Conception conjointe optimisée de lois de contrôle et d'ordonnancement

Ning Jia

Département de formation doctorale en informatique

Institut National
Polytechnique de Lorraine

École doctorale IAEM Lorraine

Conception conjointe optimisée de lois de contrôle et d'ordonnancement

THÈSE

présentée et soutenue publiquement le 15 Janvier 2009

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine

(spécialité informatique)

par

Ning JIA

Composition du jury

Rapporteurs : Sylviane GENTIL
Daniel SIMON

Examinateurs : François CHARPILLET
Pascal RICHARD
Françoise SIMONOT-LION
Ye-Qiong SONG

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Remerciements

Je remercie tout d'abord Madame Françoise Simonot-Lion, responsable de l'équipe Temps Réel et Interopérabilité (TRIO) du Loria, pour son encouragement permanent et son sontien depuis mon stage de DEA. Qu'elle soit ici assurée de mon grand respect et du plaisir que j'ai à travailler avec elle.

J'exprime toute ma reconnaissance à Monsieur Ye-Qiong Song pour m'avoir accordé sa confiance depuis le DEA et tout au long de ce travail, et pour sa disponibilité, son écoute et ses conseils qui m'ont toujours été profitables.

Un grand merci à Madmoiselle Flavia Felicioni pour le travail que nous avons fait en commun et d'avoir porté un intérêt pertinent à mes travaux.

Je tiens à exprimer mes vifs remerciements à :

- Monsieur Daniel Simon, Chargé de recherche à l'INRIA Rhone-Alpes ;
- Madame Sylviane Gentil, Professeur à l'INPG ;
- Monsieur François Charpillet, Directeur de recherche à l'INRIA Nancy Grand Est ;
- Monsieur Pascal Richard, Professeur à l'université de Poitiers.

pour m'avoir fait l'honneur de participer à mon jury de Thèse.

Je remercie particulièrement Monsieur François Simonot pour son appui patient et ses corrections avisées.

Mes remerciements vont également vers tous les membres et les anciens membres de l'équipe TRIO pour leur convivialité durant ces dernières années.

A mes parents et luan,
pour leur compréhension et soutiens.

Table des matières

D
Calcul de la dérivée de fonction de coût

Introduction générale

Aujourd'hui, le contrôle-commande temps réel est omniprésent dans beaucoup de secteurs tels que conduite de procédés, avionique, automobile, télécommunications, etc., et il prend même de plus en plus de place dans notre vie quotidienne avec des systèmes enfouis dans nos appareils électroménagers. En termes de complexité, les applications de contrôle commande temps réel couvrent un large spectre allant du simple microcontrôleur (contrôle du système de freinage d'une voiture, par exemple), jusqu'aux systèmes répartis (contrôle du trafic aérien, par exemple). Les enjeux économiques et les intérêts scientifiques liés aux systèmes temps réel sont multiples. C'est la raison pour laquelle on assiste, depuis les années soixante-dix, à une profusion de langages, de méthodes, d'algorithmes, de protocoles de communication, etc., pour le temps réel.

Au fur et à mesure que la capacité du matériel informatique augmente et son coût diminue, de plus en plus de fonctionnalités sont réalisées au moyen d'informatique, et il y a une tendance à partager une ou plusieurs ressources de calcul et/ou de communication dans une application de contrôle commande temps réel. Ainsi, on peut trouver par exemple que dans un système embarqué (cf, la figure 1), la tâche temps réel implémentant le contrôleur (appelée tâche de contrôle) s'exécute parallèlement sur un processeur avec plusieurs autres tâches, y compris d'autres tâches de contrôle. Pour un autre exemple, on pourra se reporter à l'application de contrôle commande distribué (cf, la figure 2) qui possède une ou plusieurs boucles de contrôle fermées via un réseau de communication. Le réseau est partagé non seulement pour l'échange de données entre les contrôleurs et les procédés contrôlés mais aussi pour transmettre des messages d'autres types d'applications. Puisque la qualité de contrôle (QdC) de la boucle de

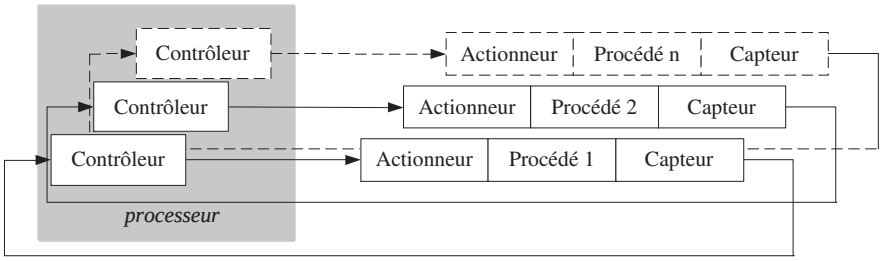

FIGURE 1 – Le partage d'un processeur par un certain nombre de boucles de contrôle

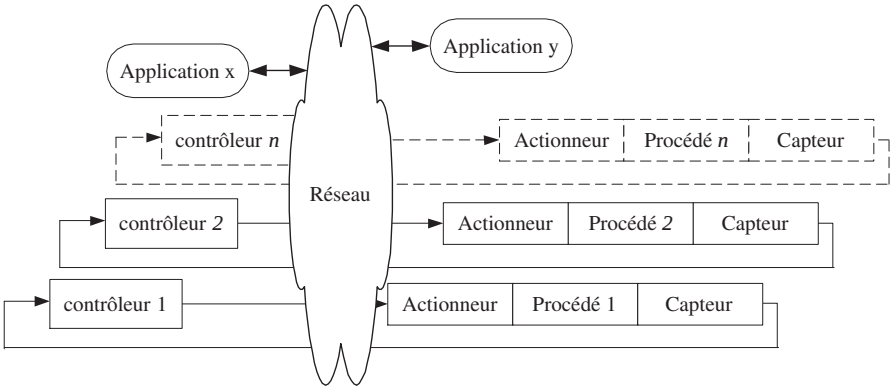

FIGURE 2 – Le partage d'un réseau par un certain nombre de boucles de contrôle et d'autres applications

contrôle dépend largement des caractéristiques temporelles de son exécution, ainsi se pose le problème de choix des paramètres d'ordonnancement des tâches ou des messages.

Avec les matériels informatiques toujours plus performants, on pourrait être convaincu que la plupart des problèmes temporels peuvent être résolus en mettant à jour le processeur ou les composants du réseau. Bien que ceci puisse être vrai dans certains cas, la conception du système est souvent soumise à une forte contrainte économique (par exemple le cas des systèmes électroniques embarqués dans l'automobile). Le concepteur est obligé de partager un processeur ou un noeud de communication du réseau déjà fortement chargé avec plus de tâches. Seulement dans des applications extrêmes, telles que les centrales nucléaires, le coût du matériel informatique peut être négligé pendant le développement de l'application.

1 Conception conjointe de commande et d'ordonnancement

Ce travail vise à réaliser la meilleure performance de contrôle sur des ressources de calcul ou de communication limitées. Notons que c'est aussi équivalent à dire qu'on vise à maintenir la meilleure performance possible (ou avec dégradation contrôlée) en cas de surcharge des ressources (i.e. situation de ressources limitées pendant une période transitoire). Pour atteindre cet objectif, un nouveau modèle de conception de commande et ordonnancement sera nécessaire. Pendant longtemps, la conception d'une application de commande s'est fait en deux étapes relativement isolées : la conception de commande et la conception d'ordonnancement. Les automaticiens conçoivent et évaluent les algorithmes de contrôle en supposant un modèle simple de la plate-forme informatique support (processeurs et réseaux). D'autre part, les informaticiens ordonnancent les tâches de contrôle avec d'autres types de tâches sans véritable connaissance de l'influence des paramètres d'ordonnancement sur la performance de contrôle. Par exemple, d'où viennent les échéances strictes sur des tâches qu'on cherche à ordonnancer dans la communauté de l'ordonnancement temps réel ? Quelle est la consé-

quence sur l'application si une échéance du temps réel dur n'est pas respectée ? L'application concernée a-t-elle une certaine capacité de tolérance au dépassement d'échéance voire non exécution d'une instance de tâche ? Dans une perspective historique, le développement séparé des théories de commande et d'ordonnancement a conduit à un nombre important de résultats utiles. Pourtant, ce modèle de développement relativement isolé présente du conservatisme et mène ainsi aux solutions non-optimales (e.g. surdimensionnement de ressources). Ainsi, une meilleure façon de concevoir l'application de contrôle commande temps réel est de mener la conception de commande et d'ordonnancement conjointement au lieu de les traiter comme deux aspects isolés : d'une part, les contrôleurs nécessitent d'être conçus en tenant compte des caractéristiques temporelles d'ordonnancement impliquées par le partage de ressources, et d'autre part, les tâches ou les messages doivent être ordonnancés sans ignorer les conséquences sur la performance de contrôle. Et enfin, la disponibilité des ressources (qui dépent fortement de la charge du système) et le besoin en ressources de boucles de contrôle (qui dépend de l'état du procédé) varient dans le temps. La conception conjointe devrait permettre la mise en oeuvre de mécanismes en-ligne conduisant ainsi à un système global adaptatif : quand le besoin en ressources d'une application diminue (e.g. procédé stable), la contrainte sur l'échéance peut être relâchée afin de favoriser le traitement d'autres tâches ; quand le besoin en ressource d'une application augmente (ou la ressource devient limitée à cause d'une surchage), l'ordonnancement initial peut être modifié afin de favoriser le traitement de cette application tout en maintenant une performance globale de l'ensemble des applications à un niveau satisfaisant.

Un système informatique possède beaucoup de ressources importantes (processeur, mémoire, entrée-sortie, réseau, etc.) dont le partage exerce un impact sur le comportement temporel des applications. Cette thèse se focalise sur l'ordonnancement des tâches de contrôle sur le processeur ou des messages nécessaires au calcul de la loi de commande sur le noeud de communication lorsqu'il s'agit de l'application distribuée. Ainsi le problème de la conception conjointe de commande et d'ordonnancement peut être formellement défini comme suit.

Définition 1 *Etant donné un ensemble de procédés à contrôler et un processeur (ou un noeud de communication) où s'implémentent les tâches de contrôle (ou se transmettent les messages nécessaires au calcul de la loi de commande), la* conception conjointe *(ou* co-conception*) consiste à concevoir conjointement un ensemble de contrôleurs et la stratégie d'ordonnancement des tâches de contrôle (ou messages) de telle sorte que la QdC globale soit optimisée.*

En général, ce problème de conception conjointe peut paraître trivial si l'on ne considère qu'un système fermé où l'ensemble de procédés et de ressources informatiques est parfaitement connu. Dans un système ouvert (e.g. boucles de contrôle via un réseau partagé, appelé aussi systèmes contrôlés en réseau ou NCS : networked control systems) en revanche, la conception conjointe n'est pas triviale, ceci à cause de la variation dans le temps de la disponibilité de ressources qui elles sont aussi partagées par d'autres applications dont les caractéristiques restent inconnues ou partiellement connues. Là nous sommes en face d'un problème de co-conception commande-ordonnancement pour optimiser la QdC avec la présence de la variation

de la disponibilité de ressources (ou en équivalent la variation de performance du système informatique support vis à vis de ces applications de contrôle).

Dans le cadre de cette thèse, nous nous intéressons à développer des méthodes de co-conception des systèmes ouverts afin de fournir la QdC requise malgré la dégradation temporaire des performances du système informatique support. Sans perdre de la généralité, nous considérons que cette dégradation de performance est provoquée par la surcharge, et dans ce cas nous proposons d'explorer une approche qui consiste à rejeter des instances (ou messages) de manière contrôlée. Le modèle (m,k)-firm [Hamdaoui95] nous paraît convenable pour contrôler le rejet. Notons qu'il existe d'autres approches telles que celle qui allonge la période des boucles de contrôle en cas de surcharge [Eker00].

2 Solution envisagée : conception conjointe avec rejet sélectif selon le modèle (m,k)-firm

La conception conjointe de commande et d'ordonnancement a attiré, dans ces derniers temps, l'attention dans le domaine de l'automatique aussi bien que dans le domaine de l'ordonnancement temps réel, et un certain nombre de résultats ont été obtenus. L'allocation de la ressource de calcul aux tâches de contrôle est le problème le plus traité parmi les approches existantes. La technique de base utilisée par ces approches consiste à calculer la période des tâches de contrôle dans l'application afin de distribuer raisonnablement la ressource de calcul entre elles et ainsi d'obtenir une performance de contrôle globale optimale sous la contrainte d'ordonnançabilité. Par exemple, en supposant que la fonction de coût (relations entre la période d'échantillonnage et la performance de contrôle) associée à chaque tâche de contrôle est exponentielle, l'algorithme proposé dans [Seto96] calcule hors ligne les périodes d'un ensemble de tâches de contrôle de sorte que la performance de contrôle globale soit optimisée et la contrainte d'ordonnançabilité des tâches soit respectée. Citons également une approche représentative proposée dans [Eker00] qui, en supposant que la fonction de coût de chaque tâche dans l'application soit quadratique, permet de calculer en ligne les périodes des tâches garantissant l'optimisation de la performance de contrôle globale et le respect de la contrainte d'ordonnançabilité.

Cependant, ces approches basées sur la régulation des périodes des tâches de contrôle souffrent d'un certain nombre de limites lorsqu'il s'agit de l'optimisation en ligne :

- Si la ressource partagée est un noeud de communication dans le réseau, la technique de régulation des périodes n'est pas applicable en règle générale. La communication du noeud de réseau vers les capteurs des procédés pour ralentir le rythme d'émission des capteurs nécessite l'implémentation de mécanismes complexes dans le noeud de communication, augmentant ainsi fortement le coût et rendant la solution souvent inenvisageable dans la pratique. Alors que le rejet des échantillons en retard est une solution naturelle puisqu'un échantillon non envoyé à temps est souvent écrasé par un nouveau.

- Lorsque des tâches sont soumises à des contraintes de précédence, le changement de période d'une tâche implique souvent le changement de période d'une (des) autre(s) tâche(s), car les périodes des tâches sous la contrainte de précédence sont souvent minutieusement choisies pour un échange efficace d'informations entre les tâches,
- Les fonctions de coût des tâches de contrôle sont supposées convexes ou peuvent être approchées par une fonction convexe pour réduire la complexité de l'algorithme d'optimisation. Cependant, cette hypothèse n'est pas toujours valide, car en pratique, il est parfois difficile de garantir que ces fonctions sont toutes convexes. En observant les caractéristiques des fonctions de coût de certains systèmes, il est clair que tout les systèmes n'ont pas de fonction de coût convexe : certains systèmes ont la fonction de coût concave, ou même ni convexe ni concave.

En raison de ces limites des approches existantes basées sur la technique de régulation de périodes, les approches envisagées dans cette thèse s'appuient sur une solution différente qui consiste à maintenir l'optimalité de la performance de contrôle globale en sélectivement rejetant les instances des tâches de contrôle (ou messages nécessaires au calcul de la loi de commande) au lieu d'ajuster leurs périodes. En réalité, le rejet d'un échantillon est équivalent à multiplier la période d'échantillonnage. Du point de vue du contrôle, il s'agit d'étudier le contrôle à période d'échantillonnage variable. Ce qui différencie notre approche de celle de régulation de périodes est simplement dans la façon d'atteindre l'objectif de changement de périodes. Cette stratégie permet de surmonter les limites mentionnés ci-dessus car :

- Le mécanisme de rejet dynamique des messages dans un noeud de communication est plus facile à implémenter que le changement de période d'échantillonnage (e.g. buffer avec écrasement) ;
- En cas des tâches sous la contrainte de précédence, comme la période nominale de chaque tâche reste constante durant l'exécution de l'application, l'adaptation de périodes au moment du réajustement n'est plus nécessaire ;
- Les rejets d'instances ou de messages peuvent être considérés comme la multiplication de la période nominale de la tâche ou du message. Ainsi, trouver une stratégie optimale de rejets des instances ou messages consiste à calculer ces périodes multiples, ce qui peut être modélisé par un problème d'optimisation discrète pour lequel il existe des solutions qui ne dépendent pas de la convexité du problème et qui sont efficaces en terme de complexité.

Concrètement, la stratégie de rejet des instances ou messages se base sur le modèle (m,k)-firm qui a été proposé par Hamdaoui et Ramanathan [Hamdaoui94, Hamdaoui95]. La contrainte (m,k)-firm consiste à imposer le respect des échéances de m instances (messages) parmi k instances (messages) consécutives quelconques et permet de spécifier explicitement le motif (ou pattern) de rejet (ou non respect des échéances), ce qui le rend particulièrement intéressant pour des systèmes de contrôle commande qui tolèrent le non respect des échéances de certaines instances ou messages sans affecter leur fonctionnement correct (par exemple : le respect de la contrainte de temps de réponse, du dépassement, du coût, etc.). De plus, le système peut

être conçu selon le modèle (m,k)-firm pour offrir différents niveaux de QdC, entre (k,k)-firm (cas idéal) et (m,k)-firm (le pire cas) avec autant de niveaux intermédiaires que de valeurs possibles entre m et k, ce qui donne un système avec la dégradation contrôlée de QdC ("graceful degradation").

3 Les objectifs et contributions

Le *premier objectif* est de démontrer l'intérêt de cette approche de contrôle de dégradation de la QdC par rejet selon le modèle (m,k)-firm. Ceci se réalise par l'étude de l'impact de rejet d'instances ou messages sur la QdC d'une boucle de contrôle dans un premier temps, puis la proposition d'une méthode de co-conception de lois de contrôle et ordonnancement. Par la conception conjointe, les éléments à considérer dans la conception de la boucle de contrôle sont, non seulement le procédé physique à contrôler, mais également les paramètres d'ordonnancement caractérisés par la contrainte (m,k)-firm. Cette étape est constituée des sous-étapes suivantes :

- Identification de la contrainte (m,k)-firm satisfaisant l'exigence de performance de la boucle de contrôle.
- Association de la distribution de rejets d'instances (ou messages) dans la séquence d'instances (ou messages) à la QdC de la boucle, et identification de la distribution donnant la performance de contrôle optimale,
- Conception du contrôleur optimal sous la contrainte (m,k)-firm.

Cette étape est soutenue par des activités de modélisation et analyse, simulation et expérimentation. Un aspect particulièrement important est l'analyse et la simulation de la boucle de contrôle en fonction des différents contraintes (m,k)-firm.

Dans cette étude, nous considérons les applications évolutives (c'est-à-dire un système ouvert) donc la configuration pourra être changée (activation ou désactivation de tâches, changement de temps d'exécution d'une tâche dû au changement de l'algorithme de contrôle) durant son exécution. Garantir le respect des contraintes temps réel avec des ressources limitées et supporter les variations de la configuration de l'application exigent une offre de services adaptatifs. La solution conservatrice consiste à définir des contraintes temporelles strictes et à mettre en oeuvre la preuve de l'ordonnançabilité (de tâches ou de messages) dans le pire cas. Cette méthode conduit généralement à surdimensionner les ressources nécessaires.

Ainsi notre *deuxième objectif* est de concevoir des mécanismes exécutifs en ligne qui ajustent les contraintes (m,k)-firm des tâches (ou messages) suivant la configuration courante de l'application de manière à ce que la performance globale de l'application soit optimale dans la configuration courante. Ces deux objectifs sont étroitement liés : l'étape de l'étude de l'impact de rejets d'instances ou message permet de fournir tous les éléments nécessaires à la mise en oeuvre des mécanismes exécutifs en identifiant formellement la relation entre la contrainte (m,k)-firm et la QdC.

Par rapport à l'état de l'art existant sur la co-conception, dans cette thèse nous avons proposé les contributions principales suivantes :

- établissement d'une relation explicite entre la QdC et le rejet d'échantillons ;
- développement d'une méthode de co-conception permettant de déterminer les paramètres (gain) optimaux de la loi de contrôle en présence de rejets ;
- proposition d'un mécanisme en-ligne réajustant le gain des contrôleurs en fonction de rejets, tout en respectant l'ordonnançabilité de l'ensemble de tâches sous contrainte (m,k)-firm.

4 Organisation du document

Ce document se compose de six chapitres. Dans le premier chapitre, nous présentons le contexte du contrôle-commande et la problématique liée à la prise en compte des performances de l'implémentation lors de la conception des systèmes de contrôle-commande, puis un état de l'art sur les travaux réalisés autour de la conception conjointe commande-ordonnancement, et enfin notre approche de réglage de rejet selon le (m,k)-firm. Le chapitre 2 présente une synthèse sur les travaux autour du modèle (m,k)-firm. Dans le chapitre 3, nous montrons tout d'abord comment déterminer, pour un système monodimensionnel, la contrainte (m,k)-firm sous laquelle la boucle de contrôle reste stable. La démarche pour concevoir le contrôleur qui minimise la détérioration de performance de contrôle à cause des rejets d'échantillons est également présentée. Une contribution technique est donnée qui consiste à proposer une fonction de coût basée sur la borne supérieure de la variance au lieu de la variance elle-même. Notre approche est généralisée et enrichie dans les chapitres 4 et 5 pour traiter ces mêmes problèmes liés à un système multidimensionnel. Dans le chapitre 4, en prenant l'exemple concret du déplacement d'un chariot sur un rail, nous décrivons comment obtenir la valeur de k, c'est à dire la période d'échantillonnage maximale, puis étudions l'impact de la valeur de m ainsi que la distribution de rejets d'échantillons (i.e. les m instances) sur la performance de contrôle au moyen de simulation. Le chapitre 5 propose une méthode analytique de conception conjointe. Pour une situation donnée, cette méthode permet de calculer non seulement le gain optimal du contrôleur mais également la meilleure répartition de m sur k. Dans le chapitre 6, nous considérons l'application de contrôle-commande embarquée dans laquelle le processeur est partagé par un ensemble de tâches de contrôle. Nous proposons un algorithme qui au moment du changement de la configuration de l'application, recalcule une contrainte (m,k)-firm de chaque tâche afin que l'ordonnançabilité de cet ensemble de tâches soit garantie et la performance de contrôle globale soit maintenue à un niveau acceptable. Enfin, un dernier chapitre donne une conclusion générale et présente nos perspectives pour étendre les travaux présentés dans ce document.

Chapitre 1

Problématique et travaux relatifs

1.1 Contexte

Cette section est consacrée à l'introduction des problématiques liés à *l'implémentation* du système numérique de contrôle-commande. Nous présentons dans un premier temps l'architecture d'une boucle de contrôle ainsi que son modèle mathématique, puis les considérations de validation du système. Enfin, nous présentons les stratégies d'implémentation du système de contrôle-commande.

1.1.1 Système numérique de contrôle-commande

Tout au long de ce manuscrit, nous supposons que le processus à contrôler peut être décrit par un ensemble d'équations linéaires, qui donnent la relation entre le signal d'entrée (commande) et de sortie (mesures du processus contrôlé). Comme le processus est un processus physique existant dans le monde réel, ainsi il est un système à temps continu. Une représentation mathématique standard d'un tel système est

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t) + Gv(t) \\ y(t) &= Cx(t) \end{aligned} \tag{1.1}$$

où x est un vecteur décrivant l'état du processus, u est le signal de commande, y est le signal de sortie, et v est un bruit blanc Gaussien à moyenne nulle qui perturbe les états du processus. A, B, C, et G sont des matrices constantes qui décrivent la dynamique du processus.

Le contrôleur est un système à temps discret car la sortie du processus contrôlé est échantillonnée périodiquement. Les états du processus sont considérés comme mesurables, ou s'ils ne le sont pas, ils peuvent être reconstitués par l'introduction d'un observateur. Le contrôle linéaire est représenté par l'équation suivante :

$$u(kh) = Lx(kh) \tag{1.2}$$

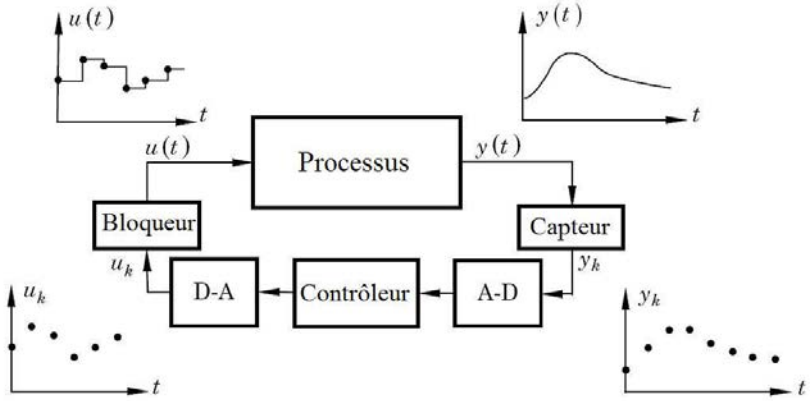

FIGURE 1.1 – Relation entre les signaux à temps continu et à temps discret dans une boucle de contrôle.

où x est l'état échantillonné, et u est maintenu constante par le bloqueur jusqu'à la prochaine mise à jour de la commande (Figure 1.1).

Pour plus d'informations sur le système numérique de contrôle-commande et ses caractéristiques, le lecteur peut se référer par exemple à [Astrom97].

1.1.2 Evaluation du système de contrôle-commande

Pendant la phase de conception des systèmes de contrôle-commande, la notion de *stabilité* revêt une importance particulière. Il existe actuellement plusieurs critères de stabilité. Mais grosso modo, la notion de stabilité s'apparente à l'idée que le système n'explose pas, ne sort pas de certaines bornes. Par exemple, considérons le système asservi dont l'objectif est de faire venir la sortie du processus contrôlé à sa référence (valeur finale). La figure 1.2 montre la trajectoire d'évolution de la sortie d'un système asservi stable (dont la sortie à tendence à venir à sa référence) et celle d'un système asservi instable (dont la sortie a tendance à s'en écarter). La première condition de bon fonctionnement d'un système est la stabilité. Autrement dit, commander correctement un système, c'est avant tout éviter qu'il ne devienne instable. Le problème de stabilité devient ainsi un sujet de préoccupation majeur du travail des automaticiens depuis le siècle dernier. Beaucoup de travaux ont été effectués dans le domaine de l'analyse de stabilité [Lyap07, Hahn67, Park81].

Une fois la stabilité du système assurée, on cherche ensuite à améliorer la *performance de contrôle*. La performance de contrôle d'une boucle de contrôle peut être mesurée de plusieurs manières selon le type du système de contrôle-commande envisagé. Pour un système asservi, afin de quantifier les spécifications imposées sur la sortie, et ainsi mesurer la performance du contrôle, on a recourt à différentes caractéristiques de la sortie du processus contrôlé. Celles-ci sont typiquement définies dans le domaine temporel sur la réponse indicielle (réponse à un échelon de consigne). La figure 1.3 illustre ces caractéristiques sur la réponse indicielle d'un

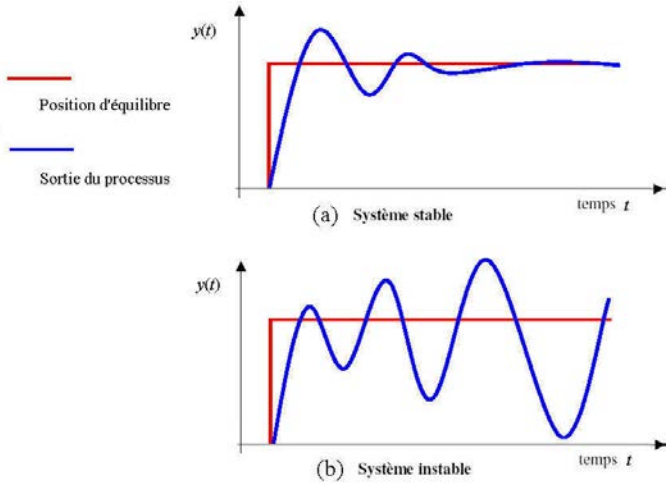

FIGURE 1.2 – (a) Système asservi stable ; (b) Système asservi instable

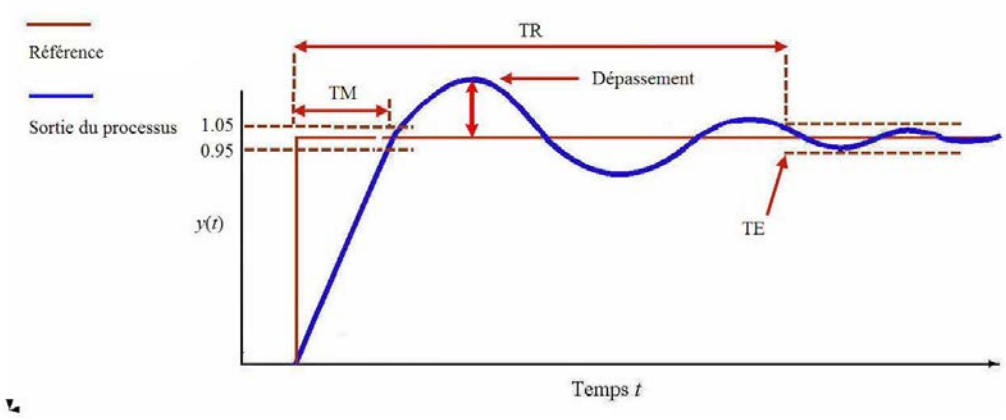

FIGURE 1.3 – Caractéristiques temporelles de la réponse indicielle d'un processus

processus.

- Dépassement (M) : exprime dans quelle mesure la sortie $y(t)$ va dépasser sa référence p avant de l'atteindre : $M = \max_{t \geq 0} \frac{y(t)-p}{p} 100\,[\%]$.
- Temps de montée (TM) : temps nécessaire à la sortie du processus pour atteindre un certain pourcentage de sa référence (nous considérerons ici 95%). Notons qu'on définit également dans certains ouvrages comme le temps pour que la sortie passe de 10% à 90% de la référence.
- Temps d'établissement (TE) : temps au-delà duquel l'écart entre $y(t)$ et la référence est borné par une certaine valeur, c'est-à-dire tel que $y(t)$ ne sorte plus d'un certain couloir centré sur la référence. Cet écart est exprimé en %.
- Temps de réponse (TR) : En fonction de la précision exigée, le temps de réponse est le temps au bout duquel la sortie du processus pénètre dans le couloir de plus ou moins 5% de la référence sans en sortir.

