Università degli studi di Cagliari
Facoltà di Ingegneria
Dipartimento di Ingegneria Elettrica ed Elettronica
Corso di Laurea Specialista in Ingegneria Elettronica

Tesi di laurea

Controllo anti-windup per sistemi a commutazione

Contrôle anti-windup pour systèmes à commutation

Ivan Malloci

Relatori

Prof. Alessandro Giua – Università degli Studi di Cagliari
Prof. Jamal Daafouz – Institut Nationale Polytechnique de Lorraine
Prof. Claude Iung – Institut Nationale Polytechnique de Lorraine

A.A. 2005/2006

Ringraziamenti

Avendo svolto la tesi durante il soggiorno Erasmus, nei laboratori del Centre de Recherche en Automatique en Nancy dell'École Nationale Supérieure d'Électricité et Mécanique – Institut Nationale Polytechnique de Lorraine a Nancy (France), desidero ringraziare l'università che mi ha ospitato e in particolare i professori Jamal Daafouz et Claude Iung che mi hanno seguito e aiutato nella realizzazione di questo lavoro. Vorrei ringraziare anche il professor Michel Dufaut e la signora Céline Morville che mi hanno aiutato in tutto ciò che concerne l'aspetto burocratico dell'Erasmus.

Inoltre vorrei ringraziare il professor Alessandro Giua, del Dipartimento di Ingegneria Elettrica ed Elettronica dell'Università degli Studi di Cagliari, che mi ha seguito sia durante il mio soggiorno a Nancy che al mio rientro in Sardegna durante la stesura definitiva della tesi.

Index

Sommaire

En étant donné un système composé de différents processus en commutation, le premier problème abordé dans ce travail a été celui-ci de trouver un contrôleur, dit contrôleur nominal, ayant la capacité de commuter lui-même en fonction du processus pour rendre le système asymptotiquement stable. On a choisi d'utiliser des méthodes de résolution basés sur la technique des inégalités matriciels linéaires (LMI: Linear Matrix Inequalities) car il existe des outils numériques pour résoudre ce type de problèmes.

La stabilisation du système par un contrôleur à commutation peut causer des incovénient cependant. Quand une commutation arrive entre deux différents sous-systèmes aussi le contrôleur nominal commute de manière que le système reste stable. Cela peut porter à des changements soudains sur la commande, c'est-à-dire des discontinuités qui sont à l'origine de la dégradation des performances. Naturellement, ces discontinuités sont indésirables et il faut minimiser leur effet.

Ce problème, dit du « bumpless transfer », est plus difficile à résoudre car en littérature on ne trouve pas beaucoup d'articles qui traitent le cas des systèmes linéaires à commutation. Les paramètres du contrôleur « bumpless transfer » aussi seront déterminés à travers les LMI à cause des motivations données en précédence.

Le travail se décompose en deux parties: la première concerne la bibliographie et décrit les méthodes permettant de construire les contrôleurs. On a choisi de traiter le cas de l'anti-windup classique avant tout, qu'il dénote le problème des modifications à une loi de contrôle stabilisant pour un processus avec des entrées illimitées dédiés au recouvrement de la stabilité et de la performance aussi en présence de limites sur l'entrée, pour comprendre les phénomènes sous examen et les solutions qui est possible d'adopter dans ce cas en avant de se poser les mêmes problèmes pour les systèmes à commutation, certainement plus compliqués. En effet un lien existe entre les méthodes classiques et les systèmes à commutation. A titre d'exemple, dans le cas d'une saturation sur la commande, on constate trois modes de fonctionnement selon que la saturation est active avec valeur positive, active avec valeur négative ou non active (fonctionnement linéaire). Le passage du mode de fonctionnement linéaire au mode de fonctionnement saturation à valeur positive peut être vu comme une commutation. Cependant, les solutions classiques ne peuvent être utilisées en l'état pour les systèmes à commutation et il faut trouver une autre solution.

Dans la deuxième partie, qui constitue la contribution originale de ce travail, on va développer des approches pour la synthèse de contrôleurs « bumpless transfer » dont est testée l'efficacité à travers des exemples numériques en Matlab. Trois différentes variantes ont été réalisées pour le contrôleur « bumpless transfer »: les premier deux se basent sur un contrôleur statique au retour d'état pendant que la troisième est basé sur un contrôleur dynamique.

Sommario

Essendo questo lavoro interamente presentato in francese, si è pensato che potessero essere utile un sommario e un'introduzione anche in italiano.

Dato un sistema a commutazione, cioè composto da differenti sottosistemi tra i quali è possibile commutare, il primo problema affrontato in questa tesi è stato quello di sintetizzare un controllore, detto controllore nominale, che avesse la capacità di commutare in funzione del processo per rendere asintoticamente stabile il sistema. Si è scelto di utilizzare dei metodi di risoluzione basati sulla tecnica delle diseguaglianze matriciali lineari (LMI: Linear Matrix Inequalities) in quanto esistono diversi programmi di risoluzione numerica per risolvere questo tipo di problemi.

La stabilizzazione del sistema tramite un controllore a commutazione può però causare degli inconvenienti. Quando avviene una commutazione tra due diversi sottosistemi commuta anche il controllore nominale in modo che il sistema rimanga stabile. Questo può portare a dei cambiamenti improvvisi sul segnale di controllo, ovvero delle discontinuità che sono all'origine della degradazione delle prestazioni. Naturalmente tali discontinuità sono indesiderate e occorre minimizzarle.

Questo problema, detto del "bumpless transfer", risulta di più difficile soluzione rispetto al precedente in quanto in letteratura si trovano ben pochi articoli che trattano il caso dei sistemi lineari a commutazione. Anche i parametri del controllore "bumpless transfer" saranno determinati tramite LMI per le motivazioni esposte in precedenza.

Il lavoro si compone fondamentalmente di due parti: la prima parte, composta dai primi tre capitoli, è bibliografica e descrive i metodi che permetteranno di sintetizzare i controllori. Si scelto di trattare innanzitutto il caso dell'anti-windup classico, che denota il problema delle modifiche ad una legge di controllo stabilizzante per un processo con ingressi illimitati volte al recupero della stabilità e della prestazione anche in presenza di limiti sull'ingresso, per comprendere i fenomeni sotto esame e le soluzioni che è possibile adottare in questo caso prima di porsi le stesse questioni per i sistemi a commutazione, certamente più complicati. Infatti esiste un legame tra i metodi classici e i sistemi a commutazione. A titolo d'esempio, nel caso di una saturazione sul segnale di controllo, si constatano tre modi di funzionamento a seconda che la saturazione sia attiva con valore positivo, attiva con valore negativo o non attiva (funzionamento lineare). Il passaggio dal modo di funzionamento lineare al modo di funzionamento a saturazione con valore positivo, ad esempio, può essere visto come una commutazione. Queste soluzioni classiche non sono applicabili in caso di sistemi a commutazione e occorre dunque trovare altri approcci.

Nella seconda parte, che costituisce il contributo originale della tesi, vengono sviluppati alcuni approcci per il progetto di controllori "bumpless transfer" di cui viene testata l'efficacia attraverso degli esempi numerici in ambiente Matlab. Sono state realizzate tre diverse varianti per il controllore "bumpless transfer": le prime due si basano su un controllore statico a ritorno di stato mentre la terza è basata su un controllore dinamico.

Introduction

En étant donné un système composé de différents processus en commutation, le premier objectif de ce travail est de trouver un contrôleur ayant la capacité de commuter lui-même en fonction du processus pour rendre le système asymptotiquement stable.

Il existe beaucoup de papiers qui traitent ce problème. On appellera le contrôleur ainsi trouvé contrôleur nominal. On a choisi d'utiliser des méthodes qui s'appuient sur les LMI (Linear Matrix Inequalities) car il existe des outils numériques puissants basés sur l'optimisation convexe pour résoudre ce type d'inégalités matricielles.

Cependant, un autre problème apparaît quand on fait commuter le processus et le contrôleur nominal. Des changements brusques sur la commande sont à l'origine de dégradations des performances. Naturellement, ces discontinuités sont indésirables et il faut minimiser leur effet.

Ce problème, dit du « bumpless transfer », est plus difficile à traiter. On trouve beaucoup de travaux dans le cas linéaire classique mais peu d'articles l'abordent dans la littérature pour les systèmes à commutation. Les paramètres du contrôleur « bumpless transfer » aussi seront déterminés à travers les LMI à cause des motivations données en précédence.

Dans ce mémoire, on travaillera toujours avec des systèmes à commutation en temps discret.

Le travail se décompose en deux parties: la première concerne la bibliographie et décrit les méthodes permettant de construire les contrôleurs. Dans la deuxième partie, on va adapter ces contrôleurs pour le problème qui nous intéresse ici et on essayera sur des exemples leur efficacité. On utilisera pour cela le logiciel Matlab et la boite à outils LMItool.

Dans le chapitre 1 on présente une structure unifiée permettant de résoudre le problème d'anti-windup classique pour systèmes LTI (linear time-invariant) en présence d'éléments non-linéaires. On donne ensuite des exemples d'application de cette méthode en temps continu. On s'intéresse au problème d'anti-windup classique pour comprendre les phénomènes en jeu et les solutions adoptées dans ce cas avant de se poser la question pour les systèmes à commutation. Un lien pourrait être fait entre les méthodes classiques et les systèmes à commutations. A titre d'exemple, dans le cas d'une saturation sur la commande, on constate trois modes de fonctionnement selon que la saturation est active avec valeur positive, active avec valeur négative ou non active (fonctionnement linéaire). Le passage du mode de fonctionnement linéaire au mode de fonctionnement saturation à valeur positive peut être vu comme une commutation. Cependant, les solutions classiques ne peuvent être utilisées en l'état pour les systèmes à commutation et il faut trouver une autre solution.

Dans le chapitre 2 on décrit une méthode permettant de calculer un correcteur à commutation en temps discret basé sur un observateur qui garantit la stabilité asymptotique du système à commutation. Les équations de l'observateur sont celles d'un observateur type Luenberger. La synthèse de ce correcteur est basée sur les LMIs.

Dans le chapitre 3 on expose une méthode permettant de calculer un contrôleur robuste en temps discret soit statique soit dynamique qui stabilise le système en boucle fermée tout en satisfaisant une condition de γ-performance. Ce dernier concept nous fournit une borne supérieure pour le rapport entre la sortie contrôlée et une entrée externe jouant le rôle d'une perturbation. En minimisant γ, on minimise l'effet de cette perturbation sur la sortie contrôlée. Ce type de problème d'optimisation nous sera très utile dans la partie synthèse d'un correcteur type « bumpless ».

Le chapitre 4 décrit comment on arrive à la structure finale du contrôleur. On fait la synthèse d'un observateur qui rend asymptotiquement stable le système pour n'importe quelle loi de commutation, c'est-à-dire le contrôleur nominal. Ensuite, on aborde le calcul du contrôleur « bumpless ». Pour ce dernier, on va voir trois méthodes: les deux premières se basent sur un contrôleur statique par retour d'état tandis que le troisième se base sur un contrôleur dynamique. On a décidé de réaliser un deuxième contrôleur statique et un contrôleur

dynamique car le premier contrôleur conçu demande la connaissance de l'état réel du système et dans la réalité ça n'est pas réalisable.

Dans le chapitre 5 on a réalisé des simulations numériques pour illustrer l'approche. On teste toutes les versions avec et sans bruit.

Les annexes A et B expliquent les bases de la théorie des LMIs et la syntaxe de l'outil LMItool de Matlab respectivement tandis que l'annexe C illustre le code Matlab utilisé pour mettre en place l'algorithme de simulation.

La contribution originale de ce travail consiste dans l'étude d'une structure de contrôleur « bumpless ». Les pistes explorées ne donnent pas toutes des résultats satisfaisants. En effet, on a eu beaucoup de difficultés à trouver rapidement une solution à un problème assez compliqué et pour lequel il n'existe presque rien dans la littérature. On a opté pour la synthèse de contrôleurs avec des techniques LMI et pour l'optimisation afin de limiter les sauts. Des structures différentes ont été essayées avant de trouver une structure satisfaisante décrite dans le chapitre 4.

Introduzione in italiano

Dato un sistema a commutazione, cioè composto da differenti sottosistemi tra i quali è possibile commutare, il primo problema affrontato in questa tesi è stato quello di sintetizzare un controllore che avesse la capacità di commutare in funzione del processo per rendere asintoticamente stabile il sistema. Nel seguito si suppone che le commutazioni del sistema siano note in tempo reale. Il controllore così trovato, che sarà composto da tanti sottosistemi quanti sono quelli del processo, sarà chiamato *controllore nominale.*

Una volta trovato quest'ultimo il problema della stabilizzazione del sistema può considerarsi risolto. Essendo ampiamente trattata in letteratura, la sintesi del controllore nominale non prevede lo sviluppo di nuove teorie, bensì la scelta di una strategia di risoluzione tra le tante disponibili. Si è deciso di utilizzare il metodo esposto nell'articolo [4].

Si è scelto di utilizzare dei metodi di risoluzione basati sulla tecnica delle disuguaglianze matriciali lineari (LMI: Linear Matrix Inequalities) in quanto esistono diversi programmi di risoluzione numerica basati sull'ottimizzazione convessa per risolvere questo tipo d'ineguaglianze matriciali e, per risolvere gli LMI, di utilizzare il toolbox LMI di Matlab.

In effetti l'utilizzo degli LMI porta anche degli svantaggi, primo fra tutti il fatto che essi sono basati su condizioni sufficienti ma non necessarie, dunque è possibile che vi sia una soluzione al problema anche se quest'ultimo non è risolvibile tramite LMI.

La stabilizzazione del sistema tramite un controllore a commutazione può però causare degli inconvenienti. Quando avviene una commutazione tra due diversi sottosistemi commuta anche il controllore nominale in modo che il sistema rimanga stabile. Questo può causare dei cambiamenti improvvisi sul segnale di controllo, ovvero delle discontinuità che sono all'origine della degradazione delle prestazioni. Naturalmente tali discontinuità sono indesiderate e occorre minimizzarle.

In particolare, come si evince dalle simulazioni, le discontinuità sul segnale di controllo possono portare a dei transitori più o meno marcati sullo stato continuo del sistema, e dunque sull'uscita misurata, in seguito alle commutazioni.

Questo problema, detto del "bumpless transfer" (che in italiano può tradursi come "commutazione senza salti"), risulta di più difficile soluzione rispetto al precedente (il problema della stabilizzazione) in quanto in letteratura si trovano diversi lavori che trattano il caso lineare ma pochi articoli che si occupano invece dei sistemi a commutazione [5]. Anche i parametri del *controllore "bumpless transfer"* saranno determinati tramite LMI per le motivazioni esposte in precedenza.

Benché tali approcci possano essere applicati sia a modelli a tempo continuo che a tempo discreto, in questo lavoro si è preferito trattare esclusivamente la seconda classe di modelli. Ciò in quanto la tesi intende essere il primo passo di uno studio di più vasta portata che risolva il problema trattato nel caso reale di un'azienda siderurgica che richiede, appunto, dei controllori che lavorino in tempo discreto.

Il lavoro si compone fondamentalmente di due parti: la prima parte, composta dai primi tre capitoli, è bibliografica e descrive i metodi che permetteranno di sintetizzare i controllori. Nella seconda parte, che costituisce il contributo originale della tesi, vengono invece sviluppati alcuni approcci per il progetto di controllori "bumpless transfer" di cui viene testata l'efficacia attraverso degli esempi numerici in ambiente Matlab.

Nel capitolo 1 viene presentata una metodologia generale atta a risolvere il problema dell'anti-windup classico per sistemi LTI (linear time-invariant) in presenza di elementi non lineari. Esso denota il problema delle modifiche ad una legge di controllo stabilizzante per un processo con ingressi illimitati volte al recupero della stabilità e della prestazione anche in presenza di limiti sull'ingresso. In seguito si danno degli esempi di applicazione del metodo su differenti tecniche anti-windup in tempo continuo.

Ci si interessa al problema dell'anti-windup classico per comprendere i fenomeni sotto esame e le soluzioni che è possibile adottare in questo caso prima di porsi le stesse questioni per i sistemi a commutazione, certamente più complicati. Infatti esiste un legame tra i metodi classici e i sistemi a commutazione. A titolo d'esempio, nel caso di una saturazione sul segnale di controllo, si constatano tre modi di funzionamento a seconda che la saturazione sia attiva con valore positivo, attiva con valore negativo o non attiva (funzionamento lineare). Il passaggio dal modo di funzionamento lineare al modo di funzionamento a saturazione con valore positivo, ad esempio, può essere visto come una commutazione. Queste soluzioni classiche non sono applicabili in caso di sistemi a commutazione e occorre dunque trovare altri approcci.

Nel capitolo 2 viene descritto un metodo per progettare un controllore a commutazione a tempo discreto basato su un osservatore che garantisca la stabilità asintotica del sistema a commutazione. Le equazioni dell'osservatore sono quelle di un osservatore di tipo Luenberger. La sintesi del controllore è basata sugli LMI.

Nel capitolo 3 si espone invece un metodo che consente di progettare un controllore robusto a tempo discreto, sia statico che dinamico, che stabilizzi il sistema a ciclo chiuso soddisfacendo delle condizioni γ-performance. Quest'ultimo concetto ci fornisce un limite superiore per il rapporto tra il modulo dell'uscita che determina le prestazioni e il modulo dell'entrata esterna del sistema che assume il ruolo di disturbo. Dunque minimizzando la grandezza γ si minimizza l'effetto del disturbo sulla suddetta uscita. Tale tipo di ottimizzazione risulterà molto utile durante la sintesi del controllore "bumpless transfer".

Il capitolo 4 descrive i passi necessari per arrivare alla struttura finale del controllore. In primo luogo si progetta un osservatore che rende asintoticamente stabile il sistema per qualsiasi valore del segnale di commutazione, cioè il controllore nominale. In seguito si passa alla sintesi del controllore "bumpless transfer", del quale sono state pensate tre differenti varianti: le prime due si basano su un controllore statico a ritorno di stato mentre la terza è basata su un controllore dinamico.

La prima variante, *il controllore statico*, utilizza in retroazione lo stato reale del processo, dunque in realtà non è realizzabile. Per ovviare a questo problema, nella seconda variante presentata, il *controllore statico con osservatore*, si è aggiunto un secondo osservatore che a partire dall'uscita del sistema produce una stima dello stato continuo del sistema. In questo caso il sistema può essere fisicamente realizzato. Il terzo metodo prevede invece un *controllore dinamico*.

Nel capitolo 5 sono presentate diverse simulazioni numeriche in modo da esplicare il funzionamento del sistema: sono testati tutti e tre i controllori con e senza disturbi esterni.

Dai risultati si evince come i due controllori statici funzionino in modo identico in assenza di disturbi esterni mentre, se vengono aggiunti questi ultimi, le prestazioni del controllore che non utilizza lo stato reale in retroazione peggiorano in quanto l'osservatore che deve riprodurre lo stato sarà soggetto ai disturbi. Nonostante questo problema anch'esso si comporta abbastanza bene sino a valori ragionevoli dei disturbi e riesce quindi a limitare i salti sul segnale di controllo dovuti alle commutazioni del controllore. Da notare che tutti e due i correttori statici forniscono al sistema una robustezza ai disturbi molto maggiore di quella posseduta dal sistema con il solo controllore nominale.

Per quanto riguarda invece il controllore dinamico, dalle simulazioni si può notare come i risultati ottenuti non siano affatto sufficienti per affermare che esso funzioni in maniera corretta. Si è deciso di presentarlo ugualmente in quanto dal punto di vista teorico risulta una valida alternativa al controllore statico e inoltre presenta vasti margini di miglioramento.

L'appendice A espone sinteticamente la teoria di base degli LMI, l'appendice B spiega la sintassi del tool LMI di Matlab mentre l'appendice C illustra i listati dei programmi sviluppati in ambiente Matlab per realizzare l'algoritmo di simulazione del controllore.

Il contributo originale di questo lavoro consiste nello studio di una struttura per il controllore "bumpless transfer". Non tutte le strade esplorate hanno dato dei risultati soddisfacenti. In effetti abbiamo avuto non poche difficoltà nel trovare rapidamente una soluzione a un problema che si presenta come molto complicato e per il

quale non esiste quasi niente in letteratura. Si è optato per la sintesi dei controllori tramite tecniche LMI e per l'ottimizzazione al fine di limitare i salti.

La necessità di utilizzare un criterio di ottimizzazione deriva dal fatto che un semplice controllore che stabilizzasse il sistema non sarebbe stato sufficiente in quanto soggetto a transitori più o meno marcati. Inoltre in questo modo si andrebbero a modificare indiscriminatamente le prestazioni del sistema mentre lo scopo del controllore "bumpless transfer" è quello di limitare esclusivamente i salti dovuti alle commutazioni del controllore.

Per quanto riguarda gli sviluppi futuri, essi sono senza dubbio numerosi in quanto il problema del "bumpless transfer" risulta sempre d'attualità a causa dell'importanza che hanno assunto i sistemi ibridi. Naturalmente questo lavoro rappresenta solo un piccolo passo verso la definitiva risoluzione del problema anche se in certi casi rappresenta una soluzione soddisfacente.

Notations

$\Re$ est l'ensemble des nombres réels

pour une matrice A, A^* dénote la matrice complexe conjuguée transposée, A^T sa transposée et A^{-1} son inverse, s'il existe

$A>0$ ($A \geq 0$) signifie que A est une matrice (semi-)définie positive, c'est à dire que $u^T Au > 0 \quad \forall u \in \Re^n$

$G(s) = C(sI - A)^{-1} B + D = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ dénote la représentation d'une matrice de transfert en termes d'espace d'état

le symbole * dans une matrice dénote le terme symétrique transposé

L_2 est l'espace de Hilbert, avec produit scalaire $< x \mid y >= \int_{-\infty}^{\infty} x(t)^* y(t) dt$ et tel que $\|x\|_2 =< x \mid x >^{\frac{1}{2}} < \infty$

$\forall x \in L_2$

Chapitre 1

Principe d'Anti-Windup

Dans les paragraphes 1.1 et 1.2 on présente une structure unifiée permettant de résoudre le problème d'anti-windup classique pour systèmes LTI (linear time-invariant) [10] tandis que dans les paragraphes 1.3.1 [1,7], 1.3.2 [6] et 1.3.3 [8,9] on donne des exemples d'application de la méthode avec différentes techniques anti-windup.

1.1. Une structure generale pour le problème du « Anti-Windup Bumpless Transfer »

Dans le cadre des systèmes de contrôle, il peut arriver qu'un contrôleur fonctionne très bien pour une certaine gamme de valeurs du signal de référence alors les performances se trouvent considérablement détériorées en cas de signaux de référence qui sortent même de peu de cette gamme. Dans beaucoup de cas, la sortie d'un contrôleur n'attaque pas directement le système mais passe par des dispositifs de limitation d'amplitude (saturation) ou par d'autres non-linéarités.

Un phénomène appelé « windup » en anglais apparaît si aucune précaution n'est prise lors de la synthèse du correcteur. L'origine de ce phénomène est le fait que la commande calculée par le correcteur est différente de celle appliquée réellement au système. Ainsi, le contrôleur continue à intégrer l'erreur et à fournir une grande valeur de commande malgré le fait que cette commande dépasse la limitation. Le résultat est l'apparition de phénomènes oscillatoires très marqués ou, parfois, de réponses divergentes.

