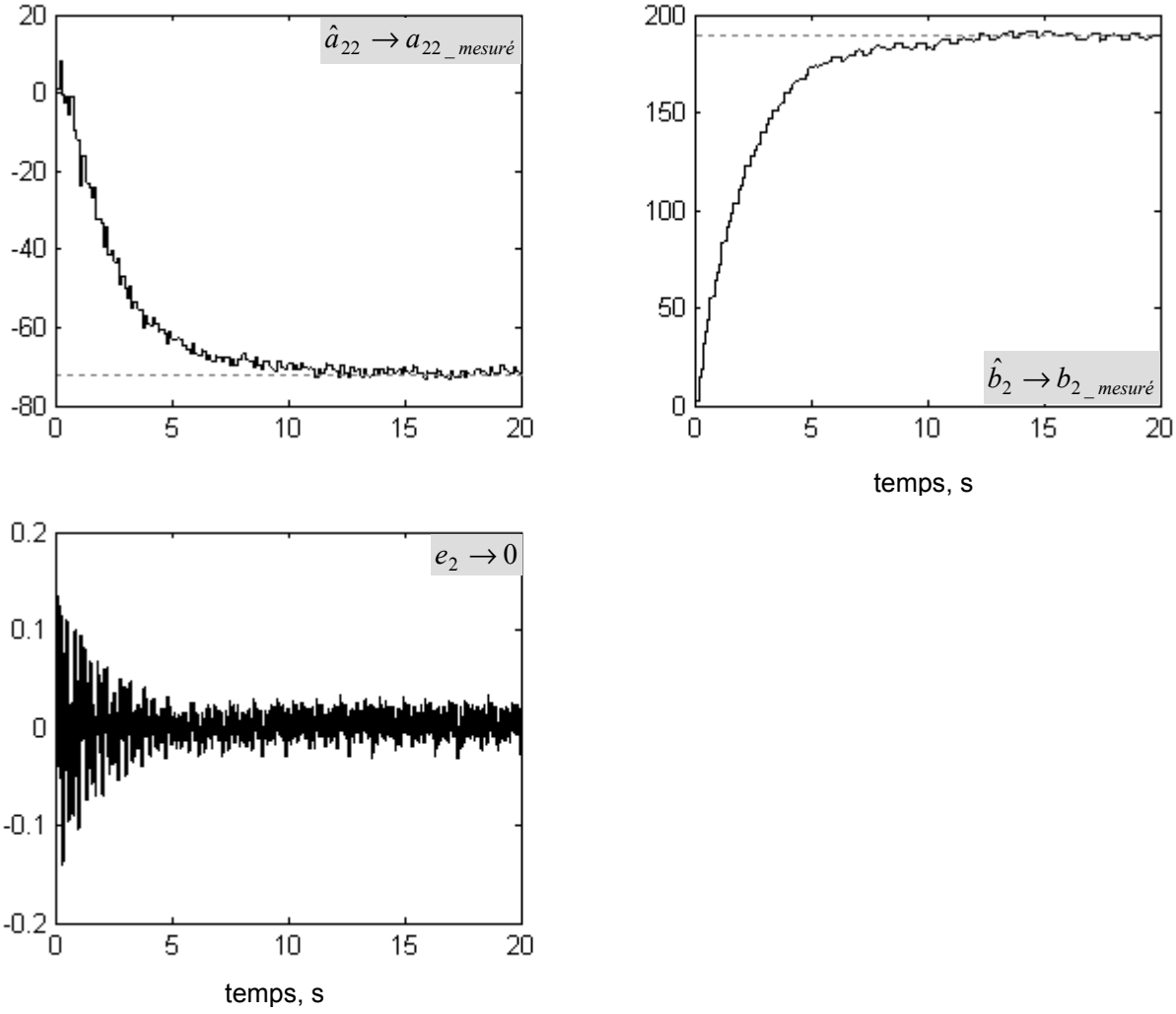


Fig. 3.15. *Clonage d'un processus réel*

nage de zéro, illustrant la performance du clonage en temps réel. À noter que les bruits sur les signaux sont dus à la qualité médiocre des capteurs. Ceci montre, néanmoins, que la méthode est robuste par rapport aux perturbations.

3.7. Conclusion

Nous avons défini au cours de ce chapitre, une nouvelle approche d'identification de processus basée sur les SFM. La méthode, qui est facilement implantable sur simulateur numérique, permet d'identifier rapidement en ligne les paramètres d'un processus linéaire.

Les essais en simulation et en temps réel ont montré que la méthode est robuste par rapport à des variations des paramètres du processus (de dynamique inférieure à celle du clone). De plus, ils ont illustré sa robustesse en présence de bruit sur la mesure d'état.

Cela s'explique par le fait que les calculs d'intégration de l'algorithme entre les instants de commutation apportent un effet de filtrage qui élimine une perturbation de type bruit blanc. Enfin, les essais ont montré une grande rapidité de convergence de l'algorithme par rapport à des approches semblables.

L'originalité de la méthode de clonage réside dans le modèle à commutation qui permet une convergence rapide. Ce modèle, inspiré des SFM, rend également possible l'implantation temps réel. De plus, il peut être intégré dans une architecture de commande SFM, telle que présentée à la fin du chapitre 4, ce qui permet d'envisager une commande adaptative (figure 3.16).

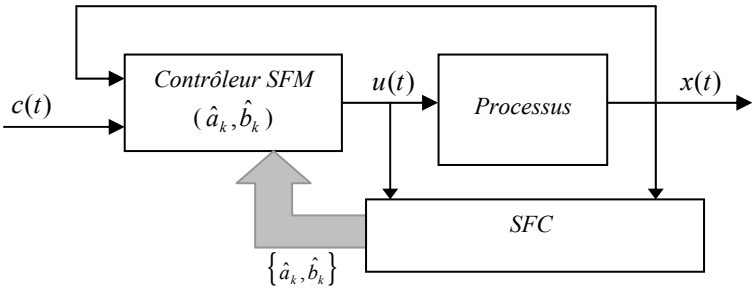


Fig. 3.16. *Clonage et commande adaptative*

L'approche proposée peut également être applicable dans le cas de processus linéaires présentant un retard pur, supposé connu, à l'entrée du processus. En effet, il suffit alors de reproduire le retard au sein du clone comme le montre la figure 3.17.

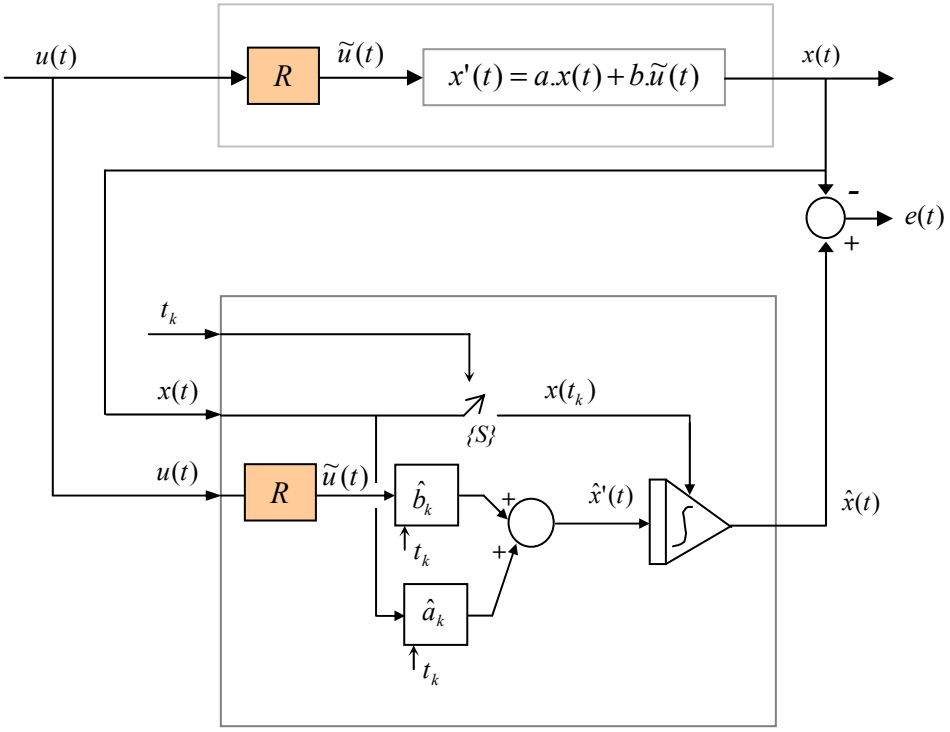


Fig. 3.17. *Clonage d'un système à retard pur*

Sur cette figure, R représente un retard connu et $\tilde{u}(t) = u(t-R)$. Tout se passe comme-ci le clonage était effectué pour la partie linéaire ($x'(t) = a.x(t) + b.\tilde{u}(t)$) du processus avec une entrée $\tilde{u}(t)$ présentant des discontinuités à l'intérieur d'un morceau k comme le requiert le théorème 3.1.

Chapitre 4

La poursuite échantillonnée par les SFM

Dans ce chapitre, nous proposons une architecture de commande particulière basée sur les SFM présentés au chapitre 2. Le type de commande développé utilise une approche par retour d'état pour contrôler des processus MIMO linéaires. L'enjeu est de paramétrer convenablement les blocs SLCM ou SLBE pour formaliser un « système associé » qui, placé en amont du processus à piloter, délivre la commande adaptée à imposer au processus. Nous réalisons, par cette approche, des contrôleurs à fonctionnement par morceaux (CFM) qui garantissent la poursuite échantillonnée d'une trajectoire d'état. Nous justifions dans un premier temps le choix de ce type de contrôle-commande et établissons ensuite les équations et la structure de l'ensemble du système bouclé. Par ailleurs, afin de prendre en compte des éventuelles discontinuités sur la consigne, nous formalisons un « amortisseur » qui permet d'éviter une commande excessivement grande. Enfin, nous proposons une structure de commande adaptative en combinant un CFM avec l'identificateur présenté chapitre 3. Des exemples de simulation et d'implantation temps réel illustrent la robustesse des contrôleurs face aux perturbations et aux variations des paramètres du processus.

4.1. Motivations

La notion d'asservissement est très importante dans la commande de processus. En effet, sauf dans le cas idéal où on possède un modèle parfait d'un système stable dépourvu de perturbations, la commande en boucle ouverte est rarement envisagée. Selon Kabamba [Kab87], la commande par feedback est généralement utilisée pour :

- stabiliser ou améliorer la stabilité d'un processus,
- donner à un processus un comportement entrée/sortie spécifique (concordance à un modèle, découplage, etc.),
- réduire la sensibilité d'un processus aux bruits et perturbations,
- garantir des performances satisfaisantes et une stabilité d'un processus même dans le cas où ses paramètres sont variants et/ou sa dynamique n'est pas modélisée.

Il est impossible de définir une loi de commande générique qui prend en compte tous les types de processus à contrôler. Le choix du type de contrôle dépend des particularités du système (linéaire/non linéaire, continu/échantillonnée, à retard, à paramètres variables, etc.), des informations disponibles pour le retour (disponibilité de l'état/de la sortie, forme continu/échantillonnée, présence d'un retard, etc.) et de la performance souhaitée. Par ailleurs, l'élaboration d'une stratégie de commande requiert, dans la plupart des cas, une modélisation et une identification du processus considéré.

Dans l'optique de commander des processus, de nombreux types d'asservissements ont déjà été proposés [Bod45, DH81, FH77, Hor63] pour les systèmes SISO invariants. Parmi les méthodes d'asservissement classiques, le régulateur PID est le plus fréquemment utilisé. Facile à mettre en œuvre, il permet de réaliser une commande qui offre un compromis de performances (rapidité, précision, stabilité) suivant ses trois actions comme le montre le tableau 4.1.

Action	Atout	Inconvénient
P	dynamique	ne permet pas d'annuler une erreur statique
I	annulation d'erreur statique, amélioration de la robustesse	action lente, ralentit le système

D	action très dynamique, améliore la rapidité	sensibilité aux bruits, forte sollicitation de l'organe de commande

Tab. 4.1. *Résumé des effets respectifs des actions* P*,* I*, et* D*.*

Néanmoins, la commande de procédés complexes a suscité de nombreuses investigations donnant lieu à des approches élaborées en fonction de la problématique considérée. Nous présentons ci-après une classification de diverses stratégies, selon les critères suivants :

- approche *représentation d'état* et approche *fonction de transfert*,
- modèle *continu* et modèle *échantillonné*,
- commande *analogique* et *numérique*,
- retour *d'état* et retour *de sortie*,
- asservissement pour la *régulation* et pour la *poursuite*.

4.1.1. Approche *représentation d'état* et approche *fonction de transfert*

Il existe deux approches principales pour aborder le développement de lois de commande. L'architecture de commande peut être construite soit à partir du modèle de la fonction de transfert du processus, soit à partir de son modèle d'état. L'approche par la fonction de transfert conduit à plusieurs techniques semblables au régulateur PID. Dans le cas discret, le régulateur R-S-T requiert la détermination des trois polynômes en z (RST), selon les performances voulues. Ces méthodes sont particulièrement adaptées aux cas SISO. Dans l'optique de généraliser un formalisme adaptable à des processus MIMO, les méthodes de contrôle s'appuient plutôt sur un modèle d'état du processus. Les méthodes les plus utilisées dans ce cas sont l'optimisation quadratique et le placement de pôles [Kai80], qui se basent sur le *retour d'état*.

Dans notre étude, afin de prendre en compte le cas général des processus MIMO, nous développons des contrôleurs utilisant la représentation d'état du processus à contrôler. Dans ce contexte, la méthode de clonage développée au chapitre 3 peut être utilisée pour identifier directement le modèle requis pour construire l'architecture de commande. À la fin du chapitre, nous envisageons également de combiner le contrôleur et l'identificateur en ligne pour proposer une structure de commande adaptative applicable à des systèmes à paramètres variants.

4.1.2. Modèle *continu* et modèle *échantillonné*

Le modèle du processus utilisé dans l'architecture de commande peut être de nature continue ou discrète. La modélisation discrète semble intuitive dans le cadre de la commande par calculateurs. Néanmoins, dans l'optique de contrôler des processus réels physiques, il est important de considérer qu'ils possèdent une dynamique continue, même si la communication avec eux par le biais d'un calculateur et de capteurs numériques nécessite une discrétisation. Ainsi, en partant d'un modèle continu du processus, nous pourrons considérer l'équation aux différences équivalente telle que le processus soit vu aux instants d'échantillonnage, comme un processus discret.

Remarque 4.1. *Même si les équations aux différences donnent l'équivalent discret d'un système continu aux instants d'échantillonnage, il faut bien noter que contrairement à un système purement discret, le processus continu possède bien une évolution entre ces instants. Certaines architectures de contrôle numérique tiennent compte de cette évolution* [UN87, KV01, KV02].

4.1.3. Commande numérique

Depuis des décennies, la venue des calculateurs numériques a incité les automaticiens à considérer la commande numérique. En adaptant les méthodes classiques basées sur l'approche de la fonction de transfert, des méthodes de commande numériques telles que le PID numérique[1], ont été développées [Vas89, AH00].

Remarque 4.2. *Selon Landau* [Lan88] *un régulateur PID numérique ne peut s'appliquer rigoureusement qu'aux procédés modélisables par un système continu caractérisé par une fonction de transfert de degré maximum égal à 2. Il précise aussi que ce régulateur peut prendre en compte un retard pur à condition qu'il soit inférieur à une période d'échantillonnage.*

Par ailleurs, la commande numérique offre d'autres avantages sur la commande analogique. La commande *deadbeat*, par exemple, apporte une stabilisation que les asservissements analogiques ne parviennent pas à réaliser [UN87, Yam94]. Il est montré dans [Kab87, HA88] que l'échantillonnage multiple et les GSHF (Generalised Sample data Hold Function) améliore les performances dans les architectures de contrôle. De plus, selon les auteurs de [KH93, Yam94], les systèmes hybrides représentent un formalisme prometteur pour améliorer la performance des contrôleurs basés sur un retour d'information échantillonnée–bloquée.

[1] Le régulateur PID numérique est basé sur le modèle de transfert en z

4.1.4. Retour *d'état* et retour *de sortie*

Afin d'asservir un processus, il est primordial que le système de commande *connaisse* son évolution de sorte à réagir en conséquence. Même si la sortie offre une observation du processus, la meilleure information sur son évolution reste, sans aucun doute, son état. Le retour d'état conduit donc à une meilleure architecture de commande, dans le sens où il permet de réaliser toutes les performances mentionnées précédemment dans le cas MIMO. Classiquement, ce type d'asservissement consiste à calculer une commande proportionnelle à l'état ($u(t) = K.x(t)$) de manière continue. Le calcul de $K \in \Re^{r \times n}$ reste toutefois assez délicat, car il n'existe pas de méthode déterministe généralisable pour tout processus. Il existe néanmoins d'autres méthodes telles que les LMI pour déterminer le gain de retour suivant les performances souhaitées et des particularités du processus.

Le type de contrôleur présenté dans ce mémoire adopte la notion de retour d'état, mais le principe est différent. En effet, le système de commande utilise, en temps réel, les signaux de consigne et de l'état du processus pour générer une commande à fonctionnement par morceaux à partir d'un SFM. Nous montrons comment seuls les paramètres du modèle d'état sont utilisés pour calculer de façon déterministe, une commande garantissant la poursuite échantillonnée de la consigne par l'état du processus.

En ce qui concerne les processus dont l'état n'est pas disponible, il est envisageable d'inclure un observateur dans la boucle de rétroaction afin de réaliser un feedback par une estimation de l'état. Cette notion repose sur la dualité de l'estimateur d'état et du contrôleur à retour d'état [Wol76, FH77, Won79, Kai80, DH81] et sur les principes de séparation visant à déterminer indépendamment les gains de l'estimateur et du contrôleur [Med69, Che70, BH75, SW77]. Toutefois, cet ensemble observateur-contrôleur dans la boucle de retour rend l'asservissement moins robuste[2] face aux perturbations et aux changements éventuels des paramètres du processus. En effet, les deux blocs faisant appel aux paramètres du processus, la conséquence d'une mauvaise modélisation/identification[3] est beaucoup plus lourde que dans le cas où on peut se dispenser de l'observateur. De plus, la combinaison des deux blocs augmente le nombre de calculs à réaliser en temps réel.

4.1.5. Asservissement pour la *régulation* et pour la *poursuite*

Parmi les nombreux objectifs qui nécessitent un asservissement, nous pouvons distinguer deux types principaux explicités tableau 4.2.

[2] Selon [DS79], il est possible de récupérer quelques degrés de robustesse.
[3] Dans la pratique, il est quasiment impossible de modéliser et identifier *parfaitement* un processus.

Régulation	En définissant une consigne constante, il s'agit de ramener une variable du processus le plus rapidement possible à cette consigne, en minimisant au mieux les dépassements et les oscillations transitoires et de la maintenir à sa valeur même en présence de perturbations. La régulation doit également apporter une stabilisation du processus. Exemples : maintien de la température d'une pièce à l'aide d'un radiateur régulé, régulation de la vitesse d'un véhicule à 130km/h sans l'intervention du conducteur.
Poursuite	Dans ce cas, il est question de faire suivre une consigne changeante par une variable du processus en minimisant la distance entre la variable et sa consigne soit continûment, soit discrètement, en apportant une stabilisation. Exemple : faire suivre une cible mobile à un robot. Les exigences de la poursuite sont supérieures à celles de la régulation, surtout si la consigne admet des discontinuités.

Tab. 4.2. *Deux types d'asservissement*

Plusieurs méthodes classiques d'asservissement garantissent la régulation. La poursuite, aussi appelée suivi de trajectoire, peut être traitée par le régulateur PID pour des systèmes SISO ou encore les contrôleurs flous pour des systèmes non linéaires [Bou00].

Le type d'asservissement que nous considérons s'inscrit dans le cadre d'un suivi de trajectoire dans le cas de systèmes linéaires MIMO. Il est, en effet, destiné à faire suivre une consigne changeante par la sortie ou l'état du processus, en vérifiant cette concordance à des instants d'échantillonnage prédéfinis. On parle alors de *poursuite échantillonnée*. Ce type d'asservissement particulier est comparable à la commande *deadbeat* qui annule l'erreur entre une variable d'un modèle du processus et de sa consigne au bout d'un nombre fini d'itération de la commande [AI78, KT81, O'Re81, EF82, Wol83, UN87]. Classiquement, la commande *deadbeat* n'utilisant le retour qu'aux instants d'échantillonnage, il peut se produire des oscillations sur la variable contrôlée entre ces instants. Il existe toutefois des méthodes garantissant la réduction de ces oscillations [UN87]. La méthode que nous proposons offre, dans ce sens, une version dite *sans optimisation* et une version dite *optimisée* selon que l'information de retour n'est prise en compte qu'aux instants d'échantillonnage ou également entre ces instants.

4.2. Contexte de travail

4.2.1. Généralités

Le principe des contrôleurs par morceaux est basé sur le concept de commande composite introduit par Laurent [Lau72] et Vasseur [Vas72]. Nous pouvons retrouver, dans la littérature, la généralisation de la notion d'échantillonneur pour réaliser le contrôle des systèmes linéaires [Kab87]. L'idée est de générer la commande à partir d'une matrice périodique agissant sur le vecteur de sortie échantillonné. Nous avons montré chapitre 2 comment formaliser le GSHF en utilisant un SLCM. Urikura et Nagata [UN87] ont proposé quant à eux une commande échantillonnée avec réduction des oscillations entre les instants d'échantillonnage. Plus récemment, Yamamoto [Yam94] a utilisé le concept de fonction définie par morceaux, pour réaliser un contrôle qu'il qualifie d'hybride et pour lequel l'état est pris en compte, non seulement aux instants d'échantillonnage, mais également entre ces instants. Par ailleurs, plusieurs ouvrages récents proposent le contrôle de processus par une commande constante par morceaux [Rao83, UN87, AP99, MS02, RP02] qui est un cas particulier de la présente stratégie de commande. Selon les auteurs de [KH93], les systèmes hybrides représentent un formalisme prometteur pour améliorer la performance des contrôleurs basés sur un retour d'information échantillonnée–bloquée entre les instants d'échantillonnage. Yamamoto [Yam94] fait ressortir qu'une architecture de commande numérique appliquée à un processus continu offre des propriétés de stabilisation qui sont irréalisables par une commande classique.

4.2.2. Approche

Nous développons, dans ce chapitre des contrôleurs basés sur les SFM linéaires présentés en chapitre 2 et particulièrement bien adaptés à la réalisation d'une architecture temps réel. Un contrôleur à fonctionnement par morceaux utilise le modèle d'état du processus et se base sur un retour d'état pour générer la commande en agissant comme un *système hybride associé* placé en amont du processus.

Dans le cas *non optimisé*, le contrôleur n'utilise l'information de retour qu'aux instants d'échantillonnage (correspondant aux instants de commutation du SFM intégré dans le contrôleur), et peuvent donc naturellement admettre la délivrance échantillonnée du feedback via un capteur numérique[4]. Ils garantissent alors la poursuite échantillonnée à chaque instant d'échantillonnage avec une période d'échantillonnage de retard. Les contrôleurs fonctionnent en *roue libre* entre ces instants, mais permettent tout de même de gérer

[4] Dans le cas où le signal de retour est sous forme échantillonnée, il suffit de synchroniser la commutation des contrôleurs SFM avec les instants d'échantillonnage du capteur. Par ailleurs, le cas des capteurs introduisant de plus un retard est traité au cours du chapitre 5.

l'entrée du système dans cet intervalle de temps par un paramétrage de leurs matrices d'état.

Dans le cas *optimisé*, le contrôleur réalise non seulement la poursuite échantillonnée comme dans le cas non optimisé, mais de plus il prend en compte le retour même entre les instants d'échantillonnage afin de réduire les éventuelles oscillations, surtout quand la période d'échantillonnage est grande.

Dans ce sens, en utilisant le formalisme des SFM, nous définissons un contrôleur à fonctionnement par morceaux (CFM).

Définition 4.1. *Un CFM (figure 4.1) est défini par :*

- *un SFM linéaire (SLCM ou SLBE) avec un espace de commutation* $S = S_k = \{t_k, k = 0,1,2,... \mid t_{k+1} > t_k\}, t_k \in \Im$,
- *deux fonctions de retour :* $\Phi(c(t), x(t))$ *et* $\Psi(c(t), x(t))$ *générant respectivement les entrées continue et à discrétiser du SFM linéaire, où* $c(t)$ *et* $x(t)$ *sont respectivement la consigne d'état et l'état du processus.*

Nous désignons l'état, la sortie, l'entrée continue et l'entrée à discrétiser (selon S ou S_k*) du SFM linéaire par* $\lambda(t)$, $w(t)$, $\varphi(t)$ *et* $\psi(t)$ *respectivement. Les dimensions des différentes matrices et vecteurs apparaissant dans la définition du contrôleur assurent la cohérence des équations et seront précisées lors de la mise en équation de l'ensemble contrôleur-processus.*

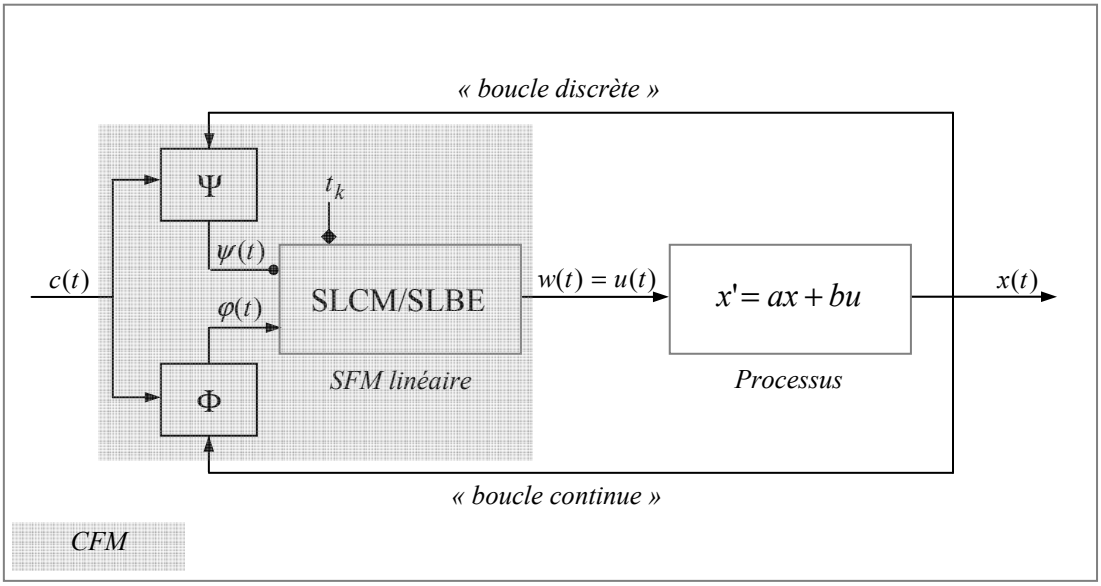


Fig. 4.1. *Asservissement par un CFM*

L'architecture générale du système bouclé de la figure 4.1 montre que le CFM agit comme *système associé* (partie grisée), dont la sortie constitue l'entrée du processus : $w(t) = u(t)$.

Remarque 4.3. *La « boucle discrète » (figure 4.1) détermine le comportement du contrôleur aux instants de commutation, alors que la « boucle continue » gère son comportement entre ces instants. Ainsi, selon que le CFM est optimisé (ou non), la boucle continue sera (ou non) utilisée.*

Comme mentionné dans l'introduction générale, nous représentons le processus linéaire MIMO à contrôler par son METC d'ordre n :

$$x'(t) = a.x(t) + b.u(t), \tag{4.1a}$$

$$y(t) = c.x(t). \tag{4.1b}$$

Ici, $a \in \Re^{n\times n}$, $b \in \Re^{n\times r}$ et $c \in \Re^{m\times n}$ représentent les matrices usuelles de la représentation d'état. Nous supposons, théoriquement, qu'elles sont parfaitement identifiées. Par ailleurs, $u(t) \in U^r$, $x(t) \in \Sigma^n$ et $y(t) \in Y^m$ désignent respectivement l'entrée, l'état et la sortie du processus.

Hypothèse 4.1. *Nous supposons, pour la définition des contrôleurs, que le processus est commandable et observable. De plus, l'état du processus à contrôler ou son estimé est supposé entièrement accessible.*

Par ailleurs, selon que l'on utilise des SLCM ou des SLBE pour générer la commande, on parle de contrôleurs continus par morceaux (CCM) ou de contrôleurs bi-échantillonnés (CBE). Dans chaque cas, nous formaliserons une version sans et avec optimisation concernant l'évolution du contrôleur entre les instants d'échantillonnage.

Dans leur forme de base, les CFM sont conçus pour faire suivre une consigne d'*état*, selon une poursuite échantillonnée, avec une période d'échantillonnage de retard. Toutefois, nous proposerons une adaptation du formalisme général des CFM pour prendre en compte une consigne de sortie.

Par ailleurs, nous envisageons à la fin du chapitre, la combinaison de l'identificateur en ligne du chapitre 3 avec un CFM dans le but de réaliser une structure de commande adaptative dans le cas des processus à paramètres variants.

4.3. Contrôleur continu par morceaux

Nous considérons ici, des CFM basés sur des SLCM. Un CCM représente donc un système associé dont l'état évolue de manière linéaire et *continue* entre les instants de commutation. La commande générée est donc *continue* par morceaux.

4.3.1. CCM non optimisé

Dans ce paragraphe, nous définissons le CCM de sorte à ce qu'il réalise la poursuite échantillonnée suivante :

$$x((k+1).T) = c(k.T),\ \forall k = 0,1,2,... \tag{4.2}$$

$$\Rightarrow\ x_{k+1} = c_k,$$

T étant la période d'échantillonnage constante.

L'équation (4.2), appelée *condition de poursuite*, conduit à réaliser la poursuite échantillonnée de $c(t)$ par $x(t)$, avec un retard d'une période d'échantillonnage. Nous paramétrons donc le CCM afin d'assurer cette adéquation à chaque instant d'échantillonnage, sans imposer de conditions sur l'évolution de l'état du processus *entre* les instants d'échantillonnage. Dans ce sens, seule la boucle discrète est utilisée dans le cas non optimisé (cf. remarque 4.3).

4.3.1.1. Mise en équations

Afin de satisfaire la condition de poursuite (4.2), nous choisissons pour le CCM, un SLCM $\Sigma_c(S,\alpha,\beta_c,\beta_d,\gamma)$ avec :

- des instants de commutations définis par : $S = \{k.T, k = 0,1,2...\}$ (morceaux de durée constante et égale à *T*),
- $\dim(\lambda(t)) = \dim(x(t)) = n$,
- $\dim(w(t)) = \dim(u(t)) = r$,
- $\dim(\alpha) = n \times n$,
- $\dim(\gamma) = r \times n$ avec γ de rang plein,
- une entrée à discrétiser $\psi(t)$ provenant d'une fonctionnelle Ψ (définition 4.1),
- aucune entrée continue ($\varphi(t) = 0 \quad \forall t$).

Nous pouvons ainsi décrire le fonctionnement du système bouclé tel que dans la figure 4.1 par le système d'équations suivant :

$$x'(t) = a.x(t) + b.u(t) \quad \forall t\,, \tag{4.3a}$$

$$\lambda'(t) = \alpha.\lambda(t) \qquad \forall t \in]k.T,(k+1).T]\,, \tag{4.3b}$$

$$\lambda_k^+ = \beta_d.\psi_k \qquad \forall k = 0,1,2,\ldots\,, \tag{4.3c}$$

$$u(t) = w(t) = \gamma.\lambda(t) \qquad \forall t\,. \tag{4.3d}$$

L'évolution du processus est représentée par (4.3a) et celle du CCM est donnée par (4.3b à 4.3c). La connexion CCM-processus est donnée par (4.3d).

Selon (4.3a), nous pouvons décrire l'évolution du processus sur le morceau *k* à partir d'une condition initiale à l'instant t_k^+ :

$$x(t) = e^{a.(t-t_k^+)}.x(t_k^+) + \int_{t_k^+}^{t} e^{a(t-\tau)}.b.u(\tau)d\tau \quad \forall t \in]k.T,(k+1).T] \tag{4.4a}$$

Par ailleurs, selon (4.3b) l'évolution de l'état du contrôleur sur le morceau *k* à partir d'une condition initiale à l'instant t_k^+ est donnée par :

$$\lambda(t) = e^{\alpha.(t-t_k^+)}.\lambda(t_k^+) \quad \forall t \in]k.T,(k+1).T] \tag{4.4b}$$

Ainsi, en tenant en compte de l'expression de la commande $u(t)$, l'évolution du système bouclé conduit en t_{k+1} à :

$$x_{k+1} = e^{a.T}.x_k + M.\lambda_k^+ \tag{4.5a}$$

avec $$M = e^{a.T}.\int_0^T e^{-a.\tau}.b.\gamma.e^{\alpha.\tau}d\tau\,,\ M \in \Re^{n\times n} \tag{4.5b}$$

À condition que M soit inversible[5], nous pouvons obtenir le fonctionnement désiré, soit $x_{k+1} = c_k$, en imposant, d'après l'équation (4.5a) :

$$\lambda_k^+ = M^{-1}.\left\{c_k - e^{a.T}.x_k\right\} \tag{4.6}$$

[5] Les conditions d'existence de la matrice inverse de *M* sont données en annexe IV

L'équation (4.6) correspond à la valeur affectée à l'état à chaque instant de commutation. Selon la structure du contrôleur (4.3c), elle suggère de choisir :

$$\beta_d = M^{-1}, \tag{4.7a}$$

$$\psi(t) = c(t) - e^{a.T}.x(t). \tag{4.7b}$$

4.3.1.2. Architecture

Nous pouvons constater que M^{-1} et $e^{a.T}$ sont deux matrices constantes[6]. Le CCM se construit alors comme illustré figure 4.2.

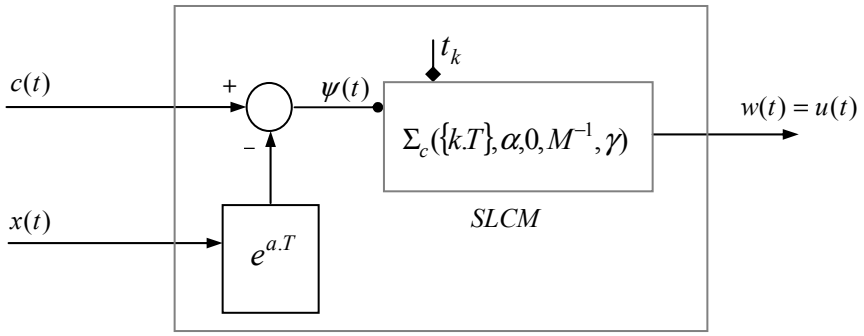


Fig. 4.2. *Architecture du CCM non optimisé*

4.3.1.3. Synthèse

Le contrôleur est donc identifié par un SLCM tel que $\Sigma_c(\{k.T\},\alpha,0,M^{-1},\gamma)$ avec seul un bouclage discret suivant l'équation (4.7b). La matrice *M* est définie par l'équation (4.5b) et les conditions d'existence de son inverse sont données en annexe IV.1. Quant aux matrices α et γ, elles sont choisies arbitrairement[7] dans ce mode de fonctionnement. Sans bouclage continu, le CCM permet de réaliser une poursuite échantillonnée garantissant $x_{k+1} = c_k$ aux instants de commutation. Par contre, aucune contrainte n'étant imposée sur la commande entre ces instants, il existe un risque d'oscillations dans cet intervalle, surtout si la période T est grande. Nous donnons ci-après la structure du CCM optimisé permettant de réduire ces oscillations.

4.3.1.4. Adaptation à une consigne de sortie

Le formalisme de base des CFM permet d'imposer au processus une consigne d'*état*. De ce fait, il est question de définir de manière cohérente les valeurs souhaitées des variables d'état dans l'espace d'état. Il faut soit faire un bon choix de l'évolution de chaque compo-

[6] À condition que la période *T* est elle-même constante.
[7] En respectant bien entendu leurs dimensions respectives.

sante de la consigne d'état, soit adopter un *modèle de référence* qui génère une consigne cohérente [UN87].

Toutefois, il est possible de considérer l'application d'une consigne de sortie $c_s(t) \in Y^m$ dans le cas non optimisé[8] en procédant comme suit :

Si la sortie $y(t)$ du processus est telle que donnée en (4.1b), soit $y(t) = c.x(t)$, alors l'équation (4.5a) s'écrit :

$$y_{k+1} = c.e^{a.T}.x_k + c.M.\lambda_k^+ \text{ avec } \lambda_k^+ = \beta_d.\psi_k$$

Ainsi, il est possible de définir une condition de poursuite par une consigne de sortie : $y_{k+1} = (c_s)_k$ et de définir $\beta_d = (c.M)^{-1}$ en faisant en sorte que $c.M$ soit une matrice carrée non singulière de dimension $m \times m$. Ceci impose que $M \in \Re^{n \times m}$. Selon (4.5b), nous redéfinissons, dans ce cas, les dimensions des paramètres du contrôleur, comme suit :

- $\dim(\lambda(t)) = m$,
- $\dim(\alpha) = m \times m$,
- $\dim(\gamma) = r \times m$.

Par ailleurs, l'équation (4.7b) devient $\psi(t) = c_s(t) - c.e^{a.T}.x(t)$.

4.3.2. CCM optimisé

Nous considérons dans ce paragraphe, une optimisation du CCM présenté précédemment en utilisant, en plus de la boucle discrète et de la condition de poursuite, son bouclage continu pour régir la commande entre les instants de commutation. Le CCM optimisé est défini par $\Sigma_c(\{k.T\}, \alpha, \beta_c, \beta_d, \gamma)$. L'idée est de minimiser les oscillations entre les instants de commutation en définissant convenablement les paramètres et les entrées du SCLM. Le mode de fonctionnement souhaité est défini par :

- $x_{k+1} = c_k$, $\forall k = 0,1,2,...$,
- minimiser la distance entre $c(t-T)$ et $x(t)$ sur le morceau k.

4.3.2.1. Mise en équations

La poursuite échantillonnée est garantie par la boucle discrète comme précédemment et l'optimisation se fait par la boucle continue en minimisant la fonctionnelle suivante :

[8] Le CCM optimisé fait une comparaison entre l'état et la consigne d'*état* sur le morceau *k* (équation (4.8)).

$$J = \tfrac{1}{2} \int_{k.T}^{(k+1).T} \left[(c(\tau - T) - x(\tau))^T .E.(c(\tau - T) - x(\tau)) + w^T(\tau).G.w(\tau) \right] d\tau \qquad (4.8)$$

où $E \in \Re^{n \times n}$ et $G \in \Re^{r \times r}$ sont deux matrices symétriques définies positives.

Dans ces conditions, la minimisation de J conduit à minimiser la distance entre $c(t-T)$ et $x(t)$ sur un morceau k, tout en modérant la commande $u(t) = w(t)$. En reprenant l'équation (4.1a) du processus avec pour entrée la sortie du CCM, la solution à ce problème de commande optimale est obtenue en appliquant le principe de Pontryagin à l'hamiltonien H défini par :

$$H = -\tfrac{1}{2} \left[(c(t-T) - x(t))^T .E.(c(t-T) - x(t)) + u^T(t).G.u(t) \right] + \lambda^T(t).[a.x(t) + b.u(t)] \qquad (4.9)$$

où $\lambda(t) \in \Re^n$ correspond au vecteur adjoint (multiplicateur de Lagrange).

Le principe de Pontryagin donne la solution suivante :

$$\frac{dH}{d\lambda} = x'(t) = a.x(t) + b.u(t), \qquad (4.10a)$$

$$-\frac{dH}{dx} = \lambda'(t) = -a^T .\lambda(t) - E.[c(t-T) - x(t)], \qquad (4.10b)$$

$$\frac{dH}{du} = 0 = -G.u(t) + b^T .\lambda(t), \quad \text{soit : } u(t) = G^{-1}.b^T .\lambda(t), \qquad (4.10c)$$

4.3.2.2. Identification des paramètres du CCM optimisé

Alors que la solution de Pontryagin nous donne en (4.10a) l'équation d'état du processus à piloter, les deux autres équations nous permettent de définir un SLCM $\Sigma_c(S, \alpha, \beta_c, \beta_d, \gamma)$ pour le CCM. Avec les notations usuelles, les paramètres du CCM optimisé sont donc :

$$S = \{k.T, k = 0,1,2...\}, \qquad (4.11a)$$

$$\alpha = -a^T, \qquad (4.11b)$$

$$\beta_c = -E, \qquad (4.11c)$$

$$\gamma = G^{-1}.b^T, \qquad (4.11d)$$

$$\varphi(t) = c(t-T) - x(t). \qquad (4.11e)$$

L'intégration du système d'équations (4.10) sur le morceau k permet de compléter la définition du SLCM et de déduire β_d. De plus, elle permet de préciser la fonctionnelle $\Psi(c(t),x(t))$ caractérisant la boucle discrète.

En utilisant les systèmes d'équations (4.10) et (4.11), on peut écrire :

$$\begin{bmatrix} x'(t) \\ \lambda'(t) \end{bmatrix} = \mathrm{H} \cdot \begin{bmatrix} x(t) \\ \lambda(t) \end{bmatrix} + \mathrm{K} \cdot c(t-T) \tag{4.12a}$$

$$\text{avec } \mathrm{H} = \begin{bmatrix} a & b.G^{-1}.b^T \\ E & -a^T \end{bmatrix} \in \Re^{2n\times 2n} \text{ et } \mathrm{K} = \begin{bmatrix} 0 \\ -E \end{bmatrix} \in \Re^{2n\times n} \tag{4.12b}$$

En intégrant (4.12a) sur le morceau k, on obtient :

$$\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \end{bmatrix} = e^{\mathrm{H}.T} . \begin{bmatrix} x_k \\ \lambda_k^+ \end{bmatrix} + \mathrm{I}_{k+1} \tag{4.13a}$$

$$\text{avec } \mathrm{I}_{k+1} = \int_{k.T}^{(k+1).T} \left\{ e^{\mathrm{H}.[(k+1).T-\tau]}.\mathrm{K}.c(\tau - T) \right\} d\tau \ \Big| \ \mathrm{I}_{k+1} \in \Re^{2n\times 1} \tag{4.13b}$$

Si l'on note :

$$e^{\mathrm{H}.T} = \Theta = \begin{bmatrix} \Theta_{11} & \Theta_{12} \\ \Theta_{21} & \Theta_{22} \end{bmatrix} \text{ et } \mathrm{I}_{k+1} = \begin{bmatrix} \mathrm{I}^h \\ \mathrm{I}^b \end{bmatrix}, \tag{4.14}$$

avec $\mathrm{I}^h \in \Re^{n\times 1}$ et $\mathrm{I}^b \in \Re^{n\times 1}$ étant les moitiés supérieure et inférieure de I_{k+1}, on obtient à partir de la première relation de (4.13a) :

$$x_{k+1} = \Theta_{11}.x_k + \Theta_{12}.\lambda_k^+ + \mathrm{I}^h \tag{4.15}$$

Rappelons que la valeur λ_k^+ de l'état du contrôleur à la commutation est donnée par $\lambda_k^+ = \beta_d.\psi_k$ selon (4.3c). Il faut donc identifier la matrice β_d et l'entrée ψ_k. Dans ce sens, nous remplaçons la condition de poursuite (4.2) et la relation (4.3c) dans (4.15) :

$$c_k = \Theta_{11}.x_k + \Theta_{12}.\beta_d.\psi_k + \mathrm{I}^h \tag{4.16}$$

En supposant que Θ_{12} soit inversible, on obtient :

$$\beta_d.\psi_k = \Theta_{12}^{-1}.(c_k - \Theta_{11}.x_k - \mathrm{I}^h) \tag{4.17}$$

L'équation (4.17) suggère d'adopter :

$$\beta_d = \Theta_{12}^{-1} \tag{4.18a}$$

$$\psi(t) = c(t) - \Theta_{11}.x(t) - \mathrm{I}^h \tag{4.18b}$$

La structure du CCM optimisé est ainsi entièrement définie par les relations (4.11a-d) et (4.18a). Le bouclage continu est défini par la relation (4.11e) et le bouclage discret est défini par (4.18b).

4.3.2.3. Architecture

4.3.2.3.1. Calculs préliminaires

Parmi les relations qui servent à définir l'architecture de commande, (4.18b) fait intervenir le terme I^h. Nous définissons donc, au préalable, un mode de calcul de ce terme permettant une réalisation simple de la fonctionnelle de bouclage $\Psi(c(t), x(t))$.

En posant $\tau = \theta + T$ dans l'équation (4.13b), il vient :

$$\mathrm{I}_{k+1} = \int_{(k-1).T}^{k.T} \left\{ e^{\mathrm{H}.(k.T-\theta)}.\mathrm{K}.c(\theta) \right\} d\theta \tag{4.19}$$

On peut encore écrire :

$$\mathrm{I}_{k+1} = \lim_{t \to k.T, t<k.T} \left[\tilde{\mathrm{I}}(t) \right] = \tilde{\mathrm{I}}_k^- \tag{4.20a}$$

$$\text{avec } \tilde{\mathrm{I}}(t) = \int_{(k-1).T}^{t} \left\{ e^{\mathrm{H}.(t-\theta)}.\mathrm{K}.c(\theta) \right\} d\theta \tag{4.20b}$$

L'équation (4.20b) fait apparaître une intégrale de convolution dont la moitié supérieure correspond, à l'instant, $k.T^-$ à I^h. Ce dernier peut donc être interprété comme la sortie, à $k.T^-$, d'un SLCM auxiliaire défini par $\Sigma_c(\{k.T\}, \mathrm{H}, \mathrm{K}, 0, [I_n \;\; 0])$ dont seule l'entrée continu est utilisée et est alimentée par $c(t)$.

4.3.2.3.2. Structure du contrôleur optimisé

Nous donnons en annexe IV.2 les conditions d'existence de la poursuite échantillonnée optimisée garantie par le CCM optimisé construit à partir du SLCM défini par $\Sigma_c(\{k.T\}, -a^T, -E, \Theta_{12}^{-1}, G^{-1}.b^T)$.