Pour les problèmes de régulation, l'objectif de contrôle est de maintenir la variable contrôlée à une valeur constante en présence des perturbations. Sans perdre de la généralité, cette valeur constante peut être supposée égale à 0 pour les systèmes linéaires (i.e. $E(x(t)) = 0$ où E représente l'espérance mathématique). Pour ce type de problème, l'utilisation de la notion de variance de la variable contrôlée est pertinente. Ce qui nous amène à nous intéresser à $E(|x(t)|^2)$. On peut par ailleurs exprimer une idée similaire en termes de fonction de coût quadratique, c-à-d :

$$J = \frac{1}{H} \int_0^H x^T(t) Q x(t) dt \tag{1.3}$$

où $x(t)$ est le vecteur de variables d'état comme précédemment défini et Q une matrice symétrique positive permettant de pondérer l'influence des variables avec des poids différents, H est l'horizon du temps pour mesurer le coût. Notons que lorsque $x(t)$ a un comportement périodique (cf. chapitre 3), la variance ne permet plus de nous donner des renseignements fins sur ce qui se passe à l'intérieur d'une période, nous proposons alors d'utiliser la valeur supérieure de la variance comme fonction de coût.

Le coût de contrôle $u(t)$ peut aussi être intégré dans la fonction de coût afin de prendre en compte la consommation d'énergie ou la limite de la sortie du contrôleur, la fonction 1.3 devient ainsi :

$$J = \frac{1}{H} \int_0^H x^T(t) Q x(t) + u^T(t) R u(t) dt \tag{1.4}$$

où Q et R sont des matrices de poids quadratiques qui représentent l'importance des éléments des vecteurs $x(t)$ et $u(t)$ dans le calcul de J, H est l'horizon de temps pour mesurer le coût. Une grande valeur de J indique de grandes déviations de l'état désiré ou un coût de contrôle important, est ainsi pire qu'une valeur plus petite de J. Un coût infini implique que le système est instable. Notons que cette fonction de coût peut être aussi utilisée pour mesurer la performance dans le cas d'un asservissement. Dans ce cas, l'écart entre l'état du processus et l'état désiré est utilisé à la place de l'état du processus dans la fonction 1.4.

1.2 Prise en compte de la plate-forme d'implémentation dans la spécification du système de contrôle-commande

1.2.1 Présentation du problème

Cette section présente principalement les problèmes liés à l'introduction de réseaux dans une boucles de contrôle (délai et perte). Il convient de souligner que le problème de délai peut aussi apparaître sans réseaux dès lors qu'un CPU est partagé entre plusieurs tâches dont celle de contrôle à cause du problème d'ordonnancement monoprocesseur (cf. figure 1). Mais ce dernier point est supposé connu et ne sera pas développé.

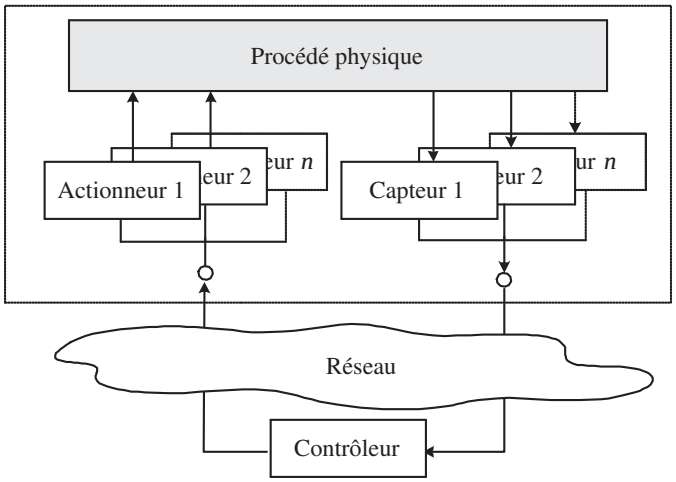

FIGURE 1.4 – Système de contrôle-commande distribué de structure directe

Durant ces dernières années, l'évolution technologique dans le domaine des réseaux informatiques a permis aux automaticiens d'inclure la liaison de communication dans des systèmes de contrôle-commande, par exemple pour éliminer le câblage en point à point, réduisant ainsi la complexité et le coût global dans les phases de conception et d'implémentation du système ; pour faciliter la mise au point ou la mise à jour du système en terme de l'insertion des actionneurs, des capteurs et des contrôleurs sans coût supplémentaire important ou modification de l'architecture du système ; ou tout simplement parce que la topologie des différents composants du système l'impose. Ceci donne naissance à un nouveau paradigme dans l'analyse et la conception des systèmes de contrôle-commande, à savoir systèmes de contrôle-commande distribués appelés aussi ces derniers temps systèmes contrôlés en réseau (Networked Control System).

L'étude du système de contrôle-commande distribué commence dans les années soixante-dix quand Honeywell a implémenté son système de contrôle distribué (DCS : distributed control system). Pourtant, les équipements d'un DCS sont rarement connectés car la plupart des fonctions de contrôle telles que l'exécution de la loi de commande, l'échantillonnage sont supportés par des modules dédiés, seuls les signaux marche/arrêt, les informations de surveillance, d'alarme sont transmis sur une liaison série. Aujourd'hui, grâce aux puces ASIC et à la baisse du coût du silicium, les capteurs et actionneurs peuvent tous être équipés d'une interface réseau, et peuvent ainsi devenir des noeuds indépendants sur un réseau de communication.

En général, les structures des systèmes de contrôle-commande distribués qui s'appliquent actuellement dans différents domaines peuvent être classées en deux types : la structure directe et la structure hiérarchisée. Dans les systèmes de contrôle-commande distribués avec la structure directe, la boucle de rétroaction est fermée sur un réseau de communication [Halevi88, Walsh99, Branicky00, Nilsson98]. La figure 1.4 illustre l'architecture de tels systèmes. Les mesures des sorties du système sont encapsulées dans un message, ce dernier est

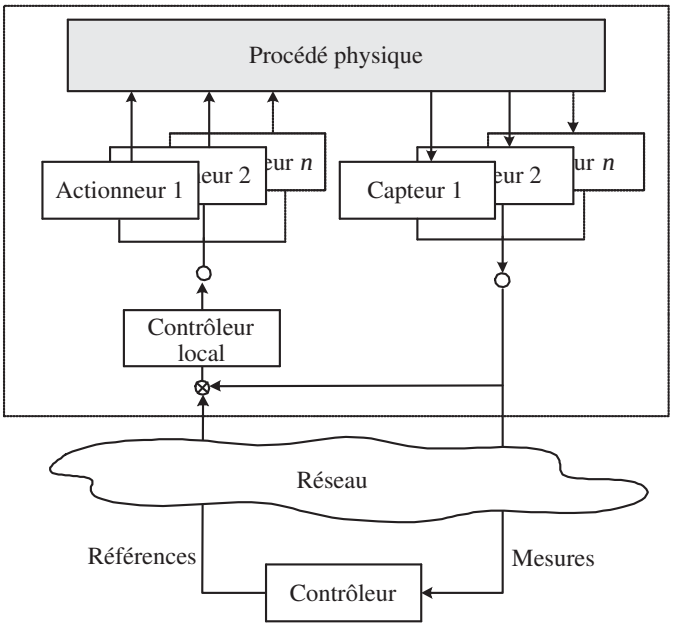

FIGURE 1.5 – Système de contrôle-commande distribué de structure hierarchisée

envoyé au contrôleur au travers du réseau. Le contrôleur prend régulièrement connaissance de l'état du processus physique par le biais des messages provenant de capteurs, calcule la prochaine action à effectuer sur la base des mesures reçues. Les consignes sont envoyées dans un message aux actionneurs et répercutées sur le processus contrôlé par le biais des actionneurs. Tandis que dans les systèmes de contrôle-commande distribué avec la structure hiérarchisée, il y a un contrôleur global dans un site maître qui planifie des actions et transmet les références au site distant dans lequel se trouve la boucle de rétroaction locale (cf. la figure 1.5). Ici, le réseau n'intervient pas dans la boucle de rétroaction. Cependant, si on considère les boucles de rétroaction locale implantées, par exemple dans le site structuré autour d'un réseau de terrain, la problématique de leur mise en oeuvre relève encore de la problématique de la structure directe. Dans la suite de ce paragraphe et dans les études menées durant cette thèse, nous nous focalisons sur l'analyse du système de contrôle-commande distribué de structure directe. Néanmoin, les méthodologies proposées pour la structure directe pourraient aussi être appliquées pour la structure hiérarchisée en considérant la boucle fermée locale comme un processus à contrôler. Dans ce cas, la boucle fermée locale peut être représentée par son modèle d'état (1.1) ou sa fonction de transfert (rapport entre les transformés de Laplace de la sortie et de l'entrée du système à conditions initiales nulles).

Notons que dans un système de contrôle-commande centralisé, le contrôleur est connecté directement aux échantillonneurs (capteurs) et actionneurs comme illustré dans la figure 1.6. Grâce à la connexion point à point directe des composants du système, le signal de sortie du processus et le signal de commande sont immédiatement disponibles pour le contrôleur et les actionneurs.

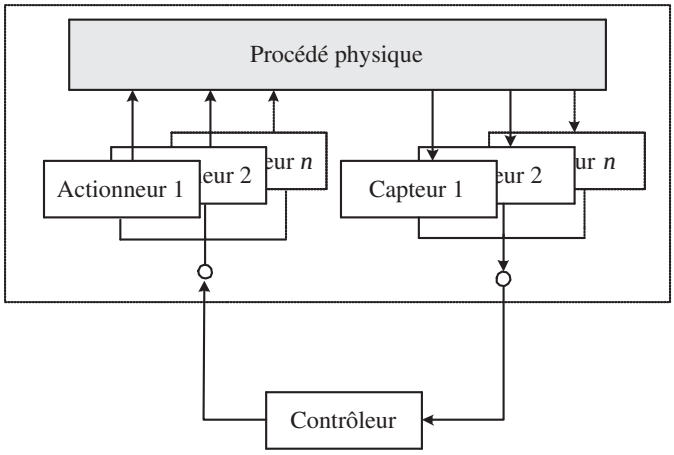

FIGURE 1.6 – Système de contrôle-commande centralisée

Les effets du retard et de la perte et leurs impacts

Considérons le système illustré par la figure 1.4. Les actionneurs et capteurs sont connectés au contrôleur au travers du réseau de communication. Puisque les messages contenant les informations de contrôle sont envoyés sur le réseau, deux effets inattendus pourraient se produire sur la performance du système . Premièrement, la communication entres les composants du système peut subir un délai de transmission (délais de transmission entre les capteurs et le contrôleur et entre le contrôleur et les actionneurs). Deuxièmement, le réseau introduit souvent une non-fiabilité des données transférées : les messages souffrent non seulement du délai de transmission mais aussi du risque de perte (une perte de messages peut être néanmoins considérée comme un délai infini). Dépendant du protocole de communication, de la charge du réseau, de la politique d'ordonnancement des messages, et même des perturbations électromagnétiques (electromagnetic interference - EMI) [Ray87], le délai de transmission peut avoir des caractéristiques différentes. Pour certaines configurations du réseau, le délai de transmission peut être constant, mais dans d'autres cas, il peut varier de manière aléatoire. La figure 1.7 illustre le système affecté par les délais de transmission avec :

- τ_{cc}, le délai de transmission entre les capteurs et le contrôleur ;
- τ_{ca}, le délai de transmission entre le contrôleur et les actionneurs.

Mis à part les délais de transmission, il existe aussi le délai de calcul de la tâche temps réel implémentant le contrôleur. En général, le délai de calcul est plus court que le délai de transmission sur le réseau et est souvent négligé dans l'analyse. En tout cas, lorsque le délai de calcul n'est pas négligeable, celui-ci peut être inclus dans le délai de transmission [Nilsson98]. La figure 1.8 montre le diagramme de la propagation des délais.

Que le délai soit constant ou variable dans le temps, celui-ci peut dégrader la performance du système conçu sans prendre en compte le délai, et même déstabiliser le système. Une étude

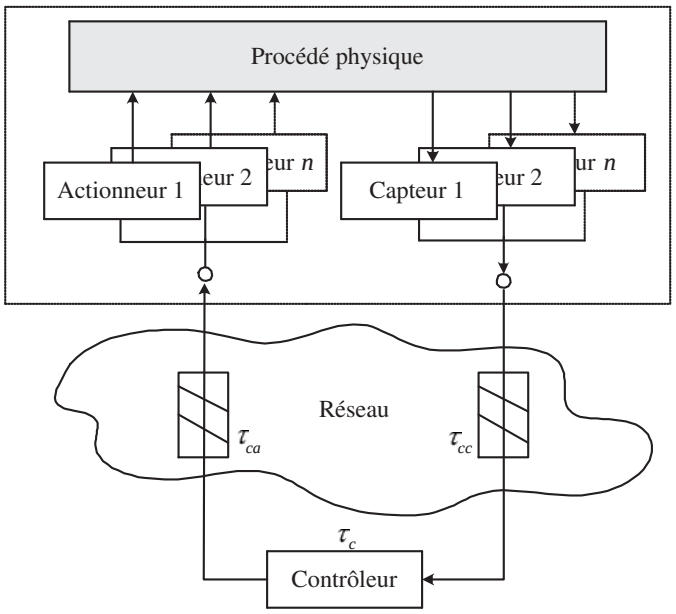

FIGURE 1.7 – Boucle de contrôle avec délais, τ_{cc} et τ_{ca} ; le temps de calcule du contrôleur τ_c est aussi indiqué

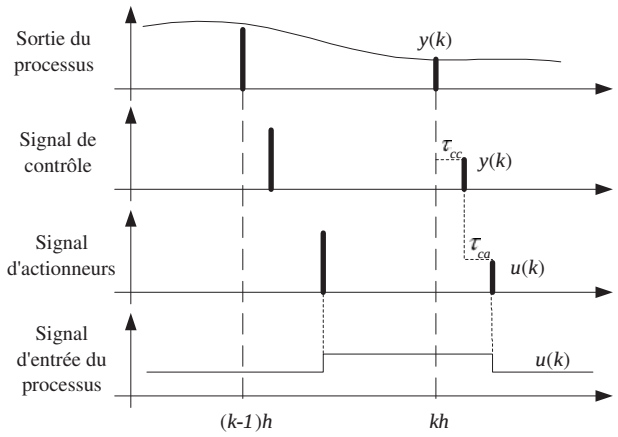

FIGURE 1.8 – Propagation des délais. Le premier diagramme illustre la sortie du processus contrôlé et les instants d'échantillonnage ; le deuxième diagramme illustre le signal reçu par le contrôleur ; le troisième diagramme illustre le signal reçu par les actionneurs ; enfin, le dernier diagramme illustre l'entrée du processus.

de cas expérimentale rapportée dans [Tipsuwan03] nous illustre ces effets que nous détaillons dans la suite.

Dégradation de performance

Le système de commande proportionnelle intégrale illustré par la figure 1.9 (a) est utilisé pour montrer la dégradation de performance induite par le délai, où $R(s)$, $U(s)$, $Y(s)$ et $E(s) = R(s) - Y(s)$ sont respectivement le signal de référence, de contrôle, de sortie et d'erreur dans le domaine de Laplace. Le contrôleur du système étudié est continu, et les fonctions de transfert du contrôleur et du processus contrôlé sans délai de transmission sont respectivement :

$$\begin{aligned} G_c(s) &= \frac{\beta K_p \left(s + \left(\frac{K_I}{K_p}\right)\right)}{s} \\ G_p(s) &= \frac{2029.826}{(s + 26.29)(s + 2.296)} \end{aligned}$$

où $K_P = 0.1701$et $K_I = 0.378$ sont respectivement le gain proportionnel et le gain intégral ; β est le paramètre pour ajuster K_P et K_I, on a par défaut $\beta = 1$; $G_p(s)$ représente un processus qui est un moteur à courant continu [Tipsuwan99].

Sur la figure 1.9 (b), on peut remarquer que plus les délais τ_{cc} et τ_{ca} (avec $\tau_{cc} = \tau_{ca} = \frac{\tau}{2}$) sont longs, plus la performance (temps de réponse, dépassement) du système est dégradée (ce qui est représenté par le dépassement plus important et le temps de réponse plus long).

D'autres types de mesures de performance (par exemple la fonction de coût) peuvent aussi être utilisés pour évaluer la dégradation de performance. Mais en général, plus le délai est long, plus la performance est dégradée [Marti04, Li02, Xia04].

Déstabilisation

Les délais inclus dans la boucle de contrôle peuvent même déstabiliser le système en réduisant sa région de stabilité. Le plan complexe illustré sur la figure 1.10 montre l'influence du délai sur la stabilité du système illustré sur la figure 1.9(a) (Des explications plus détaillées sur la stabilité en utilisant le plan complexe peuvent être trouvées dans [Kuo87]).

Ce plan complexe montre les tracés des pôles du système en boucle fermée en fonction de β pour les délais de transmission différents. Un pôle à droite de l'axe imaginaire signifie que le système correspondant est instable, et le pôle à gauche de l'axe imaginaire implique que le système correspondant est stable. On peut remarquer que plus le délai τ est important, plus l'intervalle de valeurs de β (c'est-à-dire l'intervalle d'ajustement des gains du contrôleur) garantissant la stabilité du système se réduit.

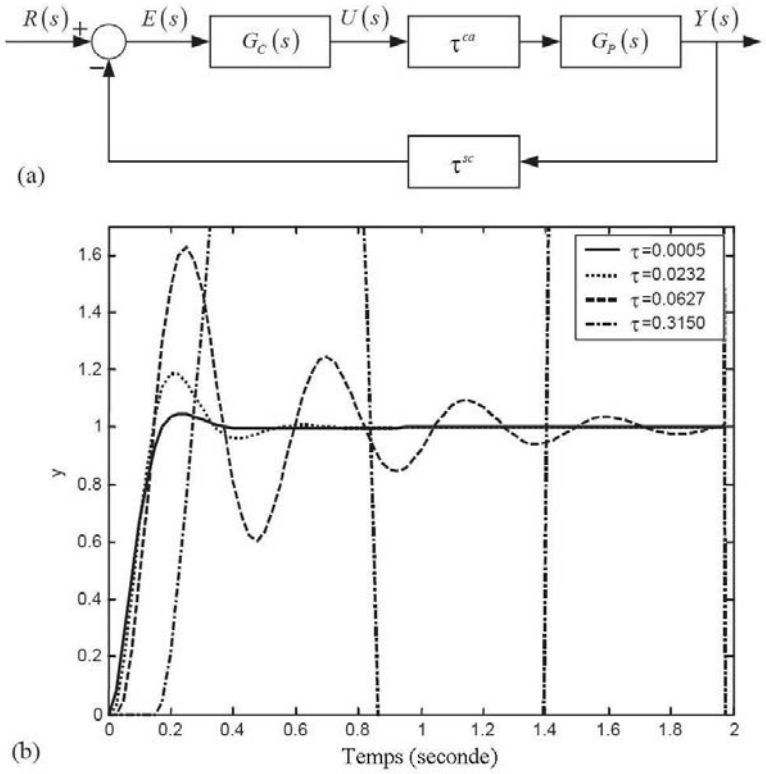

FIGURE 1.9 – La dégradation de performance induit par le délai. (a) la boucle de contrôle avec délai. (b) les réponse y à un échelon ($y_{référence} = 1$) avec différents délais, où $\tau_{cc} = \tau_{ca} = \frac{\tau}{2}$.

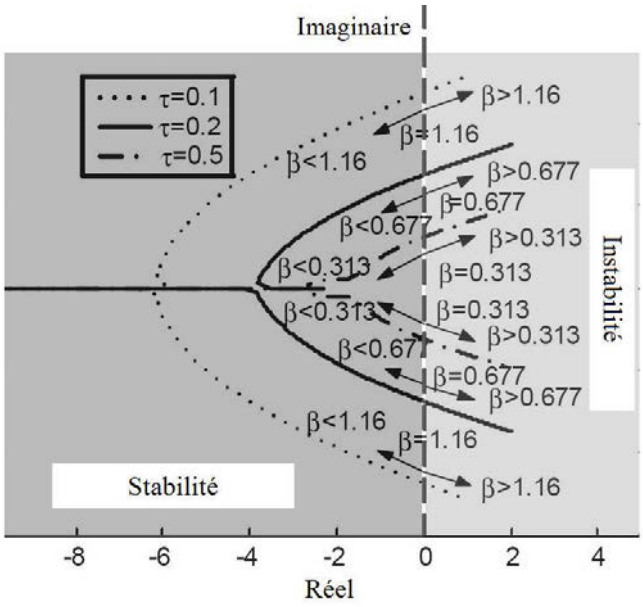

FIGURE 1.10 – Lieu des racines du système illustré par la figure 1.9(a) en fonction de β avec $\tau_{cc} = \tau_{ca} = \frac{\tau}{2}$

1.2.2 Différentes approches (de l'automatique) existantes pour la prise en compte de la plate-forme d'implémentation

A cause de ces effets inattendus induits par le plate-forme d'implémentation, beaucoup de théories conventionnelles bâties sur des hypothèses idéales tels que le contrôle synchrone et la remise de l'échantillon sans retard doivent être réévaluées avant d'être appliquées aux systèmes de contrôle-commande distribué ou avec un CPU partagé avec d'autres applications. Ainsi un nombre important d'études, prennant en compte ces effets induits par le réseau, ont été réalisées. Nous appelons cette catégorie d'approches de l'*automatique* car l'objectif étant la conception de lois de contrôle robustes en prenant en compte les influences de la plate-forme d'implémentation. Dans la suite de ce paragraphe, nous passons dans un premier temps en revue des résultats existants importants, puis formulons nos analyses quant à leur applicabilité et pertinence.

Un premier problème qui a été soulevé par l'introduction d'un réseau dans la boucle de contrôle est la gigue du délai. Un mécanisme de mise en file d'attente de messages à la réception est proposé dans [Luck90, Luck94] pour remodeler les délais aléatoires en délais déterministes, ceci permet de transformer le système temps variant en système temps invariant. La consigne et les mesures précédentes sont stockées dans une file d'attente de type FIFO (First-In-First-Out), et la méthodologie proposée utilise un observateur pour estimer l'état du processus contrôlé et un prédicteur pour calculer la consigne prédictive basé sur les mesures précédentes. Une approche similaire basée sur la mise en file d'attente de messages est proposée dans [Chan95], qui utilise les informations probabilistes ainsi que le nombre de messages dans la file pour améliorer la précision de prédiction de l'état du processus contrôlé.

Un deuxième problème soulevé est lié à la perte de données dans un réseau (erreur de transmission ou problème de congestion). Dans [Halevi88], on considère le système intégré de communication et commande qui est constitué d'un processus en temps continu et d'un contrôleur en temps discret. La synchronisation entre le processus et le contrôleur est supposée occasionnellement perdue. Le problème de stabilité du système est étudié en utilisant l'approche en temps discret : le système est représenté par un vecteur d'état qui contient, en plus de l'état du processus et la sortie du contrôleur actuel, les valeurs précédentes de l'entrée et la sortie du processus, ce qui donne un modèle à dimension finie, temps variant et à temps discret. La perte de message est aussi prise en compte dans l'analyse du système. La méthodologie proposée peut aussi être modifiée afin de supporter les périodes d'échantillonnage non-identiques du capteur et de l'actionneur comme mentionné dans [Liou90] (la perte d'un échantillon est équivalent à multiplier la période d'échantillonnage par deux).

Pour traiter le problème de délai de transmission aléatoire, la méthodologie de contrôle stochastique optimal est proposée dans [Nilsson98]. La méthodologie modélise le problème de contrôle optimal avec délais de transmission sous forme d'un problème de contrôle optimal LQG (linéaire quadratique gaussien) ; le contrôle optimal et l'estimateur optimal basé sur un filtre de Kalman sont donnés en supposant que les délais puissent être modélisés par le

processus i.i.d (indépendant et identiquement distribué) et donc par une chaîne de Markov. Cette approche est considérée comme un résultat important de la prise en compte de l'implémentation lors de la conception de lois de contrôle. Nous pouvons noter néanmoins que l'hypothèse du délai i.i.d n'est pas toujours vérifiée lorsque l'on considère un réseau concret dans la pratique. En fait, la variation de délai dans un réseau est souvent liée au phénomène de congestion dont l'impact est plutôt en rafale.

L'effet d'échantillonnage et le délai de transmission sont pris en compte dans [Seuret04] [Seuret05][Seuret06], où une méthodologie basée sur l'optimisation LMI (linear matrix inequality) est proposée pour concevoir le contrôleur ainsi que l'observateur de l'état du process de façon qu'ils garantissent la stabilité exponentielle du système global.

Le problème de stabilité du système avec perte de messages est traité dans [Zhang01], où le réseau est représenté par un interrupteur avec un certain taux de changement d'état. En étendant l'analyse de stabilité du système asynchrone dynamique mené dans [Hassibi99], une condition suffisante de stabilité est donnée.

Dans [Hadjicostis02], l'auteur considère un système mono-dimensionnel doté d'un réseau non fiable qui relie le capteur et le contrôleur. Les messages sont perdus aléatoirement sur le réseau suivant le processus i.i.d. Lorsqu'un message est perdu sur le réseau, le contrôleur envoie la valeur zéro à l'actionneur. Une approche analytique est donnée pour déterminer la stabilité du système étant donné la probabilité de perte. Le problème de stabilité du système avec la perte aléatoire de messages est aussi étudié dans [Seiler01], où le processus de perte de messages est modélisée par un processus de Markov.

Dans [Ling02, Ling03], la performance du système est caractérisée par "power semi-norm" $\sqrt{\text{Trace}\left(\text{E}\left[x\left[n\right]x\left[n\right]^{T}\right]\right)}$ qui est similaire à la fonction de coût introduite précédemment (1.3). En supposant un modèle probabiliste de perte de messages (i.i.d et processus de Markov), la performance du système définie en fonction du taux de perte est analysée. Enfin, dans [Lemmon03, Li02], en supposant que le processus de perte de messages offrant la meilleure performance puisse être modélisé par un processus de Markov, une approche d'ordonnancement de messages est proposée pour ordonnancer autant que possible les messages selon ce processus.

Dans [Gupta04, Gupta05], on considère un système distribué avec la perte aléatoire de messages. Le problème de commande optimale LQG est résolu en appliquant le principe de séparation du problème LQG [Kailath99].

Les études antérieures ont apporté beaucoup de résultats utiles, néanmoins elles souffrent de certaines limitations méthodologiques ou des hypothèses fortes.

Premièrement, certaines études se limitent à un type spécifique de configuration de réseau. En pratique, les réseaux utilisés dans les systèmes de commande en réseau sont de types variés, citons par exemple DeviceNet [Lawrenz97], Ethernet [Andrew02], FireWire [Anderson99]. Chaque type de réseaux possède son protocole de communication conçu pour un certain type d'applications. La performance d'un système dépend largement des caractéristiques du réseau

telles que le debit de transmission, le protocole d'accès au médium, la longueur de messages, la méthode de résolution de congestion ou encore la façon de traiter des erreurs de transmission (retransmission automatique ou non en cas de pertes). Par conséquent, si la configuration du réseau sur laquelle les études sont menées a changé, les résultats d'études ne resteraient plus valides.

Deuxièmement, bien que la conception du système de contrôle-commande sur une plate-forme informatique temps réel nécessite une bonne connaissance sur l'intégration du domaine du contrôle-commande et du domaine de l'informatique temps réel, la plupart des études n'a pas eu de vraies relations entres les deux domaines. La conception du système a été pratiquement menée en deux étapes séparées et en isolation : d'une part, la conception du système de contrôle-commande ; d'autre part, la conception de l'informatique temps réel. La conception du système de contrôle-commande fait généralement abstraction des aspects d'implémentation. Elle est souvent menée en s'appuyant sur l'hypothèse de déterminisme temporel du délai de transmission. Or les plate-formes d'implémentation ne permettent pas souvent d'assurer ce déterminisme temporel. Dans le cas du partage de la ressource entre plusieurs systèmes, il se peut que la ressource (le processeur ou le réseau) partagée entre plusieurs systèmes soit surchargée par les demandes de traitement (l'exécution des tâches ou la transmission des messages). Dans le cas de surcharge, les délais de calcul ou de transmission subis par les systèmes vont varier de manière imprévisible, l'hypothèse du déterminisme temporel voire du comportement Markovien devient ainsi absurde et aucune garantie de performance du système ne sera assurée. D'autre part, la conception de l'informatique temps réel se fait en général sans la compréhension de la spécificité des systèmes de contrôle-commande avec, en particulier, la conséquence du déploiement (stratégie d'ordonnancement, détermination des caractéristiques temporelles) des tâches temps réel concernées (les capteurs, actionneurs et le contrôleur) sur la performance du système car cette dernière dépend dans une large mesure des caractéristiques temporelles de la boucle de contrôle. Par conséquent, la conception isolée du système de contrôle-commande d'une part, et de l'ordonnancement des tâches temps réel associées au système d'autre part conduit nécessairement à des solutions non-optimales (e.g. problème de surdimensionnement de ressources).

1.3 Conception conjointe

1.3.1 Présentation du problème

Au fur et à mesure que la capacité des matériels informatiques augmente et que leur coût diminue, de plus en plus de fonctionnalités sont réalisées au moyen d'informatique. Il y a une tendance à partager une ou plusieurs ressources (réseau ou processeur) entre plusieurs boucles de contrôle dans une application de contrôle-commande. Ainsi, on peut trouver par exemple que dans une application embarquée (cf, la figure 1), les tâches temps réel implémentant les contrôleur (appelées tâches de contrôle) s'exécutent parallèlement sur un processeur.