Dans la *fig.1* on peut voir le schéma idéal et linéaire du cas où on veut rendre petit l'erreur *e* en dépit des changements de la commande *r* ou de la perturbation *d*:

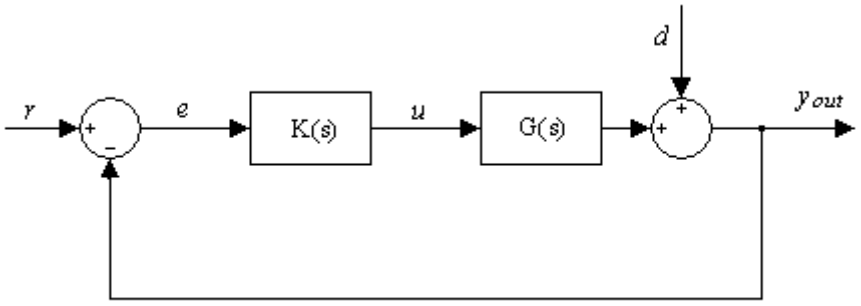


Figure 1 : schéma du cas idéale linéaire

On peut introduire une non-linéarité *N* comme montré en *fig.2*

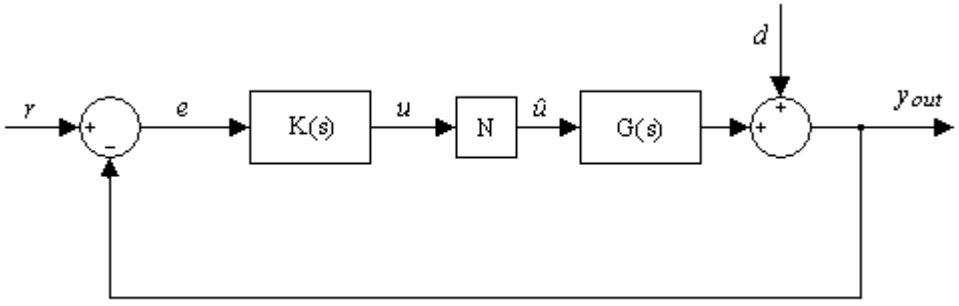


Figure 2: schéma du cas idéale linéaire avec une non-linéarité *N*

C'est la discordance entre *u* et *û* qui cause le windup, les dégradations des performances et quelque fois l'instabilité. Pour remédier à ce problème, on peut utiliser, comme on voit dans la *fig.3*, un contrôleur avec retour $\hat{K}(s)$:

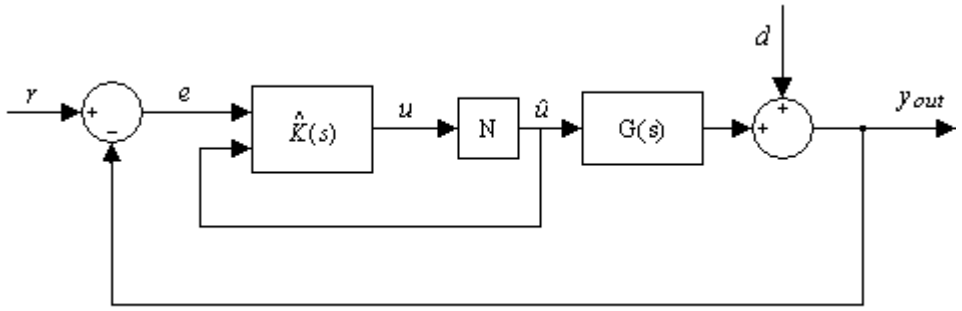


Figure 3: schéma du cas AWBT

Les valeurs mesurées ou estimées de *û* fournissent des informations sur les effets de la non-linéarité *N*. Les critères que $\hat{K}(s)$ doit satisfaire sont les suivants: le système non-linéaire en boucle fermée de la *fig.3* doit être stable; quand *N=I* les performances du système doivent correspondre à celle du système linéaire en *fig.1*; les performances de la boucle fermée en *fig.3* doivent changer graduellement quand *N* devient $\neq I$.

Dans le cas linéaire, on peut considérer la transformation linéaire fractionnaire (LFT) montrée en *fig.4* comme une interconnexion standard de systèmes linéaires. L'entrée *w* inclut tous les signaux qui entrent dans le système par l'extérieur (références et perturbations). *u* représente le signal de contrôle, *z* et y_m représentent respectivement les sorties contrôlées que le contrôleur doit maintenir petites (par exemple l'erreur de poursuite) et toutes les mesures utiles au contrôleur.

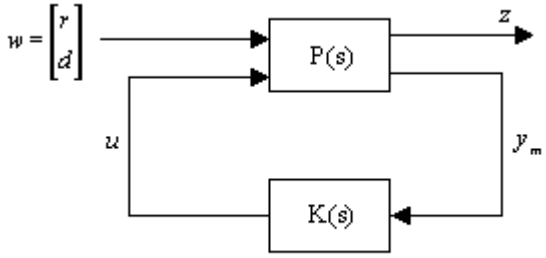


Figure 4: schéma linéaire idéalisé

Si on considère le système en *fig.1* on a $w=\begin{bmatrix} r \\ d \end{bmatrix}$, *z=e=r-* y_{out} et $y_m = e$.

Donc on a $\begin{bmatrix} z \\ y_m \end{bmatrix} = P(s)\begin{bmatrix} r \\ d \\ u \end{bmatrix} \Rightarrow \begin{bmatrix} r-y_{out} \\ r-y_{out} \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \end{bmatrix}\begin{bmatrix} r \\ d \\ u \end{bmatrix} \Rightarrow$

$$\begin{bmatrix} r-d-G(s)u \\ r-d-G(s)u \end{bmatrix} = \begin{bmatrix} I & -I & -G(s) \\ I & -I & -G(s) \end{bmatrix}\begin{bmatrix} r \\ d \\ u \end{bmatrix} \Rightarrow P(s) = \begin{bmatrix} I & -I & -G(s) \\ I & -I & -G(s) \end{bmatrix}$$

Avec ces définitions le comportement du système en *fig.4* est équivalent à celui-ci de la *fig.1*.

Le schéma AWBT général est décrit en *fig.5* :

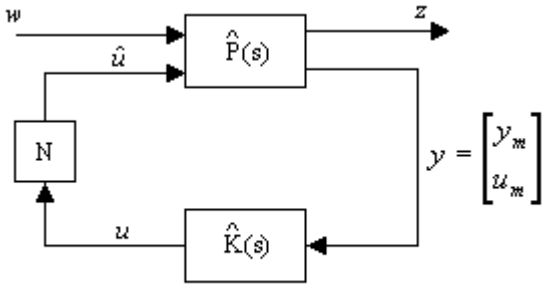


Figure 5: le problème générale AWBT

$\hat{P}(s)$ est obtenu par *P(s)* en ajoutant la sortie u_m, donc

$$\hat{P}(s) = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix}$$

où les premières 2 lignes sont définies comme dans *P(s)* et $u_m = P_{31}r + P_{32}d + P_{33}\hat{u}$.

u_m est la valeur mesurée ou estimée de $\hat{u}$: dans le cas d'une estimation parfaite de $\hat{u}$ sera $P_{31} = P_{32} = 0$ et $P_{33} = I$; si $P_{31} \neq 0$ et $P_{32} \neq 0$ on peut mesurer la perturbation à travers *r* et *d* et si $P_{33} \neq I$ on peut faire une mesure dynamique.

Résoudre le problème général AWBT équivaut donc à trouver $\hat{K}(s)$ tel que le système en *fig.5* sera stable.

1.2. Décomposition de $\hat{K}(s)$

Dans la *fig.6,* on a exprimé $\hat{K}(s)$ comme une interconnexion d'un bloc LTI $\hat{k}(s)$ et d'un opérateur AWBT Λ qui représente un élément non-linéaire.

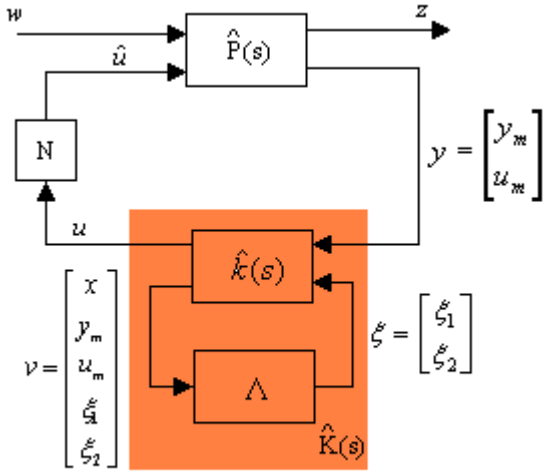


Figure 6 : décomposition de $\hat{K}(s)$

$\hat{k}(s)$ contient *K(s),* Λ utilise les informations fournies par $\hat{k}(s)$ sous la forme d'une entrée *v* pour produire une action AWBT appelée ξ qui est bouclé sur $\hat{k}(s)$. Par soucis de généralité, l'opérateur Λ reçoit en entrée l'état *x* et l'entrée de $\hat{k}(s)$ $\begin{bmatrix} y \\ \xi \end{bmatrix}$. On trouve la réalisation suivante pour $\hat{k}(s)$:

$$\hat{k}(s) = \left[\begin{array}{c|cccc} A & B & 0 & I & 0 \\ \hline C & D & 0 & 0 & I \\ I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{array}\right] \text{ avec } K(s) = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \text{ et } v = \begin{bmatrix} x \\ y_m \\ u_m \\ \xi_1 \\ \xi_2 \end{bmatrix} \qquad (1.1)$$

En terme d'espace d'état on trouve

$$\begin{cases} \dot{x} = Ax + By_m + \xi_1 \\ u = Cx + Dy_m + \xi_2 \\ x = Ix \\ y_m = Iy_m \\ u_m = Iu_m \\ \xi_1 = I\xi_1 \\ \xi_2 = I\xi_2 \end{cases}$$

Avec l'état et l'entrée de $\hat{k}(s)$, on peut caractériser complètement sa sortie, donc on peut affirmer que Λ détient des informations complètes (full information). En outre Λ peut commander l'état et la sortie de $\hat{k}(s)$ donc il agit avec un contrôle complet (full control). On peut aisément voir que pour Λ=0 il n'y a pas d'action corrective AWBT et on a $\hat{K}(s)=\begin{bmatrix}K(s) & 0\end{bmatrix}$: on a retrouvé les interconnexions de la *fig.4* avec en plus la non-linéarité *N*.

Il faut imposer deux critères d'admissibilité pour Λ :

1. $\Lambda : v \to \xi$ est causal, linéaire et temps-invariant
2. $\forall t,\ u(t)-u_m(t)=0 \ \Rightarrow\ \xi(t)=0$

La première condition assure que $\hat{K}(s)$ pourra être réalisé comme un système LTI; la deuxième assure qu'on ne va pas à utiliser le bloc Λ si ce n'est pas nécessaire (c'est-à-dire qu'il n'y a pas de problèmes dus au bloc non-linéaire) . En réalité il faudrait que $\xi(t)=0$ quand $u(t)-\hat{u}(t)=0$, mais souvent on ne dispose en entrée de Λ que d'une estimation de u_m, la condition 2 n'est donc pas exactement vérifiée.

Ces deux critères impliquent que tous les Λ admissibles doivent être des matrices constantes. En outre $\xi(t)$ doit être linéaire par rapport à $u_m(t)-u(t)$, donc $\exists$ deux paramètres λ_1, λ_2 tels que

$$\xi=\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}(u_m-u)=\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}\begin{bmatrix}-C & -D & I & 0 & -I\end{bmatrix}v=\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}\left[u_m-\left(Cx+Dy_m+\xi_2\right)\right]=\Lambda v$$

si on incorpore le bloc Λ et le bloc $\hat{k}(s)$ on obtient la réalisation standard en *fig.5* avec

$$\hat{K}(s)=\begin{bmatrix}U(s) & I-V(s)\end{bmatrix}=\left[\begin{array}{c|cc}A-H_1C & B-H_1D & H_1\\ \hline H_2C & H_2D & I-H_2\end{array}\right] \tag{1.2}$$

où

$$U(s)=\left[\begin{array}{c|c}A-H_1C & B-H_1D\\ \hline H_2C & H_2D\end{array}\right] \tag{1.3}$$

$$V(s)=\left[\begin{array}{c|c}A-H_1C & -H_1\\ \hline H_2C & H_2\end{array}\right] \tag{1.4}$$

avec
$$H_1=\lambda_1(I+\lambda_2)^{-1}$$
$$H_2=(I+\lambda_2)^{-1}$$

où *A, B, C, D* sont les mêmes matrices de la (1.1)

On peut vérifier que $K(s)=V(s)^{-1}U(s)$

Si on suppose que $K(s)$ est observable, c'est-à-dire (*C, A*) observable, les valeurs propres de $A-H_1C$ peuvent être placées de façon arbitraire en choisissant H_1. Si les valeurs propres sont choisies à partie réelle

négative $U(s)$, $V(s)$ et $\hat{K}(s)$ sont stables. On est intéressé par la stabilité globale donc il faut aussi que $\hat{P}(s)$ soit stable; la stabilité en boucle fermée d'un système ne peut pas être garantie avec $\hat{K}(s)$ ou $\hat{P}(s)$ instable.

1.3. Cas particuliers d'AWBT

1.3.1. Anti-reset windup

Le phénomène du windup a été observé originairement dans les contrôleurs PI et PID pour des systèmes de contrôle SISO avec un actionneur sujet à une saturation. On considère la sortie d'un contrôleur PI comme montré dans la *fig.7* :

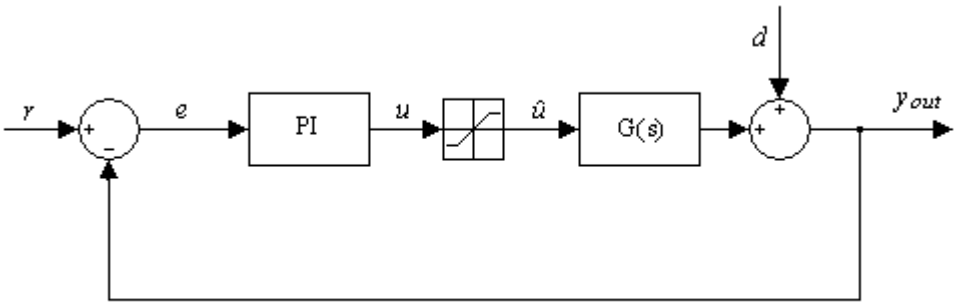


Figure 7 : contrôle PI du processus G(s)

$$K(s) = k(1 + \frac{1}{\tau_I s}) = \left[\begin{array}{c|c} 0 & \frac{k}{\tau_I} \\ \hline 1 & k \end{array}\right]$$

$$u = k(e + \frac{1}{\tau_I}\int_0^t edt)$$

$$\hat{u} = sat(u) = \begin{cases} u_{\min} & si \quad u < u_{\min} \\ u & si \quad u_{\min} \le u \le u_{\max} \\ u_{\max} & si \quad u > u_{\max} \end{cases}$$

$$e = r - y_{out}$$

Si l'erreur e reste positive pendant une certaine période de temps, le signal de contrôle $\hat{u}$ sature à la valeur maximum $u_{\max}$. Si l'erreur continue à rester positive après la saturation de $\hat{u}$, l'intégrateur continue à accumuler une erreur qui sera difficile à annuler en un intervalle de temps raisonnable. Cela peut provoquer une erreur significative sur la sortie voire même l'instabilité du système.

Pour éviter cet effet, on ajoute une boucle qui utilise un nouveau signal d'erreur défini comme la différence entre la sortie du contrôleur u et la sortie de l'actionneur $\hat{u}$ pondérée par un gain $\frac{1}{\tau_r}$ [7]. Le schéma est résumé dans la *fig.8*

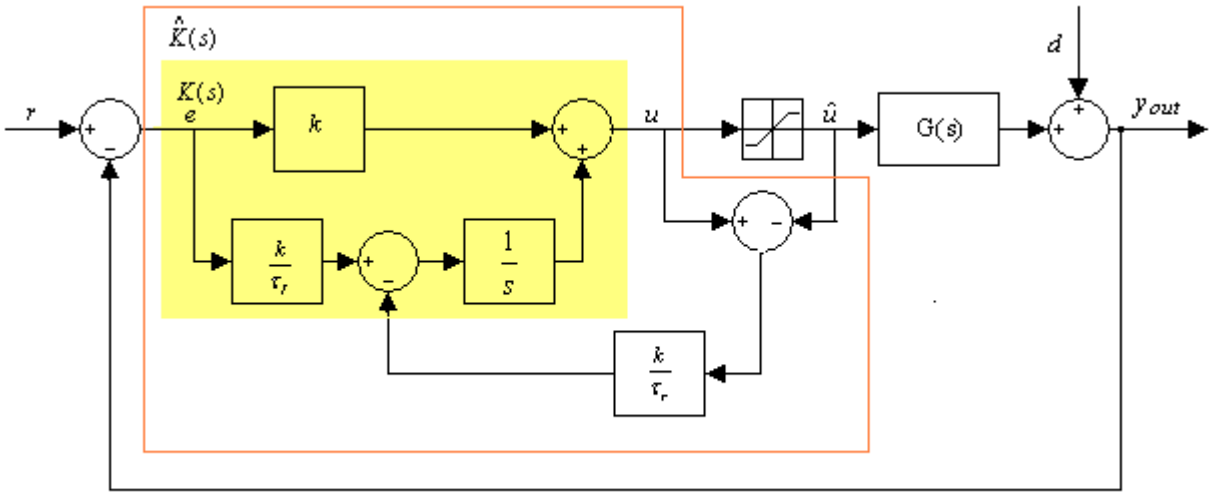


Figure 8 : anti-reset windup classique

$$u = k\left[e + \frac{1}{\tau_I}\int_0^t \left(e - \frac{\tau_I}{k\tau_r}(u-\hat{u}) \right) dt \right] \tag{1.5}$$

Quand l'actionneur sature, le signal en rétroaction essaye de ramener l'erreur $u-\hat{u}$ à zéro en recalculant l'intégral de telle sorte que la sortie du contrôleur u soit à la limite de saturation. Si l'on réécrit (1.5) dans le domaine de Laplace, on obtient :

$$u = ke + \frac{k}{\tau_I s}e - \frac{1}{\tau_r s}(u-\hat{u}) = \frac{k\tau_r(1+\tau_I s)}{\tau_I(1+\tau_r s)}e + \frac{1}{1+\tau_r s}\hat{u}$$

Dans la forme AWBT générale de la *fig.5* avec $w = \begin{bmatrix} r \\ d \end{bmatrix}$, $y = \begin{bmatrix} e \\ \hat{u} \end{bmatrix}$ et $z = e = r - y_{out}$ on a

$$\hat{P}(s) = \begin{bmatrix} I & -I & -G(s) \\ I & -I & -G(s) \\ 0 & 0 & I \end{bmatrix} \quad \text{car} \quad \begin{bmatrix} z \\ y \end{bmatrix} = \hat{P}(s)\begin{bmatrix} w \\ \hat{u} \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} e \\ e \\ \hat{u} \end{bmatrix} = \begin{bmatrix} I & -I & -G(s) \\ I & -I & -G(s) \\ 0 & 0 & I \end{bmatrix}\begin{bmatrix} r \\ d \\ \hat{u} \end{bmatrix}$$

Une réalisation de $\hat{K}(s)$ pour l'anti-reset windup est la suivante :

$$\hat{K}(s) = \begin{bmatrix} \frac{k\tau_r(1+\tau_I s)}{\tau_I(1+\tau_r s)} & \frac{1}{1+\tau_r s} \end{bmatrix} = \left[\begin{array}{c|cc} -\frac{1}{\tau_r} & \frac{k}{\tau_I\tau_r}(\tau_r - \tau_I) & \frac{1}{\tau_r} \\ \hline 1 & k & 0 \end{array}\right] \tag{1.6}$$

en comparant (1.6) avec les (1.2), (1.3), (1.4) on peut écrire, dans la forme AWBT générale

$$U(s)=\frac{k\tau_r(1+\tau_I s)}{\tau_I(1+\tau_r s)}=\left[\begin{array}{c|c} -\frac{1}{\tau_r} & \frac{k}{\tau_I\tau_r}(\tau_r-\tau_I) \\ \hline 1 & k \end{array}\right]$$

$$V(s)=1-\frac{1}{(1+\tau_r s)}=\left[\begin{array}{c|c} -\frac{1}{\tau_r} & -\frac{1}{\tau_r} \\ \hline 1 & 1 \end{array}\right]$$

$$H_1=\frac{1}{\tau_r}$$

$$H_2=1$$

Pour un contrôleur PI, quand l'action intégrale est générée comme un reset automatique, on suggère dans l'article [1] l'implémentation de la *fig.9* pour réaliser un compensateur anti-reset windup:

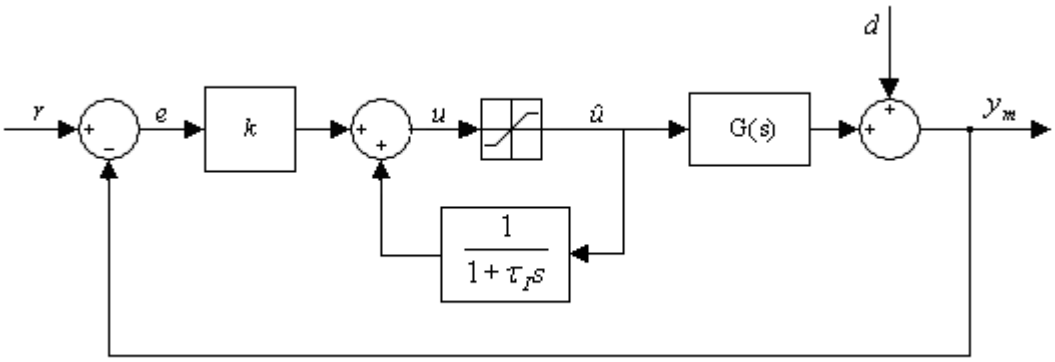


Figure 9: implémentation de l'anti-reset windup

Quand il n'y a pas de saturation on peut vérifier que cette implémentation correspond à celle standard d'un contrôleur PI

$$K(s)=k(1+\frac{1}{\tau_I s})=\left[\begin{array}{c|c} 0 & \frac{k}{\tau_I} \\ \hline 1 & k \end{array}\right]$$

S'il y a saturation, le signal de contrôle s'écrit, dans le domaine de Laplace,

$$u=ke+\frac{1}{1+\tau_I s}\hat{u}$$

Dans la forme AWBT générale de la *fig.5* avec $w=\begin{bmatrix} r \\ d \end{bmatrix}$, $y=\begin{bmatrix} e \\ \hat{u} \end{bmatrix}$ et $z=e$ on a

$$\hat{K}(s)=\begin{bmatrix} k & \frac{1}{1+\tau_I s} \end{bmatrix}=\left[\begin{array}{c|cc} -\frac{1}{\tau_I} & 0 & \frac{1}{\tau_I} \\ \hline 1 & k & 0 \end{array}\right]$$

$$U(s) = k = \left[\begin{array}{c|c} -\frac{1}{\tau_r} & 0 \\ \hline 1 & k \end{array}\right]$$

$$V(s) = \frac{\tau_I s}{1+\tau_I s} = \left[\begin{array}{c|c} -\frac{1}{\tau_I} & -\frac{1}{\tau_I} \\ \hline 1 & 1 \end{array}\right]$$

$$H_1 = \frac{1}{\tau_I}$$
$$H_2 = 1$$

Si on choisit $\tau_r = \tau_I$ les configurations de *fig.8* et *fig.9* sont identiques.

1.3.2. Anti-windup conventionnel (CAW)

On peut considérer le CAW comme l'extension de l'anti-reset windup [6]. Le compensateur AWBT utilise une boucle où la différence $\hat{u} - u$ sert d'entrée pour une matrice à grand gain $X = \alpha I$ où $\alpha >> 1$ est un scalaire.

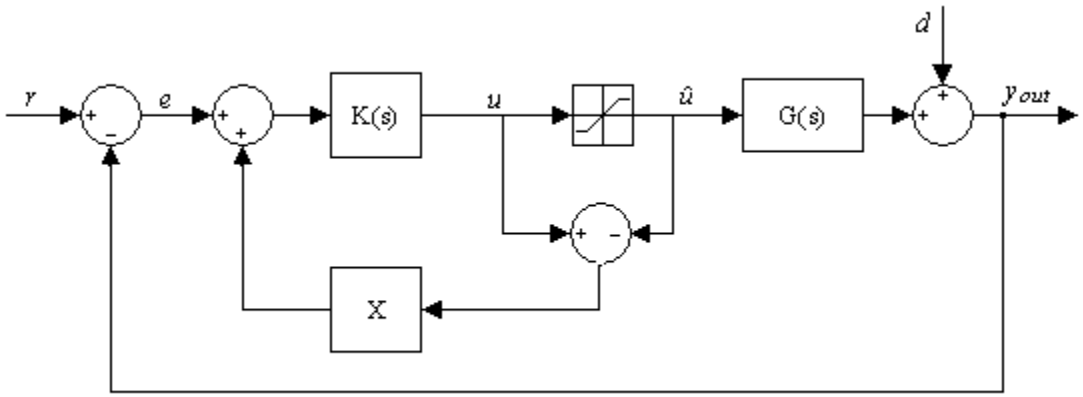


Figure 10: anti-windup conventionnel

Etant donné le contrôleur linéaire $K(s) = \left[\begin{array}{c|c} A & C \\ \hline B & D \end{array}\right]$ on veut trouver le contrôleur modifié $\hat{K}(s)$:

$$\dot{x} = Ax + B(e + X(\hat{u} - u)) \tag{1.7}$$
$$u = Cx + D(e + X(\hat{u} - u)) \quad \Rightarrow \quad u = (I + DX)^{-1}Cx + (I + DX)^{-1}De + (I + DX)^{-1}DX\hat{u} \tag{1.8}$$

en substituant (1.8) dans la (1.7) on obtient

$$\dot{x} = (A - BX(I + DX)^{-1}C)x + (B - BX(I + DX)^{-1}D)e + (BX - BX(I + DX)^{-1}DX)\hat{u} \tag{1.9}$$
$$\dot{x} = (A - BX(I + DX)^{-1}C)x + B(I + DX)^{-1}e + BX(I + DX)^{-1}\hat{u} \ (*) \tag{1.10}$$

Dans la forme AWBT générale de la *fig.5* avec $w=\begin{bmatrix} r \\ d \end{bmatrix}$, $y=\begin{bmatrix} e \\ \hat{u} \end{bmatrix}$ et $z=e$ on a

$$\hat{P}(s)=\begin{bmatrix} I & -I & -G(s) \\ I & -I & -G(s) \\ 0 & 0 & I \end{bmatrix}$$

$$\hat{K}(s)=\left[\begin{array}{c|cc} A-BX(I+DX)^{-1}C & B(I+DX)^{-1} & BX(I+DX)^{-1} \\ \hline (I+DX)^{-1}C & (I+DX)^{-1}D & (I+DX)^{-1}DX \end{array}\right]$$

$\hat{K}(s)$ peut être construit aisément à travers les équations (1.8) et (1.10) avec *y* comme entrée et *u* comme sortie

$$U(s)=\left[\begin{array}{c|c} A-BX(I+DX)^{-1}C & B(I+DX)^{-1} \\ \hline (I+DX)^{-1}C & (I+DX)^{-1}D \end{array}\right]$$

$$V(s)=\left[\begin{array}{c|c} A-BX(I+DX)^{-1}C & -BX(I+DX)^{-1} \\ \hline (I+DX)^{-1}C & (I+DX)^{-1} \end{array}\right]$$

$$H_1=BX(I+DX)^{-1}$$

$$H_2=(I+DX)^{-1}$$

En fait, si on pose $Z=(I+DX)$ on a :

$$I-H_2=I-Z^{-1}=Z^{-1}Z-Z^{-1}=Z^{-1}(Z-I)=Z^{-1}(I+DX-I)=(I+DX)^{-1}DX$$

(*) Démonstration du passage (1.9)$\Rightarrow$(1.10)

Il faut démontrer que $(B-BX(I+DX)^{-1}D)=B(I+DX)^{-1}$ (a)

et que $(BX-BX(I+DX)^{-1}DX)=BX(I+DX)^{-1}$ (b)

par simplicité on pose $Z=(I+DX)$. En outre, on rappelle que $X=\alpha I$ avec α constant.