En considérant les fonctionnelles définissant les entrées de ce SLCM telles que définies précédemment, nous proposons un schéma de construction du CCM optimisé en figure 4.3 où :

- $e^{-\varepsilon.p}$ représente un retard infiniment petit permettant de prendre en compte la sortie du SLCM auxiliaire $\Sigma_c(\{k.T\},\mathrm{H},\mathrm{K},0,[I_n \;\; 0])$ juste avant l'instant de commutation (instant marqué par une discontinuité),
- $e^{-T.p}$ représente un retard d'une période d'échantillonnage,
- l'entrée à discrétiser de $\Sigma_c(\{k.T\},-a^T,-E,\Theta_{12}^{-1},G^{-1}.b^T)$ est : $\psi(t)=c(t)-\Theta_{11}.x(t)-\vartheta(t^-)$ où $\vartheta(t)$ représente la sortie du SLCM auxiliaire $\Sigma_c(\{k.T\},\mathrm{H},\mathrm{K},0,[I_n \;\; 0])$.

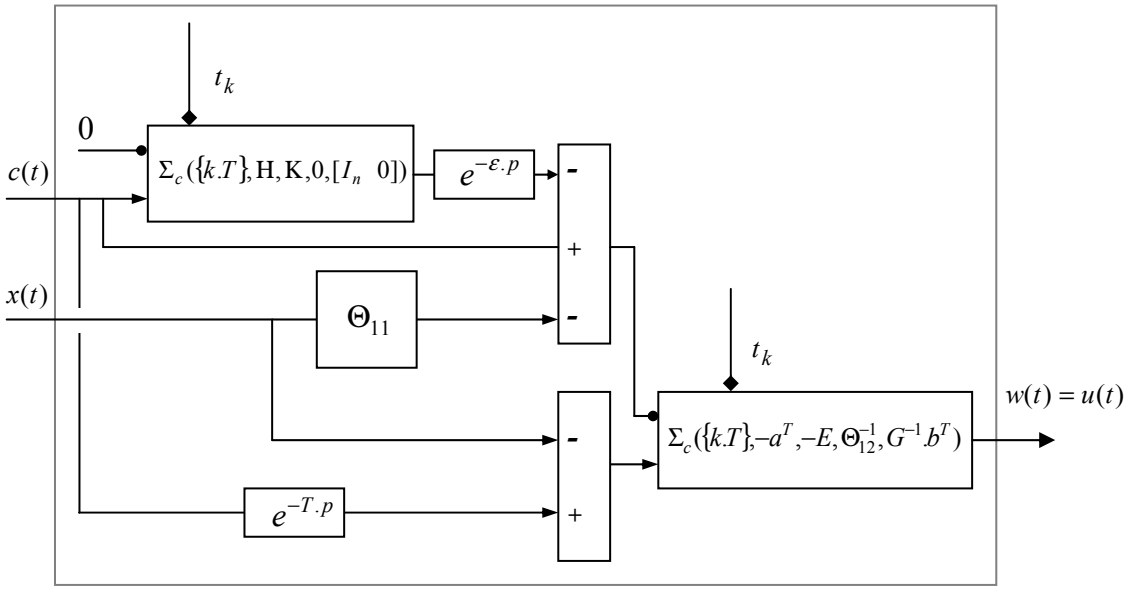


Fig. 4.3. *Architecture du CCM optimisé*

4.3.2.4. Synthèse

Le CCM optimisé est complètement défini par l'architecture décrite dans le paragraphe précédent. Il exploite complètement la structure d'un CFM (figure 4.1) en utilisant à la fois la boucle discrète pour garantir la poursuite aux instants de commutation (comme dans le cas non optimisé) et la boucle continue pour réduire, en plus, les oscillations éventuelles entre ces instants.

Les conditions d'existence de la poursuite optimisée par le CCM sont données en annexe IV.2.

4.3.3. Validation expérimentale

De la même manière que les SFM, la mise en œuvre des contrôleurs CFM est tout à fait réalisable sous Simulink® et par conséquent sur des architectures temps réel. Nous proposons, dans ce paragraphe, des exemples de simulation montrant le fonctionnement et les performances des CCM non optimisés et optimisés dans différentes situations. L'application de la méthode sur la plate-forme réelle décrite en annexe I est également donnée en exemple.

Remarque 4.4. *Dans tous les cas, les CFM garantissent une poursuite échantillonnée faisant coïncider l'état* $x(t)$ *du processus avec* $c(t-T)$ *à chaque instant de commutation. De ce fait, si* T *est faible, la commande générée par le contrôleur peut devenir excessivement grande pour satisfaire la poursuite en cas de discontinuités (ou variation à dynamique très forte) sur la consigne*[9]*. Une commande surélevée pouvant endommager un processus réel, nous proposons, en pratique, d'inclure un « amortisseur » (cf. paragraphe suivant) dans la structure de commande par mesure de précaution.*

4.3.3.1. Amortisseur

Afin d'apporter une solution au problème soulevé dans la remarque précédente, nous proposons de modifier l'architecture de commande en incluant un *amortisseur* selon la figure 4.4 où $r(t)$ représente l'information de retour. Celle-ci peut être soit l'état ou la sortie du processus, selon le cas envisagé.

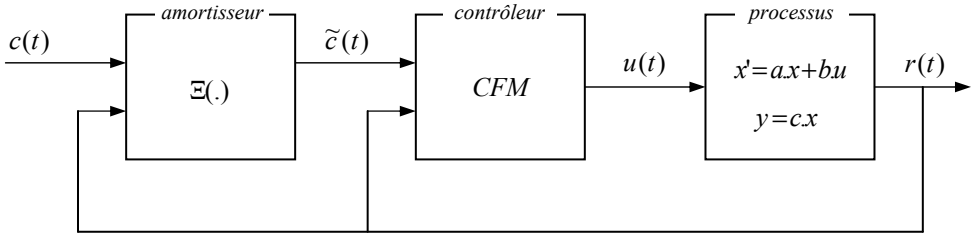


Fig. 4.4. *Utilisation de l'amortisseur dans l'architecture de commande CFM*

L'idée est de *lisser* la consigne à appliquer ($\tilde{c}(t)$), dans le but de réagir en cas de discontinuités sur $c(t)$. Il est évident que sans les discontinuités (fonctionnement *normal* de l'ensemble bouclé où la sortie du processus reste *accrochée* à sa consigne : $r(t)=c(t-T)$) l'amortisseur réalise $\tilde{c}(t)=c(t)$, restant ainsi transparent. Par ailleurs, si la consigne présente une discontinuité, une architecture de contrôle CFM classique conduit d'abord à un *décrochage* de la sortie du processus de sa consigne, et à l'instant de commutation suivant, le contrôleur génère une commande nécessairement élevée pour

[9] Les discontinuités ne peuvent survenir que sur la consigne, car l'état du processus réel est toujours continu (l'équation de son modèle d'état le confirme).

rattraper la consigne *en un coup*. Afin de pallier cet inconvénient, l'amortisseur fournit au CFM une consigne $\tilde{c}(t)$ qui se rapproche de l'état du processus de sorte à conserver au mieux l'accrochage de la sortie avec $\tilde{c}(t)$ attendant alors que $c(t)$ et $r(t)$ soient proche pour faire tendre $\tilde{c}(t) \to c(t)$. Ceci se traduit mathématiquement par :

$$\tilde{c}(t) = r(t) + \varsigma(c(t), r(t)).(c(t) - r(t)),$$

le coefficient d'amortissement $\varsigma(.)$ étant une fonction scalaire telle que $0 < \varsigma(.) \leq 1$.

Dans le fonctionnement *normal* ($r(t)$ *accroché* à sa consigne retardée de *T*), $\varsigma(.) = 1$ de sorte que $\tilde{c}(t) = c(t)$. Autrement, si $r(t)$ s'écarte de sa consigne, $\varsigma(.)$ décroît vers zéro selon une fonction en cloche donnée par :

$$\varsigma(.) = \frac{1}{1+\varepsilon}\left\{\varepsilon + e^{-\frac{\theta}{2}(\Delta^T.\Delta)}\right\}, \text{ avec } \Delta = c(t-T) - r(t),$$

ε et θ étant des entiers positifs.

La figure 4.5 illustre l'évolution de $\varsigma(.)$ en fonction de l'écart Δ qui sépare la sortie de sa consigne (retardée de *T*).

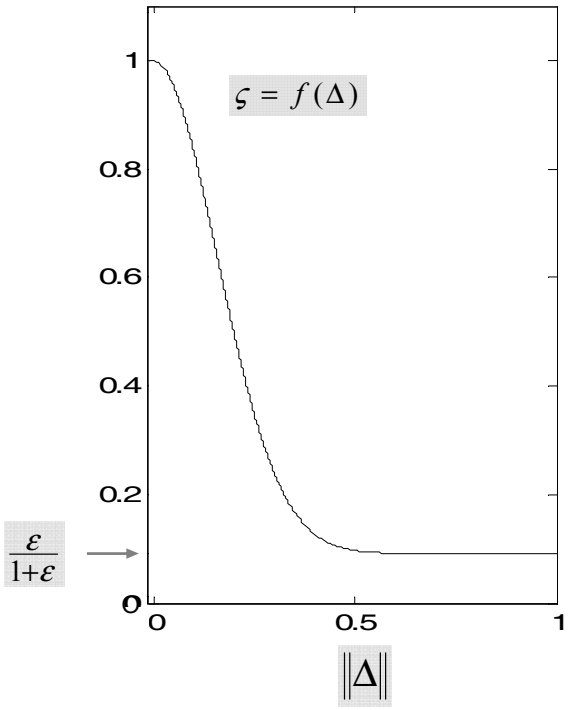


Fig. 4.5. *Fonction cloche*

L'amortisseur est paramétrable par ε et θ pour le rendre plus ou moins réactif. Nous utilisons donc la notation symbolique $\Xi(\theta,\varepsilon)$ pour le désigner.

4.3.3.2. CCM non optimisé

4.3.3.2.1. Processus stable

Soit un processus linéaire continu *stable* tel que défini par (4.1) avec :

$$a=\begin{bmatrix} 0 & 1 \\ -2 & -5 \end{bmatrix},\ b=\begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ (avec } n=2 \text{ et } r=1\text{), et } x_0=\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

commandé par un CCM non optimisé donné par $\Sigma_c(\{k.T\},\alpha,0,M^{-1},\gamma)$ avec :

$$\alpha=\begin{bmatrix} -1 & 4 \\ -2 & -1 \end{bmatrix},\ \gamma=\begin{bmatrix} 10 & 10 \end{bmatrix} \text{ et } T=1s$$

pour suivre une consigne d'état donné par : $c(t)=\begin{bmatrix} 10\sin(t) \\ 10\cos(t) \end{bmatrix}$.

Les résultats sont illustrés par les figures 4.6a et 4.6b. Sachant que la poursuite est réalisée avec retard d'une période de commutation, nous avons retardé convenablement, dans cet exemple, les signaux de la consigne pour faciliter la comparaison des courbes de la figure 4.6a.

La méthode garantit la coïncidence aux instants de commutation, mais l'évolution de l'état entre ces instants n'est régie que par le choix de la matrice α.

La figure 4.6b représente la commande générée par le CCM non optimisé. Nous y constatons son évolution par morceaux. Il est évident que la stabilité de la matrice α permet d'éviter des valeurs excessives de la commande sur chaque morceau *k*.

À noter que la période de commutation est expressément longue pour des raisons de démonstration. La poursuite est naturellement meilleure dans le cas d'une commutation rapide.

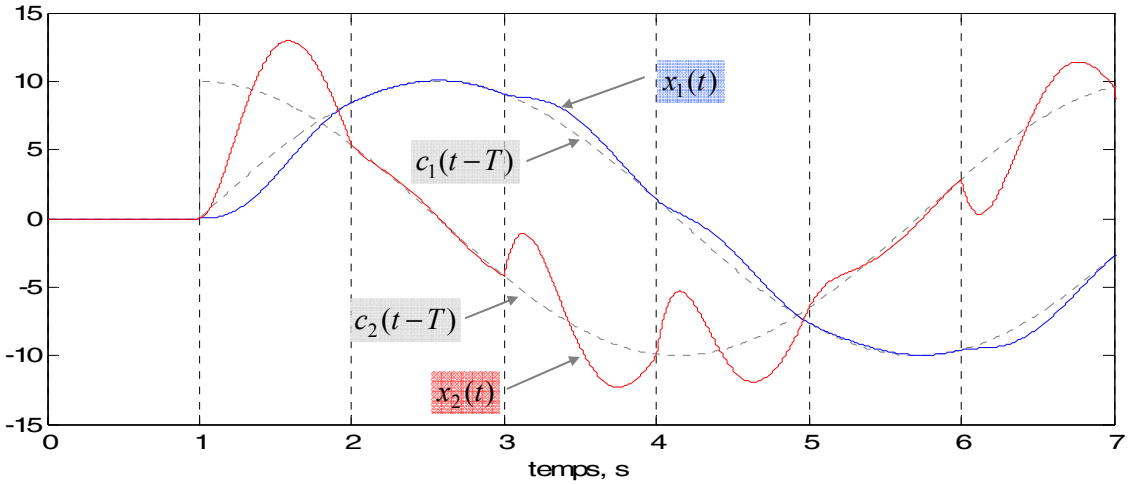


Fig. 4.6a. *Poursuite échantillonnée non optimisée, $T = 1s$: l'état et la consigne retardée*

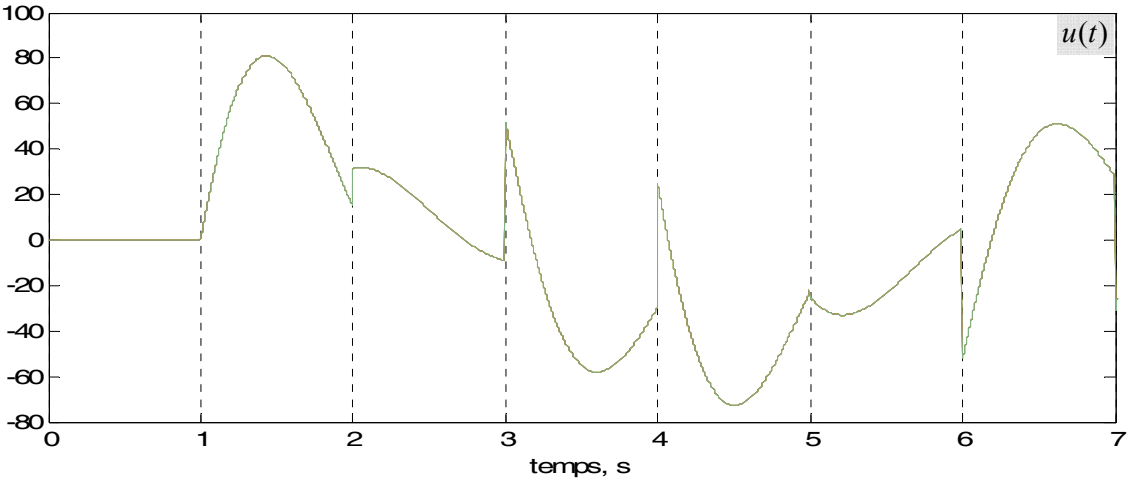


Fig. 4.6b. *Poursuite échantillonnée non optimisée, $T = 1s$: la commande*

4.3.3.2.2. Processus instable bruité

Nous considérons ici un processus instable commandé par un CCM non optimisé disposant de l'état perturbé[10] du processus. Nous résumons les paramètres intervenant dans le système bouclé par :

$$a = \begin{bmatrix} 0 & 1 \\ 1 & 3 \end{bmatrix},\ b = \begin{bmatrix} 0 \\ 4 \end{bmatrix} \text{ et } x_0 = \begin{bmatrix} 0 \\ -10 \end{bmatrix},$$

$$\alpha = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},\ \gamma = \begin{bmatrix} 1 & 5 \end{bmatrix} \text{ et } T = 50ms,\ c(t) = \begin{bmatrix} 5\sin(8t) \\ 40\cos(8t) \end{bmatrix}.$$

Les résultats sont donnés en figure 4.7. À noter qu'ici, les signaux de consigne ne sont pas retardés pour être détachés des signaux d'état.

[10] Même perturbation sur l'état que dans le §3.6.1.2.

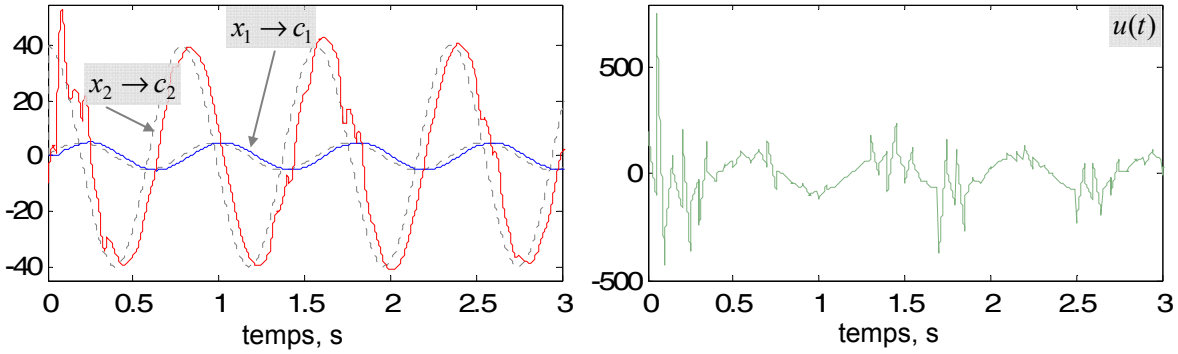


Fig. 4.7. *Poursuite échantillonnée non optimisé, T = 50ms : l'état et la consigne ; la commande*

4.3.3.2.3. Consigne de sortie

Dans cet exemple, nous imposons un signal carré (d'amplitude 0.5 et de période 4s) pour la *sortie* d'un processus d'ordre 2. Pour ce faire, nous utilisons l'adaptation donnée au §4.3.1.4. Cet exemple permet en plus d'expérimenter le CCM en régulation sur deux processus différents (P1 stable et P2 instable) définis par :

$$\text{P1}: a_1 = \begin{bmatrix} -1 & 2 \\ 1 & -5 \end{bmatrix},\ b_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix},\ \text{P2}: a_2 = \begin{bmatrix} 0 & 1 \\ 5 & -10 \end{bmatrix},\ b_2 = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$$

avec la même matrice de sortie $c_1 = c_2 = \begin{bmatrix} 1 & 0 \end{bmatrix}$ et les mêmes CI $x_0 = \begin{bmatrix} 0 \\ -10 \end{bmatrix}$.

Le même CCM est adopté dans les deux cas. Son paramétrage donné ci-dessous permet d'imposer une consigne de sortie :

$$\alpha = -1,\ \gamma = 0.3 \text{ et } T = 50ms$$

Les résultats illustrés figure 4.8 démontrent la performance du CCM en régulation. Pour le processus instable, nous remarquons qu'il existe un transitoire important. La durée de celui-ci peut être réduite en diminuant la période de commutation. Néanmoins, une commutation rapide dans des situations de discontinuité, peut aboutir à une commande excessivement élevée, car le contrôleur vise à satisfaire la condition de poursuite à chaque instant de commutation. Dans ce cas, il est possible d'atténuer la commande et les dépassements à l'aide de l'*amortisseur* décrit au §4.3.3.1.

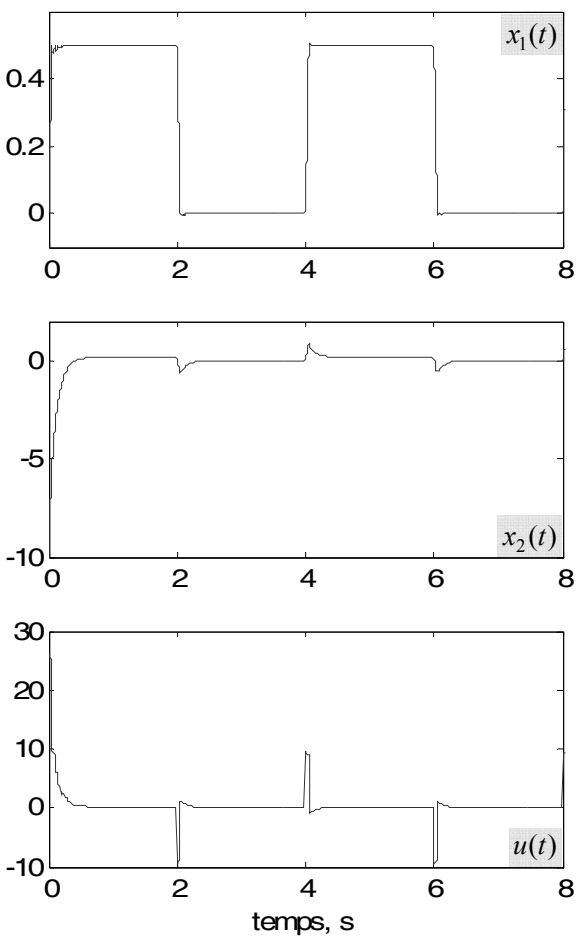


Fig. 4.8a. *Processus* P1 *stable*

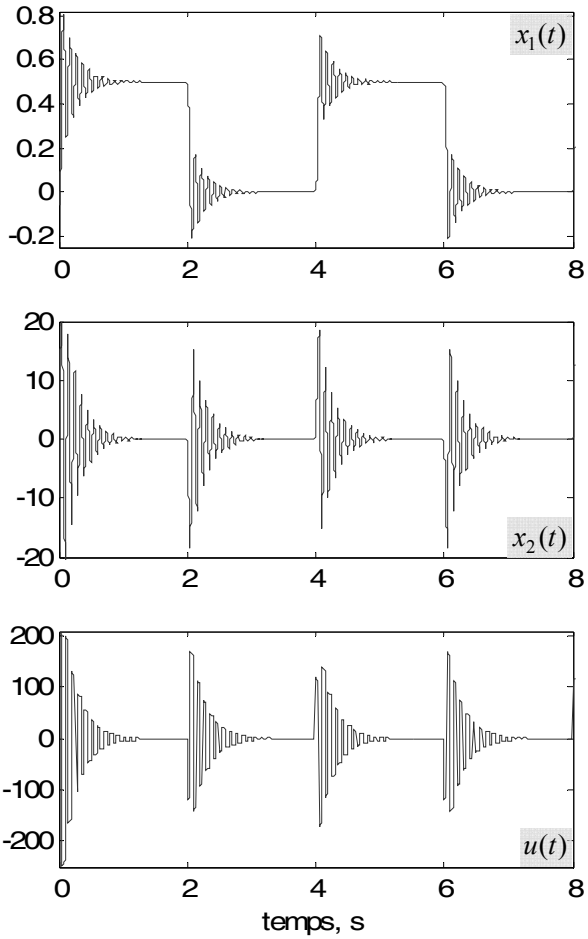


Fig. 4.8b. *Processus* P2 *instable*

4.3.3.2.4. Commande constante par morceaux

Comme nous l'avons mentionné au début de ce chapitre, les CFM et en particulier les CCM permettent de réaliser le contrôle de processus par une commande constante par morceaux, comparable à celle décrite dans [Rao83, UN87, AP99, MS02, RP02]. Il suffit de définir, pour le contrôleur, une matrice α nulle.

Exemple 1. Soit un système bouclé (CCM non optimisé – processus) défini par :

$$a = \begin{bmatrix} -1 & 2 \\ 1 & -5 \end{bmatrix},\ b = \begin{bmatrix} 1 \\ 2 \end{bmatrix},\ c = \begin{bmatrix} 1 & 0 \end{bmatrix} \text{ et } x_0 = \begin{bmatrix} 0 \\ -5 \end{bmatrix},$$

$$\alpha = 0,\ \gamma = 0.3 \text{ et } T = 0.5s,$$

sur lequel on impose une consigne de sortie $c_s(t) = 10\sin(t)$.

Les résultats présentés figure 4.9a montrent que la poursuite est réalisée sans difficulté par le contrôleur CCM.

La deuxième composante l'état est également illustrée sur la figure 4.9a et la commande, évoluant de manière constante par morceaux, est représentée figure 4.9b.

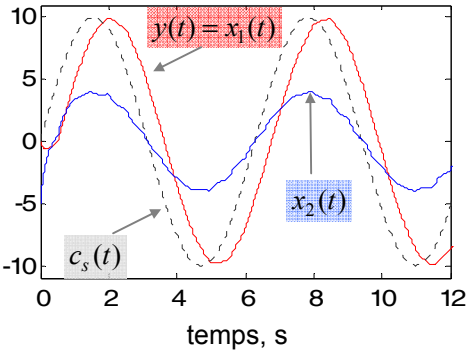


Fig. 4.9a. *Poursuite non optimisée,* $T = 0.5s$

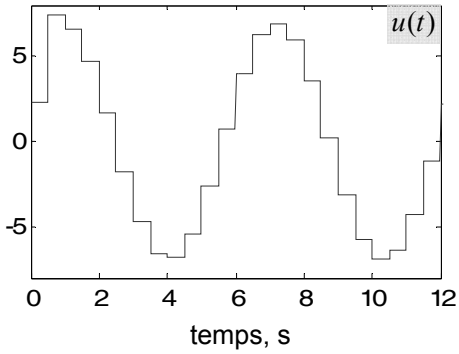


Fig. 4.9b. *Commande constante par morceaux*

Remarque 4.5. *Dans le cas général d'une consigne d'état, la mise en œuvre d'un CCM à fonctionnement constant par morceaux conduit à une matrice M singulière, rendant impossible le calcul de son inverse. Le problème est contourné, en pratique, en utilisant la pseudo-inverse de Moore Penrose.*

Exemple 2. Considérons le processus décrit dans [UN87] décrit par :

$$a=\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\0&0&0&0\end{bmatrix},\ b=\begin{bmatrix}0&0\\0&0\\0&1\\1&0\end{bmatrix},\ c=\begin{bmatrix}1&0&1&0\\1&0&0&1\end{bmatrix}\ \text{et}\ x_0=\begin{bmatrix}0\\0\\0\\0\end{bmatrix},$$

et un CCM non optimisé défini par :

$$\alpha=\begin{bmatrix}0&0\\0&0\end{bmatrix},\ \gamma=\begin{bmatrix}1&0\\0&1\end{bmatrix}.$$

La consigne de sortie adoptée est : $c_s(t)=\begin{bmatrix}t\\0.2t\end{bmatrix}$.

Les résultats sont donnés en figures 4.10a et 4.10b pour $T=0.5s$ et en 4.10c et 4.10d pour $T=1.5s$. La consigne est retardée en figure 4.10c pour faciliter la lecture des courbes. Nous remarquons que pour une période de commutation élevée, il existe des oscillations entre les instants de commutation. La poursuite est toutefois réalisée à chacun de ces instants.

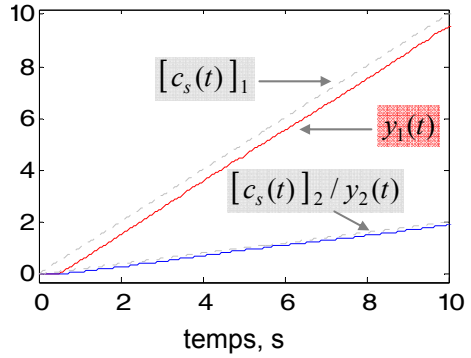


Fig. 4.10a. *Poursuite non optimisée, $T = 0.5s$*

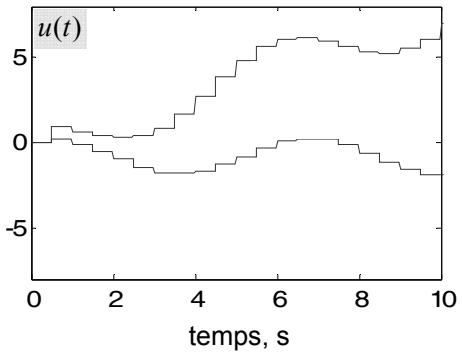


Fig. 4.10b. *Commande constante par morceaux*

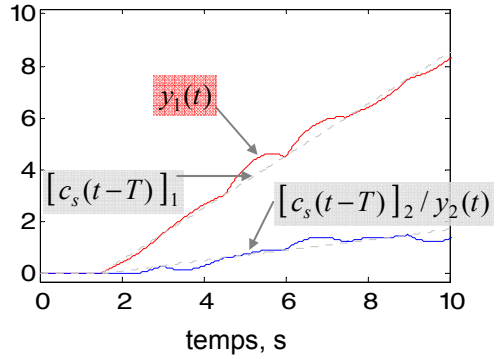


Fig. 4.10c. *Poursuite non optimisée, $T = 1.5s$*

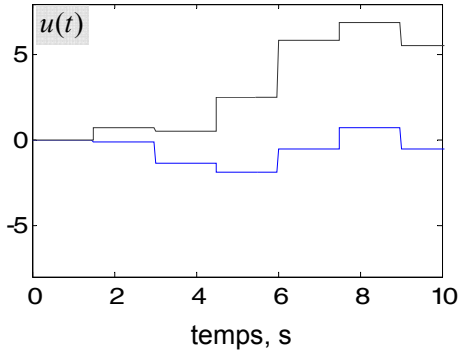


Fig. 4.10d. *Commande constante par morceaux*

4.3.3.2.5. Processus à paramètres variant dans le temps

Considérons un processus P1 invariant décrit par :

$$\text{P1}: a = \begin{bmatrix} -3 & 1 \\ 5 & -4 \end{bmatrix},\ b = \begin{bmatrix} 3 \\ 7 \end{bmatrix},\ c = \begin{bmatrix} 3 & 5 \end{bmatrix} \text{ et } x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

et un deuxième processus P2 semblable à P1, sauf que certains de ses paramètres sont soumis à une variation temporelle :

$$\text{P2}: a_2(t) = \begin{bmatrix} -3+\sin(2\pi.t) & 1 \\ 5 & -4+2\sin(4\pi t) \end{bmatrix},\ b_2 = \begin{bmatrix} 3 \\ 7 \end{bmatrix},\ c_2 = \begin{bmatrix} 3 & 5 \end{bmatrix} \text{ et } x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Nous testons ici la performance d'un CCM défini par $\alpha = -0.1$, $\gamma = 0.3$ et $T = 50ms$, pour satisfaire une consigne de sortie $c_s = 10\sin(t)$ dans les deux cas.

Les résultats de la figure 4.11 montrent en (a) et (b) la performance du CCM dans le cas idéal du processus P1 invariant et sa robustesse en (c) et (d) face à un cas très perturbé que représente le processus P2. Nous constatons (figure 4.11d) qu'une commande fortement altérée et non sinusoïdale est générée par le CCM pour palier aux variations temporelles des paramètres.

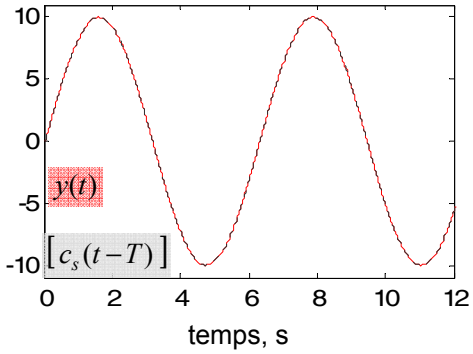


Fig. 4.11a. *Processus* P1 *invariant : poursuite*

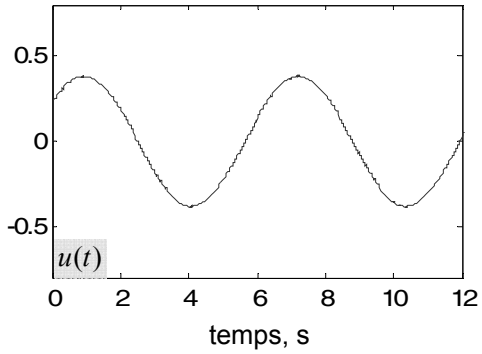


Fig. 4.11b. *Processus* P1 *invariant : commande*

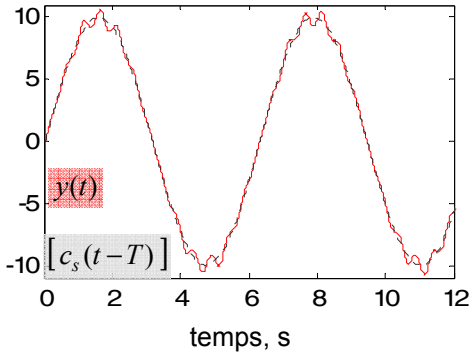


Fig. 4.11c. *Processus* P2 *changeant : poursuite*

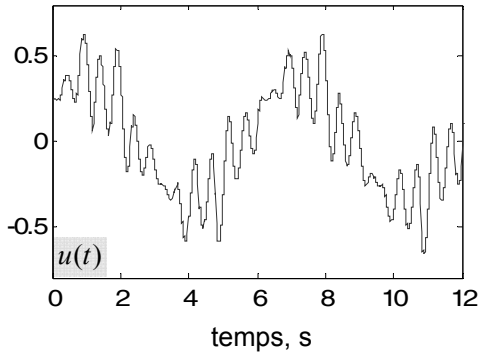


Fig. 4.11d. *Processus* P2 *changeant : commande*

4.3.3.2.6. Processus réel

Les performances temps réel du CCM ont été testées sur la maquette décrite en annexe I. La partie qui nous intéresse correspond au moteur électrique qui sert à déplacer un chariot sur un axe horizontal et rectiligne. Ce chariot, relié au moteur par le biais d'une courroie crantée, parcours un tronçon fini de l'axe. Le montage est représenté figure 4.12.

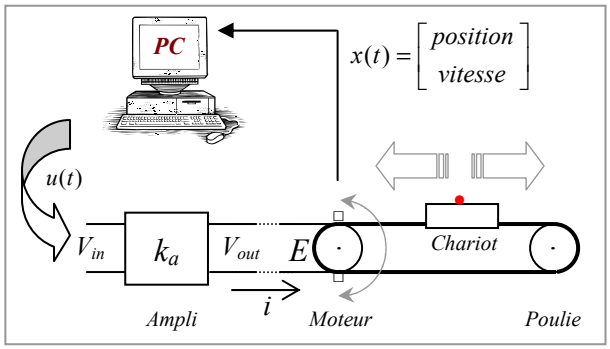


Fig. 4.12. *Montage temps réel*

Dans cet exemple, nous considérons que l'ensemble amplificateur–moteur–chariot à commander est représenté par un METC tel que décrit par (4.1), avec :

$$x(t)=\begin{bmatrix} position \\ vitesse \end{bmatrix},\ x_0=\begin{bmatrix} 0 \\ 0 \end{bmatrix},\ a=\begin{bmatrix} 0 & 1 \\ 0 & -72 \end{bmatrix},\ b=\begin{bmatrix} 0 \\ 190 \end{bmatrix}.$$

les paramètres étant ceux identifiés par la méthode du clonage (chapitre 3).

Nous définissons une consigne d'état à imposer au système :

$$c(t) = \begin{bmatrix} 0.05\sin(1.5t) \\ 0.075\cos(1.5t) \end{bmatrix}$$

Cette consigne est enclenchée à un instant t_0 tel que $1 < t_0 < 2\text{s}$ et déclenchée à un instant $t_1 \approx 7\text{s}$. Ailleurs, nous appliquons une consigne d'état nulle.

Pour réaliser l'asservissement, nous utilisons un CCM non optimisé basé sur $\Sigma_c(\{k.T\}, \alpha, 0, M^{-1}, \gamma)$ avec :

$$\alpha = \begin{bmatrix} -2 & 8 \\ -7 & 10 \end{bmatrix},\ \gamma = \begin{bmatrix} 10 & 10 \end{bmatrix} \text{ et } T = 10\text{ms}$$

Par ailleurs, afin d'atténuer la commande en cas de discontinuités sur la consigne (enclenchement ou déclenchement de la consigne), nous utilisons l'amortisseur décrit au §4.3.3.1, défini par $\Xi(20,0.15)$.

Les résultats sont présentés figure 4.13 qui illustre le suivi de la trajectoire en consigne par la position et la vitesse du chariot. À noter que le signal de retour en vitesse ($x_2(t)$) provient d'un capteur bruité. La poursuite dans ces conditions montre la robustesse de la méthode.

Par ailleurs, nous illustrons l'évolution de la commande ainsi que celle du paramètre $\varsigma(t)$ de l'amortisseur $\Xi(20,0.15)$. Nous constatons que $\varsigma(t)$ est voisin de 1 quand l'état est *accroché* à sa consigne et que ce paramètre décroît rapidement en cas de *décrochage* lié, ici, à un changement brusque de la consigne. Ainsi, une consigne proche de l'état est appliqué aux instants t_0 et t_1 pendant un transitoire (de quelques millisecondes) de sorte que la commande $u(t)$ ainsi que la vitesse $x_2(t)$ soient atténuées à ces instants : les pics correspondants sont illustrés sur la figure annexe (grisée).

Enfin, il est évident que même si nous approximons l'ensemble amplificateur–moteur–chariot par un système linéaire, l'allure de la commande par rapport à la consigne reflète un comportement non linéaire du processus réel, notamment une zone morte pour une entrée faible. Ce phénomène est lié aux frottements secs.

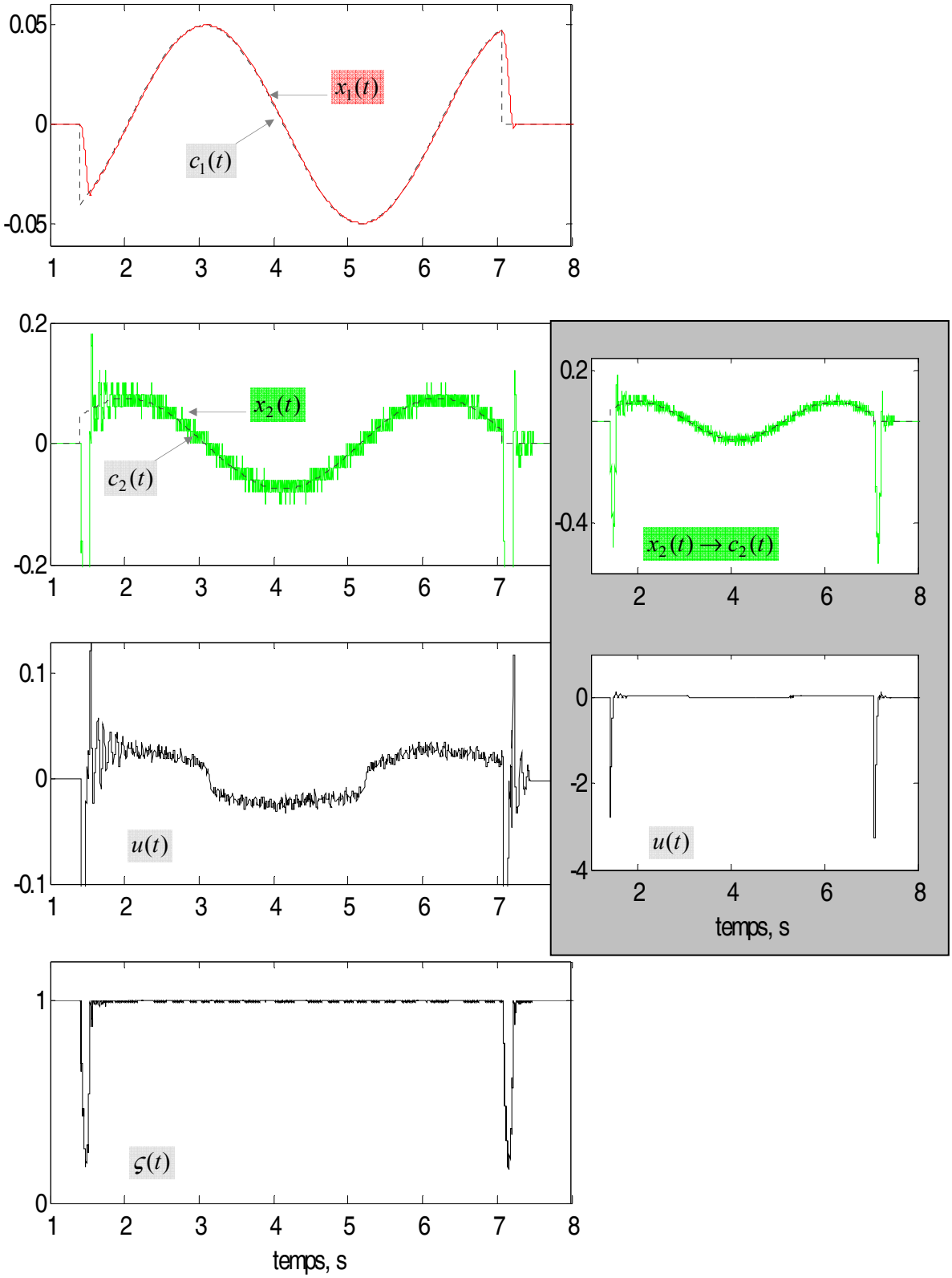


Fig. 4.13. *Asservissement d'un processus réel par un CCM*

4.3.3.3. CCM optimisé

Nous considérons le même processus que dans le §4.3.3.2.1, commandé par un CCM optimisé donné par $\Sigma_c(\{k.T\},-a^T,-E,\Theta_{12}^{-1},G^{-1}.b^T)$ avec $E=I_2$, $G=10$ et $T=1s$ pour suivre la même consigne d'état.

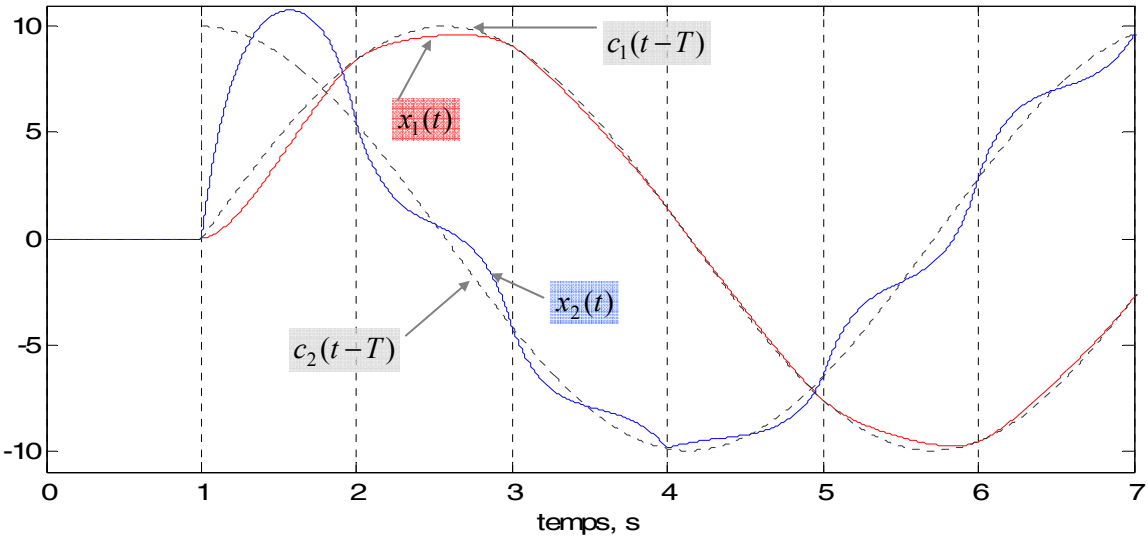


Fig. 4.14a. *Poursuite échantillonnée optimisée, T = 1s : l'état et la consigne retardée*

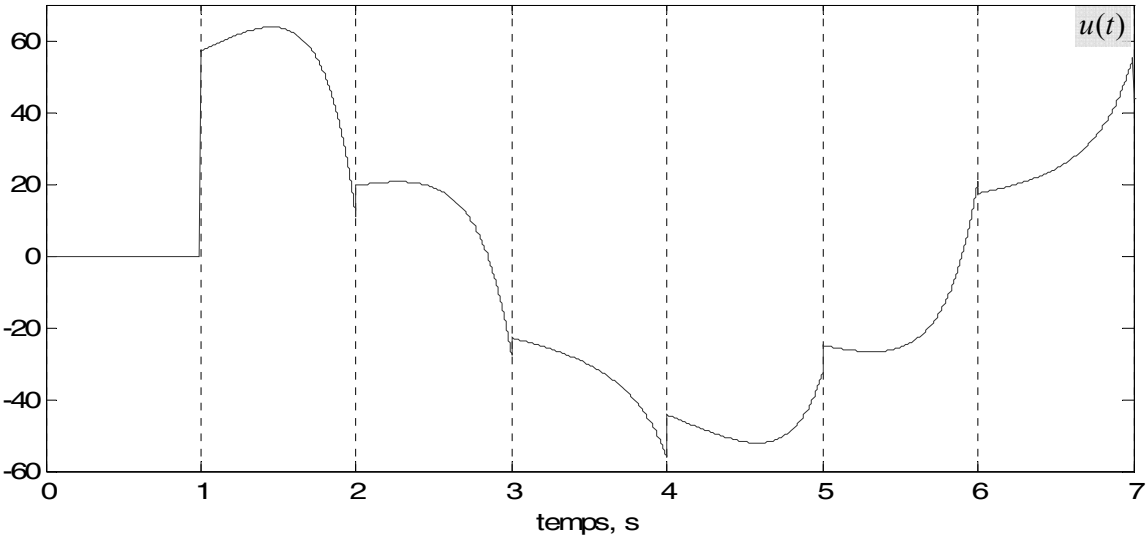


Fig. 4.14b. *Poursuite échantillonnée optimisée, T = 1s : la commande*

Les résultats de la figure 4.14 montrent que les oscillations présentes dans le §4.3.3.2.1 sont atténuées grâce à l'optimisation du contrôleur.

4.3.4. Conclusion

Dans le cas général, les CCM permettent de réaliser la poursuite échantillonnée d'une trajectoire d'état avec un retard d'une période de commutation. Le principe d'utilisation

de base est de fixer une condition de poursuite en imposant une consigne d'état et une période de commutation.