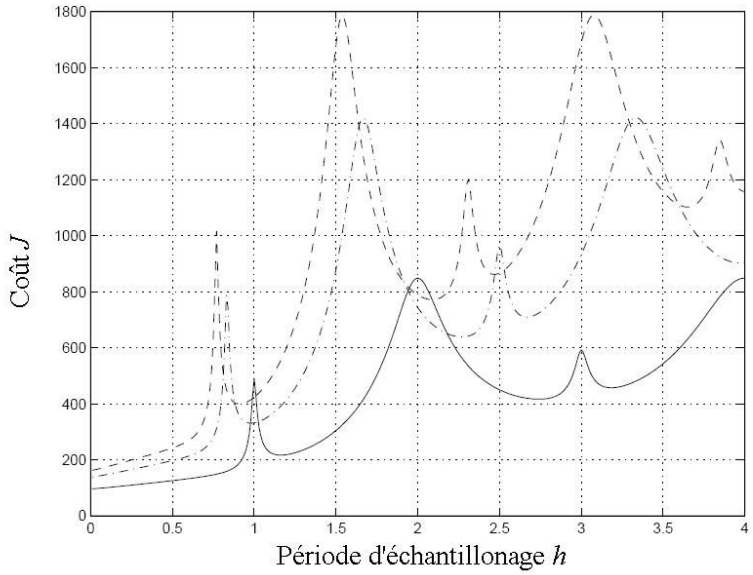

FIGURE 1.11 – Fonction de coût en fonction de la période d'échantillonnage

Pour un autre exemple, on pourra se reporter à l'application de contrôle commande distribué (cf, la figure 2) dans laquelle plusieurs boucles de contrôle sont fermées via un réseau de communication commun.

Cependant, le partage de la ressource augmente la charge de cette dernière. Dans le pire cas, la ressource peut être surchargée momentanément par les demandes de traitement arrivées en rafale. Les boucles de contrôle participant au partage de la ressource souffrent d'un délai de traitement long ou avec une grande gigue, par conséquence, la performance ou même la stabilité des boucles de contrôle est dégradée comme montré dans la section précédente. Avec les matériels informatiques toujours plus puissants, on pourra être convaincu que la plupart des problèmes temporels peuvent être résolus en mettant à jour le processeur ou les composants du réseau. Bien que ceci puisse être vrai dans certains cas, la conception du système est souvent soumise à une forte contrainte économique. Le concepteur est obligé de partager un processeur ou un noeud de communication du réseau déjà fortement chargé avec plus de tâches ou messages. C'est notamment le cas dans la conception des systèmes embarqués dans l'automobile par exemple où le nombre de calculateurs qu'on peut embarquer est tout simplement limité. L'effet de surcharge reste toujours à envisager à la phase de conception de l'application.

Une autre solution pour supprimer ou diminuer les effets inattendus du partage de la ressource est de distribuer raisonnablement la ressource partagée entre les boucles de contrôle. Ceci peut être réalisé en ajustant les caractéristiques temporelles (par exemple, la période d'échantillonnage) des boucles de contrôle. En observant sur la figure 1.11 (extraite de [Eker00]) des fonctions de coût en fonction de la période d'échantillonnage h, correspondant à la commande de trois pendules aux paramètres différents, on constate que la performance de la

boucle de contrôle dépend largement de la fréquence des demandes de traitement, et ainsi du déploiement des tâches temps réel implémentant les composants des boucles de contrôle. Par conséquent, la conception des systèmes de contrôle-commande nécessite d'être menée en tenant en compte non seulement de l'exigence de performance de l'application, mais aussi du déploiement des tâches temps réel correspondantes. Ce qui donne naissance à une nouvelle stratégie de conception des systèmes de contrôle-commande, à savoir la conception conjointe du contrôle et de déploiement des tâches temps réel. Rappelons que le problème de la conception conjointe *commande* et *ordonnancement* est formellement défini comme suit. Etant donné un ensemble de processus à contrôler et un processeur (ou un noeud de communication) où s'implémentent les tâches de contrôle (ou se transmettent les messages nécessaires au calcul de la loi de commande), la conception conjointe (ou co-conception) consiste à concevoir un ensemble de contrôleurs et la stratégie d'ordonnancement des tâches de contrôle (ou messages) de telle sorte que la QdC globale soit optimisée.

La stratégie de la conception conjointe a attiré, dans ces derniers temps, l'attention dans le domaine de l'automatique aussi bien que dans le domaine d'informatique, et un certain nombre de travaux ont été effectués en se basant sur le principe de conception conjointe. Dans la section suivante, nous allons discuter en détail ces travaux.

1.3.2 Etat de l'art sur les approches de la conception conjointe

Dans cette section, nous présentons l'état de l'art de la conception conjointe. Dans un premier lieu, les travaux les plus cités sur la conception conjointe sont présentés, puis les limites de ces approches seront analysées.

Grosso modo, il existe aujourd'hui trois familles d'approches basées sur la conception conjointe pour la gestion de la distribution de ressources : les approches basées sur le réglage de périodes d'échantillonnage, les approches basées sur le rejet de demandes de traitement et les approches basées sur le réglage de priorités de demandes de traitement. Nous discutons en détail ces trois familles d'approches dans la suite.

Approches basées sur le réglage de périodes d'échantillonnage

Les premiers travaux qui ont traité le problème de conception conjointe peuvent être trouvés dans [Seto96]. L'auteur considère l'application centralisée dont la structure est illustrée par la figure 1. La performance de chaque contrôleur est représentée par une fonction de coût en fonction de la période d'échantillonnage :

$$J(f) = \lim_{H \to \infty} \int_0^H \left(x^T(t) Q x(t) + u^T(t) R u(t) \right) dt$$

où f est la fréquence d'échantillonnage, et la fonction est supposée pouvoir être approchée par une fonction exponentielle :

$$J(f) = ae^{-\beta f}$$

où a est le coefficient de grandeur, et β est le taux d'affaiblissement.

L'approche proposée calcule la période de chaque tâche de contrôle afin que la performance globale de l'application soit optimisée sous contrainte d'ordonnançabilité des tâches. Ceci consiste à résoudre le problème d'optimisation suivant :

$$\begin{array}{l} \min_{(f_1,..,f_n)} \sum_{i=1}^{n} w_i J(f_i) = \sum_{i=1}^{n} w_i a_i e^{-\beta_i f_i} \\ \text{sous contrainte : } \sum_{i=1}^{n} C_i f_i \leq A \end{array}$$

où n est le nombre de tâches partageant le processeur ; w_i est une pondération de l'importance de chaque contrôleur ; C_i est le temps d'exécution de la tâche de contrôle i ; A est le seuil de charge garantissant l'ordonnançabilité, la valeur de A varie selon la politique d'ordonnancement avec $0 < A \leq 1$; $\sum_{i=1}^{n} C_i f_i \leq A$ est la condition d'ordonnançabilité.

A cause de la complexité importante de l'algorithme proposé, cette approche ne peut être utilisée qu'hors-ligne, ne permettant pas ainsi une adaptation au changement de la configuration de l'application (par exemple l'activation d'une nouvelle tâche de contrôle). Afin de surmonter ce défaut, l'approche proposée dans [Seto96] est étendue dans [Eker00]. Comm dans [Seto96], la performance de chaque contrôleur est représentée par la fonction de coût en fonction de la période d'échantillonnage :

$$J(h) = \lim_{H \to \infty} \int_0^H \left(x^T(t) Q x(t) + u^T(t) R u(t) \right) dt \tag{1.5}$$

où h est la période d'échantillonnage. Ainsi, calculer la période optimale pour chaque contrôleur consiste à résoudre le même problème d'optimisation suivant :

$$\begin{array}{l} \min_{(h_1,..,h_n)} \sum_{i=1}^{n} J(h_i) \\ \text{sous contrainte : } \sum_{i=1}^{n} C_i f_i \leq A \end{array} \tag{1.6}$$

Les auteurs montrent dans un premier temps que le calcul des périodes optimales nécessite la résolution d'équations de Lyapunov et de Riccatti qui est très coûteuse, ce qui empêche une adaptation rapide des périodes au changement de configuration de l'application. Pour diminuer la complexité temporelle de l'algorithme d'optimisation, l'auteur suppose que la fonction de coût puisse être approchée par une fonction quadratique :

$$J(h) = \alpha + \beta h^2$$

En utilisant celle-ci pour représenter la performance de contrôle de chaque tâche de contrôle, le calcul des périodes optimales devient particulièrement simple, et ainsi permet la mise en oeuvre du mécanisme de régulation en ligne qui ajuste les périodes des tâches de contrôle en fonction de la configuration de l'application.

Cette approche est davantage étendue dans [Cervin02], où l'auteur a proposé un mécanisme d'ordonnancement avec rétroaction-anticipation pour améliorer la vitesse de réaction

à un changement de configuration de l'application. Concrètement, le mécanisme d'ordonnancement en ligne proposé est constitué d'un régulateur et d'un ordonnançeur conventionnel (e.g., un ordonnançeur EDF ou RM [1]). Le régulateur fonctionne à un rythme plus fréquent que l'ordonnançeur et il surveille la configuration courante de l'application. Lors de la détection de la tendance à un changement de configuration, les périodes optimales pour la nouvelle configuration de l'application sont calculées. En plus de l'approximation quadratique, l'auteur suppose que la fonction de coût de certains systèmes peut être approchée par une fonction linéaire :

$$J(h) = \alpha + \beta h$$

Cette hypothèse permet de réduire davantage la complexité de temps pour calculer les périodes optimales.

L'approche de [Eker00] est aussi étendue dans [Henriksson05] où les périodes des tâches de contrôle sont calculées en fonction des états courants des processus contrôlés. Avec l'approche proposée, les perturbations (ou références) affectant certaines boucles de contrôle impliquent que les tâches de contrôle correspondant s'exécutent plus fréquemment que celles pour les boucles en état d'équilibre.

Dans [Palopoli02], l'auteur considère des processus du premier ordre. Un algorithme d'optimisation est proposé pour dériver les gains et les périodes des contrôleurs pour qu'un certain critère de robustesse soit respecté sous la contrainte d'ordonnaçabilité. L'application de l'approche proposée est limitée car elle ne traite que les processus du premier ordre.

Dans [Simon05b], en partant du fait que toutes les composantes de l'algorithme de commande n'ont pas la même importance vis-à-vis de leur impact sur la performance de contrôle, l'algorithme de commande est décomposé en un certain nombre de tâche afin de minimiser la latence de calcul. L'exécution de ces tâches est supposée être interférée par des tâches perturbatrices, et les auteurs proposent une méthodologie qui ajuste en ligne les périodes des tâches pour maximiser l'utilisation du processeur.

L'effet de latence de calcul a été pris en compte dans [Ryu97, Ryu98], où la performance de contrôle (exprimée en termes de dépassement, temps de montée et temps de réponse) est exprimée en fonction de la période de la tâche de contrôle et de la latence de calcul. Un algorithme heuristique basé sur l'approche de calibrage de période (PCM) [Gerber95] est proposé pour ajuster les paramètres d'implantation de façon à optimiser la performance sous la contrainte d'ordonnançabilité.

Approches basées sur le rejet de demandes de traitement

Dans [Ramanathan99], l'auteur considère une application qui comprend un certain nombre de boucles de contrôle. Les contrôleurs sont implémentés par des tâches temps réel qui s'exécutent sur deux processeurs. En cas de panne d'un processeur, l'exécution de toutes les tâches

1. Les politiques d'ordonnancement EDF et RM seront étudiées en détail dans la section 2.1.3.1

est prise en charge par l'autre processeur, se pose ainsi le problème de surcharge. Pour éviter la surcharge du processeur, la proposition consiste à sélectivement rejeter des instances des tâches afin de diminuer la charge du processeur. Comme le rejet d'instances conduit à une dégradation de performance, l'auteur donne ensuite une approche de conception du contrôleur optimal sous le rejet d'instances pour réduire la dégradation de performance.

Le système de contrôle-commande avec les capteurs et actionneurs distribués est considéré dans [BenGaid05, BenGaid06a], où la bande passante du réseau est une ressource limitée et ne permet pas pour chaque période d'échantillonnage de transmettre les mesures de tous les capteurs au contrôleur, et de mettre à jour la consigne de chaque actionneur. L'approche proposée consiste, d'une part à sélectivement transmettre les informations jugées nécessaires sur le réseau, et d'autre part à déterminer la loi de commande optimale en tenant compte de la séquence d'informations impliquée afin que la dégradation de performance due à l'insuffisance d'informations soit minimisée.

Approches basées sur le réglage de priorités des demandes de traitement

Dans [Zhao99], les auteurs considèrent l'application dans laquelle plusieurs processus sont contrôlés par un seul contrôleur. A tout moment, le contrôleur ne peut servir qu'un seul processus. Une stratégie d'ordonnancement est proposée qui consiste à choisir le processus avec le plus grand écart entre la performance effective et la performance espérée à contrôler, alors que la sortie des autres contrôleurs dans l'application est mise à zéro. Basé sur cette stratégie d'ordonnancement, un série de méthodologies sont proposées pour calculer les paramètres des systèmes, notamment les priorités, afin de garantir la stabilité des systèmes et de satisfaire l'exigence de performance.

Dans [Rehbinder00], on étudie le problème du partage du processeur par un ensemble de contrôleurs qui contrôlent chacun un processus. Un algorithme est proposé qui détermine, étant donné la durée d'exécution de l'application ainsi que les caractéristiques temporelles des tâches de contrôle, une séquence d'exécution périodique des tâches de telle sorte que la somme des fonctions de coût soit minimisée. Une approche similaire est proposée dans [Lincoln00, Lincoln01, Lincoln02]. À la différence de l'approche précédente, l'ordre d'exécution des tâches n'est pas nécessairement périodique. Le problème de stabilité du système de contrôle-commande avec la période d'échantillonnage variable est étudié dans [Schinkel02]. Dans cet article, on considère qu'un processus est contrôlé par plusieurs contrôleurs conçus avec des périodes d'échantillonnage différentes. Il est montré que même si chaque contrôleur mène à un système stable, la permutation entre les contrôleurs durant l'exécution du système peut rendre le système instable. Ainsi, l'auteur propose une approche pour concevoir les contrôleurs de telle sorte que le système reste stable quelle que soit la séquence de permutations entre les contrôleurs. Dans [Hetel06], on considère le système à commutation linéaire qui est contrôlé par un ensemble de contrôleurs, l'approche proposée consiste à concevoir les contrôleurs stabilisant le sysème en présence du délai de transmission induit par le réseau.

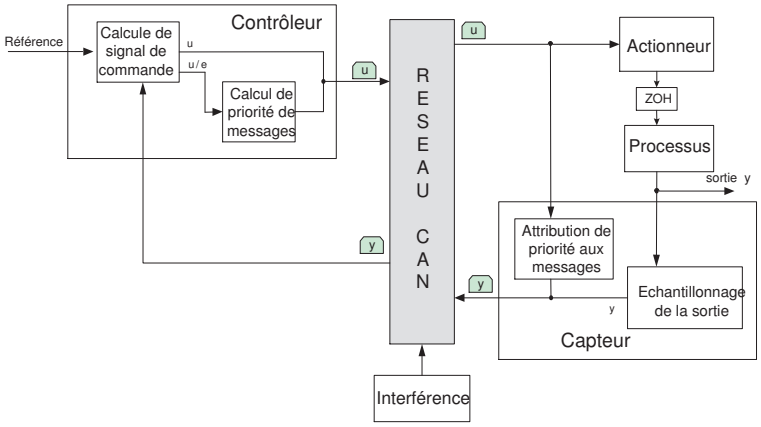

FIGURE 1.12 – Relation entre les signaux à temps continu et à temps discret dans une boucle de contrôle.

Dans [Juanole07], on considère un système de contrôle-commande monodimensionnel dont la boucle de rétroaction est fermée sur un réseau CAN (c.f. la figure 1.12). Le réseau est partagé par un certain nombre d'applications. Les messages contenant les informations de contrôle sont échangés sur le réseau sous l'interférence des messages de plus fortes priorités provenant d'autres applications, subissant ainsi un délai de transmission. Pour diminuer l'influence du délai sur la performance de contrôle représentée par la fonction de coût $J = \int_0^T t\,(r\,(t) - y\,(t))^2\,dt$ avec $r\,(t)$ la référence de sortie à l'instant t, l'approche proposée consiste à dynamiquement ajuster la priorité des messages en fonction de l'état courant du processus contrôlé.

Mis à part de ces trois familles d'approches mentionnées ci-dessus, il existe aussi d'autres types d'approches utilisant le principe de conception conjointe. Dans [Walsh02][Walsh99] [Walsh99a][Walsh99b], on considère que le réseau est partagé par plusieurs boucles de contrôle. Le réseau est utilisé pour connecter les capteurs et le contrôleur, et la donnée de chaque capteur est envoyée directement au contrôleur sans être encapsulée dans un message avec les données des autres capteurs. Au lieu d'étudier le problème de déploiement des tâches temps réel comme font les approches ci-dessus, les auteurs introduisent la notion d'intervalle de transfert maximal autorisé (noté MATI pour maximum allowable transfer interval) qui définit le temps maximal entre deux réceptions consécutives de messages, et proposent le protocole de communication TOD (pour Try-Once-Discard) pour ordonnancer les demandes d'accès au réseau partagé sous la contrainte de MATI. Avec le protocole TOD, si une demande d'accès au réseau ne peut pas être acceptée à cause des autres demandes plus prioritaires, elle est rejetée. Une approche est ensuite proposée pour calculer la borne supérieure du MATI garantissant la stabilité des systèmes.

1.3.3 Limites des approches précédentes

Bien que les approches basées sur le réglage de périodes mentionnées ci-dessus aient apportés des résultats utiles pour certaines applications centralisées, ces approches souffrent d'un certain nombre de limites lorsqu'il s'agit d'un réglage en ligne des périodes. Premièrement, les fonctions de coût des tâches de contrôle sont supposées convexes ou pouvant être approchées par une fonction convexe pour réduire la complexité en temps et en espace de l'algorithme d'optimisation. Cependant, comme déjà expliqué dans l'introduction générale, cette hypothèse n'est pas toujours valide, car dans la pratique, il est parfois difficile de garantir que ces fonctions sont toutes convexes. En observant les caractéristiques des fonctions de certains systèmes, il est clair que tout système n'a pas la fonction de coût convexe : certains systèmes ont la fonction de coût concave, ou même ni convexe ni concave (voir la figure 1.11). Deuxièmement, pour les systèmes purement gérés par l'horloge, lorsque des tâches sont soumises à des contraintes de précédence, le changement de période d'une tâche implique souvent le changement de période des autres tâches liées. Enfin, si la ressource partagée est un noeud de communication dans le réseau, la technique de réglage en ligne des périodes d'échantillonnage représente un coût élevé car elle implique l'implémentation d'un mécanisme de communication du noeud de réseau vers les capteurs des processus (souvent non prévu) pour ralentir le rythme d'émission des capteurs (une sorte de contrôle de congestion). Même s'il est réalisable (avec l'option "source quench" du protocole ICMP ou le contrôle de congestion "slow-start" et "congestion-avoidance" de TCP par exemple), la variation de délai est souvent difficile à prévoir rendant difficile la garantie de la stabilité de la boucle de contrôle.

Quant aux approches basées sur le réglage de priorités, la plupart des approches dans cette catégorie ne peuvent être utilisés qu'en hors-ligne à cause de la complexité importante des algorithmes. De plus, contrairement aux approches de réglage de périodes et de rejet des échantillons, les approches basées sur le réglage de priorités ne permettent pas de résoudre le problème de surcharge de la ressource partagée. En effet, le changement de priorités peut certes favoriser le traitement des tâches (ou messages) prioritaires, mais la charge globale de la ressource est toujours maintenue. Bref, on ne fait que déplacer le pic de la charge au lieu de l'éliminer.

1.4 Solution envisagée dans cette thèse

Suite aux constats que la conception disjointe des systèmes de contrôle-commande et le déploiement des tâches temps réel associées conduit à une solution non-optimale, les études de cette thèse sont menées sur la base de la conception conjointe de ces deux aspects afin d'obtenir une solution optimale. De plus, en raison des limites des approches existantes basées sur le principe de régulation de périodes et de priorités, les approches envisagées dans cette thèse s'appuient sur la stratégie de rejet qui consiste à maintenir l'optimalisation de la performance de l'application sous contrainte de la ressource limitée en rejetant sélectivement les

instances des tâches de contrôle ou les messages. Ces rejets sont spécifiés en utilisant le modèle (m,k)-firm initialement proposé par Hamdaoui et Ramanathan [Hamdaoui94, Hamdaoui95]. La contrainte (m,k)-firm consiste à imposer le respect des échéances de m instances (messages) parmi k instances (messages) consécutives quelconques et permet de spécifier explicitement le motif (ou pattern) des rejets (ou non respect des échéances), ce qui le rend particulièrement intéressant pour des systèmes de contrôle-commande qui tolèrent le non respect des échéances de certaines instances ou messages jusqu'à une certaine limite sans affecter leur fonctionnement correct. Cette stratégie permet de surmonter les problèmes mentionnés ci-dessus car comme déjà expliqué dans l'introduction générale :

- Les rejets des instances des tâches ou des messages peuvent être considérés comme la multiplication de la période de la tâche. Ainsi déterminer la stratégie optimale de rejet consiste à calculer ces périodes multiples, ce qui peut être modélisé en un problème d'optimisation discrète pour lequel il existe des solutions qui ne dépendent pas de la convexité du problème et qui sont efficaces en temps aussi bien qu'en espace.
- En cas de tâches sous la contrainte de précédence, comme la période nominale de chaque tâche reste constante durant l'exécution de l'application, l'adaptation de périodes au moment de la régulation n'est plus nécessaire pour les systèmes purement gérés par l'horloge.
- Le mécanisme de rejet dynamique de messages dans un noeud de communication est plus facile à implémenter par rapport au changement de période d'échantillonnage, car les différents composants de la boucle de contrôle sont totalement indépendants quelle que soit la stratégie de rejet.

Pour avoir une idée intuitive sur la différence et la similitude des approches basées sur la régulation de périodes et de rejets citées ci-dessus, la relation entre la QdC et la période d'échantillonnage ainsi que celle entre QdC et contrainte (m,k)-firm sont illustrées par la figure 1.13. P_{max} et P_{min} sont deux limites de la période du système (au delà de ces deux limites, le fonctionnement correct du système ne peut plus être garanti), et qui sont souvent déterminées en appliquant la théorie de contrôle conventionnelle (par exemple la règle donnée dans [Astrom97]), et la période nominale $P_{nominal}$ est choisie entre P_{max} et P_{min}. Au moment de la surcharge du processeur, les approches basées sur la régulation de la période peut agrandir la période jusqu'à P_{max}. La relation entre QdC et la contrainte (m,k)-firm peut être illustrée de la même façon en simplement remplaçant P_{max} par la contrainte (m,k)-firm et $P_{nominale}$ par la contrainte (k,k)-firm. Bien que l'espace de la régulation de la contrainte (m,k)-firm ne soit pas aussi lisse et large que celle de la régulation de période, les inconvénients des approches basées sur la régulation de période sont éliminés.

Il faut comprendre bien entendu que le réglage de rejets est une approche complémentaire à celle du réglage direct de périodes. En réalité, le rejet d'un échantillon est équivalent à multiplier la période d'échantillonnage. Du point de vue du contrôle, il s'agit alors d'étudier le contrôle à période d'échantillonnage variable pour les deux approches. Ce qui différencie notre approche de celle de la régulation de périodes est simplement dans la façon d'atteindre

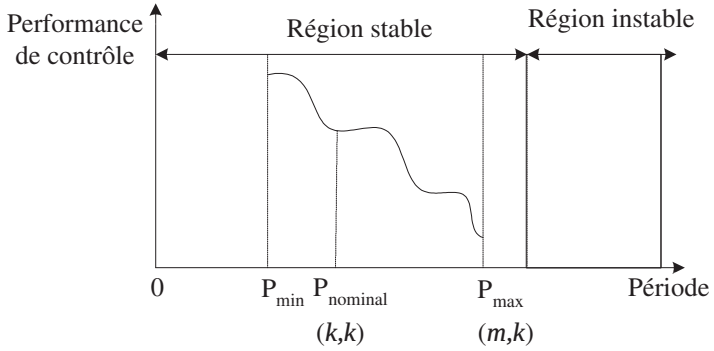

FIGURE 1.13 – Le partage d'un réseau par un certain nombre de boucles de contrôle et d'autres applications

l'objectif de changement de périodes. Le réglage de rejet est plus simple à implémenter. En revanche, dans notre approche la variation de périodes qui en résulte ne peut être que des multiples de la période de base (réglage discret), tandis qu'avec la régulation directe de périodes, on obtient un réglage continu.

Nous présenterons dans le chapitre 2 une vue d'ensemble sur les travaux réalisés autour de (m,k)-firm.

Chapitre 2

Ordonnancement temps réel sous contrainte (m,k)-firm

Dans ce chapitre, nous proposons tout à bord, pour offrir un confort de la lecture, un résumé des notions préliminaires sur l'ordonnancement temps réel (nous renvoyons le lecteur à l'annexe A pour une description détaillée des tâches temps réel). Puis dans la seconde partie nous présentons un l'état de l'art sur les travaux autour du modèle (m,k)-firm. Dans la trosième partie nous donnons deux conditions suffisantes d'ordonnançabilité pour des tâches non-préemptives à priorité fixe et sous contrainte (m,k)-firm afin qu'on dispose de contrainte d'implémentation lors de la conception conjointe de commande et d'ordonnancement de messages.

2.1 Ordonnancement et Analyse d'ordonnancement

Dans la communauté de l'ordonnancement, le temps est une grandeur physique mesurable (espace discret ou dense) qui permet de caractériser les éléments du système, dont les traitements font partie (on parle de tâches). Le système est modélisé sous la forme de demandeurs de ressources (les tâches, messages) d'un côté, et de serveurs de ressources de l'autre, le tout étant arbitré par un composant central : le support d'exécution (en général un système d'exploitation). La capacité des serveurs correspond à une densité temporelle (puissance du processeur, bande passante du réseau). Et les demandeurs en ressources ont des contraintes de validité fonctionnelles et temporelles qui sont issues de la dynamique du processus physique contrôlé et ces contraintes temporelles devront être prises en compte tout au long du cycle de vie de l'application, en même temps que des besoins en ressources quantifiés.

Effectuer un ordonnancement sur une ressource (processeur ou un noeud de communication) consiste à déterminer pour un ensemble d'opérations provenant des différents demandeurs en ressource dans quel ordre relativement à une date, chacune d'elles doit être exécutée sur la ressource. Un ordonnancement est faisable si toutes les contraintes temporelles associées à chaque demandeur en ressource sont respectées sur la ressource disponible. Suivant les

caractéristiques du modèle de tâches et du système, un ordonnancement faisable peut exister ou non. L'objectif que poursuit l'analyse d'ordonnancement est d'établir l'ordonnançabilité du modèle, et/ou la faisabilité d'un ordonnancement donné.

Dans la suite, nous introduisons quelques éléments de terminologie, puis nous donnerons une série de résultats qui seront utiles pour la suite de cette thèse. En règle générale, le mécanisme d'ordonnancement sera le même quelle que soit la nature de demandeurs de resources (tâches dans un processeur, messages dans un réseau). Dans la suite, sauf précision en cas de nécessité, nous ne parlerons que des tâches dont l'exécution se fait sur un processeur.

2.1.1 Classes d'ordonnancements

On indique ici les caractéristiques liées à l'ordonnancement, qui forment les paramètres du problème d'ordonnancement qu'on cherche à résoudre.

Préemptif/non-préemptif

L'ordonnancement (en-ligne ou hors-ligne) est préemptif si la tâche en cours peut perdre le processeur au profit d'une autre tâche jugée plus urgente. L'ordonnancement préemptif est supporté par la plupart des systèmes d'opération, par contre, l'ordonnancement des messages dans un noeud de communication est en général non-préemptif.

En-ligne/hors-ligne

Un ordonnancement hors-ligne signifié que la séquence d'ordonnancement est prédéterminée à l'avance. En pratique, l'ordonnancement prend la forme d'un plan hors-ligne (ou statique), exécuté de façon répétitive (on parle aussi d'ordonnancement cyclique, qui définit le cycle majeur) : à chaque top d'horloge une tâche particulière du cycle majeur courant est exécutée jusqu'à terminaison ou jusqu'au top d'horloge suivant. L'ordonnancement hors-ligne introduit un surcoût d'exécution lié à l'ordonnancement très faible [Kopetz92], car les traitements pour l'établissement du plan sont effectués hors-ligne, et peuvent se permettre d'être très complexes. Nous ne détaillons pas davantage ce type d'ordonnancement dans ce travail. Dans la suite de la thèse, nous nous intéressons aux algorithmes d'ordonnancement en-ligne.

Un ordonnancement en-ligne correspond au déroulement d'un algorithme qui tient compte des tâches effectivement présentes dans la file d'ordonnancement lors de chaque décision d'ordonnancement. Les ordonnanceurs en-ligne peuvent reposer sur la construction de plans d'ordonnancements en cours de fonctionnement, auquel cas ils sont dits à plan dynamique [Kaneko96]. Mais plus couramment, la majorité des ordonnanceurs temps-réel en-ligne reposent sur la notion de priorité : lors de chaque décision d'ordonnancement, la tâche de plus haute priorité est élue. Les priorités peuvent être attribuées hors-ligne, auquel cas l'algorithme d'ordonnancement est dit à priorité statique ou fixe ; ou en-ligne, auquel cas il est dit à priorité dynamique. Ce type d'ordonnanceur est plus complexe que les ordonnanceurs à plan statique : les décisions d'ordonnancement consistent à élire la tâche prête de plus haute priorité. Dans

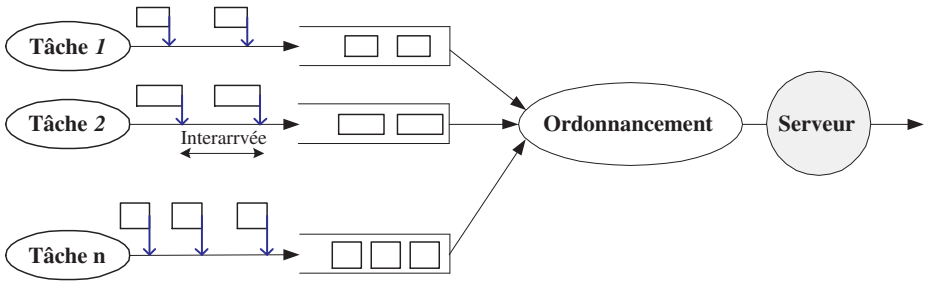

FIGURE 2.1 – Le modèle général pour la description des systèmes : MIQSS

le cas des ordonnanceurs à priorité dynamique, il s'agit en plus de déterminer les priorités relatives des différentes tâches prêtes.