En prenant en considération (a) on trouve que $(I-XZ^{-1}D)=Z^{-1}$ est équivalent à (b) $(I-Z^{-1}DX)=Z^{-1}$ si on déplace *X*.

Donc

$(I-XZ^{-1}D)=(I-Z^{-1}DX)=Z^{-1}Z-Z^{-1}DX=Z^{-1}(Z-DX)=Z^{-1}(I+DX-DX)=Z^{-1}$

1.3.3. Contrôleur conditionné de Hanus

La technique de conditionnement a été proposée par Hanus et al [8,9]. Dans cette technique le phénomène du windup est interprété comme une absence de consistance entre l'état interne du contrôleur et l'entrée en présence d'une non-linéarité entre la sortie du contrôleur et ses entrées, ce qui est le cas d'une saturation par exemple. La consistance est rétablie en modifiant les entrées du contrôleur. En considérant la *fig.2*

$$\dot{x} = Ax + B(r - y_{out})$$
$$u = Cx + D(r - y_{out})$$
$$\hat{u} = sat(u)$$

on peut appliquer une référence r^r au controleur telle que la sortie du contrôleur sera $\hat{u}$

$$\dot{x} = Ax + B(r^r - y_{out})$$
$$\hat{u} = Cx + D(r^r - y_{out})$$
$$\hat{u} - u = D(r^r - r)$$

en supposant D non singulier on trouve

$$r^r = r + D^{-1}(\hat{u} - u)$$

en combinant les équations précédentes on obtient le contrôleur conditionné

$$\dot{x} = (A - BD^{-1}C)x + BD^{-1}\hat{u}$$

Dans la forme AWBT générale de la *fig.5* avec $w = \begin{bmatrix} r \\ d \end{bmatrix}$, $y = \begin{bmatrix} r - y_{out} \\ \hat{u} \end{bmatrix}$ et $z = r - y_{out}$ on a

$$\hat{K}(s) = \left[\begin{array}{c|cc} A - BD^{-1}C & 0 & BD^{-1} \\ \hline C & D & 0 \end{array}\right]$$

$$U(s) = \left[\begin{array}{c|c} A - BD^{-1}C & 0 \\ \hline C & D \end{array}\right]$$

$$V(s) = \left[\begin{array}{c|c} A - BD^{-1}C & -BD^{-1} \\ \hline C & I \end{array}\right]$$

$$H_1 = BD^{-1}$$
$$H_2 = I$$

Chapitre 2

Correcteur à commutation basé sur un observateur pour systèmes à commutation en temps discret

Dans ce chapitre, on expose une méthode de synthèse d'un correcteur basé observateur qui garantit la stabilité asymptotique d'un système à commutation en temps discret en boucle fermée pour n'importe quelle valeur du signal de commutation. Tous les théorèmes ont été empruntés à [4]. Les techniques décrites nous seront utiles pour le calcul du contrôleur nominal.

2.1. Formulation du problème

On considère le système à commutation défini par

$$\begin{cases} x_{k+1} = A_\alpha x_k + B_\alpha u_k \\ y_k = C_\alpha x_k \end{cases} \tag{2.1}$$

où $x_k \in \Re^n$ est l'état, $u_k \in \Re^r$ est la commande, $y_k \in \Re^m$ est la sortie et α est le signal de commutation qui peut valoir n'importe quelle valeur dans $\{1,...,N\}$.

On veut calculer une loi de contrôle à commutation basée sur un observateur de la forme suivante:

$$\begin{cases} \hat{x}_{k+1} = A_\alpha \hat{x}_k + B_\alpha u_k + L_\alpha (y_k - \hat{y}_k) \\ \hat{y}_k = C_\alpha \hat{x}_k \\ u_k = K_\alpha \hat{x}_k \end{cases} \tag{2.2}$$

pour que le système en boucle fermée

$$\begin{bmatrix} x \\ \mu \end{bmatrix}_{k+1} = \begin{bmatrix} \widetilde{A}_\alpha & \widetilde{B}_\alpha \\ 0 & \hat{A}_\alpha \end{bmatrix} \begin{bmatrix} x \\ \mu \end{bmatrix}_k \tag{2.3}$$

où $\mu_k = x_k - \hat{x}_k$ est l'erreur d'observation et les matrices

$$\widetilde{A}_\alpha = A_\alpha + B_\alpha K_\alpha$$
$$\hat{A}_\alpha = A_\alpha - L_\alpha C_\alpha$$
$$\widetilde{B}_\alpha = -B_\alpha K_\alpha$$

soient asymptotiquement stables.

2.2. Synthèse séparée du contrôleur et de l'observateur

2.2.1. Synthèse d'un contrôleur par retour d'état à commutation

Le calcul d'un contrôleur classique par retour d'état à commutation se réduit à la détermination de

$$u_k = K_\alpha x_k \tag{2.4}$$

qui assure la stabilité du système en boucle fermée $\forall$ la valeur de α.

$$x_{k+1} = (A_\alpha + B_\alpha K_\alpha) x_k = \widetilde{A}_\alpha x_k \tag{2.5}$$

Une solution possible est donnée par le théorème suivant:

Théorème 2.1 *S'il existe des matrices symétriques S_i et des matrices G_i, R_i, $\forall i \in 1,...,N$, telles que*

$$\begin{bmatrix} G_i + G_i^T - S_i & * \\ A_i G_i + B_i R_i & S_j \end{bmatrix} > 0$$

$\forall (i, j) \in NxN$ *alors le contrôleur donné par la (2.4) avec*

$$K_i = R_i G_i^{-1} \quad \forall i \in 1,...,N$$

stabilise asymptotiquement le système (2.5) $\forall$ la valeur de α.

2.2.2. Synthèse d'un observateur à commutation

On considère le système défini par (2.1). Le calcul d'un observateur à commutation

$$\begin{cases} \hat{x}_{k+1} = A_\alpha \hat{x}_k + B_\alpha u_k + L_\alpha (y_k - \hat{y}_k) \\ \hat{y}_k = C_\alpha \hat{x}_k \end{cases} \tag{2.6}$$

pour ce système consiste à déterminer les matrices L_i, $\forall i \in 1,...,N$ telles que l'erreur d'observation entre l'état x_k du système (2.1) et l'état $\hat{x}_k$ de l'observateur (2.6) soient asymptotiquement stables. La convergence à 0 de l'erreur d'observation est indépendante par rapport aux conditions initiales x_0 et $\hat{x}_0$, de u_k et de la loi de commutation α.

La dynamique de l'erreur est définie par $\mu_{k+1} = (A_\alpha - L_\alpha C_\alpha)\mu_k = \hat{A}_\alpha \mu_k$

Le théorème suivant donne des conditions suffisantes pour réaliser l'observateur à commutation

Théorème 2.2 *S'il existe des matrices symétriques* S_i *et des matrices* $G_i, F_i, \ \forall i \in 1,...,N$ *, tel que*

$$\begin{bmatrix} G_i + G_i^T - S_i & * \\ A_i^T G_i - C_i^T F_i & S_j \end{bmatrix} > 0$$

$\forall (i, j) \in NxN$ *alors l'observateur donné par la (2.6) existe et les gains* L_i *sont donnés par*

$$L_i = G_i^{-T} F_i \quad \forall i \in 1,...,N$$

Le théorème suivant nous donne un principe de séparation pour les systèmes à commutation en temps discret:

Théorème 2.3 *Supposons que les gains* K_i et $L_i, \ \forall i \in 1,...,N$, o*nt été calculés comme indiqué dans les théorèmes 2.1 et 2.2, alors le contrôleur à commutation basé sur l'observateur (2.2) stabilise asymptotiquement le système à commutation en boucle fermée (2.3).*

Chapitre 3

Correcteur dynamique par retour de sortie pour systèmes à commutation

Dans ce chapitre on expose une méthode permettant de calculer un contrôleur robuste à temps discret par retour de sortie dynamique qui stabilise le système en boucle fermée tout en satisfaisant une condition de γ-performance. Ce dernier concept nous fournit une borne supérieure pour le rapport entre la sortie contrôlée et une entrée externe jouant le rôle de perturbation. Minimiser γ permet de minimiser l'effet de la perturbation sur cette sortie. Ce type d'optimisation nous sera utile lors de la synthèse d'un contrôleur type « bumpless ». Tous les théorèmes ont été empruntés à [3].

3.1. Analyse de stabilité et γ-performance

On considère un système autonome à commutation

$$x_{k+1} = A_\alpha x_k \tag{3.1}$$

où $x_k \in \Re^n$ est l'état et α est le signal de commutation défini comme dans le chapitre précédent.

La stabilité asymptotique du système (3.1) peut être analysée en utilisant la théorie de Lyapunov.

On peut écrire le système (3.1) comme

$$x_{k+1} = \sum_{i=1}^{N} \varepsilon_i(k) A_i x_k$$

où ε_i est un scalaire qui vaut 1 si le mode A_i est actif et 0 sinon.

Le théorème suivant donne les conditions nécessaires et suffisantes pour l'existence de la fonction de Lyapunov

$$V(k, x_k) = x_k^T P(\varepsilon(k)) x_k = x_k^T \left(\sum_{i=1}^{N} \varepsilon_i(k) P_i \right) x_k \tag{3.2}$$

avec $P_1, ..., P_N$ matrices symétriques définies positives.

Théorème 3.1 *Il existe une fonction de Lyapunov définie par (3.2) dont la différence est définie négative permettant ainsi de tester la stabilité asymptotique du système (3.1) si et seulement si il existe N matrices symétriques $S_1, ..., S_N$ et N matrices $G_1, ..., G_N$ qui satisfont les LMIs suivantes :*

$$\begin{bmatrix} G_i + G_i^T - S_i & G_i^T A_i^T \\ A_i G_i & S_j \end{bmatrix} > 0 \tag{3.3}$$

$\forall (i, j) \in NxN$. *La fonction de Lyapunov est donnée par*

$$V(x_k, \varepsilon_k) = x_k^T \left(\sum_{l=1}^{N} \varepsilon_k^l(k) P_l \right) x_k \quad avec \quad P_l = S_l^{-1}$$

Maintenant on va considérer un système à commutation et à temps discret donné par

$$\begin{cases} x_{k+1} = A_\alpha x_k + B_\alpha^w w_k \\ z_k = C_\alpha^z x_k + D_\alpha^{zw} w_k \end{cases} \tag{3.4}$$

où $x_k \in \Re^n$ est l'état, $x_0 = 0$, $w_k \in \Re^r$ est le bruit et $z_k \in \Re^m$ est la sortie. α est le signal de commutation défini comme précédemment. Les matrices $\left(A_\alpha, B_\alpha^w, C_\alpha^z, D_\alpha^{zw}\right)$ peuvent prendre toutes les valeurs dans l'ensemble $\left\{\left(A_1, B_1^w, C_1^z, D_1^{zw}\right), ..., \left(A_N, B_N^w, C_N^z, D_N^{zw}\right)\right\}$.

Etant donné $\gamma > 0$, la définition suivante concerne le critère de γ-performance pour systèmes à commutation et à temps discret.

Définition 3.1 *On dit que le système autonome (3.4) vérifie une condition de γ-performance s'il est asymptotiquement stable et si* $\gamma^{-1} \sum_{k=0}^{\infty} z_k^2 < \gamma \sum_{k=0}^{\infty} w_k^2, \quad \forall \sum_{k=0}^{\infty} w_k^2 > 0$

Le théorème suivant donne une condition suffisante pour savoir si le système (3.4) vérifie la condition de γ-performance.

Théorème 3.2 *Le système (3.4) vérifie la condition de γ-performance s'il existe des matrices symétriques $S_i \in \Re^{nxn}$ et des matrices $G_i \in \Re^{nxn}$, $\forall i \in 1, ..., N$, tels que*

$$\begin{bmatrix} G_i + G_i^T - S_i & * & * & * \\ 0 & \gamma I & * & * \\ A_i G_i & B_i^w & S_j & * \\ C_i^z G_i & D_i^{zw} & 0 & \gamma I \end{bmatrix} > 0$$

$\forall \{i; j\} \in NxN$

3.2. Synthèse d'un contrôleur par retour d'état

On considère dans cette partie un système à commutation et à temps discret donné par

$$\begin{cases} x_{k+1} = A_\alpha x_k + B_\alpha u_k + B_\alpha^w w_k \\ z_k = C_\alpha^z x_k + D_\alpha^z u_k + D_\alpha^{zw} w_k \end{cases}$$

où $x_k \in \Re^n$ est l'état, $x_0 = 0$, $u_k \in \Re^r$ est la commande, $w_k \in \Re^r$ est une perturbation et $z_k \in \Re^m$ est la sortie. α est le signal de commutation définie comme précédemment.

Le signal de commutation α n'est pas connu à priori mais on suppose qu'on peut le connaître en temps réel pour pouvoir appliquer instantanément le contrôleur adapté.

Pour résoudre le problème de γ-performance par retour d'état, il faut trouver

$$u_k = K_\alpha x_k \tag{3.5}$$

tel que le système à commutation en boucle fermée

$$\begin{cases} x_{k+1} = (A_\alpha + B_\alpha K_\alpha) x_k + B_\alpha^w w_k \\ z_k = (C_\alpha^z + D_\alpha^z K_\alpha) x_k + D_\alpha^{zw} w_k \end{cases} \tag{3.6}$$

vérifie la condition de γ-performance $\forall$ la valeur de α. Le théorème suivant donne des conditions suffisantes pour calculer un tel contrôleur:

Théorème 3.3 *Il existe un contrôleur à commutation par retour d'état (3.5) tel que le système en boucle fermée (3.6) vérifie la condition de γ-performance s'il existe des matrices symétriques définies positives $S_i \in \Re^{nxn}$, des matrices $G_i \in \Re^{nxn}$ et $R_i \in \Re^{rxn}$, $\forall i \in 1,...,N$, telles que*

$$\begin{bmatrix} G_i + G_i^T - S_i & * & * & * \\ 0 & \gamma I & * & * \\ A_i G_i + B_i R_i & B_i^w & S_j & * \\ C_i^z G_i + D_i^z R_i & D_i^{zw} & 0 & \gamma I \end{bmatrix} > 0 \tag{3.7}$$

$\forall \{i; j\} \in NxN$. La loi de commande par retour d'état avec γ-performance est donnée par la (3.5) avec $K_i = R_i G_i^{-1}$.

3.3. Retour de sortie dynamique (Dynamique Output Feedback)

On considère la classe de systèmes à commutation donnée par

$$\begin{cases} x_{k+1} = A_\alpha x_k + B_\alpha u_k + B_\alpha^w w_k \\ z_k = C_\alpha^z x_k + D_\alpha^z u_k + D_\alpha^{zw} w_k \\ y_k = C_\alpha x_k + D_\alpha^{yw} w_k \end{cases} \tag{3.8}$$

où $x_k \in \Re^n$ est l'état, $x_0 = 0$, $u_k \in \Re^r$ est l'entrée du contrôleur, $w_k \in \Re^r$ est une perturbation , $z_k \in \Re^m$ est la sortie contrôlée et $y_k \in \Re^m$ est la sortie mesurée. α est le signal de commutation définie précédemment.

On est intéressé par un contrôleur dynamique à commutation

$$\begin{cases} \eta_{k+1} = A_\alpha^c \eta_k + B_\alpha^c y_k \\ u_k = C_\alpha^c \eta_k + D_\alpha^c y_k \end{cases} \tag{3.9}$$

permettant de garantir la stabilité du système en boucle fermée:

$$\begin{bmatrix} x \\ \eta \end{bmatrix}_{k+1} = \begin{bmatrix} A_\alpha + B_\alpha C_\alpha^c C_\alpha & B_\alpha C_\alpha^c \\ B_\alpha^c C_\alpha & A_\alpha^c \end{bmatrix} \begin{bmatrix} x \\ \eta \end{bmatrix}_k$$

Le système (3.8) et le contrôleur (3.9) ont le même signal de commutation α .

Le théorème suivant donne une condition suffisante pour réaliser une loi de contrôle qui stabilise le système:

Théorème 3.4 *S'il existe des matrices symétriques* S_i, N_i *et des matrices* $M_i, X, Y, W, L_i, F_i, Q_i, R_i$ $\forall i \in N$ *telles que* $\forall (i, j) \in NxN$,

$$\begin{bmatrix} X_i + X_i^T - S_i & * & * & * \\ I + W - M_i & Y_i + Y_i^T - N_i & * & * \\ A_i X + B_i L_i & A_i + B_i R_i C_i & S_j & * \\ Q_i & YA_i + F_i C_i & M_j & N_j \end{bmatrix} > 0$$

alors on peut trouver les matrices du contrôleur (3.9) comme

$VU = W - YX$

$$A_i^c = V^{-1}[Q_i - Y(A_i + B_i R_i C_i)X - VB_i C_i X - YB_i C_i U]U^{-1}$$

$$B_i^c = V^{-1}(F_i - YB_i R_i)$$

$$C_i^c = (L_i - R_i C_i X)U^{-1}$$

$$D_i^c = R_i$$

qui stabilisent le système (3.8) $\forall i \in N$.

3.4. Contrôleur DOF à commutation avec γ -performance

On est intéressé par la synthèse du contrôleur (*3*.8) tel que le système en boucle fermée

$$\begin{bmatrix} x \\ \eta \end{bmatrix}_{k+1} = \begin{bmatrix} A_\alpha + B_\alpha D_\alpha^c C_\alpha & B_\alpha C_\alpha^c \\ B_\alpha^c C_\alpha & A_\alpha^c \end{bmatrix} \begin{bmatrix} x \\ \eta \end{bmatrix}_k + \begin{bmatrix} B_\alpha^w + B_\alpha D_\alpha^c D_\alpha^{yw} \\ B_\alpha^c D_\alpha^{yw} \end{bmatrix} w_k \tag{3.10}$$

$$z_k = [C_\alpha^z + D_\alpha^z D_\alpha^c C_\alpha \quad D_\alpha^z C_\alpha^c] \begin{bmatrix} x \\ \eta \end{bmatrix}_k + (D_\alpha^{zw} D_\alpha^c D_\alpha^{yw}) w_k \tag{3.11}$$

vérifie la condition de γ -performance. Le théorème suivant donne une condition suffisante pour réaliser un contrôleur DOF à commutation:

Théorème 3.5 *S'il existe des matrices symétriques* S_i, N_i *et des matrices* $M_i, X, Y, W, L_i, F_i, Q_i, R_i$ *solutions de la*

$$\begin{bmatrix} X + X^T - S_i & * & * & * & * & * \\ I + W - M_i & Y + Y^T - N_i & * & * & * & * \\ 0 & 0 & \gamma I & * & * & * \\ A_i X + B_i L_i & A_i + B_i R_i C_i & B_i^w + B_i R_i D_i^{yw} & S_j & * & * \\ Q_i & YA_i + F_i C_i & F_i D_i^{yw} + YB_i^w & M_j & N_j & * \\ C_i^z X + D_i^z L_i & C_i^z + D_i^z R_i C_i & D_i^{zw} + D_i^z R_i D_i^{yw} & 0 & 0 & \gamma I \end{bmatrix} > 0$$

$\forall (i, j) \in NxN$, *alors le contrôleur DOF à commutation donné par la (3.10) avec*

$VU = W - YX$

$$A_i^c = V^{-1}[Q_i - Y(A_i + B_i R_i C_i)X - VB_i C_i X - YB_i C_i U]U^{-1}$$

$$B_i^c = V^{-1}(F_i - YB_i R_i)$$

$$C_i^c = (L_i - R_i C_i X)U^{-1}$$

$$D_i^c = R_i$$

$\forall i \in N$ *assure que le système en boucle fermée (3.11) est asymptotiquement stable avec* γ *-performance.*

Chapitre 4

Synthèse d'un correcteur type « bumpless »

Dans ce chapitre on va décrire la structure des contrôleurs qu'on utilisera pour traiter le problème du « bumpless transfer ». D'abord on explique le fonctionnement du contrôleur qui stabilise le système en boucle fermée. Ensuite on illustre la structure du contrôleur « bumpless » pour lequel on présentera trois possibilités: deux contrôleurs statiques (le deuxième avec un observateur) et un dynamique.

4.1. Contrôleur nominal

Étant donné un système à commutation

$$\begin{cases} x_{k+1} = A_\alpha x_k + B_\alpha u_k + B_\alpha^w w_k \\ y_k = C_\alpha x_k + D_\alpha^w w_k \end{cases}$$

où $x_k \in \Re^n$ est l'état, $u_k \in \Re^r$ la commande, $y_k \in \Re^m$ la sortie, $w_k \in \Re^r$ la perturbation et α est le signal de commutation défini comme dans les chapitres précédents

et un correcteur basé observateur

$$\begin{cases} \hat{x}_{k+1} = A_\alpha \hat{x}_k + B_\alpha u_k + L_\alpha (y_k - \hat{y}_k) \\ \hat{y}_k = C_\alpha \hat{x}_k \\ u_k = K_\alpha \hat{x}_k \end{cases}$$

qui garantit la stabilité du système en boucle fermée et qu'on appellera contrôleur nominal K_α^{nom} on peut construire le système en boucle fermée en *fig.11*

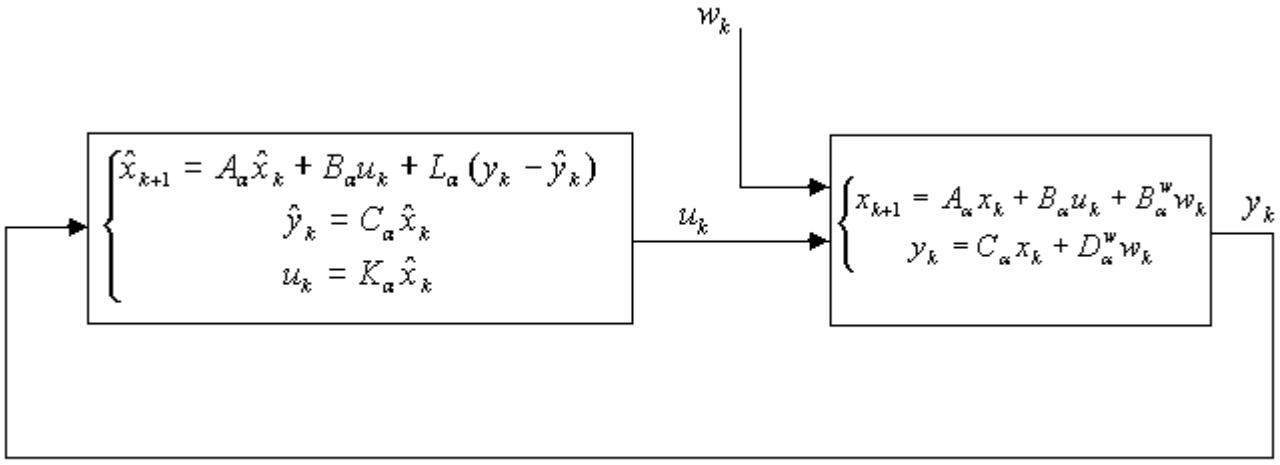


Figure 11: contrôleur nominal

Les matrices K_α et L_α du K_α^{nom} seront fournies à partir des théorèmes 2.1 et 2.2, c'est à dire

$$\begin{bmatrix} G_i + G_i^T - S_i & * \\ A_i G_i + B_i R_i & S_j \end{bmatrix} > 0 \qquad \forall (i,j) \in NxN$$

$$K_i = R_i G_i^{-1} \in \Re^{rxn}, \quad \forall i \in N$$

$$\begin{bmatrix} G_i + G_i^T - S_i & * \\ A_i^T G_i - C_i^T F_i & S_j \end{bmatrix} > 0 \qquad \forall (i,j) \in NxN$$

$$L_i = G_i^{-T} F_i \in \Re^{nxm}, \quad \forall i \in N$$

4.2. Contrôleur type « bumpless »

4.2.1. Contrôleur statique

Comme on l'a précisé dans l'introduction de ce mémoire, à cause des commutations, la commande *u* subit des discontinuités qui peuvent détériorer les performances. Il faut donc chercher à minimiser l'amplitude de ces sauts de commande lors des commutations de correcteurs.