La performance des contrôleurs est meilleure s'ils commutent à période T courte, car l'adéquation entre l'état et sa consigne est vérifiée à des instants rapprochés (dans les exemples, nous avons choisi volontairement de longues périodes pour des raisons de démonstration). Toutefois, le formalisme de base des CCM ne permet pas de faire $T = 0$, l'espace de commutation étant nécessairement discret.

Les tests effectués montrent la robustesse des CCM face aux bruits et aux variations des paramètres du processus.

Par ailleurs, le formalisme est particulièrement adapté au retour d'état échantillonné. Il suffit de faire commuter le CCM à une période égale à celle de l'échantillonnage du retour pour exploiter au mieux les propriétés du CCM.

À noter que dans le cas non optimisé, il est possible d'imposer une trajectoire de sortie, évitant ainsi de construire un modèle de référence pour imposer une consigne d'état.

4.4. Contrôleur bi-échantillonnée

Dans ce cas, nous traitons des CFM basés sur des SLBE. Un CBE représente donc un système associé dont l'état évolue de manière linéaire et *échantillonnée* selon S_i entre les instants de commutation. Quant à la commande générée, elle sera par conséquent de forme *échantillonnée* par morceaux.

Un CBE alimente le processus linéaire MIMO continu (4.1) par une commande bloquée-échantillonnée selon S_i et requiert un retour discrétisé de son état (selon S_k en non optimisé et selon S_i en optimisé). Le processus décrit en (4.1) est donc *vu* comme un système discret décrit (avec les notations du formalisme des SLBE (chapitre 2)) par :

$$x_k^{i+1} = f.x_k^i + h.u_k^i \text{ pour } i = 0,...,q-1 \text{ sur un morceau } k, \tag{4.21a}$$

$$x_k^0 = x_{k-1}^q \qquad \forall t_k^0, \tag{4.21b}$$

$$y_k^i = c.x_k^i \qquad \forall t_k^i, \tag{4.21c}$$

avec $$f = e^{a.t_e} \text{ et } h = f.\int_0^{t_e} e^{-a.\tau}.b.d\tau, \tag{4.21d}$$

t_e étant la période d'échantillonnage.

Dans le contexte du CBE, les instants discrets sont notés t_k^i, où k désigne l'échelle de temps relative aux commutations et i l'échelle de temps relative au fonctionnement du système entre deux instants de commutation relatifs au contrôleur. Dans ce sens, le système d'équation (4.21) représente l'évolution échantillonnée du processus selon $S_i = \{i.t_e, i = 0,...,q-1\}$ sur un morceau k de période T constante ($S_k = \{k.T, k = 0,1,2,...\}$) telle qu'il existe q échantillons sur un morceau k. La période d'échantillonnage t_e constante est donc telle que $T = q.t_e$, avec q entier positif.

4.4.1. CBE non optimisé

Dans ce paragraphe, nous définissons le CBE de sorte à ce qu'il réalise la poursuite échantillonnée selon $S_k = \{k.T, k = 0,1,2,...\}$. L'équation de poursuite est alors :

$$x((k+1).T) = c(k.T),\ \forall k = 0,1,2,... \tag{4.22a}$$

Selon la notation des échelles de temps du formalisme des SLBE, cette relation peut se traduire par :

$$x_{k+1}^0 = c_k^0,\ \forall k = 0,1,2,... \tag{4.22b}$$

ou encore $x_k^{q_k} = c_k^0$ (4.22c)

Comme dans le cas du CCM non optimisé, nous omettrons l'entrée continue du SLBE dans la mise en œuvre du CBE non optimisé (selon la remarque 4.3). La matrice B dans $\Sigma_d(S_i, S_k, A, B, \beta_d, C)$ peut donc être considérée nulle.

4.4.1.1. Mise en équations

Afin de satisfaire la condition de poursuite (4.22), nous choisissons pour le CBE, un SLBE défini par $\Sigma_d(S_i, S_k, \alpha, 0, \beta_d, \gamma)$ avec :

- des instants de commutations définis par : $S_k = \{k.T, k = 0,1,2,...\}$ (morceaux de durée constante et égale à T),
- des instants d'échantillonnage à l'intérieur de chaque morceau définis par $S_i = \{i.t_e, i = 0,...,q-1\}$, (le fonctionnement entre les instants de commutation est régi par une période d'échantillonnage constante t_e telle que $T = q.t_e$, avec q : entier positif),
- $\dim(\lambda(t)) = \dim(x(t)) = n$,
- $\dim(w(t)) = \dim(u(t)) = r$,

- $\dim(\alpha) = n \times n$,
- $\dim(\gamma) = r \times n$ avec γ de rang plein,
- une entrée à discrétiser $\psi(t)$ provenant d'une fonctionnelle Ψ (définition 4.1),
- aucune entrée continue ($\varphi(t) = 0 \quad \forall t$).

Nous pouvons ainsi décrire le fonctionnement du système bouclé tel que dans la figure (4.1) par le système d'équations suivant :

$$x_k^{i+1} = f.x_k^i + h.u_k^i \qquad \text{pour } i = 0,...,q-1 \text{ sur un morceau } k, \tag{4.23a}$$

$$x_k^0 = x_{k-1}^q \qquad \forall t_k^0 \in S_k \tag{4.23b}$$

$$\lambda_k^{i+1} = \alpha.\lambda_k^i \qquad \text{pour } i = 0,...,q-1 \text{ sur un morceau } k, \tag{4.23c}$$

$$\lambda_k^0 = \beta_d.\psi_k^0 \qquad \forall t_k^0 \in S_k \tag{4.23d}$$

$$u_k^i = w_k^i = \gamma.\lambda_k^i \qquad \forall t_k^i \tag{4.23e}$$

L'évolution du processus est représentée par (4.23a et 4.23b) et celle du CBE est donnée par (4.23c à 4.23e). La liaison CBE-processus est incluse dans (4.23e).

En combinant les équations de (4.23) sur le morceau *k*, il vient :

$$x_k^q = f^q.x_k^0 + \sum_{j=1}^{q} f^{q-j} h.\gamma.\alpha^{j-1}.\lambda_k^0 \tag{4.24}$$

Ce qui donne sous forme matricielle :

$$x_k^q = f^q.x_k^0 + N.\lambda_k^0 \tag{4.25a}$$

$$\text{avec } N = \left[f^{q-1}.h \mid f^{q-2}.h \mid \cdots \mid f^0.h \right] \begin{bmatrix} \gamma.\alpha^0 \\ \gamma.\alpha^1 \\ \vdots \\ \gamma.\alpha^{q-1} \end{bmatrix} \in \Re^{n \times n} \tag{4.25b}$$

Ainsi, compte tenu de (4.23d) la condition de poursuite (4.22c) impose, sous réserve de l'existence de N^{-1} :

$$\lambda_k^0 = \beta_d.\psi_k^0 = N^{-1}.(c_k^0 - f^q.x_k^0) \tag{4.26}$$

Si N^{-1} existe[11], alors on peut définir le CBE non optimisé avec :

$$\beta_d = N^{-1}, \tag{4.27a}$$

$$\psi(t) = c(t) - f^q.x(t). \tag{4.27b}$$

4.4.1.2. Architecture

Grâce à la mise en équations, la construction d'un CBE non optimisé se fait très aisément. La figure 4.15 résume le paramétrage de SLBE donne le schéma de construction.

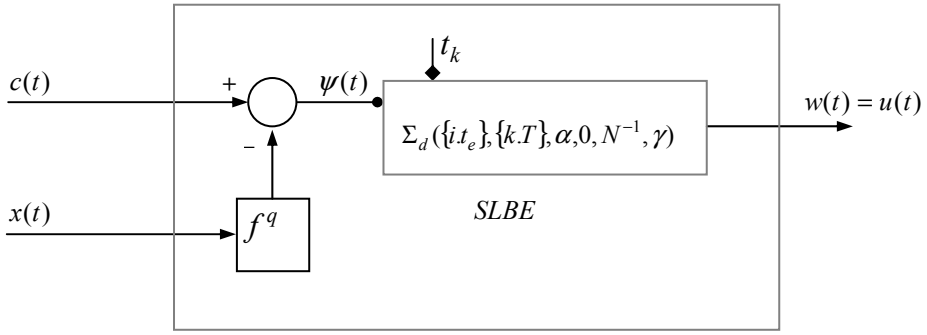


Fig. 4.15. *Architecture du CBE non optimisé*

4.4.1.3. Synthèse

Le CBE non optimisé garantit, comme le CCM non optimisé, la poursuite échantillonnée d'une consigne d'état par le processus avec un retard d'une période de commutation. L'architecture donnée dans le paragraphe précédent montre que le contrôleur est construit à partir d'un SLE défini par $\Sigma_d(\{i.t_e\},\{k.T\},\alpha,0,N^{-1},\gamma)$ dont seule la boucle discrète est utilisée et définie selon (4.27b).

Les conditions d'existence de la poursuite non optimisée sont données en annexe IV.3. En particulier, la solution n'est possible que si q est au moins égal à l'ordre du processus à commander.

Remarque 4.6. *Comme dans le cas du CCM non optimisé, il est possible d'adapter le CBE non optimisé pour prendre en compte une consigne de sortie au lieu d'une consigne d'état.*

4.4.2. CBE optimisé

Nous considérons dans ce paragraphe, une optimisation du CBE présenté précédemment en utilisant, en plus de la boucle discrète et de la condition de poursuite, son bouclage

[11] Les conditions d'existence sont données en annexe IV.3.

« continu »[12] pour réaliser la commande entre les instants de commutation. Le CBE optimisé est défini par $\Sigma_d(\{i.t_e\},\{k.T\},\alpha,\beta_e,\beta_d,\gamma)$. L'idée est de minimiser les oscillations dans cet intervalle en paramétrant convenablement le contrôleur. Le mode de fonctionnement souhaité est défini par :

- $x_k^{q_k} = c_k^0$, $\forall k = 0,1,2,...$,
- minimiser la distance entre $c(t-T)$ et $x(t)$ sur le morceau k.

4.4.2.1. Mise en équations

Le critère de coût exprimant la contrainte supplémentaire sur le morceau *k* est le suivant :

$$J = \frac{1}{2} * \sum_{i=0}^{q-1} \left[\left(c_{k-1}^i - x_k^i\right)^T .E.\left(c_{k-1}^i - x_k^i\right) + (w_k^i)^T .G.w_k^i \right] \tag{4.28}$$

où $E \in \Re^{n \times n}$ et $G \in \Re^{r \times r}$ sont deux matrices symétriques définies positives.

La minimisation du critère *J* vise à réduire, sur un morceau *k*, l'erreur entre l'état et la consigne d'état retardée de T, tout en modérant la commande. Selon la théorie de la commande optimale, l'hamiltonien correspondant s'écrit [PDR+90] :

$$H = -\frac{1}{2} * \sum_{i=0}^{q-1} \left[\left(c_{k-1}^i - x_k^i\right)^T .E.\left(c_{k-1}^i - x_k^i\right) + (w_k^i)^T .G.w_k^i \right] + \sum_{i=0}^{q-1} (\lambda_k^{i+1})^T .\left[f.x_k^i + h.w_k^i \right] \tag{4.29}$$

λ_k^i étant le vecteur multiplicateur de Lagrange de dimension *n*.

Le principe de Pontryagin conduit à :

$$\lambda_k^i = \partial H / \partial x_k^i = E.(c_{k-1}^i - x_k^i) + f^T \lambda_k^{i+1}$$

soit $\lambda_k^{i+1} = (f^T)^{-1} \lambda_k^i - (f^T)^{-1}.E.(c_{k-1}^i - x_k^i)$ (4.30a)

et $w_k^i = G^{-1} \cdot h^T .\lambda_k^{i+1}$. (4.30b)

4.4.2.2. Interprétation

Les équations (4.30a) et(4.30b) peuvent être interprétées respectivement comme les équations d'état et de sortie du SLBE (interne au contrôleur) sur le morceau *k*. L'entrée du SLBE est $(c_{k-1}^i - x_k^i)$.

[12] Le qualificatif « continu » utilisé pour garder l'analogie avec les CCM est abusif. Le qualificatif suréchantillonné serait mieux adapté

Il reste à déterminer la valeur de λ_k^0 permettant de satisfaire la condition de poursuite telle que défini en (4.22c).

4.4.2.3. Calcul de la condition initiale

Pour calculer la valeur initiale de l'état du CBE à chaque commutation, on considère le système augmenté suivant obtenu en combinant (4.21a), (4.30a) et (4.30b) :

$$\begin{bmatrix} x_k^{i+1} \\ \lambda_k^{i+1} \end{bmatrix} = \mathrm{H}.\begin{bmatrix} x_k^{i} \\ \lambda_k^{i} \end{bmatrix} + \mathrm{K}.c_{k-1}^{i} \tag{4.31a}$$

avec :

$$\mathrm{H} = \begin{bmatrix} f + h.G^{-1}.h^T.(f^T)^{-1}.E & h.G^{-1}.h^T.(f^T)^{-1} \\ (f^T)^{-1}.E & (f^T)^{-1} \end{bmatrix},\ \mathrm{K} = \begin{bmatrix} -h.G^{-1}.h^T.(f^T)^{-1}.E \\ -(f^T)^{-1}.E \end{bmatrix} \tag{4.31b}$$

La résolution se traduit par une solution de la forme :

$$\begin{bmatrix} x_k^{q} \\ \lambda_k^{q} \end{bmatrix} = \mathrm{H}^q.\begin{bmatrix} x_k^{0} \\ \lambda_k^{0} \end{bmatrix} + \left[\mathrm{H}^{q-1}.\mathrm{K}\cdots\mathrm{H}^{q-2}.\mathrm{K}\cdots\mathrm{H}^{0}.\mathrm{K}\right]\begin{bmatrix} c_{k-1}^{0} \\ c_{k-1}^{1} \\ \vdots \\ c_{k-1}^{q-1} \end{bmatrix} \tag{4.32}$$

Ce qui donne :

$$x_k^q = \Theta_{11}.x_k^0 + \Theta_{12}.\lambda_k^0 + \mathrm{I}^h \tag{4.33a}$$

$$\text{avec } \begin{bmatrix} \Theta_{11} & \Theta_{12} \\ \Theta_{21} & \Theta_{22} \end{bmatrix} = \mathrm{H}^q \text{ et } \begin{bmatrix} \mathrm{I}^h \\ \mathrm{I}^b \end{bmatrix} = \left[\mathrm{H}^{q-1}.\mathrm{K}\cdots\mathrm{H}^{q-2}.\mathrm{K}\cdots\mathrm{H}^{0}.\mathrm{K}\right]\begin{bmatrix} c_{k-1}^{0} \\ c_{k-1}^{1} \\ \vdots \\ c_{k-1}^{q-1} \end{bmatrix}, \tag{4.33b}$$

$\mathrm{I}^h \in \Re^{n\times 1}$ et $\mathrm{I}^b \in \Re^{n\times 1}$ étant les moitiés supérieure et inférieure de $\begin{bmatrix} \mathrm{I}^h & \mathrm{I}^b \end{bmatrix}^T$.

Donc la condition initiale de l'état du CBE à chaque commutation est :

$$\lambda_k^0 = \Theta_{12}^{-1}.\left[c_k^0 - \Theta_{11}.x_k^0 - \mathrm{I}^h\right] \tag{4.34}$$

4.4.2.4. Identification des paramètres du contrôleur

Si Θ_{12}^{-1} existe, le CBE optimisé, $\Sigma_d(\{i.t_e\},\{k.T\},\alpha,\beta_e,\beta_d,\gamma)$, est complètement défini par :

$$\lambda_k^{i+1} = \alpha.\lambda_k^i + \beta_e(c_{k-1}^i - x_k^i) \text{ pour } i = 0,...,q-1 \text{ sur un morceau } k, \quad (4.35a)$$

$$\lambda_k^0 = \beta_d.\psi_k^0 \qquad \forall t_k^0 \in S_k \quad (4.35b)$$

$$u_k^i = w_k^i = \gamma.\lambda_k^i \qquad \forall t_k^i \quad (4.35c)$$

avec :

- $\alpha = (f^T)^{-1}$ et $\beta_e = (f^T)^{-1}.E$ selon (4.30a),
- $\gamma = G^{-1}.h^T$ selon (4.30b),
- $\varphi(t) = c(t-T) - x(t)$,
- $\beta_d = \Theta_{12}^{-1}$, selon (4.34),
- $\psi(t) = c(t) - \Theta_{11}.x(t) - \mathrm{I}^h$ tel que $\psi_k^0 = c_k^0 - \Theta_{11}.x_k^0 - \mathrm{I}^h$ aux instants de commutation.
- I^h calculé selon (4.33b).

4.4.2.5. Architecture

Selon l'identification des paramètres, il est possible de proposer une construction du CBE optimisé selon la figure 4.16.

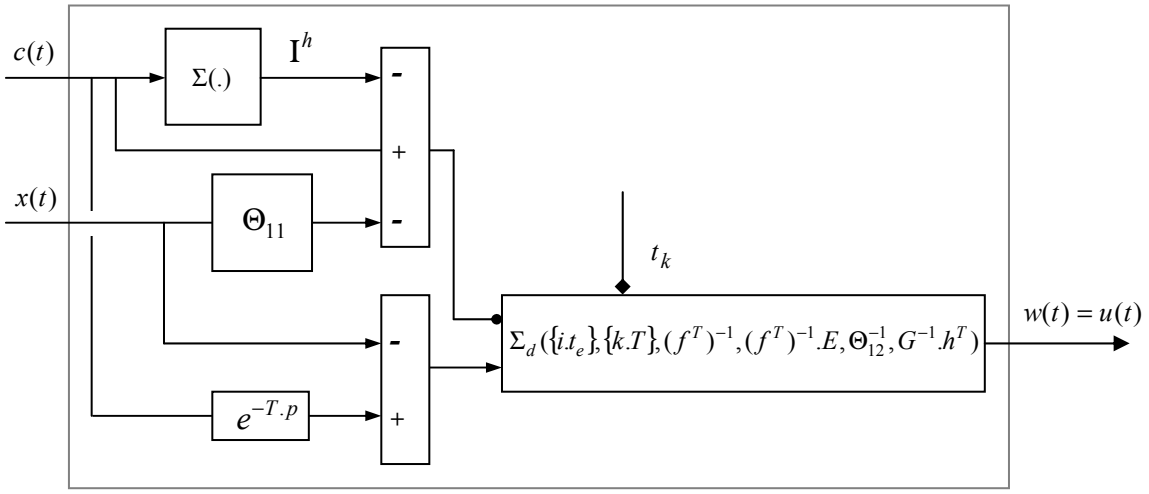


Fig. 4.16. *Architecture du CBE optimisé*

Sur cette figure, $\Sigma(.)$ est représente le calcul de I^h selon (4.33b). En pratique, I^h peut être interprété comme étant la sortie d'un SBE auxiliaire défini par $\Sigma_d(\{i.t_e\},\{k.T\},\mathrm{H},\mathrm{K},0,[I_n \quad 0])$ avec $\psi(t) = 0$ et $\varphi(t) = c(t-T)$.

4.4.2.6. Synthèse

Le CBE optimisé est complètement défini par l'architecture décrite dans le paragraphe précédent. Il exploite complètement la structure d'un CFM (figure 4.1) en utilisant à la fois la boucle discrète pour garantir la poursuite aux instants de commutation (comme dans le cas non optimisé) et la boucle « continue » pour réduire, en plus, les oscillations éventuelles entre ces instants.

Les conditions d'existence de la poursuite optimisée par le CBE sont données en annexe IV.4.

4.4.3. Validation expérimentale

De la même manière que les CCM, nous avons réalisé les CBE sur calculateur numérique à l'aide du logiciel Simulink®. Quelques exemples sont proposés pour comparer le fonctionnement d'un CBE à celui d'un CCM. Dans ce sens, il est important de considérer la remarque suivante.

Remarque 4.7. *Nous comprenons que si l'adéquation de l'état et de sa consigne retardée d'une période de commutation est assurée par tous les types de contrôleurs présentés, l'évolution du système bouclé, dans les cas non optimisé dépend soit de (4.3b) pour un CCM soit de (4.23c) pour un CBE. Ainsi, si l'on veut faire évoluer un CBE de la même manière que son homologue continu, il faut choisir* $\alpha_{CBE} = e^{\alpha_{CCM}.t_e}$ *pour faire l'équivalent discret (*$\lambda_k^{i+1} = e^{\alpha.t_e}.\lambda_k^i$*) de l'équation continue* $\lambda'(t) = \alpha.\lambda(t)$.

4.4.3.1. CBE non optimisé

Considérons un processus tel que défini par (4.1) avec :

$$a = \begin{bmatrix} 0 & 1 \\ 5 & -4 \end{bmatrix},\ b = \begin{bmatrix} 0 \\ 7 \end{bmatrix},\ \text{et}\ x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

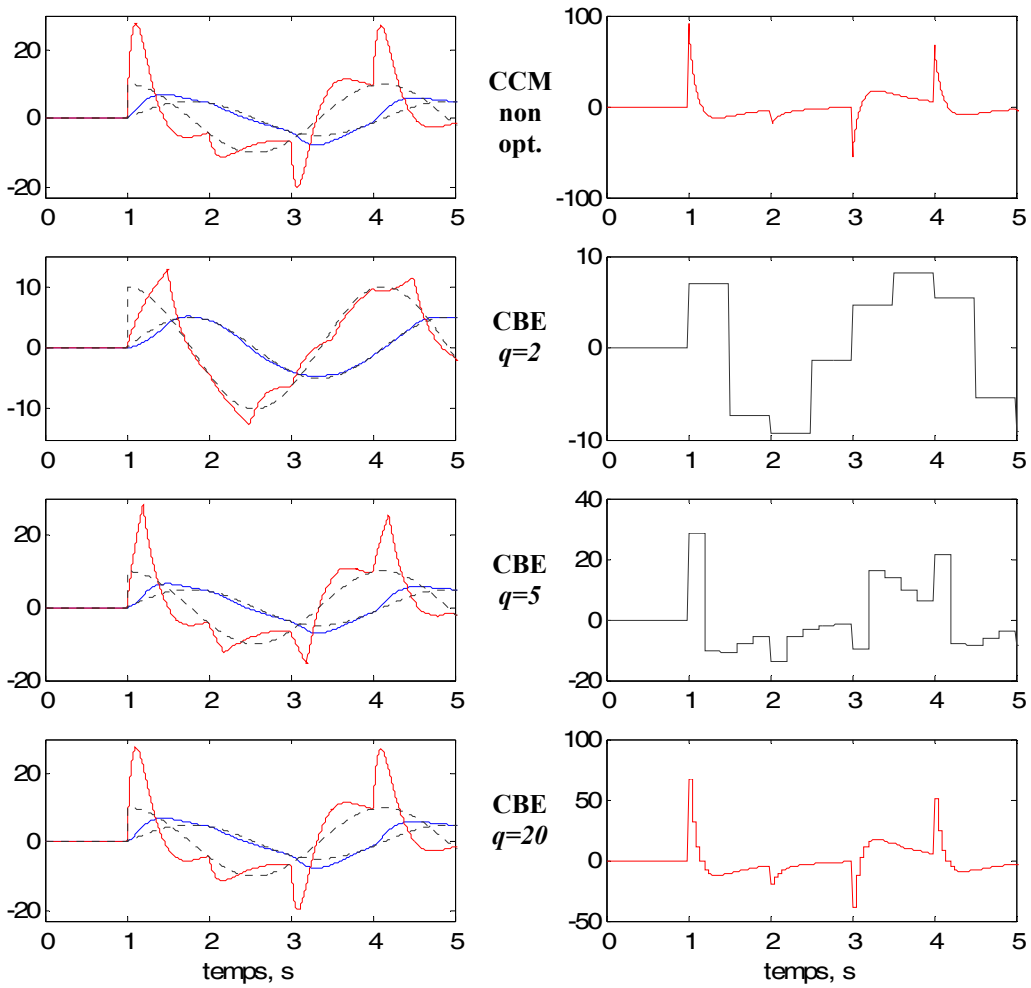
On souhaite lui imposer une consigne d'état donnée par :

$$c(t) = \begin{bmatrix} 5\sin(2t) \\ 10\cos(2t) \end{bmatrix}$$

Nous réalisons ici un asservissement par un CBE et un autre par un CCM, les contrôleurs étant défini respectivement par $\Sigma_d(\{i.(T/q)\},\{k.T\},e^{\alpha.(T/q)},0,N^{-1},\gamma)$ et $\Sigma_c(\{k.T\},\alpha,0,M^{-1},\gamma)$ avec :

$$\alpha = \begin{bmatrix} -2 & 0 \\ 0 & -10 \end{bmatrix},\ \gamma = [10 \quad 10] \text{ et } T = 1s$$

Les résultats relatifs au CBE pour différentes valeurs de q, illustrés figure 4.17, sont comparés à ceux du CCM non optimisé équivalent. La poursuite est garantie à chaque instant de commutation mais des oscillations importantes existent sur chaque morceau k. Nous constatons, par ailleurs que la commande CBE devient proche de celle du CCM quand q augmente (en rouge sur la courbe des commandes).



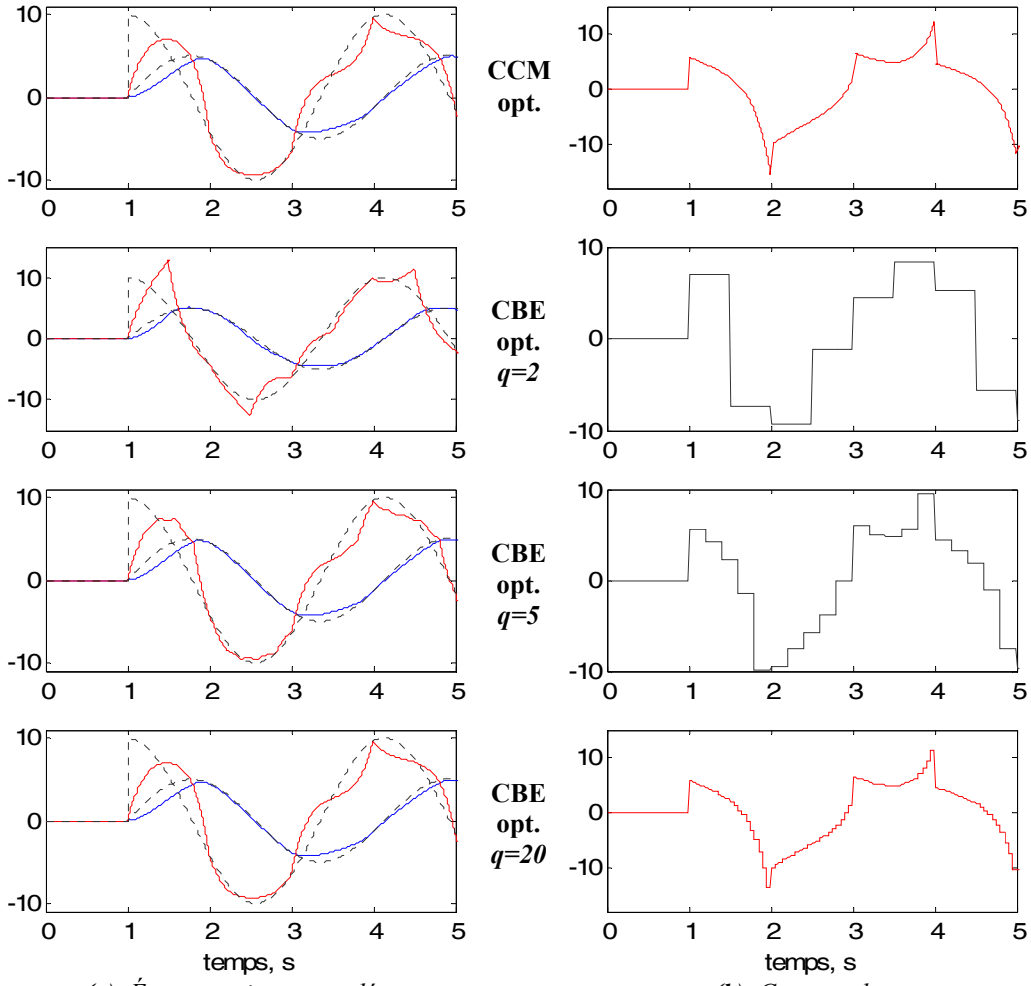
(a). *État - consigne retardée* ***(b).*** *Commande*

Fig. 4.17. *CCM et CBE non optimisé, T = 1s*

4.4.3.2. CBE optimisé

Nous reprenons exactement le même cas de figure que précédemment en proposant cette fois ci, une commande par un CBE optimisé défini l'architecture donnée au §4.4.2.5 avec $E = 0.01 \times I_n$ et $G = 100$.

Les résultats donnés en figure 4.18 offrent une comparaison avec la version non optimisée. Nous constatons d'une part que l'évolution entre les instants de commutation est améliorée et d'autre part que la commande est fortement atténuée dans le cas optimisé.



(a). *État - consigne retardée* ***(b).*** *Commande*

Fig. 4.18. *CCM et CBE optimisé,* $T = 1s$

Comme précédemment, nous proposons une comparaison avec un CCM optimisé. Il est évident que le CBE optimisé tend vers le comportement du CCM optimisé pour une valeur élevée de q.

Nous constatons en pratique que la performance d'un CBE optimisé est meilleure avec une faible valeur de E et une grande valeur de G.

Remarque 4.8. *Il existe une solution quasi-optimale en choisissant la valeur de q égale à l'ordre n du processus à commander : les résultats sont les mêmes sur les figures 4.17 et 4.18 pour* $q = 2$. *En effet, dans ce cas, il n'existe qu'une seule solution pour que la commande satisfasse la condition de poursuite. De ce fait, la solution est forcément la même dans les deux cas (optimisé ou non optimisé).*

4.4.4. Conclusion

Le fonctionnement des CBE est quasiment le même que celui des CCM. La seule différence réside dans l'évolution échantillonnée de la commande entre les instants de commutation. De ce fait, la commande générée est nécessairement *constante* sur un morceau $]i.t_e,(i+1).t_e]$.

Tout comme les CCM, les CBE sont robustes face aux bruits et aux variations des paramètres du processus et leur performance est supérieure dans le cas d'une commutation rapide. Ils sont également adaptés aux retours échantillonnés et nous pouvons envisager une consigne de sortie dans le cas non optimisé.

Toutefois, ils offrent un degré de liberté supplémentaire (paramètre q) concernant l'évolution entre les instants de commutation.

4.5. Commande adaptative

Un système adaptatif, au sens large, peut être considéré comme une structure qui, sans aide extérieure peut s'adapter aux changements de son environnement pour maintenir l'objectif défini au préalable. Plusieurs auteurs se sont intéressés à ce type de systèmes [AMS58, AW95, KHH01, Del06].

Les architectures de commande requièrent, en général, les paramètres du processus à contrôler, d'où la nécessité de la phase préalable d'identification dans le cas de processus invariants. Toutefois, si ce dernier présente une variation de ses paramètres, il est intéressant de considérer une structure de commande incluant un identificateur en ligne destiné à fournir au contrôleur, la valeur des paramètres changeant en temps réel. En effet, même si les contrôleurs proposés dans ce chapitre sont robustes face aux variations de paramètres du processus, il y a un risque de *décrochage* de la poursuite si les changements sont notoires, comme dans le cas d'un processus multimodèle.

Remarque 4.9. *Les changements temporels des paramètres peuvent représenter, par exemple, un changement du comportement du processus lié à des perturbations externes dans l'environnement indépendamment de l'entrée appliquée. Les paramètres du METC d'un moteur électrique en charge, par exemple, sont différents de ceux à vide.*

Dans ce sens, vu la compatibilité marquante entre la méthode de clonage et les CFM, nous proposons une combinaison des méthodes d'identification en ligne et de la commande pour bâtir une structure de commande *adaptative*. L'aptitude adaptative repose sur l'identificateur en ligne. Elle est donc soumise aux conditions de fonctionnement du SFC dans le cas des processus changeant (§3.6.1.3).

La figure 4.19 montre une structure adaptative SFM où le contrôleur utilise directement les valeurs courantes des estimateurs des paramètres du processus provenant de l'identificateur en ligne.

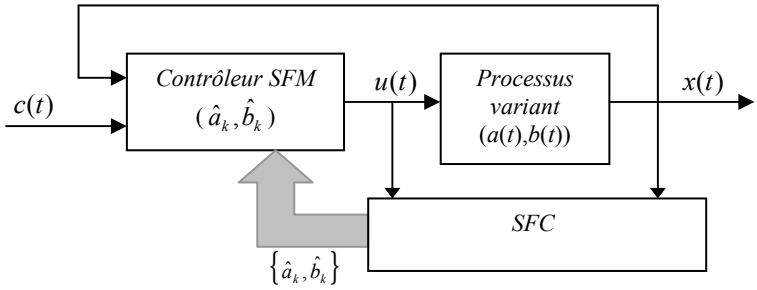


Fig. 4.19. *Structure de commande adaptative*

4.5.1. Compatibilité identificateur/contrôleur

La réalisation d'une telle structure adaptative est motivée par la compatibilité entre l'identificateur et le contrôleur développés au cours de ce mémoire. Utilisant tous deux une représentation d'état du processus à traiter, le contrôleur utilise immédiatement les paramètres identifiés par l'identificateur en ligne (SFC). Parallèlement, le contrôleur à fonctionnement par morceaux permet de remplir les conditions requises par le SFC sur l'entrée du processus.

4.5.2. Mise en œuvre

Selon sa définition (chapitre 3) l'identificateur en ligne (SFC) requiert une entrée *riche*, qui est traduit par des discontinuités sur un morceau k relatif au SFC. Afin d'assurer une telle entrée, deux possibilités sont offertes :

- l'utilisation d'un CBE commutant, par exemple, de façon synchrone au clone : les discontinuités proviennent alors de l'évolution échantillonnée de la commande générée entre les instants de commutation,
- l'utilisation d'un CCM commutant plus rapidement que le clone : les discontinuités proviennent alors de la commutation de l'état du CCM (et donc sur la commande) entre les instants de commutation du clone.

Toutefois, même s'il est possible d'utiliser un CBE, il est préférable d'opter pour le CCM qui utilise directement les paramètres du METC identifiés par le SFC, réduisant ainsi le coût de calcul temps réel.

Remarque 4.10. *Dans le fonctionnement de base d'un CCM (asservissement de processus invariants), l'architecture du contrôleur se résume par une construction telle que présentée en figure 4.2, les matrices* $e^{a.T}$ *et* $\beta_d = M^{-1}$ *étant constantes et donc calculables préalablement et de manière hors ligne. Toutefois, dans l'architecture adaptative, ces matrices doivent obligatoirement être évaluées en ligne par des méthodes numériques.*

4.5.3. Validation expérimentale

4.5.3.1. Variations continues des paramètres du processus

Soit un processus variant décrit par :

$$a(t) = \begin{bmatrix} 0 & 1 \\ -4 & a_{22}(t) \end{bmatrix},\ b = \begin{bmatrix} 0 \\ 4 \end{bmatrix} \text{ et } x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{ avec } a_{22}(t) = -2(1 - 2\sin(0.2t)).$$

Nous souhaitons imposer à ce processus une consigne d'état : $c(t) = \begin{bmatrix} \frac{5}{2}(1+\cos(4t)) \\ 10\sin(4t) \end{bmatrix}$.

Pour ce faire, nous proposons de réaliser une structure adaptative avec :

- Un SFC à fonctionnement *boîte grise* utilisant un bloc adaptatif pour identifier a_{22} en ligne. Les autres paramètres sont fixés convenablement dans $\hat{\varphi}_k^T$ et la condition initiale de l'estimateur de a_{22} est : $(\hat{\varphi}_{22}^T)_0 = (\hat{a}_{22})_0 = 1$. Par ailleurs, par rapport à la dynamique de la variation temporelle du paramètre, nous commutons le clone à période $T_{SFC} = 100ms$. Enfin, nous utilisons un gain adaptatif $g_k^i = g^i = 1 \times 10^{20} \quad \forall i$ garantissant une convergence rapide du clonage.

- Un CCM non optimisé, cadencé à $T_C = 50ms < T_{SFC}$ pour satisfaire la condition du clonage et défini par :

$$\alpha = \begin{bmatrix} -1 & 4 \\ -2 & -1 \end{bmatrix},\ \gamma = \begin{bmatrix} 10 & -10 \end{bmatrix}.$$

  À noter que le CCM admet comme entrées les estimateurs des paramètres pour réaliser les calculs de $e^{a.T}$ et $\beta_d = M^{-1}$ en ligne.

La figure 4.20 illustre les résultats obtenus. Afin de comparer la performance de la structure adaptative à celle du CCM classique, nous avons, pendant les dix premières secondes, alimenté le CCM par des paramètres *a* et *b* constants, notamment $a_{22} = -2$ (valeur moyenne du paramètre changeant). Ensuite, à $t = 10s$, nous avons commuté les

entrées du CCM aux valeurs des estimateurs pour réaliser la commande adaptative. À noter que les deux composantes de la consigne sont retardées convenablement, conduisant à une superposition des signaux sur les figures 4.20b et 4.20c.

L'identificateur en ligne identifie pendant toute la durée de la simulation le paramètre changeant (figure 4.20a).

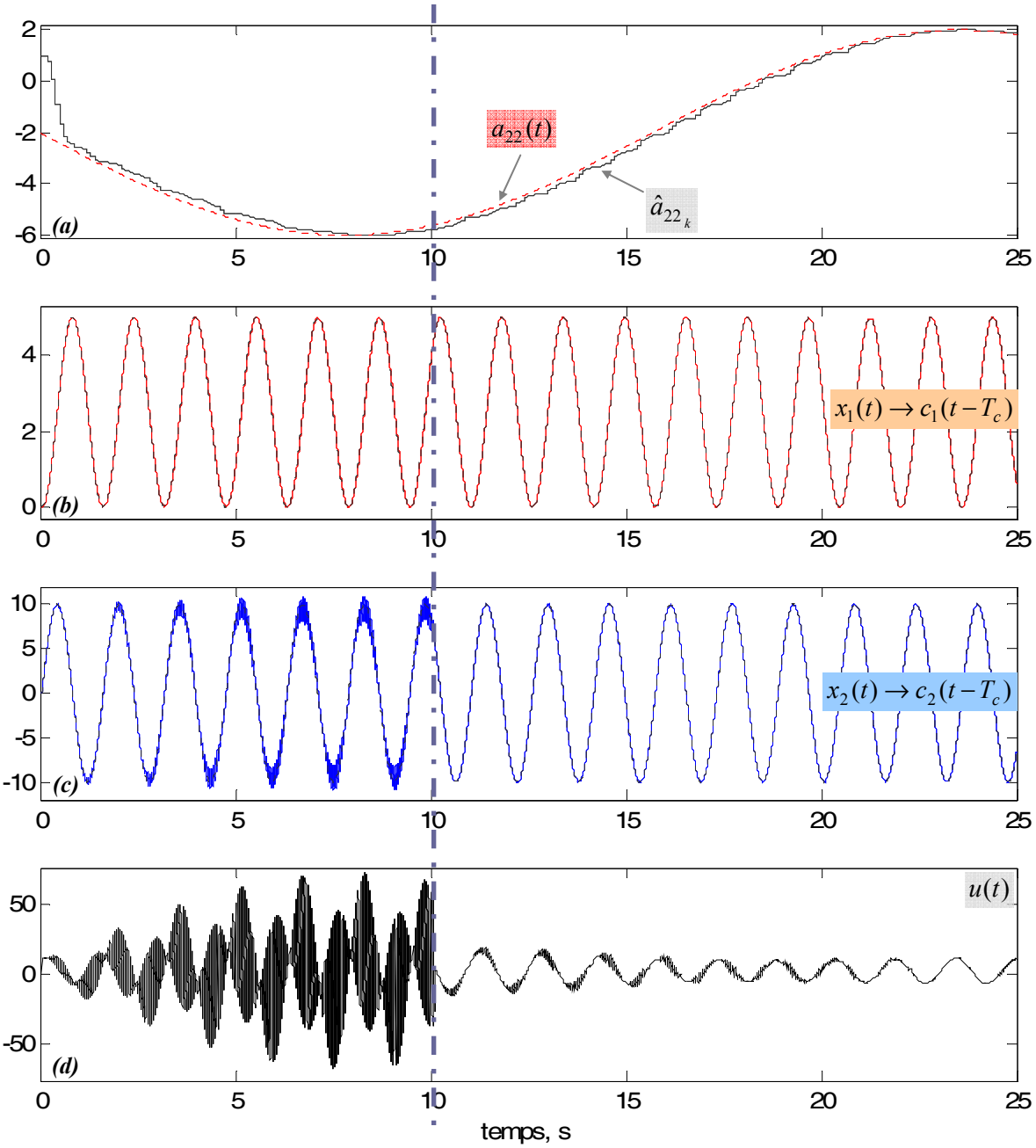


Fig. 4.20. *Commande adaptative : variation continue du processus*

Dans la zone non adaptative $t<10$, nous constatons (figure 4.20b) que la robustesse du CCM permet d'asservir correctement la première composante de l'état même quand $a_{22}(t)$ s'éloigne de la valeur utilisée par le CCM ($t \to 10^-$). Par contre, nous observons

(figure 4.20d) que même si la commande est fortement modulée pour prendre en compte la variation du processus (robustesse) quand $t \to 10^-$, la deuxième composante présente des oscillations en cas de fortes courbures de la consigne à suivre (figure 4.20c).

Dans la zone adaptative ($t > 10$), la deuxième composante de l'état suit parfaitement sa consigne (figure 4.20c) avec une commande plus *lisse* (figure 4.20d), quelle que soit la valeur du paramètre changeant.

4.5.3.2. Variations discontinues : processus multimodèle

Considérons un processus *bimodèle* instable décrit par $x'(t) = a_i.x(t) + b_i.u(t)$, $i \in \{1,2\}$ avec :

$$a_1 = \begin{bmatrix} 0 & 1 \\ 5 & 20 \end{bmatrix},\ b_1 = \begin{bmatrix} 0 \\ 10 \end{bmatrix},\ a_2 = \begin{bmatrix} 0 & 1 \\ 7 & 10 \end{bmatrix},\ b_2 = \begin{bmatrix} 0 \\ 15 \end{bmatrix} \text{ et } x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Le processus commute entre ces deux modèles selon la règle suivante :

$$i = \begin{cases} 1 & \text{si } 0 \le t < 10s \\ 2 & \text{si } t > 10s \end{cases}$$

Nous souhaitons imposer à ce processus la même consigne d'état que dans le cas précédent. La structure adaptative que nous utilisons possède le même CCM que précédemment. Par contre, le SFC est en fonctionnement *boîte noire* utilisant deux blocs adaptatifs pour identifier chacun des paramètres du processus en ligne. La commutation du clone est à période $T_{SFC} = 100ms$ (supérieure à $T_C = 50ms$ pour garantir le clonage) et le gain adaptatif est $g_k^i = g^i = 1 \times 10^{20} \quad \forall i$. Le SFC part d'une condition initiale donnée par :

$$\hat{\varphi}_0^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

La figure 4.21 illustre les résultats obtenus. Comme précédemment, les deux composantes de la consigne sont retardées convenablement, conduisant à une superposition des signaux sur les figures 4.21c et 4.21d.

Les figures 4.21b et 4.21b représente l'identification en ligne des paramètres du processus par le SFC. Nous observons deux processus de clonage l'une démarrant à $t = 0s$ et l'autre à $t = 10s$. À chaque fois, il existe une phase transitoire liée à des conditions initiales éloignées.

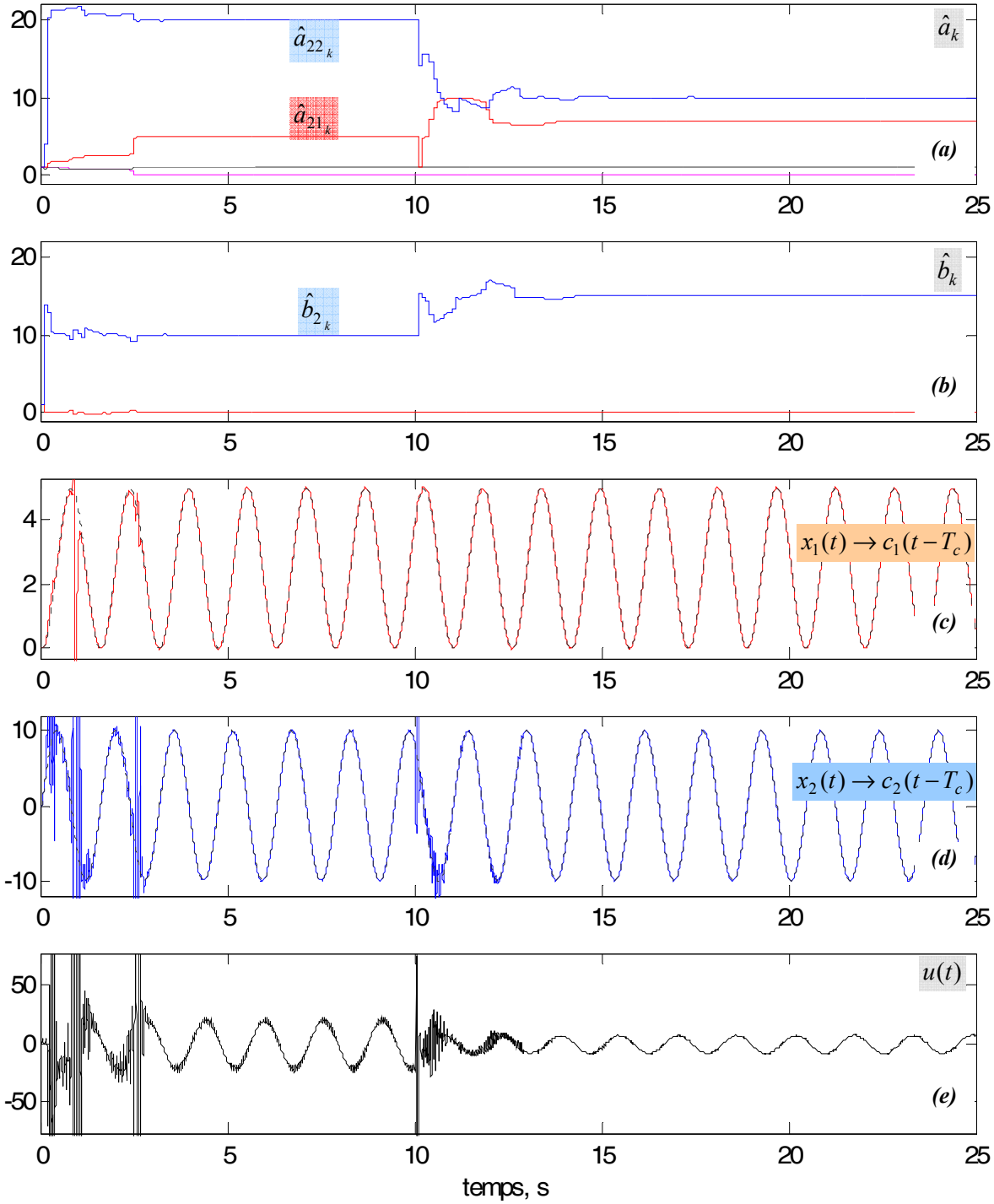


Fig. 4.21. *Commande adaptative : processus bimodèle*

La poursuite est illustrée figures 4.21c et 4.21d. Les oscillations s'expliquent d'une part par le fait que le SFC qui délivre de *fausses* estimations des paramètres au CCM pendant les phases transitoires et d'autre part que les deux modèles du processus sont instables.