2.1.2 Modèle de système considéré

Dans cette étude, nous nous appuyons sur le modéle MIQSS (Multiple Input Queues Single Server) illustré par la figure 2.1. Le système est soumis à des demandes d'accès provenant d'un ensemble de tâches. Le problème fondamental dans de tels systèmes est d'ordonnancer des demandes d'accès au serveur, tout en satisfaisant leurs contraintes temporelles.

2.1.3 Solutions aux problèmes d'ordonnancement

Dans cette partie, nous donnons tout d'abord deux algorithmes d'ordonnancement courants, puis des conditions d'ordonnançabilité qui leur sont associées seront présentées.

2.1.3.1 Algorithmes d'ordonnancement

Nous nous plaçons ici dans le cadre simple de l'ordonnancement monoprocesseur. Les ordonnancements à priorités les plus courants sont les suivants :

- ***EDF*** (pour Earliest Deadline First) : ordonnancement à priorité dynamique. Une tâche est d'autant plus prioritaire que sa date d'échéance absolue est proche de la date courante.
- ***RM*** (pour Rate Monotonic) : ordonnancement à priorité statique pour tâches avec échéance relative égale à la période. Une tâche est d'autant plus prioritaire que sa période est petite.

2.1.3.2 *Quelques conditions de faisabilité*

Dans la suite, nous donnerons quelques conditions de faisabilité pour un ensemble de tâches dont les contraintes temporelles sont définies par l'échéance des tâches. Notons que si la date d'activation de la première instance de la tâche est donnée, la tâche est dite concrète ; si les dates d'activation de la première instance de toutes les tâches sont identiques, les tâches

sont dites synchrones. Et les notations suivantes seront utilisées : dans le système de N tâches, chaque tâche τ_i est caractérisée par les paramètres suivants :

- T_i : l'intervalle de temps entre deux instances consécutives de τ_i , désigné sous le nom de période ;
- C_i : le temps maximum nécessaire pour finir l'exécution de chaque instance de τ_i, désigné sous le nom de temps d'exécution ;
- D_i : l'échéance de chaque instance de τ_i.

Le taux d'utilisation du système de tâches s'écrit alors : $U = \sum_{i=}^{N} \frac{C_i}{T_i}$, et on rappelle qu'une condition nécessaire pour que le système soit faisable est $U \leq 1$.

EDF préemptif

Pour un ensemble de tâches synchrones tel que $\forall i$, $D_i = T_i$, la condition nécessaire et suffisante de faisabilité [Liu73] est :

$$U = \sum_{i=1}^{N} \frac{C_i}{T_i} \leq 1 \tag{2.1}$$

Dans [Coffman76], il est prouvé que cette condition de faisabilité est également valable pour des tâches asynchrones.

La condition reste nécessaire et suffisante dans le cas où $\forall i$, $D_i \geq T_i$ [Baruah90]. Pour un ensemble de tâches tel que $\forall i$, $D_i \leq T_i$, la condition précédente n'est bien sûr plus suffisante. Dans [Baruah90, Spuri96], deux conditions nécessaires et suffisantes pour la faisabilité d'un ensemble de tâches avec D_i et T_i arbitraires sont données. Dans [Baruah90], il s'agit du déroulement de l'ordonnancement par l'intermédiaire de la demande en capacité processeur $h(t)$, définie par :

$$t \in \mathbb{R}^+, h(t) = \sum_{i=1}^{N} C_i \cdot \max\left[0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right] = \sum_{D_i \leq t} C_i \cdot \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right)$$

La condition de faisabilité nécessaire et suffisante pour le systéme de tâches est alors :

$$U \leq 1 \, \forall t \in \mathbb{R}^+, h\left(t\right) \leq t$$

Vérifier cette condition pour toutes les valeurs de t est infaisable en pratique. Dans [Baruah90, George96], on indique comment restreindre correctement le domaine de variation de t (une fraction discrète de l'hyperpériode), mais la complexité reste élevée (bien que la taille du domaine de variation de t soit pseudo-polynomiale dans la majorité des cas).

Dans [Spuri96], il s'agit du calcul du temps de réponse dans le pire cas ptr pour chacune des tâches, reposant sur le calcul de l'interférence qu'une tâche peut subir suite à l'activation de tâches plus prioritaires. Une fois ce temps déterminé, il suffit de vérifier que $\forall i$, $ptr \leq D_i$. Pourtant, la complexité de ce calcul reste élevée.

En pratique, dans le cas général (T_i et D_i arbitraires) on utilise souvent la condition suffisante plus simple suivante :

$$\sum_{i=1}^{N} \frac{C_i}{\min(D_i, T_i)} \leq 1$$

EDF non préemptif

Dans le cas de tâches non-concrètes, une condition nécessaire et suffisante pseudo-polynomiale est donnée dans [Jeffay91] et repose sur la demande en ressource processeur.

Dans le cas des tâches synchrones, la même condition devient suffisante seulement, et il est prouvé dans [Jeffay91, George95] que le problème qui consiste à déterminer l'ordonnançabilité du système est NP-complet (des conditions de faisabilité dans le cas synchrone et non synchrone sont données dans [George95]).

Ordonnancement préemptif à priorité statique

Dans ce paragraphe et le suivant, les tâches τ_i sont classées par ordre de priorité décroissantes : $i < j \Rightarrow$ la priorité de τ_i est plus haute que celle de τ_j (0 étant la plus faible priorité possible) ; dans le cas d'une affectation des priorités suivant RM, la priorité de τ_i est $\frac{1}{\min(D_i,T_i)}$.

Dans [Liu73], l'auteur considère que les priorités des tâches sont affectées suivant RM, et la condition suffisante suivante sur la faisabilité d'un ensemble de tâches synchrones tel que $\forall i$, $D_i = T_i$ est donnée :

$$\sum_{i=1}^{N} \frac{C_i}{\min(D_i, T_i)} \leq N \cdot \left(2^{\frac{1}{N}} - 1\right) \tag{2.2}$$

Dans [Liu73], il est également montré que le pire temps de réponse est obtenu pour la première instance des tâches lorsque les tâches sont synchrones. Il en découle que la condition 2.2 reste une condition de faisabilité suffisante pour des ensembles de tâches non-concrètes.

Une condition nécessaire et suffisante pour les tâches non concrètes est donnée dans [Tin92a, Bur94, Joseph86]. Elle repose sur le calcul du temps de réponse dans le pire cas ptr_i des tâches, s'intitule RTA (pour Response Time Analysis), et s'écrit $\forall i$, $ptr_i \leq D_i$. Le calcul de ptr_i ne suppose pas d'affectation des priorités particulières, et peut se calculer sans hypothèse sur la relation entre les T_i et les D_i.

Dans le cas où $\forall i$, $D_i \leq T_i$ [Joseph86], ptr_i se calcule en prenant en compte les interférences venant des tâches de priorité supérieure : c'est un calcul itératif qui vise à élargir un intervalle de temps jusqu'à ce que celui-ci contienne à la fois le temps d'exécution de τ_i, et les interférences par les tâches de priorité supérieure (i.e. pour une fenêtre de taille w, cette interférence s'écrit $\sum_{j<i} C_j \left\lceil \frac{w}{T_i} \right\rceil$). Lorsque $U \leq 1$ et $\forall i$, $D_i \leq T_i$, ptr_i est ainsi le point fixe de la suite :

$$ptr_i^0 = C_i$$

$$\forall k \geq 1, ptr_i^{(k)} = C_i + \sum_{j<i} C_j \cdot \left\lceil \frac{ptr_i^{(k-1)}}{T_j} \right\rceil$$

Dans le cas général [Tin92a, Bur94] (D_i et T_i non reliés), il faut en plus prendre en compte les interférences de τ_i avec elle-même quand $D_i > T_i$ (i.e. dans ce cas, plusieurs instances de τ_i peuvent être prêtes à s'exécuter en même temps).

Lorsque les tâches sont concrètes, ces conditions d'ordonnançabilité deviennent suffisantes seulement. En effet, lorsque les dates d'activation des premières instances des tâches font que les tâches ne sont jamais activées simultanément, le temps de réponse pire cas qui se manifeste effectivement peut être strictement plus petit que celui calculé dans les formules ci-dessus.

Notons que les résultats de [Liu73] est transposable au cas de l'ordonnancement non préemptif des tâches, et dans le cas général (D_i et T_i non reliés) de tâches non concrètes, [George96] donne une condition nécessaire et suffisante d'ordonnançabilité d'un ensemble de tâches, analogue à RTA.

2.2 Etat de l'art sur les travaux autour du modèle de la contrainte (*m,k*)-firm

La technique présentée dans ce document pour résoudre le problème de surcharge de la ressource partagée est basé sur le modèle (m,k)-firm initialement proposé par Hamdaoui et Ramanathan. Dans cette partie, nous montrons l'intérêt du modèle (m,k)-firm et présentons un état de l'art sur les travaux réalisés autour de ce modèle depuis sa proposition.

2.2.1 Les caractéristiques du modèle de la contrainte (*m,k*)-firm

Dans le domaine d'ordonnancement, les systèmes temps-réel sont traditionnellement classées en deux catégories : (1) système temps-réel dur, (2) système temps-réel souple.

Application temps réel dure : Les applications ayant des contraintes temporelles strictes ne tolèrent aucune violation de la contrainte temporelle spécifiée sous la forme d'échéances temporelles à respecter. En conséquence, la violation de cette contrainte d'une tâche ou d'un message pourrait conduire le système à un état d'échec. Dans la pratique, plusieurs systèmes pourraient être considérés comme des systèmes temps-réel durs. Par exemple, l'exemple d'une application steer-by-wire [Wilwert03] qui consiste à envoyer des signaux du volant vers les roues du véhicule. Cette application exige que la commande du conducteur parvienne aux actionneurs des roues avant une échéance critique, au-delà de laquelle le véhicule risque de devenir incontrôlable. Ainsi il est impératif que le temps de réponse de la commande soit borné et que cette borne soit respectée. Les résultats présentés dans le paragraphe précédent sont tous basés sur les contraintes temporelles strictes (ou dures).

Application temps réel souple : Il existe aussi de nombreuses applications temps-réel où le respect de l'échéance temporelle de chaque instance n'est pas aussi strict pour assurer le

fonctionnement correct du système. Pour ce genre d'applications, il suffit que la plupart des instances de tâches ou messages terminent avant leurs échéances tant qu'un certain nombre d'instances ou message ne ratent pas leurs échéances. Ce genre d'applications tolérant dans une certaine mesure la violation de la contrainte temporelle sont désigné sous le terme de systèmes temps réel souples

Beaucoup de systèmes de contrôle-commande tolèrent la violation occasionnelle de la contrainte temporelle. Pourtant, la notion du système temps-réel souple ne convient pas à la plupart de ces systèmes dans lesquels la façon des pertes d'instances ou messages nécessitent d'être explicitement spécifiés. Prenons l'exemple du contrôle d'un pendule inversé : la tâche calcule périodiquement la prochaine action à effectuer pour maintenir le pendule dans la position d'équilibre stable (un angle de 180°). Rater quelques échantillons de cette tâche peut être toléré (le pendule reste en position d'équilibre, mais avec plus de vibrations) mais si plusieurs instances consécutives de cette tâche ratent leurs échéances, le pendule va tomber.

Le modèle (m,k)-firm : Le modèle de (m,k)-firm a été proposée pour la première fois par Hamdaoui et Ramanathan dans [Hamdaoui94] pour caractériser d'une manière précise le niveau de la garantie temporelle offert à des applications temps-réel tolérant la perte de certaines instances de tâches ou certains messages. Le modèle (m,k)-firm est caractérisé par deux paramètres m et k. Plus formellement, une application est dite sous contrainte temporelle (m,k)-firm [Hamdaoui94] si au moins m instances ou messages parmi k instances ou messages consécutifs doivent impérativement respecter leurs échéances. Ceci dit que si la contrainte temporelle (m,k)-firm d'une application est respectée, alors dans n'importe quel intervalle de k instances ou messages, il existe au moins m instances (ou messages) qui respectent leur échéances.

Notons que le modèle (m,k)-firm est flexible pour décrire la contrainte temporelle d'une application temps-réel : le cas $m = k$ représente la contrainte temporelle stricte ; la contrainte (m,k)-firm avec $m < k$ pourrait être utilisée pour représenter la contrainte temporelle souple avec un taux de perte maximum autorisé égal à $\frac{k-m}{k}$. Ainsi, la contrainte (m,k)-firm se trouve dans un niveau intermédiaire entre la garantie déterministe et la garantie souple. Par ailleur, le modèle (m,k)-firm permet de spécifier explicitement le motif (ou pattern) de rejet (ou non respect des échéances). Grâce aux caractéristiques du modèle (m,k)-firm, ce dernier est approprié pour modéliser la façon dont le système de contrôle-commande ne respecte pas sa contrainte temporelle.

2.2.2 Les algorithmes d'ordonnancement basé sur le modèle (m,k)-firm et les travaux existants

L'avantage de la spécification explicite du modèle de rejets selon le modèle (m,k)-firm a attiré une attention particulière dans le domaine de l'ordonnancement temps réel, et un certain nombre de résultats ont été obtenus :

– la proposition de nouveaux algorithmes d'ordonnancement temps-réel tenant compte de

la contrainte (m,k)-firm, tels que l'algorithme DBP (Distance Based Priority) proposé dans [Hamdaoui95] et sa version multi-sauts appelé DBP-M (DBP in Multi hop) proposée dans [Lindsay99]. Une autre extension est présentée dans [Poggi03] pour améliorer l'ordonnancement d'un ensemble de tâches périodiques avec DBP. Un algorithme, dans le même esprit que DBP, appelé DWCS (Dynamic Window Constrained-Scheduling), a été proposé dans [West99] dans le contexte particulier des applications multimédias. D'autres algorithmes tels que MDA (Markov chain Driven Algorithm)¸DDA (Dropout-rate Driven Algorithm) et FDA (Feedback Driven Algorithm) sont proposés dans [Liu03] spécifiquement pour des systèmes de contrôle-commande en boucle fermée.

– l'amélioration des algorithmes d'ordonnancement existants pour les adapter aux contraintes (m,k)-firm tels que la proposition de [Quan00] qui améliore l'ordonnancement à priorité fixe, et la proposition de [Ramanathan99] qui adapte Rate Monotonic aux contraintes (m,k)-firm. Ces deux propositions classent les instances d'une tâche en deux parties : optionnelle ou critique selon leurs contraintes (m,k)-firm et essaient de fournir la garantie déterministe aux instances critiques d'une tâche.

Nous présentons dans cette section l'ensemble des algorithmes proposés dans la littérature pour l'ordonnancement des tâches sous contraintes (m,k)-firm ainsi que les travaux déjà réalisés autour ces algorithmes.

Outre les paramètres de la tâche présentés dans la section 2.1.3, deux nouveaux paramètres seront présents dans la suite pour caractériser la contrainte (m,k)-firm attribuée à la tâche :

– m_i et k_i : contrainte (m,k)-firm de τ_i avec $m_i \leq k_i$;
– Le (m,k)-pattern Π_i : la chaîne de caractères $\Pi_i = \Pi_i(1)\,\Pi_i(2)\ldots\Pi_i(n)$ telle que la $j^{ième}$ instance de τ_i doit être exécutée par le processeur si $\Pi_i(j-1) = 1$, ou peut être ignorée si $\Pi_i(j-1) = 0$, et que $\sum_{j=l}^{l+k_i-1} \Pi_i(j) = m_i$ pour $l \in [1..n]$.

Il existe principalement deux familles d'algorithmes qui prennent en compte les contraintes (m,k)-firm : (1) dynamique (exemple : DBP (*Distance Based Priority*)), (2) statique (exemple : EFP ou RM). Dans un algorithme d'ordonnancement dynamique la priorité affectée à chaque instance (ou message) est ajustée dynamiquement en fonction de l'état courrant du système. Tandis que dans l'algorithme d'ordonnancement statique, l'affectation de priorité est déterminée en fonction d'un paramètre fixe (le taux $\frac{m}{k}$ par exemple).

Les algorithmes d'ordonnancement dynamique permettent une adaptation dynamique au changement de charge du système. Les algorithmes d'ordonnancement statique sont moins flexibles mais permettent une vérification hors-ligne du système. Dans ce qui suit nous présentons deux algorithmes représentatifs des algorithmes d'ordonnancement dynamiques et statiques proposés dans la littérature.

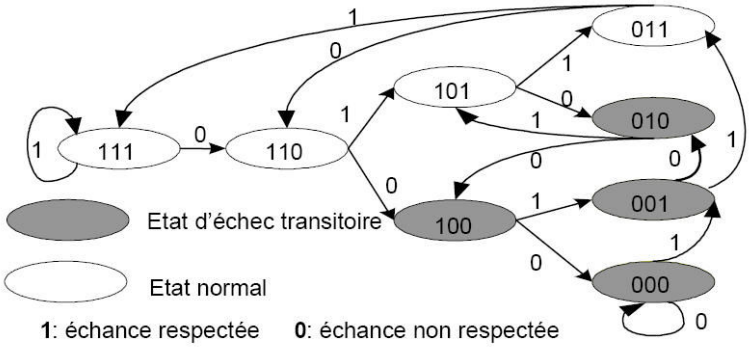

FIGURE 2.2 – Diagramme d'état-transition d'une source avec (2,3)-firm

2.2.2.1 Distance Based Priority (DBP)[Hamdaoui94, Hamdaoui95, Hamdaoui97]

[Hamdaoui94, Hamdaoui95] traitent le problème des messages dans des systèmes de communication (réseaux embarqués, systèmes de contrôle-commande, réseaux à commutation de paquets) sous contraintes (m,k)-firm.

Les auteurs définissent la notion de l'échec dynamique d'un système : une application sous contrainte (m,k)-firm est dite en état d'échec dynamique à un instant t, si à cet instant il existe plus de (k-m) messages ayant raté leurs échéances parmi les k derniers messages. Ainsi, une application sous contraintes (m,k)-firm peut se trouver dans l'un des deux états : normal ou échec dynamique. La connaissance de son état à l'instant t dépend de l'historique du traitement des k derniers instances ou messages. Si on associe '1' à une instance (ou message) respectant son échéance et '0' à un message ratant son échéance, cet historique est entièrement décrit par une suite des '0' et des '1' de longueur k bits appelée un (m,k)-pattern. La figure 2.2 donne un exemple de diagramme d'état-transition de (m,k)-pattern d'une source sous contrainte (2,3)-firm. Par convention, le déplacement des bits dans un (m,k)-pattern se fait de la droite vers la gauche.

L'algorithme d'ordonnancement dynamique sous contraintes (m,k)-firm, appelé DBP (Distance Based Priority) est proposé pour le but de réduire la probabilité de l'échec dynamique. Comme son nom l'indique, DBP définit la notion de distance de l'état actuel du flux, à l'instant t, à un état d'échec dynamique (cf. la figure 2.2). Pour un (m,k)-pattern en état normal à l'instant t, cette distance est définie par le nombre consécutif de bits '0' qu'on doit rajouter au (m,k)-pattern pour être en échec dynamique.

De ce fait, la priorité affectée par DBP aux différents flux est inversement proportionnelle à la distance séparant l'état actuel de l'échec dynamique. Plus la distance est petite, plus la priorité est élevée. Si un flux se trouve déjà en état d'échec dynamique, c'est-à-dire il existe moins de m échéances respectées dans le (m,k)-pattern, la plus haute priorité '0' est affectée au flux.

Par exemple, pour une source sous contrainte (3,5)-firm, si à l'instant $t0$ le (m,k)-pattern du flux est égal à 11011 alors la priorité du flux à cet instant est positionnée à 2 puisque après

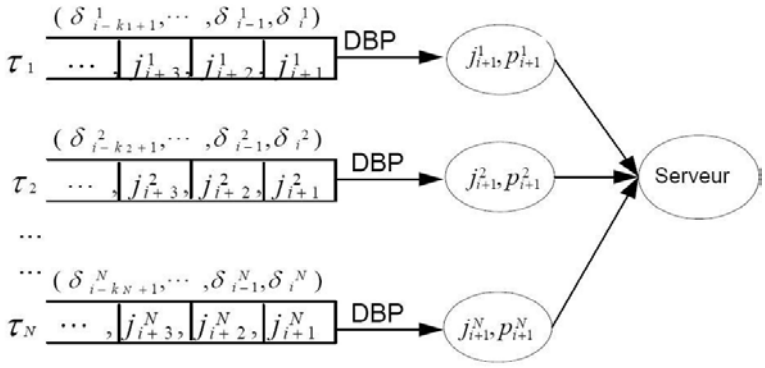

FIGURE 2.3 – Ordonnancement et affectation de priorités avec DBP ; j_i^x : i^{ime} instance de la tâche x ; p_i^x : priorité de i^{ime} instance de la tâche x

deux échéances ratées, le flux se trouverait en état d'échec dynamique (son (m,k)-pattern serait 01100) et donc la contrainte (3,5)-firm ne serait plus respectée. si à l'instant $t1$ le (m,k)-pattern du flux est égal à 01100 (le flux est en état d'échec dynamique) alors la priorité la plus élevée 0 est attribuée au flux.

Après chaque transmission de message, l'algorithme DBP met à jour le (m,k)-pattern du flux ainsi que sa priorité en ajoutant un '1' à droite si la contrainte temporelle est respectée, sinon il ajoute un '0'.

Formellement, l'affectation de priorité du message se fait comme suit :

La priorité du $(i+1)^{ieme}$ message de τ_j est donnée par :

$$DBP_{i+1}^{j} = k_j - l_j(m_j, s) \tag{2.3}$$

où

- $s_i \left(\delta_{i-k_j+1}^{j}, ...\delta_{i-1}^{j}, \delta_i^j\right)$ le (m,k)-pattern de la source τ_i où $\delta_i^j \in \{0,1\}$ représente l'état de transmission du i^{ime} message.
- $l_j\,(n,s)$ la position (en comptant à partir de la droite) de la n^{ime} échéance respectée dans s.

Notons que lorsqu'il y a moins de m_j symboles '1' dans s_j, alors $l_j\,(m_j, s_j) = k_j + 1$ afin que la plus haute priorité (zéro) soit affectée.

La figure 2.3 montre le schéma d'affectation de priorité par l'algorithme DBP. Cette politique d'affectation dynamique de priorité peut être facilement et efficacement implémentée en matériel car l'historique de chaque source peut être stocké dans un registre de k_j bits.

En cas d'égalité des priorités de deux ou plusieurs flux, la politique d'ordonnancement EDF est utilisée pour choisir le message à transmettre.

Après la proposition de l'algorithme DBP, un certain nombre de travaux ont été menés pour mesurer et améliorer la performance de l'algorithme [Hamdaoui95, Hamdaoui97, Li03]. Nous donnons dans la suite une courte description de chacun de ces travaux.

Dans [Hamdaoui95], les auteurs ont évalué par simulation le comportement de DBP+EDF pour deux modèles de flux : l'un selon la loi de Poisson et l'autre un trafic en rafale modélisé par une source ON/OFF où ON représente une période de temps exponentiellement distribuée dans laquelle la source génère des messages périodiquement. L'état OFF désigne la période de silence qui est de même exponentiellement distribuée. Les résultats des différents scénarios montrent une meilleure performance de DBP comparé à FIFO en terme de réduction de la probabilité d'échec dynamique.

Dans [Hamdaoui97], un modèle analytique utilisant le formalisme probabiliste et les chaînes de Markov est développé pour le calcul de la probabilité d'échec dynamique d'un flux temps-réel pour l'ordonnancement DBP et FIFO. Ce modèle permet de fournir une garantie probabiliste de qualité de service aux flux temps-réel. Le modèle analytique est validé par une étude par simulation et montre l'avantage de DBP par rapport à FIFO pour la réduction de la probabilité de l'échec dynamique.

Dans [Li03], une analyse d'ordonnançabilité a été effectuée pour déterminer la condition d'ordonnançabilité suffisante pour un ensemble de messages périodiques/sporadiques, sous contraintes (m,k)-firm et ordonnancées avec l'algorithme DBP+EDF non préemptif.

A part l'algorithme DBP, un autre algorithme d'ordonnancement dynamique appelé DWCS (Dynamic Window Constrained Scheduling) peut être trouvé dans [West99]. L'objectif de cet algorithme est de maximiser l'utilisation de la bande passante du réseau en cas de surcharge. Les résultats de simulation montrent que DWCS et DBP ont des performances similaires en termes du nombre d'échéances ratées et la violation de la contrainte de fenêtre.

2.2.2.2 Enhanced Fixed Priority (EFP) [Ramanathan99, Quan00]

Dans [Ramanathan99], l'auteur considère une application de contrôle-commande temps réel qui comprend des boucles de contrôle. Les contrôleurs sont implémentés par des tâches préemptives qui s'exécutent sur deux processeurs. En cas d'échec d'un processeur, l'exécution des tâches est assurée par l'autre processeur. Pour éviter la surcharge du processeur, chaque tâche est supposée sous une contrainte (m,k)-firm et un algorithme d'ordonnancement statique est proposé pour ordonnancer cet ensemble de tâches.

Selon cet algorithme, les instances de chaque tâche sont classées en deux catégories : instance critique et instance optionnelle. Toutes les instances critiques sont ordonnancées selon leur priorité fixe, alors qu'aux les instances optionnelles est assignée la priorité la plus basse. Il est montré que si toutes les instances critiques finissent leur exécution avant l'échéance, la contrainte (m,k)-firm est satisfaite.

La classification des instances d'une tâche τ_i selon sa contrainte (m_i, k_i)-firm est donnée par la règle suivante :

La a^{ime}instance de la tâche avec $(a = 0, 1, ...)$ est marquée critique (i.e. $\Pi_i\,(a-1) = 1$) si a vérifie :

$$a = \left\lfloor \left\lceil \frac{a \cdot m_i}{k_i} \right\rceil \cdot \frac{k_i}{m_i} \right\rfloor \tag{2.4}$$

Partant de l'idée qu'une classification judicieuse et globale des instances critiques de toutes les sources devrait donner une meilleure ordonnançabilité, [Quan00] a amélioré l'algorithme proposé dans [Ramanathan99]. Il a été d'abord prouvé que trouver une classification optimale des instances est NP-complet. Puis, en prenant en compte les effets de blocage et de préemption d'une tâche par une autre, un algorithme heuristique est proposé pour trouver une classification des instances offrant une meilleur ordonnançabilité par rapport à la stratégie de classification 2.4.

Dans [Jia05a], nous montrons une nouvelle méthode d'analyse de l'ordonnançabilité des ensembles de tâches sous contraintes (m,k)-firm en utilisant des propriétés des mots mécaniques (théorie des mots). Nous avons montré que les patterns introduits dans [Ramanathan99] peuvent se caractériser sous la forme de mots mécaniques [2]. Les preuves d'ordonnançabilité en sont ainsi simplifiées. En identifiant les défauts de ces patterns, nous proposons ensuite une nouvelle technique basée sur la ligne cellulaire pour déterminer les (m,k)-patterns des tâches. Les résultats expérimentaux montrent que cette nouvelle technique permet une amélioration de la région ordonnançable.

Il est prouvé dans [Koren96, Quan00] que le problème de déterminer s'il existe un (m,k)-pattern pour chaque tâche sous contrainte (m,k)-firm tel que l'ensemble des tâches sont ordonnançables, ainsi que le problème de déterminer l'ordonnançabilité d'un ensemble des tâches sous contraintes (m,k)-firm étant donné les (m,k)-pattern sont NP-difficiles que ce soit pour l'algorithme d'ordonnancement à priorités dynamiques ou pour l'algorithme d'ordonnancement à priorités statiques. Il est toutefois possible de construire des conditions d'ordonnançabilité appropriées si les (m,k)-patterns des tâches sont déterminés suivant certaines stratégies spécifiques tel que la théorie du mot mécanique [Ramanathan99, Jia05a]. A partir des résultats de l'analyse de l'ordonnançabilité dans les travaux cités dans cette section, et en considérant la façon dont les (m,k)-pattern sont déterminés, nous donnerons dans la suite deux conditions d'ordonnançabilité pour l'ordonnancement non-préemptif des tâches sous contrainte (m,k)-firm afin que nous puissions les appliquer lors qu'il s'agit d'ordonnancer des messages (qui sont non-préemptifs).

2.3 Conditions d'ordonnançabilité pour l'ordonnancement non-préemptif à priorité fixe sous contraintes (m,k)-firm

Nous considérons ici que les instances des tâches sont ordonnancées selon la politique d'ordonnancement non-préemptif à priorité fixe. Nous donnons d'abord une condition d'ordonnançabilité pour les tâches avec des (m,k)-patterns quelconques. Puis, une condition d'or-

2. Le mot mécanique et ses propriétés seront présentés dans la section 5.2.2.4

donnançabilité appropriée pour le cas où les (m,k)-patterns sont déterminés selon la théorie des mots mécaniques sera donnée.

• ***Ordonnançabilité avec (m,k)-pattern quelconque***

Dans [Quan00], l'auteur a prouvé que si un ensemble de tâches sont ordonnançables avec les (m,k)-patterns dit "deeply-red", alors cet ensemble de tâches seront ordonnançables avec les (m,k)-patterns quelconques. Le (m,k)-pattern "deeply-red" est défini comme suit :

Le (m,k)-pattern "deeply-red" d'une tâche sous contrainte (m,k)-firm satisfait :

$$\Pi(j) = \begin{cases} 1 & 1 \leq j \leq m \\ 0 & m < j \leq k \end{cases}$$

Par abus de langage, nous appelons dans la suite les instances représentées par les lettres 1 du (m,k)-pattern les instances critiques.

La condition d'ordonnançabilité avec le (m,k)-pattern "deeply-red" est donnée par le théorème suivant.