On prévoit un compensateur additionnel dit « bumpless » pour le système en boucle fermée de *fig.11* pour réduire ces sauts. Une structure possible pour le compensateur bumpless est celle proposé dans [5], où la commande *u* est bouclée sur un nouveau contrôleur dit K_α^{bt} dont la sortie est injectée en entrée de K_α^{nom}.

Cette méthode a été développée pour des systèmes LPV (linear parameter varying), c'est à dire une classe de systèmes bien plus générale que celle qu'on veut traiter dans ce travail. Dans le cas LPV, la complexité des conditions proposées ne permet pas d'évaluer l'intérêt d'un tel schéma. Le but ici est de reprendre cette structure de correction « bumpless » dans un cas plus simple pour évaluer son efficacité.

Le contrôleur « bumpless » K_α^{bt} est calculé en utilisant le théorème 3.3 :

$$\begin{bmatrix} G_i + G_i^T - S_i & * & * & * \\ 0 & \gamma I & * & * \\ A_i^{bt} G_i + B_i^{bt} R_i & B_i^{btw} & S_j & * \\ C_i^{bt} G_i + D_i^y R_i & D_i^{btw} & 0 & \gamma I \end{bmatrix} > 0 \qquad \forall (i;j) \in NxN .$$

$$K_i^{bt} = R_i G_i^{-1} \in \Re^{n+rx2n}, \quad \forall i \in N$$

Le schéma sur lequel on travaillera sera celui-ci montré de la *fig.12*:

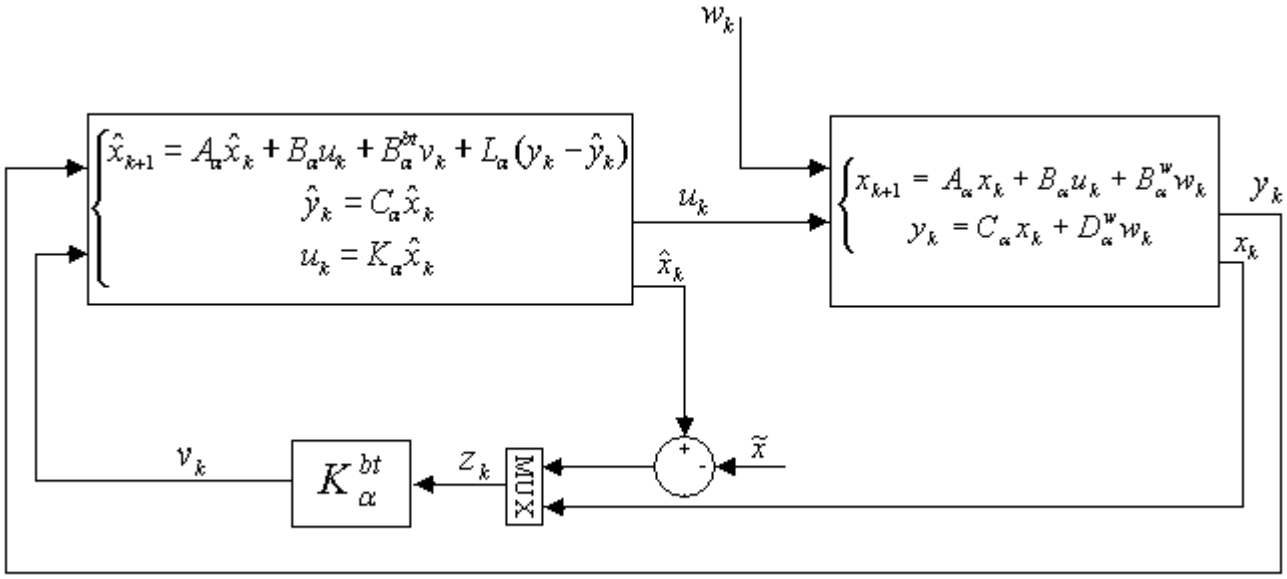


Figure 12: contrôleur bumpless transfer statique

Nous avons essayé une première approche qui consiste à désactiver le compensateur, c'est à dire de mettre $\tilde{x} = 0$, quand $\|u(t) - u(t_m^-)\| < \varepsilon$ comme dans les méthodes classiques d'anti-windup. Nous avons constaté que cette norme est difficile à interpréter pour des cas où les matrices du système commutent surtout en présence de perturbations et cette condition devient presque inutilisable. Comme la forme du contrôleur choisi est un retour d'état statique et qu'on est en temps discret, on a décidé de tester ce que peut donner un raisonnement sur l'état et non sur la commande. On utilise donc comme entrée du contrôleur « bumpless » l'état augmenté $z_k = \begin{bmatrix} x \\ \hat{x} - \tilde{x} \end{bmatrix}_k$. Le signal $\tilde{x}$ est défini comme suit : quand le système commute à l'instant t_m, $\tilde{x}$ est mis à la valeur précédente de l'état observé $\tilde{x} = \hat{x}(t_m^-)$, et on le maintient un certain temps à cette valeur. La solution adoptée est empirique et consiste à désactiver le compensateur après un nombre fini d'échantillons. Cette solution donne de meilleurs résultats en pratique mais ne constitue pas la solution optimale.

L'entrée $\tilde{x}$ sera utile pour aboutir à un critère d'optimisation qui permet de limiter les sauts sur la commande. En fait la méthode choisie pour calculer K_α^{bt} utilise un critère de γ-performance. Comme déjà expliqué, ça signifie que la quantité γ qu'on cherche à minimiser nous donne une borne supérieure sur le rapport entre la sortie correspondant aux performances et l'entrée externe du système ($\gamma > \sqrt{\frac{\sum_{k=0}^{\infty} \|z_k\|^2}{\sum_{k=0}^{\infty} \|w_k\|^2}}$) .

Dans notre cas on a une borne supérieure sur la quantité $\sqrt{\frac{\sum_{k=0}^{\infty}\left\| \begin{matrix} x_k \\ \hat{x}_k - \widetilde{x}_k \end{matrix} \right\|^2}{\sum_{k=0}^{\infty}\|\widetilde{x}_k\|^2}}$, donc en minimisant $z_k = \begin{bmatrix} x \\ \hat{x} - \widetilde{x} \end{bmatrix}_k$ on minimise l'écart entre $\hat{x}$ et $\widetilde{x}$, c'est à dire qu'on essaye rapprocher l'état de la valeur avant la commutation.

Maintenant on est en mesure de décrire la stratégie dans sa totalité:

$$\begin{cases} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}_{k+1} = \begin{bmatrix} A_\alpha & B_\alpha K_\alpha \\ L_\alpha C_\alpha & A_\alpha + B_\alpha K_\alpha - L_\alpha C_\alpha \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}_k + B_\alpha^{bt} v_k + B_\alpha^{btw} W_k \\ z_k = C_\alpha^{bt} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}_k + D_\alpha^{btw} W_k \\ v_k = K_\alpha^{bt} z_k \end{cases}$$

Il reste encore à définir les matrices A_α^{bt}, B_α^{bt}, B_α^{btw}, C_α^{bt}, D_α^{btw}, D_α^{y} et W_k

W_k représente la perturbation du système définie comme $W_k = \begin{bmatrix} w \\ \widetilde{x} \end{bmatrix}_k \in \Re^{n+r}$

La matrice A_α^{bt} sera naturellement $A_\alpha^{bt} = \begin{bmatrix} A_\alpha & B_\alpha K_\alpha \\ L_\alpha C_\alpha & A_\alpha + B_\alpha K_\alpha - L_\alpha C_\alpha \end{bmatrix}$

Pour B_α^{bt} on choisit $B_\alpha^{bt} = \begin{bmatrix} 0 \\ I_n \end{bmatrix} \in \Re^{2nxn}$ car K_α^{bt} doit agir seulement sur l'état observé $\hat{x}$.

Avec le même raisonnement on choisira $B_\alpha^{btw} = \begin{bmatrix} B_\alpha & 0 \\ 0 & 0 \end{bmatrix} \in \Re^{2nxn+r}$ car la perturbation agit seulement sur l'état *x* et l'entrée artificielle $\widetilde{x}$ n'agit pas directement sur l'état donc toutes les colonnes en dehors de la première seront nulles.

En étant donné $z_k = C_\alpha^{bt} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}_k + D_\alpha^{btw} W_k = \begin{bmatrix} x \\ \hat{x} - \widetilde{x} \end{bmatrix}_k$ il faut choisir les matrices C_α^{bt} et D_α^{btw} comme $C_\alpha^{bt} = I_{2n} \in \Re^{2nx2n}$ et $D_\alpha^{btw} = -\begin{bmatrix} 0 & 0 \\ 0 & I_n \end{bmatrix} \in \Re^{2nxn+r}$

Finalement, $D_\alpha^{y} = 0 \in \Re^{2nxn+r}$.

4.2.2. Synthèse du correcteur statique avec un observateur additionnel

La méthode qu'on a présentée dans le paragraphe précédent présente l'inconvénient de connaître l'état *x* du système ce qui n'est pas toujours possible. Un moyen de contourner cette difficulté consiste à remplacer *x* par $\hat{x}$. Le problème est que pendant le transitoire $\hat{x}$ s'écarte beaucoup de *x* dans le transitoire à cause du fait que l'observateur présente dans le K_α^{nom} n'est pas un observateur 'classique' vu qu'il prend en entrée le signal *v*, c'est à dire la sortie du K_α^{bt}. Ce fait peut provoquer l'instabilité du système.

On a décidé d'ajouter un nouvel observateur qui prend en entrée y_k et u_k et qui a la structure suivante:

$$\begin{cases} \hat{\hat{x}}_{k+1} = A_\alpha \hat{\hat{x}}_k + B_\alpha u_k + L_\alpha (y_k - \hat{\hat{y}}_k) \\ \hat{\hat{y}}_k = C_\alpha \hat{\hat{x}}_k \end{cases}$$

où $\hat{\hat{x}}$ est le nouvel état observé.

Le schéma du système modifié est montré en *fig.13*:

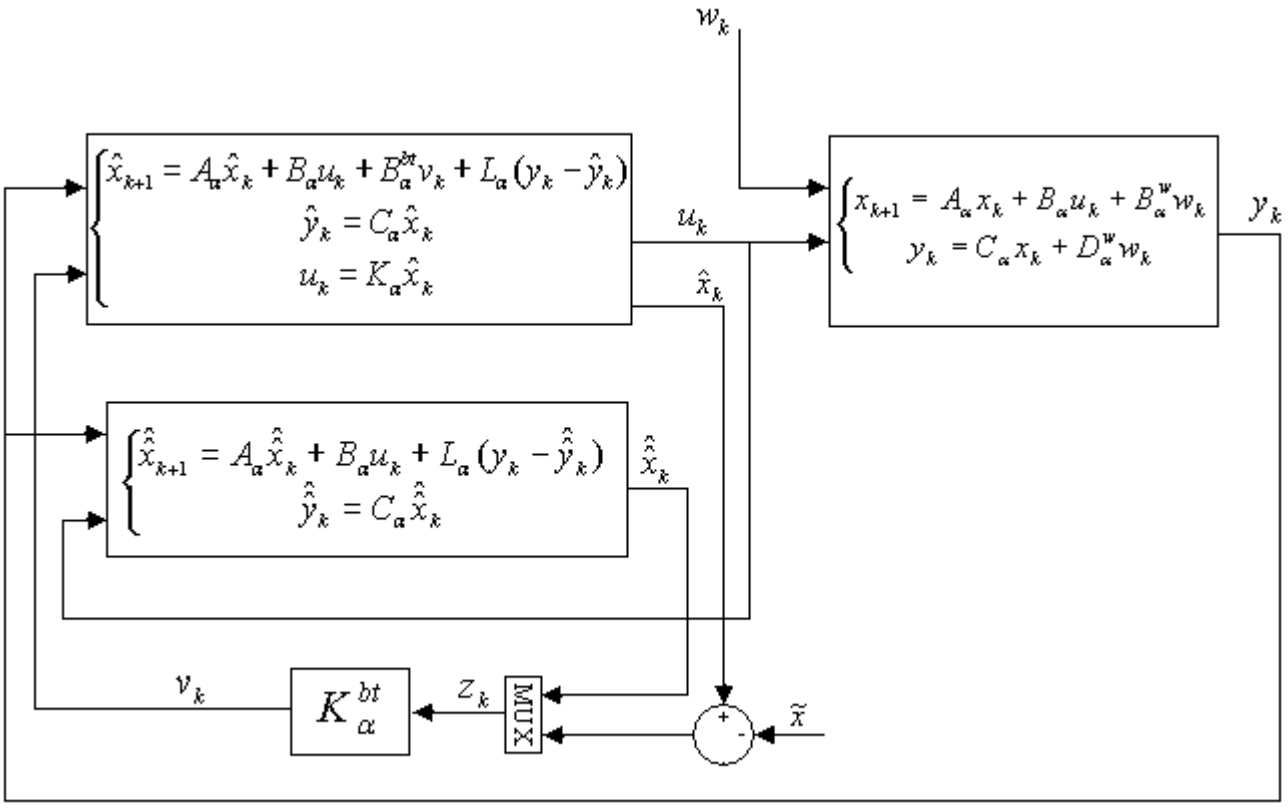


Figure 13: contrôleur bumpless transfer statique avec observateur additionnel

$\hat{\hat{x}}$ peut être utilisé pour construire le nouveau signal $z_k = \begin{bmatrix} \hat{\hat{x}} \\ \hat{x} - \tilde{x} \end{bmatrix}_k$ qui sera donc indépendant de l'état *x*. Cette méthode présente un inconvénient: dans le cas où le bruit est trop fort $\hat{\hat{x}}$ s'écart beaucoup de *x* ce qui compromet la capacité de prévention des sauts. On paye le fait de ne pas utiliser l'état *x*.

4.2.3. Contrôleur dynamique

Une autre solution pour éviter d'utiliser l'état *x* consiste à remplacer le correcteur statique par un correcteur dynamique.

Le système en boucle fermée aura la suivante structure:

$$\begin{cases} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}_{k+1} = \begin{bmatrix} A_\alpha & B_\alpha K_\alpha \\ L_\alpha C_\alpha & A_\alpha + B_\alpha K_\alpha - L_\alpha C_\alpha \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}_k + B_\alpha^{bt} v_k + B_\alpha^{btw} W_k \\ z_k = C_\alpha^{bt} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}_k + D_\alpha^{btw} W \end{cases}$$

tandis que le contrôleur dynamique, qui génère la commande *v*, aura la forme

$$\begin{cases} \eta_{k+1} = A_\alpha^c \eta_k + B_\alpha^c z_k \\ v_k = C_\alpha^c \eta_k + D_\alpha^c z_k \end{cases}$$

Il est claire que la structure générale du système ne change pas: on utilisera toujours le même observateur pour construire K_α^{nom} et une méthode de γ-performance pour minimiser $z_k = \hat{x}_k - \tilde{x}_k$, qui cette fois ne dépend pas de *x* ou de $\hat{\hat{x}}$.

Le schéma du système est celui-ci montré en *fig. 14*:

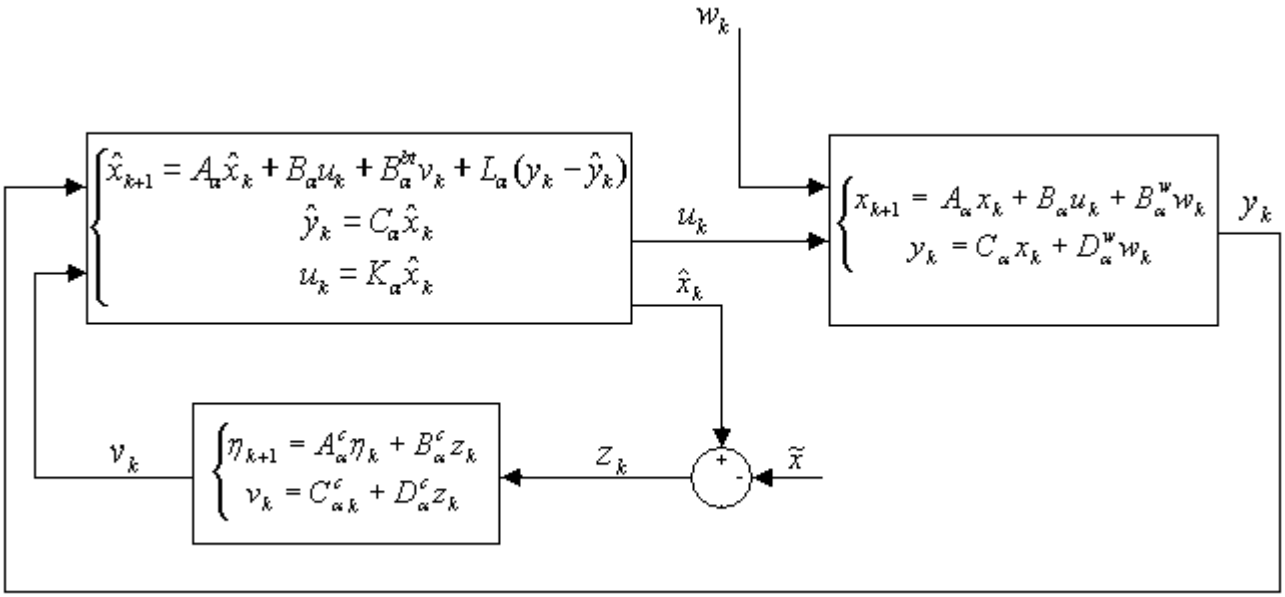


Figure 14: contrôleur bumpless transfer dynamique

En particulier pour calculer les matrices du K_α^{bt} on a décidé d'utiliser les résultats du théorème 3.5:

d'abord il faut résoudre les LMIs

$$\begin{bmatrix} X + X^T - S_i & * & * & * & * & * \\ I + W - M_i & Y + Y^T - N_i & * & * & * & * \\ 0 & 0 & \gamma I & * & * & * \\ A_i^{bt} X + B_i^{bt} L_i & A_i^{bt} + B_i^{bt} R_i C_i^{bt} & B_i^{btw} + B_i^{bt} R_i D_i^{btw} & S_j & * & * \\ Q_i & Y A_i^{bt} + F_i C_i^{bt} & F_i D_i^{bt} + Y B_i^{btw} & M_j & N_j & * \\ C_i^u X + D_i^u L_i & C_i^u + D_i^u R_i C_i^{bt} & D_i^{uw} + D_i^u R_i D_i^{bt} & 0 & 0 & \gamma I \end{bmatrix} > 0 \ \forall (i, j) \in NxN ,$$

où S_i et N_i sont des matrices symétriques $\forall i \in N$.

Ensuite, il est facile de calculer les matrices du K_α^{bt} à travers les formules suivantes, $\forall i \in N$:

$VU=W-YX$

$A_i^c = V^{-1}[Q_i - Y(A_i + B_i R_i C_i)X - VB_i C_i X - YB_i C_i U]U^{-1}$ $\in \Re^{2nx2n}$

$B_i^c = V^{-1}(F_i - YB_i R_i)$ $\in \Re^{2nxn}$

$C_i^c = (L_i - R_i C_i X)U^{-1}$ $\in \Re^{n+rx2n}$

$D_i^c = R_i$ $\in \Re^{n+rxn}$

Il reste à définir la forme des matrices A_α^{bt} B_α^{bt} , B_α^{btw} , C_α^{bt} , C_α^u , D_α^{btw} , D_α^u , D_α^{uw}

La matrice A_α^{bt} sera encore $A_\alpha^{bt} = \begin{bmatrix} A_\alpha & B_\alpha K_\alpha \\ L_\alpha C_\alpha & A_\alpha + B_\alpha K_\alpha - L_\alpha C_\alpha \end{bmatrix}$

La structure de K_α^{nom} n'ayant pas changé les matrices B_α^{bt} et B_α^{btw} seront les mêmes que le cas précédent, donc $B_\alpha^{bt} = \begin{bmatrix} 0 \\ I_n \end{bmatrix} \in \Re^{2nxn}$ et $B_\alpha^{btw} = \begin{bmatrix} B_\alpha & 0 \\ 0 & 0 \end{bmatrix} \in \Re^{2nxn+r}$.

En étant donné $z_k = C_\alpha^{bt} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}_k + D_\alpha^{btw} W_k = \hat{x}_k - \tilde{x}$ on choisit les matrices C_α^{bt} et D_α^{btw} comme $C_\alpha^{bt} = \begin{bmatrix} 0 & I_n \end{bmatrix} \in \Re^{nx2n}$ et $D_\alpha^{btw} = -\begin{bmatrix} 0 & I_n \end{bmatrix} \in \Re^{nxn+r}$

En outre on aura $D_\alpha^{uw} = D_\alpha^{bt} \in \Re^{nxn+r}$, $D_\alpha^u = 0 \in \Re^{nxn+r}$ et $C_\alpha^u = \begin{bmatrix} 0 & I_n \end{bmatrix} \in \Re^{nx2n}$.

Chapitre 5

Simulations

Dans ce dernier chapitre on va effectuer des simulations numériques à l'aide du logiciel Matlab. On verra deux exemples: dans le premier cas on va tester le contrôleur statique sans et avec l'observateur additionnel avec un système à trois variables d'état tandis que dans le deuxième on va tester tant le contrôleur statique que celui dynamique sur un système à deux variables d'état. On verra que tandis qu'avec le contrôleur statique les résultats sont assez positifs avec le contrôleur dynamique on n'a pas réussi à obtenir des résultats significatifs.