La commande est représentée figure 4.21e.

Les essais effectués sur le présent processus avec un CCM classique utilisant les paramètres du premier modèle montrent que son état diverge à l'instant de passage au deuxième

modèle. Cela s'explique d'une part par le fait que les paramètres sont très différents du premier modèle et d'autre part que le processus est instable dans tous les cas. Ainsi, même si le CCM est une méthode robuste, de telles situations ne peuvent être prises en compte que par une structure adaptative ou alors par la connaissance *a priori* des modèles du processus et de la règle de commutation.

4.6. Conclusion

Nous avons établi, au cours de ce chapitre, la formalisation de contrôleurs basés sur les SFM. Deux types de CFM sont présentés : les contrôleurs à fonctionnement continu par morceaux (CCM) et ceux à fonctionnement bi-échantillonné (CBE). Ces CFM sont très facilement transposable sur calculateur numérique.

Les CFM se basent sur un retour d'état pour réaliser la poursuite échantillonnée d'une consigne d'état (ou éventuellement de sortie dans le cas non optimisé) par l'état (ou la sortie) du processus avec un retard d'une période de commutation et une coïncidence exacte aux instants de commutation ($x_{k+1} = c_k$ ou $y_{k+1} = (c_s)_k$).

Dans le cas des CFM non optimisés, la poursuite est garantie uniquement à ces instants. Le retour n'est pris en compte qu'aux instants de commutation et le processus évolue en roue libre sur un morceau *k*, avec une entrée qui ne dépend que du choix des matrices caractéristiques du CFM non optimisé. Par ailleurs, dans le cas d'une période de commutation élevée, il est judicieux de considérer les CFM optimisés qui minimisent l'écart entre l'état et sa consigne sur le morceau *k*, tout en assurant la poursuite échantillonnée.

Les CFM, tels que présenté dans ce chapitre, permettent de contrôler des processus linéaire MIMO. Dans le cas continu, la méthode permet d'atteindre de très bonnes performances avec un choix de commutation rapide[13] du contrôleur. Par ailleurs, le contrôle des processus à retour échantillonné est particulièrement réalisable par les CFM dont la commutation est alors synchronisée avec l'arrivée des données.

Les exemples présentés montrent la robustesse de l'asservissement CFM face à des perturbations telles que le bruit sur la mesure d'état ou la variation temporelle des paramètres du processus. De plus, l'exemple d'asservissement du processus réel montre que la commande peut s'appliquer même en présence de non linéarités dans le comportement du processus à commander.

[13] La période de commutation peut être très courte, la limitation étant liée à la technologie utilisée pour implanter la méthode (à titre indicatif, nous pouvons la réduire à 0,2 ms sur une architecture temps réel utilisant la technologie dSpace).

Toutefois, la méthode réalisant une poursuite échantillonnée qui fait coïncider $x(t)$ avec $c(t-T)$ à chaque instant de commutation, la commande générée peut prendre de grandes valeurs en cas de discontinuités sur la consigne si T est faible. Afin d'atténuer la commande dans ces conditions, nous proposons l'utilisation d'un *amortisseur* qui régule automatiquement la consigne injectée au contrôleur.

Alors que les CBE utilisent un modèle discret du processus, les CCM utilisent directement les paramètres de son METC. Ainsi de par la compatibilité entre la commande et l'identification par les SFM, il est tout à fait envisageable de réaliser une structure adaptative, telle que décrite à la fin du chapitre, en utilisant un CCM et un SFC.

Le formalisme présenté offre une exploitation originale du retour d'état. Contrairement à d'autres architectures de commande, son utilisation fait appel à des calculs simples et déterministes des paramètres nécessaires à la construction du contrôleur. Toutefois, afin de traiter d'autres conditions d'utilisation, nous proposons dans le chapitre suivant, des adaptations de la méthode, dans des contextes encore plus contraignants. En particulier, les cas suivants seront traités :

- retour par la sortie,
- retour par l'état retardé,
- retour par l'état échantillonné,
- retour par la sortie retardée,
- retour par la sortie échantillonnée,
- combinaisons des cas précédents.

Chapitre 5

Adaptation des contrôleurs à fonctionnement par morceaux

Nous avons vu chapitre 4 comment réaliser des architectures de commande basées sur les SFM permettant de réaliser la poursuite échantillonnée en faisant un retour d'état (soit disponible sur le processus soit reconstruit à partir de sa sortie par un observateur). Nous proposons dans ce chapitre, des adaptations du formalisme de base des CFM pour prendre en compte divers types de retour, incluant notamment un retard et un échantillonnage. Typiquement, nous proposons des approches pour aborder la problématique mentionnée dans l'introduction générale, où la seule observation sur l'évolution du processus provient d'un capteur numérique qui engendre un retard en plus de l'échantillonnage du signal de sortie du processus. En effet, ce cas est très fréquemment rencontré dans les architectures de commande utilisant des calculateurs et des capteurs numériques parfois gourmands en temps de calcul. Il est donc intéressant de considérer l'approche CFM qui est particulièrement adaptée à la commande numérique. Dans tous les cas, nous donnons les adaptations mathématiques, des exemples numériques simulés et des exemples sur système réel.

5.1. Motivations

Les contrôleurs par morceaux présentés au chapitre 4 conduisent à un asservissement de qualité pour les systèmes linéaires MIMO en présence d'un retour d'état. Néanmoins, pour certaines applications, il est nécessaire d'adapter ce formalisme. En effet, même s'il est très représentatif de l'évolution d'un processus, l'état n'est pas toujours entièrement disponible. Les cas que nous envisageons sont ceux pour lesquels le seul retour disponible est :

- la sortie continue,
- l'état retardé,
- l'état échantillonné,
- la sortie retardée,
- la sortie échantillonnée,
- des combinaisons retard/échantillonnage.

5.1.1. Contraintes liées au *retard*

Les problèmes liés au *retard* sont apparus très tôt. Les automaticiens l'ont rencontré dans de simples exemples de systèmes thermiques où le capteur de température, placé à une distance notable de la source de chaleur du système, engendre un retard pur dans la chaîne de retour. Nous pouvons également citer d'autres exemples comme le contrôle des processus à distance reposant sur des lignes de transmission ou sur la commutation satellitaire faisant intervenir un *retard* dans l'obtention et la transmission des informations. Le phénomène de retard peut également surgir dans la manipulation de données par ordinateur, comme dans le cas d'un traitement d'images ou encore dans les systèmes biologiques ou chimiques.

La présence d'un retard fait tendre l'ordre d'un système vers l'infini. Le tracé du lieu de Nyquist d'un système thermique du premier ordre observé par un thermostat éloigné le montre clairement. Dans tous les cas, le retard apporte une complexité supplémentaire dans le développement de lois de commande. La question de stabilisation de l'ensemble d'un système bouclé en présence d'un retard est toujours d'actualité, surtout dans le cas d'un retard important ou variable, rendant impossible la généralisation d'une méthode de commande pour tout processus linéaire ou non linéaire. En général, les ouvrages recensés traitent le cas d'un retard connu ou borné. Plusieurs approches sont proposées pour pren-

dre en compte le phénomène de retard dans le développement des lois de commande [DR94, DGR95, DRB95, DV97, RPL97, Mah00, Ric03, MD05].

Par ailleurs, le phénomène de retard peut être ramené à un problème de prédiction. En effet, ayant une information retardée, il semble intuitif de prédire l'évolution du système en fonction de son entrée pour avoir une estimation de l'information actuelle. Néanmoins, la prédiction fait appel à une bonne modélisation du processus considéré. Un modèle n'étant jamais parfait, une commande basée sur des données estimées par la prédiction perd beaucoup en robustesse face à des perturbations.

5.1.2. Contraintes liées à l'*échantillonnage*

En sus du phénomène de retard, l'avènement des calculateurs et capteurs numériques nécessite la manipulation de données discrétisées, même lors de la manipulation de processus réels continus.

Selon [Yam94], la discrétisation d'un processus continu est pénalisante dans le sens où elle engendre une perte d'information entre les instants d'échantillonnage (ou du moins la rend implicite). Certains problèmes, comme les oscillations de la variable contrôlée entre ces instants, peuvent alors survenir, comme nous l'avons constaté en chapitre 4 dans le cas d'une période de commutation importante des contrôleurs par morceaux.

Encore une fois, il est possible de se baser sur un modèle du processus pour estimer son évolution entre les instants d'échantillonnage afin d'améliorer la performance de la régulation sur une période d'échantillonnage.

On peut comprendre que la combinaison retard/échantillonnage apporte une difficulté de taille dans le développement de lois de commande. Plusieurs approches sont proposées pour tenter de répondre à cette problématique. Les asservissements classiques offrent une adaptation dans ce sens. Le régulateur PID, par exemple, se présente sous forme numérique et permet de plus de traiter des processus à retard, particulièrement dans le cas SISO. Toutefois, selon Landau [Lan88], un régulateur PID numérique ne peut s'appliquer rigoureusement qu'aux procédés ayant un retard pur inférieur à une période d'échantillonnage.

De nombreux ouvrages tentent, de manière restrictive, d'apporter une solution à ce problème, notamment en proposant l'adaptation d'une commande échantillonnée ou hybride [HCCS02, Len89]. C'est dans cette optique que nous proposons des solutions utilisant les CCM.

5.1.3. Contraintes liées à l'*indisponibilité de l'état complet*

Les contraintes de retard et d'échantillonnage conduisent à des problèmes *d'observabilité* sur l'évolution du processus quand elles apparaissent sur la chaîne de retour. En effet, pour commander un processus, il est préférable de connaître son évolution *actuelle*. La meilleure observation sur cette évolution est donnée directement par l'état du processus. Ceci explique d'ailleurs la performance des architectures de commande par retour d'état comme, par exemple, les CFM.

Dans de nombreux cas, on ne dispose que de la sortie du processus pour réaliser son contrôle. Il faut alors reconstituer l'état afin de faire un retour d'état par une estimée de cette grandeur. Cette reconstitution, réalisée par un *observateur*, se base sur une modélisation du processus pour générer une estimation de l'état à partir des valeurs courantes de l'entrée et de la sortie du processus. Il est évident que, comme précédemment, les architectures de commande incluant un observateur dans la boucle de rétroaction souffrent d'une baisse de robustesse face aux perturbations. Bien souvent, l'observation repose sur la dualité estimateur d'état/contrôleur par retour d'état [Wol76, FH77, Won79, Kai80, DH81] et sur les principes de séparation visant à déterminer indépendamment les gains de l'estimateur et du contrôleur [Med69, Che70, BH75, SW77].

5.1.4. Combinaison des *contraintes*

Nous considérons, dans ce chapitre, les processus linéaires MIMO afin de formaliser une méthode de commande dans le cas où on ne dispose que de la sortie retardée et échantillonnée, délivrée par un capteur numérique tel que mentionné dans l'introduction générale. Il est évident que cette problématique combine les trois aspects contraignant une commande par retour d'état classique. En plus du contrôleur lui-même, chacun de ces aspects fait appel aux paramètres du processus. Dans ce sens, l'architecture résultante est obligatoirement moins robuste et plus coûteuse en temps de calcul.

5.2. Contexte de travail

Nous considérons que le processus à contrôler est défini par le METC d'ordre n suivant :

$$x'(t) = a.x(t) + b.u(t) \tag{5.1a}$$

$$y(t) = c.x(t)\,, \tag{5.1b}$$

avec les notations usuelles du présent mémoire.

Hypothèse 5.1. *Nous supposons que la paire* (a,c) *est observable et que la paire* (a,b) *est commandable.*

Par ailleurs, suivant le cas traité, nous considérons quatre types de retour décrivant l'évolution de la dynamique du processus :

$$\xi(t) = x(t-R)\,, \tag{5.1c}$$

$$\xi^*(t) = x^*(t-R)\,, \tag{5.1d}$$

$$z(t) = y(t-R)\,, \tag{5.1e}$$

$$z(t)^* = y^*(t-R)\,, \tag{5.1f}$$

avec *R* étant le retard sur l'information de retour et (*) représentant un échantillonnage de période t_e.

Hypothèse 5.2. *Dans le cas d'un retour échantillonné ((5.1d) et (5.1f)), nous faisons l'hypothèse que le retard est un multiple de la période d'échantillonnage :* $R = N.t_e$, *N étant entier positif.*

Remarque 5.1. *Le retour décrit par (5.1f) correspond à l'observation de la sortie du processus par un capteur numérique introduisant un retard et un échantillonnage. Dans le cas où le capteur est un système de vision, l'échantillonnage à la période* t_e *correspond à la délivrance des trames d'images,* t_e *représentant la durée d'une prise d'image. Quant au retard, il représente le temps de calcul nécessaire au traitement des images. Cela implique pour le système de vision, qu'il faut un certain nombre* N *d'images pour obtenir l'information requise.*

Remarque 5.2. *Dans tous les cas, nous utilisons une adaptation de l'architecture de commande par les CFM. Dans ce sens, nous reprenons le formalisme du chapitre 4 pour définir le SFM associé qui génère la commande à injecter au processus. Les stratégies proposées faisant souvent référence au CCM non optimisé, il est important de rappeler brièvement le contexte de son fonctionnement de base.*

L'équation d'un système bouclé par une structure CCM est donné par :

$$x_{k+1} = e^{a.T}.x_k + M.\lambda_k\,,\ avec\ M = e^{a.T}.\int_0^T e^{-a.\tau}.b.\gamma.e^{\alpha.\tau}\,d\tau\,. \tag{5.2}$$

$\lambda(t)$ *désigne l'état du CCM qui est régi par* $\lambda'(t) = \alpha.\lambda(t)$ *et* $w(t) = \gamma.\lambda(t)$ *dénote sa sortie qui est la commande à appliquer au processus :* $u(t) = w(t)$. *La condition initiale à appli-*

quer à cet état à chaque instant de commutation étant $\lambda_k^+ = \beta_d.\psi_k$*, l'enjeu est de déterminer pour le SLCM* $\Sigma_c(S,\alpha,\beta_c,\beta_d,\gamma)$ *du CCM non optimisé, les instants de commutation* S*, la matrice* β_d *et l'entrée* $\psi(t)$ *à discrétiser selon le cas considéré. À noter que dans le cas non optimisé, l'entrée continue n'est pas utilisée et la matrice* β_c *est considérée nulle (remarque 4.3).*

5.3. Retour par la sortie continue

Dans un premier temps, nous considérons que nous avons à disposition la sortie *continue* $y(t)$ du processus pour réaliser son contrôle.

5.3.1. Utilisation d'un observateur

Une première possibilité est d'inclure, dans la boucle de rétroaction, un observateur continu de type Luenberger [Lue64, Lue66, Lue71]. Ainsi, une estimation $\hat{x}(t)$ de l'état du processus est fournie au CFM à la place de son état réel et on est ramené au cas présenté au chapitre 4 (figure 5.1).

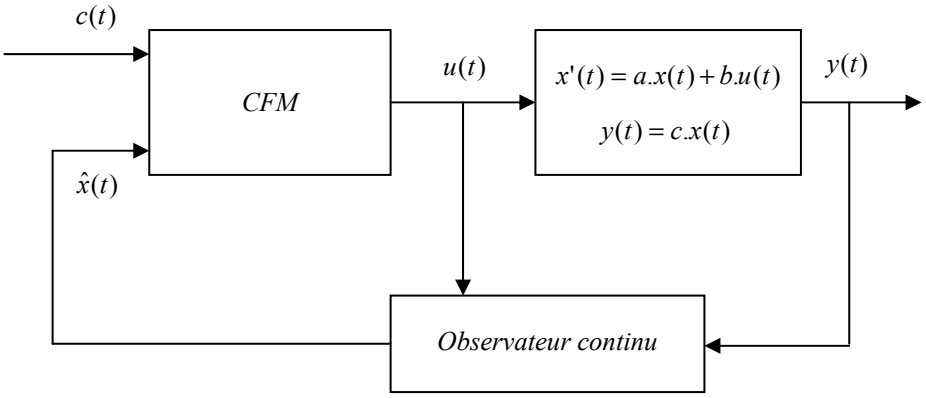


Fig. 5.1. *Utilisation d'un observateur dans la boucle de rétroaction*

L'asservissement par un retour d'état continu classique à partir d'un tel observateur requiert une détermination des gains selon le principe de séparation. Dans notre cas, cette démarche est inutile, car les CCM n'utilisent le retour d'état qu'aux instants de commutation, le processus étant en roue libre entre ces instants.

5.3.2. Adaptation du CCM

Dans sa forme de base, le formalisme du CCM fait intervenir le vecteur d'état pour établir la commande. Ainsi, le retour de l'ensemble des variables d'état offre une connaissance

suffisante de l'évolution du processus pour surmonter la contrainte d'un éventuel échantillonnage du retour.

Toutefois, dans le cas d'un retour continu, il est possible d'adapter la mise en équation d'un CCM en utilisant directement la sortie du processus, sans observateur et en se limitant à une consigne de sortie. Pour cela, nous proposons une mise en œuvre particulière du CCM pour lequel la période de commutation tend vers 0 ($T \to 0^+$).

5.3.2.1. Mise en équations

Considérons un processus tel que décrit dans (5.1) commandé par un CCM non optimisé basé sur $\Sigma_c(k.T, \alpha, 0, \beta_d, \gamma)$. Le système bouclé évoluant selon (5.2), nous pouvons écrire, en multipliant par *c* à gauche :

$$y_{k+1} = c.e^{a.T}.x_k + c.e^{a.T}.\int_0^T e^{-a.\tau}.b.\gamma.e^{\alpha.\tau} d\tau.\lambda_k \tag{5.3}$$

Cette mise en forme est semblable à celle du §4.3.1.4 « Adaptation à une consigne de sortie ». Les dimensions des paramètres du contrôleur sont donc :

- $\dim(\lambda(t)) = m$,
- $\dim(\alpha) = m \times m$,
- $\dim(\gamma) = r \times m$.

En imposant au CCM une commutation à période $T \to 0^+$, il est possible de remplacer, dans l'équation (5.3), les termes en exponentiel par un développement limité à l'ordre 1 :

$$y_{k+1} = c.(I_n + a.T).x_k + c.(I_n + a.T).\int_0^T (I_n - a.\tau).b.\gamma.(I_n + \alpha.T).d\tau.\lambda_k \tag{5.4}$$

Ce qui donne :

$$y_{k+1} = y_k + c.a.T.x_k + (c.b.\gamma.T + \varepsilon(T^2)).\lambda_k, \tag{5.5}$$

$\varepsilon(T^2)$ étant négligeable pour $T \to 0^+$.

Le but étant de calculer la condition initiale de l'état du CCM aux instants de commutation, nous imposons la condition de poursuite $y_{k+1} = (c_s)_k$, $c_s(t)$ étant la consigne de *sortie*, afin d'écrire pour le système bouclé :

$$(c.b.\gamma.T + \varepsilon(T^2)).\lambda_k = (c_s)_k - y_k - c.a.T.x_k \tag{5.6}$$

5.3.2.2. Paradoxe

Dans l'équation (5.6), la grandeur $(c.b.\gamma.T + \varepsilon(T^2))$ représente la matrice $(c.M)$ inversible, telle que définie au §4.3.1.4. De cette équation, on peut déduire :

$$\lambda_k - \lambda_k + (c.b.\gamma.T + \varepsilon(T^2)).\lambda_k = (c_s)_k - y_k - c.a.T.x_k \tag{5.7}$$

Soit :

$$\lambda_k = (I_m - c.b.\gamma.T - \varepsilon(T^2)).\lambda_k + (c_s)_k - y_k - c.a.T.x_k \tag{5.8}$$

Et, lorsque $T \to 0^+$:

$$\lambda_k = I_m^-.\lambda_k + (c_s)_k - y_k \tag{5.9}$$

Interprétation algorithmique :

D'un point de vue algorithmique, l'équation (5.9) traduit une évaluation récursive de λ_k, qui correspond (en informatique) à une opération de réaffectation, à chaque pas de calcul du simulateur[1], soit :

$$\lambda_k \leftarrow I_m^-.\lambda_k + (c_s)_k - y_k \tag{5.10}$$

La relation (5.10) se traduit par un schéma fonctionnel bouclé tel que celui de la figure 5.2.

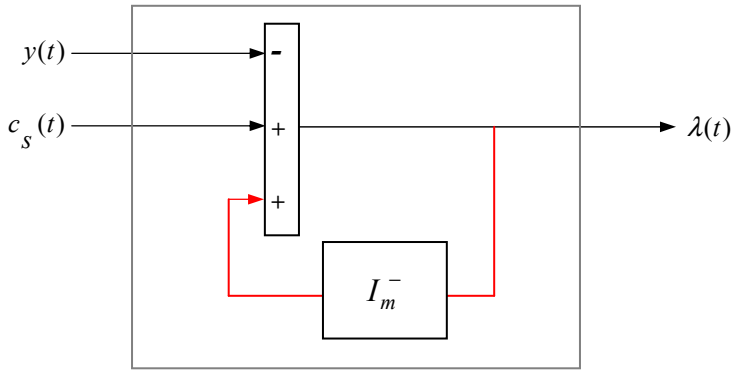


Fig. 5.2. *Bouclage récursif*

[1] En simulation, Matlab/Simulink peut admettre un pas de calcul de l'ordre du dixième de nanoseconde sur un PC récent. Sur une architecture temps réel, le pas de calcul peut être de l'ordre du dixième de milliseconde avec la technologie dSpace/Matlab/Simulink.

5.3.2.3. Architecture

Dans le formalisme des CCM, il convient de déterminer selon la mise en équation, la matrice β_d et l'*entrée à discrétiser* du SLCM. Ce paramétrage consiste à imposer la condition initiale λ_k de l'état du SLCM à chaque instant de commutation. Toutefois, la démarche est différente dans ce cas car nous obtenons directement λ_k par le montage de la figure 5.2. Il suffit alors d'adopter ce signal comme entrée *à discrétiser* du SLCM et de choisir $\beta_d = I_m$, la période de commutation T étant égale au pas de calcul du simulateur.

Il est donc possible de modéliser le contrôleur comme illustré figure 5.3.

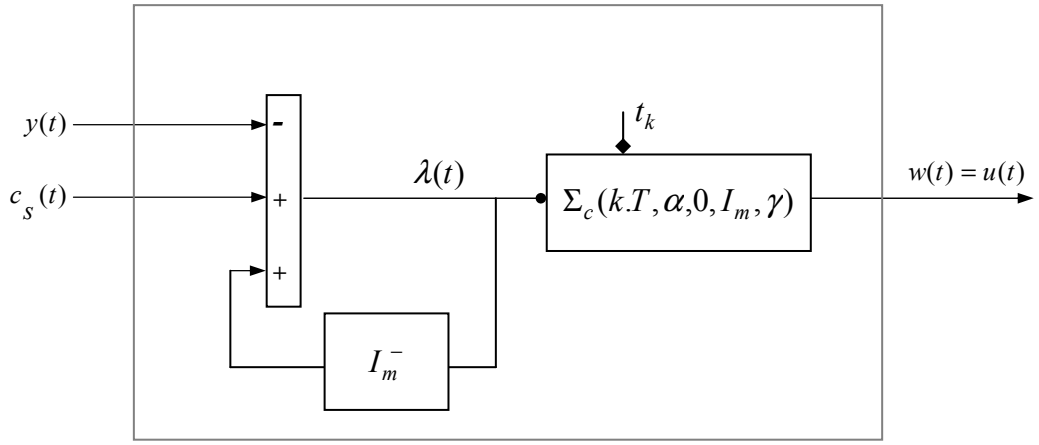


Fig. 5.3. *Contrôleur « paradoxal » à retour de sortie*

De plus, quand $T \to 0^+$, l'état du contrôleur n'évolue pas de sa condition initiale sur un morceau *k* infiniment petit. Cela conduit à une commande constante par morceaux infiniment petits. Dans ce sens, le SLCM $\Sigma_c(k.T, \alpha, 0, I_m, \gamma)$ peut être remplacé par un BOZ cadencé à $T \to 0^+$ suivi d'un gain γ. Par ailleurs, T étant égale au pas de calcul du simulateur, le BOZ ne présente aucun intérêt. Le schéma fonctionnel du contrôleur devient alors encore plus simple, comme l'illustre figure 5.4.

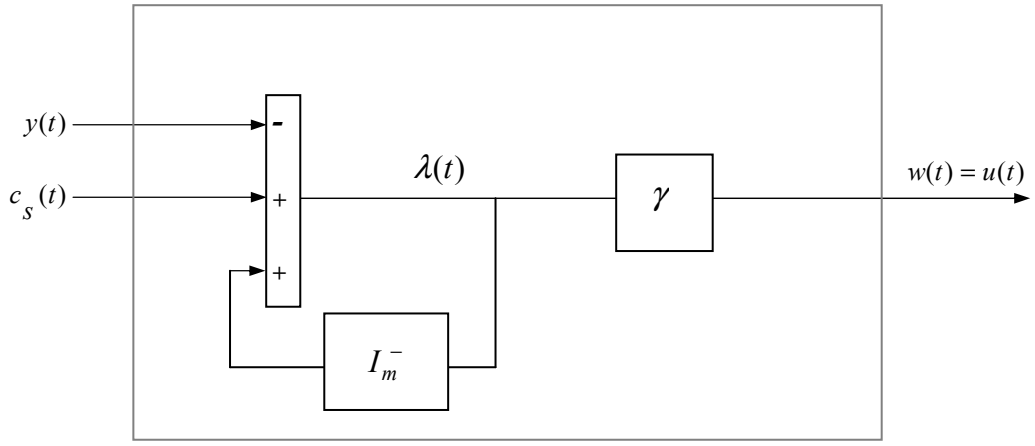


Fig. 5.4. *Contrôleur « paradoxal » à retour de sortie (schéma simplifié)*

5.3.2.4. Synthèse

Cette mise en équation garantit, par un retour de sortie continu, l'asservissement d'un processus *sans* connaissance a priori de ses paramètres. La méthode aboutit à un suivi de trajectoire d'une consigne de *sortie* avec un retard de seulement un pas de calcul.

À noter qu'il est également possible d'aboutir à une simplification semblable du CCM dans le cas d'un retour d'*état* continu en conservant $T \to 0^+$.

Le principe est semblable à celui de la commande à haut gain.

5.3.3. Validation expérimentale

Nous proposons, dans cette section, la comparaison des deux méthodes et des exemples illustrant la performance de l'approche paradoxale. L'implantation de ces méthodes basées sur les CFM est semblable à celle donnée au chapitre 4. En particulier, l'*amortisseur* du §4.3.3.1 s'applique ici. Toutefois, dans le cas du contrôleur paradoxal, il est possible de définir plus simplement (paragraphe suivant) un type d'amortissement équivalent pour atténuer la commande en cas de discontinuité.

5.3.3.1. Contrôleur paradoxal avec amortisseur

Remarque 5.3. *Retour sur l'équation (5.9) :*

$$\lambda_k = I_m^-.\lambda_k + (c_s)_k - y_k$$

La solution en λ_k de cette équation nécessite que $c_s(t)$ soit continue. En effet, dans ce cas, $y(t)$ étant continue, il peut exister une solution $\lambda(t)$ finie. Par contre, si $c_s(t)$ présente des discontinuités, la seule solution possible est : $\lambda(t) = \infty$. En pratique, pour éviter ces situations, nous avons intégré, au sein du contrôleur, un « amortisseur » semblable à celui du §4.3.3.1 qui limite la commande aux discontinuités, en apportant des transitoires dans la poursuite.

5.3.3.1.1. Stratégie

Considérons l'équation du contrôleur paradoxal :

$$\lambda_k = \varsigma.I_m^-.\lambda_k + (c_s)_k - y_k,\ \varsigma \in \Re \mid 0 \le \varsigma < 1$$

Le contrôleur garantit un asservissement robuste si $\varsigma = 1$. Toutefois, nous constatons en pratique que pour si ς décroît vers zéro, l'asservissement devient plus *mou* (moins réactif car équivalent à un asservissement proportionnel (P) à gain réduit). L'idée est donc de

définir ς comme un coefficient d'amortissement fonction de l'écart entre la sortie du processus et sa consigne, comme au §4.3.3.1, soit :

$$\varsigma(.) = \frac{1}{1+\varepsilon}\left\{\varepsilon + e^{-\frac{\theta}{2}(\Delta^T.\Delta)}\right\}, \text{ avec } \Delta = c_s(t-T) - y(t),$$

ε et θ étant des entiers positifs et T représentant un pas de calcul du simulateur.

5.3.3.1.2. Architecture

Ainsi, nous pouvons proposer (figure 5.5) une structure du contrôleur paradoxal qui admet une consigne présentant des discontinuités.

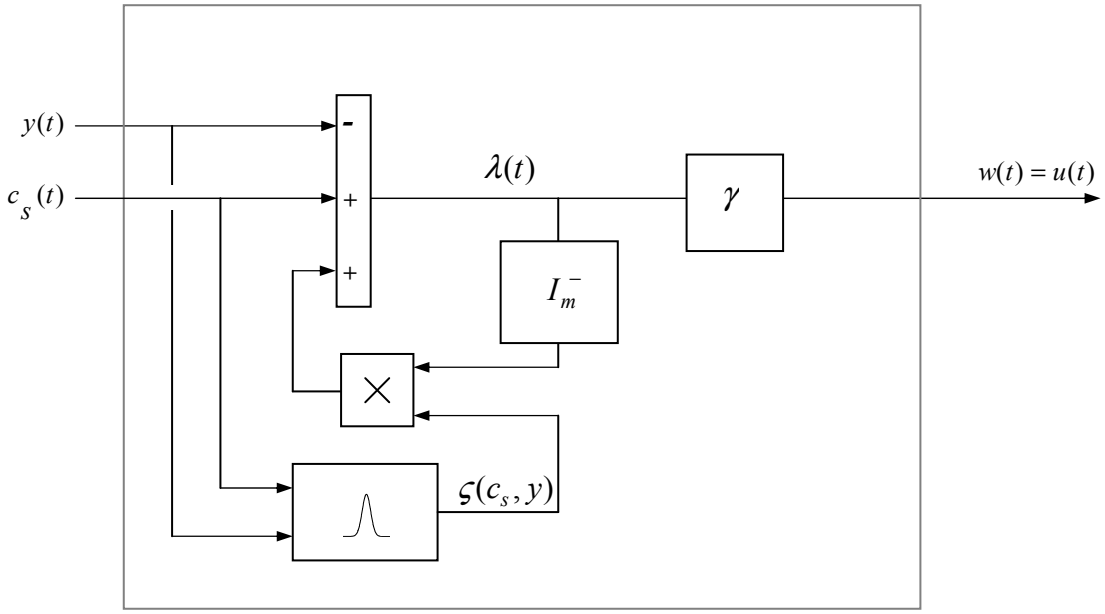


Fig. 5.5. *Contrôleur paradoxal avec amortisseur intégré*

5.3.3.2. Exemple de comparaison

Afin de comparer la performance des deux approches, nous considérons un processus du second ordre dont certains paramètres présentent de fortes variations autour d'une valeur moyenne :

$$a(t)=\begin{bmatrix} -5(1+2\sin(5t)) & 1 \\ 0 & -10(1+2.1\sin(4t)) \end{bmatrix},\ b(t)=\begin{bmatrix} 6(1+0.5\sin(2t)) \\ 25(1+1.5\sin(3t)) \end{bmatrix} \text{ et } c=\begin{bmatrix} 1 & 0 \end{bmatrix},$$

$$x_0=\begin{bmatrix} 0 \\ 2 \end{bmatrix}.$$

Dans le but d'imposer une consigne de sortie, $c_s(t)=5\sin(3t)$, nous proposons deux types d'asservissement :

- une structure (telle qu'en figure 5.1) combinant un observateur Luenberger d'ordre réduit (pour estimer le deuxième vecteur d'état) avec une condition initiale $(\hat{x}_2)_0=20$ et un CCM non optimisé basé sur $\Sigma_c(k.T,\alpha,0,M^{-1},\gamma)$ avec :

  $$T=2\text{ms},\ \alpha=0,\ \gamma=300,$$

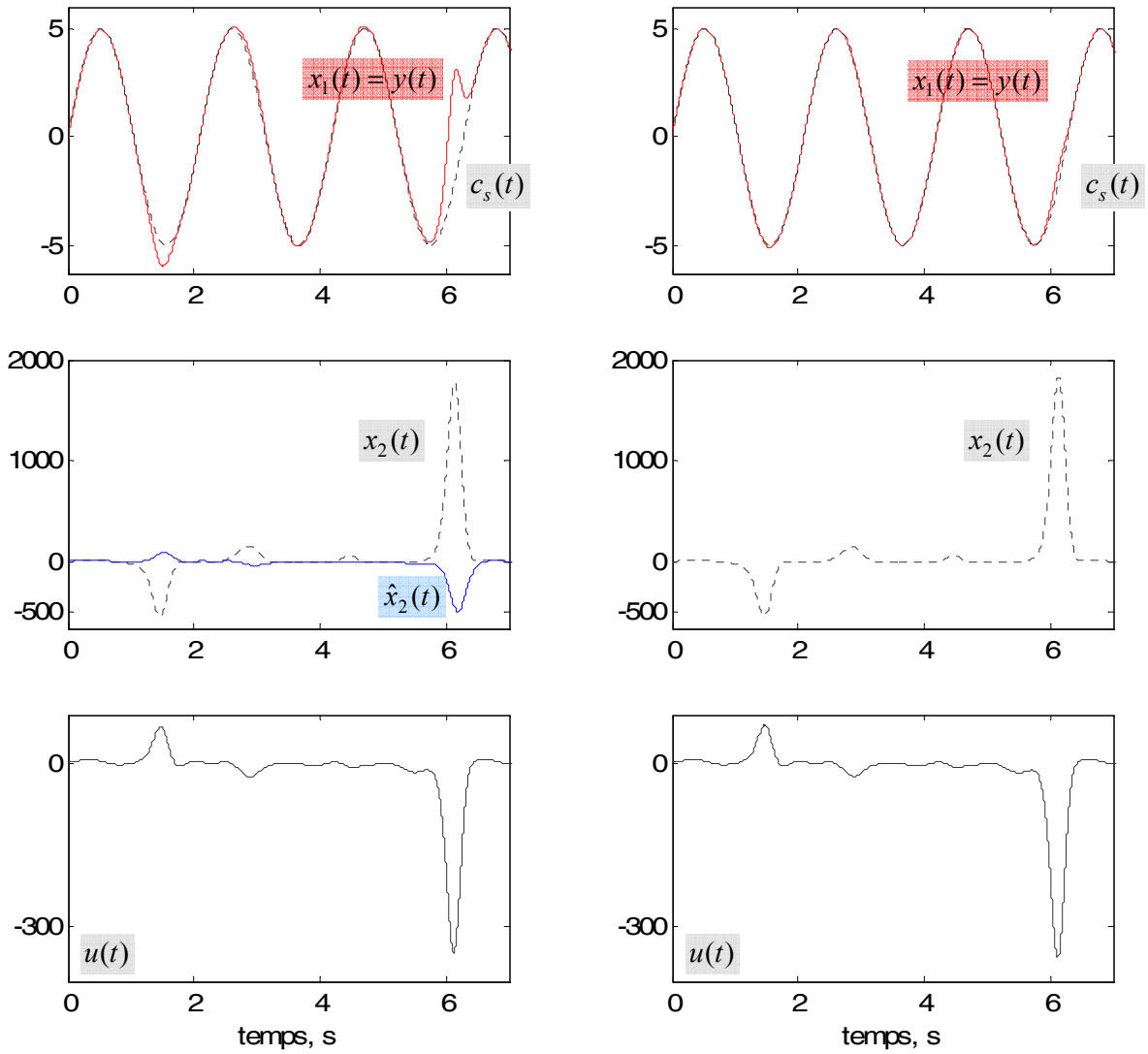
  les paramètres de l'observateur et du CCM étant approximés à la valeur moyenne des paramètres du processus,

- un contrôleur paradoxal avec un pas de calcul de 0.1ms et avec les paramètres suivants : $I_m^-=0.998$ et $\gamma=1$.

Les résultats sont illustrés figure 5.6. Pour chaque type d'asservissement, nous donnons l'évolution de la sortie du processus suivant sa consigne, ainsi que celle de la deuxième composante d'état et de la commande. Sur la figure 5.6a, l'estimation de la deuxième composante par l'observateur est également illustrée.

Nous constatons que la variation des paramètres influe très légèrement sur la poursuite dans le cas du CCM simplifié. En revanche, le calcul des paramètres de l'observateur et du CCM étant basé sur des valeurs constantes des paramètres du processus, l'estimation de la deuxième composante d'état est erronée et la sortie est *décrochée* de sa consigne lorsque l'écart entre la valeur actuel des paramètres du processus et la valeur moyenne correspondante est important.

Il est évident que la méthode basée sur le CCM simplifié pour $T\to 0^+$ conduit à un asservissement plus robuste face aux variations des paramètres, pour la simple raison que ceux-ci n'interviennent pas dans le paramétrage.



(a). *Observateur / CCM à T = 2ms* ***(b).*** *CCM simplifié ($T \to 0$)*

Fig. 5.6. *Commande par retour de sortie continue*

5.3.3.3. Performances du contrôleur paradoxal

5.3.3.3.1. Processus non linéaire

Considérons un processus non linéaire à paramètres variants défini par :

$$a(t) = \begin{bmatrix} 0 & 1 \\ 0 & -1/\tau(t) \end{bmatrix},\ b(u,t) = \begin{bmatrix} 0 \\ K(u)/\tau(t) \end{bmatrix},\ c = \begin{bmatrix} 1 & 0 \end{bmatrix} \text{ et } x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

avec $\tau(t) = 5.10^{-4}(10 + \sin(5t))$ et $K(u) = \text{sat}_{0.005}^{4.500}(|u|)$, $\text{sat}_{\min}^{\max}()$ étant une fonction de saturation avec les bornes spécifiés et $u(t)$ étant la commande appliquée au processus.

Afin d'imposer une consigne de sortie définie par $c_s(t) = 5\sin(3t)$, nous utilisons un contrôleur paradoxal.

Les résultats sont donnés en figure 5.7. Ils montrent que la méthode est robuste face à des non linéarités apparaissant dans le comportement du processus qui plus est à paramètres variants.

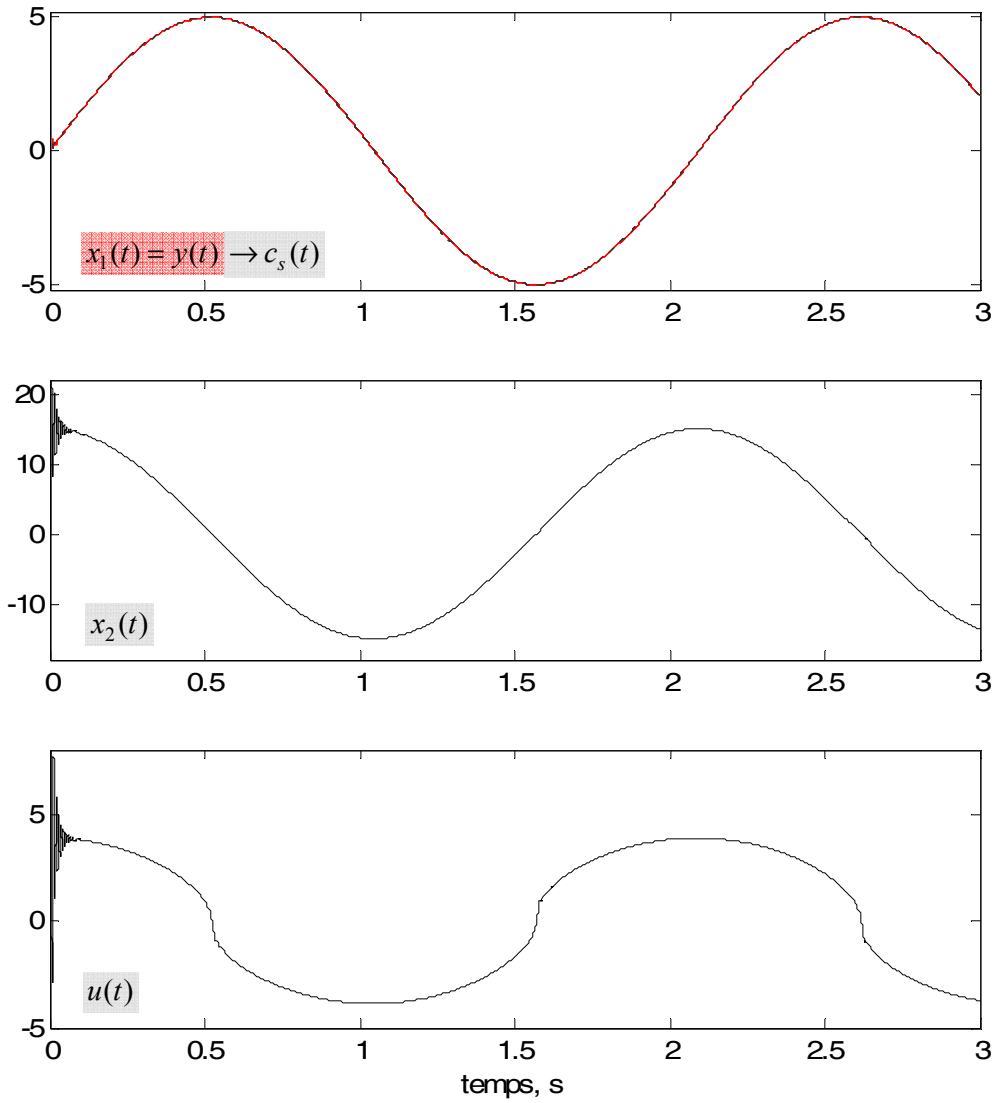


Fig. 5.7. *Asservissement d'un processus non linéaire par le contrôleur paradoxal*

5.3.3.3.2. Consigne présentant des discontinuités

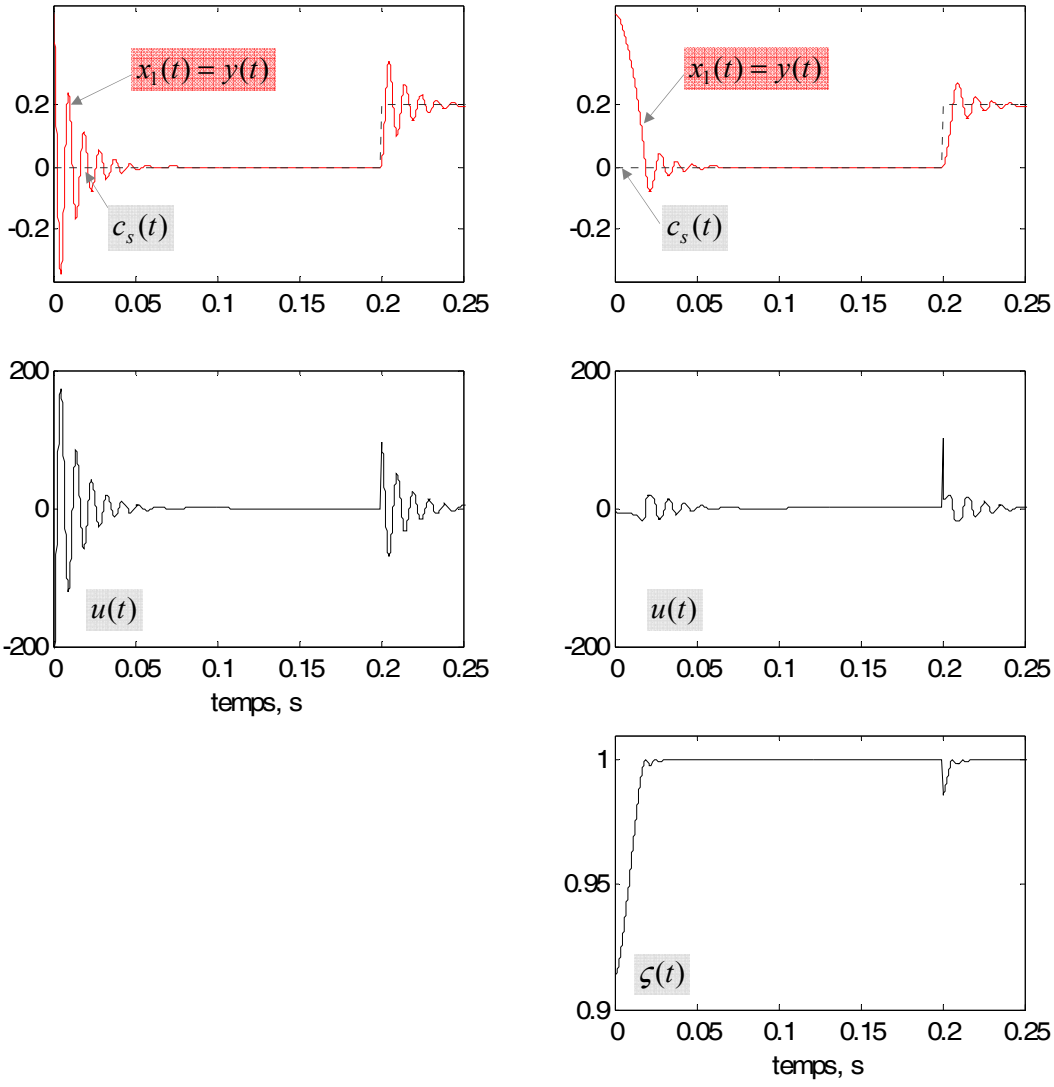
Nous souhaitons imposer au processus non linéaire du §5.3.3.3.1 une consigne de sortie $c_s(t)$ en échelon (d'amplitude 0.2 et déclenché à 0.2s). Pour ce faire, nous proposons de réaliser deux types d'asservissement :

- utilisation du contrôleur paradoxal sans amortisseur,
- utilisation du contrôleur paradoxal avec un amortisseur dont les paramètres sont : $\theta = 0.8$ et $\varepsilon = 0.1$.