Théorème 1 *Soit un ensemble de tâches $\tau_1,\tau_2..\tau_n$ tel que l'instance de τ_i est plus prioritaire que celle de τ_j si $i < j$. Les instances de chaque tâche sont rejetées selon le (m_i,k_i)-pattern "deeply-red". La borne supérieure du temps de réponse R_l des instances de τ_l est la limite, quand q tend vers l'infini, de la suite :*

$$\begin{aligned} R_l^0 &= C_l \\ R_l^q &= \max_{j>l}(C_j) + \sum_{j<l} \left(\left\lfloor \frac{\left\lceil \frac{R_l^{q-1}}{T_j} \right\rceil}{k_j} \right\rfloor + \sum_{n=0}^{H} \Pi(n) \right) C_j \end{aligned} \tag{2.5}$$

où $H = \left\lceil \frac{R_l^{q-1}}{T_j} \right\rceil - \left(\left\lfloor \frac{\left\lceil \frac{R_l^{q-1}}{T_j} \right\rceil}{k_j} \right\rfloor \right)$ *; R_l est calculé en partant de $R_l^0 = 0$ jusqu'à la convergence ou jusqu'à ce que $R_l^q > D_l - C_l$. Si $R_l \leq D_l$ pour tout $1 \leq l \leq n$, alors la contrainte (m,k)-firm de chaque tâche est respectée.*

Démonstration : Supposons que toutes les tâches se mettent à s'exécuter à l'instant 0 (c'est-à-dire que les tâches sont synchrones). Puisque les instances sont sélectivement rejetés selon le (m,k)-pattern "deeply-red", le nombre des instances critiques de τ_l dans l'intervalle de temps $[0,b]$ pour $b \in \mathbb{R}$ est ainsi maximal par rapport à tout autre intervalle de temps de même la longueur. La première instance de τ_l subit ainsi la plus grande interférence venant des instances plus prioritaires. Noter qu'il existe au plus une instance moins prioritaire qui a commencé à être exécutée sans l'avoir finie avant la date de démarrage d'une instance plus

prioritaire. Ainsi la plus grande interférence éventuellement subit par la première instance de chaque tâches doit inclure l'interférence de l'instance moins prioritaire ayant le temps d'exécution maximal. En conséquence, si la première instance de chaque tâche respecte son échéance en présence de la plus grande interférence, toutes les autres instances respecteront leur échéance.

Le terme $\left\lfloor \frac{\left\lceil \frac{R_l^{q-1}}{T_j} \right\rceil}{k_j} \right\rfloor + \sum_{n=0}^{H} \Pi(n)$ donne le nombre des instances critiques de τ_j dans l'intervalle de temps $\left[0,\ R_l^{q-1}\right]$. Le terme $\sum_{j<l} \left(\left\lfloor \frac{\left\lceil \frac{R_l^{q-1}}{T_j} \right\rceil}{k_j} \right\rfloor + \sum_{n=0}^{H} \Pi(n) \right) C_j$ donne ainsi la plus grande interférence à l'instant R_l^{q-1} venant de tous les instances plus prioritaires que la première instance de τ_l. Le temps de réponse de la première instance de τ_l est borné par la limite de l'équation récurrente 5.6 qui donne le temps d'exécution total requis par toutes les instances critiques à exécuter avant l'exécution de la première instance de τ_l. Si la borne du temps de réponse de chaque tâche ne dépasse pas son échéance, alors les contrainte (m_i, k_i)-firm seront respectée. CQFD

□

- ***Ordonnançabilité avec (m,k)-pattern défini par le mot mécanique***

Dans le cas où le (m,k)-pattern est défini en utilisant le mot mécanique, une condition d'ordonnançabilité plus stricte peut être obtenue. Dans la suite, nous présentons d'abord deux propriétés du mot mécanique qui seront utilisées pour démontrer la condition d'ordonnançabilité.

Lemma 1 *[Lothaire02] Le nombre de 1 du mot w de pente α entre la lettre d'indice 0 et celle d'indice k est égal à :*

$$\lceil (k+1)\,\alpha \rceil \qquad k = 0, 1, 2...$$

Lemma 2 *[Ramanathan99, Jia05a] Le nombre de lettres 1 d'un mot mécanique w entre les lettres $w(n)$ et $w(n+b)$, $b \geq 0$ est maximum pour $n = 0$ et b quelconque.*

La condition d'ordonnançabilité est donnée par le théorème suivant :

Théorème 2 *Soit un ensemble de sources de messages (ou un ensemble de tâches non-préemptives) τ_1,τ_2..τ_n tel que l'instance de τ_i est plus prioritaire que celle de τ_j si $i < j$. Les instances de chaque tâche sont rejetés selon le (m_i, k_i)-pattern calculé suivant la méthode décrite dans la section 5.2.2.4. La borne supérieure du temps de réponse des instances critiques de τ_l est la limite, quand q tend vers l'infini, de la suite :*

$$\begin{aligned} R_l^0 &= C_l \\ R_l^q &= \max_{j>l}(C_j) + \sum_{j<l} \left\lceil \frac{m_j}{k_j} \left\lceil \frac{R_l^{q-1}}{T_j} \right\rceil \right\rceil C_j \end{aligned} \tag{2.6}$$

R_l est calculé en partant de $R_l^0 = 0$ jusqu'à la convergence ou jusqu'à ce que $R_l^q > D_l - C_l$ où D_l est l'échéance du message de la source τ_i. Si $R_l \leq D_l$ pour tout $1 \leq i \leq n$, alors la contrainte (m,k)-firm des messages provenant de chaque source τ_i est respectée.

Démonstration : Supposons que toutes les tâches se mettent à s'exécuter à l'instant 0 (c'est-à-dire que les tâches sont synchrones). Par lemme 2, le nombre des instances critiques de la source τ_l dans l'intervalle de temps $[0, b]$ pour $b \in \mathbb{R}$ est maximal par rapport à tout autre intervalle de temps de même longueur. La première instance de τ_l subit ainsi la plus grande interférence venant des instances plus prioritaires. Notons qu'il existe au plus une instances moins prioritaire qui a commencé à être exécutée sans avoir fini avant la date de démarrage d'une instance plus prioritaire, ainsi la plus grande interférence éventuellement subit par la première instance de chaque tâche doit inclure l'interférence de l'instance moins prioritaire ayant le temps de transmission maximal. En conséquence, si la première instance de chaque tâche respecte son échéance en présence de la plus grande interférence, toutes les autres instances respecteront leur échéance.

Par le lemme 1, le terme $\left\lceil \frac{m_j}{k_j} \left\lceil \frac{R_l^{q-1}}{T_j} \right\rceil \right\rceil$ donne le nombre des instances critiques de τ_j dans l'intervalle de temps $\left[0,\ R_l^{q-1}\right]$. Le terme $\sum_{j<l} \left\lceil \frac{m_j}{k_j} \left\lceil \frac{R_l^{q-1}}{T_j} \right\rceil \right\rceil C_j$ donne ainsi la plus grande interférence à l'instant R_l^{q-1} venant de toutes les instances plus prioritaires que la première instance de τ_l. Le temps de réponse de la première instance de τ_l est borné par la limite de l'équation récurrente 5.6 qui donne le temps d'exécution total requis par toutes les instances critiques à exécuter avant l'exécution de la première instance de τ_l. Si la borne du temps de réponse de chaque tâche ne dépasse pas son échéance, alors les contraintes (m_i, k_i)-firm seront respectée. CQFD

□

2.4 Conclusion

Que ce soit pour l'ordonnancement temps réel classique ou pour l'ordonnancement temps réel sous contrainte (m,k)-firm, nous venons de voir que deux familles d'algorithmes existent : dynamiques tels que EDF et DBP, et statiques tels que RM et EFP. Du point de vue de la performance d'ordonnancement, les algorithmes dynamiques sont meilleurs. Mais pour des applications de contrôle échantillonné, un algorithme dynamique tel que EDF, mis à part de son coût d'implantation plus élevé, n'est pas forcément plus intéressant qu'un algorithme

statique tel que RM. Considérons un ensemble de tâches de contrôle périodiques de périodes différentes avec échéances sur requête et partageant un même processeur. Si la priorité statique permet de favoriser les tâches les plus importantes en leur attribuant des priorités élevées, en revanche EDF peut bouleverser cet ordre car une tâche moins importante obtiendra tôt ou tard la priorité la plus élevé en cas de surcharge. Or il aurait été peut-être plus judicieux de le rejeter et laisser la ressource le temps de traiter une tâche de contrôle d'un procédé plus important. D'ailleurs, il est connu qu'EDF s'adapte mal à la situation de surcharge (i.e. le cas du dépassement des échéances).

Dans la suite nous allons considérer que l'ensemble de tâches sont ordonnancées en utilisant l'algorithme EFP et la conception d'ordonnancement se réduit à adapter dynamiquement les paramètres de la contrainte (m,k)-firm en fonction du besoin applicatif.

Chapitre 3

Etude d'un système monodimensionnel : rejet contrôlé d'échantillons et optimisation de performances

Nous étudions dans ce chapitre la conception d'un système de contrôle-commande monodimensionnel distribué pour lequel la transmission des échantillons de sortie du système contrôlé s'effectue en utilisant un réseau partagé avec d'autres applications. Plus particulièrement, nous présentons une méthode de conception conjointe de la loi de commande (le contrôleur) et des règles de contrôle des transmissions d'échantillons entre capteur et contrôleur qui minimisent la bande passante requise par l'application, garantissent la stabilité du système et fournissent une valeur optimisée d'un critère de performance du contrôle. La solution envisagée est de rejeter sélectivement des messages selon une stratégie qui repose sur une contrainte de type (m,k)-firm (au moins m messages parmi k messages consécutifs doivent être reçus par le contrôleur). L'emploi de cette stratégie dans un système de contrôle-commande nécessite de répondre à deux questions : comment déterminer la contrainte (m,k)-firm et comment concevoir le contrôleur pour que le système fonctionne correctement (stabilité) et de manière optimisée ? Ce chapitre fournit des réponses à ces questions dans le cas où le système contrôlé est monodimensionnel.

Dans la suite, nous donnons tout d'abord le modèle mathématique du système monodimensionnel étudié en section 3.1. Puis en section 3.2, nous établissons la condition nécessaire et suffisante de stabilité et nous montrons comment déterminer la contrainte (m,k)-firm sous laquelle le système reste stable. Ensuite en section 3.3, nous présentons une approche pour déterminer le paramètre optimal du contrôleur qui optimise un critère de performance du système, à savoir la borne supérieure de la variance de l'état du processus contrôlé. Un exemple numérique est donné en section 3.4. Enfin, nous concluons cette étude dans la section 3.5.

3.1 Modèle de système

La figure 3.1 fournit une représentation générique d'un système linéaire monodimensionnel et de son contrôleur. Le processus à contrôler est représenté ici, sous forme discrétisée, par l'équation suivante :

$$x_{i+1} = \alpha x_i + \beta u_i \quad i = 0, 1, .. \tag{3.1}$$

- u_i et x_i sont respectivement l'entrée (la commande) et la sortie du système,
- α et β sont des constantes ; $|\alpha| > 1$ et $\beta \neq 0$.

Ce système est contrôlé en temps discret et, plus particulièrement, nous considérons dans tout ce qui suit que la sortie du système (y) est envoyée périodiquement, avec une période h à destination du contrôleur. Le contrôleur élabore une consigne z en fonction de y. La sortie mesurée est donnée par $y_i = x_i + p_i$, où p_i représente un bruit. L'entrée du système (la commande) est obtenue par $u_i = z_i + q_i$, où q_i représente un bruit. Dans cette étude, les bruits p_i et q_i sont supposés être des bruits blancs d'espérance nulle, indépendants (c-à-d $E[p_i q_{i+\tau}] = 0$ pour tout i et τ), et de variances σ_p^2 et σ_q^2 respectivement.

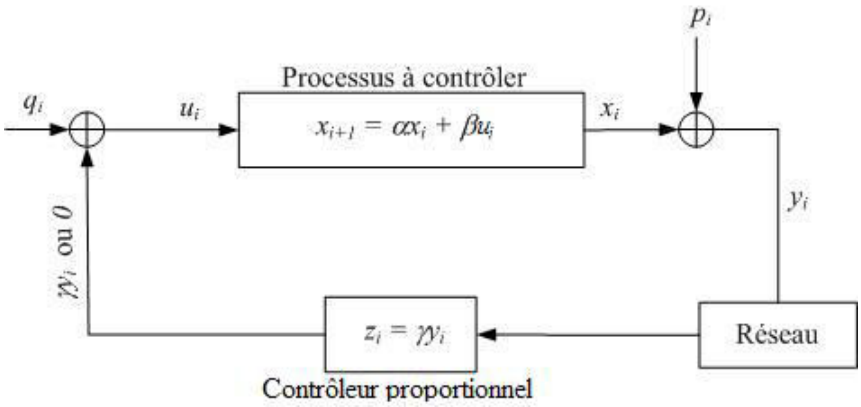

FIGURE 3.1 – Modèle du système considéré

Dans cette étude, de plus, nous considérons que le capteur et le contrôleur sont reliés via un réseau de communication. Le message contenant la mesure y_i est envoyé au contrôleur au travers du réseau. Nous supposons que le réseau est une ressource limitée et que la bande passante est partagée entre plusieurs systèmes en plus du système de contrôle-commande considéré. La solution proposée dans cette étude consiste à diminuer la bande passante réservée par le système de contrôle-commande en rejetant sélectivement des messages contenant des mesures tout en assurant le bon fonctionnement du système contrôlé.

Concrètement, le rejet de messages s'effectue suivant la contrainte (m,k)-firm. Au sens strict cela signifie que, à tout instant, au moins m messages parmi k messages consécutivement envoyés ont été reçus par le contrôleur avant leur échéance ; nous considérons ici que l'échéance relative des messages est égale à la période d'échantillonnage du processus. En fait,

pour simplifier l'analyse du système, nous considérons que exactement m messages parmi k messages consécutifs sont reçus par le contrôleur. De plus, le rejet de messages est supposé périodique, c'est-à-dire que si le message envoyé à l'instant i est reçu par le contrôleur (respectivement rejeté), le message envoyé à l'instant $i+k$ est aussi reçu par le contrôleur (respectivement rejeté). En l'occurrence, le (m,k)-pattern Π peut être défini comme un mot binaire de longueur k contenant exactement m '1'. La séquence de messages reçus par le contrôleur est ainsi représentée par ce (m,k)-pattern avec $\Pi(i) = \Pi(i+k)$ (on note par la suite $\Pi_i = \Pi(i)$). Cette stratégie de rejet périodique peut être justifiée par le fait qu'une telle séquence de messages transmis ou rejetés est appropriée à la garantie d'un comportement attendu du système ainsi qu'il est montré dans [Ramanathan99].

Nous supposons que le contrôleur est relié à l'actionneur par un canal dédié à cet échange et garantissant l'absence de perte de message. Si un message contenant y_i est disponible en entrée du contrôleur, celui-ci calcule la commande z , $z_i = \gamma y_i$ envoyé à l'actionneur, où γ est le gain ; dans le cas contraire, le contrôleur envoie la valeur $z_i = 0$ puisque la position d'équilibre du processus est $x_{\text{référence}} = 0$. Avec le bruit discret q, la commande appliquée au procédé est $u_i = z_i + q_i$.

Si un message est reçu par le contrôleur, l'équation d'état 3.1 peut être récrite :

$$\begin{aligned} x_{i+1} &= \alpha x_i + \beta\left(q_i + \gamma\left(x_i + p_i\right)\right) \\ &= \left(\alpha + \beta\gamma\right) x_i + \beta\left(q_i + \gamma p_i\right) \end{aligned} \tag{3.2}$$

Si le message est rejeté sur le réseau, le contrôleur envoie la valeur zéro, et l'évaluation de l'équation 3.1 se ramène à :

$$x_{i+1} = \alpha x_i + \beta q_i \tag{3.3}$$

Ces deux égalités se résument en :

$$x_{i+1} = \left(\alpha + \gamma\beta\Pi_i\right) x_i + \beta\left(q_i + \gamma p_i \Pi_i\right) \tag{3.4}$$

avec $i \geq 0$ et $x_0 = 0$.

Cette formule met en évidence le rôle joué par le (m,k)-pattern.

3.2 Condition de stabilité

Le problème de stabilité est un sujet de préoccupation majeure en phase de conception du système de contrôle-commande. Dans cette section, nous nous intéressons au problème de stabilité du système décrit dans la section précédente.

Il existe plusieurs critères de stabilité. Nous considérons ici qu'un système est stable si la variance de l'état du processus contrôlé est bornée et de valeur moyenne nulle pour une longue durée d'exécution du système. Dans la suite, nous donnons dans un premier temps

l'expression pour calculer la variance de l'état du processus x_i. Puis, la condition sous laquelle la variance de l'état du processus reste finie pour une longue durée d'exécution du système est donnée.

On pose :

- $v_i = \sigma_{x_i}^2$,
- $a_i = (\alpha + \beta\gamma\Pi_i)^2$,
- et $b_i = \beta^2 \left(\sigma_q^2 + \gamma^2\sigma_p^2\Pi_i\right)$.

Les deux suites $(a_n)_{n\geq 0}$ et $(b_n)_{n\geq 0}$ sont positives. $(b_n)_{n\geq 0}$ est non identiquement nulle. De plus, ces deux suites sont k-périodiques : $a_{n+k} = a_n$, $b_{n+k} = b_n$, $\forall n \geq 0$ du fait de la k-périodicité du (m,k)-pattern.

La variance de l'état du processus est donnée par le théorème suivant :

Théorème 3 *La variance v_i de l'état du système, exprimée en fonction du (m,k)-pattern, vérifie la formule de récurrence suivante, à coefficients k-périodiques :*

$v_{i+1} = a_i v_i + b_i$ *pour* $i \geq 0$, *avec la condition initiale* $v_0 = 0$.

Démonstration

La relation 3.4 $x_{i+1} = (\alpha + \gamma\beta\pi_i)\, x_i + \beta\,(q_i + \gamma p_i \Pi_i)$, $i \geq 0$, $x_0 = 0$ donne :

$E\,(x_{i+1}) = (\alpha + \gamma\beta\Pi_i)\, E(x_i)$, $i \geq 0$, $x_0 = 0$.

puisque $E\,(p_i) = 0$ et $E\,(q_i) = 0$ pour $i \geq 0$.

Il s'ensuit que $E\,(x_i) = 0$ pour $i \geq 0$ et on peut écrire la variance de x_i : $v_i = E\left(x_i^2\right)$.

En élevant 3.4 au carré et en prenant l'espérance, on a :

$v_{i+1} = a_i v_i + 2\beta(\alpha + \gamma\beta\Pi_i) E\left\{(q_i + \gamma p_i \Pi_i)\, x_i\right\} + \beta^2 E\left\{(q_i + \gamma p_i \Pi_i)^2\right\}$

L'équation 3.4 montre que x_i est une combinaison linéaire des p_j, q_j pour $0 \leq j < i$. Par conséquent, $E\left\{(q_i + \gamma p_i \Pi_i)\, x_i\right\} = 0$.

De même, en raison des propriétés des bruits blancs indépendants, on a : $E\left\{(q_i + \gamma p_i \Pi_i)^2\right\} = \left(\sigma_q^2 + \gamma^2\sigma_p^2\Pi_i\right)$.

Ce qui donne bien $v_{i+1} = a_i v_i + b_i$.

□

Théorème 4 *La solution de l'équation* $v_{i+1} = a_i v_i + b_i$, $i \geq 0$ *et* $v_0 = 0$ *s'exprime par :*

$v_n = b_{n-1} + \sum_{j=0}^{n-2} b_j a_{j+1}..a_{n-1}$ *pour* $n \geq 1$ *(avec* $\sum_0^{-1} = 0$*)*

On l'écrit par la suite brièvement : $v_n = \sum_{j=0}^{n-1} b_j a_{j+1}..a_{n-1}$ *avec* $b_j a_{j+1}..a_{n-1} = b_{n-1}$ *si* $j = n-1$

Démonstration :

Soit $w_n = \sum_{j=0}^{n-1} b_j a_{j+1}..a_{n-1}$, pour $n \geq 1$.

w_n vérifie la relation de récurrence $w_{i+1} = a_i w_i + b_i$. En effet, $a_i w_i + b_i = b_i + \sum_{j=0}^{i-1} b_j a_{j+1}..a_{i-1}a_i = w_{i+1}$.

De plus, $w_1 = b_0 = v_1$, donc par unicité de la solution de l'équation de récurrence, on a $v_n = w_n$ ce qui termine la démonstration.

□

En remplaçant n par $n+k$ dans l'égalité du théorème 4, on écrit :

$$\begin{aligned} v_{n+k} &= \sum_{j=0}^{n-1} b_j a_{j+1}..a_{n+k-1} + \sum_{j=n}^{n+k-1} b_j a_{j+1}..a_{n+k-1} \\ &= a_n a_{n+1}..a_{n+k-1} v_n + \sum_{j=n}^{n+k-1} b_j a_{j+1}..a_{n+k-1} \end{aligned}$$

La suite (a_n) est k-périodique, ce qui donne :

$$a_n a_{n+1}..a_{n+k-1} = a_0 a_1..a_{k-1} = (\alpha+\beta\gamma)^{2m}\,\alpha^{2(k-m)} = A$$

Soit $B_n = \sum_{j=n}^{n+k-1} b_j a_{j+1}..a_{n+k-1}$, B_n est k-périodique. En effet :

$$B_{n+k} = \sum_{j=n+k}^{n+2k-1} b_j a_{j+1}..a_{n+2k-1}$$

En posant, $j = l+k$, on a :

$$\begin{aligned} B_{n+k} &= \sum_{l=n}^{n+k-1} b_{l+k} a_{l+k+1}..a_{n+2k-1} \\ &= \sum_{l=n}^{n+k-1} b_l a_{l+1}..a_{n+k-1} \\ &= B_n \end{aligned}$$

On a obtenu finalement : $v_{n+k} = Av_n + B_n$, pour $n \geq 0$.
On divise n par k : $n = kq+r$ avec $0 \leq r < k$. On a alors :

$$v_{(q+1)k+r} = Av_{qk+r} + B_r \text{ pour } 0 \leq r < k.$$

Ainsi, les suites $(v_{kq+r})_{q\geq 0}$vérifient une récurrence linéaire affine à *coefficients constants* A et B_r.

Théorème 5 *Si $A \geq 1$, la suite (v_n) n'est pas majorée.*
Si $A < 1$, on a :

$$sup_{n\geq 0} v_n = \frac{B}{1-A},\ où\ B = max_{0\leq i<k}(B_i)\ et\ B_i = \sum_{j=i}^{i+k-1} b_j a_{j+1} \cdots a_{i+k-1}.$$

Démonstration

1. Si $A \geq 1$, la suite (v_n) n'est pas majorée.

La suite (v_n) est positive. La relation $v_{(q+1)k+r} = Av_{qk+r} + B_r$ donne :

$$v_{qk+r} \geq v_r + qB_r,\ q \geq 0 \text{ et } 0 \leq r < k.$$

On note que $B_r \geq b_{r+k-1} = b_{r-1}$. Donc$v_{qk+r} \geq qb_{r-1}$.

Comme β est non nul et σ_q^2 est strictement positif, on a $b_{r-1} > 0$, ce qui entraîne :

$\lim_{q\to+\infty} v_{qk+r} = +\infty$ et la suite (v_n) n'est pas majorée.

2. Si $A < 1$

La relation $v_{(q+1)k+r} = Av_{qk+r} + B_r$ pour $0 \leq r < k$ fournit facilement :

$$v_{kq+r} = A^q v_r + \left(\frac{1-A^q}{1-A}\right) B_r = \frac{B_r}{1-A} + A^q \left(v_r - \frac{B_r}{1-A}\right)$$

$A < 1$, on a $\lim_{q\to+\infty} v_{qk+r} = \frac{B_r}{1-A}$.

De plus, la suite $(v_{qk+r})_{q\geq 0}$ est croissante. En effet :

$B_r = \sum_{j=r}^{r+k-1} b_j a_{j+1} \cdots a_{r+k-1} \geq \sum_{j=k}^{r+k-1} b_j a_{j+1} \cdots a_{r+k-1} = \sum_{j=0}^{r-1} b_j a_{j+1} \cdots a_{r-1} = v_r$ pour $r \geq 1$ d'après le théorème 4 et $B_0 \geq v_0 = 0$. On a donc toujours $v_r \leq B_r$ pour $0 \leq r < k$.

L'égalité $v_{kq+r} = \frac{B_r}{1-A} + A^q \left(v_r - \frac{B_r}{1-A}\right)$ montre que la suite $(v_{qk+r})_{q\geq 0}$ est croissante. Donc :

$$\sup_{q\geq 0} v_{kq+r} = \lim_{q\to+\infty} v_{qk+r} = \frac{B_r}{1-A}.$$

Ceci donne finalement : $\sup_{n\geq 0} v_n = \frac{1}{1-A} \max_{0\leq r<k} B_r$.

□

3.3 Le gain optimal pour un (m, k)-pattern Π donné

Nous traitons dans cette section un problème d'optimisation à savoir trouver le gain γ qui optimise la performance du système. Dans ce chapitre, nous considérons que la performance du système est donnée par $sup_{n\geq 0} v_n = \frac{B}{1-A}$, où $B = max_{0\leq i<k}(B_i)$ et $B_i = \sum_{j=i}^{i+k-1} b_j a_{j+1} \cdots a_{i+k-1}$ (théorème 5) lorsque le système est stable c'est-à-dire :

$$0 \leq A < 1$$

On peut remarquer que, comme $A = (\alpha + \beta\gamma)^{2m} \alpha^{2(k-m)}$, A dépend :

– du gain γ du contrôleur,

– des valeurs m et k de la contrainte (m,k)-firm.

Algorithm 1 Calcul de la borne supérieure de la variance du système pour un gain γ, une contrainte (m,k) et un (m,k)-pattern donnés

```
/*données :                                                    */
/*  caractéristiques du système alpha, beta                    */
/*  variances des bruits sur l'entrée et la sortie : s2p, s2q, */
/*  gain du contrôleur : gamma                                 */
/*  contraintes (m,k) et (m,k)-pattern : m,k,PI                */
/*                                                             */
for J in 0..k-1 loop
 a(J) :=(alpha+beta*gamma*PI(J)*(alpha+beta*gamma*PI(J)) ;
 b(J) :=beta*beta*(s2q+gamma*gamma*s2p*PI(J)) ;
end loop
A :=exp(2.0*m*Log(|alpha+beta*gamma|)+2.0*(k-m)*Log(|alpha|)) ;
pa :=1 ;
B(0) :=b(k-1)*pa ;
for I in k-2..0 loop
 pa :=pa*a(I+1) ;
 B(0) :=B(0)+b(I)*pa ;
end loop
Bmax :=B(0) ;
for J in 0..k-2 loop
 B(J+1) :=a(J)*B(J)+b(J)*(1-A) ;
 if B(J+1)>Bmax then
   Bmax :=B(J+1)
 end if
end loop
BorneSupVariance :=Bmax/(1-A) ;
```

La condition de stabilité ci-dessus est équivalente à :

$$\gamma_{inf} = -\frac{\alpha}{\beta} - \frac{|\alpha|^{1-\frac{k}{m}}}{|\beta|} < \gamma < -\frac{\alpha}{\beta} + \frac{|\alpha|^{1-\frac{k}{m}}}{|\beta|} = \gamma_{sup}$$

La stabilité ne dépend pas de la forme du (m,k)-pattern, Π. Les coefficients B_i, eux, dépendent de γ, des valeurs m et k, et de la forme de Π. Nous noterons par la suite : $M(\gamma, \Pi) = sup_{n \geq 0} v_n$. Le problème revient alors à déterminer la valeur de γ qui minimise $M(\gamma, \Pi)$ pour une contrainte (m,k)-firm et un (m,k)-pattern, Π donnés. Pour ceci, nous calculons $M(\gamma, \Pi)$ pour γ décrivant l'intervalle $]\gamma_{inf}, \gamma_{sup}[$ avec un pas donné.

Le calcul de $M(\gamma, \Pi)$ est réalisé par l'algorithme 1.

3.4 Exemple numérique

L'objectif de cette section est de fournir des techniques pour spécifier conjointement une contrainte (m,k)-firm et le gain du contrôleur. Deux systèmes sont étudés : un système instable par nature et un système stable.

3.4.1 Système instable

3.4.1.1 Spécification de l'étude de cas

(1) Le système contrôlé est défini par les caractéristiques suivantes :

- $\alpha = 3$, $\beta = 1$,
- les bruits blancs en entrée et sortie du système sont de variance 1 : $\sigma_p^2 = 1$ et $\sigma_q^2 = 1$. De plus, le contrôle du système doit respecter la contrainte suivante : la variance maximale supportée par ce système est $v_{max} = 10000$.

(2) Au niveau de l'architecture distribuée support :

- la mémoire disponible dans l'équipement gérant les rejets doit respecter une contrainte sur k, afin de pouvoir mémoriser (m,k)-pattern : $k \leq 20$,
- la bande passante réservée à l'application de contrôle distribuée impose une contrainte sur m et k : $\frac{m}{k} \leq 0.5$.

Notons que, en l'absence de rejets d'échantillons, c'est-à-dire, en situation idéale, $\gamma_{opt} = -2,7$ et la limite supérieure de la variance de l'état du système est égale à $9,1099E+00$.

3.4.1.2 Influence de la longueur d'une suite d'échantillons consécutifs rejetés (suite de 0)

Dans un premier temps, nous montrons l'influence d'une suite de 0 dans un (m,k)-pattern de longueur k ($k \leq 20$) et de forme $(1,k)$. Pour ceci, nous calculons le gain optimal γ_{opt} qui fournit la plus petite borne supérieure de la variance de l'état du système pour des (m,k)-pattern de type $(1,k)$. La table 3.1 fournit les résultats obtenus.

TABLE 3.1 – $(1,k)$-pattern - Influence de la longueur d'une suite de rejets consécutifs

Contrainte (m,k)	(m,k)-pattern π	γ_{opt}	$M(\gamma_{opt},\pi)$
(1, 2)	10	-2,966667	9,0111E+01
(1, 3)	100	-2,995556	8,1915E+02
(1, 4)	1000	-2,999259	7,3807E+03
(1, 5)	10000	-3	6,6430E+04
(1, 6)	100000	-3	5,9787E+05
(1, 7)	1000000	-3	5,3808E+06
(1, 8)	10000000	-3	4,8428E+07
(1, 9)	100000000	-3	4,3585E+08
(1,10)	1000000000	-3	3,9226E+09
(1,11)	10000000000	-3	3,5304E+10
(1,12)	100000000000	-3	3,1773E+11
(1,13)	1000000000000	-3	2,8596E+12
(1,14)	10000000000000	-3	2,5736E+13
(1,15)	100000000000000	-3	2,3163E+14
(1,16)	1000000000000000	-3	2,0846E+15
(1,17)	10000000000000000	-3	1,8762E+16
(1,18)	100000000000000000	-3	1,6886E+17
(1,19)	1000000000000000000	-3	1,5197E+18
(1,20)	10000000000000000000	-3	1,3677E+19

Les résultats obtenus montrent que la borne supérieure de la variance croit très vite (exponentiellement) avec la longueur de la suite de 0 consécutifs (voir les résultats obtenus pour $k = 1..4$ à la figure 3.2.)