5.1. Contrôleur « bumpless » statique

5.1.1. Description du système

Le but de ce chapitre est de montrer le fonctionnement du K_α^{bt} sur des exemples numériques:

on considère un système à commutation donné par la

$$\begin{cases} x_{k+1} = A_\alpha x_k + B_\alpha u_k + B_\alpha^w w_k \\ y_k = C_\alpha x_k + D_\alpha^w w_k \end{cases}$$

où

$\{A_i : i \in N\}$, $\{B_i : i \in N\}$ et $\{C_i : i \in N\}$ avec

$$A_i = \begin{bmatrix} 0 & 0.89 & 0.5 \\ h_i & 0.89 & 0 \\ -0.1 & 0 & 0.9 \end{bmatrix}$$

et i=1,2, $h_1 = -\lambda_1$, $h_2 = \lambda_2$

$$B_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 \\ -6(\lambda_1 + \lambda_2) \\ 0 \end{bmatrix}$$

$B_1^w = B_1$, $B_2^w = B_2$

$C_1 = \begin{bmatrix} -1 & 1 & -2 \end{bmatrix}$, $C_2 = \begin{bmatrix} -2 & 0.35 & 1 \end{bmatrix}$

$D_1^w = \begin{bmatrix} 1 \end{bmatrix}$, $D_2^w = \begin{bmatrix} 1 \end{bmatrix}$

et $\lambda_1 = 1.12$, $\lambda_2 = 2$

En appliquant les résultats du chapitre 2 et 3 on peut trouver les matrices du contrôleur nominal K_α^{nom} :

$$K_1 = \begin{bmatrix} 0.7465 & -0.6057 & -0.9517 \end{bmatrix}$$

$$K_2 = \begin{bmatrix} -0.1038 & -0.0329 & -0.0195 \end{bmatrix}$$

$$L_1 = \begin{bmatrix} 0.4139 \\ 0.8912 \\ -0.0060 \end{bmatrix}$$

$$L_2 = \begin{bmatrix} -0.2949 \\ -1.4198 \\ 0.0306 \end{bmatrix}$$

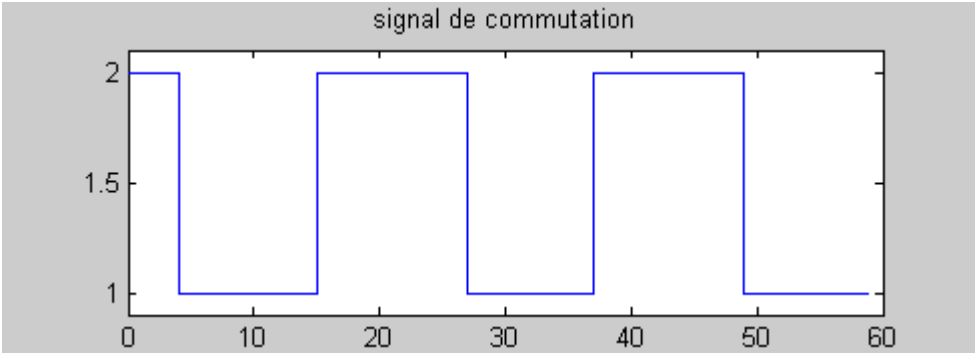
et du contrôleur « bumpless » K_α^{bt} :

$$K_1^{bt} = \begin{bmatrix} 0.1302 & 0.4328 & 1.1821 & -0.4096 & -0.4796 & -1.3332 \\ 0.5686 & -0.5740 & 2.0772 & 0.4349 & -0.1660 & -2.0450 \\ -0.3606 & 0.2280 & 0.2911 & -0.3745 & 0.3838 & -0.2755 \end{bmatrix}$$

$$K_2^{bt} = \begin{bmatrix} 0.1416 & 1.2441 & 0.9764 & -0.1453 & -1.2263 & -0.9328 \\ -2.0136 & 1.0315 & 1.7597 & 1.9524 & -1.0339 & -1.2108 \\ 0.9683 & 0.4272 & 0.2577 & -0.8721 & -0.2783 & -1.0405 \end{bmatrix}$$

$$\gamma = 89.8632$$

En appliquant le signal de commutation suivant avec $i = 1,2$



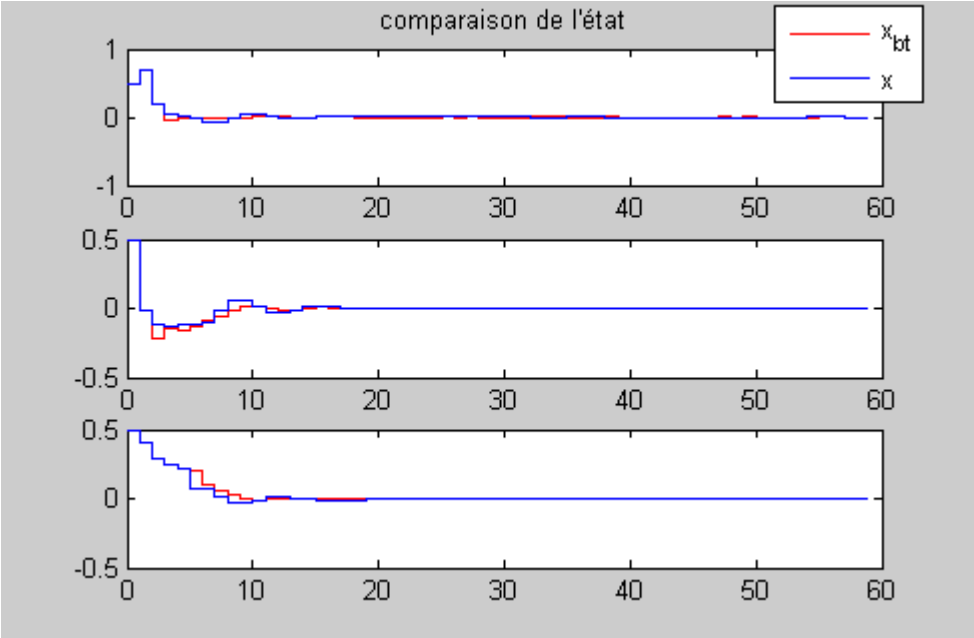
on trouve les résultats suivants:

5.1.2. Simulation 1.1

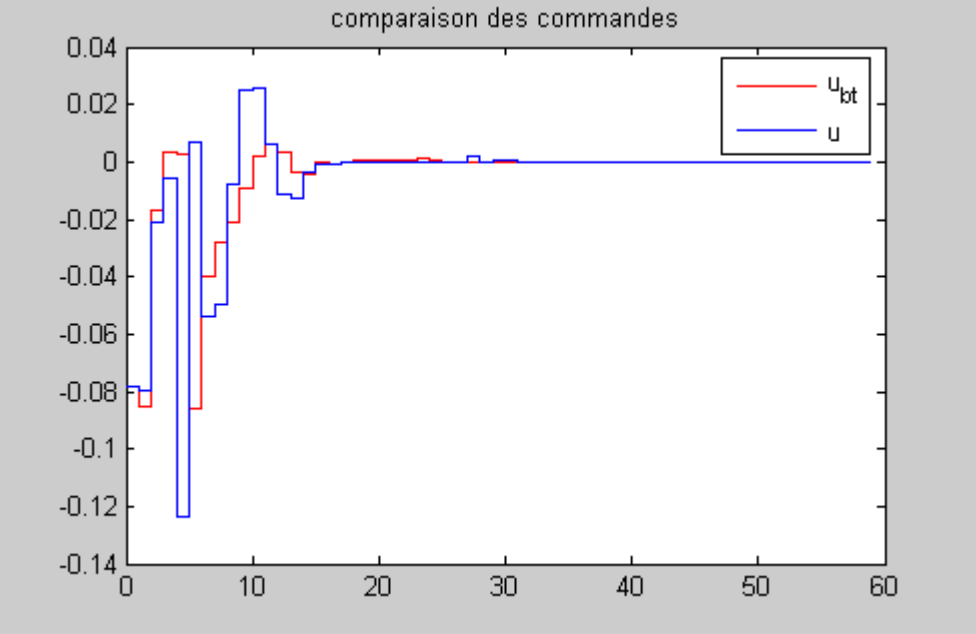
La première simulation a été faite sans bruit et avec conditions initiales $x_0 = 0.5 \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

Le premier graphique décrit l'évolution de l'état: en bleu on peut voir le comportement sans le contrôleur « bumpless » tandis que le signal rouge représente l'état quand le contrôleur est actif.

Cette notation sera préservée pour toutes les simulations.

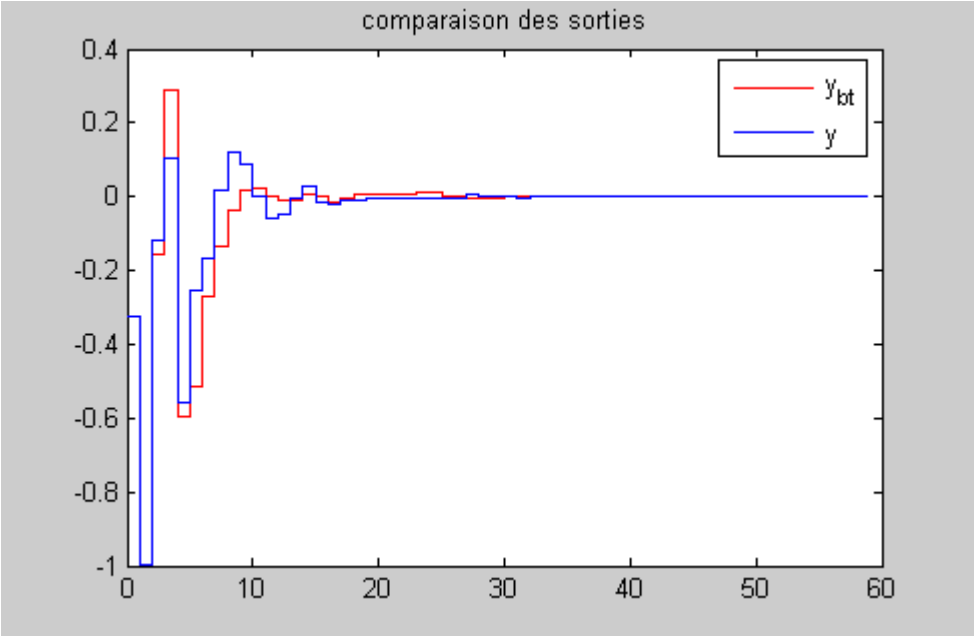


Le graphique suivant montre la différence entre $u_k = K_\alpha \hat{x}_k$ (c'est-à-dire l'évolution de la commande sans le K_α^{bt}, en bleu) et $u_k^{bt} = K_\alpha \hat{x}_k$ (en rouge):



On constate que dans les instants successifs aux commutations (surtout pour les premières 2 qui arrivent dans les instants 4 et 10) le système avec le K_α^{bt} a des sauts d'amplitude plus faible. Il est important de remarquer que le K_α^{bt} limite les sauts déjà au premier échantillon après la commutation.

Le dernier graphique montre l'évolution de la sortie:

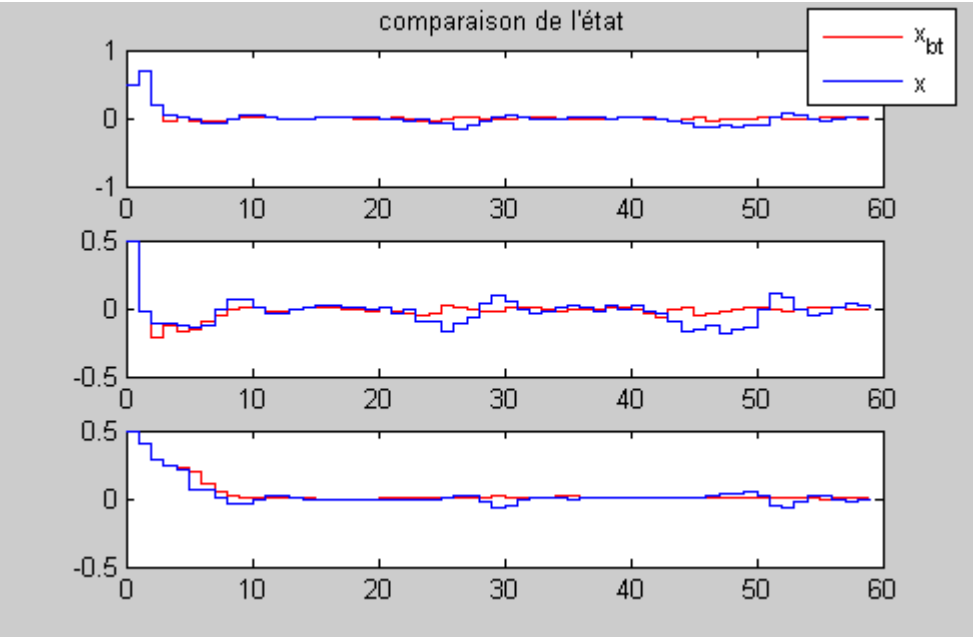


En effet dans l'instant 4 on peut constater que le système avec le K_α^{bt} a un saut plus fort mais ce problème est dû au transitoire et non pas à une commutation du système car dans cet exemple la première commutation se passe à l'instant 5.

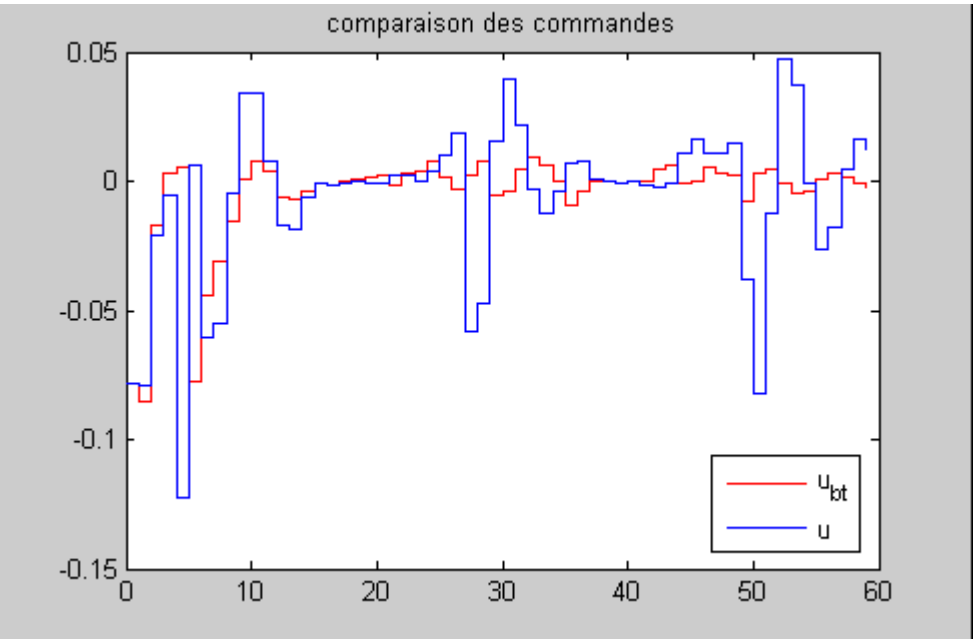
Si on ajoute du bruit les avantages du K_α^{bt} sont encore plus évidents comme on peut voir dans la suite.

5.1.3. Simulation 1.2

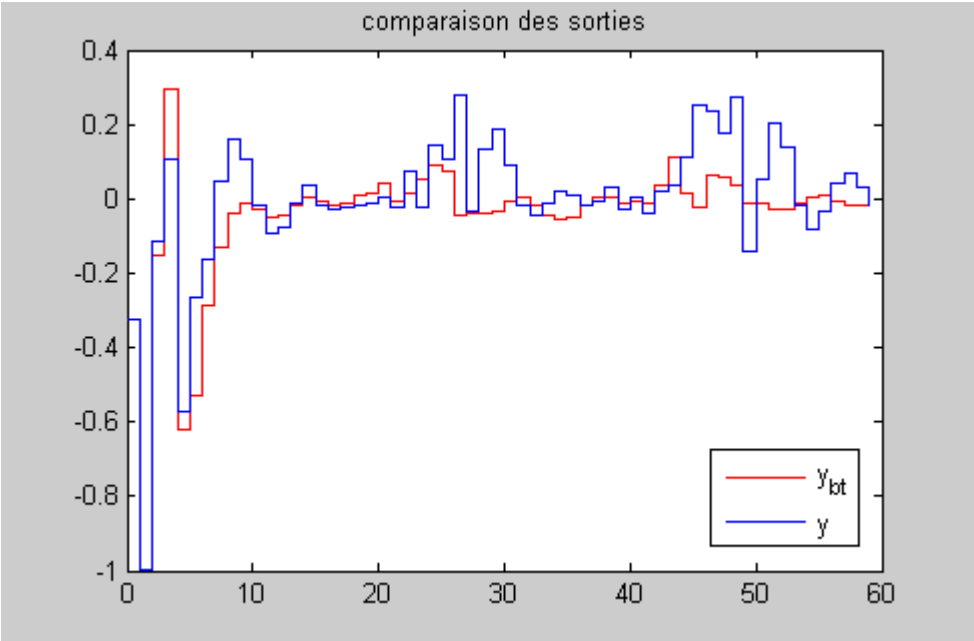
Dans cette deuxième simulation on a ajouté un bruit à distribution normal, moyenne nulle et variance unitaire d'amplitude $A_w = 10^{-3}$. Les conditions initiales sont toujours mises à $x_0 = 0.5 \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.



On peut déjà voir au niveau de l'état que quand le K_α^{bt} est actif (en rouge) l'évolution du signal est beaucoup plus régulière. Mais c'est en regardant l'évolution de la commande qu'on peut s'apercevoir que les choses marchent mieux avec le K_α^{bt} :



On peut voir que dans le cas sans K_α^{bt} (en bleu) le bruit amplifie beaucoup les sauts sur la commande tandis que si on ajoute le K_α^{bt} (en rouge) la commande suit une trajectoire similaire au cas sans bruit, c'est à dire que le système avec le K_α^{bt} possède une robustesse au bruit bien supérieure et il maintient sa capacité de limiter les sauts. Sur la sortie aussi on peut raisonner de façon analogue:

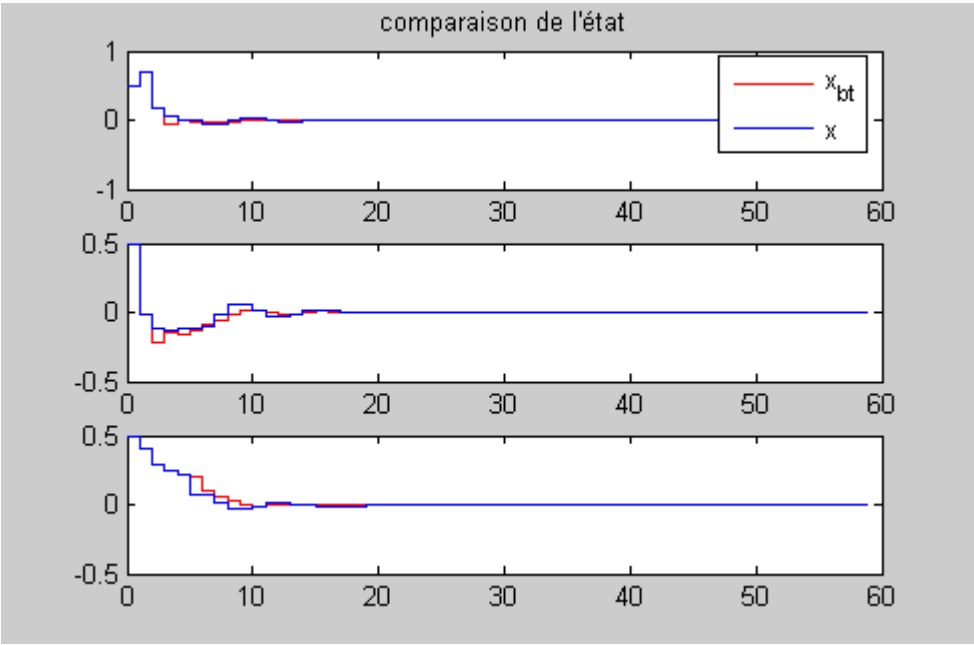


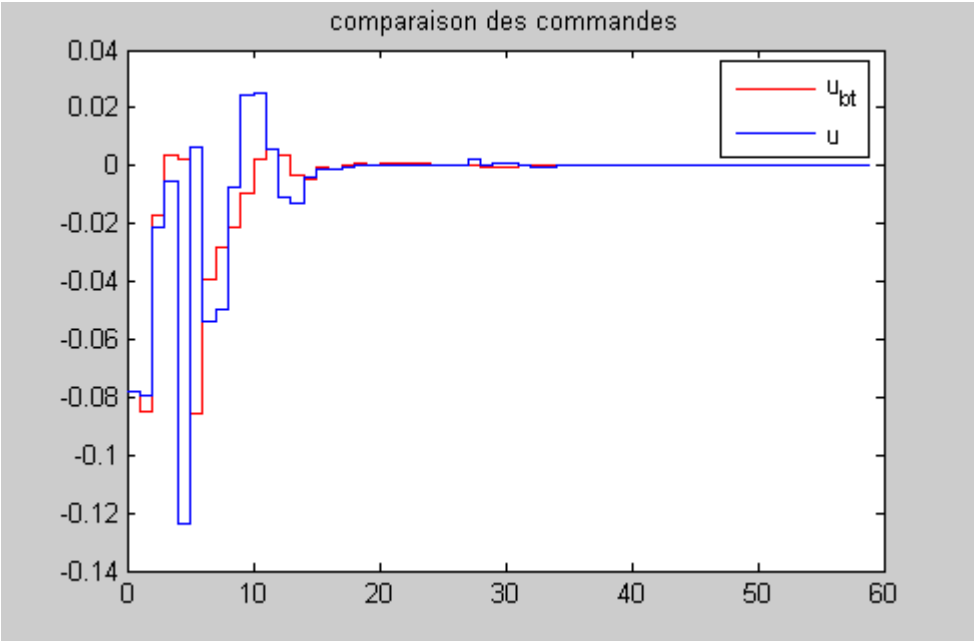
5.2. Contrôleur « bumpless » statique avec observateur additionnel

Le système sur lequel on testera le K_{α}^{bt} sera celui-ci du paragraphe précédent donc les matrices des contrôleurs seront les mêmes. Le signal de commutation n'a pas été changé:

5.2.1. Simulation 2.1

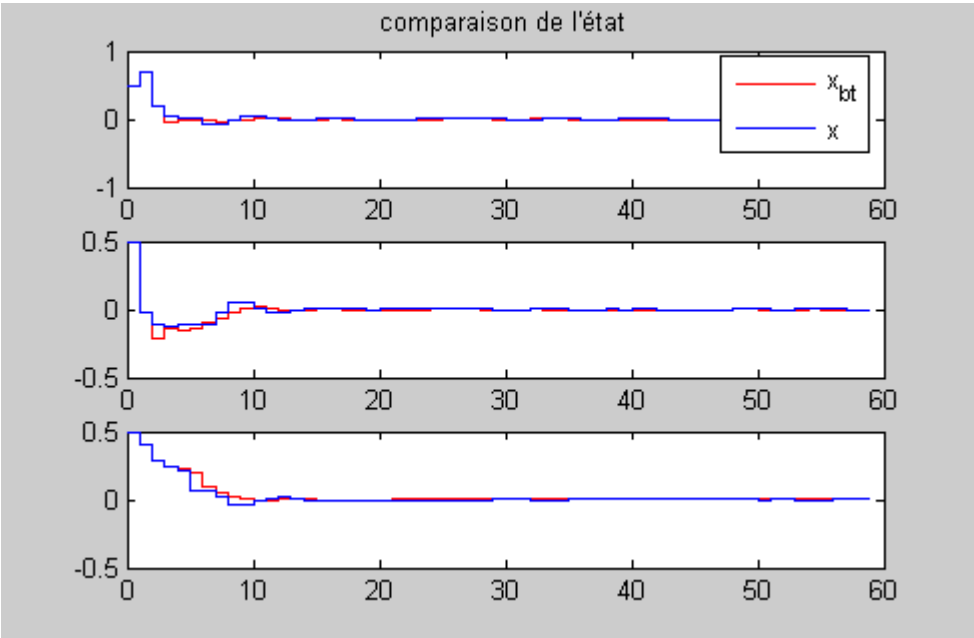
D'abord on a testé le système sans bruit et on peut constater comme les résultats sont très proches au cas précédent:

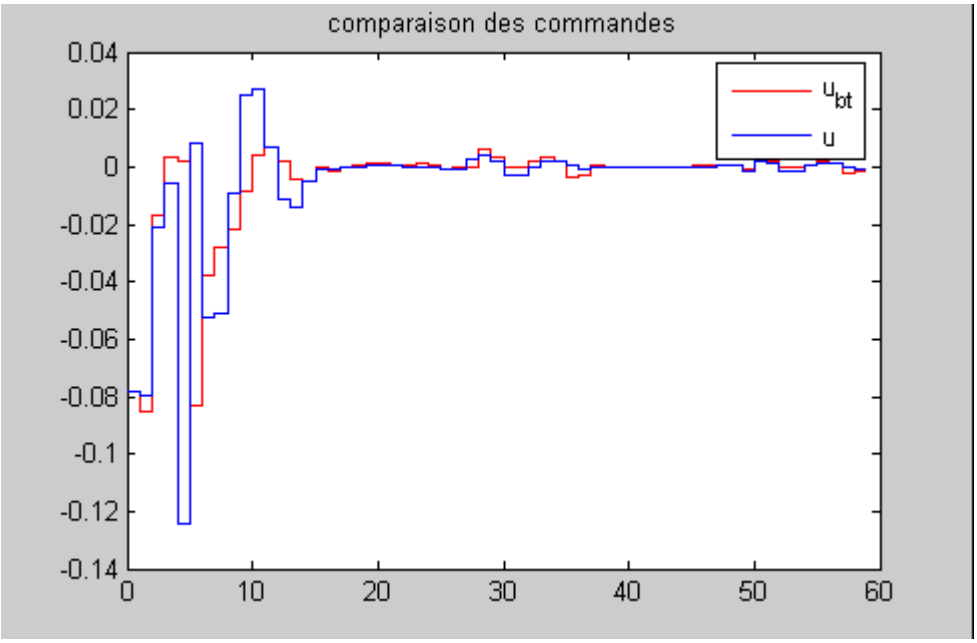


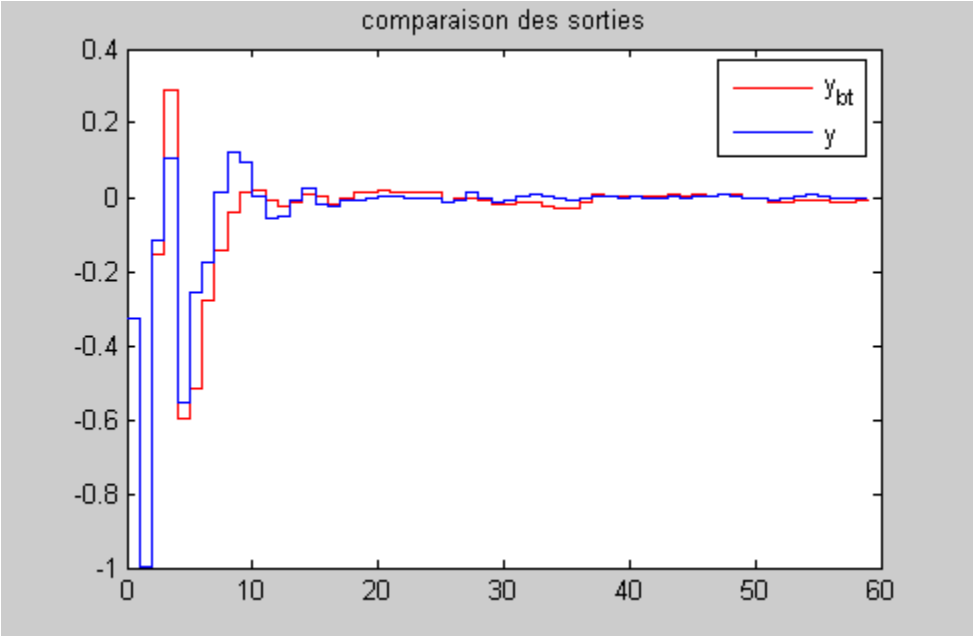


5.2.2. Simulation 2.2

Si on teste le système avec un bruit pas trop fort, par exemple avec amplitude $A_w = 0.5 \cdot 10^{-4}$ on voit comme le K_α^{bt} est encore efficace. Les avantages sont visibles surtout sur le graphique qui décrit l'évolution de la commande:

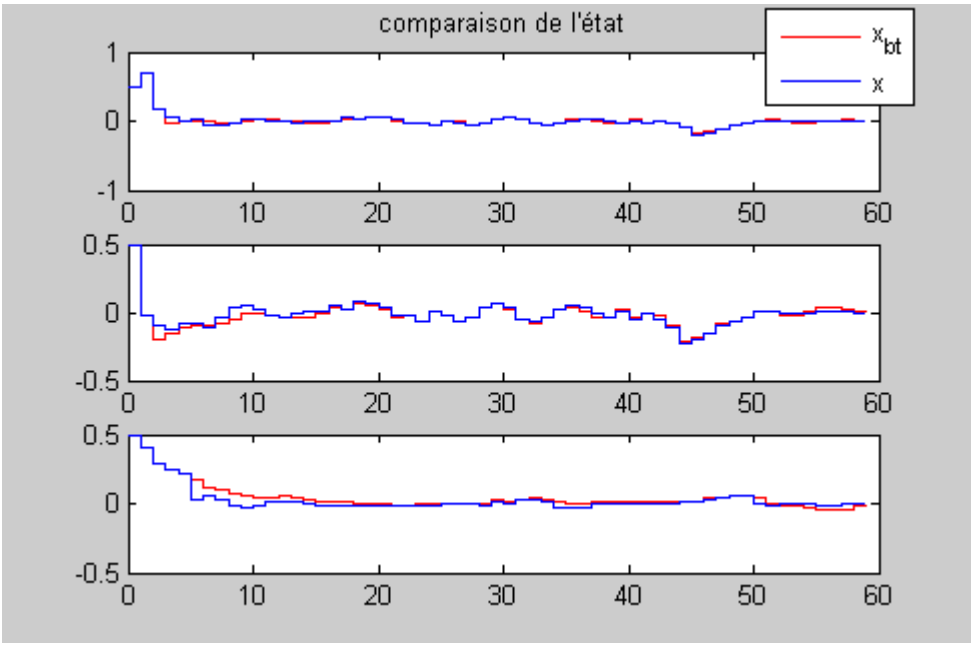


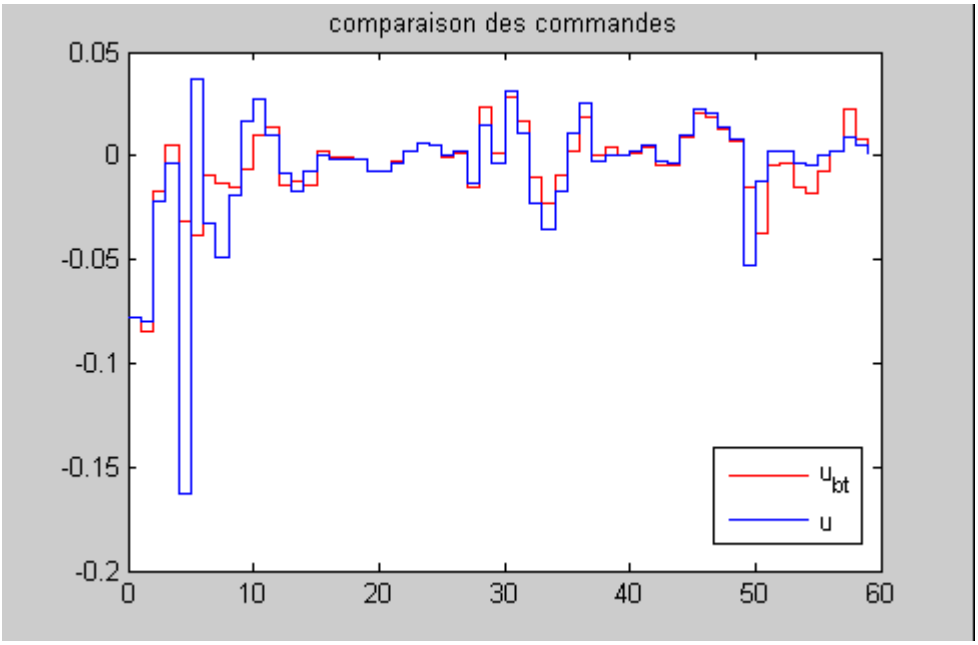

comparaison des commandes
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
-0.12
-0.14
0
10
20
30
40
50
60
u_{bt}
u

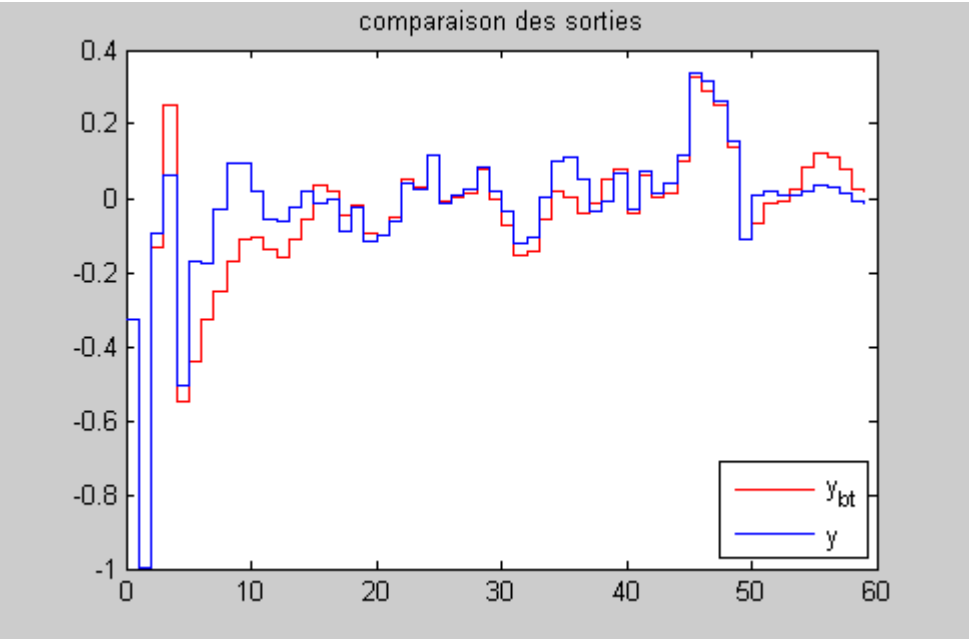

comparaison des sorties
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
10
20
30
40
50
60
y_{bt}
y

5.2.3. Simulation 2.3

Si par contre on ajoute un bruit avec une plus grande amplitude, par exemple $A_w = 10^{-3}$ comme dans le cas sans observateur additionnel, on voit que le K_α^{bt} perd presque toute sa capacité de limiter les sauts. Comme déjà expliqué, c'est le prix à payer pour éviter d'utiliser l'état *x* dans le schéma de correction.







5.3. Contrôleur « bumpless » dynamique

5.3.1. Description du système

En simulant le système du cas statique avec le K_α^{bt} dynamique on s'est aperçu que les LMIs du K_α^{bt} étaient infaisables. Pour simuler cette méthode, on a donc utilisé le système suivant:

$$A_1 = \begin{bmatrix} 0.1 & 1 \\ 0 & -0.3 \end{bmatrix},\ A_2 = \begin{bmatrix} -0.1 & 0 \\ 1 & -0.1 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},\quad B_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$B_1^w = \begin{bmatrix} 0 \\ 0.001 \end{bmatrix},\quad B_2^w = \begin{bmatrix} 0 \\ 0.001 \end{bmatrix}$$

$$C_1 = \begin{bmatrix} 1 & 0 \end{bmatrix},\quad C_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

$$D_1^w = \begin{bmatrix} 0 \end{bmatrix},\ D_2^w = \begin{bmatrix} 0 \end{bmatrix}$$

avec $i = 1,2$

On a simulé aussi le K_α^{bt} statique avec le même système pour donner une idée des résultats qu'on peut obtenir.

En appliquant les résultats du chapitre 2 on peut trouver les matrices du K_α^{nom} :

$$K_1 = \begin{bmatrix} 0 & 0.3 \end{bmatrix}$$

$$K_2 = \begin{bmatrix} -1 & 0.1 \end{bmatrix}$$

$$L_1 = \begin{bmatrix} -0.1648 \\ 0.0794 \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 0.0255 \\ -0.3546 \end{bmatrix}$$

Le K_α^{bt} aura une forme différente dans le cas du contrôleur statique ou dynamique:

dans le cas statique en utilisant le théorème 3.3 on trouve

$$\gamma = 1.0061$$

$$K_1^{bt} = \begin{bmatrix} 0.1828 & 0.1813 & -0.2704 & -1.0001 \\ -0.0812 & -0.0190 & 0.0799 & -0.0003 \end{bmatrix}$$

$$K_2^{bt} = \begin{bmatrix} -0.0057 & -0.0260 & 0.0901 & 0.0264 \\ 0.0071 & 0.3540 & -0.0047 & -0.3541 \end{bmatrix}$$

tandis que dans le cas dynamique en utilisant le théorème 3.5 on trouve

$$\gamma = 1.0002$$

$$A_1^c = \begin{bmatrix} -0.2558 & 1.3342 & -0.0209 & 0.2464 \\ 0.0071 & 0.1724 & 0.0016 & 0.0693 \\ 0.2333 & -1.2287 & 0.0148 & -0.2378 \\ 0.0130 & -0.3216 & 0.0074 & -0.1065 \end{bmatrix}$$

$$A_2^c = \begin{bmatrix} 0.4341 & -0.2522 & 0.0198 & -0.0093 \\ 1.1633 & -0.6401 & 0.0772 & -0.0255 \\ -0.2752 & 0.1665 & -0.0050 & 0.0058 \\ -1.4949 & 0.8265 & -0.1040 & 0.0250 \end{bmatrix}$$

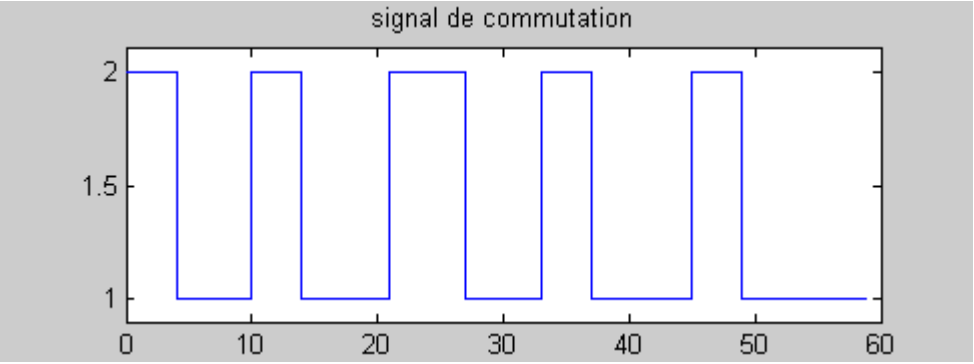
$$B_1^c = \begin{bmatrix} 0.0250 & -0.2614 \\ -0.0060 & -0.0791 \\ -0.0393 & 0.1798 \\ 0.0131 & 0.1118 \end{bmatrix}, \quad B_2^c = \begin{bmatrix} 0.0012 & 0.0117 \\ -0.0881 & 0.0376 \\ -0.0008 & -0.0073 \\ 0.0923 & -0.0687 \end{bmatrix}$$

$$C_1^c = 10^{-3}\begin{bmatrix} -0.0853 & 0.1196 & -0.0037 & 0.0913 \\ 0.0498 & -0.0218 & 0.0025 & -0.0018 \end{bmatrix},$$

$$C_2^c = 10^{-3}\begin{bmatrix} -0.0166 & 0.0211 & -0.0085 & 0.0012 \\ 0.1015 & -0.2638 & 0.0083 & -0.0133 \end{bmatrix}$$

$$D_1^c = 10^{-4}\begin{bmatrix} 0.0816 & -0.9889 \\ -0.0663 & -0.0560 \end{bmatrix}, \quad D_2^c = 10^{-4}\begin{bmatrix} 0.1939 & -0.0325 \\ -0.0951 & 0.2081 \end{bmatrix}$$

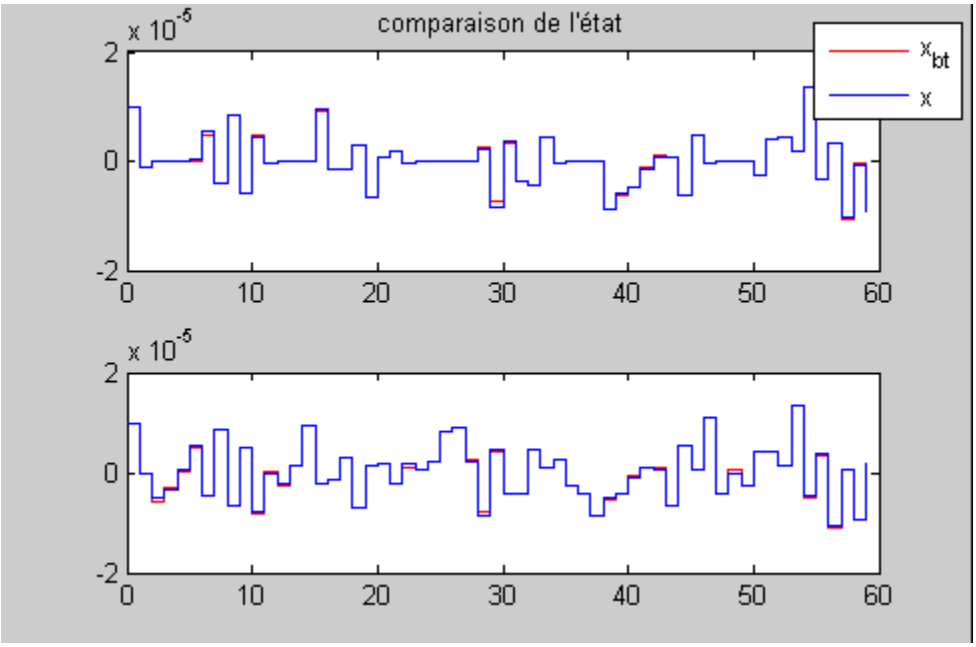
le signal de commutation sera le suivant:

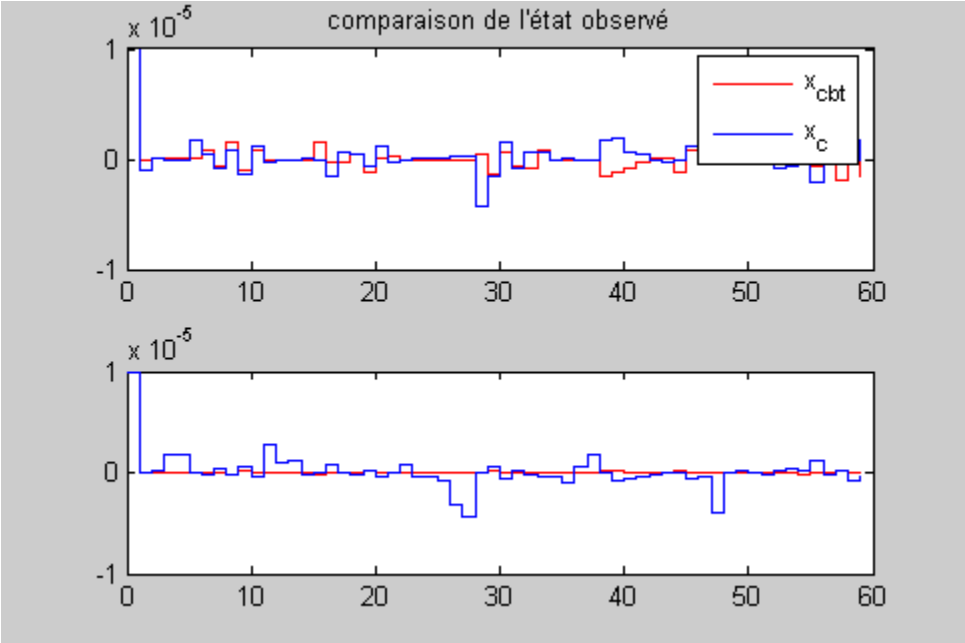


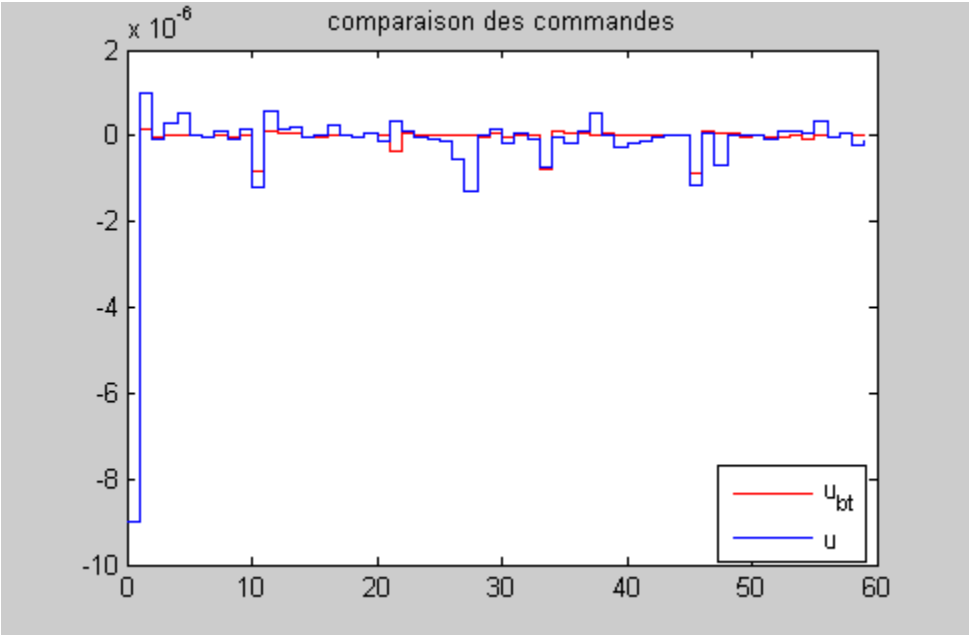
Les simulations ont été faites avec conditions initiales $x_0 = 10^{-5}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ et un bruit à distribution normal à moyen nulle et variance unitaire d'amplitude $A_w = 5 \cdot 10^{-4}$.

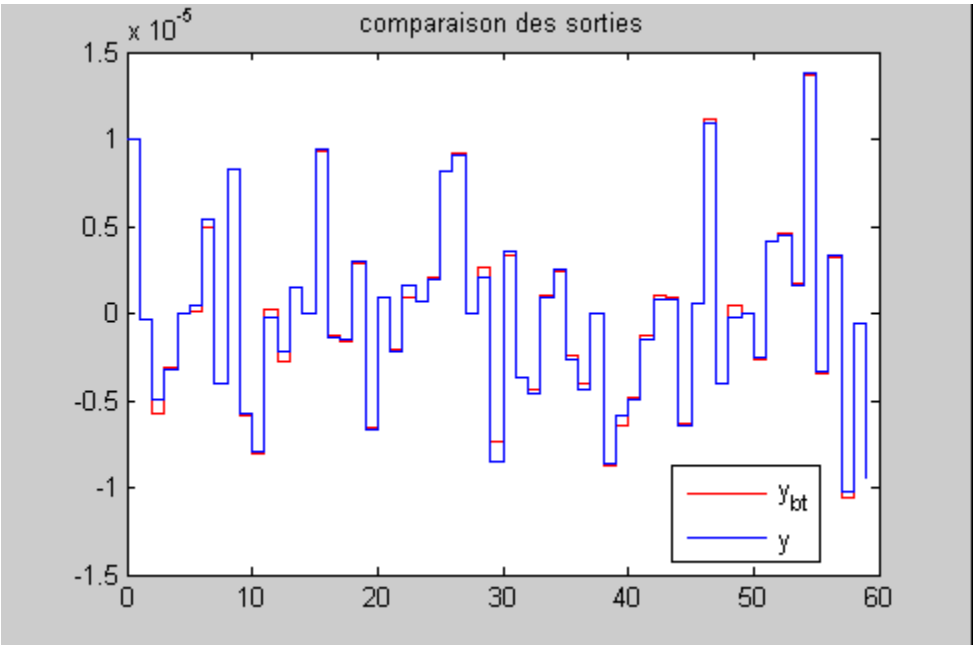
5.3.2. Simulation 3.1 (cas statique)

Dans ce cas on peut voir dans les graphiques suivants que les avantages du K_α^{bt} sont évidents sur la commande *u*, qui dépend de l'état observé $\hat{x}$, tandis qu'il n'y a pas de bénéfice sur la sortie *y*, qui dépend de l'état *x*. Ceci justifie l'affirmation faite auparavant, c'est-à-dire que dans beaucoup de cas l'observateur du K_α^{nom} n'est plus capable de bien estimer l'état du système à cause du fait qu'il est perturbé par la sortie *v* du K_α^{bt}. Ç'est ici qu'on trouve la différence entre utiliser *x* ou $\hat{x}$.