Les résultats sont illustrés figure 5.8. À noter que dans le cas présent, nous avons fixé la condition initiale du processus à :

$$x_0 = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$$

Nous constatons que la poursuite est assurée même dans le cas d'un processus non linéaire. Par ailleurs, la figure 5.8b montre que l'amortisseur atténue les dépassements aux discontinuités sur la consigne. Nous pouvons constater que le coefficient d'amortissement chute à chaque instant de discontinuité. Le paramétrage de l'amortisseur permet de trouver un compromis entre le temps de réponse et l'amplitude des dépassements.



(a). *Sans amortisseur* ***(b).*** *Avec amortisseur*

Fig. 5.8. *Consigne présentant des discontinuités*

5.3.3.3.3. Processus réel

Les performances temps réel du contrôleur paradoxal ont été testées sur la maquette décrite en annexe I. Les paramètres étant inutiles pour réaliser le contrôleur, nous définissons le processus réel par :

$$x(t) = \begin{bmatrix} position \\ vitesse \end{bmatrix} \text{ et } x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

En considérant que le processus admet une sortie par la matrice $c = [1 \quad 0]$, nous définissons une consigne de position donnée par $c_s(t) = 0.03\sin(2t)$, enclenché à l'instant t_0.

L'asservissement est assuré par un contrôleur paradoxal intégrant un amortisseur caractérisé par $\theta = 5000$ et $\varepsilon = 0.05$ et les résultats sont représentés sur la figure 5.9.

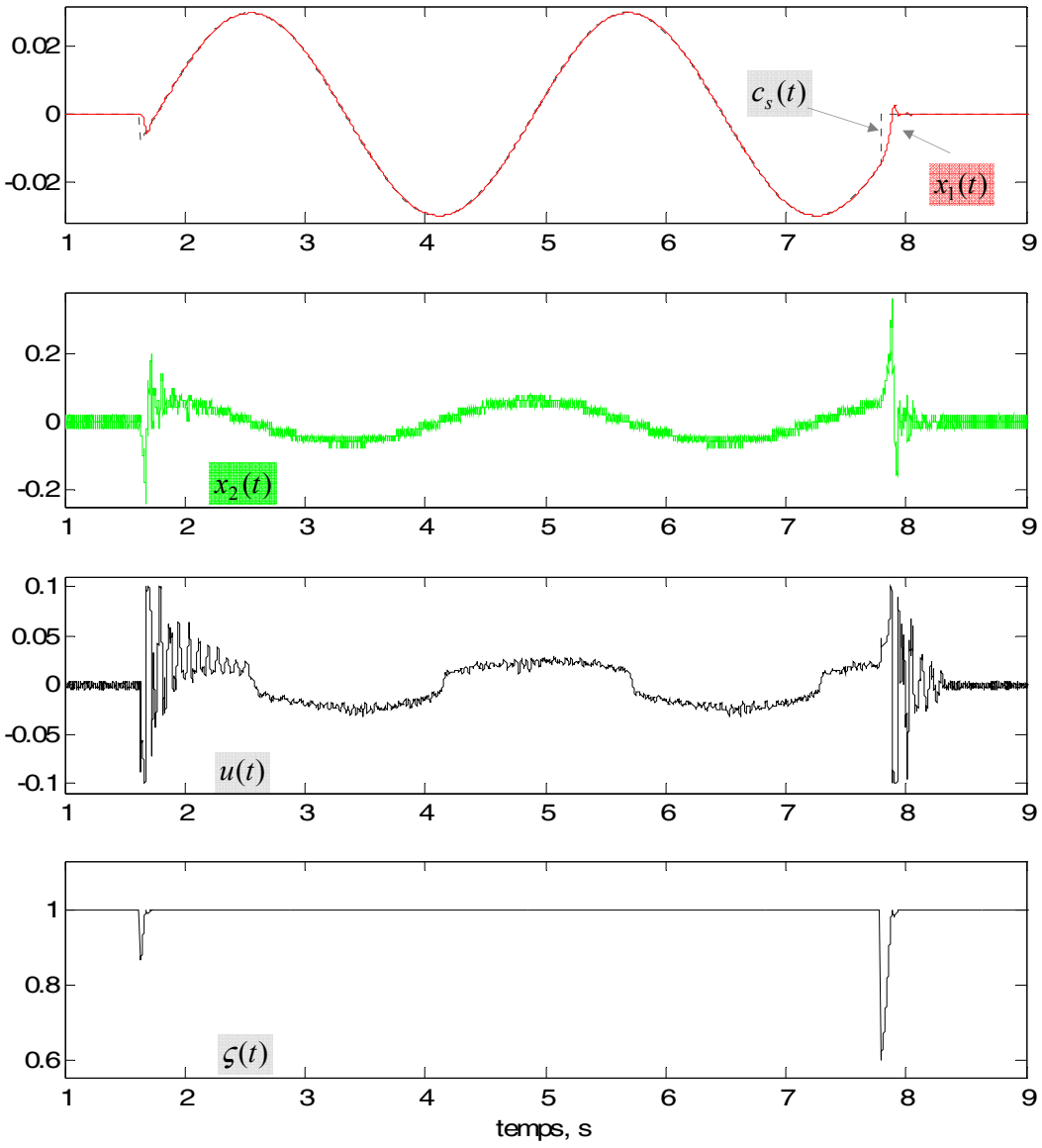


Fig. 5.9. *Asservissement d'un processus réel par un contrôleur paradoxal*

Nous constatons que la poursuite de la consigne de sortie par la sortie du processus réel est garantie. De plus, l'amortisseur permet d'atténuer les dépassements, notamment sur la l'entrée $u(t)$ et sur la vitesse $x_2(t)$ en modérant le coefficient d'amortissement $\varsigma(t)$ aux discontinuités.

Il est intéressant de noter que la forme de la commande est semblable à celle du §5.3.3.3.1 qui traite un processus non linéaire. En effet, comme spécifié au §4.3.3.2.6, le processus réel considéré présente des non linéarités liés aux frottement secs. Le système décrit au §5.3.3.3.1 peut être considéré comme un bon modèle du processus réel.

À noter que l'évolution de la vitesse $x_2(t)$, donnée à titre indicatif, provient d'un capteur fortement perturbé.

5.3.4. Conclusion

L'asservissement par retour de sortie est un cas très fréquemment rencontré dans la commande de processus réels, d'où l'intérêt d'adapter le formalisme des CFM.

Nous avons vu qu'il est possible d'inclure, dans la boucle de rétroaction, un observateur permettant de substituer une estimation de l'état dans une structure CFM classique. À noter qu'il est également envisageable, dans le cas d'un retour *échantillonné* de la sortie, d'utiliser un observateur discret conduisant à une mise en œuvre semblable. Toutefois, comme le montre les exemples de simulation, l'utilisation de l'observateur réduit la robustesse d'un CFM. Ceci s'explique par le fait que les paramètres du processus sont sollicités à deux reprises dans la chaîne de retour : par conséquent, les imperfections de modélisation/identification ont plus d'impact.

Par ailleurs, nous proposons un contrôleur paradoxal dans le cas d'un retour continu. L'atout principal réside dans le fait que ce contrôleur ne requiert pas les paramètres du processus à commander, excluant la nécessité d'une connaissance *a priori* du modèle du processus. De ce fait, la méthode présente une robustesse remarquable face aux changements des paramètres du processus. La méthode présente une simplicité de mise en œuvre et un coût réduit de calcul en ligne.

5.4. Retour par l'état continu retardé

Nous considérons, ici, que nous avons à disposition l'état du processus sous forme retardée et continue, soit $\xi(t) = x(t-R)$.

Hypothèse 5.3. *Nous supposons dans ce paragraphe que le retard R est une constante connue.*

5.4.1. Prédiction

L'idée est d'estimer, à partir du signal de retour $\xi(t) = x(t-R)$, l'état actuel du processus. Pour ce faire, nous partons de l'équation d'état du processus qui permet d'écrire :

$$\hat{x}(t) = F.x(t-R) + \int_{t-R}^{t} e^{a.(t-\tau)}.b.u(\tau).d\tau, \tag{5.11a}$$

avec $$F = e^{a.R} \tag{5.11b}$$

Le calcul de l'intégrale peut se décomposer en :

$$\int_{t-R}^{t} e^{a.(t-\tau)}.b.u(\tau).d\tau = \underbrace{\int_{0}^{t} e^{a.(t-\tau)}.b.u(\tau).d\tau}_{\mathrm{I}_1} - \underbrace{\int_{0}^{t-R} e^{a.(t-\tau)}.b.u(\tau).d\tau}_{\mathrm{I}_2} \tag{5.12}$$

Le terme I_1 est interprété comme l'état d'un système ayant pour paramètres d'état *a* et *b*, évoluant à partir d'une condition initiale nulle. Si $V(t)$ représente l'état d'un tel système, nous considérons que I_1 est solution de :

$$\begin{cases} V'(t) = a.V(t) + b.u(t) \\ V(t_0) = 0 \end{cases} \tag{5.13}$$

Pour calculer I_2, nous faisons le changement de variable : $\theta = \tau + R$. Cela permet d'écrire :

$$\mathrm{I}_2 = \underbrace{-\int_{0}^{R} e^{a.(t-\theta+R)}.b.u(\theta-R).d\theta}_{\mathrm{I}_3} + \underbrace{\int_{0}^{t} e^{a.(t-\theta+R)}.b.u(\theta-R).d\theta}_{\mathrm{I}_4} \tag{5.14}$$

Toutefois, en considérant que la commande est nulle pour $t < 0$, le terme I_3 s'annule. Par ailleurs, nous effectuons le calcul de I_4 de la même façon que celui de I_1, en écrivant d'abord :

$$\mathrm{I}_4 = F.\underbrace{\int_0^t e^{a.(t-\tau)}.b.u(\tau - R).d\tau}_{\mathrm{I}_5} \tag{5.15}$$

Ensuite, nous considérons que le terme I_5 est solution de l'équation différentielle suivante, avec $W(t)$ vecteur d'état du système à paramètres d'état *a* et *b* :

$$\begin{cases} W'(t) = a.W(t) + b.u(t-R) \\ W(t_0) = 0 \end{cases} \tag{5.16}$$

Sachant que $\int_{t-R}^{t} e^{a.(t-\tau)}.b.u(\tau).d\tau = \mathrm{I}_1 - \mathrm{I}_4$, nous aboutissons alors à l'équation de prédiction donnée par :

$$\hat{x}(t) = V(t) + F.(\xi(t) - W(t)) \tag{5.17}$$

Remarque 5.4. *En pratique, les grandeurs $V(t)$ et $W(t)$ sont générées à partir de deux SLCM distincts définis par $\Sigma_c(t_0, a, b, 0, I_n)$ et avec une entrée à discrétiser nulle. Par contre, le SLCM relatif au calcul de $V(t)$ a pour entrée continue la commande $u(t)$ à appliquer au processus et celui relatif au calcul de $W(t)$ a pour entrée continue cette même commande, mais retardée de R.*

Remarque 5.5. *La solution proposée n'est réalisable que si les équations (5.13) et (5.16) sont stables, c'est-à-dire si le système à commander (matrice a) est lui-même stable.*

5.4.2. Commande

Le prédicteur fournissant l'état actuel, nous sommes ramenés au cas décrit au chapitre 4 en faisant un retour d'état. Il est donc possible d'envisager les contrôleurs CFM de base.

5.4.3. Architecture

Nous proposons, figure 5.10, une structure combinant le prédicteur d'état et un CCM non optimisé basé sur $\Sigma_c(k.T,\alpha,0,M^{-1},\gamma)$, sous réserve que M soit inversible, avec :

$$M = e^{a.T}.\int_0^T e^{-a.\tau}.b.\gamma.e^{\alpha.\tau}d\tau$$

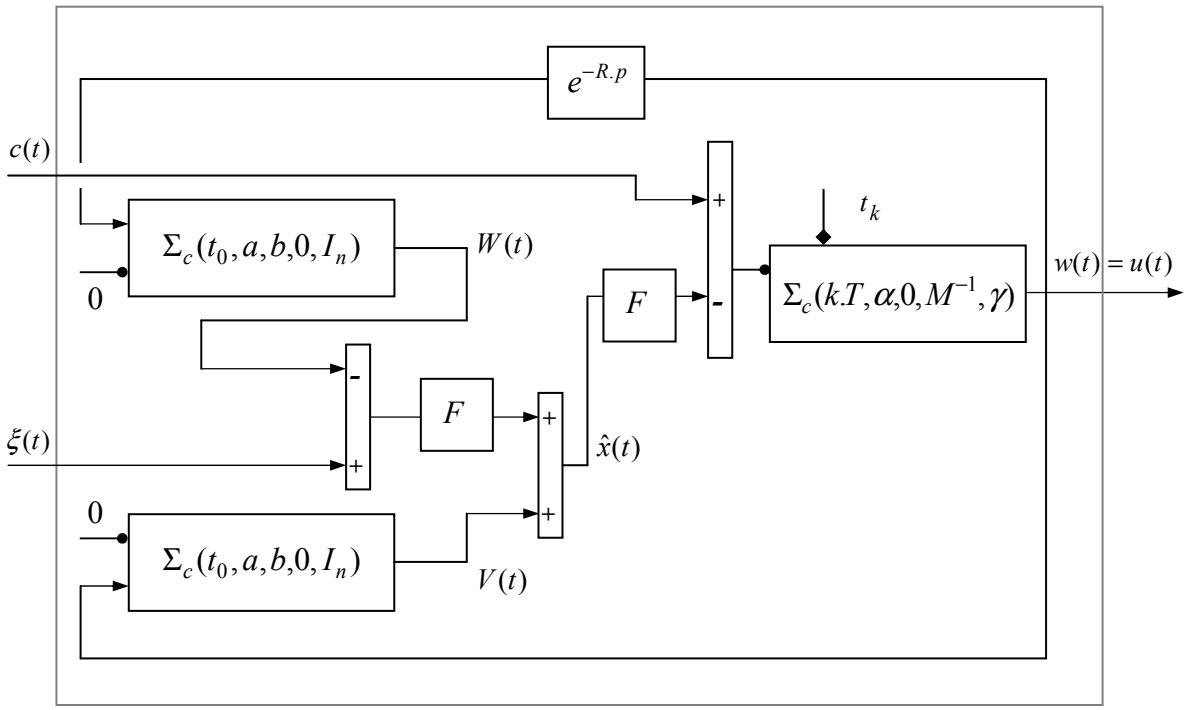


Fig. 5.10. *Architecture d'un contrôleur CCM intégrant un prédicteur d'état*

Sur cette figure, $e^{-R.p}$ représente un retard égal à R et $c(t)$ représente la consigne d'état.

Par ailleurs, les paramètres du CCM sont tels que définis au chapitre 4 dans sa forme de base.

Remarque 5.6. *À noter que la période de commutation T du CFM peut être quelconque car l'information de retour reconstituée est continue. Dans ce sens, il est possible d'utiliser le contrôleur paradoxal présenté dans la section précédente.*

5.4.4. Validation expérimentale

Considérons un processus linéaire défini par :

$$a=\begin{bmatrix}-3 & 1\\ 2 & -1\end{bmatrix},\ b=\begin{bmatrix}1\\ 2\end{bmatrix},\ c=\begin{bmatrix}1 & 0\end{bmatrix} \text{ et } x_0=\begin{bmatrix}0\\ 0\end{bmatrix}$$

dont l'état est disponible avec un retard R.

Afin d'asservir ce processus par un retour d'état, nous mettons en œuvre une architecture incluant un prédicteur (tel que donnée sur la figure 5.10).

Par ailleurs, dans le but de lui imposer une consigne de *sortie*, nous utilisons un CCM basé sur $\Sigma_c(k.T,\alpha,0,M^{-1},\gamma)$ adapté à cet usage (§4.3.1.4) avec les paramètres suivants :

$$T=1\text{ms},\ \alpha=0,\ \gamma=30,$$

À noter que nous avons appliqué une perturbation à partir de $t=4\text{s}$ de sorte que les paramètres *a* et *b* du processus sont alors :

$$a(t)=\begin{bmatrix}-3(1+0.4\sin(5t)) & 1\\ 2(1+0.3\sin(4t)) & -(1+0.1\sin(4t))\end{bmatrix},\ b(t)=\begin{bmatrix}1+0.1\sin(2t)\\ 2(1+0.2\sin(3t))\end{bmatrix}$$

Les résultats sont donnés figure 5.11a pour $R=1\text{s}$ et figure 5.11b pour $R=0.05\text{s}$.

Nous constatons que quand le processus est invariant ($t<4\text{s}$), le prédicteur estime parfaitement l'état du processus et la poursuite de $c_s(t)$ par $\xi_1(t)$ est garantie avec un retard de $R+T$.

Toutefois, pour $R=1\text{s}$, nous constatons qu' à partir de $t=4\text{s}$ l'estimation de l'état est erroné car le prédicteur se base sur les valeurs constantes (moyennes) des paramètres changeants pendant une période d'une seconde. Par conséquent, la poursuite n'est plus garantie rigoureusement.

La performance est meilleure pour $R=0.05\text{s}$ car le prédicteur n'a pas le temps de s'écarter de trop de la valeur réelle de l'état pendant la durée du retard. Nous voyons en figure 5.11b que la commande s'*arrange* pour surmonter la perturbation afin de maintenir la poursuite.

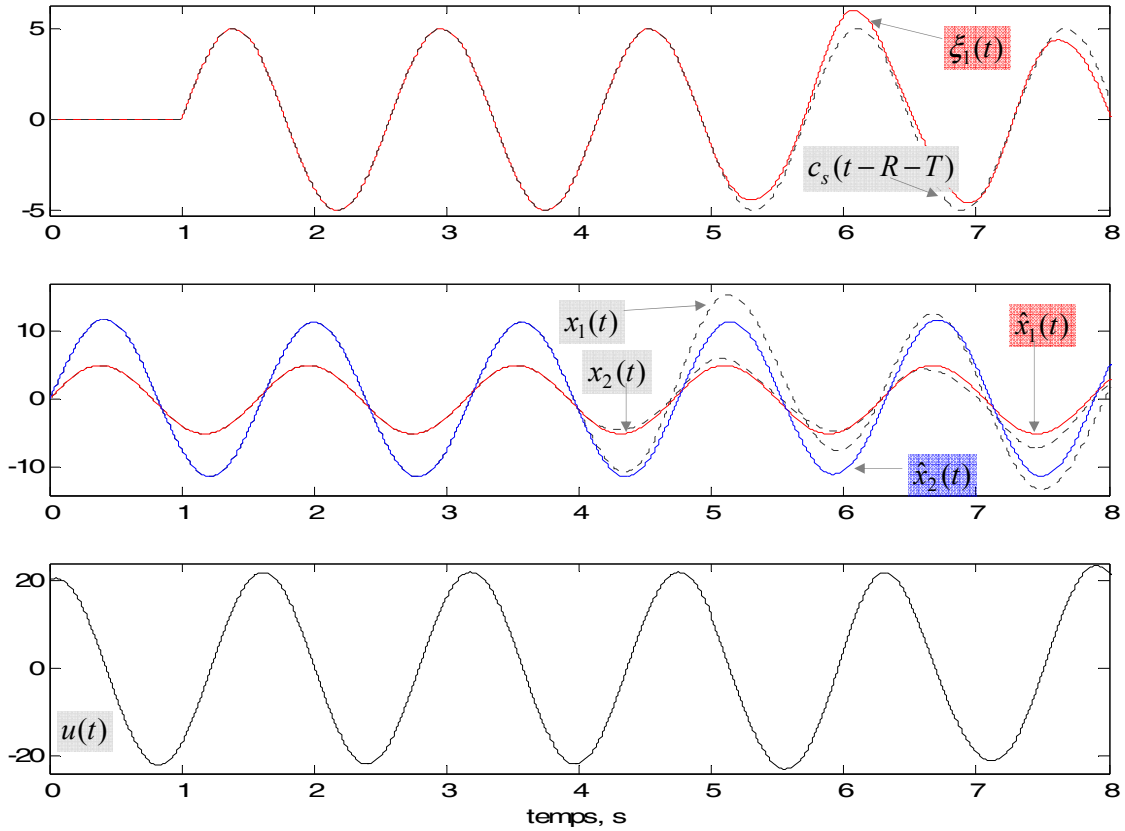


Fig. 5.11a. *Asservissement d'un processus par l'état retardé (R* = 1s*)*

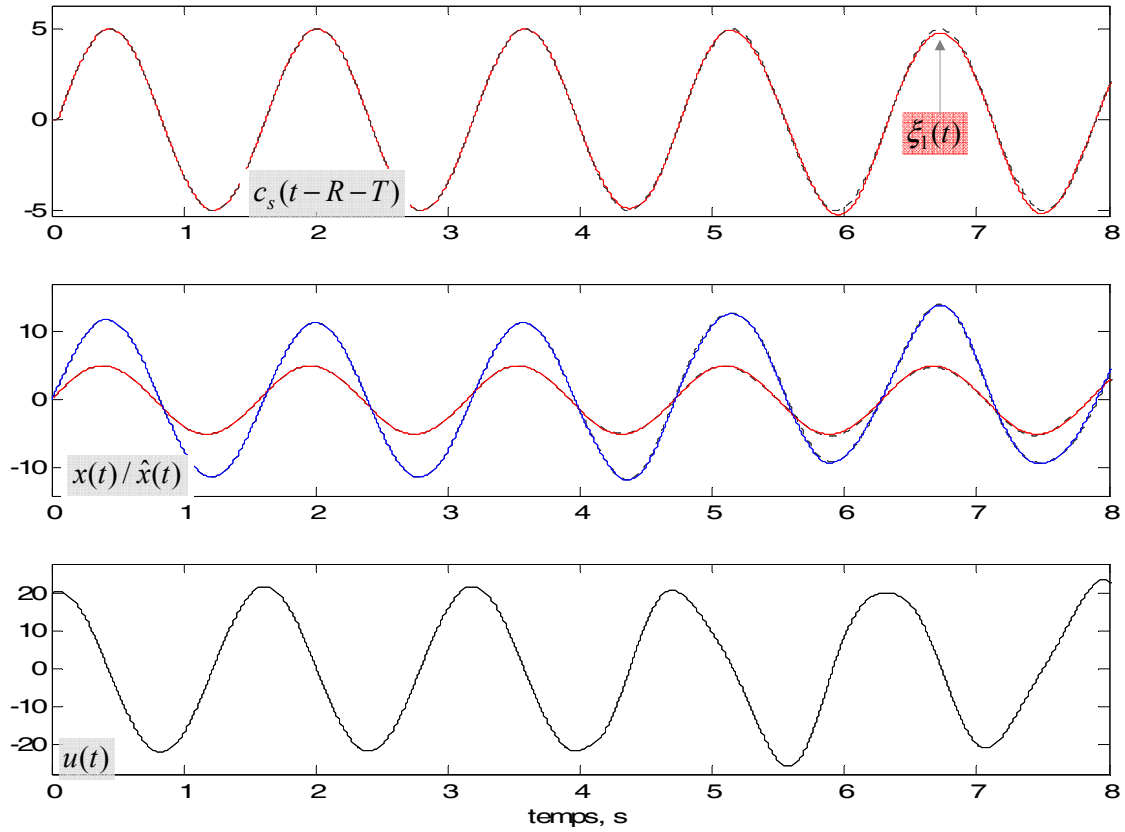


Fig. 5.11b. *Asservissement d'un processus par l'état retardé (R* = 0.05s*)*

5.4.5. Conclusion

Le retour par l'état retardé continu peut survenir dans une structure contenant, par exemple, un capteur analogique éloigné de la grandeur à mesurer. La méthode proposée réalise une prédiction de l'état actuel à partir du retour. Ainsi, nous pouvons utiliser un CFM classique avec l'état estimé. La prédiction et le contrôleur se basent tous deux sur les paramètres du processus. De ce fait, comme le montre l'exemple, la robustesse de l'architecture de commande globale décroît avec l'augmentation du retard sur le retour.

5.5. Retour par l'état retardé et échantillonné

Nous considérons, dans ce cas, que nous avons à disposition, la grandeur $\xi^*(t)$ correspondant à l'état du processus délivré sous forme retardée (de $R = N.t_e$) et échantillonnée (à t_e).

Hypothèse 5.4. *Nous supposons dans ce paragraphe que le retard R et la période d'échantillonnage sont des constantes connues.*

5.5.1. Adaptation d'un CCM commutant à *R*

L'idée est d'effectuer une prédiction de l'état actuel à partir de l'information retardée, en considérant des intervalles de temps de longueur égale au retard *R*, nécessitant une commutation du contrôleur à période égale au retard *R*. Cela signifie que l'on ne se préoccupe pas de l'échantillonnage à t_e du signal de retour. Ainsi, lors de l'écriture des équations, nous comprenons que le morceau *k* est de durée *R*. Le retard sur l'état étant *R*, l'information la plus récente sur l'évolution du processus est notée $\xi_k^* = x_{k-1}$.

5.5.1.1. Mise en équations

En considérant l'utilisation d'un CCM non optimisé basé sur $\Sigma_c(S, \alpha, \beta_c, \beta_d, \gamma)$, l'objectif est de définir convenablement les paramètres internes et les entrées du SLCM. Nous choisissons, au préalable :

- $S = \{k.R, k = 0,1,2...\}$, commutant ainsi le contrôleur à une période égale au retard,
- $\varphi(t) = 0$, $\beta_c = 0$ car le CCM ne possède pas d'entrée continue dans le cas non optimisé,
- $\alpha \in \Re^{n \times n}$ tel que l'évolution du SLCM soit stable sur un morceau *k*,
- $\gamma \in \Re^{r \times n}$ de rang plein,

Selon la démarche du chapitre 4, il est question de déterminer la matrice β_d et l'entrée $\psi(t)$ à discrétiser du SLCM. Toutefois, ne pouvant faire apparaître le terme x_k qui est indisponible à l'instant de calcul de la condition initiale de l'état du SLCM., nous proposons de réaliser une prédiction de l'état actuel à partir de l'information de retour. Dans ce sens, nous considérons l'équation de fonctionnement du système bouclé qui est régie par :

$$x_{k+1} = F.x_k + M.\lambda_k , \tag{5.18}$$

avec $F = e^{a.R}$ et, selon les notations habituelles : $M = F.\int_0^R e^{-a.\tau}.b.\gamma.e^{\alpha.\tau} d\tau$.

5.5.1.2. Prédiction

À partir de cette équation, on peut effectuer la prédiction à partir de l'état retardé $\xi_k^* = x_{k-1}$, selon l'expression suivante :

$$\hat{x}_k = F.\xi_k^* + M.\lambda_{k-1} \tag{5.19}$$

5.5.1.3. Commande

Ainsi, en imposant la condition de poursuite $x_{k+1} = c_k$ ($c(t)$ étant la consigne d'état) et en utilisant un retour par l'état estimé tel que donné en (5.19), l'équation (5.18) donne :

$$\lambda_k = -M^{-1}.F.M.\lambda_{k-1} + M^{-1}.\left(c_k - F^2.\xi_k^*\right) \tag{5.20}$$

La relation (5.20) permet de calculer par récurrence la valeur de la condition initiale de l'état d'un CCM. Ainsi, nous utilisons pour l'adaptation d'un CCM non optimisé, un SLCM défini par $\Sigma_c(k.R, \alpha, 0, I_n, \gamma)$ avec une entrée *à discrétiser* donnée par :

$$\psi(t) = M^{-1}\left\{c(t) - F.(F.\xi^*(t) + M.\lambda(t-R))\right\} \tag{5.21}$$

Remarque 5.7. *La détermination de λ_k provenant d'une récurrence, il est nécessaire de faire apparaître le signal $\lambda(t)$ à l'extérieur du SLCM afin de faire un bouclage. De ce fait, nous prenons, pour le SLCM, $\beta_d = I_m$ en injectant directement $\lambda(t)$ dans son entrée à discrétiser.*

Remarque 5.8. *Le calcul de la condition initiale λ_k (5.20) fait apparaître une récurrence en λ. Cette relation est stable si le processus, et donc la matrice F, sont stables. La solution proposée n'est donc réalisable que pour des processus stables.*

5.5.1.4. Architecture

L'architecture correspondant au formalisme proposé est donnée en figure 5.12.

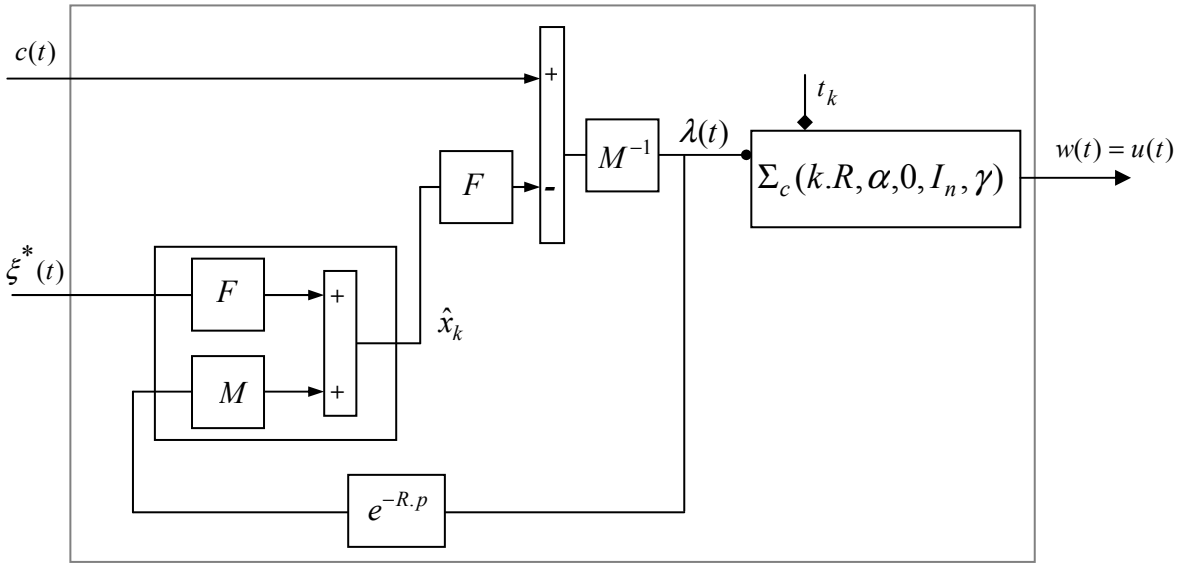


Fig. 5.12. *Architecture d'un CCM cadencé à une période égale au retard*

5.5.1.5. Conclusion

La présente méthode permet de réaliser la poursuite échantillonnée (à période R) d'une trajectoire de consigne par l'état du processus stable avec un retard R. Elle garantie donc $\xi_k^* = x_{k-1} = c_{k-2}$.

Néanmoins, elle présente un intérêt seulement dans le cas où $N = 1$, car autrement, l'information de retour (issue du sur-échantillonnage à t_e) n'est pas totalement exploitée.

Toutefois, afin d'exploiter l'information de retour entre les instants de commutation, il serait intéressant de considérer une optimisation du contrôleur. Le retour étant sous forme échantillonnée, il est tout de même préférable d'opter pour une telle stratégie par le biais d'un CBE. Cette étude est considérée dans le prochain paragraphe.

5.5.2. Adaptation d'un CBE commutant à *R*

En considérant un retour par $\xi^*(t) = x^*(t-R)$, il est possible d'envisager un asservissement par un CBE en imposant à celui-ci une commutation suivant $S_k = \{k.R, k = 0,1,2,...\}$ avec $R = N.t_e$ et un fonctionnement échantillonné à $i.t_e$ entre les instants de commutation, soit $S_i = \{i.t_e, i = 0,...,N-1\}$ (dans le présent cas, puisqu'il existe N échantillons du retour sur un morceau k, nous choisissons pour le contrôleur $q = N$). Selon les échelles de temps discret du CBE, l'information la plus récente sur le retour est donnée par $\xi_k^i = x_{k-1}^i$.

5.5.2.1. Mise en équation

Par rapport au formalisme des CBE, le processus est *vu* comme un système discret défini par :

$$x_k^{i+1} = f.x_k^i + h.u_k^i \text{ pour } i = 0,...,N-1 \text{ sur un morceau } k, \tag{5.22a}$$

$$x_k^0 = x_{k-1}^N \qquad \forall t_k^0 \tag{5.22b}$$

$$y_k^i = c.x_k^i \qquad \forall t_k^i \tag{5.22c}$$

avec $$f = e^{a.t_e} \text{ et } h = f.\int_0^{t_e} e^{-a.\tau}.b.d\tau \tag{5.22d}$$

5.5.2.2. Prédiction

Contrairement au chapitre 4, nous ne disposons que de l'état retardé. L'idée est donc de concevoir un estimateur qui, à partir du signal $\xi^*(t)$ et de la commande, calcule une estimée de l'état actuel :

$$\tilde{x}_k^i = f^N.\xi_k^i + \underbrace{\left[f^{N-1}.h \quad f^{N-2}.h \quad \cdots \quad f^0.h\right]\begin{bmatrix} u_k^{i-N} \\ u_k^{i-N+1} \\ \vdots \\ u_k^{i-1} \end{bmatrix}}_{P} \tag{5.23}$$

Le terme P est facilement réalisable en pratique grâce au formalisme des SLBE (cf. §5.5.2.4).

5.5.2.3. Commande

Ainsi, en posant comme condition de poursuite $x_{k+1}^0 = x_k^N = c_k^0$, $\forall k = 0,1,2,...$, $c(t)$ étant la consigne d'état désirée, nous pouvons utiliser les CBE tel que présentés chapitre 4 en utilisant l'estimation $\tilde{x}_k^i$ à la place de l'état réel du processus. Il est évident qu'une version optimisée d'un CBE convient à cette mise en œuvre afin d'exploiter tous les échantillons du signal de retour.

Remarque 5.9. *De la même manière que précédemment, la mise en équation résultant de la combinaison prédicteur/contrôleur permet de réaliser la poursuite seulement si le processus est stable. Autrement, le prédicteur diverge.*

5.5.2.4. Architecture

En pratique, le terme *P* de l'équation de prédiction (5.23) peut être considéré comme étant la sortie d'un SLBE défini par un SLBE auxiliaire $\Sigma_d(\{i.t_e\}, t_0, f, I_n, 0, I_n)$ avec $\psi(t) = 0$ et $\varphi(t) = h.u(t) - f^N.u(t-R)$.

Ainsi, nous pouvons proposer comme structure du présent formalisme, le schéma de la figure 5.13, les paramètres du SLBE du contrôleur étant tels que définis au chapitre 4.

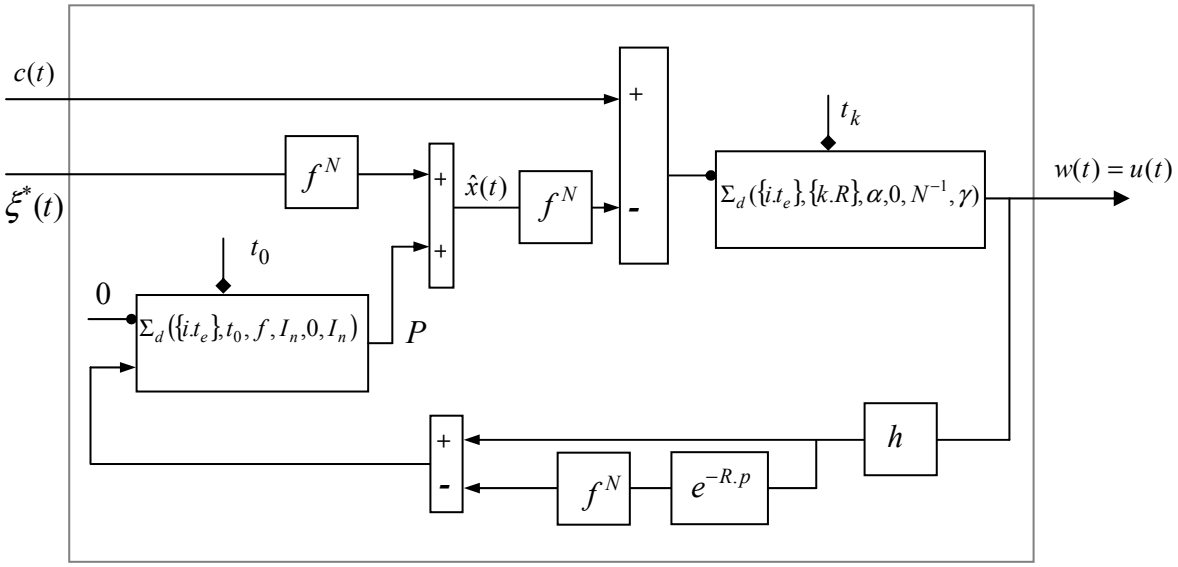


Fig. 5.13. *Architecture d'un CBE cadencé à t_e et adapté à un retard R*

5.5.2.5. Conclusion

De la même manière que précédemment, la présente méthode permet de réaliser la poursuite échantillonnée d'une trajectoire de consigne par l'état du processus stable avec un retard *R*, garantissant $\xi_k^0 = x_{k-1}^0 = c_{k-2}^0$. Toutefois, une éventuelle utilisation d'un CBE optimisé permet de prendre en compte le retour même entre les instants de commutation dans le cas où $N > 1$.

À noter que selon la définition des CBE au chapitre 4, la poursuite n'existe que si $N > n$, *n* étant l'ordre du processus à commander.

Par ailleurs, afin de réduire le retard de la poursuite, nous proposons au prochain paragraphe, l'utilisation d'un contrôleur commutant à la période d'échantillonnage du retour.

5.5.3. Adaptation d'un CCM commutant à t_e

D'après les conclusions tirées du chapitre 4, la performance d'un CCM est meilleure si la commutation est réalisée plus fréquemment. Dans ce sens, afin d'améliorer la méthode précédente, nous envisageons, ici, la commande par un CCM non optimisé cadencé à période égale à la période d'échantillonnage du signal de retour, soit $T = t_e$. Ainsi, lors de l'écriture des équations, nous comprenons que le morceau *k* est de durée t_e. Sachant que le retard induit est de $R = N.t_e$, l'information la plus récente sur l'évolution du processus est $\xi_k^* = x_{k-N}$.

5.5.3.1. Mise en équations

Comme précédemment, l'objectif est de paramétrer convenablement $\Sigma_c(S, \alpha, \beta_c, \beta_d, \gamma)$ pour définir un CCM adapté au cas présent. Nous choisissons, au préalable :

- $S = \{k.T, k = 0,1,2...\}$ avec $T = t_e$, commutant ainsi le contrôleur à la période d'échantillonnage du retour,
- $\varphi(t) = 0$, $\beta_c = 0$ car le CCM ne possède pas d'entrée continue dans le cas non optimisé,
- $\alpha \in \Re^{n \times n}$ tel que l'évolution du SLCM soit stable sur un morceau *k*,
- $\gamma \in \Re^{r \times n}$ de rang plein,

Reste à déterminer la matrice β_d et l'entrée $\psi(t)$ à discrétiser du SLCM.

Par ailleurs, selon le formalisme des CCM, l'évolution de l'ensemble contrôleur-processus est donnée par :

$$x_{k+1} = f.x_k + M.\lambda_k \tag{5.24a}$$

$$\text{avec } f = e^{a.t_e} \text{ et } M = f.\int_0^{t_e} e^{-a.\tau}.b.\gamma.e^{\alpha.\tau} d\tau \tag{5.24b}$$

Toutefois, l'état actuel du processus étant indisponible, nous proposons une méthode de prédiction basée sur le formalisme des SLCM, qui calcule une estimée de l'état à partir de l'information retardée.

5.5.3.2. Prédiction

L'état actuel étant indisponible, nous proposons une estimation à partir de l'information retardée comme suit :

$$\widetilde{x}_k = F.x_{k-N} + V_k \tag{5.25a}$$

$$\text{avec } F = e^{a.N.t_e} \text{ et } V_k = F.\int_{(k-N)t_e}^{kt_e} e^{-a\tau}.b.u(\tau).d\tau \tag{5.25b}$$

Remarque 5.10. *V_k peut être calculé comme étant la solution de l'équation différentielle suivante :*

$$V'(t) = a.V(t) + b.u(t) - F.b.u(t - N.t_e)\,, \tag{5.26a}$$

$$\text{avec } V(0) = 0\,. \tag{5.26b}$$

En pratique, nous obtenons V_k en utilisant un SLCM défini par $\Sigma_c(\{0\}, a, b, I_n, I_n)$ avec :

- *pour l'entrée $\psi(t)$ à discrétiser, une entrée vectorielle nulle de dimension n,*
- *une entrée continue donnée par $\varphi(t) = b.u(t) - F.b.u(t-R)$.*

5.5.3.3. Commande

En remplaçant la valeur de l'estimée de l'état actuel dans l'équation de fonctionnement de la structure bouclée d'un CCM (remarque 5.2), nous avons :

$$x_{k+1} = f.(F.x_{k-N} + V_k) + M.\lambda_k \tag{5.27a}$$

$$\text{avec } f = e^{a.t_e} \text{ et } M = f.\int_0^{t_e} e^{-a.\tau}.b.\gamma.e^{\alpha.\tau}d\tau \tag{5.27b}$$

Ainsi, en imposant la condition de poursuite $x_{k+1} = c_k$, nous pouvons définir, pour le contrôleur CCM adapté :

$$\beta_d = M^{-1}\,, \tag{5.28a}$$

$$\psi(t) = c(t) - f.(F.\xi^*(t) + V(t))\,, \tag{5.28b}$$

sous réserve que M soit inversible.

5.5.3.4. Architecture

L'architecture correspondant au formalisme proposé est donnée en figure 5.14.

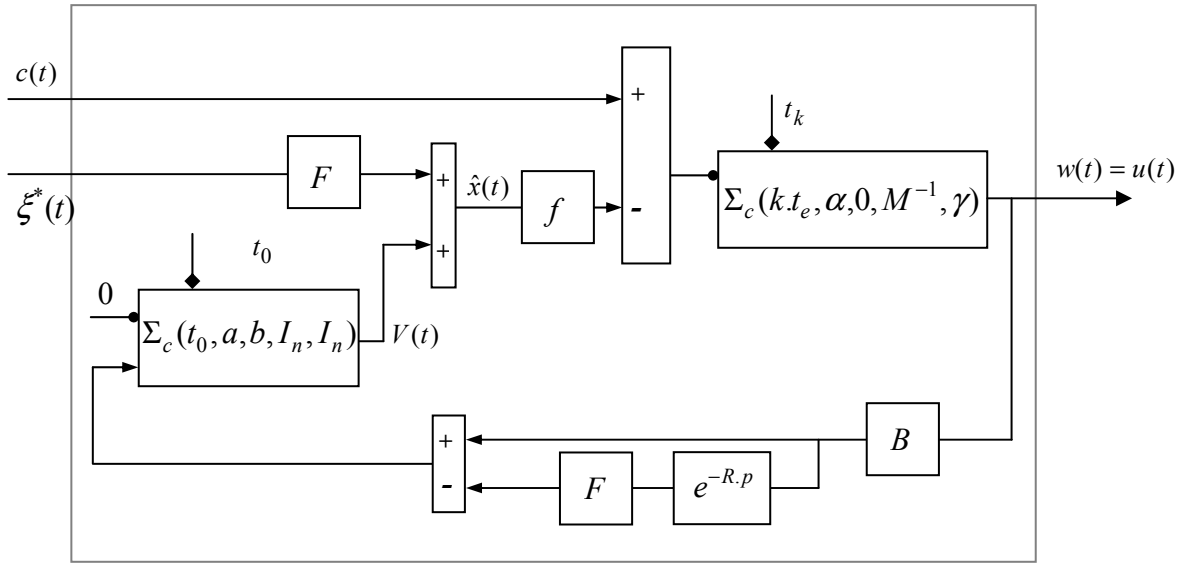


Fig. 5.14. *Architecture d'un CCM cadencé à t_e et adapté à un retard R*

Remarque 5.11. *De la même manière que précédemment, la mise en équation résultant de la combinaison prédicteur/contrôleur permet de réaliser la poursuite seulement si le processus est stable. Autrement, le prédicteur diverge.*

5.5.3.5. Conclusion

La présente méthode permet de réaliser la poursuite échantillonnée (à t_e) d'une trajectoire de consigne par l'état du processus avec un retard égal à la période d'échantillonnage du signal de retour. Elle réalise donc $\xi_k^* = x_{k-N} = c_{k-N-1}$.

Par conséquent, elle présente un avantage par rapport aux contrôleurs commutant à une période égale au retard présent sur le retour d'information dans le cas où $N > 1$. Garantissant une poursuite rigoureuse à chaque $k.t_e$, elle est plus performante que la méthode du CBE optimisé qui non seulement introduit un retard *R* sur la poursuite, mais qui *minimise* les oscillations sur un morceau de durée *R*, sans imposer une coïncidence exacte à chaque $k.t_e$.