Pour respecter les contraintes imposées sur le contrôle du système (contrainte sur la valeur supérieure de la variance de l'état du système), nous ne pouvons conserver, pour une forme de contrainte $(1,k)$, que les contraintes et (m,k)-pattern suivants :

- $(1,2)$ et $\Pi = 10$,
- $(1,3)$ et $\Pi = 100$,
- $(1,4)$ et $\Pi = 1000$.

On peut alors se demander quelle est l'influence d'une suite cumulative de 1 ; ceci est l'objet des deux expériences suivantes.

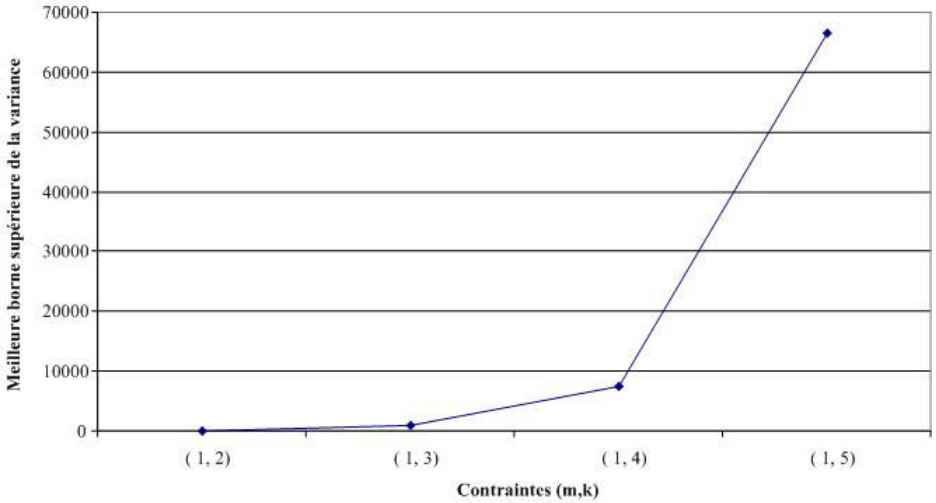

FIGURE 3.2 – Evolution de la meilleure borne sup de la variance en fonction de la longueur de la suite de rejets consécutifs (les (m,k)-patterns sont de type $(1,k)$-pattern, avec $k \leq 5$)

3.4.1.3 Rôle d'une suite d'échantillons transmis (influence de la longueur d'une suite de 1)

Pour évaluer cette influence, dans un premier temps, nous étudions tout d'abord les contraintes (m,k) telle que $m \leq k-3$, $k \leq 20$ (c'est-à-dire la contrainte qui possède la plus longue suite de 0 et qui soit acceptable dans le cas de contrainte $(1,k)$-firm). Les résultats sont présentés dans la table 3.3. Ces résultats ainsi que la figure 3.3 montrent que la borne supérieure de la variance est peu améliorée par une suite de 1 (au plus $8,79\%$ entre le pattern $(17,20)$qui fournit la plus petite limite supérieure de la variance de l'état du système et le pattern $(1,4)$ qui en fournit la plus grande).

Dans un deuxième temps, nous avons évalué l'influence de la longueur d'une suite d'échantillons acceptés dans le cas où le pattern contient déjà une suite de longueur strictement supérieure à 3 de rejets consécutifs. Pour ceci, nous avons étudiés les (m,k)-pattern tels que $k-m=4$ et les 4 rejets sont consécutifs. Les résultats sont donnés dans la table. Ils prouvent que, même en transmettant au contrôleur une suite d'échantillons de la plus grande longueur possible dans cette étude de cas (16 échantillons consécutifs), ceci ne permet pas d'atteindre les performances exigées (toutes les valeurs calculées de la limite supérieure de la variance sont plus grande que $v_{max} = 10000$).

De l'ensemble des expériences précédentes, nous concluons que les seules contraintes acceptables sont telles que :

- $k \leq 20$,
- $\frac{m}{k} \leq 0,5$,
- la longueur d'une suite de rejets consécutifs doit être inférieure ou égale à 3.

Dans la suite, nous étudions systématiquement tous les patterns remplissant ces conditions et conservons ceux qui respectent la limite imposée sur la borne supérieure de la variance de

TABLE 3.3 – Influence de la longueur d'une suite d'échantillons acceptés dans un (m, k)-pattern, tel que $m \leq k - 3$, $k \leq 20$ et les 3 rejets sont consécutifs.

Contrainte (m, k)	(m, k)-pattern π	γ_{opt}	$M(\gamma_{opt}, \pi)$
(1, 4)	1000	-2,999259	7,3807E+03
(2, 5)	11000	-2,946114	7,2122E+03
(3, 6)	111000	-2,853333	6,9520E+03
(4, 7)	1111000	-2,780654	6,8049E+03
(5, 8)	11111000	-2,731013	6,7479E+03
(6, 9)	111111000	-2,711325	6,7342E+03
(7,10)	1111111000	-2,700249	6,7323E+03
(8,11)	11111111000	-2,708571	6,7323E+03
(9,12)	111111111000	-2,708788	6,7323E+03
(10,13)	1111111111000	-2,697926	6,7323E+03
(11,14)	11111111111000	-2,703561	6,7320E+03
(12,15)	111111111111000	-2,696066	6,7324E+03
(13,16)	1111111111111000	-2,705097	6,7321E+03
(14,17)	11111111111111000	-2,699708	6,7321E+03
(15,18)	111111111111111000	-2,711013	6,7325E+03
(16,19)	1111111111111111000	-2,707017	6,7321E+03
(17,20)	11111111111111111000	-2,703445	6,7320E+03

l'état du système ($M(\gamma_{opt}, \Pi) \leq 10000$).

3.4.1.4 Construction, évaluation des pattern acceptables et conclusions sur cet exemple

La dernière expérience prend en compte :

- toutes les contraintes (m,k), telles que : $k \leq 20$, $\frac{m}{k} \leq 0.5$
- pour chaque couple (m,k) retenu, tous les patterns possibles (après élimination des pattern redondants, comme les patterns équivalents par rotation circulaire des lettres du pattern ou ceux qui sont obtenus à partir de la répétition d'un autre pattern)
- et $v_i \leq v_{max}$.

Le nombre de pattern restant est 17136 pour $k \leq 20$; il descend à 931 pour $k \leq 15$ et à 71 pour $k \leq 10$. Les expériences donnent les résultats suivants :

- Le (m,k)-pattern donnant la plus petite borne supérieure de la variance de l'état du système est le pattern $\Pi = 10$ (borne supérieure de la variance de l'état du système, $M(\gamma_{opt}, \Pi)$, égale à $9,0111E+01$ et gain optimal γ_{opt} égal à $-2,9667$).
- Les (m,k)-pattern donnant la plus grande borne supérieure acceptable de la variance

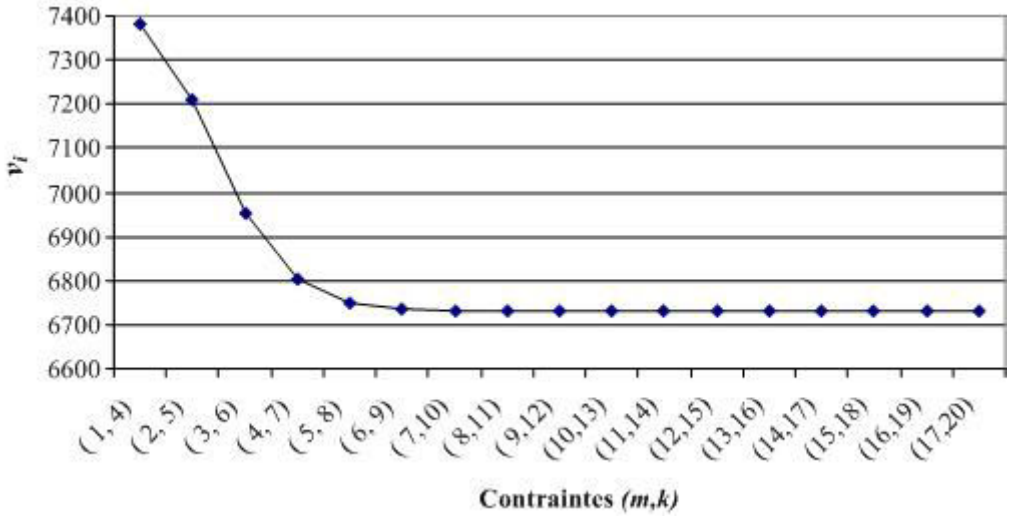

FIGURE 3.3 – Influence de la longueur d'une suite d'échantillons consécutifs acceptés sur la borne supérieure de l'état du système

de l'état du système correspond à tous les (m,k)-pattern qui contiennent au moins une suite de rejets consécutifs de longueur 3. La valeur de $M(\gamma_{opt}, \Pi)$ est entre $7,3810E+03$ et $6,7705E+03$ (ces deux valeurs correspondant, respectivement, à des gains optimum γ_{opt}, de $-3,0000$ et $-2,7600$.

- Si on étudie la bande passante, la contrainte et le pattern acceptable minimisant la bande passante est obtenu pour $\Pi = 1000$. l'indicateur d'occupation de bande passante est $m/k = 0,25$. Dans ce cas, $\gamma_{opt} = -2,9992$ et $M(\gamma_{opt}, \Pi) = 7,3807E+03$.
- Tous les (m,k)-pattern qui contiennent au moins une suite de rejets consécutifs de longueur 3 sont acceptables.
- Tous les (m,k)-pattern qui contiennent au moins une suite de rejets consécutifs de longueur 2 sont acceptables. Parmi ceux-ci ceux qui ne contiennent pas de suite de rejets consécutifs de longueur 3 conduisent à des bornes supérieures de la variance de l'état du système comprises entre $7,6828E+02$ et $8,1971E+02$.
- On peut remarquer que pour un même taux d'occupation de la bande passante, on obtient des performances du système très différentes. Ainsi, pour un rapport $\frac{m}{k} = 0.5$, les (m,k)-pattern $\Pi = 10$, $\Pi = 1100$, $\Pi = 111000$ donnent des valeurs optimales de la variance de l'état du système respectivement $90,1$, $785,9$ et $6952,04$. En corollaire, on peut affirmer que si on fixe une longueur k et un taux $\frac{m}{k}$, la plus petite variance est obtenue pour une répartition uniforme des 1 dans le pattern.
- Dans chaque cas, l'algorithme 1 donne la valeur du gain à utiliser. Notons que, dans cet exemple, les valeurs de gains obtenues pour des (m,k)-pattern dont la plus grande longueur de rejets consécutifs est identique sont très proches (écarts relatifs inférieur à 8%).

A titre indicatif, les (m,k)-pattern calculés pour $k \leq 10$ figurent en annexe C.

TABLE 3.4 – Influence de la longueur d'une suite d'échantillons acceptés dans un (m, k)-pattern, tel que $m \leq k - 4$, $k \leq 20$ et les 4 rejets sont consécutifs.

Contrainte (m, k)	(m, k)-pattern π	γ_{opt}	$M(\gamma_{opt}, \pi)$
(1, 5)	10000	-3,0000E+00	6,6430E+04
(2, 6)	110000	-2,9733E+00	6,5649E+04
(3, 7)	1110000	-2,8983E+00	6,3601E+04
(4, 8)	11110000	-2,8267E+00	6,1962E+04
(5, 9)	111110000	-2,7675E+00	6,1063E+04
(6,10)	1111110000	-2,7212E+00	6,0700E+04
(7,11)	11111110000	-2,7118E+00	6,0605E+04
(8,12)	111111110000	-2,6998E+00	6,0592E+04
(9,13)	1111111110000	-2,7054E+00	6,0590E+04
(10,14)	11111111110000	-2,7036E+00	6,0589E+04
(11,15)	111111111110000	-2,7049E+00	6,0589E+04
(12,16)	1111111111110000	-2,7088E+00	6,0591E+04
(13,17)	11111111111110000	-2,7005E+00	6,0590E+04
(14,18)	111111111111110000	-2,7078E+00	6,0591E+04
(15,19)	1111111111111110000	-2,7016E+00	6,0589E+04
(16,20)	11111111111111110000	-2,6961E+00	6,0593E+04

3.4.2 Système stable

Considérons maintenant un système stable défini ainsi :

(1) Le système contrôlé est décrit par les paramètres suivants :

- $\alpha = 0.5117$, $\beta = 0.4883$,
- les bruits blancs en entrée et sortie du système sont de variance1 : $\sigma_p^2 = 1$ et $\sigma_q^2 = 1$. Le contrôle du système doit respecter la contrainte suivante : la variance maximale supportée par ce système est $v_{max} = 0.323$.

(2) Au niveau de l'architecture distribuée support :

- la mémoire disponible dans l'équipement gérant les rejets doit respecter une contrainte sur k, afin de pouvoir mémoriser (m,k)-pattern : $k \leq 10$,
- la bande passante réservée à l'application de contrôle distribuée impose une contrainte sur m et k : $\frac{m}{k} \leq 0.5$.

3.4.2.1 Influence de la longueur d'une suite de rejets consécutifs

Comme le système est naturellement stable, il est moins sensible que celui décrit en section 3.4.1 à une suite de rejets d'échantillons (voir table 3.5 et figure 3.4 pour lesquels les

TABLE 3.5 – Système stable - Influence d'une suite de rejets consécutifs

Contrainte (m,k)	(m,k)-pattern Π	γ_{opt}	$M(\gamma_{opt}, \Pi)$
(1, 2)	10	-0,247483	0,317388
(1, 3)	100	-0,265785	0,321586
(1, 4)	1000	-0,130818	0,322731
(1, 5)	10000	-0,450499	0,322973
(1, 6)	100000	0,119604	0,346842
(1, 7)	1000000	-1,047921	0,500274
(1, 8)	10000000	-1,047921	0,500274
(1, 9)	100000000	-1,047921	0,500274
(1,10)	1000000000	-1,047921	0,500274
(1,11)	10000000000	-1,047921	0,500274
(1,12)	100000000000	-1,047921	0,500274
(1,13)	1000000000000	-1,047921	0,500274
(1,14)	10000000000000	-1,047921	0,500274
(1,15)	100000000000000	-1,047921	0,500274
(1,16)	1000000000000000	-1,047921	0,500274
(1,17)	10000000000000000	-1,047921	0,500274
(1,18)	100000000000000000	-1,047921	0,500274
(1,19)	1000000000000000000	-1,047921	0,500274
(1,20)	10000000000000000000	-1,047921	0,500274

expériences ont été conduites jusqu'à $k = 20$.)

3.4.2.2 Influence de la longueur d'une suite d'échantillons consécutifs acceptés

De cette expérience, nous conservons uniquement les $(1, k)$-pattern qui respectent la contrainte $M(\gamma_{opt}, \pi) \leq v_{max}$, c'est-à-dire :

$(1, 2)$ avec $\Pi = 10$ $(M(\gamma_{opt}, \Pi) = 0, 317388)$,

$(1, 3)$ avec $\Pi = 100$ $(M(\gamma_{opt}, \Pi) = 0, 321586)$,

$(1, 4)$ avec $\Pi = 1000$ $(M(\gamma_{opt}, \Pi) = 0, 322731)$,

$(1, 5)$ avec $\Pi = 1000$ $(M(\gamma_{opt}, \Pi) = 0, 322973)$.

Nous calculons alors la borne supérieure optimisée pour toutes les les contraintes (m,k) telle que $m \leq k - 5$; dans ce cas, nous menons également les expériences pour $k \leq 20$. Les résultats sont présentés dans la table 3.6. De la même manière que dans l'exemple précédent, on peut voir que les performances du système sont peu influencées par la longueur d'une suite d'échantillons consécutifs transmis (moins de 7% d'amélioration.)

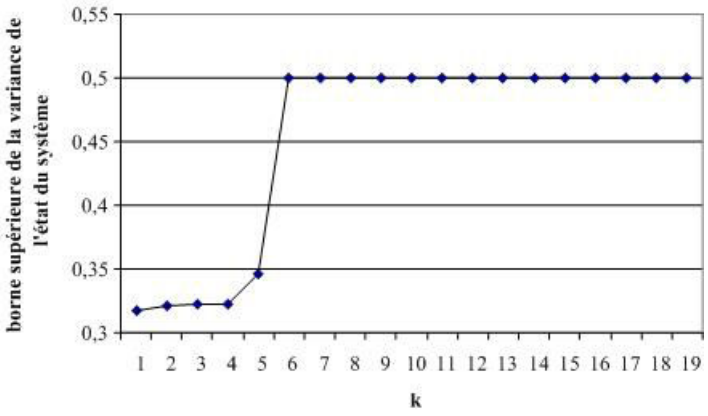

FIGURE 3.4 – Système stable - Influence d'une suite de rejets consécutifs sur la borne supérieure de v_i, $M(\gamma_{opt}, \pi)$

3.4.2.3 Choix d'une contrainte (m,k) et d'un (m,k)-pattern optimisé

Enfin, nous analysons les résultats obtenus pour toutes les contraintes (m,k) correspondant à la spécification de l'étude de cas ($k \leq 10$, $\frac{m}{k} \leq 0.5$) et telles que $M(\gamma_{opt}, \Pi) \leq 0.323$. Les résultats complets figurent en annexe C.

Pour optimiser les performances du système, on cherche à minimiser la valeur de v_i et on applique alors une contrainte (1,2) qui permet d'obtenir $v_i = 0.3173$ en utilisant un gain $\gamma = -0.247483$. Par contre, si on désire minimiser la bande passante consommée, on applique un pattern $(1,5)$ pour lequel, avec un gain $\gamma = -0,450499$, on obtient $v_i = 0.3229$. En fait, en étudiant les résultats obtenus pour tous les (m,k)-pattern possibles, nous remarquons que l'écart relatif sur la borne supérieure de la variance de l'état du système est inférieur à $1,75\%$. On peut conclure, dans ce cas, qu'il faut simplement éviter d'avoir une suite de rejets consécutifs de longueur strictement supérieure à 4.

3.5 Conclusion

Dans l'industrie moderne, les différents dispositifs des systèmes de contrôle commande deviennent de plus en plus distribués, la performance de tels systèmes dépend non seulement des caractéristiques du système lui-même (l'algorithme de contrôle, les propriétés du processus contrôlé), mais aussi d'effets induits par l'introduction du réseau (pertes de messages, délai, etc). Dans ce chapitre, nous considérons le cas où il est possible de mettre en oeuvre une stratégie de gestion de surcharge reposant sur une contrainte (m,k)-firm pour diminuer la bande passante consommée par une application de contrôle : les messages contenant les mesures de l'état du processus sont sélectivement rejetés pour éviter la surcharge du réseau. Nous avons identifié les critères de stabilité du système dans un contexte de rejets d'échantillons et déterminé les paramètres admissibles du réseau en termes de contrainte (m,k)-firm qui garantissant la stabilité du système. De cette condition, nous avons déterminé comment,

TABLE 3.6 – Système stable - Influence de la longueur d'une suite d'échantillons consécutifs acceptés

Contrainte (m,k)	(m,k)-pattern Π	γ_{opt}	$M(\gamma_{opt}, \Pi)$
(1, 6)	100000	0,119604	0,346842
(2, 7)	1100000	-0,17321	0,322987
(3, 8)	11100000	-0,297217	0,322985
(4, 9)	111100000	-0,196163	0,322985
(5,10)	1111100000	-0,247483	0,322984
(6,11)	11111100000	-0,260471	0,322984
(7,12)	111111100000	-0,254743	0,322984
(8,13)	1111111100000	-0,238542	0,322984
(9,14)	11111111100000	-0,21591	0,322984
(10,15)	111111111100000	-0,24631	0,322984
(11,16)	1111111111100000	-0,214815	0,322984
(12,17)	11111111111100000	-0,235693	0,322984
(13,18)	111111111111100000	-0,25295	0,322984
(14,19)	1111111111111100000	-0,215417	0,322984
(15,20)	11111111111111100000	-0,228593	0,322984

sous une contrainte (m,k) donnée, calculer la valeur inférieure et la valeur supérieure du gain du contrôleur qui garantisse la stabilité du système. Nous montrons, en particulier que la stabilité ne dépend que de m et de k et non du (m,k)-pattern.

Si on considère comme critère à optimiser, la borne supérieure de la variance de l'état du système, le problème de recherche de gain optimal revient, alors à parcourir toutes les valeurs du gain dans l'intervalle de stabilité pour chaque contrainte (m,k) et chaque (m,k)-pattern possible. Nous avons montré comment déterminer ce gain expérimentalement en considérant deux études de cas.

Dans ce chapitre, nous avons traité le problème de bande passante contrainte pour la transmission d'échantillons entre capteurs et contrôleur dans le cas d'un système monodimensionnel. Le critère de performance utilisé est la borne supérieure de la variance de l'état du système. Dans les chapitres suivants, nous considérons des systèmes plus complexes ainsi que d'autres critères de performances.

Chapitre 4

Impact d'un (m,k)-pattern sur la qualité de contrôle d'un système multi-dimensionnel

Dans ce chapitre, nous considérons un système multi-dimensionel pour lequel les informations de sortie du système contrôlé sont transmises au contrôleur par un réseau de communication. Nous présentons dans un premier temps la représentation mathématique du système étudié ainsi qu'une étude de cas sur laquelle nous appliquerons les techniques développées (section 4.1). Puis, dans la section 4.2, en nous appuyant sur l'étude des systèmes de deuxième ordre, nous proposons une approche pour concevoir les paramètres du réseau, exprimés en termes de contrainte (m,k)-firm, sous lesquels le système reste stable. Cette analyse nous permet de calculer les valeurs acceptables de k, au sens de la stabilité (le système reste stable pour une contrainte $(1,k)$-firm. Enfin, nous étudions, en section 4.3, un problème d'optimisation et, pour ceci, nous identifions un critère de performance du contrôle ainsi que la fonction objectif correspondant, puis nous spécifions une méthode pour trouver les valeurs de m pour toute valeur de k acceptable et les (m,k)-pattern qui fournissent une valeur optimisée de ce critère. Nous montrons, en particulier, que la répartition de rejets de messages dans la séquence de messages exerce, dans le contexte de l'étude de cas, une influence sur la performance de contrôle. Nous concluons ce chapitre en section 4.4.

4.1 Spécification du système

4.1.1 Description générique et notations

Nous nous intéressons dans ce chapitre à des systèmes dont l'architecture est illustrée par la figure 4.1. Le système à contrôler est défini un processus à temps continu de la forme :

$$dx = Axdt + Budt + dv_c \tag{4.1}$$

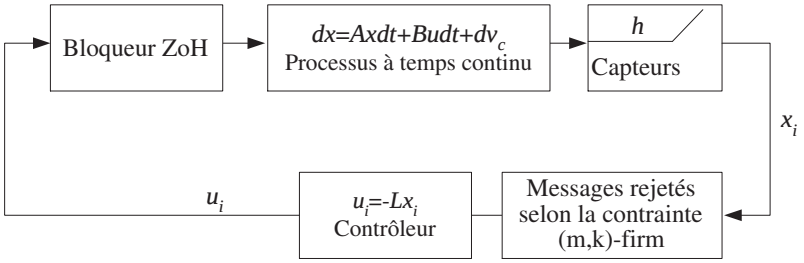

FIGURE 4.1 – Système de commande en réseau sous la contrainte (m,k)-firm

où x est le vecteur d'état (variables d'état du système) et u est la sortie du contrôleur avec $x \in \mathbb{R}^p$, $u \in \mathbb{R}^q$. Les paramètres A et B sont des matrices de dimensions respectives (p,p) et (p,q). v_c est un bruit blanc de covariance $R_c dt$ où R_c est une matrice constante de dimension (p,p).

L'état du processus contrôlé est échantillonné périodiquement avec une période d'échantillonnage h. L'état échantillonné à l'instant ih, noté x_i est envoyé au contrôleur linéaire discret qui est décrit comme suit :

$$u_i = -Lx_i \quad i = 0, 1, 2... \tag{4.2}$$

Nous supposons que les messages contenant le vecteur d'état du processus contrôlés sont rejetés sélectivement suivant une stratégie implantant une contrainte (m,k)-firm et (m,k)-pattern donnés afin de diminuer la bande passante occupée par cette application de contrôle. De plus, nous appliquons la règle suivante : lorsqu'un message est rejeté sur le réseau, le contrôleur ne produit pas la commande et sa valeur précédente est maintenue. On obtient donc un système dynamique à temps discret :

$$x_{i+1} = \Phi_i x_i + \Gamma_i u_i + v_i, \; i = 0, 1, 2... \tag{4.3}$$

avec

$$\begin{aligned} \Phi_i &= e^{Af_i} \\ \Gamma_i &= \int_0^{f_i h} e^{As} ds B \end{aligned}$$

où la variable f_i donne la distance, en terme de nombre de périodes, entre deux mises à jours de la commande. v_i est le bruit blanc discret de valeur moyenne nulle avec :

$$E v_i v_i^T = R_1(f_i h) = \int_0^{f_i h} e^{As} R_c e^{A^T s} ds \tag{4.4}$$

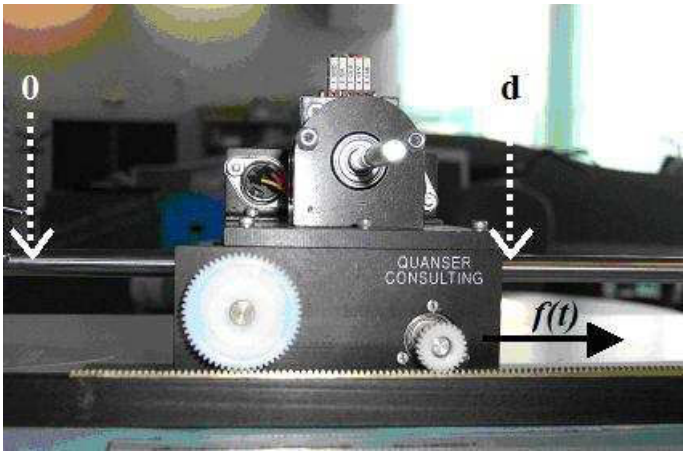

FIGURE 4.2 – Le contrôle du déplacement du chariot

Puisque les messages sont rejetés périodiquement selon un (m, k)-pattern donné, on a $f_i + f_{i+1} + f_{i+m-1} = k$ et $f_i = f_{i+m}, \forall i > 0$, le système 4.3 est ainsi périodique avec la période m, à savoir $\Phi_i = \Phi_{i+m}$, $\Gamma_i = \Gamma_{i+m}$.

Nous pouvons noter que les variables f_i peuvent aussi être utilisés pour représenter la distribution de rejets de message dans la séquence de messages : à chaque (m,k)-pattern donné correspond une suite de valeurs des variables f_i pour $i \in [0, m-1]$. Par exemple, étant donné le $(2, 7)$-pattern $\Pi = 1001000$, on a $f_0 = 3$, $f_1 = 4$.

Par la suite, nous illustrerons les propositions sur une étude de cas que nous présentons dans la section suivante.

4.1.2 Présentation de l'étude de cas

Le système étudié en exemple est un chariot qui peut se déplacer sur un rail (figure 4.2). L'objectif du contrôle est de guider le chariot selon une référence de position. Dans cette étude, le calcul des paramètres du contrôleur ainsi que les analyses de la stabilité et de la performance du système sont menés sous hypothèse d'un chariot à roulement avec frottement visqueux.

Notons d, la position du chariot mesurée par rapport à un repère de position, et $\dot{d}$ la vitesse du chariot. Le vecteur d'état est $x^T = [d, \dot{d}]$. Dans cet exemple, les dimensions p et q du modèle générique précédent sont donc égales respectivement à 2 et 1. Le modèle simplifié du système est décrit par l'équation différentielle suivante :

$$\ddot{d} = -k_1 \dot{d} + k_2 u \tag{4.5}$$

avec les paramètres $k_1 = 12.6559$ et $k_2 = 1.9243$ obtenus après identification du système.

Le modèle d'état continu du système est donné par l'équation générique 4.1 avec, dans cet

exemple, $A = \begin{bmatrix} 0 & 1 \\ 0 & -k_1 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ k_2 \end{bmatrix}$ et $v_c = 0$.

La sortie du chariot est échantillonnée périodiquement avec la période h. Nous supposons que l'état échantillonné est directement accessible au contrôleur. Le modèle à temps discret en boucle fermée est alors donné par :

$$x_{i+1} = \Phi(h)\, x_i + \Gamma(h)\, u_i \tag{4.6}$$

avec $\Phi(h) = \begin{bmatrix} 1 & \frac{1-e^{-k_1 h}}{k_1} \\ 0 & e^{-k_1 h} \end{bmatrix}$ et $\Gamma(h) = \frac{k_2}{k_1} \begin{bmatrix} h - \frac{1-e^{-k_1 h}}{k_1} \\ 1 - e^{-k_1 h} \end{bmatrix}$.

La commande du chariot est calculée comme suit :

$$u_i = L\,(x_{ref} - x_i) \tag{4.7}$$

où L est le gain $L = \begin{bmatrix} k_c & k_d \end{bmatrix}$ et x_{ref} la référence donnée par $x_{ref} = \begin{bmatrix} d_{ref} & 0 \end{bmatrix}^T$, avec d_{ref} la référence de position du chariot.

Calcul de la période d'échantillonnage.

La période d'échantillonnage h est déterminée en respectant la règle empirique ("*rule of thumb*") donnée dans [Astrom97] : $0.2 \leq \omega_0 h \leq 0.6$ où ω_0 est la pulsation naturelle du chariot ($\omega_0 = 20$) dans cet exemple. La période h doit donc être choisie dans l'intervalle $[0.01, 0.03]$. En examinant les valeurs dans cet intervalle, nous trouvons que la performance du système (exprimée en terme de temps de montée et de dépassement) est la meilleure lorsque la période est égale à la borne inférieure. On définit ainsi $h = 0.01s$. Notons que l'algorithme de calcul de la période h a pour objectif de permettre de reconstruire le signal de sortie du système.