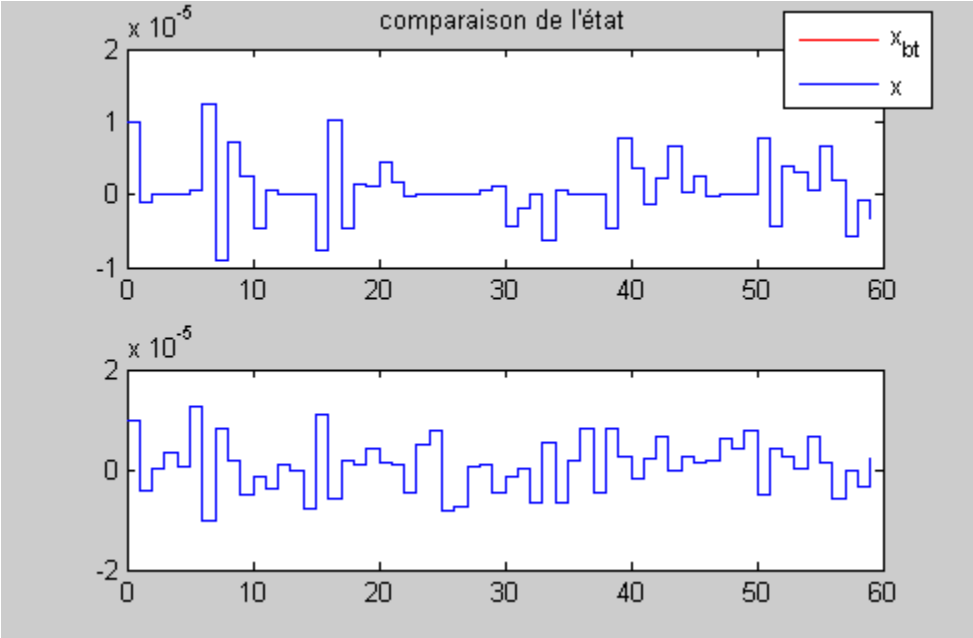


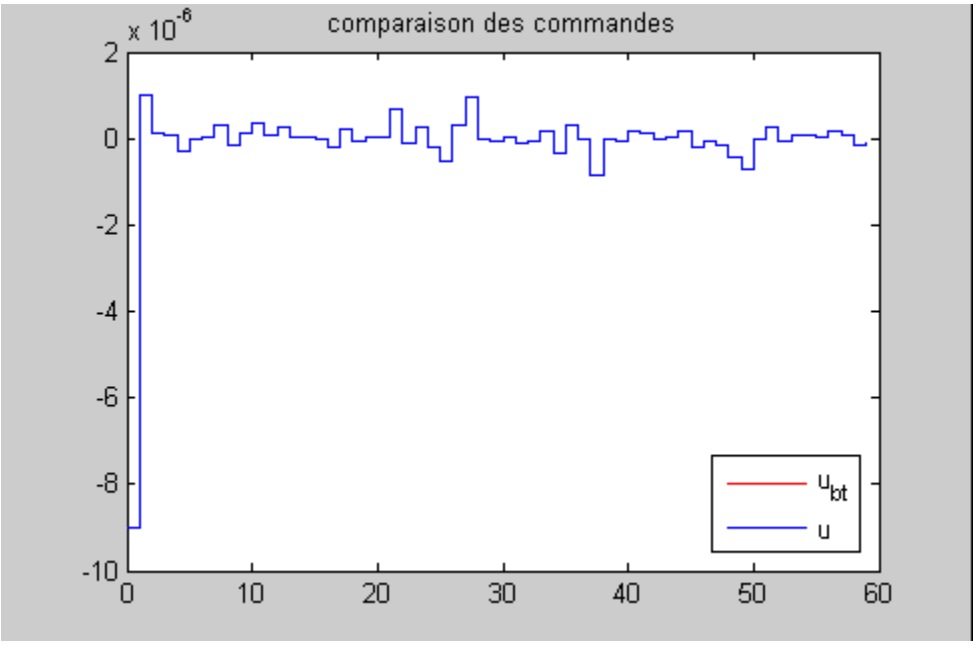

comparaison des commandes
x 10^-6
2
0
-2
-4
-6
-8
-10
0
10
20
30
40
50
60
u_bt
u


comparaison des sorties
x 10^-5
1.5
1
0.5
0
-0.5
-1
-1.5
0
10
20
30
40
50
60
y_bt
y

5.3.3. Simulation 3.2 (cas dynamique)

Dans le cas dynamique on n'a pas atteint des résultats satisfaisants, au moins avec ce système: on trouve que la commande *v* du K_{α}^{bt} est très faible ($\approx 10^{-9}$) pour influencer l'état *x* qui est de l'ordre de 10^{-5}.





On a décidé quand même de présenter la méthode car au niveau théorique elle présente une alternative pour le contrôleur statique.

Conclusions

Dans ce travail on a essayé d'aborder le problème du « bumpless transfer » pour des systèmes à commutation. Dans le cas LTI, on a vu que si le système contient une non-linéarité, par exemple une saturation, entre le processus et le contrôleur le problème peut être résolu aisément par des méthodes anti-windup classiques. À cette intention une méthode générale de modelage du contrôleur anti-windup (AWBT) et différents exemples d'application ont été présentés dans le chapitre 1.

Dans le cas de commutation de modèles et de correcteurs, les méthodes classiques ne sont pas suffisantes et ne peuvent pas être appliquées en l'état.

Pour résoudre le problème on a choisi de procéder à travers la synthèse de deux contrôleurs séparés: un contrôleur nominal qui s'occupe de stabiliser le système à commutation en boucle fermée et un contrôleur « bumpless transfer » qui s'occupe de limiter les sauts dus aux commutations. Pour la synthèse des contrôleurs, on a opté pour une approche basée sur les LMI car on dispose d'outils numériques efficaces pour la résolution. On a utilisé la LMI toolbox de Matlab.

En ce qui concerne le contrôleur « bumpless transfer » on s'est intéressé à une méthode de résolution qui utilise un critère d'optimisation. Le critère d'optimisation qu'on a choisi d'utiliser est celui de γ-performance. Il s'agit de minimiser le rapport entre la sortie permettant d'évaluer les performances et une entrée externe du système. Comme on est en temps discret, la sortie choisie pour la mesure des performances est la différence entre l'état observé $\hat{x}$ à l'instant k et la valeur de l'état observé à l'instant précédant la commutation. On arrive ainsi à minimiser les sauts sur l'état observé. Indirectement, dans certains cas, cette stratégie peut aussi limiter les sauts sur la commande $u = K_\alpha \hat{x}$. Cela montre que le travail fait ici ne résoud pas le problème dans le cas général et qu'il faut réfléchir à une stratégie qui agit directement sur les sauts de commande.

On a présenté trois méthodes: la première prévoit un contrôleur statique par retour d'état, c'est-à-dire qu'il contient la rétroaction de l'état réel du processus x et donc il a un intérêt théorique car en pratique l'état n'est pas connu. Pour contourner ce problème, dans la deuxième méthode présentée on a ajouté un second observateur qui à partir de la sortie du système y et de la commande u il produit une estimation de l'état x. Dans ce cas le système est moins robuste au bruit mais il peut être réalisé physiquement.

La troisième méthode prévoit un contrôleur « bumpless » dynamique pour lequel la connaissance de la valeur de l'état n'est pas nécessaire. Comme déjà souligné on n'a pas réussi à obtenir des résultats satisfaisants avec cette dernière méthode mais on a décidé de la préserver car d'un point de vue théorique elle se présente comme une alternative à la méthode statique qu'il faudrait explorer pour trouver des schémas qui agissent directement sur les sauts de commande.

Pour les développements futurs, l'étude faite montre qu'ils sont nombreux car le problème du « bumpless transfer » est toujours d'actualité à cause de l'importance des systèmes hybrides aujourd'hui. Naturellement ce travail représente seulement un petit pas vers la définitive résolution du problème aussi si dans certains cas représente une solution satisfaisante.

Annexe A

Linear Matrix Inequalities

Une inégalité linéaire matricielle a la forme

$$F(x) = F_0 + \sum_{i=1}^{m} x_i F_i > 0 \tag{A.1}$$

où $x \in \Re^m$ est la variable et les matrices symétriques $F_i = F_i^T \in \Re^{nxn}$, *i=0,...,m* sont données.

On peut avoir aussi des LMIs non stricts de la forme

$$F(x) \geq 0 \tag{A.2}$$

Dans la suite on considérera les LMIs dans la forme stricte.

La LMI (A.1) est une contrainte convexe en *x*, c'est à dire que l'ensemble $H : \{x / F(x) > 0\}$ est convexe. Par exemple si on donne $x', x'' \in H$ et $\alpha \in (0,1)$ on trouve

$$F(\alpha x' + (1-\alpha)x'') = \alpha F(x') + (1-\alpha)F(x'') > 0$$

Une LMI multiple $F^{(1)}(x) > 0, ..., F^{(p)}(x) > 0$ peut être remplacée par une seule LMI de la forme

$$F(x) = \begin{bmatrix} F^{(1)}(x) & 0 & \cdots & 0 \\ 0 & F^{(2)}(x) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & F^{(p)}(x) \end{bmatrix} > 0$$

Les inégalités non-linéaires convexes peuvent être converties en LMI en utilisant le complément de Schur. L'idée de base est la suivante: la LMI

$$\begin{bmatrix} Q(x) & S(x) \\ S(x)^T & R(x) \end{bmatrix} > 0, \tag{A.3}$$

avec *Q(x)* et *R(x)* symétriques, est équivalente à la forme

$$\begin{cases} R(x) > 0 \\ Q(x) - S(x)R(x)^{-1}S(x)^T > 0 \end{cases} \tag{A.4}$$

c'est à dire l'ensemble d'inégalités non-linéaires (A.4) peut être représenté par la LMI (A.3).

Annexe B

Matlab LMI toolbox

Dans ce travail on a ramené tous les problèmes à une forme LMI et on a utilisé la boite à outils LMI de Matlab. Le but de cette annexe est d'expliquer la syntaxe:

pas 1: initialisation

`setlmis([])` mets à zéro les variables internes utilisées pour créer une LMI de façon qu'on puisse créer un nouveau système LMI.

`ineg_i = newlmi` ajoute une nouvelle LMI au système LMI en lui donnant l'étiquette `ineg_i`.

pas 2: définition d'une nouvelle variable

`varX = lmivar(TYPE,STRUCT)` ajoute une nouvelle variable matricielle au système LMI spécifié dans le pas 1. Il faut définir deux entrées:

`TYPE` spécifie la structure de `varX`:
1: matrice symétrique à blocs diagonaux
2: matrice rectangulaire pleine
3: autres types de matrices

`STRUCT` spécifie des donnés additionnels à la structure de varX:

`TYPE 1:` la $i^{ème}$ ligne de `STRUCT` décrit le $i^{ème}$ bloc diagonal
`STRUCT(i,1):` dimension du bloc
`STRUCT(i,2):` type du bloc (0 pour un bloc scalaire, 1 pour un bloc plein et –1 pour un block de zéros)

`TYPE 2:` `STRUCT=[N,M]` si `varX` est une matrice `NxM`

`TYPE 3:` `STRUCT` est une matrice avec les mêmes dimensions de `varX` où `STRUCT(i,j)` peut être
`0` if `X(i,j) = 0`
`+n` if `X(i,j)` = $n^{ème}$ variable décisionnel
`-n` if `X(i,j)` = $(-1)\cdot n^{ème}$ variable décisionnel

pas 3: définition des termes

`lmiterm([ineg_i i j var],A,B,flag)` ajoute un terme à une LMI donnée. Le terme peut être une matrice constante, un terme avec une variable `A*varX*B` ou `A*varX'*B` ou un autre terme.

À noter que les blocs hors-diagonal (i,j) et (j,i) sont les transposé l'un de l'autre donc il faut en spécifier seulement un.

On a 4 entrées:
la première `ineg_i` spécifie à quelle LMI appartient le terme
il faut mettre un – devant l'étiquette si l'LMI est défini positive

les entrées i et j spécifient la position du bloc à l'intérieur du LMI. Si on a un autre terme il faut mettre `[i j] = [0 0]`

l'entrée `var` définit le type de terme:

```
      0          ->   terme constante
     varX        ->   A*varX*B
    -varX        ->   A*varX'*B
```

où `varX` est la variable retourné par `lmivar`

`A` peut être la valeur d'un autre facteur, un terme constant ou le coefficient de gauche dans le terme `A*X*B` ou `A*X'*B`

`B` est le coefficient de droite dans le terme `A*X*B` ou `A*X'*B`

`flag='s'` permet de spécifier avec une seule commande l'expression
`A*X*B+B'*X'*A'` dans un bloc diagonal

pas 4: achèvement de l'implémentation

`ProblemLMI = getlmis` retourne une description complète du problème LMI implémenté.

Pour trouver la solution numérique du problème il faut utiliser la fonction

`[tmin xsol] = feasp(ProblemLMI)` qui trouve une solution faisable

ou

`[copt xopt] = mincx(ProblemLMI,c)` qui trouve la solution optimale avec la contraint `c`

Annexe C

Code Matlab

Cas statique

```
function static_switched_control(A1,A2,B1,B2,C1,C2,D1,D2)

clear all
close all

reponse = input('\nvoulez-vous utiliser l''observateur supplementaire?(oui/non):','s');

if (reponse == 'oui')
      flag_obs = 1;
elseif (reponse == 'non')
      flag_obs = 0;
else
      flag_obs = 0;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calcul des contrôleurs  et construction des matrices du système en boucle fermée%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


[K1 K2] = lmi_sol_K(A1, A2, B1, B2, n, r);
[L1 L2] = lmi_sol_L(A1, A2, C1, C2, n, m);


A1_bt = [A1 B1*K1; L1*C1 A1+B1*K1-L1*C1];
A2_bt = [A2 B2*K2; L2*C2 A2+B2*K2-L2*C2];

B1_bt = [zeros(n,n); eye(n)];
B2_bt = [zeros(n,n); eye(n)];

B1_w_bt  = [B1_w zeros(n) ; zeros(n,n+r)];
B2_w_bt  = [B2_w zeros(n) ; zeros(n,n+r)];

C1_bt  = eye(2*n);
C2_bt  = eye(2*n);

D1_y   = zeros(2*n,n+r);
D2_y   = zeros(2*n,n+r);
D1_w_bt  = -[zeros(n,n+r); zeros(n,r)  eye(n)];
D2_w_bt  = -[zeros(n,n+r); zeros(n,r)  eye(n)];

[n_bt n_bt] = size(A1_bt);
[n_bt r_bt] = size(B1_bt);
[m_bt n_bt] = size(C1_bt);

[K1_bt K2_bt] = lmi_sol_K_bt(A1_bt, A2_bt, B1_bt, B2_bt, B1_w_bt, B2_w_bt, C1_bt, C2_bt, ...
                                        D1_y, D2_y, D1_w_bt, D2_w_bt, n_bt, r_bt, m_bt);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%               initialisation des variables              %%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

T_fin         = 60;
w             = 0.00001*randn(1, T_fin);
x(:,1)        = 0.5*ones(n,1);
x_c(:,1)      = x(:,1);
x_bt(:,1)     = x(:,1);
```

```
x_c_bt(:,1)  = x(:,1);
x_obs(:,1)   = x(:,1);
x_tilde      = zeros(n,1);


alfa  = [1 0];
comm  = 0;
flag  = 0;
stop_comm = 0;
des_contr = 7;


for i = 1:T_fin
    T(i)=i-1;

    alfa_prec = alfa;
    if (i <5 || i>15 && i<28 || i>37 && i<50)
        alfa = [0 1];
    else
       alfa = [1 0];
    end

    if (alfa*alfa_prec'==1) comm = 0;
    else comm = 1;
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%       boucle sans le contrôleur bumpless transfer        %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    X_succ   = alfa(1)*(A1_bt*[x(:,i); x_c(:,i)] + [B1_w; zeros(n,r)]*w(:,i)) + ...
               alfa(2)*(A2_bt*[x(:,i); x_c(:,i)] + [B2_w; zeros(n,r)]*w(:,i));

    x(:,i+1)     = X_succ(1:n,1);
    x_c(:,i+1)  = X_succ(n+1:2*n,1);

    u(:,i)       = alfa(1)*K1*x_c(:,i)  + alfa(2)*K2*x_c(:,i);

    y(:,i)       = alfa(1)*(C1*x(:,i)+D1*w(:,i)) + alfa(2)*(C2*x(:,i)+D2*w(:,i));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%         boucle avec le contrôleur bumpless transfer     %%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if (comm == 1 && i ~= 1)
        x_tilde = x_c_bt(:,i);
        flag = 1;
        stop_comm = i;
    end

    if (flag==1 && i-stop_comm == des_contr)
        x_tilde = zeros(n, 1);
        flag = 0;
    end

        W = [w(:,i); x_tilde];

        u_bt(:,i)       = alfa(1)*K1*x_c_bt(:,i) + alfa(2)*K2*x_c_bt(:,i);

    if (flag_obs == 1)
        x_obs(:,i+1)    = alfa(1)*(A1*x_obs(:,i) + B1*u_bt(:,i) + ...
                                           L1*(C1*x_bt(:,i)-C1*x_obs(:,i))) + ...
                          alfa(2)*(A2*x_obs(:,i) + B2*u_bt(:,i) + ...
                                    L2*(C2*x_bt(:,i)-C2*x_obs(:,i)));

        z(:,i)          = alfa(1)*(C1_bt*[x_obs(:,i);x_c_bt(:,i)] + D1_w_bt*W) + ...
                          alfa(2)*(C2_bt*[x_obs(:,i);x_c_bt(:,i)] + D2_w_bt*W);
    else
        z(:,i)          = alfa(1)*(C1_bt*[x_bt(:,i);x_c_bt(:,i)] + D1_w_bt*W) + ...
                          alfa(2)*(C2_bt*[x_bt(:,i);x_c_bt(:,i)] + D2_w_bt*W);
    end
```

```
        v(:,i)          = alfa(1)*K1_bt*z(:,i) + alfa(2)*K2_bt*z(:,i);

        X_succ_bt       = alfa(1)*(A1_bt*[x_bt(:,i); x_c_bt(:,i)] + ...
                                   B1_bt*v(:,i) + B1_w_bt*W) + ...
                          alfa(2)*(A2_bt*[x_bt(:,i); x_c_bt(:,i)] + ...
                                           B2_bt*v(:,i) + B2_w_bt*W);

        x_bt(:,i+1)     = X_succ_bt(1:n,1);
        x_c_bt(:,i+1)   = X_succ_bt(n+1:2*n,1);

        y_bt(:,i)       = alfa(1)*(C1*x_bt(:,i) + 1*w(:,i))+...
                               alfa(2)*(C2*x_bt(:,i) + D2*w(:,i));

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%                         LMI                              %%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%contrôleur nominal

function [K1 K2] = lmi_sol_K(A1, A2, B1, B2, n, r);

    setlmis([]);

    varS1 = lmivar(1,[n,1]);
    varS2 = lmivar(1,[n,1]);
    varR1 = lmivar(2,[r,n]);
    varR2 = lmivar(2,[r,n]);
    varG1 = lmivar(2,[n,n]);
    varG2 = lmivar(2,[n,n]);

    ineg1 = newlmi;
    ineg2 = newlmi;
    ineg3 = newlmi;
    ineg4 = newlmi;

    lmiterm([ineg1 1 1 varG1],-1,1,'s');
    lmiterm([ineg1 1 1 varS1],1,1);
    lmiterm([ineg1 2 1 varG1],-A1,1);
    lmiterm([ineg1 2 1 varR1],-B1,1);
    lmiterm([ineg1 2 2 varS1],-1,1);

    lmiterm([ineg2 1 1 varG1],-1,1,'s');
    lmiterm([ineg2 1 1 varS1],1,1);
    lmiterm([ineg2 2 1 varG1],-A1,1);
    lmiterm([ineg2 2 1 varR1],-B1,1);
    lmiterm([ineg2 2 2 varS2],-1,1);

    lmiterm([ineg3 1 1 varG2],-1,1,'s');
    lmiterm([ineg3 1 1 varS2],1,1);
    lmiterm([ineg3 2 1 varG2],-A2,1);
    lmiterm([ineg3 2 1 varR2],-B2,1);
    lmiterm([ineg3 2 2 varS1],-1,1);

    lmiterm([ineg4 1 1 varG2],-1,1,'s');
    lmiterm([ineg4 1 1 varS2],1,1);
    lmiterm([ineg4 2 1 varG2],-A2,1);
    lmiterm([ineg4 2 1 varR2],-B2,1);
    lmiterm([ineg4 2 2 varS2],-1,1);

    ProblemLMI = getlmis;

    [tmin_K xsol] = feasp(ProblemLMI);
    tmin_K

    R1 = dec2mat(ProblemLMI,xsol,varR1);
    R2 = dec2mat(ProblemLMI,xsol,varR2);

    G1 = dec2mat(ProblemLMI,xsol,varG1);
    G2 = dec2mat(ProblemLMI,xsol,varG2);
```

```
    K1 = R1*inv(G1)
    K2 = R2*inv(G2)
end


function [L1 L2] = lmi_sol_L(A1, A2, C1, C2, n, m);

    setlmis([]);

    varS1 = lmivar(1,[n,1]);
    varS2 = lmivar(1,[n,1]);
    varF1 = lmivar(2,[m,n]);
    varF2 = lmivar(2,[m,n]);
    varG1 = lmivar(2,[n,n]);
    varG2 = lmivar(2,[n,n]);

    ineg1 = newlmi;
    ineg2 = newlmi;
    ineg3 = newlmi;
    ineg4 = newlmi;

    lmiterm([ineg1 1 1 varG1],-1,1,'s');
    lmiterm([ineg1 1 1 varS1],1,1);
    lmiterm([ineg1 2 1 varG1],-A1',1);
    lmiterm([ineg1 2 1 varF1],C1',1);
    lmiterm([ineg1 2 2 varS1],-1,1);

    lmiterm([ineg2 1 1 varG1],-1,1,'s');
    lmiterm([ineg2 1 1 varS1],1,1);
    lmiterm([ineg2 2 1 varG1],-A1',1);
    lmiterm([ineg2 2 1 varF1],C1',1);
    lmiterm([ineg2 2 2 varS2],-1,1);

    lmiterm([ineg3 1 1 varG2],-1,1,'s');
    lmiterm([ineg3 1 1 varS2],1,1);
    lmiterm([ineg3 2 1 varG2],-A2',1);
    lmiterm([ineg3 2 1 varF2],C2',1);
    lmiterm([ineg3 2 2 varS1],-1,1);

    lmiterm([ineg4 1 1 varG2],-1,1,'s');
    lmiterm([ineg4 1 1 varS2],1,1);
    lmiterm([ineg4 2 1 varG2],-A2',1);
    lmiterm([ineg4 2 1 varF2],C2',1);
    lmiterm([ineg4 2 2 varS2],-1,1);

    ProblemLMI = getlmis;

    [tmin_L xsol] = feasp(ProblemLMI);
    tmin_L

    F1 = dec2mat(ProblemLMI,xsol,varF1);
    F2 = dec2mat(ProblemLMI,xsol,varF2);
    G1 = dec2mat(ProblemLMI,xsol,varG1);
    G2 = dec2mat(ProblemLMI,xsol,varG2);

    L1 = inv(G1')*F1'
    L2 = inv(G2')*F2'

end


%controleur bumpless transfer statique avec minimisation de gamma

function [K1_bt K2_bt] = lmi_sol_K_bt(A1_bt, A2_bt, B1_bt, B2_bt, B1_w, B2_w, ...
                                      C1_z, C2_z,D1_z, D2_z, D1_zw, D2_zw, n, r, m);

    setlmis([]);

    varGAMMA = lmivar(1,[1,0]);
    varS1 = lmivar(1,[n,1]);
    varS2 = lmivar(1,[n,1]);
    varG1 = lmivar(2,[n,n]);
```

```
varG2 = lmivar(2,[n,n]);
varR1 = lmivar(2,[r,n]);
varR2 = lmivar(2,[r,n]);

ineg1 = newlmi;
ineg2 = newlmi;
ineg3 = newlmi;
ineg4 = newlmi;

lmiterm([-ineg1 1 1 varG1],1,1,'s');
lmiterm([-ineg1 1 1 varS1],-1,1);
lmiterm([-ineg1 2 1 0],zeros(r,n));
lmiterm([-ineg1 2 2 varGAMMA],1,1);
lmiterm([-ineg1 3 1 varG1],A1_bt,1);
lmiterm([-ineg1 3 1 varR1],B1_bt,1);
lmiterm([-ineg1 3 2 0],B1_w);
lmiterm([-ineg1 3 3 varS1],1,1);
lmiterm([-ineg1 4 1 varG1],C1_z,1);
lmiterm([-ineg1 4 1 varR1],D1_z,1);
lmiterm([-ineg1 4 2 0],D1_zw);
lmiterm([-ineg1 4 3 0],zeros(m,n));
lmiterm([-ineg1 4 4 varGAMMA],1,1);

lmiterm([-ineg2 1 1 varG1],1,1,'s');
lmiterm([-ineg2 1 1 varS1],-1,1);
lmiterm([-ineg2 2 1 0],zeros(r,n));
lmiterm([-ineg2 2 2 varGAMMA],1,1);
lmiterm([-ineg2 3 1 varG1],A1_bt,1);
lmiterm([-ineg2 3 1 varR1],B1_bt,1);
lmiterm([-ineg2 3 2 0],B1_w);
lmiterm([-ineg2 3 3 varS2],1,1);
lmiterm([-ineg2 4 1 varG1],C1_z,1);
lmiterm([-ineg2 4 1 varR1],D1_z,1);
lmiterm([-ineg2 4 2 0],D1_zw);
lmiterm([-ineg2 4 3 0],zeros(m,n));
lmiterm([-ineg2 4 4 varGAMMA],1,1);

lmiterm([-ineg3 1 1 varG2],1,1,'s');
lmiterm([-ineg3 1 1 varS2],-1,1);
lmiterm([-ineg3 2 1 0],zeros(r,n));
lmiterm([-ineg3 2 2 varGAMMA],1,1);
lmiterm([-ineg3 3 1 varG2],A2_bt,1);
lmiterm([-ineg3 3 1 varR2],B2_bt,1);
lmiterm([-ineg3 3 2 0],B2_w);
lmiterm([-ineg3 3 3 varS1],1,1);
lmiterm([-ineg3 4 1 varG2],C2_z,1);
lmiterm([-ineg3 4 1 varR2],D2_z,1);
lmiterm([-ineg3 4 2 0],D2_zw);
lmiterm([-ineg3 4 3 0],zeros(m,n));
lmiterm([-ineg3 4 4 varGAMMA],1,1);

lmiterm([-ineg4 1 1 varG2],1,1,'s');
lmiterm([-ineg4 1 1 varS2],-1,1);
lmiterm([-ineg4 2 1 0],zeros(r,n));
lmiterm([-ineg4 2 2 varGAMMA],1,1);
lmiterm([-ineg4 3 1 varG2],A2_bt,1);
lmiterm([-ineg4 3 1 varR2],B2_bt,1);
lmiterm([-ineg4 3 2 0],B2_w);
lmiterm([-ineg4 3 3 varS2],1,1);
lmiterm([-ineg4 4 1 varG2],C2_z,1);
lmiterm([-ineg4 4 1 varR2],D2_z,1);
lmiterm([-ineg4 4 2 0],D2_zw);
lmiterm([-ineg4 4 3 0],zeros(m,n));
lmiterm([-ineg4 4 4 varGAMMA],1,1);

ProblemLMI = getlmis;

sum = count(n);

c = [1 zeros(1,2*sum + 2*n^2 + 2*r*n)];

[gamma xsol] = mincx(ProblemLMI,c);
```

```
    gamma
    G1 = dec2mat(ProblemLMI,xsol,varG1);
    G2 = dec2mat(ProblemLMI,xsol,varG2);
    R1 = dec2mat(ProblemLMI,xsol,varR1);
    R2 = dec2mat(ProblemLMI,xsol,varR2);

    K1_bt = R1*inv(G1)
    K2_bt = R2*inv(G2)
end

function [sum] = count(n)

    sum = 0;
    for i = 1:n-1
        sum = sum + n-i;
    end
    sum = sum + n;
end

end
```