À noter qu'une optimisation est inutile dans ce cas car l'information entre les instants de commutation est nécessairement constante (bloquée).

5.5.4. Validation expérimentale

5.5.4.1. Commutation du contrôleur à *R*

5.5.4.1.1. Exemple d'un CCM en simulation

Considérons un processus tel que décrit par (5.1), avec :

$$a = \begin{bmatrix} 0 & 1 \\ -1 & -10 \end{bmatrix},\ b = \begin{bmatrix} 0 \\ 5 \end{bmatrix} \text{ et } x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

et dont l'évolution n'est disponible que depuis $\xi^*(t)$.

Dans le but d'imposer, à ce processus, une consigne d'état définie par :

$$c(t) = \begin{bmatrix} \sin(2t) \\ 2\cos(2t) \end{bmatrix},$$

nous utilisons une structure de commande telle que donnée sur la figure 5.12.

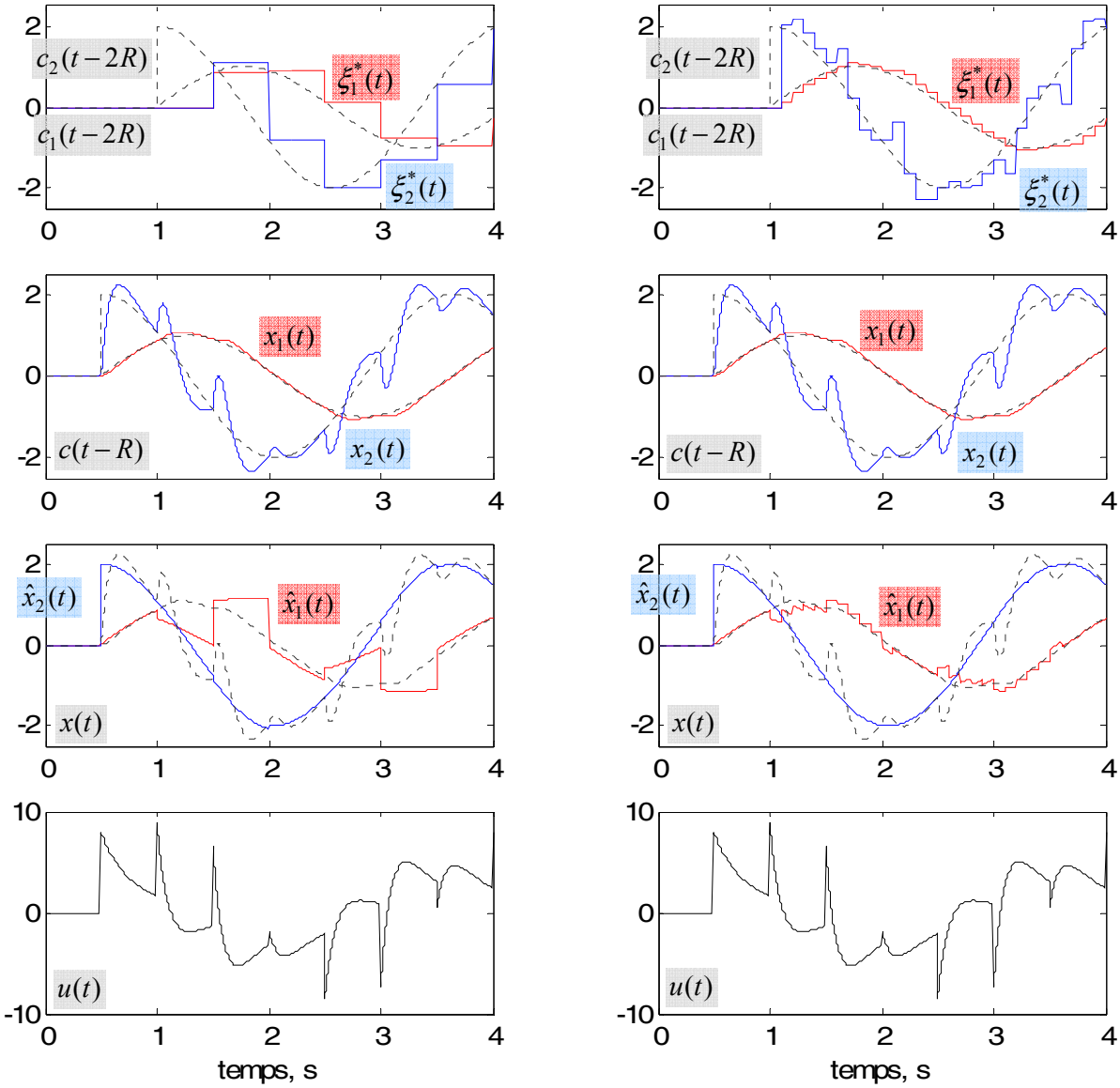
Le contrôleur est basé sur $\Sigma_c(k.R, \alpha, 0, M^{-1}, \gamma)$ qui est caractérisé par :

$$\alpha = \begin{bmatrix} -3 & 1 \\ 1 & -10 \end{bmatrix},\ \gamma = \begin{bmatrix} 10 & 10 \end{bmatrix}.$$

Les résultats sont représentés sur la figure 5.15. Dans tous les cas, nous considérons que le retard induit sur le retour d'état est $R = 0.5\text{s}$. Par ailleurs, nous proposons en figure 5.15a, le cas où $N = 1$ et $t_e = 0.5\text{s}$ et en figure 5.15b, le cas où $N = 5$ et $t_e = 0.1\text{s}$.

Afin de mieux interpréter les courbes, la consigne d'état $c(t)$ est retardée respectivement de $2R$ et de R pour être comparée à $\xi^*(t)$ et à $x(t)$. Ainsi, nous constatons que le suivi est réalisé, notamment par l'état du processus avec un retard de *R*.

Toutefois, il est évident que même si $t_e = 0.1\text{s}$, R étant le même, les résultats sont les mêmes, car les informations entre les instants de commutation ne sont pas prises en compte. Nous pouvons constater que dans ce cas, $\xi^*(t)$ n'est pas égal à $c(t-2R)$ à chaque instants d'échantillonnage du retour, mais aux instants de commutation du contrôleur, soit à chaque 0.5s.



(a). $N = 1,\ t_e = 0.5s$ ***(b).*** $N = 5,\ t_e = 0.1s$

Fig. 5.15. *Commande par retour d'état retardé et échantillonné avec un CCM cadencé à* $R = 0.5s$

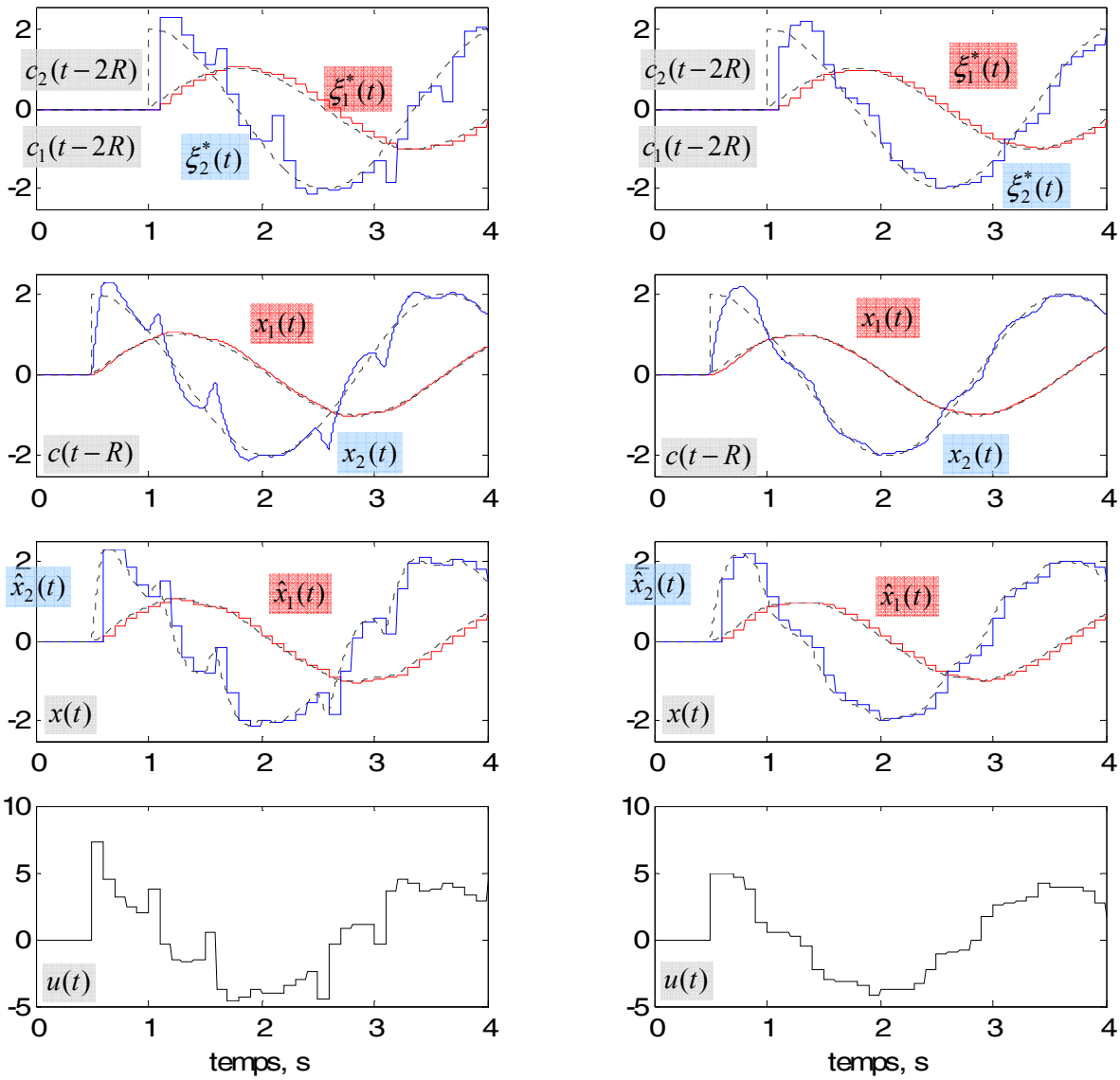
5.5.4.1.2. Exemple d'un CBE en simulation

En reprenant le même processus que dans l'exemple précédent, nous réalisons un asservissement par un CBE cadencé à R (§5.5.2) pour lui imposer la même consigne. Toutefois, le processus étant d'ordre 2, nous ne pouvons pas considérer le cas où $N = 1$ et $t_e = 0.5\text{s}$ comme précédemment (une commande CBE existe seulement si $N > n$ comme précisé précédemment).

Les résultats sont illustrés sur la figure 5.16a pour un CBE non optimisé et sur la figure 5.16b pour un CBE optimisé. Dans les deux cas, $N = 5$ et $t_e = 0.1\text{s}$.

Nous constatons que même si la commande diffère, la poursuite avec le CBE non optimisé est presque identique que celle du CCM (figure 5.15b), car dans les deux cas, le contrôleur commute à R et *néglige* les informations entre les instants de commutation. Par ailleurs, les résultats sont améliorés (figure 5.16b) grâce à l'optimisation qui minimise l'écart entre l'état et sa consigne *entre* les instants de commutation.

Toutefois, un contrôleur commutant à période R implique nécessairement un retard de R dans la poursuite de la consigne par $x(t)$ et de $2R$ dans celle de la consigne par $\xi^*(t)$.



(a). *CBE non optimisé, $N = 5$, $t_e = 0.1s$* ***(b).*** *CBE optimisé, $N = 5$, $t_e = 0.1s$*

Fig. 5.16. *Commande par retour d'état retardé et échantillonné avec un CBE cadencé à $R = 0.5s$*

5.5.4.1.3. *Commande d'un processus réel*

Nous considérons, ici, l'asservissement du processus réel de l'annexe I par un retour d'état retardé et échantillonné ($\xi^*(t)$), avec $N=1$ et $t_e=0.03\text{s}$. Dans le but d'imposer une consigne d'état $c(t)$ telle que représentée sur la figure 5.17 (trajectoire de position sinusoïdale d'amplitude 0.05 et de pulsation 2.5rad/s), nous mettons en œuvre une architecture de commande identique à celle du §5.5.4.1.1, en imposant une commutation du contrôleur à $R=0.03\text{s}$.

Les résultats de la figure 5.17 montrent la robustesse de la méthode face aux bruits de mesure et aux non linéarités du processus.

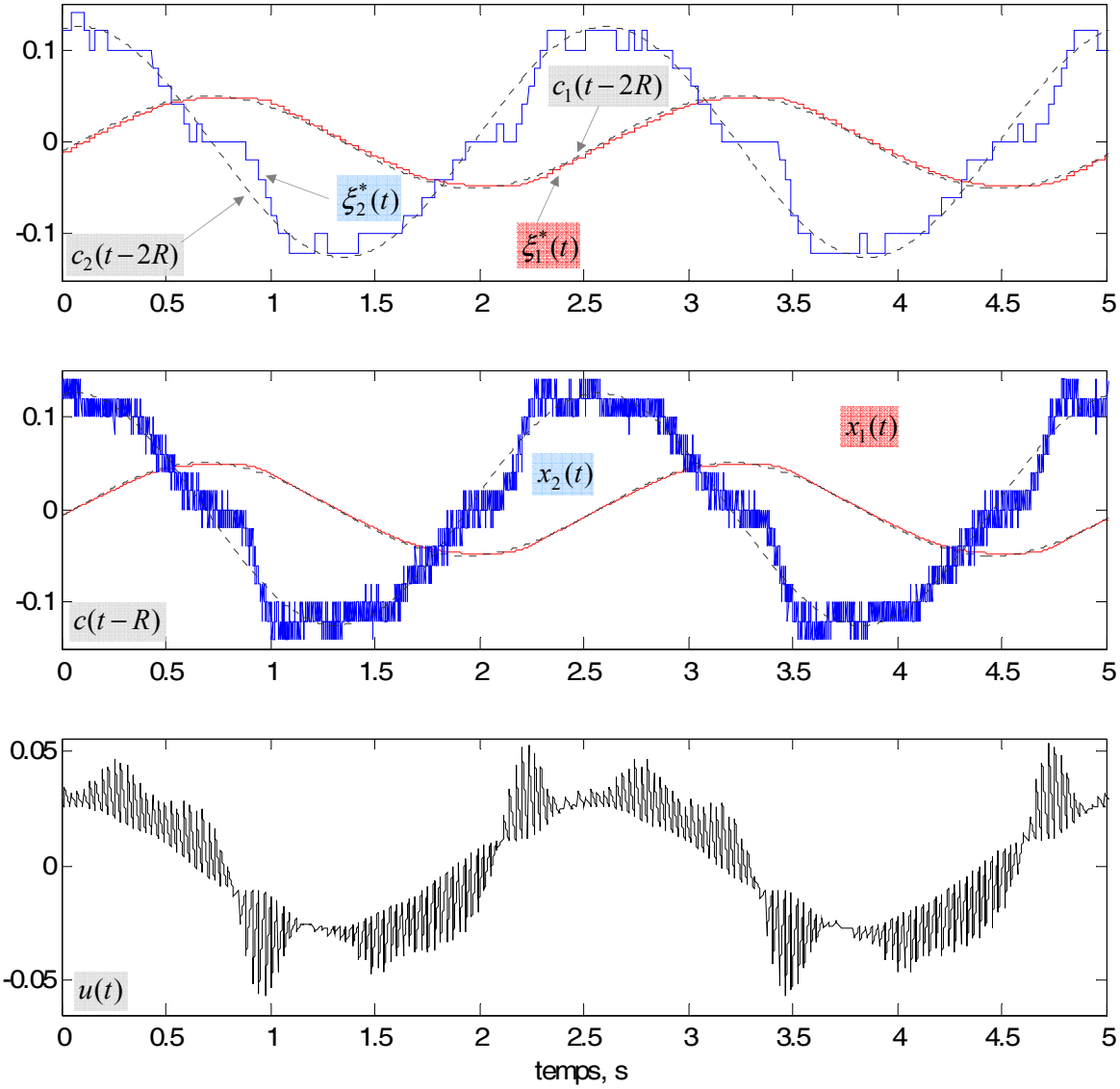
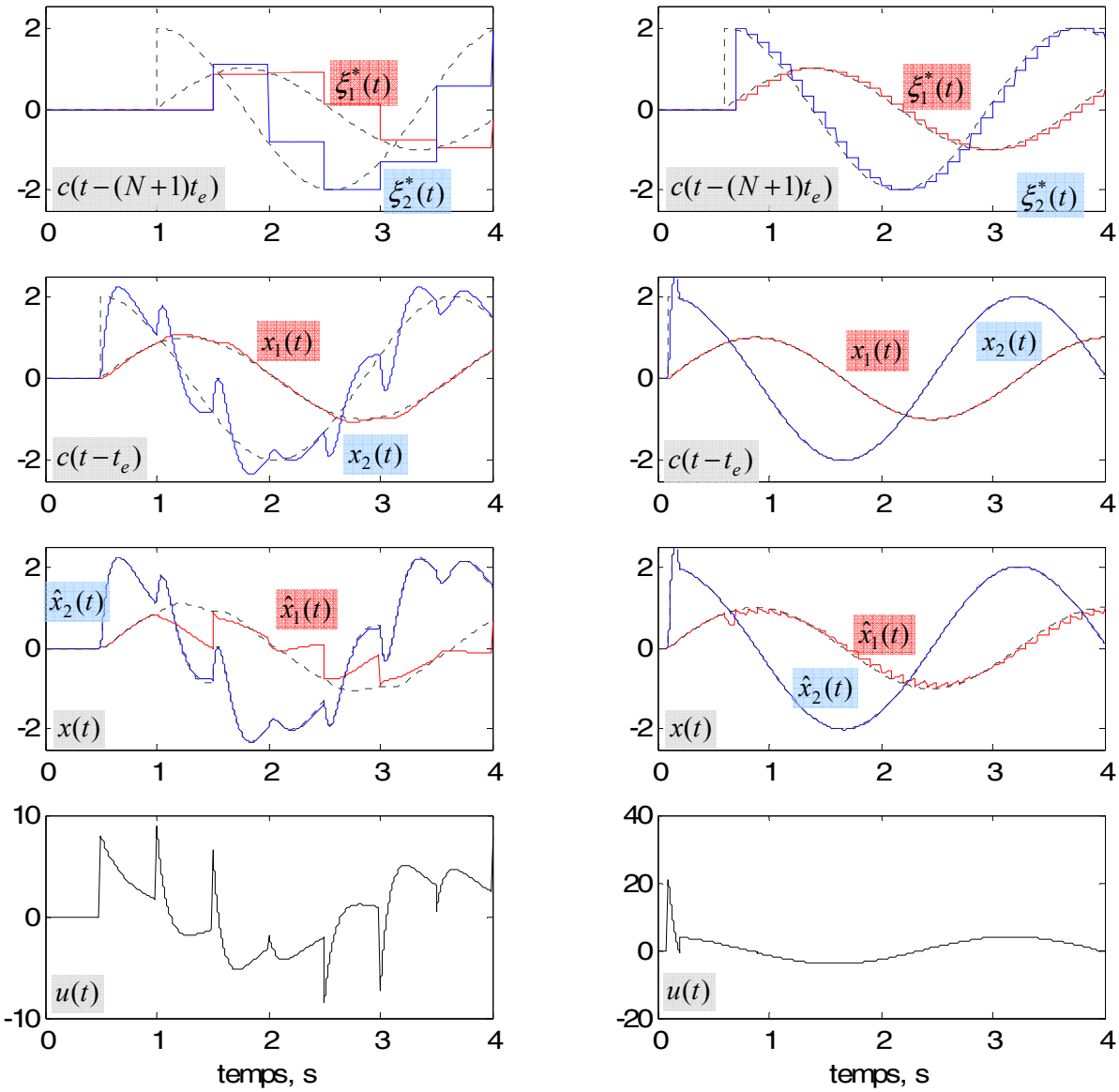


Fig. 5.17. *Commande par retour d'état retardé et échantillonné avec un CCM cadencé à R = 0.5s*

5.5.4.2. Commutation du contrôleur à t_e

Nous considérons ici, le même processus et la même consigne que dans le §5.5.4.1.1 (commutation du contrôleur à R). Par contre, nous réalisons un asservissement par l'architecture décrite au §5.5.3. Le contrôleur utilisé est basé sur un SLCM défini par $\Sigma_c(k.t_e, \alpha, 0, M^{-1}, \gamma)$ avec les mêmes paramètres, sauf que la commutation est à la période d'échantillonnage du signal de retour.

Les résultats sont représentés sur la figure 5.18 pour les mêmes paires de valeurs de (N, t_e). Nous pouvons voir que la poursuite est meilleure dans le présent cas pour $N > 1$.



(a). $N = 1,\ t_e = 0.5s$ ***(b).*** $N = 5,\ t_e = 0.1s$

Fig. 5.18. *Commande par retour d'état retardé et échantillonné avec un CCM cadencé à t_e*

En effet, les résultats montrent que dans tous les cas, l'état du processus (supposé inaccessible pour l'asservissement) suit sa consigne avec un retard de seulement t_e avec une coïncidence exacte à chaque $k.t_e$ (instants de commutation du contrôleur). Par conséquent, la méthode garantie $\xi_k^* = x_{k-N} = c_{k-N-1}$ impliquant un retard de $R + t_e$ entre $\xi^*(t)$ et $c(t)$. Ainsi, la méthode présente un intérêt dans le cas où $N > 1$.

5.5.5. Conclusion

Dans le cas d'un retour d'état échantillonné, nous avons effectué une estimation de l'état actuel par un formalisme SFM afin de se ramener à une utilisation classique des CFM.

Le retard induit, dans le cas général, étant multiple de la période d'échantillonnage du retour ($R = N.t_e$), nous avons envisagé des contrôleurs commutant soit à une période égale au retard *R*, soit à une période égale à t_e.

Dans le premier cas, nous proposons l'adaptation d'un CCM et celle d'un CBE. Il est évident que la commutation à *R* introduit un sur-échantillonnage négligeant les échantillons de retour d'informations entre les instants de commutation. Une telle approche permet néanmoins de réaliser une optimisation du système bouclé (identique à l'optimisation du chapitre 4), particulièrement avec le CBE, en exploitant au mieux le retour échantillonnée. Dans ce cas, la méthode garantit une poursuite échantillonnée d'une consigne d'état avec un retard *R*.

Toutefois, le deuxième cas aboutit à une architecture de commande qui exploite naturellement tous les échantillons du signal de retour, car le contrôleur commute à t_e. Dans ce cas, l'optimisation ne présente aucun intérêt car le signal de retour ne présente aucune évolution entre les instants de commutation du contrôleur. Par ailleurs, la commutation étant plus rapide que dans le premier cas, la performance du contrôleur est meilleure. Dans ce cas, la mise en œuvre de la version non optimisée (moins coûteuse) garantit naturellement une poursuite échantillonnée à chaque t_e, l'état du processus suivant sa consigne avec un retard de seulement t_e.

5.6. Retour par la sortie retardée

Le retour par la sortie retardée correspond à une combinaison de deux contraintes : le retard et l'indisponibilité de l'état complet. Il est évident que la seule possibilité d'utilisation d'un CFM est d'inclure, dans la boucle de rétroaction, un observateur et un prédicteur d'état. En effet, les stratégies de prédiction que nous avons évoqué précédemment requièrent toutes les variables d'état. Dans ce sens, nous proposons de reconstruire par un observateur, l'état retardé à partir de la sortie retardée. Ainsi, l'état actuel peut être estimé par le prédicteur à partir de la cette reconstitution de l'état retardé. Finalement, avec une estimation de l'état actuel provenant du prédicteur, l'asservissement par un CFM de base devient possible. Ce principe est illustré par la figure 5.19 dans le cas d'un retour de sortie continue (traits unis) ou échantillonnée (traits pointillés).

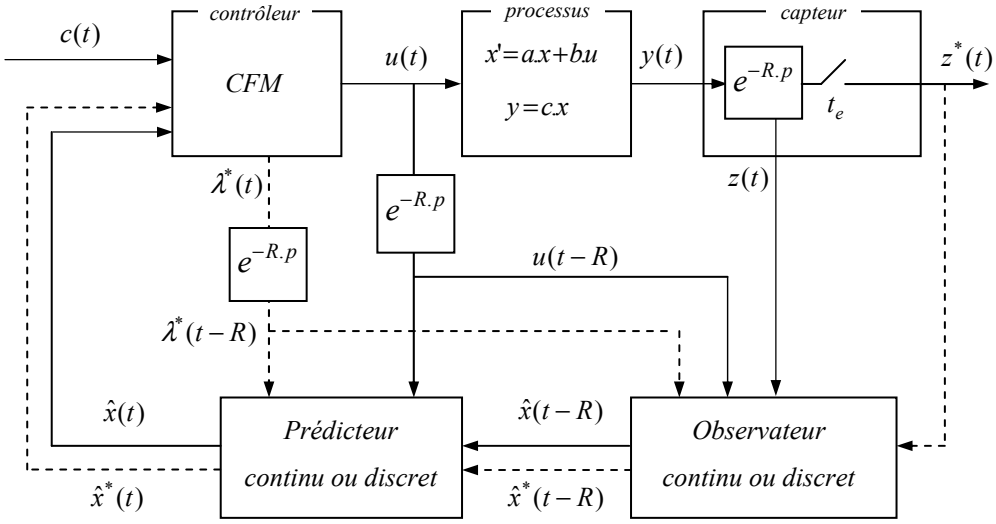


Fig. 5.19. *Schéma de principe de commande CFM par la sortie retardée*

Remarque 5.12. *Grâce au fonctionnement par morceaux des CFM, l'utilisation d'un observateur associé ne requiert pas de détermination de paramètres suivant le principe de séparation dans le cas non optimisé, comme mentionné dans le §5.3.1.*

Dans tous les cas, la structure garantit la poursuite échantillonnée d'une consigne d'état telle que l'état du processus suit sa consigne avec un retard égal à la période de commutation T du contrôleur et avec une coïncidence à chaque instant de commutation. Par conséquent, la sortie du capteur suit sa consigne correspondante avec un retard de $T+R$ et avec une coïncidence à chaque instant de commutation.

5.6.1. Retour continu

Dans le cas d'une sortie continue, il convient d'utiliser un observateur continu de type Luenberger, éventuellement d'ordre réduit dans le cas où la sortie correspond à une (ou plusieurs) variable(s) d'état. Cet observateur prend comme entrées :

- la commande continue par morceaux (provenant d'un CCM adapté) retardée de R ($u(t-R)$),
- la sortie continue et retardée ($z(t)$),

afin de générer une estimation de l'état retardé continu ($\hat{x}(t-R)$).

En référence à la figure 5.19, les connexions de l'observateur sont alors comme l'indiquent les traits unis.

À partir de cette estimation de l'état retardé disponible à la sortie de l'observateur continu, l'ensemble prédicteur-contrôleur prend la forme présentée à la section 5.4.

5.6.2. Retour échantillonné

Dans le cas d'une sortie échantillonnée, une version discrète de l'observateur est envisageable. Comme dans le cas continu, celui-ci peut être d'ordre réduit si la sortie correspond à une (ou plusieurs) variable(s) d'état.

En général, un observateur discret utilise l'entrée et la sortie *échantillonnées* du processus pour générer une estimation de l'état sous forme échantillonnée. Classiquement, pour un système tel que décrit par (5.1), il convient d'effectuer, pour l'observateur discret, une mise en forme discrète de l'évolution du processus :

$$x_{k+1} = f.x_k + h.u_k \tag{5.29a}$$

$$y_k = c.x_k \tag{5.29b}$$

$$z_k^* = y_{k-1} \tag{5.29c}$$

avec $$f = e^{a.t_e} \text{ et } h = f.\int_0^{t_e} e^{-a.\tau}.b.d\tau \tag{5.29d}$$

Afin de calculer une estimation de l'état, l'observateur discret requiert que les signaux d'entrée et de sortie du processus soient échantillonnés à la même période.

En présence d'un capteur numérique, la sortie $z^*(t)$ est échantillonnée à t_e. En considérant (5.29) il convient d'appliquer au processus une commande échantillonnée à t_e afin que l'observateur prenne comme entrée u_{k-1} et $z^*(t)$ pour générer $\hat{x}_{k-1}$.

Grâce aux CFM, la génération d'une commande échantillonnée à t_e est possible, mais on est limité à l'utilisation d'un CBE avec $S_i = \{i.t_e, i = 0,...,N-1\}$ ou alors un CCM avec $S_k = \{k.t_e, k = 0,1,2,...\}$ restreint à une évolution constante par morceaux de son état.

Toutefois, afin d'admettre une commande *continue* par morceaux générée par un CCM (tel qu'au §5.5.1.2), nous considérons non pas uniquement le processus, mais l'ensemble CCM-processus dont l'évolution est régie par :

$$x_{k+1} = f.x_k + M.\lambda_k\,, \tag{5.30a}$$

$$y_k = c.x_k \tag{5.30b}$$

$$z_k^* = y_{k-1} \tag{5.30c}$$

avec $f = e^{a.t_e}$ et $M = f.\int_0^{t_e} e^{-a.\tau}.b.\gamma.e^{\alpha.\tau}d\tau$ (5.30d)

Ainsi, même si l'entrée $u(t)$ n'est pas échantillonnée à t_e, nous pouvons utiliser comme entrée de l'observateur, $z^*(t)$ et λ_{k-1} (toutes deux échantillonnées à t_e) afin d'obtenir une estimation de l'état retardé sous forme échantillonnée à t_e, soit $\hat{x}_{k-1}$. Ce principe d'observation est illustré figure 5.19 par les connexions pointillées de l'observateur discret. Il suffit alors d'effectuer une prédiction pour faire une commande par retour d'état estimé comme dans la section 5.5.

5.6.3. Utilisation d'un observateur/prédicteur

Dans le cas d'un retour de sortie retardée (échantillonnée ou continue), il est possible d'envisager la combinaison de l'observation et de la prédiction pour générer directement l'état actuel à partir de l'information de retour. Il suffit alors d'utiliser un CFM tel que défini au chapitre 4 pour réaliser la poursuite échantillonnée en faisant un retour par l'état estimé. Dans ce sens, nous proposons d'utiliser un observateur/prédicteur proposé dans [CSV$^+$06, SMRD06] qui permet d'obtenir une estimation continue de l'état actuel à partir de la sortie retardée du processus.

L'observateur de [SMRD06] s'applique dans le cas où la sortie du processus présente un retard et un échantillonnage. Le principe de cet observateur est de considérer que l'échantillonnage engendre un retard variable de $t - t_k$ où t_k est l'instant

d'échantillonnage le plus récent. La méthode admet un échantillonnage apériodique, mais suppose tout de même que l'intervalle d'échantillonnage est borné et que la valeur maximale T est connue. Ainsi, $0 \le t_{k+1} - t_k \le T$ et le phénomène de retard lié à l'échantillonnage est noté $\delta = t - t_k$.

Par ailleurs, en considérant un retour par $z^*(t)$ qui engendre, par définition, un retard de R sur la mesure de la sortie du processus, il convient d'écrire $z^*(t) = y(t - \delta(t))$ où $\delta(t) = R + t - t_k$. Ainsi, le phénomène d'échantillonnage et celui du retard liés au capteur (par exemple du système de vision artificielle) sont ramenés à un problème de retard variable.

L'idée est alors de considérer un observateur Luenberger continu à retard pour le processus :

$$\hat{x}'(t) = a.\hat{x}(t) + b.u(t) - L(y(t - \delta(t)) - \hat{y}(t - \delta(t))) \tag{5.31a}$$

$$\hat{y}(t) = c.\hat{x}(t) \tag{5.31b}$$

Sachant que la paire (a, c) est observable, il est alors possible de déterminer un gain linéaire L tel que l'observateur converge exponentiellement vers le processus en l'absence du retard. Toutefois, les auteurs de [SMRD06] proposent un théorème permettant de définir ce gain L de sorte que la convergence soit possible même dans le cas d'un retard $\delta(t) = R + t - t_k$ sur la sortie du processus. L'observateur garantit une estimation continue de $x(t)$ même entre les instants d'échantillonnage et, considérant (5.1) et (5.31), le vecteur d'erreur $e(t) = x(t) - \hat{x}(t)$ évolue selon :

$$e'(t) = a.e(t) + L.c.e(t - \delta(t)) \tag{5.32}$$

Le théorème décrit dans [CSV$^+$06, SMRD06] permet de poser les conditions des LMI dont la résolution conduit à déterminer le gain L.

5.6.4. Validation expérimentale

5.6.4.1. Exemple de simulation

Considérons un exemple de simulation reflétant la problématique décrite dans l'introduction générale : l'asservissement en position d'un chariot par un retour visuel.

Le processus considéré (représentant l'ensemble amplificateur–moteur–chariot) est défini par (5.1) avec :

$$a = \begin{bmatrix} 0 & 1 \\ 0 & -120 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 350 \end{bmatrix} \text{ et } c = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Par ailleurs, en considérant que la position du chariot est donnée sous forme retardée et échantillonnée, nous prenons comme grandeur de retour, $z(t)^* = y^*(t-R)$, avec $R = t_e = 28\text{ms}$ ($N = 1$).

Nous utilisons un observateur/prédicteur tel que décrit au §5.6.3 pour estimer l'état actuel à partir de l'information de retour. La résolution des conditions des LMI conduit dans ce cas à :

$$L = \begin{bmatrix} -3.1225 \\ 0.0569 \end{bmatrix}$$

En ce qui concerne la commande, nous mettons en œuvre un CCM basé sur $\Sigma_c(\{k.T\}, \alpha, 0, M^{-1}, \gamma)$ avec :

$$\alpha = \begin{bmatrix} -0.1 & 0 \\ 0 & -0.2 \end{bmatrix}, \ \gamma = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

La consigne d'état imposée sur le processus est de la forme :

$$c(t) = \begin{bmatrix} c_1(t) \\ c_2(t) \end{bmatrix} = \begin{bmatrix} r.\sin(\omega.t) \\ r.\omega.\cos(\omega.t) \end{bmatrix}$$

Les résultats sont illustrés par la figure 5.20 pour un contrôleur cadencé à $T = 10\text{ms}$ et par la figure 5.21 pour $T = 20\text{ms}$ (afin de montrer le fonctionnement). À noter que les composantes de la consigne sont retardées convenablement afin de permettre une meilleure lecture des courbes.

La poursuite de la consigne par l'état du processus est illustrée par les figures 5.20a et 5.20c et les figures 5.21a et 5.21b. Nous constatons que cette poursuite est réalisée avec un retard de *T* et une coïncidence à chaque $k.T$.

Par ailleurs, la figure 5.20b représente la sortie du capteur suivant sa consigne avec un retard de *R*, et les figures 5.20e 5.21c représentent la commande.

Enfin, l'estimation de l'état est représentée sur la figure 5.20d. Afin de montrer la performance de l'observateur/prédicteur, nous donnons l'évolution de l'état estimé dans un cas où ses conditions initiales sont éloignées de celles du processus (figure 5.22).

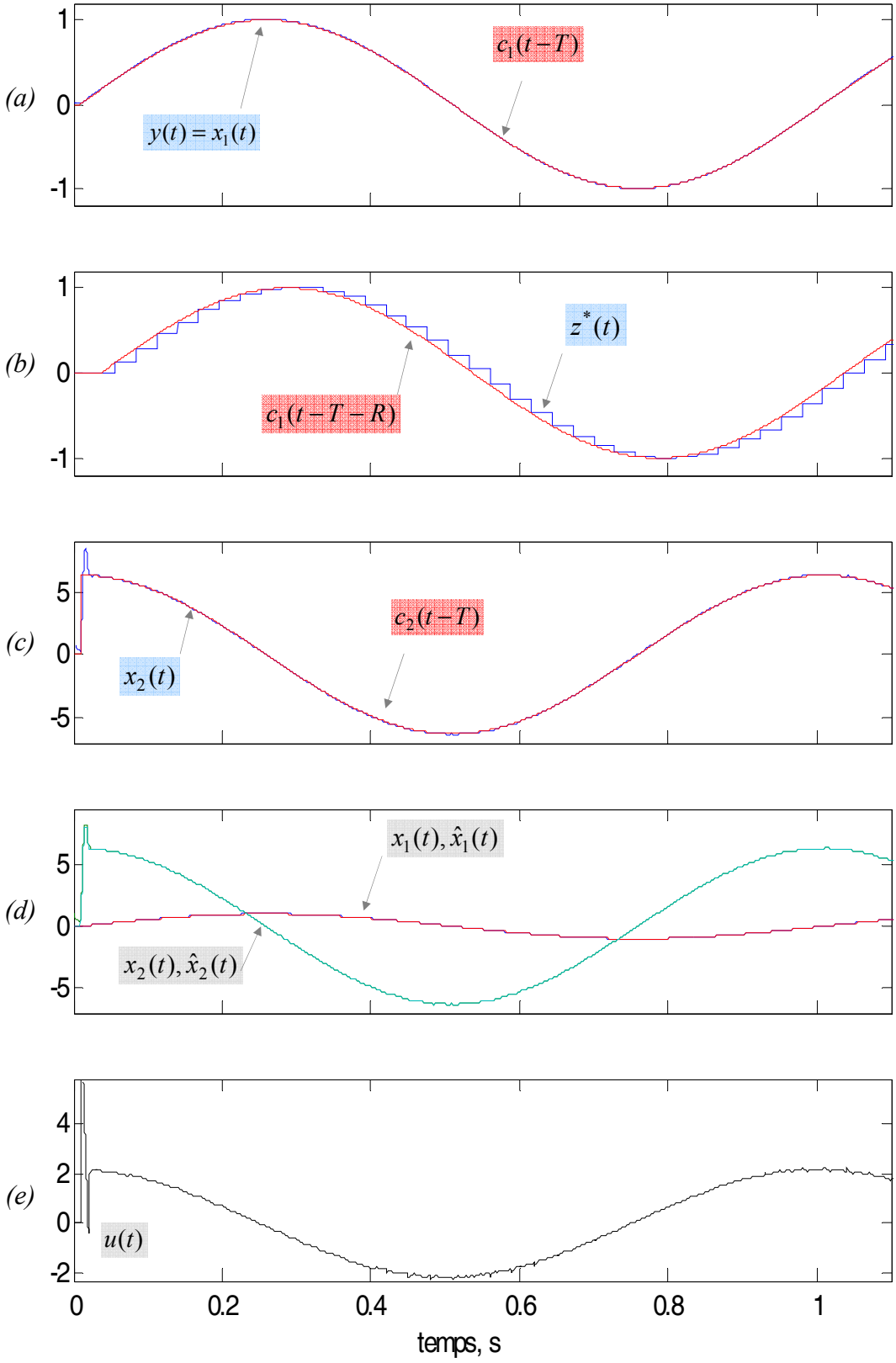


Fig. 5.20. *Poursuite échantillonnée ($T = 10ms$, $r = 1$, $\omega = 2\pi\, rad/s$)*

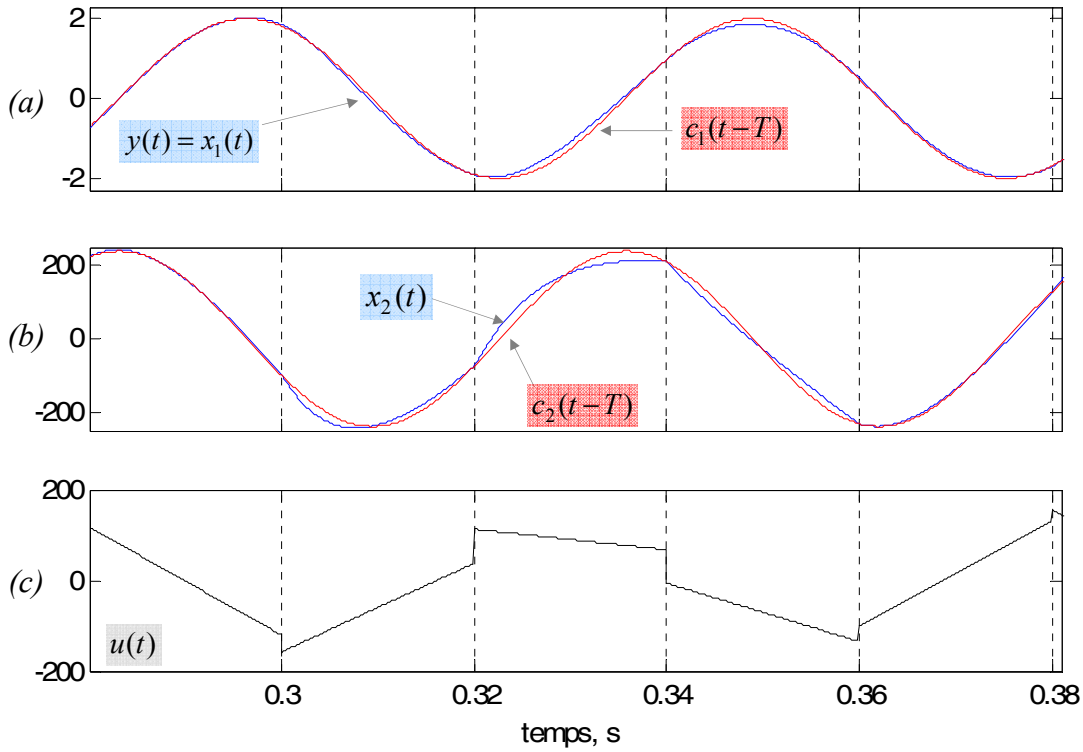


Fig. 5.21. *Poursuite échantillonnée ($T = 20ms$, $r = 2$, $\omega = 38\pi$ rad/s)*

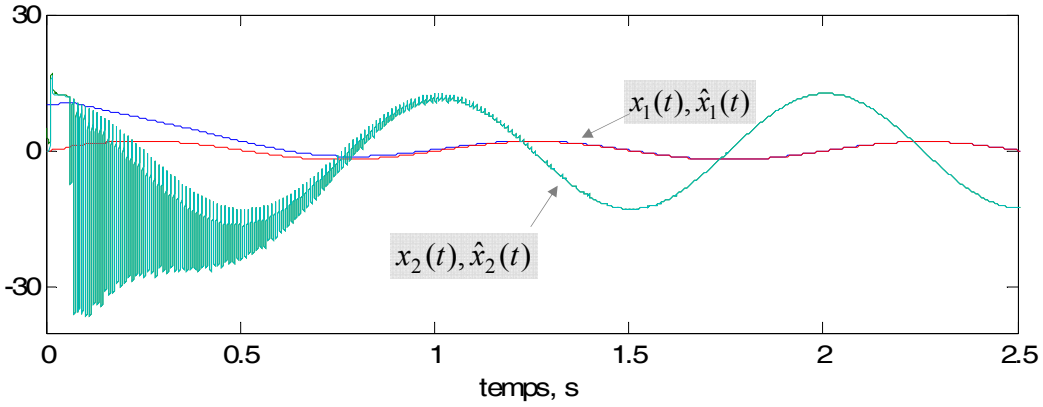


Fig. 5.22. *Performance de l'observateur/prédicteur ($r = 2$, $\omega = 2\pi$ rad/s)*

5.6.4.2. Processus réel

Nous avons réalisé le même type d'asservissement sur la maquette décrite en annexe I, en lui associant un modèle linéaire (comme au chapitre 4) et en utilisant un retour par le système de vision, soit $z(t)^* = y^*(t-R)$. En effet, la prise d'image est réalisée par un mode *reset* qui permet de définir une période d'échantillonnage constante suffisamment grande pour entreprendre l'acquisition d'une trame d'image et les calculs de traitement de l'image nécessaires. Ainsi, $R = t_e = 28\text{ms}$ ($N = 1$).

Les résultats sont illustrés sur la figure 5.23 comme dans l'exemple précédent.

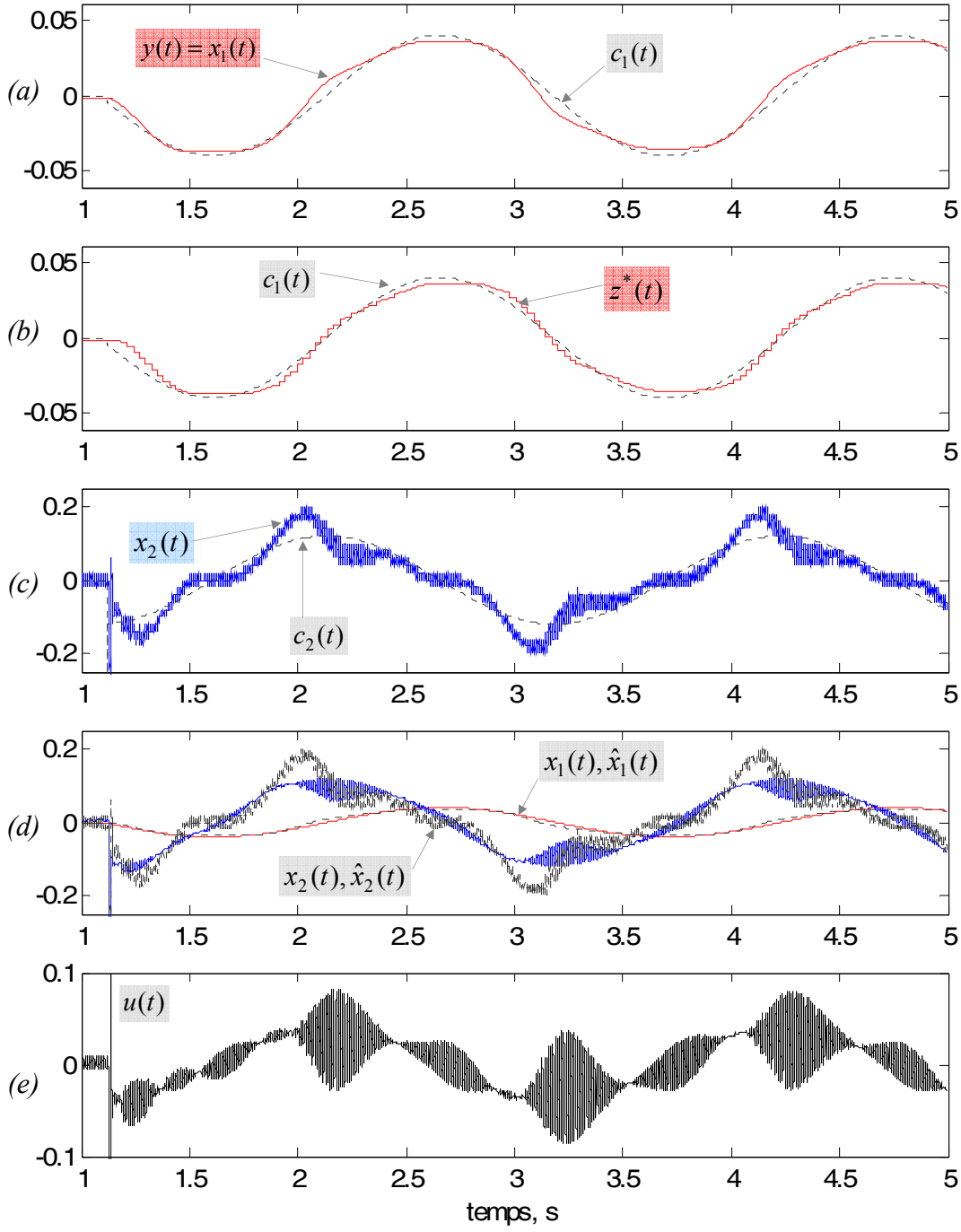


Fig. 5.23. *Asservissement visuel d'un chariot en position*

Les résultats montrent que la poursuite est réalisée avec quelques déformations (figures 5.23a et 5.23b). En effet, comme évoqué précédemment, le processus réel présente des non linéarités qui ne sont pas prises en compte par le modèle utilisé pour définir la loi de commande. Dans le cas d'un retour par le(s) codeur(s) du moteur, le retour quasi-continu permet de surmonter ces non linéarités. Par contre, dans le présent cas, la perte d'information sur l'évolution de l'état réel entre les instants d'échantillonnage combinée à l'erreur de prédiction due à une mauvaise modélisation conduit à un asservissement de qualité inférieure à celui dans le cas du retour par codeur(s).