Calcul des paramètres du contrôleur.

Avec la période d'échantillonnage $h = 0.01s$ choisie, $k_{c,0.01}$ et $k_{d,0.01}$ sont calculés classiquement en résolvant le problème LQR (linear quadratic regulator) donné dans [Astrom97] et on obtient : $k_{c,0.01} = 121$ et $k_{d,0.01} = 6.5$.

4.2 Analyse de stabilité - Calcul de la valeur de k

Dans cette section, nous illustrons comment déterminer la contrainte (m,k)-firm afin que le système reste stable. Bien que les analyses soient menées ici sur un système du deuxième ordre, l'approche proposée peut être généralisée sans grandes difficultés pour traiter d'autres systèmes multi-dimensionnels.

Un objectif de l'application de la contrainte (*m,k*)-firm à la transmission d'échantillons entre le capteur d'état du système et le contrôleur est de décroître le niveau de bande passante requise par l'application tout en préservant la propriété de stabilité du système. Cependant, à chaque type de contrainte (*m,k*)-firm et à chaque forme de (*m,k*)-pattern, correspond une valeur d'un indicateur qui représente la Qualité de Contrôle (QdC).

Nous étudierons cet aspect dans la section suivante (section 4.3). La configuration de la politique de rejet d'échantillons repose sur la spécification de k, de m et du (*m,k*)-pattern. En fait, ainsi que nous l'avons fait pour un système mono-dimensionnel au chapitre précédent, nous considérons que la bande passante réservée à cette application est spécifiée et peut s'exprimer sous la forme $\frac{m}{k} \leq taux_{max}$ ce qui permet de définir les valeurs possibles de m pour toute valeur de k admissible.

Intuitivement, nous pouvons dire que, si le système est stable pour une contrainte $(1,k)$-firm, il restera stable pour toute contrainte (*m,k*)-firm. Nous proposons alors de déterminer k_{max}, valeur maximale de k dans la contrainte $(1,k)$-firm, en deçà de laquelle le système reste stable pour toutes les contraintes (*m,k*)-firm avec $m \leq k$ et $k \leq k_{max}$. Nous considérons ici que si un système est stable sous une période donnée, il restera stable pour toutes périodes plus petites (dans des cas extrêmes, que nous ne considérons pas ici, le séquencement de différentes périodes sous chacune desquelles le système est stable, peut conduire à la non stabilité [Schinkel02]).

Par l'équation d'état 4.3, un système sous une contrainte (*m,k*)-firm peut être considéré comme un système avec des périodes d'échantillonnage variants sous une forme régulière, et le système sous la contrainte $(1,k)$-firm est équivalent au système avec la période k fois plus grande que sa période nominale h. Par conséquent, la détermination de la valeur maximale de k de la contrainte $(1,k)$-firm consiste à déterminer la période maximale h_c garantissant la stabilité du système.

Soit $\Psi(h_c) = \Phi(h_c) - \Gamma(h_c) L$, on a :

$$\Psi(h_c) = \begin{bmatrix} 1 - k_c \frac{k_2}{k_1} \left(h_c - \frac{1-e^{-k_1 h_c}}{k_1} \right) & \frac{1-e^{-k_1 h_c}}{k_1} - k_d \frac{k_2}{k_1} \left(h_c - \frac{1-e^{-k_1 h_c}}{k_1} \right) \\ k_c \frac{k_2}{k_1} \left(1 - e^{-k_1 h_c} \right) & e^{-k_1 h_c} - k_d \frac{k_2}{k_1} \left(1 - e^{-k_1 h_c} \right) \end{bmatrix}$$

D'après le critère de Jury [Freeman63], une condition suffisante et nécessaire de stabilité est de satisfaire les trois conditions suivantes :

$$\begin{aligned} 1.\ a_2 &< 1, \\ 2.\ a_2 &> a_1 - 1, \\ 3.\ a_2 &> -a_1 - 1 \end{aligned}$$

où $a_1 = \Psi_{1,1}\Psi_{2,2} - \Psi_{1,2}\Psi_{2,1}$, $a_2 = -\Psi_{1,1} - \Psi_{2,2}$ avec $\Psi_{i,j}$ l'élément se trouvant dans la i^{ime} ligne et la j^{ime} de la matrix Ψ ; $a_1$et a_2 sont exprimées en fonction des paramètres du contrôleur (k_c, k_d) et de la période d'échantillonnage h_c. Ceci définit donc un domaine admissible de triplets (k_c, k_d, h_c).

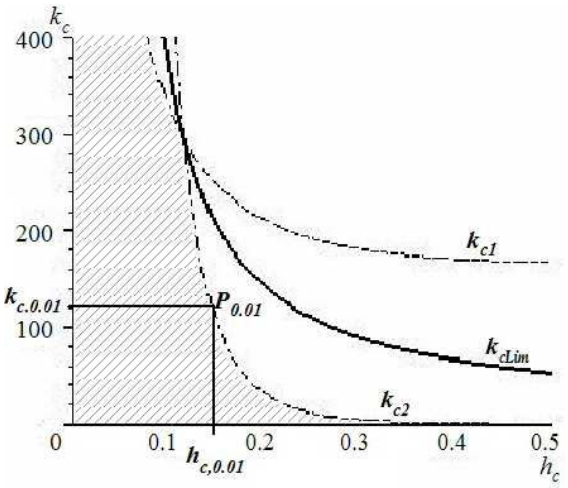

FIGURE 4.3 – Le contrôle du déplacement du chariot

Afin de déterminer les valeurs extrêmes de la période d'échantillonnage pour une valeur k_d donnée, on fixe $k_d = 6.5$ qui est obtenu avec la période nominale $h = 0.01s$. A l'aide d'un logiciel de calcul scientifique (*Mathematica*, par exemple), on trace la relation entre la période d'échantillonnage h_c et le paramètre du contrôleur k_c avec la contrainte suivante déduite des conditions de Jury :

La valeur maximale de k_c en fonction de h_c, pour k_d fixé, est notée $k_{cMax}(h_c)$; elle est donnée par :

$$k_{cMax}(h_c) = \begin{cases} k_{c1}(h_c) & \text{si } k_{c1}(h_c) < k_{cLim}(h_c) \\ k_{c2}(h_c) & \text{si } k_{c1}(h_c) > k_{cLim}(h_c) \end{cases}$$

où

$$\begin{aligned} k_{cLim}(h_c) &= \frac{4k_1 e^{k_1 h_c}}{h_c k_2 (e^{k_1 h_c} - 1)} \\ k_{c1}(h_c) &= \frac{2k_1 \left(k_1 + k_1 e^{k_1 h_c} + k_2 k_d - k_2 k_d e^{k_1 h_c}\right)}{k_2 (2 - 2e^{k_1 h_c} + h_c k_1 + h_c k_1 e^{k_1 h_c})} \\ k_{c2}(h_c) &= \frac{k_1 (k_1 + k_2 k_d)}{k_2 (-1 - h_c k_1 + e^{k_1 h_c})} \end{aligned}$$

et on obtient la région de stabilité dans le plan (h_c, k_c) où les conditions de Jury sont satisfaites. Cette région est hachurée sur la figure 4.3.

Pour la valeur de k_c, calculée initialement et notée $k_{c,0.01}$, la valeur maximale de la période d'échantillonnage préservant la stabilité du système est notée $h_{c,0.01}$ et correspond à l'abscisse du point P sur la figure 4.3. Puisque le rejet de message consiste à multiplier la période nominale du système, ainsi la période maximale correspondante, notée h_{max}, doit être multiple de

Paramètres du chariot $\ddot{d} = -k_1\dot{d} + k_2 u$	$k_1 = 12.6559$ $k_2 = 1.9243$
Paramètres du contrôleur $u_t = \begin{bmatrix} k_c & k_d \end{bmatrix} (x_{ref} - x_t)$	$h = 0.01$ $k_c = 121$ $k_d = 6.5$
Valeurs admissibles du paramètre k	$1 < k \leq 14$

TABLE 4.1 – Paramètres du système, du contrôleur et valeur de k de la contrainte (m,k)-firm

la période nominale. Par conséquence, on a $h_{max} = \left\lfloor \frac{h_{c,0.01}}{h_{nominal}} \right\rfloor h_{nominal}$ et la valeur maximale du paramètre k de la contrainte (m,k)-firm est donnée par $k_{max} = \frac{h_{max}}{h_{nominal}}$.

Etant donnés les paramètres du contrôleur, si la période maximale identifiée par la région de stabilité ne satisfait pas l'exigence du réseau en terme de bande passante induite $\frac{m}{k} > taux_{max}$, les paramètres du contrôleur doivent être recalculés et l'approche ci-dessus est ré-appliquée.

Les résultats pour le système chariot et un contrôleur préservant sa stabilité sont résumés dans le tableau 4.1.

4.3 Recherche d'une contrainte (m,k)-firm optimisée - Discussions sur le choix de m et d'un (m,k)-pattern

Dans la section précédente, nous avons proposé une approche pour identifier les contraintes (m,k)-firm, plus précisémment les valeurs de k avec $1 \leq m \leq k$, sous lesquelles le système reste stable. Dans cette section, nous définissons un indicateur de performance du contrôle (Qualité du Contrôle - QdC) et nous nous attachons à étudier l'influence de la valeur de m, ainsi que des différents (m,k)-patterns sur cet indicateur. Pour évaluer cette influence et définir une solution optimisée en respectant la contrainte imposée sur la bande passante utilisée par l'application de contrôle ($taux_{max} \leq \frac{m}{k}$), nous simulons le système sous Matlab/Simulink selon un plan d'expériences que nous détaillons ci-dessous.

4.3.1 Indicateurs de performance

Nous étudions la réponse du chariot initialement immobile ($\dot{d}_0 = 0$) à une consigne de déplacement de deux centimètres ($d_{ref} = 0.02m$) à partir de la position courante ($d_0 = 0$). La performance du système est représentée par le coût LQR, $J_{(m,k,\Pi)}$, pour une contrainte donnée (m,k,Π), qui est donné par :

$$J_{(m,k,\pi)} = \frac{1}{N}\sum_{i=1}^{N}\left((d_i - d_{ref})^2 + Ru_i^2\right) \tag{4.8}$$

où :

- Nh est l'horizon temporel, avec N le nombre de périodes d'échantillonnage,
- et R est le coefficient de pondération pour prendre en compte la limite de la tension.

Plus le coût 4.8 est grand, pire sera la performance de contrôle. Optimiser le système reviendrait donc à trouver la configuration qui fournit le plus petit coût. Un premier indicateur de performance est la dégradation de performance $Q_{(m,k,\Pi)}$ qui est fournie par :

$$Q_{(m,k,\Pi)} = \frac{J_{(m,k,\pi)} - J_{idéal}}{J_{idéal}}$$

où J_{ideal} est le coût quadratique LQR obtenu en l'absence de rejets d'échantillons (contrainte $(1,1)$).

L'autre indicateur est une indication de taux d'occupation de bande passante, noté $T_{(m,k)}$:

$$T_{(m,k)} = \frac{m}{k}$$

Le système est simulé en utilisant Matlab/Simulink et la fonction de coût 4.8 est évaluée pour chaque (m,k)-pattern Π possible. Dans la suite, nous donnerons les résultats de simulation pour $k \leq 10$.

Evaluation du coût en l'absence de rejet d'échantillons. La valeur idéale trouvée dans ce cas (contrainte (k,k)-firm) est :

$$J_{\text{idéal}} = 4.516.10^{-3}$$

La réponse du système à un échelon de valeur 2cm est donnée en figure 4.4.

4.3.2 Influence de la longueur d'une suite de rejets consécutifs et impact de l'instant de la demande d'échelon dans une séquence Π

Dans cette série d'expériences, nous considérons $k = 10$ et étudions successivement :

- tous les pattern $(1, 10)$,
- tous les pattern $(2, 10)$ qui contiennent une suite de rejets consécutifs de longueur 8,
- puis, tous les pattern $(5, 10)$ qui contiennent une suite de rejets consécutifs de longueur 5,

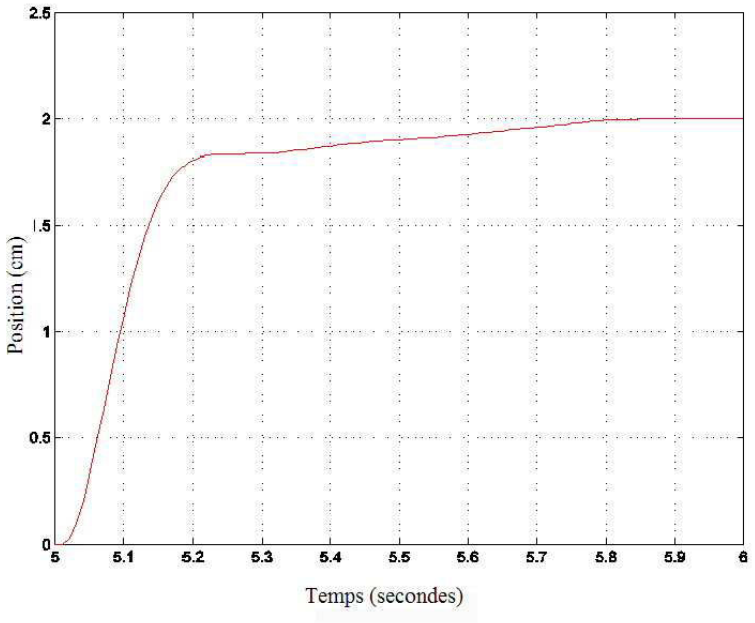

FIGURE 4.4 – La réponse du système à un échelon de valeur 2 cm sous la contrainte (k,k)-firm (sans rejet d'échantillons).

4.3.2.1 Pattern $(1,10)$

Cette première expérimentation consiste à évaluer les valeurs de l'indicateur de QdC dans le cas de contraintes $(1,10)$-firm ($T_{(m,k)} = \frac{m}{k} = 0.1$). Nous étudions tous les patterns obtenus pour un 1, respectivement en position 0, 1, ..., 9. Ceci nous permet d'évaluer l'impact de la désynchronisation entre l'instant de demande d'une nouvelle référence (t_0) et l'instant d'acceptation du premier échantillon (à t_0, la règle appliquée est donnée par $\Pi[0]$).

Les résultats sont résumés dans la table 4.2. Les résultats confirment ce qu'intuitivement on pouvait penser : plus la séquence de rejets est longue à partir de l'instant t_0, moins bonne est la performance (en effet, à partir du deuxième essai, la prise en compte de demande de référence est retardée, respectivement, de h, $2h$, ..., $9h$).

La dégradation de coût par rapport à la situation idéale est importante. Notons néanmoins que le système reste stable.

4.3.2.2 Pattern $(2,10)$ comprenant 8 rejets consécutifs

La deuxième expérience consiste à utiliser des contraintes $(2,10)$-firm en positionnant une suite de deux 1 consécutifs en positions $0,1$, puis $1,2$, etc. On obtient les coûts donnés dans la table 4.3. Dans ce cas, les résultats obtenus font apparaitre, comme dans l'expérience 4.2, l'influence du retard de prise en compte de la demande de référence pour les 9 premiers essais.

Nous remarquons que le dernier essai permet d'obtenir un coût meilleur que le premier. En effet, pour la séquence $\Pi = 1100000000$ on applique une valeur de u_0 et on réduit cette

TABLE 4.2 – Contrainte $(1, 10)$-firm. Impact de l'instant d'acceptation d'un échantillon dans une séquence temporisée décrite par un pattern Π.

Π, $(1, 10)$-pattern	Coût $J_{(1,10,\Pi)}$	Dégradation $Q_{(1,10,\Pi)}(\%)$
1000000000	9,71E-03	114,93%
0100000000	1,01E-02	123,79%
0010000000	1,05E-02	132,64%
0001000000	1,09E-02	141,50%
0000100000	1,13E-02	150,35%
0000010000	1,17E-02	159,21%
0000001000	1,21E-02	168,07%
0000000100	1,25E-02	176,92%
0000000010	1,29E-02	185,78%
0000000001	1,33E-02	194,64%

TABLE 4.3 – Contraintes $(2, 10)$-firm. Impact de l'instant d'acceptation d'une suite de 2 échantillons dans une séquence temporisée décrite par un pattern Π.

Π, $(2, 10)$-pattern	Coût $J_{(2,10,\Pi)}$	Dégradation $Q_{(2,10,\Pi)}(\%)$
1100000000	7,75E-03	71,65%
0110000000	8,15E-03	80,51%
0011000000	8,55E-03	89,37%
0001100000	8,95E-03	98,22%
0000110000	9,35E-03	107,08%
0000011000	9,75E-03	115,94%
0000001100	1,02E-02	124,79%
0000000110	1,06E-02	133,65%
0000000011	1,10E-02	142,51%
1000000001	7,15E-03	58,30%

TABLE 4.4 – Contraintes $(5,10)$-firm- impact de l'instant d'acceptation d'une suite de 5 échantillons dans une séquence temporisée décrite par un pattern Π.

Π, $(5,10)$-patterns	Coût $J_{(5,10,\Pi)}$	Dégradation $Q_{(5,10,\Pi)}(\%)$
1111100000	5,38E-03	19,01%
0111110000	5,78E-03	27,87%
0011111000	6,18E-03	36,72%
0001111100	6,58E-03	45,58%
0000111110	6,98E-03	54,44%
0000011111	7,38E-03	63,29%
1000001111	4,98E-03	10,34%
1100000111	5,18E-03	14,78%
1110000011	5,34E-03	18,20%
1111000001	5,40E-03	19,64%

valeur u_1 à $t_1 = h$. Cette deuxième valeur sera maintenue jusqu'à $t_{10} = 10h$: pour le pattern Π =1000000001, on applique la même valeur initiale u_0 et on maintient cette valeur jusqu'à $t_9 = 9h$; u_0 est plus grande que le u_1 maintenu durant la même durée dans l'essai précédent et le système va réagir plus rapidement et il est possible que quand on arrive à $t_{k-1} = (k-1)h$, la sortie du système soit plus proche de sa valeur finale.

Globalement les résultats sont peu différents de ceux obtenus avec une contrainte $(1,10)$. Sur les 9 premières lignes des deux tableaux, on peut voir que la présence d'une suite de 1 plus longue permet d'améliorer légèrement le coût. La pire dégradation est de $194,6\%$ dans le cas d'une contrainte $(1,10)$ alors qu'elle est de 142% dans le cas d'une contrainte $(2,10)$. Les indicateurs de taux d'occupation sont respectivement égaux à 0.1 et 0.2.

4.3.2.3 Pattern $(5,10)$ comprenant 5 rejets consécutifs

Nous avons ensuite réalisé une troisième expérience pour évaluer si en accroîssant la suite consécutive de 1, les performances s'amélioraient. Pour ceci, nous étudions toutes les séquences comportant une suite de 5 échantillons transmis successifs sous une contrainte $(5,10)$-pattern. Les résultats sont dans la table 4.4 et montrent une amélioration des résultats avec au pire une dégradation de 63.3% tandis que l'indicateur de taux d'occupation de bande passante est de 0.5.

4.3.2.4 Conclusions sur ces expériences

Nous pouvons remarquer que, contrairement à ce qui se passait au chapitre précédent, la longueur d'une suite de rejets consécutifs n'est pas le seul facteur qui impacte la qualité du contrôle. La distance entre l'instant de la demande de référence et la position des rejets dans

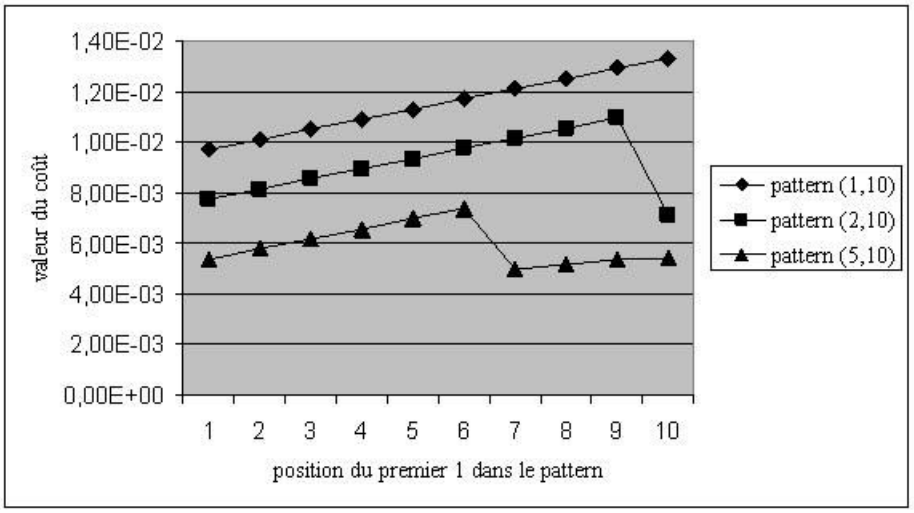

FIGURE 4.5 – Coûts obtenus pour les pattern $(1,10)$, $(2,10)$ et $(5,10)$ en fonction de la synchronisation de demande de changement de consigne par rapport au premier échantillon accepté de la suite Π. Les pattern $(m,10)$ considérés comprennent tous une suite de $10-m$ 1 consécutifs.

le temps joue également un rôle important dans le cas du système étudié.

La figure 4.5 reprend les résultats précédents et compare les coûts obtenus en appliquant les pattern $(1,10)$, $(2,10)$ et $(5,10)$ avec une suite, respectivement de 9, 8 et 5 échantillons consécutifs non transmis. On peut voir que, pour chaque position de la suite d'échantillons acceptés, le pattern $(5,10)$ est associé au plus petit coût tandis que le pattern $(1,10)$ est associé au plus grand. L'amélioration relative du coût entre le pattern $(1,10)$ et le pattern $(5,10)$ varie entre 38% et 60%.

4.3.2.5 Indicateur de qualité de contrôle

S'il est impossible, dans l'architecture opérationnelle, de synchroniser le (m,k)-pattern avec l'instant de demande de changement de référence, il est raisonnable de ne considérer que les pires coûts pour chaque (m,k)-pattern et de conserver celui qui offre les meilleures performances. C'est ce critère de choix (indicateur), noté $J^{max}_{(m,k)}$ que nous utiliserons dans la section suivante :

$$J^{max}_{(m,k)} = \max_{\Pi \epsilon \Pi_{(m,k)}} J_{(m,k,\Pi)}$$

où $\Pi_{(m,k)}$ désigne l'ensemble des (m,k)-pattern possibles pour un couple (m,k).

Le problème d'optimisation revient donc à trouver les paramètres m et k (paramètre k assurant la stabilité) ainsi que le (m,k)-pattern qui minimise la valeur de $J^{max}_{(m,k)}$. De la même manière que précédemment, nous faisons cette étude par un plan d'expérience sous Matlab/Simulink (voir section suivante).

TABLE 4.5 – Contrainte (m,k)-firm admissibles ($k \leq 10$ et $T_{(m,k)} \leq 0.25$) et (m,k)-pattern donnant la meilleure valeur de $J^{max}_{(m,k)}$

(m,k)	$\pi^{root}_{(m,k)}$	Coût $J^{max}_{(m,k)}$	$Q_{(5,10,\Pi)}$ max (%)	Coût $J^{min}_{(m,k)}$	$T_{(m,k)}$	$s(\pi^{root}_{(m,k)})$
$(1,4)$	1000	$6,28E-03$	$39,14$	$5,08E-03$	$0,250$	3
$(1,5)$	10000	$7,04E-03$	$55,88$	$5,44E-03$	$0,200$	4
$(1,6)$	100000	$11,7E-03$	$159,21$	$5,91E-03$	$0,167$	5
$(1,7)$	1000000	$12,1E-03$	$168,07$	$6,53E-03$	$0,143$	6
$(1,8)$	10000000	$12,5E-03$	$176,92$	$7,34E-03$	$0,125$	7
$(1,9)$	100000000	$12,9E-03$	$185,78$	$8,37E-03$	$0,111$	8
$(1,10)$	1000000000	$13,3E-03$	$194,64$	$7,15E-03$	$0,100$	9
$(2,8)$	11000000	$10,2E-03$	$124,79$	$5,90E-03$	$0,250$	6
$(2,8)$	10100000	$8,83E-03$	$95,59$	$5,49E-03$	$0,250$	5
$(2,8)$	10010000	$7,22E-03$	$59,95$	$5,23E-03$	$0,250$	4
$(2,9)$	110000000	$10,6E-03$	$133,65$	$6,43E-03$	$0,222$	7
$(2,9)$	101000000	$9,23E-03$	$104,44$	$6,03E-03$	$0,222$	6
$(2,9)$	100100000	$8,17E-03$	$80,93$	$5,61E-03$	$0,222$	5
$(2,9)$	100010000	$7,11E-03$	$57,36$	$5,32E-03$	$0,222$	4
$(2,10)$	1100000000	$11,0E-03$	$142,51$	$7,15E-03$	$0,200$	8
$(2,10)$	1010000000	$9,63E-03$	$113,30$	$6,13E-03$	$0,200$	7
$(2,10)$	1001000000	$8,57E-03$	$89,78$	$5,62E-03$	$0,200$	6
$(2,10)$	1000100000	$7,70E-03$	$61,61$	$5,41E-03$	$0,200$	5

4.3.3 Meilleur (m,k)-pattern Π pour un taux borné d'occupation de la bande passante

Nous considérons, dans cette section, que l'occupation de la bande passante par l'application de contrôle est limitée par la condition :

$$T_{(m,k)} \leq T_{max} = 0.25$$

et que, pour des raisons de limite mémoire du système dans lequel la politique (m,k)-firm est implantée, nous imposons :

$$k \leq 10$$

Le problème revient à chercher les valeurs de m et de k ainsi que le (m,k)-pattern qui donne la plus petite valeur de $J^{max}_{(m,k)}$.

Pour ceci, nous avons testé toutes les contraintes admissibles, dans le cas de l'étude de cas présentée dans ce chapitre, et avons obtenu les résultats présentés dans le tableau 4.5.

Les ensembles $\Pi^{root}_{(m,k)}$ sont générés à partir de (m,k)-pattern génériques, notés $\pi^{root}_{(m,k)}$ par permutation circulaire des lettres des (m,k)-pattern. Par exemple, à partir du $(2,8)$-pattern, $\pi^{root}_{(2,8)} = 10100000$, on génère l'ensemble $\Pi^{root}_{(m,k)}$ ={ 10100000, 01010000, 00101000, 00010100, 00001010, 00000101, 10000010, 01000001} .La plus longue suite de rejets consécutifs, pour chaque pattern générique, est notée $s(\pi^{root}_{(m,k)})$. Notons que $\pi^{root}_{(2,8)}$ tel que $s(\pi^{root}_{(2,8)}) = 3$ (respectivement $\pi^{root}_{(2,10)}$ tel que $s(\pi^{root}_{(2,10)}) = 4$) est équivalent à $\pi^{root}_{(1,4)}$ (respectivement $\pi^{root}_{(1,5)}$.) Dans ce tableau, nous donnons également $J^{min}_{(m,k)}$ la valeur minimale du coût trouvée pour chaque ensemble $\Pi^{root}_{(m,k)}$ afin de voir la dispersion des coûts dans l'ensemble.

Nous voyons dans ce tableau (3^{me} colonne) que la plus petite valeur de l'indicateur, $\Pi_{optimal}$ est obtenue pour $\Pi_{optimal} = \Pi_{1,4} = 1000$ (ou pour le pattern équivalent $\Pi_{2,8} = 10001000$). De plus, la colonne 5 montre que pour ce pattern, la plus petite valeur de $J^{min}_{(m,k)}$est également obtenue pour $\Pi_{optimal}$. Nous pouvons conclure, pour ce système, et sous les contraintes énoncées ($\frac{m}{k} \leq 0.25$ et $k \leq 10$), que les pattern "uniformes" offrent les meilleures performances.

4.4 Conclusions

Dans ce chapitre, nous proposons une technique pour minimiser la bande passante du réseau et qui est consommée par la transmission d'échantillons entre capteurs et contrôleur. Cette technique repose sur une politique de rejets d'échantillons selon une contrainte (m,k)-firm. Dans un premier lieu, en nous appuyant sur un système de deuxième ordre, nous identifions les paramètres des contraintes (m,k)-firm sous lesquelles le système de contrôle commande reste stable. Puis, par simulation sous Matlab/Simulink, nous montrons que la distribution de rejets de messages a une influence importante sur la performance de contrôle et nous déterminons le (m,k)-pattern optimal. Bien que les études se basent sur un système de deuxième ordre, elles peuvent être généralisées sans grande difficulté aux autres types de systèmes multi-dimensionnels.

Noter que dans [BenGaid06b], l'auteur montre que, en utilisant la fonction de coût de forme 4.8 comme la critère de QdC, le choix de la séquence optimale de rejets dépend largement de l'état initial x_0 du processus. Afin d'obtenir le (m,k)-pattern optimisant la QdC à long terme, nous développons, dans le chapitre suivant, une critère de QdC qui ne dépend pas de l'état initial du processus.

Ce chapitre considère que le contrôleur est donné (en particulier les paramètres $k_c$et k_d) pour une période d'échantillonnage h et que ces valeurs sont utilisées pour une période admissible kh. Nous montrons, dans le chapitre suivant comment, en utilisant une autre fonction de coût, il est possible de définir un contrôleur "adaptatif" et ainsi développer une conception conjointe des paramètres réseau et du contrôleur.

Chapitre 5

Méthode analytique de conception conjointe d'un système de contrôle distribué

Dans le chapitre précédent, le contrôleur était spécifié indépendamment de la politique (m,k)-firm utilisée pour gérer la transmission des échantillons de sortie du système au contrôleur. Cette loi de contrôle était déterminée pour assurer la stabilité du système dans le cas d'une contrainte $(1,k)$-firm. Ceci fournissait une borne supérieure pour k. L'identification de m ainsi que le (m,k)-pattern qui fournissait une solution minimisant une fonction de coût donnée était faite par simulation exhaustive de l'ensemble des pattern acceptables. Néanmoins, ces travaux ne prenaient pas en compte la possibilité, pour le contrôleur, de s'adapter, en temps réel, au (m,k)-pattern. Lorsque un message est rejeté sur le réseau, la commande pour contrôler le processus n'est pas mise à jour, ce qui conduit nécessairement à la dégradation de performance du système.