Cas dynamique

```
function dynamic_switched_control(A1,A2,B1,B2,C1,C2,D1,D2)

clear all
close all


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calcul des contrôleurs et construction des matrices du système en boucle fermée %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


[K1 K2] = lmi_sol_K(A1, A2, B1, B2, n, r);
[L1 L2] = lmi_sol_L(A1, A2, C1, C2, n, m);


A1_bt = [A1 B1*K1; L1*C1 A1+B1*K1-L1*C1];
A2_bt = [A2 B2*K2; L2*C2 A2+B2*K2-L2*C2];

B1_bt = [zeros(n,n); eye(n)];
B2_bt = [zeros(n,n); eye(n)];

B1_w_bt  = [B1_w zeros(n); zeros(n,n+r)];
B2_w_bt  = [B2_w zeros(n); zeros(n,n+r)];

C1_bt  = [zeros(n) eye(n)];
C2_bt  = [zeros(n) eye(n)];

C1_u = [zeros(n) eye(n)];
C2_u = [zeros(n) eye(n)];

D1_u   =  zeros(n,r+n);
D2_u   =  zeros(n,r+n);
D1_bt  = -[zeros(n,r) eye(n)];
D2_bt  = -[zeros(n,r) eye(n)];
D1_uw  = -[zeros(n,r) eye(n)];
D2_uw  = -[zeros(n,r) eye(n)];


[n_bt n_bt] = size(A1_bt);
[n_bt r_bt] = size(B1_bt);
[m_bt n_bt] = size(C1_bt);

[A1_c A2_c B1_c B2_c C1_c C2_c D1_c D2_c] = lmi_sol_K_bt(A1_bt, A2_bt, B1_bt, ...
                    B2_bt, B1_w_bt, B2_w_bt, C1_bt, C2_bt, C1_u, C2_u, D1_bt, ...
                         D2_bt, D1_u, D2_u, D1_uw, D2_uw, n_bt, r_bt, m_bt);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%              initialisation des variables               %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

T_fin        = 60;
w            = 0.0005*randn(1, T_fin);
x(:,1)       = 0.00001*ones(n,1);
x_c(:,1)     = x(:,1);
x_bt(:,1)    = x(:,1);
x_c_bt(:,1)  = x(:,1);
eta(:,1)     = zeros(n_bt, 1);
x_tilde      = zeros(n,1);

alfa  = [1 0];
comm  = 0;
flag  = 0;
stop_comm = 0;
des_contr = 5;


for i = 1:T_fin
    T(i)=i-1;

    alfa_prec = alfa;
    if (i <5 || i>10 && i<15 || i>21 && i<28 || i>33 && i<38 || i>45 && i<50)
        alfa = [0 1];
    else
        alfa = [1 0];
    end

    if (alfa*alfa_prec'==1) comm = 0;
    else comm = 1;
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%       boucle sans le contrôleur bumpless transfer       %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    u(:,i)       = alfa(1)*K1*x_c(:,i) + alfa(2)*K2*x_c(:,i);

    X_succ       = alfa(1)*(A1_bt*[x(:,i); x_c(:,i)] + ...
                                [B1_w; zeros(n,r)]*w(:,i)) + ...
                   alfa(2)*(A2_bt*[x(:,i); x_c(:,i)] + ...
                           [B2_w; zeros(n,r)]*w(:,i));

    x(:,i+1)     = X_succ(1:n,1);
    x_c(:,i+1)   = X_succ(n+1:2*n,1);

    y(:,i)       = alfa(1)*(C1*x(:,i) + D1*w(:,i)) + ...
                   alfa(2)*(C2*x(:,i) + D2*w(:,i));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%        boucle avec le contrôleur bumpless transfer      %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if (comm == 1 && i ~= 1)
        x_tilde = x_c_bt(:,i);
        flag = 1;
        stop_comm = i;
    end

    if (flag==1 && i-stop_comm == des_contr)
        x_tilde = zeros(n,1);
        flag = 0;
    end

        W = [w(:,i); x_tilde];

        u_bt(:,i)       = alfa(1)*K1*x_c_bt(:,i) + alfa(2)*K2*x_c_bt(:,i);

        z(:,i)          = alfa(1)*(C1_bt(:,n+1:2*n)*x_c_bt(:,i) + D1_bt*W) + ...
                          alfa(2)*(C2_bt(:,n+1:2*n)*x_c_bt(:,i) + D2_bt*W);
```

```
        eta(:,i+1)      = alfa(1)*(A1_c*eta(:,i) + B1_c*z(:,i)) + ...
                          alfa(2)*(A2_c*eta(:,i) + B2_c*z(:,i));

        v(:,i)          = alfa(1)*(C1_c*eta(:,i) + D1_c*z(:,i)) + ...
                          alfa(2)*(C2_c*eta(:,i) + D2_c*z(:,i));

        X_succ_bt       = alfa(1)*(A1_bt*[x_bt(:,i); x_c_bt(:,i)] + ...
                                   B1_bt*v(:,i) + B1_w_bt*W) + ...
                          alfa(2)*(A2_bt*[x_bt(:,i); x_c_bt(:,i)] + ...
                                   B2_bt*v(:,i) + B2_w_bt*W);

        x_bt(:,i+1)     = X_succ_bt(1:n,1);
        x_c_bt(:,i+1)   = X_succ_bt(n+1:2*n,1);

        y_bt(:,i)       = alfa(1)*(C1*x_bt(:,i) + D1*w(:,i)) + ...
                          alfa(2)*(C2*x_bt(:,i) + D2*w(:,i));

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%                          LMI                           %%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%contrôleur nominal

function [K1 K2] = lmi_sol_K(A1, A2, B1, B2, n, r);

    setlmis([]);

    varS1 = lmivar(1,[n,1]);
    varS2 = lmivar(1,[n,1]);
    varR1 = lmivar(2,[r,n]);
    varR2 = lmivar(2,[r,n]);
    varG1 = lmivar(2,[n,n]);
    varG2 = lmivar(2,[n,n]);

    ineg1 = newlmi;
    ineg2 = newlmi;
    ineg3 = newlmi;
    ineg4 = newlmi;

    lmiterm([ineg1 1 1 varG1],-1,1,'s');
    lmiterm([ineg1 1 1 varS1],1,1);
    lmiterm([ineg1 2 1 varG1],-A1,1);
    lmiterm([ineg1 2 1 varR1],-B1,1);
    lmiterm([ineg1 2 2 varS1],-1,1);

    lmiterm([ineg2 1 1 varG1],-1,1,'s');
    lmiterm([ineg2 1 1 varS1],1,1);
    lmiterm([ineg2 2 1 varG1],-A1,1);
    lmiterm([ineg2 2 1 varR1],-B1,1);
    lmiterm([ineg2 2 2 varS2],-1,1);

    lmiterm([ineg3 1 1 varG2],-1,1,'s');
    lmiterm([ineg3 1 1 varS2],1,1);
    lmiterm([ineg3 2 1 varG2],-A2,1);
    lmiterm([ineg3 2 1 varR2],-B2,1);
    lmiterm([ineg3 2 2 varS1],-1,1);

    lmiterm([ineg4 1 1 varG2],-1,1,'s');
    lmiterm([ineg4 1 1 varS2],1,1);
    lmiterm([ineg4 2 1 varG2],-A2,1);
    lmiterm([ineg4 2 1 varR2],-B2,1);
    lmiterm([ineg4 2 2 varS2],-1,1);
    ProblemLMI = getlmis;

    [tmin_K xsol] = feasp(ProblemLMI);
    tmin_K

    R1 = dec2mat(ProblemLMI,xsol,varR1);
    R2 = dec2mat(ProblemLMI,xsol,varR2);
```

```
    G1 = dec2mat(ProblemLMI,xsol,varG1);
    G2 = dec2mat(ProblemLMI,xsol,varG2);

    K1 = R1*inv(G1)
    K2 = R2*inv(G2)
end



function [L1 L2] = lmi_sol_L(A1, A2, C1, C2, n, m);

    setlmis([]);

    varS1 = lmivar(1,[n,1]);
    varS2 = lmivar(1,[n,1]);
    varF1 = lmivar(2,[m,n]);
    varF2 = lmivar(2,[m,n]);
    varG1 = lmivar(2,[n,n]);
    varG2 = lmivar(2,[n,n]);

    ineg1 = newlmi;
    ineg2 = newlmi;
    ineg3 = newlmi;
    ineg4 = newlmi;

    lmiterm([ineg1 1 1 varG1],-1,1,'s');
    lmiterm([ineg1 1 1 varS1],1,1);
    lmiterm([ineg1 2 1 varG1],-A1',1);
    lmiterm([ineg1 2 1 varF1],C1',1);
    lmiterm([ineg1 2 2 varS1],-1,1);

    lmiterm([ineg2 1 1 varG1],-1,1,'s');
    lmiterm([ineg2 1 1 varS1],1,1);
    lmiterm([ineg2 2 1 varG1],-A1',1);
    lmiterm([ineg2 2 1 varF1],C1',1);
    lmiterm([ineg2 2 2 varS2],-1,1);

    lmiterm([ineg3 1 1 varG2],-1,1,'s');
    lmiterm([ineg3 1 1 varS2],1,1);
    lmiterm([ineg3 2 1 varG2],-A2',1);
    lmiterm([ineg3 2 1 varF2],C2',1);
    lmiterm([ineg3 2 2 varS1],-1,1);

    lmiterm([ineg4 1 1 varG2],-1,1,'s');
    lmiterm([ineg4 1 1 varS2],1,1);
    lmiterm([ineg4 2 1 varG2],-A2',1);
    lmiterm([ineg4 2 1 varF2],C2',1);
    lmiterm([ineg4 2 2 varS2],-1,1);

    ProblemLMI = getlmis;

    [tmin_L xsol] = feasp(ProblemLMI);
    tmin_L

    F1 = dec2mat(ProblemLMI,xsol,varF1);
    F2 = dec2mat(ProblemLMI,xsol,varF2);
    G1 = dec2mat(ProblemLMI,xsol,varG1);
    G2 = dec2mat(ProblemLMI,xsol,varG2);

    L1 = inv(G1')*F1'
    L2 = inv(G2')*F2'

end
```

```
%contrôleur bumpless transfer dynamique avec minimisation de gamma

function [A1_c A2_c B1_c B2_c C1_c C2_c D1_c D2_c] = lmi_sol_K_bt(A1, A2, B1, B2, B1_w, B2_w, ...
                        C1, C2, C1_z, C2_z, D1_yw, D2_yw, D1_z, D2_z, D1_zw, D2_zw, n, r, m);

setlmis([]);

varGAMMA = lmivar(1,[1,0]);
varS1 = lmivar(1,[n,1]);
varS2 = lmivar(1,[n,1]);
varN1 = lmivar(1,[n,1]);
varN2 = lmivar(1,[n,1]);
varM1 = lmivar(2,[n,n]);
varM2 = lmivar(2,[n,n]);
varQ1 = lmivar(2,[n,n]);
varQ2 = lmivar(2,[n,n]);
varX  = lmivar(2,[n,n]);
varY  = lmivar(2,[n,n]);
varW  = lmivar(2,[n,n]);
varL1 = lmivar(2,[r,n]);
varL2 = lmivar(2,[r,n]);
varF1 = lmivar(2,[n,m]);
varF2 = lmivar(2,[n,m]);
varR1 = lmivar(2,[r,m]);
varR2 = lmivar(2,[r,m]);

ineg1 = newlmi;
ineg2 = newlmi;
ineg3 = newlmi;
ineg4 = newlmi;

lmiterm([ineg1 1 1 varX],-1,1,'s');
lmiterm([ineg1 1 1 varS1],1,1);
lmiterm([ineg1 2 1 0],-eye(n));
lmiterm([ineg1 2 1 varW],-1,1);
lmiterm([ineg1 2 1 varM1],1,1);
lmiterm([ineg1 2 2 varY],-1,1,'s');
lmiterm([ineg1 2 2 varN1],1,1);
lmiterm([ineg1 3 1 0],-zeros(m,n));
lmiterm([ineg1 3 2 0],-zeros(m,m));
lmiterm([ineg1 3 3 varGAMMA],-1,1);
lmiterm([ineg1 4 1 varX],-A1,1);
lmiterm([ineg1 4 1 varL1],-B1,1);
lmiterm([ineg1 4 2 0],-A1);
lmiterm([ineg1 4 2 varR1],-B1,C1);
lmiterm([ineg1 4 3 0],-B1_w);
lmiterm([ineg1 4 3 varR1],-B1,D1_yw);
lmiterm([ineg1 4 4 varS1],-1,1);
lmiterm([ineg1 5 1 varQ1],-1,1);
lmiterm([ineg1 5 2 varY],-1,A1);
lmiterm([ineg1 5 2 varF1],-1,C1);
lmiterm([ineg1 5 3 varF1],-1,D1_yw);
lmiterm([ineg1 5 3 varY],-1,B1_w);
lmiterm([ineg1 5 4 varM1],-1,1);
lmiterm([ineg1 5 5 varN1],-1,1);
lmiterm([ineg1 6 1 varX],-C1_z,1);
lmiterm([ineg1 6 1 varL1],-D1_z,1);
lmiterm([ineg1 6 2 0],-C1_z);
lmiterm([ineg1 6 2 varR1],-D1_z,C1);
lmiterm([ineg1 6 3 0],-D1_zw);
lmiterm([ineg1 6 3 varR1],-D1_z,D1_yw);
lmiterm([ineg1 6 4 0],-zeros(m,n));
lmiterm([ineg1 6 5 0],-zeros(m,n));
lmiterm([ineg1 6 6 varGAMMA],-1,1);

lmiterm([ineg2 1 1 varX],-1,1,'s');
lmiterm([ineg2 1 1 varS1],1,1);
lmiterm([ineg2 2 1 0],-eye(n));
lmiterm([ineg2 2 1 varW],-1,1);
lmiterm([ineg2 2 1 varM1],1,1);
lmiterm([ineg2 2 2 varY],-1,1,'s');
lmiterm([ineg2 2 2 varN1],1,1);
lmiterm([ineg2 3 1 0],-zeros(r,n));
```

```
lmiterm([ineg2 3 2 0],-zeros(r,n));
lmiterm([ineg2 3 3 varGAMMA],-1,1);
lmiterm([ineg2 4 1 varX],-A1,1);
lmiterm([ineg2 4 1 varL1],-B1,1);
lmiterm([ineg2 4 2 0],-A1);
lmiterm([ineg2 4 2 varR1],-B1,C1);
lmiterm([ineg2 4 3 0],-B1_w);
lmiterm([ineg2 4 3 varR1],-B1,D1_yw);
lmiterm([ineg2 4 4 varS2],-1,1);
lmiterm([ineg2 5 1 varQ1],-1,1);
lmiterm([ineg2 5 2 varY],-1,A1);
lmiterm([ineg2 5 2 varF1],-1,C1);
lmiterm([ineg2 5 3 varF1],-1,D1_yw);
lmiterm([ineg2 5 3 varY],-1,B1_w);
lmiterm([ineg2 5 4 varM2],-1,1);
lmiterm([ineg2 5 5 varN2],-1,1);
lmiterm([ineg2 6 1 varX],-C1_z,1);
lmiterm([ineg2 6 1 varL1],-D1_z,1);
lmiterm([ineg2 6 2 0],-C1_z);
lmiterm([ineg2 6 2 varR1],-D1_z,C1);
lmiterm([ineg2 6 3 0],-D1_zw);
lmiterm([ineg2 6 3 varR1],-D1_z,D1_yw);
lmiterm([ineg2 6 4 0],-zeros(m,n));
lmiterm([ineg2 6 5 0],-zeros(m,n));
lmiterm([ineg2 6 6 varGAMMA],-1,1);

lmiterm([ineg3 1 1 varX],-1,1,'s');
lmiterm([ineg3 1 1 varS2],1,1);
lmiterm([ineg3 2 1 0],-eye(n));
lmiterm([ineg3 2 1 varW],-1,1);
lmiterm([ineg3 2 1 varM2],1,1);
lmiterm([ineg3 2 2 varY],-1,1,'s');
lmiterm([ineg3 2 2 varN2],1,1);
lmiterm([ineg3 3 1 0],-zeros(r,n));
lmiterm([ineg3 3 2 0],-zeros(r,n));
lmiterm([ineg3 3 3 varGAMMA],-1,1);
lmiterm([ineg3 4 1 varX],-A2,1);
lmiterm([ineg3 4 1 varL2],-B2,1);
lmiterm([ineg3 4 2 0],-A2);
lmiterm([ineg3 4 2 varR2],-B2,C2);
lmiterm([ineg3 4 3 0],-B2_w);
lmiterm([ineg3 4 3 varR2],-B2,D2_yw);
lmiterm([ineg3 4 4 varS1],-1,1);
lmiterm([ineg3 5 1 varQ2],-1,1);
lmiterm([ineg3 5 2 varY],-1,A2);
lmiterm([ineg3 5 2 varF2],-1,C2);
lmiterm([ineg3 5 3 varF2],-1,D2_yw);
lmiterm([ineg3 5 3 varY],-1,B2_w);
lmiterm([ineg3 5 4 varM1],-1,1);
lmiterm([ineg3 5 5 varN1],-1,1);
lmiterm([ineg3 6 1 varX],-C2_z,1);
lmiterm([ineg3 6 1 varL2],-D2_z,1);
lmiterm([ineg3 6 2 0],-C2_z);
lmiterm([ineg3 6 2 varR2],-D2_z,C2);
lmiterm([ineg3 6 3 0],-D2_zw);
lmiterm([ineg3 6 3 varR2],-D2_z,D2_yw);
lmiterm([ineg3 6 4 0],-zeros(m,n));
lmiterm([ineg3 6 5 0],-zeros(m,n));
lmiterm([ineg3 6 6 varGAMMA],-1,1);

lmiterm([ineg4 1 1 varX],-1,1,'s');
lmiterm([ineg4 1 1 varS2],1,1);
lmiterm([ineg4 2 1 0],-eye(n));
lmiterm([ineg4 2 1 varW],-1,1);
lmiterm([ineg4 2 1 varM2],1,1);
lmiterm([ineg4 2 2 varY],-1,1,'s');
lmiterm([ineg4 2 2 varN2],1,1);
lmiterm([ineg4 3 1 0],-zeros(r,n));
lmiterm([ineg4 3 2 0],-zeros(r,n));
lmiterm([ineg4 3 3 varGAMMA],-1,1);
lmiterm([ineg4 4 1 varX],-A2,1);
lmiterm([ineg4 4 1 varL2],-B2,1);
lmiterm([ineg4 4 2 0],-A2);
```

```
lmiterm([ineg4 4 2 varR2],-B2,C2);
lmiterm([ineg4 4 3 0],-B2_w);
lmiterm([ineg4 4 3 varR2],-B2,D2_yw);
lmiterm([ineg4 4 4 varS2],-1,1);
lmiterm([ineg4 5 1 varQ2],-1,1);
lmiterm([ineg4 5 2 varY],-1,A2);
lmiterm([ineg4 5 2 varF2],-1,C2);
lmiterm([ineg4 5 3 varF2],-1,D2_yw);
lmiterm([ineg4 5 3 varY],-1,B2_w);
lmiterm([ineg4 5 4 varM2],-1,1);
lmiterm([ineg4 5 5 varN2],-1,1);
lmiterm([ineg4 6 1 varX],-C2_z,1);
lmiterm([ineg4 6 1 varL2],-D2_z,1);
lmiterm([ineg4 6 2 0],-C2_z);
lmiterm([ineg4 6 2 varR2],-D2_z,C2);
lmiterm([ineg4 6 3 0],-D2_zw);
lmiterm([ineg4 6 3 varR2],-D2_z,D2_yw);
lmiterm([ineg4 6 4 0],-zeros(m,n));
lmiterm([ineg4 6 5 0],-zeros(m,n));
lmiterm([ineg4 6 6 varGAMMA],-1,1);

ProblemLMI = getlmis;

sum = count(n);
c = [1 zeros(1,4*sum+7*n^2+2*m*n+2*n*r+2*r*m)];

[gamma xsol] = mincx(ProblemLMI,c);
gamma

X  = dec2mat(ProblemLMI,xsol,varX);
Y  = dec2mat(ProblemLMI,xsol,varY);
W  = dec2mat(ProblemLMI,xsol,varW);
L1 = dec2mat(ProblemLMI,xsol,varL1);
L2 = dec2mat(ProblemLMI,xsol,varL2);
F1 = dec2mat(ProblemLMI,xsol,varF1);
F2 = dec2mat(ProblemLMI,xsol,varF2);
Q1 = dec2mat(ProblemLMI,xsol,varQ1);
Q2 = dec2mat(ProblemLMI,xsol,varQ2);
R1 = dec2mat(ProblemLMI,xsol,varR1);
R2 = dec2mat(ProblemLMI,xsol,varR2);

V = eye(n);
U = W-Y*X;

B1_c = inv(V)*(F1-Y*B1*R1);
B2_c = inv(V)*(F2-Y*B2*R2);
C1_c = (L1-R1*C1*X)*inv(U);
C2_c = (L2-R2*C2*X)*inv(U);
D1_c = R1;
D2_c = R2;
A1_c = inv(V)*(Q1-Y*(A1+B1*R1*C1)*X-V*B1_c*C1*X-Y*B1*C1_c*U)*inv(U);
A2_c = inv(V)*(Q2-Y*(A2+B2*R2*C2)*X-V*B2_c*C2*X-Y*B2*C2_c*U)*inv(U);
end

end


function [sum] = count(n)

    sum = 0;
    for i = 1:n-1
        sum = sum + n-i;
    end
    sum = sum + n;
end
```

Bibliographie

[1] K.J. Åström, T. Hägglund, Automatic tuning of PID controllers, *Instrument Society of America*, 1988.

[2] S.P.Boyd, L. El Ghaoui, E. Feron, V. Balakrishnan, Linear matrix Inequalities in systems and control theory, *LMIbook*: 19-20, 1994.

[3] J.Daafouz, J.Bernussou, Robust dynamic output feedback control for switched systems, *RAPPORT LAAS N°02110 41st IEEE Conference on Decision and Control*: 4389-4394, 10-13 December 2002.

[4] J.Daafouz, P.Riedinger, C.Iung, Observed-based switched control design for discrete-time switched systems, *European Control Conference (ECC2003)*: 1-4, september 2003.

[5] K. Dong, F. Wu, Online switching control designs of LFT systems, *45th IEEE Conference on Decision and Control*: 1-8, March 1, 2006.

[6] J.C. Doyle, R.S. Smith, D.F. Enns, Control of plants with input saturation non linearities, *Proceedings of the 1987 American Control Conference*: 1034-1039, 1987.

[7] H.A. Fertik, C.W. Ross, Direct digital control algorithm with anti-windup feature, *ISA Transactions, 6*: 317-328, 1967.

[8] R. Hanus, M. Kinnaert, Control of constrained multivariable systems using the conditioning technique, *Proceedings of the 1989 American Control Conference*: 1711-1718, 1989.

[9] R. Hanus, M. Kinnaert, J.-L. Henrotte, Conditioning technique, a general anti-windup and bumpless transfer method, *Automatica,* 23: 729-739, 1987.

[10] M. Kothare, P. Campo, M. Morari, A unified framework for study of anti-windup designs, *CDS Technical Report No. CIT-CDS 93-011*: 1-16, June 16, 1993.

.