5.7. Conclusion

Les CFM, tel que définis au chapitre 4, permettent de réaliser un asservissement robuste avec un retour d'état (continu ou échantillonné). Nous avons établi, au cours de ce chapitre, des adaptations du formalisme des CFM permettant de prendre en compte des retours plus restrictifs, dans le but de traiter des cas fréquemment rencontrés en pratique.

Afin de surmonter les contraintes liées à l'indisponibilité de l'état complet et à la présence d'un retard sur la chaîne de retour, nous proposons de manière générale des architectures incluant respectivement un observateur et un prédicteur. Bien évidemment, ces artifices sollicitant les paramètres du processus combinés à un contrôleur qui lui-même requiert ces paramètres, rendent la structure globale moins robuste face aux variations éventuelles du modèle du processus que dans le cas idéal d'un retour d'état.

Dans le cas d'un retour de sortie continue sans retard, nous proposons, en sus d'une approche par observateur, un contrôleur paradoxal provenant d'une simplification d'un CCM à commutation rapide. La méthode présente une robustesse accrue car elle ne nécessite pas la connaissance des paramètres du processus. De plus, elle est très simple à mettre en œuvre et très peu coûteuse en calcul temps réel.

En présence d'un retour retardé (par capteur), les méthodes proposées garantissent la poursuite échantillonnée d'une consigne d'état telle que l'état du processus suit sa consigne avec un retard égal à la période de commutation T du contrôleur et avec une coïncidence à chaque instant de commutation. Par conséquent, la sortie du capteur suit sa consigne correspondante avec un retard de $T + R$ et avec une coïncidence à chaque instant de commutation. Dans le cas d'un retour continu, la commutation du contrôleur peut être choisie très faible (la limitation étant liée à la technologie du calculateur) pour obtenir des performances idéales. Par contre, dans le cas d'un retour échantillonné (à t_e), la période commutation peut être au mieux égale à la période d'échantillonnage $T = t_e$.

Dans le cas de données échantillonnées (éventuellement retardées) il est possible de définir un modèle hybride au sens des SFM d'un processus pour estimer son évolution entre les instants d'échantillonnage. L'état de ce modèle peut alors être réinitialisé, à chacun de ces instants, à celui du processus (synchronisation). Ceci permet une amélioration de la performance entre les instants d'échantillonnage, réduisant ainsi les oscillations éventuelles. Toutefois, ce modèle utilise également les paramètres du processus. Par ailleurs, dans le cas des capteurs numériques, la période d'échantillonnage est suffisamment courte pour se passer d'une telle stratégie.

La contrainte retard pénalise le contrôle en terme de robustesse. En effet, le prédicteur estime l'état actuel du processus sur une période égale au retard. Il suppose non seulement

connaître le modèle du processus, mais également que celui-ci ne subit aucune perturbation extérieure. De la même façon, un échantillonnage contribue à une perte de robustesse, surtout quand le modèle présente une évolution différente de celle du processus et qu'en plus, la période d'échantillonnage est grande. Il est évident qu'avec un retour échantillonné à haute fréquence (ou continu dans le meilleur des cas) permet de surmonter une mauvaise modélisation. Toutefois, dans les architectures numériques, le retard et la période d'échantillonnage induits par les capteurs numériques restent suffisamment petits conduisant à un asservissement tolérable. Concrètement, l'exemple d'asservissement visuel montre que même si le processus (ensemble amplificateur–moteur–chariot) présente des non linéarités souvent négligées dans la loi de commande, un retour par capteur numérique permet tout de même un asservissement en position plutôt satisfaisant.

Chapitre 6

Conclusion et perspectives

Le développement de nouvelles méthodes d'identification en ligne et de commande temps réel par calculateur en présence de capteurs numérique constitue la principale motivation de ce mémoire. Ces nouvelles méthodes se basent sur une classe particulière de systèmes hybrides (les SFM) et utilisent une approche par la représentation d'état continu pour prendre en compte le cas général des processus réels MIMO.

Dans ce sens, nous avons commencé par exposer un état de l'art (chapitre 2) sur les systèmes hybrides au sens large et sur les SFM en particulier. Nous répertorions ainsi plusieurs conceptions hybrides existant dans la littérature et constatons qu'il est possible de classifier les diverses approches dans une classification universelle qui est celle donnée par la taxonomie de Branicky [Bra95, Bra98]. Toutefois, la majorité des propositions hybrides est destinée à la modélisation de systèmes réels complexes (systèmes naturellement hybrides, systèmes à commutation, systèmes non linéaires, etc.). Les modèles ainsi formulés servent donc à une étude théorique permettant de comprendre, commander et/ou améliorer le fonctionnement d'un système complexe. En général, ces modèles hybrides ne sont donc pas destinés à une implantation sur calculateur numérique. Par ailleurs, les auteurs de [KV01, KV02] ont conçu la classe des SFM dans le but de faire des systèmes hybrides *artificiels* et proposent une mise en œuvre directement transposable sur des architectures temps réel, comme nous l'avons montré au sein du chapitre 2. Les SFM sont des systèmes, représentés sous forme d'état, qui sont marqués par une commutation de leur état à des instants discrets. Par ailleurs, l'état d'un SFM présente une évolution continue (ou échantillonnée) entre les instants de commutation à partir d'une condition initiale correspondant à la valeur imposée aux instants de commutation. Selon que l'évolution soit continue ou échantillonnée entre ces instants, nous parlons respectivement de SCM et de SBE. Au cours de ce mémoire, nous avons utilisé le formalisme des SFM pour cons-

truire principalement des architectures d'identification en ligne et de commande temps réel. Toutefois, ce formalisme généralisé nous a également permis d'expliciter synthétiquement plusieurs opérations, notamment la prédiction.

Par la suite (chapitre 3), nous avons entrepris le développement d'une méthode d'identification récursive en ligne de processus linéaires MIMO par une approche boîte noire. Pour ce faire, nous avons utilisé un SFM multimodèle pour construire un *clone* dont les paramètres d'état changent à chaque commutation pour tendre vers ceux du processus. Le clone est un modèle de référence possédant une nature hybride, dans le sens où l'évolution de son état subit une *synchronisation* avec l'état du processus à des instants de commutation rendant l'erreur d'état nulle à ces instants. Par ailleurs, le changement de ses paramètres est régi par un algorithme adaptatif qui utilise comme critère, l'erreur d'état *entre* les instants de commutation. La méthode, appelé *clonage* garantit, après un certain nombre d'itérations (dépendant d'un gain ajustable), la reproduction du *comportement* du processus par le clone (annulation de l'erreur d'état $\forall t \to \infty$) et en même temps, l'adéquation entre les paramètres du clone et celui du processus (identification). Le chapitre apporte, dans un premier temps, un état de l'art sur les méthodes d'identification existantes, en précisant les aspects que nous privilégions. Il donne ensuite une mise en équation de la méthode qui est très facilement transposable sur calculateur numérique (l'algorithme est proposé en annexe III). Plusieurs exemples de simulation sont proposés, démontrant une convergence rapide des estimateurs, même en présence de perturbations, de variations temporelles des paramètres du processus ou de retard connu à son entrée. Une comparaison avec un exemple existant montre que la méthode du clonage permet une accélération de la convergence grâce à la propriété de synchronisation que possède le clone. Enfin, l'application de la méthode du clonage sur un système réel a démontré sa fiabilité et sa robustesse.

Le chapitre 4 est consacré au développement d'architectures de commande temps réel basées sur les SFM. Ce chapitre offre d'abord un état de l'art sur la commande en général et propose l'utilisation du formalisme des SFM pour réaliser des contrôleurs à fonctionnement par morceaux. Selon que l'on se base sur un SLCM ou un SLBE, nous pouvons définir un contrôleur à fonctionnement continu par morceaux ou un contrôleur bi-échantillonné. Dans leur forme de base, ces contrôleurs utilisent un retour d'état du processus à commander pour réaliser une poursuite échantillonnée d'une consigne d'état avec un retard égal à une période de commutation T et avec une coïncidence exacte entre l'état et sa consigne (retardée de T) à chaque instant de commutation. Sans optimisation, un CFM n'utilise le retour d'état qu'aux instants de commutation et peuvent donc facilement admettre un retour d'état échantillonné. Par ailleurs, dans le cas où le retour est continu ou est échantillonné à une période $t_e < T$, il est possible d'envisager l'optimi-

sation d'un CFM pour minimiser l'écart entre l'état et la consigne (retardée de T) même entre les instants de commutation. La commande par l'approche des SFM permet donc d'aboutir à la réalisation d'un suivi de trajectoire en utilisant une représentation d'état du processus à commander. Ce type d'asservissement aboutit à un résultat comparable à la commande *deadbeat* et permet de mettre en œuvre une commande constante par morceaux comme le montrent les exemples.

En plus de la formalisation des contrôleurs, nous proposons, dans ce chapitre :

- une adaptation pour prendre en compte une consigne de *sortie*,
- la mise en œuvre d'un *amortisseur* pour atténuer les dépassements dans le cas d'une consigne présentant des discontinuités,
- l'élaboration d'une structure de commande adaptative combinant le contrôleur par morceaux et l'identificateur en ligne du chapitre 2.

Dans tous les cas, les exemples de simulation et d'implantation sur architecture temps réel permet de conclure sur la robustesse de la commande CFM face aux perturbations telles que les bruits de mesure, la variation des paramètres du processus, voire même la présence de non linéarités dans le comportement du processus.

Les CFM tels que définis au chapitre 4 requiert la disponibilité de l'état complet du processus à l'instant présent. Toutefois, afin de considérer des cas pratiques, nous proposons au cours du chapitre 5 des adaptations du formalisme des CFM pour prendre en compte un retour par capteur introduisant un retard et /ou un échantillonnage sur l'état ou la *sortie* du processus. Nous réalisons alors des prédicteurs en utilisant le formalisme des SFM pour prendre en compte le retard, et proposons des architectures de commande intégrant un observateur pour reconstruire l'état à partir de la sortie du processus (à noter que le formalisme des CFM permet une combinaison directe avec un observateur d'état). Il est évident que le modèle du processus étant sollicités par l'observateur, le prédicteur et le contrôleur, leur combinaison rend la structure globale moins robuste face aux variations des paramètres du processus que dans le cas d'un retour d'état direct. Toutefois, la performance dépendant surtout du prédicteur, les exemples montrent que pour un retards et un échantillonnage liés aux capteurs numériques classiques, la méthode aboutit à une poursuite suffisamment robuste pour entreprendre la commande d'un processus réel.

Perspectives

L'étude des systèmes hybrides au sens large devient de plus en plus courante avec l'avènement de la technologie numérique. Dans ce mémoire, nous avons mis l'accent sur la formalisation des systèmes à fonctionnement par morceaux qui sont très facilement transposables sur calculateur et avons utilisé dans le but de construire des systèmes artificiels.

Nous les avons utilisés pour construire des architectures d'identification, de commande et de prédiction dans un domaine linéaire. Il serait également intéressant de reprendre le formalisme général des SFM pour entreprendre la démarche dans le cas des processus non linéaires.

Par ailleurs, une des perspectives à envisager serait d'utiliser le formalisme des SFM pour *modéliser* des systèmes complexes existants. En utilisant un SFM multimodèle non linéaire, il est possible de modéliser bon nombre de systèmes naturellement hybrides tels que la boîte de vitesse d'un véhicule ou la trajectoire d'un ballon de football pendant un match.

En ce qui concerne l'identification, nous envisageons également le développement d'une méthode basée non pas sur l'état du processus, mais sur sa sortie. Ainsi, l'approche pourra être généralisée à des cas pratiques. De plus, nous envisageons une adaptation du formalisme afin de tenir en compte la présence de perturbation sur les grandeurs mesurées.

Dans le cadre du développement des contrôleurs, une étude de robustesse est prévue. Elle permettra de déterminer théoriquement la marge de tolérance des variations des paramètres du processus.

Enfin, il est à préciser que les méthodes développées au chapitre 5 sont destinées à des applications sur deux projets déjà en cours : « le pendule inverse 2D à retour visuel » et « l'attelage virtuel d'un véhicule esclave à un meneur ». Ces deux projets requièrent une commande utilisant un retour provenant de capteurs numériques immatériels introduisant un retard et un échantillonnage. L'échantillonnage n'étant pas nécessairement régulier, nous envisageons une commutation des SFM non pas périodique, mais déclenché à chaque instant d'échantillonnage du signal de retour.

Annexe I

La plate-forme 2D à retour visuel

Afin de valider les méthodes d'identification en ligne et de commande temps réel développées au cours des chapitres 3 à 5, nous les avons implantées sur une plate-forme réelle que nous décrivons ici. Celle-ci est une adaptation de la maquette du pendule inverse 2D à retour visuel située au LAGIS, bâtiment P2 (USTL).

I.1. Déplacement motorisé d'un chariot

La maquette du pendule inverse 2D est constituée notamment d'une table XY permettant de faire déplacer un chariot sur un plan horizontal (figure I.1). Le déplacement est assuré par deux moteurs de type brushless avec variateurs, pilotés en +/-10V depuis un PC équipé d'une carte dSpace® à travers un amplificateur de puissance. Chaque motoréducteur brushless de 200W est alimenté en 240V mono, offrant un courant maximal de 15A et délivrant un couple nominal de 3.0Nm.

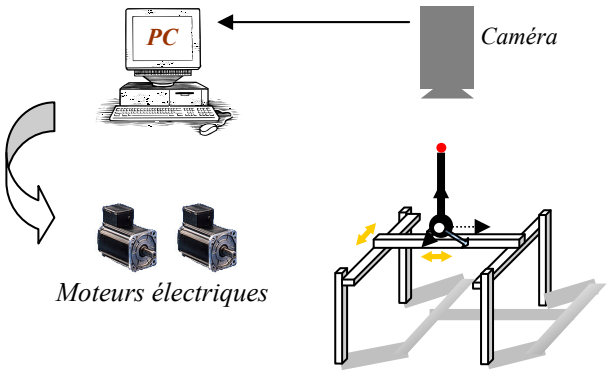


Fig. I.1. *Asservissement visuel du pendule inverse*

En ce qui concerne les essais temps réel des méthodes d'identification et de commande, nous considérons le déplacement du chariot uniquement sur un axe en admettant selon le cas considéré :

- soit un retour d'information par les codeurs de position/vitesse disponible sur le moteur (en pointillés sur la figure I.2),
- soit par l'information de la position du chariot délivrée par un système de vision artificielle[1] (en trait uni sur la figure I.2).

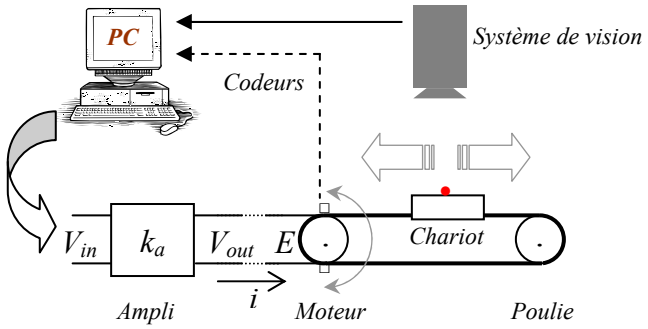


Fig. I.2. *Asservissement du chariot en position*

I.2. Modèle du système à commander

L'ensemble amplificateur-moteur-chariot est approximé à un système linéaire du second ordre défini par le METC suivant :

$$x'(t) = a.x(t) + b.u(t),$$

le vecteur d'état étant : $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} position \\ vitesse \end{bmatrix}$,

les matrices d'état étant : $a = \begin{bmatrix} 0 & 1 \\ 0 & -1/\tau \end{bmatrix}$, $b = \begin{bmatrix} 0 \\ K/\tau \end{bmatrix}$,

τ et K étant respectivement la constante de temps et le gain statique de l'ensemble amplificateur-moteur-chariot.

L'identification des paramètres du processus a donné : $\tau \approx 10\text{ms}$ et $K \approx 2.5\text{m/s/V}$.

Par ailleurs, dans le cas d'un asservissement du chariot en position par les codeurs incrémentaux, nous considérons que ceux-là délivrent l'état.

[1] Le système de vision artificielle est décrit en section I.3.

I.3. Système de vision artificielle

Dans le cadre d'un asservissement visuel du chariot en position, l'information provenant des codeurs incrémentaux est considérée comme étant inaccessible. La seule information disponible sur l'évolution du processus est alors celle délivrée par un système de vision artificielle qui *observe* le déplacement du chariot grâce à une LED infrarouge fixé sur celui-ci (figure I.1).

Le système de vision est constitué d'une caméra CCD doté d'un filtre infrarouge, reliée à un PC équipé d'un logiciel de traitement d'image. Grâce à la caméra située au dessus du chariot, une opération d'extraction d'objet dans la scène permet au système de vision de fournir sous forme échantillonnée et retardée, la position du chariot. Ainsi nous considérons que la sortie du capteur visuel est :

$$z(t) = x_1^*(t - N.t_e),$$

avec (*) : échantillonnage à t_e et N entier positif.

À noter que la période d'échantillonnage t_e correspond à la durée d'une prise d'image et que le retard $N.t_e$ est associé au temps requis pour le traitement d'image(s) en admettant qu'il faut N images pour effectuer un traitement.

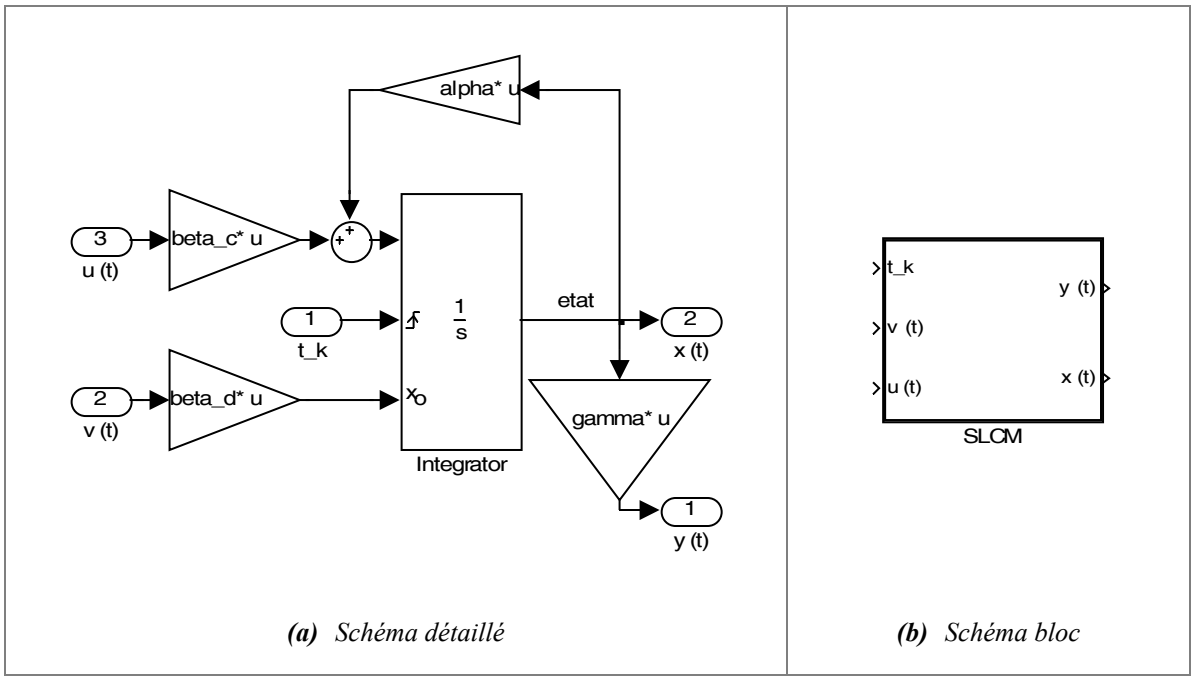
Remarque I.1. *Il est toutefois nécessaire d'effectuer un calibrage préliminaire de la caméra afin de faire correspondre le repère du champ de vision de la caméra à celui de la table XY et de prendre en compte la déformation de la lentille optique. Pour ce faire, un calibrage par la méthode décrite dans* [Tsa87] *est réalisé.*

Annexe II

Réalisation des SFM sous Simulink®

Les SFM présentés en chapitre 2 sont particulièrement adaptés à la technologie temps réel. Utilisant des signaux bornés, leur implantation sur des architectures temps réel par calculateur numérique est immédiatement réalisable par le biais de logiciels adaptés tels que Matlab/Simulink®. Nous décrivons ici comment utiliser les blocs existants de ce logiciel pour construire les SLCM et SLBE.

II.1. Mise en œuvre des SLCM



(a) *Schéma détaillé* ***(b)*** *Schéma bloc*

Fig. II.1. *Schéma de réalisation d'un SLCM sous Matlab/Simulink®*

La figure II.1 illustre la construction d'un SLCM sous Matlab/Simulink®. Le modèle présenté utilise des simples blocs (additionneur, gains matriciels) et un intégrateur continu qui suscite une attention particulière. En effet, pour réaliser la commutation de l'état d'un SLCM, il est nécessaire de paramétrer convenablement l'intégrateur continu. Dans ce sens, nous spécifions dans la fenêtre de configuration du bloc « $1/s$ » :

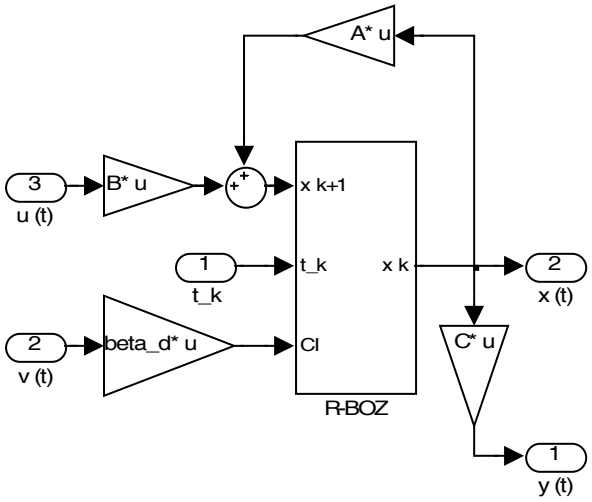
- un mode *reset* qui est déclenché sur front montant/descendant (ou les deux) d'un signal d'horloge noté « t_k » sur la figure II.1 pour forcer la sortie de l'intégrateur à une condition initiale,
- un mode *externe* de prise en compte de la condition initiale à chaque *reset* par le biais d'une entrée supplémentaire.

Ainsi, ce bloc réalise l'intégration de l'entrée continue $u(t)$ du SLCM à partir de la condition initiale[1] donnée par $v(t)$ à chaque front du signal d'horloge.

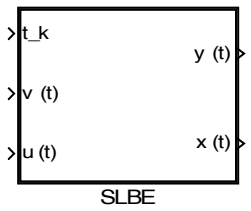
Remarque II.1. *En choisissant par exemple un déclenchement sur front montant du mode reset, il suffit d'injecter dans l'entrée « t_k » (horloge) de l'intégrateur, un signal carré de période T pour réaliser les commutations suivant $S = \{k.T,\ k = 0,1,2,...\}$. Toutefois, dans certains cas, il est nécessaire de déclencher les commutations au moment de l'arrivée des données du capteur numérique, afin d'assurer la synchronisation, notamment lorsque la période d'échantillonnage du capteur n'est pas constante.*

[1] Condition *initiale* définie à chaque commutation (équivalent à une *réinitialisation*).

II.2. Mise en œuvre des SLBE



(a) Schéma détaillé

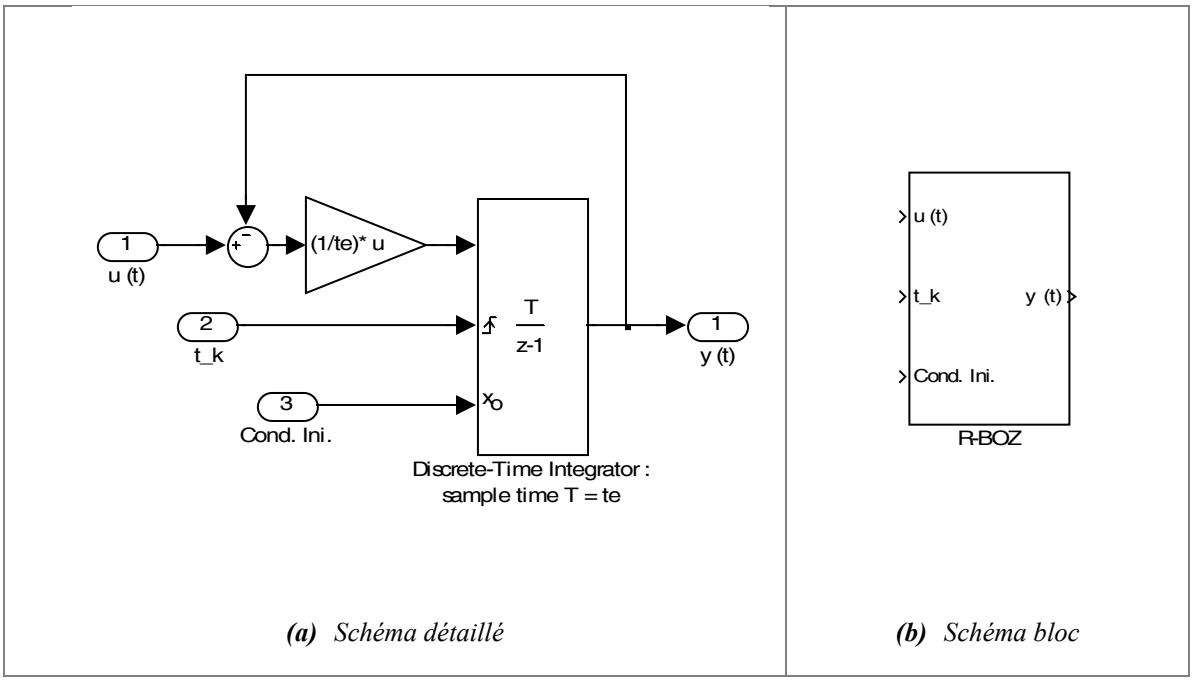


(b) Schéma bloc

Fig. II.2. *Schéma de réalisation d'un SLBE sous Matlab/Simulink®*

La construction sur calculateur d'un SLBE est illustrée par la figure II.2. L'état d'un SLBE évoluant selon une équation aux différences conformément à sa définition (2.25), le schéma fonctionnel (figure II.2a) fait apparaître un bloc R-BOZ à sortie *réinitialisable*. Contrairement à l'intégrateur continu qui permet directement l'utilisation d'un mode *reset*, la bibliothèque de Simulink® ne dispose pas explicitement d'un tel bloc R-BOZ. Il faut donc construire ce bloc à partir d'un intégrateur discret qui admet le mode *reset* souhaité selon le montage donné en figure II.3.

Remarque II.2. *L'intégrateur discret utilisé dans la figure II.3 requiert une période d'intégration nécessairement fixe. Il faut donc faire, en pratique,* $t_k^{i+1} - t_k^i = t_e$ *où* t_e *est une période constante telle qu'il existe* $q_k = (t_{k+1}^0 - t_k^0)/t_e$ *pas d'intégration discrète dans un morceau k.*



(a) *Schéma détaillé* ***(b)*** *Schéma bloc*

Fig. II.3. *Schéma de réalisation d'un R-BOZ sous Matlab/Simulink®*

La mise en équation de la sortie $y(t)$ en fonction de son entrée $u(t)$ montre que le R-BOZ réalise un retard-blocage à période t_e entre deux instants *reset* (i.e. sur un morceau *k*) de l'entrée $u(t)$:

$$y_k^{i+1} - y_k^i = t_e.(1/t_e).(u_k^i - y_k^i)$$

$$\Rightarrow \quad y_k^i = u_k^{i-1}, \tag{II.1}$$

en conservant la notation des SLBE pour désigner les deux échelles de temps.

À noter que le bloc génère un signal discrétisé à période t_e sur un morceau *k* même si l'entrée $u(t)$ est continue.

Par ailleurs, la construction du R-BOZ étant basée sur l'intégrateur discret (figure II.3), elle hérite de son fonctionnement en mode *reset*. Ceci permet de réinitialiser la sortie du R-BOZ à une valeur donnée par une entrée externe aux instants (t_k^0) de commutation quand il est utilisé dans la construction d'un SLBE tel que présenté en figure II.2a. Le déclenchement du mode *reset* est semblable à celui de l'intégrateur continu (cf. remarque II.1).

Annexe III

Réalisation du clonage

Basée sur le formalisme des SFM, l'architecture d'identification proposée au chapitre 3 est aisément implantable sur calculateur numérique. Nous proposons ici l'algorithme du clonage ainsi qu'un schéma sous Simulink® illustrant les n blocs adaptatifs et le clone.

III.1. Algorithme

D'après les équations établies au sein du chapitre 3, nous proposons un algorithme de nature hybride. En effet, aux instants de commutation, il doit réaliser la récurrence donnée en (3.14), correspondant à l'aspect *discret*. Par contre, entre les instants de commutation il doit effectuer deux opérations suivantes en *continu* :

- évaluer, selon (3.9), la quantité $\widetilde{W}(t) = \int_{t_k}^{t} W(\theta).d\theta$,
- calculer, les intégrales selon (3.12) et (3.13) dans le but d'évaluer ε_k^i et V_k^i .

L'algorithme du clonage est représenté ci-après.

Début Préambule

n = ordre du procédé ; k = numéro du morceau $]t_k, t_{k+1}]$ *courant.*

L'algorithme fournit, pour chaque morceau, les matrices $\hat{a}_k$ *et* $\hat{b}_k$ *du clone. Ces matrices tendent vers les matrices du procédé, lorsque k tend vers l'infini.*

$x(t)$ *et* $u(t)$*, désignent respectivement l'état du procédé et la commande.*

On note : $W^T(t) = [x^T(t), u^T(t)]$ *et* $e(t) = \hat{x}(t) - x(t)$.

Avant chaque instant de commutation t_{k+1}*, on dispose de :*

$\hat{\varphi}_k^T = (\hat{a}_k, \hat{b}_k)$*, résultant du calcul à l'instant* t_k *ou de l'initialisation pour* $k = 0$*,*

$$\varepsilon_k^i = \int_{\text{morceau } k} \left\{ signe[e^i(\theta)] \right\} . e^i(\theta) . d\theta \quad \text{pour } i = 1, \cdots, n \text{ selon (3.11),}$$

$$V_k^i = \int_{\text{morceau } k} \left\{ signe[e^i(\theta)] \right\} . \tilde{W}^T(\theta) . d\theta \quad \text{pour } i = 1, \cdots, n \text{ avec } \tilde{W}(t) = \int_{t_k}^{t} W(\theta) . d\theta$$

Fin Préambule

Début Algorithme

Faire $k = 0$ *et donner* $\hat{\varphi}_0^T = (\hat{a}_0, \hat{b}_0)$ *(initialisation)*

Tant que fonctionnement, faire :

A l'instant de commutation t_k*, faire :*

$\hat{x}_k = x_k$ *(synchronisation de l'état du clone à celui du processus)*

Pour chaque colonne $\hat{\varphi}_k^i$ *de* $\hat{\varphi}_k$ *(*$i = 1, \cdots, n$*), faire :*

choisir un $g_k^i > 0$

calculer et mémoriser

$$\hat{\varphi}_{k+1}^i = \hat{\varphi}_k^i - \frac{g_k^i . V_k^i . \varepsilon_k^i}{1 + g_k^i . V_k^{iT} . V_k^i} \qquad \text{selon (3.14)}$$

Concaténer les colonnes φ_{k+1}^i *et transposer le résultat pour constituer* $\varphi_{k+1}^T = (\hat{a}_{k+1}, \hat{b}_{k+1})$

Sur le morceau $]t_{k+1}, t_{k+2}]$*, faire :*

Pour chaque composante $e^i(t)$ *de* $e(t)$ *(*$i = 1, \cdots, n$*), évaluer :*

$$\varepsilon_{k+1}^i = \int_{\text{morceau } (k+1)} \left\{ signe[e^i(\theta)] \right\} . e^i(\theta) . d\theta$$

$$V_{k+1}^i = \int_{\text{morceau } (k+1)} \left\{ signe[e^i(\theta)] \right\} . \tilde{W}^T(\theta) . d\theta$$

Faire $k = k + 1$

Fin_Tant que fonctionnement

Fin Algorithme

III.2. Structure fonctionnelle de clonage

En vue d'une réalisation sous Simulink®, nous avons établi selon le principe de clonage donné au chapitre 3, une structure fonctionnelle de clonage (SFC) qui incorpore le clone et le bloc adaptatif (figure III.1).

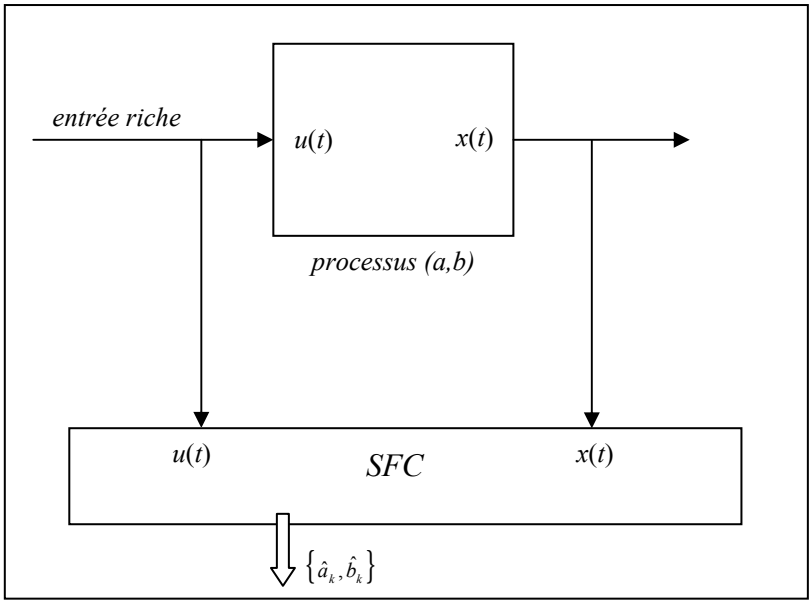


Fig. III.1. *Schéma de réalisation du clonage*

Le SFC utilise comme entrée, l'entrée et l'état du processus. Sa sortie correspond aux estimateurs des paramètres du processus.

La réalisation d'un SFC sous Simulink® est donnée en figure III.2. Ce schéma correspond au clonage d'un processus d'ordre 3.

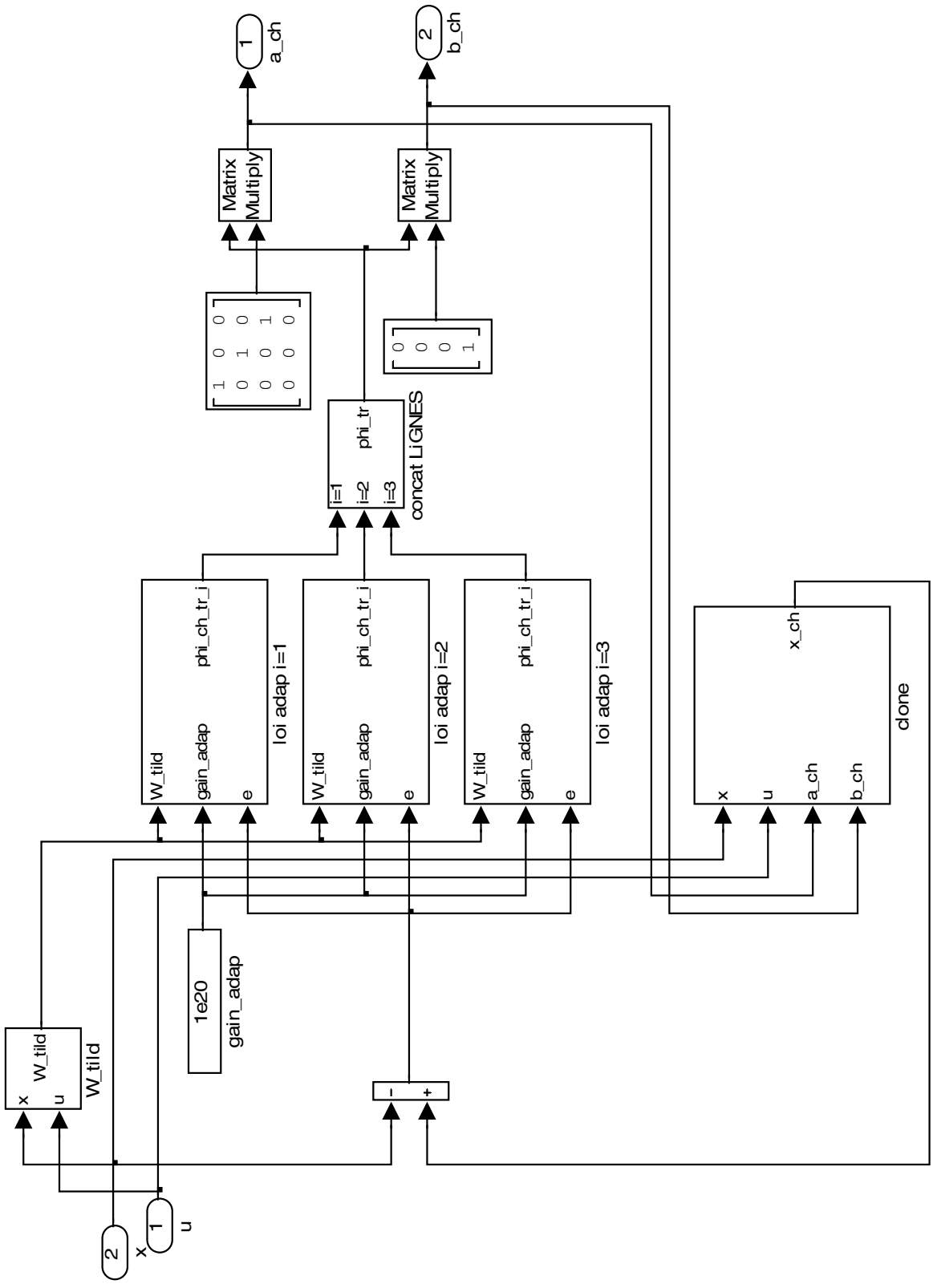


Fig. III.2. *Schéma détaillé d'un SFC d'ordre 3*

La figure fait apparaître 3 blocs adaptatifs distincts correspondant à chacune des lignes de la matrice $\hat{\varphi}^T$ (notée *phi_ch_tr* sur la figure)

Annexe IV

Conditions d'existence de la poursuite

En formalisant, au cours du chapitre 4, la mise en œuvre des contrôleurs à partir des SFM linéaires, nous introduisons des inversions de matrices. Nous donnons, ici, les conditions d'existence.

IV.1. CCM non optimisé

La matrice $M \in \Re^{n\times n}$ définie dans le CCM non optimisé est de la forme $M = e^{a.T}.\widetilde{M}$ avec $\widetilde{M} = \int_0^T e^{-a.\tau}.b.\gamma.e^{\alpha.\tau}d\tau$ en utilisant les notations du chapitre. Il est évident que les conditions d'existence de M^{-1} sont les mêmes que celles de $\widetilde{M}^{-1}$. Pour poser les conditions d'existence, nous notons :

$$e^{-a\tau} = \sum_{j=0}^{n-1} p_j(-\tau).a^j, \tag{IV.1a}$$

$$e^{\alpha\tau} = \sum_{j=0}^{n-1} q_j(\tau).\alpha^j, \tag{IV.1b}$$

où les coefficients réels $p_j(-\tau)$ et $q_j(\tau)$, $(j = 0,...,n-1)$, notés p_j et q_j, constituent les coordonnées de deux vecteurs $P(-\tau)$ et $Q(\tau)$, notés plus simplement P et Q, tels que :

$$P^T = [p_0, p_1,..., p_{n-1}], \tag{IV.2a}$$

$$Q^T = [q_0, q_1,..., q_{n-1}]. \tag{IV.2b}$$

Dans ces conditions, nous pouvons écrire, d'après (IV.1a) et (IV.1b) :

$$\tilde{M} = \int_0^T \left[\sum_{j=0}^{n-1} p_j.a^j \right].b.\gamma.\left[\sum_{j=0}^{n-1} q_j.\alpha^j \right].d\tau \qquad \text{(IV.3)}$$

Ou encore :

$$\tilde{M} = \int_0^T \left[\sum_{j=0}^{n-1} a^j.b.(p_j.I_r) \right].\left[\sum_{j=0}^{n-1} (q_j.I_r).\gamma.\alpha^j \right].d\tau,$$

avec I_r étant la matrice d'identité d'ordre r.

Soit, d'après (IV.2a) et (IV.2b) :

$$\tilde{M} = \int_0^T K.(P \otimes I_r).(Q^T \otimes I_r).\Omega.d\tau, \qquad \text{(IV.4)}$$

Expression dans laquelle $\otimes$ désigne le produit de Kronecker, $K \in \Re^{n \times nr}$ correspond à la matrice de commandabilité de la paire (a,b) et $\Omega \in \Re^{nr \times n}$ représente la matrice d'observabilité de la paire (α,γ).

L'expression (IV.4) peut encore s'écrire comme :

$$\tilde{M} = \int_0^T K.(P.Q^T \otimes I_r).\Omega.d\tau = K.\left[\int_0^T (P.Q^T \otimes I_r) d\tau \right].\Omega, \qquad \text{(IV.5)}$$

En posant $\Pi = \int_0^T P(-\tau).Q^T(\tau) d\tau$ et $\hat{\Pi} = \Pi \otimes I_r$, on obtient :

$$\tilde{M} = K.\hat{\Pi}.\Omega \qquad \text{(IV.6)}$$

Théorème IV.1. *(Condition d'existence d'une commande par le CCM optimisé)*

Soient un processus linéaire MIMO continu tel que présenté dans l'introduction générale (1.1) et un CCM non optimisé basé sur $\Sigma_c(\{k.T\},\alpha,0,\beta_d,\gamma)$. *Si l'on note :*

- $K = \left[b \mid a.b \mid \ldots \mid a^{n-1}.b \right]$, *matrice de commandabilité de la paire* (a,b),

- $\Omega = \left[\gamma^T \mid \alpha^T.\gamma^T \mid ... \mid (\alpha^T)^{n-1}.\gamma^T\right]^T$, *matrice d'observabilité de la paire* (γ,α),
- $\Pi = \int_0^T P(-\tau).Q^T(\tau)d\tau$, *où* $P(t)$ *et* $Q(t)$ *sont respectivement les vecteurs des coordonnées de* $e^{a.t}$ *et* $e^{\alpha.t}$ *sur les* a^i α^i $(i=0,...,n-1)$,
- $\hat{\Pi} = \Pi \otimes I_r$.

Alors, pour que la poursuite échantillonnée, avec un retard d'une période d'échantillonnage, soit réalisable, il faut et il suffit que :

- $Ker(\Omega) = \{0\}$
- $Ker(\hat{\Pi}) \bigcap \mathrm{Im}(\Omega) = \{0\}$
- $Ker(K) \bigcap \mathrm{Im}(\hat{\Pi}.\Omega) = \{0\}$

La matrice β_d *existe, est régulière et est définie par :*

$$(\beta_d)^{-1} = M = e^{a.T} \int_0^T e^{-a.\tau}.b.\gamma.e^{\alpha.\tau}d\tau$$

Démonstration : *Sachant que* $\tilde{M} = K.\hat{\Pi}.\Omega$ *est une matrice carrée (de dimension* $n \times n$*), son inversion est possible si et seulement si* $Ker(K.\hat{\Pi}.\Omega) = \{0\}$.

Pour évaluer cette condition nécessaire et suffisante, nous écrivons :

$$\forall \eta \in Ker(K.\hat{\Pi}.\Omega) \quad \Leftrightarrow$$
$$[\eta \in Ker(\Omega)]\, OU\, \left[\Omega.\eta \in Ker(\hat{\Pi})\right] OU\, \left[\hat{\Pi}.\Omega.\eta \in Ker(K)\right]$$

Ainsi, $Ker(K.\hat{\Pi}.\Omega) = \{0\}$ *est équivalent à :*

$$\forall \eta \neq 0 \Leftrightarrow \left\{ \begin{array}{c} [\eta \notin Ker(\Omega)] \Rightarrow Ker(\Omega) = \{0\} \\ ET\, \left[\Omega.\eta \notin Ker(\hat{\Pi})\right] \Rightarrow Ker(\hat{\Pi}) \bigcap \mathrm{Im}(\Omega) = \{0\} \\ ET\, \left[\hat{\Pi}.\Omega.\eta \notin Ker(K)\right] \Rightarrow Ker(K) \bigcap \mathrm{Im}(\hat{\Pi}.\Omega) = \{0\} \end{array} \right\}$$

La matrice M *étant de même rang que* $\tilde{M}$, *nous avons donc ses conditions d'existence nécessaires pour effectuer la poursuite échantillonnée.*

IV.2. CCM optimisé

IV.2.1. Bloc-triangularisation de H

Selon (4.14), Θ_{12} est un bloc de $e^{\mathrm{H}.T}$. De plus, les matrices G^{-1} et E apparaissant dans H (4.12b) sont symétriques et définies positives. On peut donc les factoriser selon la méthode de Cholesky [RB95] et écrire :

$$G^{-1} = U.U^T \text{ et } E = V^T.V\,, \tag{IV.7}$$

expression dans lesquelles $U \in \Re^{r\times r}$ et $V^T \in \Re^{n\times n}$ sont des matrices triangulaires inférieures régulières, dont les éléments diagonaux sont tous strictement positifs.

Dans ces conditions, en posant $b.U = \tilde{b}$, on peut écrire H sous la forme suivante :

$$\mathrm{H} = \begin{bmatrix} a & \tilde{b}.\tilde{b}^T \\ V^T.V & -a^T \end{bmatrix} \tag{IV.8}$$

H est une matrice hamiltonienne. Elle est donc bloc-triangularisable selon la transformation régulière :

$$\tilde{\mathrm{H}} = \begin{bmatrix} I_n & 0 \\ X & I_n \end{bmatrix}.\begin{bmatrix} a & \tilde{b}.\tilde{b}^T \\ V^T.V & -a^T \end{bmatrix}.\begin{bmatrix} I_n & 0 \\ -X & I_n \end{bmatrix}, \tag{IV.9}$$

expression dans laquelle $X \in \Re^{n\times n}$ est une solution de l'équation de Ricatti symétrique :

$$-X.\tilde{b}.\tilde{b}^T.X + X.a + a^T.X + V^T.V = 0\,. \tag{IV.10}$$

On obtient alors :

$$\tilde{\mathrm{H}} = \begin{bmatrix} L & \tilde{b}.\tilde{b}^T \\ 0 & -L^T \end{bmatrix}, \text{ avec } L = a - \tilde{b}.\tilde{b}^T.X \tag{IV.11}$$

IV.2.2. Interprétation

La transformation (IV.9) appliquée à l'équation (4.12) conduit à la nouvelle équation :

$$\begin{bmatrix} x'(t) \\ \widetilde{\lambda}'(t) \end{bmatrix} = \begin{bmatrix} L & \widetilde{b}.\widetilde{b}^T \\ 0 & -L^T \end{bmatrix} . \begin{bmatrix} x(t) \\ \widetilde{\lambda}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ E \end{bmatrix} \cdot c(t-T) \tag{IV.12a}$$

$$\text{avec } \widetilde{\lambda}(t) = \lambda(t) + X.x(t). \tag{IV.12b}$$

Cette équation conduit à un système de la même structure que le système (4.3) dans lequel on ferait :

$$a = L, \tag{IV.13a}$$

$$\alpha = -L^T, \tag{IV.13b}$$

$$b = \widetilde{b}, \tag{IV.13c}$$

$$\gamma = \widetilde{b}^T. \tag{IV.13d}$$

On a donc, en intégrant (IV.12a), sur le morceau k :

$$x_{k+1} = e^{L.T}.x_k + \widetilde{\Theta}_{12}.\widetilde{\lambda}_k^+ - \hat{\mathrm{I}}_{k+1} \tag{IV.14a}$$

$$\text{avec } \widetilde{\Theta}_{12} = \int_0^T e^{L(T-\tau)}.\widetilde{b}.\widetilde{b}^T.e^{-L^T.\tau}.d\tau \tag{IV.14b}$$

$$\text{et } \hat{\mathrm{I}}_{k+1} = \int_{k.T}^{(k+1).T} e^{L[(k+1).T-\tau]}.E.c(\tau - T).d\tau \tag{IV.14c}$$

Dès lors, on peut énoncer les propositions suivantes :

- Θ_{12} inversible $\Leftrightarrow$ λ_k^+ calculable, d'après (4.15),
- λ_k^+ calculable $\Leftrightarrow$ $\widetilde{\lambda}_k^+$ calculable, d'après (IV.12b),
- $\widetilde{\lambda}_k^+$ calculable $\Leftrightarrow$ $\widetilde{\Theta}_{12}$ inversible d'après (IV.14a).

Donc, Θ_{12} inversible $\Leftrightarrow$ $\widetilde{\Theta}_{12}$ inversible.

IV.2.3. Calcul de $\widetilde{\Theta}_{12}$

On peut écrire :

$$\widetilde{\Theta}_{12} = e^{L.T} . \int_0^T e^{-L.\tau} . \widetilde{b} . \widetilde{b}^T . e^{-L^T . \tau} . d\tau \qquad \text{(IV.15)}$$

Par un raisonnement analogue à celui utilisé pour calculer *M*, dans la section IV.1, on obtient :

$$\widetilde{\Theta}_{12} = e^{L.T} . K . \hat{\Pi} . \Omega \qquad \text{(IV.16a)}$$

avec $$K = \left[\widetilde{b}, L.\widetilde{b}, ..., L^{n-1} . \widetilde{b}\right], \qquad \text{(IV.16b)}$$

$$\Omega = \left[\widetilde{b}, L.\widetilde{b}, ..., L^{n-1} . \widetilde{b}\right]^T = K^T , \qquad \text{(IV.16c)}$$

$$\Pi = \int_0^T P(-\tau) . P^T(-\tau) . d\tau , \qquad \text{(IV.16d)}$$

où $P(t)$ est le vecteur des coordonnées de $e^{L.t}$ sur les L^i $(i = 0, ..., n-1)$,

$\hat{\Pi} = \Pi \otimes I_r$.

Les conditions d'existence de $\widetilde{\Theta}_{12}^{-1}$ s'en déduisent immédiatement, d'après le théorème IV.1 :

- $Ker(\Omega) = \{0\}$,
- $Ker(\hat{\Pi}) \bigcap \mathrm{Im}(\Omega) = \{0\}$,
- $Ker(K) \bigcap \mathrm{Im}(\hat{\Pi} . \Omega) = \{0\}$,

expressions dans lesquelles on a : $\Omega = K^T$.

Théorème IV.2. *(Condition d'existence d'une commande par le CCM optimisé)*

Soient un processus linéaire MIMO continu tel que présenté dans l'introduction générale (1.1) et un CCM optimisé basé sur $\Sigma_c(\{k.T\}, -a^T, -E, \beta_d, G^{-1} . b^T)$, *avec* $G^{-1} = U . U^T$ *et* $E = V^T . V$ *, matrices définies positives. Si l'on note :*

- $\widetilde{b} = b . U$,
- $L = a - \widetilde{b} . \widetilde{b}^T . X$, X *étant solution de l'équation de Riccati symétrique :*
  $-X . \widetilde{b} . \widetilde{b}^T . X + X . a + a^T . X + V^T . V = 0$,

- $K = \left[\widetilde{b}, L.\widetilde{b}, ..., L^{n-1}.\widetilde{b}\right]$,
- $\Pi = \int_T P(-\tau).P^T(-\tau).d\tau$, où $P(t)$ est le vecteur des coordonnées de $e^{L.t}$ sur les L^i $(i = 0,...,n-1)$,
- $\hat{\Pi} = \Pi \otimes I_r$,
- $\mathrm{H} = \begin{bmatrix} a & \widetilde{b}.\widetilde{b}^T \\ V^T.V & -a^T \end{bmatrix}$.

Alors, pour que la poursuite échantillonnée sans oscillations, avec un retard d'une période d'échantillonnage, soit réalisable, il faut et il suffit que :

- $Ker(K^T) = \{0\}$,
- $Ker(\hat{\Pi}) \cap \mathrm{Im}(K^T) = \{0\}$,
- $Ker(K) \cap \mathrm{Im}(\hat{\Pi}.K^T) = \{0\}$,

La matrice β_d est alors régulière et est défini par :

$$\beta_d = \Theta_{12} = \textit{bloc supérieur droit } (n \times n) \textit{ de } e^{\mathrm{H}.T}.$$

IV.3. CBE non optimisé

Selon (4.25b), on peut écrire $N = K.\Omega$ avec :

- $K = \left[f^{q-1}.h \mid f^{q-2}.h \mid \cdots \mid f^0.h\right]$,
- $\Omega^T = \left[(\alpha^T)^0.\gamma^T \mid (\alpha^T)^1.\gamma^T \mid \cdots \mid (\alpha^T)^{q-1}.\gamma^T\right]$.

$N \in n \times n$ est inversible si et seulement si :

$$Ker(K.\Omega) = \{0\}$$

$$\equiv \quad Ker(\Omega) = \{0\} \text{ et } Ker(K) \cap \mathrm{Im}(\Omega) = \{0\}$$

Remarque IV.1. *Si $q = n$, c'est à dire si le nombre de périodes d'échantillonnage t_e entre deux commutations est égal à l'ordre n du système, alors K et Ω^T sont respectivement les matrices de commandabilité du système à commander et d'observabilité du contrôleur. En général, la matrice N^{-1} n'existe pas si $q < n$.*

IV.4. CBE optimisé

Dans le cas où q tend vers l'infini et le produit $q.t_e$ reste fini égal à T, il est possible d'exprimer des conditions géométriques d'existence de Θ_{12}^{-1} comme dans la section IV.2. Dans le cas où q est fini, nous ne disposons actuellement que de la condition d'existence numérique : $\det(\Theta_{12}^{-1}) \neq 0$).

Annexe V

Bibliographie personnelle

[CVK05] A. Chamroo, C. Vasseur and V. Koncar, HAOPI: Hybrid Adaptive Online Plant Identification, *IMACS'05, 17th World Congress on Scientific Computation, Applied Mathematics and Simulation*, Paris, France, July 2005. (http://sab.sscc.ru/imacs2005/papers/T5-I-113-0936.pdf)

[CV05] A. Chamroo and C. Vasseur, Poursuite échantillonnée des systèmes à sortie retardée, *JDMACS'05, Journées Doctorales du GDR MACS (Groupe de Recherche « Modélisation, Analyse et Conduite des Systèmes Dynamiques*), Lyon, France, September 2005.

[CVW05] A. Chamroo, C. Vasseur and H. P. Wang, Plant control using digital sensors that introduce a delayed and sampled output, *ELMA'05, 11th International Conference on Electrical Machines, Drives and Power Systems (Supported by IEEE Bulgaria Section)*, Sofia, Bulgaria, Vol. 1 : 119 – 124, September 2005.

[WCVK06] H. P. Wang, A. Chamroo, C. Vasseur, and V. Koncar, Poursuite échantillonnée à partir de l'état retardé et échantillonné, *CIFA'06, 4e IEEE Conférence International Francophone d'Automatique*, Bordeaux, France, 2006.

[WVCK06] H. P. Wang, C. Vasseur, A. Chamroo and V. Koncar, Sampled tracking for delayed systems using piecewise functioning controller, *CESA'06 IMACS-IEEE Multiconference on Computational Engineering in Systems Applications*, Beijing, China, October 2006.

[CSV+06] A. Chamroo, A. Seuret, C. Vasseur, J.-P. Richard and H. P. Wang, Observing and controlling plants using their delayed and sampled outputs, *CESA'06 IMACS-IEEE Multiconference on Computational Engineering in Systems Applications*, Beijing, China, October 2006.

[CVC06] A. Chamroo, C. Vasseur and N. Christov, Piecewise continuous control of an anaerobic digestion process, *AI'06 International Conference on Automatics and Informatics*, Sofia, Bulgaria, October 2006.

[KWCV06] V. Koncar, H. P. Wang, A. Chamroo and C. Vasseur, Piecewise continuous systems used for control and identification, *Invited Lecture in 3rd International Conference of Applied Mathematics – Included in International Journal of Pure and Applied Mathematics (IJPAM, ISSN 1311 – 8080)*, Plovdiv, Bulgaria, August 2006.

Références bibliographiques

[Abi88] H. Abidin-Tjokronegoro, *Estimation adaptative des paramètres de systèmes variant dans le temps et détection de ruptures*, Ph.D. dissertation, Institut National Polytechnique de Grenoble, Grenoble, France, 1988.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. -H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine, The algorithmic analysis of hybrid systems, *Theoretical Computer Science*, Vol 138 : 3 – 34, 1995

[ACHH93] R. Alur, C. Courcoubetis, T. A. Henzinger and P. -H. Ho, Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems, *Hybrid Systems, Lecture Notes in Computer Science*, Vol. 736 (R. L. Grossman, A. Nerode, A. R. Ravn and H. Rischel, Eds.), Springer-Verlag, New York : 209 – 229, 1993.

[ADG+98] C. Attaianese, A. Damiano, G. Gatto, I. Marongiu and A. Perfetto, Induction motor drive parameters identification, *IEEE Transactions on Power Electronics*, Vol. 13, N° 6 : 1112 – 1122, 1998.

[AH00] K. J. Aström and T. Häglund, The future of PID control, *IFAC Workshop on Digital Control: Past, Present and Future of PID Control*, Terrassa, Spain, April 2000.

[AI78] H. Akashi and H. Imai, Output deadbeat controllers with stability for discrete-time multivariable linear systems, *IEEE Transactions on Automatic Control*, Vol. 23 : 1036 – 1043, December 1978.

[AKZ98] P. Antsaklis, X. Koutsoukos and J. Zaytoon, On hybrid control of complex systems: A survey, *APII-JESA*, Vol. 32, N° 9-10 : 1023 – 1045, 1998.

[AMS58] J. A. Aseltine, A. R. Mancini and C. W. Sarture, A survey of adaptive control systems, *IRE Transactions on Automatic Control*, Vol. 6, N° 1 : 102 – 108, December 1958.

[Ant00] P. J. Antsaklis, A brief introduction to the theory and applications of hybrid sytems, *Proceedings of the IEEE, Special Issue on Hybrid Systems: Theory and Applications*, Vol. 88, N° 7 : 879 – 886, 2000.

[AP99] G. Angelis and P. Philips, Piecewise-constant control for systems with discrete measurements, *Proceedings of the American Control Conference*, San Diego, California : 2965 – 2966, 1999.

[AW95] K. J. Aström and B. Wittenmark, A survey of adaptive control applications, *Proceedings of the 34th IEEE Conference on Decision and Control*, Vol. 1 : 649 – 654, December 1995.

[BBBM05] F. Borrelli, M. Baotić, A. Bemporad and M. Morari, Dynamic programming for constrained optimal control of discrete-time linear hybrid systems, *Automatica*, Vol. 41 : 1709 – 1721, 2005.

[BBM94] M. S. Branicky, V. Borkar, and S. K. Mitter, A unified framework for hybrid control, *Proceedings of the IEEE Conference of Decision Control,* Lake Buena Vista, FL., USA : 4228–4234, December 1994.

[BBM98] M. S. Branicky, V. S. Borkar, S. K. Mitter, A unified framework for hybrid control: Model and optimal control theory, *IEEE Transaction on Automatic Control*, Vol 43, N° 1 : 31 – 45, 1998.

[Bem04] A. Bemporad, Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form, *IEEE Transactions on Automatic Control*, Vol. 49, N° 5 : 832 – 838, 2004.

[Ben96] S. Bennett, A brief history of automatic control, *IEEE Control Systems*, Vol. 16, N° 3 : 17 – 25, June 1996.

[BGM93] A. Back, J. Guckenheimer and M. Myers, A dynamical simulation facility for hybrid systems, *Hybrid Systems, Lecture Notes in Computer Science*, Vol. 736 (R. L. Grossman, A. Nerode, A. R. Ravn and H. Rischel, Eds.), Springer-Verlag, New York : 255 – 267, 1993.

[BH75] A. E. Bryson and Y. -C. Ho, *Applied optimal control*, Hemisphere, New York, 1975.

[BHS96] B. Bishop, S. Hutchinson and M. Spong, Camera modelling for visual servo control applications, *Mathematical Computation and Modelling*, Vol. 24, N° 5/6 : 79 – 102, 1996.

[BL84] A. Bensoussan and J.-L. Lions, *Impulse control and quasi-variational inequalities*, Paris, Gauthier-Villars, 1984.

[Bod45] H. W. Bode, *Network analysis and feedback amplifier design*, Van Nostrand Company, Inc, New York, 1945.

[Bou00] R. Boukezzoula, *Commande floue d'une classe de systèmes non linéaires : application au problème de suivi de trajectoire*, Ph.D. dissertation, Université de Savoie, France, 2000.

[Bra95] M. S. Branicky, *Studies in hybrid systems: Modeling, analysis, and control*, Ph.D. dissertation, Department of Electrical and Computer Engineering, Massachusetts Institute of Technology, 1995.

[Bra98] M. S. Branicky, Multiple Lyapunov functions and other analysis tools for switched and hybrid systems, *IEEE Transaction on Automatic Control*, Vol 43, N° 4 : 475 – 482, 1998.

[BRS98] T. Bastogne, A. Richard, and P. Sibille, Identification des systèmes multivariables : méthodes des sous-espaces. partie 2 : applicabilité et intérêt. *Journal Européen des Systèmes Automatisés*, Vol. 32 : 235 – 365, 1998.

[BS89] D. D. Bainov and P. S. Simeonov, *Systems with impulse effect*, Chichester, UK, Ellis Horwood, 1989.

[BSR98] T. Bastogne, P. Sibille, and A. Richard, Identification des systèmes multivariables : méthodes des sous-espaces. partie 1 : état de l'art, *Journal Européen des Systèmes Automatisés*, Vol. 32 : 207 – 234, 1998.

[BT99] V. D. Blondel, J. N. Tsitsiklis, Complexity of stability and controllability of elementary hybrid systems, *Automatica*, Vol. 35, N° 3 : 479 – 489, 1999.

[Che70] C. T. Chen, *Introduction to linear system theory*, Holt, Rinehart, Winston, New York, 1970.

[CR01] F. J. Carrillo and F. Rotella, Identification en temps réel des paramètres d'un système continu pour l'estimation d'usure d'outils de coupe, *Journées Identification et Modélisation Expérimentale*, Vandoeuvre-les-Nancy, France, March 2001.

[CSS03] J. -H. Chou, J. -H. Sun and J. -N. Shieh, On-line identification and optimal control of continuous-time systems, *Mathematics and Computers in Simulation*, Vol. 63 : 493 – 503, 2003.

[CSV$^+$06] A. Chamroo, A. Seuret, C. Vasseur, J.-P. Richard and H. P. Wang, Observing and controlling plants using their delayed and sampled outputs, *CESA'06 IMACS-IEEE Multiconference on Computational Engineering in Systems Applications*, Beijing, China, October 2006.

[DBL$^+$01] B. Dubuisson, B. O. Bouamama, R. Litwak, D. Maquin, I. Nikiforov, J. Ragot, M. Staroswiecki, and G. Zwingelstein, *Automatique et statistiques pour le diagnostic*, Vol. Systèmes automatisés, Information Commande Communication, Hermès, 2001.

[DBPL00] R. A. Decarlo, M. S. Branicky, S. Peterson and B. Lennartson, Perspective and results on the stability and stabilization of hybrid systems, *Proceedings of the IEEE*, Vol. 88, N° 7 : 1069 – 1082, July 2000.

[Del06] M. De la sen, Robustly stable pole-placement based adaptive control of continuous linear systems with multiestimation, *Communicatons in Nonlinear Science and Numerical Simulation*, Vol. 11 : 233 – 261, 2006.

[DGR95] M. Dambrine, A. Goubet and J. -P. Richard, New results on constrained stabilizing control of time-delay systems, *Proceedings of the 34th Conference on Decision and Control*, New Orleans, LA, December 1995.

[DH81] J. J. D'Azzo, and C. H. Houpis, *Linear control system analysis and design*, McGraw-Hill, New York, 1981.

[DR94] M. Dambrine and J. -P. Richard, Stabilization of constrained nonlinear time-delay systems, *Proceedings of the IMACS Symposium on Mathematical Modelling*, Vienna, Austria, Vol. 3 : 488 – 491, February 1994.

[DRB95] M. Dambrine, J. -P. Richard and P. Borne, Feedback control of time-delay systems with bounded control and state, *Journal of Mathematical Problems in Engineering*, Vol 1, N° 1, 1995.

[DS79] J. C. Doyle and G. Stein, Robustness with observers, *IEEE Tansactions on Automatic Control*, Vol 24 : 607 – 611, August 1979.

[DV97] L. Dugard and E. Verriest (Eds.), *Stability and control of time-delay systems*, Springer-Verlag, New York, 1997.

[DVS93] R. De Callafon, P. Van den Hof and M. Steinbuch, Control relevant identification of a compact disc pick-up mechanism, *Proceedings of the 32nd Conference on Decision and Control*, San Antonio, TX, Vol. 3 : 2050 – 2055, 1993.

[EF82] A. Emami-Naeini and G. F. Franklin, Deadbeat control and tracking of discrete-time systems, *IEEE Transactions on Automatic Control*, Vol. 27, N° 1 : 176 – 181, February 1982.

[EH89] J. Ezzine, A. H. Haddad, Controllability and observability of hybrid systems, *International Journal on Control*, Vol. 49, N° 6 : 2045 – 2055, 1989.

[FH77] T. E. Fortmannand and K. L. Hitz, *An introduction to linear control systems*, Marcel Dekker, New York, 1977.

[FL99] U. Forsell and L. Ljung, Closed-loop identification revisited, *Automatica*, Vol. 35 : 1215 – 1241, 1999.

[GBT99] G. O. Glentis, K. Berberidi, and S. Theodoridis, Efficient least squares adaptive algorithms for FIR transversal filtering, *IEEE Signal Processing Magazine*, Vol. 16 : 13 – 41, 1999.

[GMR03] H. Garnier, M. Mensler and A. Richard, Continuous-time model identification from sampled data. Implementation issues and performance evaluation, *International Journal of Control*, Vol. 76, N° 13 : 1337 – 1357, 2003.

[GNRR93] R. Grossman, A. Nerode, A. Ravn, and H. Rischel, Hybrid Systems, *Lecture Notes in Computer Scien*ce, Vol. 736, H. Rischel, Ed. New York: Springer-Verlag, 1993.

[GSMR96] H. Garnier, P. Sibille, M. Mensler, and A. Richard, Pilot crane identification and control in presence of friction, *The 13th IFAC World Congress*, San Fransisco, California, USA, July 1996.

[Gus00] F. Gustafsson, *Adaptive filtering and change detection*, John Wiley & Sons, Ltd, 2000.

[GY04] H. Garnier and P. Young, Time-domain approaches to continuous-time model identification of dynamical systems from sampled data, *(Technical Report N° 04-CA-086 related to the paper published in) American Control Conference*, Boston, MA (USA), 2004.

[HA88] T. Hagiwara and M. Araki, Design of a stable feedback controller based on the multirate sampling of the plant output, *IEEE Transactions on Automatic Control*, Vol. 33 : 812 – 819, 1988.

[Hal77] J. Hall, *Theory of functional differential equations*, Springer, New York, 1977.

[HCCS02] L. Hu, Y. Cao, C. Cheng and H. Shao, Sampled-data control for time-delay systems, *Journal of the Franklin Institute*, Vol. 339 : 231 – 238, 2002.

[HCK01] W. M. Haddad, V. Chellaboina and N. A. Kablar, Nonlinear impulsive dynamical systems: stability and dissipativity, *International Journal on Control*, Vol. 74 : 1631 – 1658, 2001.

[HCK99] W. M. Haddad, V. Chellaboina and N. A. Kablar, Nonlinear impulsive dynamical systems – Part I: stability and dissipativity, *Proceedings of the 38th Conference on Decision and Control*, Phoenix, Arizona USA, December 1999.

[HKCN05] W. M. Haddad, N. A. Kablar , V. Chellaboina and S. G. Nersesov, Optimal disturbance rejection control for linear impulsive dynamical systems, *Nonlinear Analysis*, Vol. 62 : 1466 – 1489, 2005.

[Hor63] I. M. Horowitz, *Synthesis of feedback systems*, Academic, New York, 1963.

[HSB01] W. P. M. H. Heemels, B. De Schutter and A. Bemporad, Equivalence of hybrid dynamical models, *Automatica*, Vol. 37, N° 7 : 1085 – 1091, 2001.

[HV69] T. C. Hsia and V. Vimolvanich, An on-line technique for system identification, *IEEE Transaction on Automatic Control*, Vol. 14, N° 1 : 92 – 96, 1969.

[Imu04] J. Imura, Optimal control of sampled-data piecewise affine systems, *Automatica*, Vol. 40 : 661 – 669, 2004.

[JKDW01] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied interval analysis with examples in parameter and state estimation, robust control and robotics*, Springer Verlag, 2001.

[Kab03a] N. A. Kablar, Singularly impulsive or generalized impulsive dynamical systems, *Proceedings of the American Control Conference*, Denver, Colorado : 5292 – 5293, June 2003.

[Kab03b] N. A. Kablar, Singularly impulsive or generalized impulsive dynamical systems: Lyapunov and asymptotic stability, *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, USA : 173 – 175, December 2003.

[Kab05a] N. A. Kablar, Optimal control for singularly impulsive dynamical systems, *Proceedings of the American Control Conference*, Portland, OR, USA : 2793 – 2798, June 2005.

[Kab05b] N. A. Kablar, Robust stability analysis of nonlinear uncertain singularly impulsive dynamical systems, *Proceedings of the American Control Conference*, Portland, OR, USA : 7 – 11, June 2005.

[Kab87] P. T. Kabamba, Control of Linear Systems Using Generalized Sampled-Data Hold Functions, *IEEE Transaction on Automatic Control*, Vol. 32, N° 9 : 772 – 783, 1987.

[Kai80] T. Kailath, *Linear systems*, Englewood Cliffs, NJ: Prentice-Hall, 1980.

[KH93] P. T. Kabamba and S. Hara, Worst case analysis and design of sampled-data control systems, *IEEE Transaction on Automatic Control*, Vol. 38 : 1337-1357, 1993.

[KHH01] N. A. Kablar, T. Hayakawa and W. M. Haddad, Adaptive control for thermoacoustic combustion instabilities, *Proceedings of the American Control Conference*, Arlington, VA : 2468 – 2473, 2001.

[KJN$^+$95] W. Kohn, J. James, A. Nerode, K. Harbison and A. Agrawala, Hybrid systems approach to computer-aided control engineering, *IEEE Control Systems Magazine*, Vol. 15, N° 2 : 14 – 25, 1995.

[KKKN01] S. Kim, Y. Kim, H. Kim, and C. Nam, Adaptive reconfigurable flight control system based on recursive system identification, *The 15th International Sessions in the 39th Aircraft Symposium*, Gifu, Japan, October 2001.

[KNR96] W. Kohn, A. Nerode and J. B. Remmel, Continualization: A hybrid systems control technique for computing, *Proceedings of CESA'96 IMACS Multi-conference* : 507 – 511, 1996

[Knu06] M. H. Knudsen, Experimental modeling of dynamic systems: an educational approach, *IEEE Transactions on Education*, Vol. 49, N° 1 : 29 – 38, February 2006.

[KT81] H. Kimura and Y. Tanaka, Minimum-time minimum-order deadbeat regulator with internal stability, *IEEE Transactions on Automatic Control*, Vol. 26 : 1276 – 1282, December 1981.

[KV00] V. Koncar and C. Vasseur, Tracking by compound control, *Studies in Informatics and Control Journal*, Vol. 9, N° 4, December 2000.

[KV01] V. Koncar and C. Vasseur, Systèmes à fonctionnement par morceaux et poursuite échantillonnée, *APII-JESA*, Vol. 35, N° 5 : 665 – 689, 2001.

[KV02] V. Koncar and C. Vasseur, Piecewise functioning systems: bi-sampled controllers, *Studies in Informatics and Control*, Vol. 11, N° 2 : 185-198, 2002.

[KV03] V. Koncar and C. Vasseur, Control of linear systems using piecewise continuous systems, *IEE Control Theory & Applications*, Vol. 150, N° 6 : 565 – 576, 2003.

[LAMK95] W. S. Lee, B. D. O. Anderson, I. M. Y. Mareels and R. L. Kosut, On some key issues in the windsurfer approach to adaptive robust control, *Automatica*, Vol. 31 : 1619 – 1636, 1995.

[Lan88] I. D. Landau, *Identification et commande des systèmes*, Hermès, Paris, 1988.

[Lau72] F. Laurent, Sur la commande d'un filtre linéaire par des impulsions multimodulées, *C.R. Acad. Sc*, Paris, Vol. 270 : 288-289, 1972.

[LBB+01] I. D. Landau, A. Besançon-Voda, G. Besançon, C. Durieu, A. Karimi, M. Namar, T. Poinot, L. Pronzato, H. F. Raynaud, J. Richalet, J. C. Trigeassou, E. Walter, and A. Zolghandi, *Identification des systèmes*, Vol. Systèmes automatisés, Information Commande Communication, Hermès, 2001.

[Len89] B. Lennartson, Sampled-data control for time-delayed plants, *International Journal of Control*, Vol 49 : 1601 – 1614, 1989.

[Lju96] L. Ljung, Development of system identification, *The 13th IFAC World Congress*, San Francisco, California, July 1996.

[Lju99] L. Ljung, *System identification. Theory for the user*, PTR Prentice Hall Information and System Sciences Series (T. Kailath, Series Editor), Upper Saddle River, 2*nd* edition, 1999.

[LLM97] I. Landau, R. Lozano, and M. M'Saad, *Adaptive Control*, Springer Verlag, New York, 1997.

[LM99] D. Liberzon, A. S. Morse, Basic problems in stability and design of switched systems, *IEEE Control Systems*, Vol. 19, N° 5 : 59 – 70, 1999.

[LN73] G. Lüders and K. S. Narendra, An adaptive observer and identifier for a linear system, *IEEE Transactions on Automatic Control*, Vol. 18, N° 5 : 496 – 499, 1973.

[LS83] L. Ljung and T. Söderström. *Theory and practice of recursive identification*. The MIT Press, Cambridge, 1983.

[LTEP96] B. Lennartson, M. Tittus, B. Egardt, and S. Pettersson, Hybrid systems in process control, *IEEE Contr. Syst. Mag.*, Vol. 16, N° 5 : 45 – 55, 1996.

[Lue64] D. G. Luenberger, Observing the state of a linear system, *IEEE Transactions on Military Electronics*, Vol. 8 : 74 – 80, April 1964.

[Lue66] D. G. Luenberger, Observers for multivariable systems, *IEEE Transactions on Automatic Control*, Vol. 11, N° 2 : 190 – 197, April 1966.

[Lue71] D. G. Luenberger, An introduction to observers, *IEEE Transactions on Automatic Control*, Vol. 16, N° 6 : 596 – 602, December 1971.

[Mah00] M. S. Mahmoud, *Robust control and filtering for time-delay systems*, Marcel Dekker, New York, 2000.

[MD05] D. E. Miller and D. E. Davidson, Stabilization in the presence of an uncertain arbitrarily large delay, *IEEE Transactions on Automatic Control*, Vol 50, N° 8 : 1074 – 1089, August 2005.

[Med69] J. S. Meditch, *Stochastic optimal linear estimation and control*, McGraw-Hill, New York, 1969.

[Mer04] G. Mercère, *Contribution à l'identification récursive des systèmes par l'approche des sous-espaces*, Ph.D. dissertation, Université Lille 1, Villeneuve d'Ascq, France, 2004.

[MH99] A. N. Michel and B. Hu, Toward stability theory of general hybrid dynamical systems, *Automatica*, Vol. 35 : 371 – 384, 1999.

[MS02] L. Magni and R. Scattolini, State-feedback MPC with piecewise constant control for continuous-time nonlinear systems, *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA : 4625 – 4630, December 2002.

[NA89] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*, Prentice-Hall, Englewood Cliffs, 1989.

[NI02] G. Nakura and A. Ichikawa, Stabilization of a nonlinear jump system, *Systems & Control Letters*, Vol. 47 : 79 – 85, 2002.

[NK93] A. Nerode and W. Kohn, Models for hybrid systems: automata, topologies, controllability, observability, *Hybrid Systems, Lecture Notes in Computer Science*, Vol. 736 (R. L. Grossman, A. Nerode, A. R. Ravn and H. Rischel, Eds.), Springer-Verlag, New York : 317 – 356, 1993.

[O'Re81] J. O'Reilly, The discrete linear time invariant time-optimal control problem – An overview, *Automatica*, Vol. 17, N° 2 : 363 – 370, 1981.

[ÖG02] M. Östring and S. Gunnarsson, *Recursive identification of physical parameters in a flexible robot arm*, Technical report, Department of Electrical Engineering, Linköping University, Linköping, Sweden, 2002.

[Oku03] H. Oku, Application of a recursive subspace identification algorithm to change detection, *The 13th IFAC Symposium on System Identification*, Rotterdam, The Netherlands, August 2003.

[Öst02] M. Östring, *Identification, diagnosis and control of a flexible robot arm*, Master's dissertation, Department of Electrical Engineering, Linköping University, Linköping, Sweden, 2002.

[PD88] P. Peleties and R. DeCarlo, Modeling of interacting continuous time and discrete event systems: An example, *Proceedings of the 26th Annual Allerton Conference* : 1150 – 1159, September 1988.

[PDR+90] P. Borne, C. Dauphin-Tanguy, J. -P. Richard, F. Rotella and I. Zambetakis, *Commande et optimisation des processus*, Éditions Technip, 1990.

[Pel92] P. A. Peleties, *Modeling and design of interacting continuous-time/discrete event systems*, Ph.D. dissertation, School of Electrical Engineering, Purdue University, West Lafayette, IN, December 1992.

[Pet99] S. Pettersson, *Analysis and design of hybrid systems*, Ph.D. dissertation, Chalmers University of Technology, Goteborg, Sweden, June 1999.

[QL03] S. J. Qin and L. Ljung, Closed loop subspace identification with innovation estimation, *The 13th IFAC Symposium on System Identification*, Rotterdam, The Netherlands, August 2003.

[Rao83] G. P. Rao, *Piecewise constant orthogonal functions and their application to systems and control*, Springer-Verlag, LNCIS-55, 1983.

[RB95] F. Rotella and P. Borne, *Théorie et pratique du calcul matriciel*, Éditions Technip, Paris, 1995.

[RDI05] P. Riedinger, J. Daafouz and C. Iung, About solving hybrid optimal control problems, *IMACS'05, 17th World Congress on Scientific Computation, Applied Mathematics and Simulation*, Paris, France, July 2005.

[Ric01] J. -P. Richard, *Algèbre et Analyse pour l'Automatique*, Collection Information Commande Communication, Éditions Hermès – Lavoisier, 2001.

[Ric03] J. -P. Richard, Time-delay systems: an overview of some recent advances and open problems, *Automatica*, Vol. 39, N° 10 : 1667 – 1694, 2003.

[RP02] V. Răsvan and D. Popescu, Control of systems with input delay by piecewise constant signals, *9th Medit. Conference on Control and Automation*, WM1-B/122, Dubrovnik, Croatia, 2002.

[RPL97] Y. N. Rosenwasser, K. Y. Polyakov and B. P. Lampe, Frequency-domain method for H_2 optimization of time-delayed sampled-data systems, *Automatica*, Vol. 33, N° 7 : 1387 – 1392, 1997.

[SAL96] J. A. Stiver, P. J. Antsaklis, and M. D. Lemmon, A logical DES approach to the design of hybrid control systems, *Mathematical and Computer Modeling*, Vol. 23, N° 11/12 : 55 – 76, June 1996.

[SK01] T. Shiotsuki and A. Kimura, Direct identification method of continuous-time model from time-series, *SICE 1st Annual Conference on control Systems* : 413 – 416, 2001.

[SM65] K. Steiglitz and L. E. McBride, A technique for the identification of linear systems, *IEEE Transactions on Automatic Control*, Vol. 10, N° 4 : 461 – 464, 1965.

[SMRD06] A. Seuret, F. Michaut, J.-P. Richard and T. Divoux, Networked Control using GPS Synchronization, *American Control Conference*, 2006.

[Son81] E. D. Sontag, Nonlinear regulation: The piecewise linear approach, *IEEE Transactions on Automatic Control*, Vol. 26, N° 2 : 346 – 358, 1981.

[Son96] E. D. Sontag, Interconnected automata and linear systems: a theoretical framework in discrete-time, *Lecture Notes on Computer Science*, Hybrid Systems III, R. Alur, T. A. Henzinger and E. D. Sontag Eds, Vol. 1066 : 436 – 448, 1996.

[SR83] D. C. Saha and G. P. Rao, *Identification of continuous systems – A Poisson moment functional approach*, Springer-Verlag, LNCIS-56, 1983.

[SR91] N. K. Sinha and G. P. Rao, *Identification of continuous-time systems*, Kluwer Academic Publishers, Dordrecht, 1991.

[SR93] A. V. B. Subrahmanyam and G. P. Rao, Identification of continuous-time SISO systems via Markov parameter estimation, *IEE Proceedings – D*, Vol. 140, N° 1, January 1993.

[SS89] T. Söderström and P. Stoica, *System identification*, Prentice Hall International Series in Systems and Control Engineering, New York, 1989.

[SSR96] A. V. B. Subrahmanyam, D. C. Saha and G. P. Rao, Irreducible continuous model identification via Markov parameter estimation, *Automatica*, Vol. 32, N° 2 : 249 – 253, 1996.

[Sun04] Z. Sun, Sampling and control of switched linear systems, *Journal of the Franklin Institute*, Vol. 341, N° 7 : 657 – 674, November 2004.

[SW77] A. P. Sage and C. C. White, *Optimum systems control*, NJ: Prentice-Hall, Englewood Cliffs, 1977.

[Tav87] L. Tavernini, Differential automata and their discrete simulators, *Nonlinear Analysis, Theory, Methods, Applications*, Vol. 11, N° 6 : 665 – 683, 1987.

[TD96] K. W. V. To and A. K. David, On-line identification and control of an AC/DC power system, *Electrical Power & Energy Systems*, Vol. 18, N° 4 : 223 – 227, 1996.

[TE98] M. Tittus and B. Egardt, Control Design for Integrator Hybrid Systems, *IEEE Transactions on Automatic Control*, Vol. 43, N° 4 : 491 – 500, April 1998.

[Tho04] J. Thomas, *Estimation et commande prédictive à horizon glissant de systèmes hybrides*, Ph.D. dissertation, Université Paris XI, Orsay, France, 2004.

[Tsa87] R. Y. Tsai, A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, *IEEE Journal of Robotics and Automation*, Vol. RA-3, N° 4: 323 – 344, 1987.

[UB03] T. Uhl and M. Bogacz, Real time modal analysis and its application for flutter testing, *The 13th IFAC Symposium on System Identification*, Rotterdam, The Netherlands, August 2003.

[UN87] S. Urikura and A. Nagata, Ripple-Free Deadbeat Control for Sampled-Data Systems, *IEEE Transaction on Automatic Control*, Vol. 32, N° 6 : 474 – 482, June 1987.

[UR87] H. Unbehauen and G .P. Rao, Identification of continuous systems, *Systems and control series*, North Holland, Amsterdam, 1987.

[UR90] H. Unbehauen and G .P. Rao, Continuous-time approaches to system identification – a survey, *Automatica*, Vol. 26, N°1 : 23 – 25, 1990.

[UR98] H. Unbehauen and G .P. Rao, A review of identification in continuous-time systems, *Annual Reviews in Control*, Vol. 22 : 145 – 171, 1998.

[Vas72] C. Vasseur, *Contribution à l'étude des systèmes échantillonnés commandés par impulsions multimodulées*, Ph.D. dissertation, Université Lille 1, Villeneuve d'Ascq, France, 1972.

[Vas89] C. Vasseur, Réalisation numérique des correcteurs, *Technique de l'Ingénieur*, R7417, April 1989.

[VD96] P. Van Overschee and B. De Moor. *Subspace identification for linear systems, theory, implementation, applications*, Kluwer Academic Publishers, 1996.

[Vib95] M. Viberg. Subspace based methods for the identification of linear time invariant systems, *Automatica*, Vol. 31 : 1835 – 1851, 1995.

[Wit66] H. S. Witsenhausen, A Class of Hybrid-State Continuous-Time Dynamic Systems, *IEEE Transaction on Automatic Control*, Vol. 11, N° 2 : 161 – 167, 1966.

[WJ80] W. T. Wu and L. Y. Juang, On recursive parameter estimation of multivariable systems, *Journal of Chin. Inst. Eng.*, Vol. 3 : 89 – 93, 1980.

[Wol76] W. A. Wolovich, *Linear multivariable systems*, Springer-Verlag, New York, 1976.

[Wol83] W. A. Wolovich, Deadbeat error control of discrete multivariable systems, *International Journal of Control*, Vol. 37 : 567 – 582, 1983.

[Won79] W. M. Wonham, *Linear multivariable control: A geometric approach*, Springer-Verlag, New York, 1979.

[WP94] E. Walter and L. Pronzato, *Identification de modèles paramétriques à partir de données expérimentales*, Masson, Paris, 1994.

[XW03] G. M. Xie and L. Wang, Controllability and stabilizability of switched linear systems, *Systems Control Letters*, Vol 48, N° 2 : 135 – 155, 2003.

[Yam94] Y. Yamamoto, A Function Space Approach to Sampled Data Control Systems and Tracking Problems, *IEEE Transaction on Automatic Control*, Vol. 39, N° 4 : 703 – 713, April 1994.

[YMH98] H. Ye, A. N. Michel and L. Hou, Stability theory for hybrid dynamical systems, *IEEE Transactions on Automatic Control*, Vol 43, N° 4 : 461 – 474, 1998.

[You81] P. Young, Parameter estimation for continuous-time models – a survey, *Automatica*, Vol. 17, N° 1 : 23 – 39, 1981.

[You84] P. C. Young, *Recursive estimation and time series analysis*, Springer Verlag, Berlin, 1984.

[Zab73] J. Zabczyk, Optimal control by means of switching, *Studia Mathematica*, Vol. 65 : 161 – 171, 1973.

[Zay05] M. Zayed, *Véhicules Intelligents : Étude et développement d'un capteur intelligent de vision pour l'attelage virtuel*, Ph.D. dissertation, Université Lille 1, Villeneuve d'Ascq, France, 2005.

[ZBG95] Z. Zang, R. R. Bitmead and M. Gevers, Iterative weighted least-squares identification and weighted LQG control design, *Automatica*, Vol 31 : 1577 – 1594, 1995.

[ZVDL94] Y. Zhu, P. Van Overschee, B. De Moor, and L. Ljung, Comparison of three classes of identification methods, *The 10th IFAC Symposium on System Identification*, Copenhagen, Denmark, July 1994.

Contribution à l'Étude des Systèmes à Fonctionnement par Morceaux :
Application à l'Identification en Ligne et à la Commande en Temps Réel

Ce travail de recherche concerne une approche nouvelle d'identification et de commande de processus réels. Les travaux sont fondés sur une classe particulière de systèmes qui possèdent des propriétés hybrides et qui ont une dynamique caractérisée par un fonctionnement par morceaux. Ces systèmes permettent de développer des outils particulièrement adaptés à une architecture temps réel. Le mémoire consacre un chapitre au concept de système hybride et à l'origine et à la nature des systèmes à fonctionnement par morceaux (SFM). Les autres chapitres fournissent la mise en œuvre théorique et pratique de nouvelles méthodes d'identification et de commande utilisant les SFM et donc adaptées au temps réel. L'identification en ligne, assurée par une méthode appelée « clonage », est régie par un algorithme adaptatif qui garantit une convergence rapide. La commande, quant à elle, vise à réaliser la poursuite échantillonnée d'une trajectoire consigne par l'état d'un système linéaire, même dans le cas où le seul retour possible correspond à l'information provenant d'un capteur numérique qui délivre la sortie du système sous forme retardée et échantillonnée. Chaque méthode est fournie avec une introduction permettant de la situer par rapport à l'existant, une formalisation mathématique et des exemples de simulation et d'implantation temps réel.

Mots clés : *systèmes dynamiques hybrides, systèmes à commutation, systèmes à impulsions, systèmes à fonctionnement par morceaux, identification en ligne, estimation de paramètres, algorithme adaptatif, commande adaptative, poursuite échantillonnée, systèmes à sortie retardée et échantillonnée, commande temps réel*

Studies in Piecewise Continuous Systems:
Application in Online Identification and Real Time Control Fields

This research work deals with a new concept in the field of system identification and control. The common denominator to every proposed method is a particular class of systems that are characterised by some hybrid properties and by a piecewise functioning nature. These systems are particularly useful in designing tools for real time control. The present dissertation proposes a chapter on the concept of hybrid systems and the origin and nature of the piecewise functioning systems (PFS). The remaining chapters detail the theoretical and practical explanation concerning new methods of identification and control based on PFS, and thus compatible with real time application. The online identification, which is carried out by a "cloning" method, is ensured by an adaptive algorithm that enables fast convergence. As for the control command, it is designed to perform sampled tracking of a given trajectory by the state of a linear plant, even in the case where the only available feedback comes from a digital sensor delivering the output of the plant in a delayed and sampled format. For each method, the reader can find a state of the art introduction, a mathematical description and examples coming from computer simulations and real time applications.

Keywords: *dynamic hybrid systems, switched systems, impulsive systems, piecewise functioning systems, online identification, parameter estimation, adaptive algorithm, adaptive control, sampled tracking, delayed and sampled systems, real time control*