Afin de minimiser cette dégradation de performance, nous proposons, dans ce chapitre, une méthode pour calculer un contrôleur optimisé afin de compenser au mieux une suite de messages rejetés. Pour ceci, nous considérons un gain non constant et qui prend ses valeurs dans un ensemble de valeurs précalculées en fonction du (m,k)-pattern. Nous résolvons, en fait, un problème de commande optimale adaptative en minimisant la fonction de coût proposée dans [Gustafsson91]. Nous nous appuyons sur des travaux réalisés à l'Université de Wisconsin-Madison [Ramanathan99]. Dans ce cas, les auteurs proposent d'utiliser un pattern périodique et appliquent, à un système sans bruit, une technique d'évaluation de gains pour chaque séquence du pattern. Les travaux présentés dans [Eker00] considèrent, eux, un système avec bruit et des périodes qui peuvent être différentes ; néanmoins, ils conçoivent le contrôleur pour une période fixe constante (période de base). Dans les sections suivantes, nous étendons les méthodes proposées pour les appliquer au calcul d'un gain adaptatif. Puis sachant qu'on peut calculer la suite des gains correspondant à un (m,k)-pattern donné, nous montrons comment,

connaissant m et k, on peut calculer un (m,k)-pattern qui optimise une fonction de coût. Les travaux sont appliqués au système présenté au chapitre précédent.

5.1 Fonction de coût et commande optimale

Le système est donné, de la même manière que dans le chapitre précédent, par l'équation différentielle linéaire stochastique suivante :

$$dx = Axdt + Budt + dv_c$$

avec $x \in \mathbb{R}^p$, $u \in \mathbb{R}^q$, les matrices A et B de dimensions respectives (p,p) et (p,q). v_c est un bruit blanc de covariance $R_c dt$ où R_c est une matrice constante de dimension (p,p).

5.1.1 Loi de commande adaptative optimale

Considérons la fonction de coût à temps continu suivante :

$$J = E\left(\int_0^{Nh} (x^T(t)Qx(t) + u^T(t)Ru(t))dt + x^T(Nh)Qx(Nh)\right) \tag{5.1}$$

où Q et R sont des matrices pondérant respectivement l'état et l'entrée du système ; N est multiple du paramètre k de la contrainte (m,k)-firm. Nous supposons que le paramètre k est évalué pour assurer la stabilité du système de la même manière que dans le chapitre précédent.

Pour calculer le contrôleur optimal, la fonction de coût 5.1 est discrétisée en fonction de la période d'échantillonnage h, et on obtient, pour une valeur du paramètre m ainsi qu'un (m,k)-pattern donné :

$$J = E\left(\sum_{i=0}^{m\frac{N}{k}-1} \left(x_i^T Q_i' x_i + 2x_i^T M_i u_i + u_i^T R_i' u_i\right)\right) + E\left(x_{m\frac{N}{k}}^T Q_0' x_{m\frac{N}{k}}\right) + \sum_{i=0}^{m\frac{N}{k}-1} \bar{J}_i \tag{5.2}$$

où

$$\begin{aligned} Q_i' &= \int_0^{f_i h} \Phi^T(s)\, Q \Phi(s)\, ds \\ M_i &= \int_0^{f_i h} \Phi^T(s)\, Q \Gamma(s)\, ds \\ R_i' &= \int_0^{f_i h} \left(\Gamma^T(s)\, Q \Gamma(s) + R\right) ds \\ \bar{J}_i &= tr\left(Q \int_0^{f_i h} R_c(s)\, ds\right) \end{aligned} \tag{5.3}$$

avec $\Phi(t) = e^{At}$ et $\Gamma(s) = \int_0^s e^{At} dt B$.

Considérons la loi de commande suivante :

$$u = -L_i x \quad i = 0, 1, 2... \tag{5.4}$$

où L_i est le gain pour la période, de longueur f_i, comprise entre le i^{me} échantillon et le $(i+1)^{me}$ échantillon.

Le problème revient donc à déterminer l'ensemble $\{L_i \mid i\epsilon\,[0, m]\}$, c'est-à-dire l'ensemble des m gains correspondants au (m,k)-pattern, qui minimise la fonction de coût 5.2.

La loi de commande 5.4 est optimale si les L_i sont solutions de l'équation de Ricatti suivante [Gustafsson91] :

$$L_i = \left(\Gamma_i^T S_i \Gamma_i + R_i^{'}\right)^{-1} \left(\Gamma_i^T S_i \Phi_i + M_i^T\right) \tag{5.5}$$

où $\Gamma_i = \Gamma(f_i h)$ et $\Phi_i = \Phi(f_i h)$.

Les valeurs de S_i, $i = 0, 1, ..$ sont obtenues par l'équation récurrente suivante :

$$\begin{aligned} S_{m\frac{N}{k}} &= Q_0^{'} \\ S_j &= \Phi_j^T S_{j+1} \Phi_j + Q_j^{'} \\ &\quad - \left(\Gamma_j^T S_{j+1} \Phi_j + M_j^T\right)^T \left(\Gamma_j^T S_{j+1} \Gamma_j + R_j^{'}\right)^{-1} \left(\Gamma_j^T S_{j+1} \Phi_j + M_j^T\right) \text{ pour } 0 \leq j \leq m\frac{N}{k} - 1 \end{aligned} \tag{5.6}$$

Du fait que $\Phi_i = \Phi_{i+m}$ et $\Gamma_i = \Gamma_{i+m}$, la solution de l'équation récurrente 5.6 est aussi périodique avec une période correspondant à m échantillons et de durée kh quand l'horizon temporel $m\frac{N}{k} - 1$ est suffisamment long [Bittanti91]. Ceci donne :

$$S_i = S_{i+m}$$

Et par conséquent, la solution de l'équation 5.5 est périodique avec la période m :

$$L_i = L_{i+m}$$

Nous pouvons, ainsi, pour un (m,k)-pattern donné, calculer, au mieux la suite des L_i qui permettent de compenser le rejet des échantillons.

5.1.2 Calcul du coût pour une loi de contrôle avec gains adaptatifs

Nous donnons dans cette section la formule qui nous permet de calculer la fonction de coût 5.2 dans le cas où la loi de commande optimale est donnée par 5.4.

Etant donnés m, k et un (m,k)-pattern, la valeur de la fonction de coût 5.2 obtenue avec le contrôleur optimal adaptatif LQ 5.4 est donnée dans [Gustafsson91] par :

$$J = m_0^T S_0 m_0 + \sum_{i=0}^{m\frac{N}{k}-1} trS_{i+1} R_1 (f_i h) + \sum_{i=0}^{m\frac{N}{k}-1} \bar{J}_i \tag{5.7}$$

où $m_0 = E(x_0)$ et $R_1(t) = \int_0^t e^{As} Q e^{A^T s} ds$

L'intégrale $R_1(t) = \int_0^t e^{As} Q e^{A^T s} ds$ peut se calculer ainsi [Loan78] :

Considérons le système linéaire :

$$\begin{bmatrix} \dot{x_1}(t) \\ \dot{x_2}(t) \end{bmatrix} = \begin{bmatrix} -A & Q \\ 0 & A^T \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

qui a la solution suivante :

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \exp\left(\begin{bmatrix} -A & Q \\ 0 & A^T \end{bmatrix} t \right) \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} \tag{5.8}$$

que l'on peut calculer ainsi :

$$\begin{aligned} x_2(t) &= e^{A^T t} x_2(0) \\ \Rightarrow \dot{x_1}(t) &= -A x_1(t) + Q e^{A^T t} x_2(0) \\ \Rightarrow x_1(t) &= e^{-At} x_1(0) + e^{-At} \int_0^t e^{As} Q e^{A^T s} ds x_2(0) \end{aligned}$$

Si on pose :

$$\begin{bmatrix} \Psi_{11} & \Psi_{12} \\ \Psi_{21} & \Psi_{22} \end{bmatrix} = \exp\left(\begin{bmatrix} -A & Q \\ 0 & A^T \end{bmatrix} t \right)$$

alors l'équation 5.8 s'écrit :

$$\begin{aligned} x_1(t) &= \Psi_{11} x_1(0) + \Psi_{12} x_2(0) \\ x_2(t) &= \Psi_{21} x_1(0) + \Psi_{22} x_2(0) \end{aligned}$$

Soit :

$$R_1(t) = \int_0^t e^{As} Q e^{A^T s} ds = \Psi_{22}^T \Psi_{12}$$

Quand l'horizon temporel tend vers l'infini, c'est-à-dire $\lim m\frac{N}{k} \to \infty$, l'influence exercée par la condition initiale diminue puisque le système 4.3 est périodique. Il est alors équivalent d'étudier, à la place de la fonction 5.7, la fonction de coût simplifiée, exprimée sur une période (notée J_p), et présentée ci-dessous :

TABLE 5.1 – Gains adaptatifs (L_1 et L_2) et coût J pour l'ensemble des $(2, 10)$-pattern possibles

(m,k)-pattern	L_1	L_2	Coût J	Dégradation $\frac{J_{ideal}-J}{J_{ideal}}$
(1100000000)	$\begin{bmatrix} 134,3256 \\ 7,1250 \end{bmatrix}$	$\begin{bmatrix} 76,2856 \\ 4,8871 \end{bmatrix}$	$1,9808E-04$	$5,29\%$
(1010000000)	$\begin{bmatrix} 122,7504 \\ 6,7256 \end{bmatrix}$	$\begin{bmatrix} 80,9296 \\ 5,0942 \end{bmatrix}$	$1,9698E-04$	$4,71\%$
(1001000000)	$\begin{bmatrix} 113,3148 \\ 6,3860 \end{bmatrix}$	$\begin{bmatrix} 86,0930 \\ 5,3175 \end{bmatrix}$	$1,9617E-04$	$4,28\%$
(1000100000)	$\begin{bmatrix} 105,3446 \\ 6,0894 \end{bmatrix}$	$\begin{bmatrix} 91,8736 \\ 5,5593 \end{bmatrix}$	$1,9573E-04$	$4,05\%$
(1000010000)	$\begin{bmatrix} 98.3922 \\ 5.8224 \end{bmatrix}$	*sans objet*	$1,9559E-04$	$3,97\%$
pas de rejet	$\begin{bmatrix} 128,4591 \\ 6,8222 \end{bmatrix}$	*sans objet*	$1,8812E-04$	*sans objet*

$$J_p = \sum_{i=0}^{m-1} trS_{i+1}R_1 (f_i h) + \sum_{i=0}^{m-1} \bar{J}_i \tag{5.9}$$

Ceci signifie que seul le coût stationnaire est considéré.

5.1.3 Application

Nous avons réalisé les calculs pour le système étudié au chapitre précédent en considérant toutes les contraintes $(2, 10)$-firm. La fonction de coût 5.2, contrairement à la fonction de coût présentée au chapitre précédent, ne prend pas en compte la condition initiale mais intégre le bruit. Par conséquent, il n'y a pas de problème de synchronisation avec le (m,k)-pattern .

Soit un bruit donné par la matrice de covariance suivante :

$$R_c = \begin{bmatrix} 0,005625 & -0,075 \\ -0,075 & 1 \end{bmatrix}$$

Les matrices Q et R considérées sont respectivement :

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.10}$$

$$R = 0,00006 \tag{5.11}$$

Les résultats obtenus pour les pattern $(2, 10)$ sont résumés dans la table 5.1.

Sur cet exemple, nous pouvons remarquer que le $(2, 10)$-pattern uniforme (1000010000) donne le meilleur coût. Les différences de coûts sont très faibles et la dégradation relative du coût par rapport au coût obtenu dans une situation idéale (J_{ideale}) sans rejet d'échantillon reste inférieure à $5,3\%$ pour tous les $(2, 10)$-pattern.

5.2 Calcul d'un (m,k)-pattern optimal

Dans cette section, nous proposons, pour le système sous une contrainte (m,k)-firm donnée (m et k donnés), une approche analytique qui dérive le (m,k)-pattern optimal minimisant la fonction de coût 5.2.

Tout (m,k)-pattern peut être décrit par les variables f_i pour $i \in [0, m-1]$ où chaque variable f_i donne la distance, en terme de nombre de périodes élémentaires h, entre deux mises à jour de la commande. Il s'agit donc de calculer les valeurs des variables f_i pour $i \in [0, m-1]$ qui minimisent une fonction de coût.

Dans un premier temps, nous exprimons la dérivée de la fonction de coût $J_p(f_0, f_1, .., f_{m-1})$ en f_i, pour tout $i \in [0, m-1]$, qui servira lors des calculs des meilleurs f_i pour $i \in [0, m-1]$. Puis, le problème de calcul du (m,k)-pattern optimal est formalisé pour appliquer une technique de programmation entière non-linéaire. Enfin, un algorithme est donné pour trouver la solution optimale.

5.2.1 Dérivée de la fonction de coût

Pour utiliser la méthode d'optimisation développée dans la suite, nous avons besoin de calculer les dérivées, par rapport à f_i ($i \in [0, m-1]$), de la fonction de coût (équation 5.9) suivante :

$$J_p(f_0, f_1, .., f_{m-1}) = \sum_{i=0}^{m-1} (trS_{i+1}(f_0, f_1, .., f_{m-1}) R_1(f_i h)) + \sum_{i=0}^{m-1} \bar{J}_i(f_0, f_1, .., f_{m-1})$$

On note $F = \{f_i\}_{i=0,1..m-1}$. La dérivée au premier ordre de la fonction de coût 5.9 en f_i pour $i \in [0, m-1]$ est alors :

$$\frac{dJ_p(F)}{df_i} = \sum_{i=0}^{m-1} tr \frac{dS_{j+1}(F)}{df_i} R_1(f_i h) + trS_{i+1}(F) \frac{dR_1(f_i h)}{df_i} + \frac{d\bar{J}_i(F)}{df_i} \tag{5.12}$$

Le calcul de cette dérivée fait l'objet de l'annexe D.

5.2.2 Résolution du problème d'optimisation

Le (m,k)-pattern optimal est celui qui minimise la fonction de coût 5.9, ainsi on pose le problème d'optimisation suivant :

Problème 1

$$\begin{array}{rl} \textit{Trouver les valeurs de} & f_0, f_1, \ldots, f_{m-1} \\ \textit{qui minimisent} & J_p(F) \\ \textit{sous les contraintes} & \sum_{i=0}^{m-1} f_i = k \\ & f_i \geq 1 \textit{ et } f_i \epsilon \mathbb{N} \textit{pour } i \in [0, m-1] \\ & m, k \epsilon \mathbb{N}^*, m < k, \textit{ } m, k \textit{ donnés} \end{array}$$

Le problème 1 est un problème de programmation entière non-linéaire. Plusieurs techniques sont disponibles pour le traiter. Parmi les plus courantes, nous pouvons citer : l'algorithme Séparation et Evaluation (*Branch and Bound*, 1985 ; [Nabar91, Borchers94, Stubbs96, Leyffer98]) et certaines variantes comme l'algorithme *LP/NLP based Branch and Bound* [Quesada92], la méthode de décomposition de Benders généralisée (*Generalized Benders Decomposition*) [Geoffrion72], l'algorithme *Outer-Approximation* [Duranand86, Yuan88, Fletcher94] ou la méthode Plant Sécant (*Extended Cutting Plane Method*) [Westerlund95]. Dans cette étude, nous utilisons l'algorithme Séparation et Evaluation pour résoudre le problème 1. Dans la suite, avant de présenter cet algorithme, nous donnons d'abord les conditions de Kuhn Tucker qui y seront utilisées.

5.2.2.1 Conditions de Kuhn Tucker - résolution dans $\mathbb{R}$

En mathématiques, les conditions de Kuhn-Tucker ou conditions de Karush-Kuhn-Tucker [Kuhn51] permettent de résoudre des problèmes d'optimisation sous K contraintes non-linéaires d'inégalité et J contraintes d'égalité.

Considérons le problème non linéaire suivant :

Problème 2

$$\begin{array}{rl} \textit{Déterminer} & X_0 \\ \textit{qui minimise la fonction} & f(X) \\ \textit{sous les contraintes suivant :} & g_i(X_0) \geq 0 \textit{ pour } i = 1, 2.., K \\ \textit{et} & h_j(X_0) = 0 \textit{ pour } j = 1, 2.., J \\ \textit{avec} & f : \mathbb{R}^n \rightarrow \mathbb{R} \textit{ la fonction objectif} \\ & g_i : \mathbb{R}^n \rightarrow \mathbb{R} \\ & h_j : \mathbb{R}^n \rightarrow \mathbb{R} \end{array}$$

Soit u_i et v_j sont des multiplicateurs de Lagrange associés respectivement à la i^{ime} contrainte d'inégalité et à la j^{ime} contrainte d'égalité. Les conditions de Kuhn Tucker sont décrites comme suit :

1. $\frac{df}{dx_l}(X_0) - \sum_{i=1}^{K} u_i \frac{dg_i(X_0)}{dx_l} - \sum_{j=1}^{J} v_j \frac{dh_j(X_0)}{dx_l} = 0$ pour $l = 1, 2.., card(X_0)$
2. $g_i(X_0) \geq 0$ pour $i = 1, 2.., K$
3. $h_j(X_0) = 0$ pour $j = 1, 2.., J$;
4. $u_i g_i(X_0) = 0$ pour $i = 1, 2.., K$
5. $u_i \geq 0$ pour $i = 1, 2.., K$.

Théorème 6 *(Conditions de Kuhn Tucker)*

Supposons que f soit une fonction convexe, que les contraintes d'égalité soient toutes linéaires et que les contraintes d'inégalité soient toutes concaves. Alors, si le triplet (X_0, U_0, V_0) satisfait les conditions de Kuhn Tucker ci-dessus, X_0 est la solution optimale du problème 1.

Par le théorème de Kuhn Tucker, on obtient X_0 minimisant la fonction $f(X)$ en résolvant le système décrit par les conditions précédentes.

Avant d'introduire l'algorithme Séparation et Evaluation, nous donnons un exemple pour illustrer comment utiliser le théorème de Kuhn Tucker sur notre problème d'optimisation. Grâce à cet exemple, nous découvrons aussi un phénomène intéressant : plus les '1' sont distribués uniformément dans le (m,k)-pattern, plus la valeur de la fonction de coût est petite.

Dans cet exemple, nous appliquons directement le théorème de Kuhn Tucker sur le problème d'optimisation 1 sans prendre en compte la contrainte $f_i \epsilon \mathbb{N}$ pour $i \in [0, m-1]$; c'est-à-dire que nous cherchons une solution dans $\mathbb{R}$. Les termes du problème d'optimisation 1 peut être décrit comme suit :

$$\begin{aligned} f(F) &= J_p(F) = \sum_{l=0}^{m-1} (trS_{j+1}(F) R_1(f_l h) + \bar{J}_l(F)) \\ g_1(F) &= f_0 - 1 \\ &\vdots \\ g_m(F) &= f_{m-1} - 1 \\ h_1(F) &= f_0 + f_1 + \ldots + f_{m-1} - k \end{aligned}$$

Les dérivées de la fonction $f(F)$ par rapport à chaque f_i sont :

$$\frac{dJ_p(F)}{df_i} = \sum_{l=0}^{m-1} tr \frac{dS_{j+1}(F)}{df_i} R_1(f_l h) + trS_{i+1}(F) \frac{dR_1(F)}{df_i} + \frac{d\bar{J}_l(F)}{df_i}$$

En construisant les conditions de Kuhn Tucker pour le problème d'optimisation 1, nous avons :

1. $\frac{dJ_p(F)}{df_i} - u_1 \frac{dg_1(F)}{df_i} ... - u_m \frac{dg_m(F)}{df_i} - v_1 \frac{dh_1(F)}{df_i} = 0$ pour $i = 0, 2.., m-1$
2. $f_i - 1 \geq 0$ pour $i = 0, 1.., m-1$;
3. $f_0 + f_1 + \ldots + f_{m-1} - k = 0$
4. $u_i (f_{i-1} - 1) = 0$ pour $i = 1, 2.., m$;
5. $u_i \geq 0$ pour $i = 1, 2.., m$.

Puisque $u_1 \frac{dg_1(F)}{df_i} ... + u_m \frac{dg_{m-1}(F)}{df_i} = u_i$ et $v_1 \frac{dh_1(F)}{df_i} = v_1$, la première condition devient :

$$\frac{dJ_p(F)}{df_i} - u_i - v_1 = 0 \text{ pour } i = 1, 2.., m$$

De D.2, D.3 et D.4 présentés dans l'annexe D, on a

$$\sum_{j=0}^{m-1} tr \frac{dS_{j+1}(F)}{df_i} R_1(f_i h) + tr S_{i+1} h e^{Af_i} R_c e^{A^T f_i} + tr(Q R_1(f_i) h) - u_i - v_1 = 0 \text{ pour } i = 1, 2.., m$$

Si on pose $u_i = 0$ pour $i = 1, 2.., m$, alors les conditions 4 et 5 sont satisfaites. Si on suppose maintenant que $f_i = \frac{k}{m}$ pour $i = 1, 2.., m$, alors les conditions 2 et 3 sont satisfaites. De [Bittanti91], il est montré que pour le système périodique, on a $S_i = S_{i+1}$ lorsque l'horizon temporel est suffisamment long. Ainsi si $f_0 = f_1 \ldots = f_{m-1}$, les termes $\sum_{j=0}^{m-1} tr \frac{dS_{j+1}(F)}{df_i} R_1(f_i h)$, $tr S_{i+1} h e^{Af_i} R_c e^{A^T f_i}$ et $tr(Q R_1(f_i) h)$ sont constants quelque soit i, la première condition est satisfaite. La solution optimale du problème est donc $f_i = \frac{k}{m}$, $u_i = 0$ pour $i = 1, 2.., m$. Dans le cas où f_i ne prend que des valeurs entières, puisque on suppose que $J_p(F)$ est convexe, alors plus proches sont toutes les valeurs de f_i $i = 0, 1, .., m-1$ sont proches, plus la valeur de $J_p(F)$ diminue.

5.2.2.2 Solution optimale dans $\mathbb{N}$ - Algorithme Séparation et Evaluation (*Branch and Bound*)

Cet algorithme est une méthode générique de résolution de problèmes d'optimisation, et plus particulièrement d'optimisation combinatoire ou discrète. Il a été proposé en 1960 par Land et Doig [Land60] pour la résolution du problème du voyageur de commerce (*traveling salesman problem*). Les premiers travaux sur l'algorithme de Séparation et Evaluation aient visé les problèmes linéaires [Dakin65, Garfinkel72, Taha75], mais récemment, il a été montré qu'on pouvait utiliser cet algorithme pour traiter les problèmes non linéaires [Gupta85, Nabar91, Borchers94, Stubbs96, Leyffer98].

Pour résoudre le problème 1, l'algorithme vise à explorer, d'une manière intelligente, l'ensemble des solutions faisables du problème, c'est pourquoi il peut être classé parmi les méthodes d'énumération implicite : toutes les solutions possibles du problème peuvent être énumérées mais l'analyse des propriétés du problème permet d'éviter l'énumération de larges classes de mauvaises solutions.

L'algorithme Séparation et évaluation est constitué de deux phases : la séparation et l'évaluation. La séparation permet d'obtenir une méthode générique pour énumérer toutes les solutions tandis que l'évaluation évite l'énumération systématique de toutes les solutions.

Séparation

La phase de séparation consiste à diviser le problème 1 en un certain nombre de sous-problèmes qui ont chacun leur ensemble de solutions réalisables de telle sorte que tous ces ensembles forment un recouvrement (idéalement une partition) de l'ensemble de toutes les solutions possibles (y compris les solutions infaisables). Ainsi, en résolvant tous les sous-problèmes et en prenant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Ce principe de séparation est appliqué de manière récursive à chacun des sous-ensembles de solutions obtenus, et ceci tant qu'il y a des ensembles contenant plusieurs solutions. Les ensembles de solutions (et leurs sous-problèmes associés) ainsi construits ont une hiérarchie naturelle en arbre, appelée arbre de recherche.

Evaluation

L'évaluation d'un noeud de l'arbre de recherche a pour but de déterminer l'optimum de l'ensemble des solutions réalisables associé au noeud en question ou, au contraire, de prouver que cet ensemble ne contient pas de solution optimale. Lorsqu'un tel noeud est identifié dans l'arbre de recherche, il est donc inutile d'effectuer la séparation de son espace de solutions.

Pour déterminer qu'un ensemble de solutions réalisables ne contient pas de solution optimale, la méthode la plus générale consiste à déterminer une borne inférieure pour le coût des solutions contenues dans l'ensemble. Si on arrive à trouver une borne inférieure de coût supérieur au coût de la meilleure solution trouvée jusqu'à présent, on a alors l'assurance que le sous-ensemble ne contient pas l'optimum. Les techniques pour le calcul de bornes sont basées sur la relaxation de certaines contraintes.

L'algorithme complet est donné dans l'algorithme 2 où :

- $\mathcal{U}$ représente le coût de la meilleure solution faisable courante,
- $\mathcal{H}$ la borne inférieure du coût optimal du $\mathcal{P}$
- l'ensemble $\mathcal{M}$, utilisé pour le calcul de $\mathcal{L}$, représente l'ensemble de noeuds qui ont été évalués et qui contiennent éventuellement la solution optimale,
- $\mathcal{P}'$ est la forme relaxée du problème $\mathcal{P}$ obtenue en relaxant les contraintes sur les variables $X \epsilon \mathbb{N}^n$ par $X \epsilon \mathbb{R}^{+n}$ (en particulier, dans le cas du problème 1, on remplace $f_i = \mathbb{N}$, pour $i \in [1, m-1]$ par $f_i = \mathbb{R}^+$ pour $i \in [0, m-1]$). Les variables sont ainsi consédérées comme des variables continues dans $\mathbb{R}^+$. La solution du problème $\mathcal{P}'$ constitue une borne inférieure du coût optimal de $\mathcal{P}$.

Pour le problème particulier de recherche de pattern optimum, $\mathcal{P}'$ peut être résolu en utilisant les conditions de Kuhn Tucker, ce qui donne :

$$\begin{aligned} \left[\frac{dJ_p(F)}{df_0} \ldots \frac{dJ_p(F)}{df_{m-1}}\right] - \Lambda &= 0 \\ \sum_{i=0}^{m-1} f_i - k &= 0 \\ f_i &\geq 1 \end{aligned} \tag{5.13}$$

Algorithm 2 Séparation et Evaluation

$\mathcal{L} := \{\mathcal{P}\}$;
$\mathcal{M} := \{\mathcal{P}\}$;
$\mathcal{U} := +\infty$;
$\mathcal{H} := +\infty$;
Tant que $\mathcal{L} \neq \emptyset$ ou $\mathcal{U} - \mathcal{H} > \epsilon$ **faire**
  Choisir un noeud $\mathcal{Q}$ de la liste $\mathcal{L}$;
  Enlever le noeud $\mathcal{Q}$ de la liste $\mathcal{L}$;
  Résoudre le problème relaxé $\mathcal{P}'$du problème représenté par $\mathcal{Q}$;
  Si $\mathcal{P}'$admet une solution $x^*(\mathcal{P}')$ qui donne le coût $J^*(\mathcal{P}')$, avec $J^*(\mathcal{P}') < \mathcal{U}$ **alors**
    Borne_inférieure $(\mathcal{Q}) := J^*(\mathcal{P}')$;
    Si $x^*(\mathcal{P}')$ est une solution faisable entière **alors**
      $\mathcal{U} := J^*(\mathcal{P}')$;
      $x^*(\mathcal{P}) := x^*(\mathcal{P}')$;
      $J^*(\mathcal{P}) := J^*(\mathcal{P}')$;
    Sinon
      Séparer le noeud $\mathcal{Q}$ et mettre à jour la liste $\mathcal{L}$ avec ses noeuds enfants ;
    Fin si
    Si tous les noeuds frères de $\mathcal{Q}$ ont été évalués **alors**
      Enlever le noeud père de $\mathcal{Q}$ de $\mathcal{M}$;
      Mettre $\mathcal{Q}$ et ses noeuds frères dans $\mathcal{M}$;
      $\mathcal{H} = \min\{\text{Borne\_inférieure}(\mathcal{Q}),\ \mathcal{Q} \in \mathcal{M}\}$;
    Fin si
  Sinon
    Borne_inférieure $(\mathcal{Q}) := +\infty$;
  Fin si
Fin Tant que

où Λ est un vecteur contenant m multiplicateurs de Lagrange.

L'algorithme commence par résoudre le problème $\mathcal{P}'$. Si la solution $x^*(\mathcal{P}')$ est entière, alors elle est aussi la solution optimale du problème $\mathcal{P}$; sinon, il existe au moins une variable f_j pour $j \in [0, m-1]$, qui prend une valeur réelle. Dans ce denier cas, l'algorithme procède à la procédure *Séparation* et génère ainsi deux sous-problèmes. Sur l'exemple précédent, on obtient alors :

$$\begin{array}{rl} \text{Minimiser} & J_p(F) \\ \text{Sous les contraintes} & \sum_{i=0}^{m-1} f_i = k \\ & f_j \geq \left\lceil f_j^* \right\rceil \\ & f_i \geq 1 \text{ et } f_i \epsilon \mathbb{R}^+ \text{pour } i \in [0, m-1] - \{j\} \end{array}$$

et

$$\begin{array}{rl} \text{Minimiser} & J_p(F) \\ \text{Sous les contraintes} & \sum_{i=0}^{m-1} f_i = k \\ & f_j \leq \left\lfloor f_j^* \right\rfloor \\ & f_i \geq 1 \text{ et } f_i \epsilon \mathbb{R}^+ \text{pour } i \in [0, m-1] - \{j\} \end{array}$$

où f_j^*est un nombre réel de $x^*(\mathcal{P})$.

Les deux sous-problèmes sont ajoutés à la liste $\mathcal{L}$. La procédure est réitérée par la sélection d'un sous-problème $\mathcal{Q}$ de la liste $\mathcal{L}$. Il se peut que le sous-problème $\mathcal{P}'$ n'ait pas de solution faisable. Dans ce cas, $\mathcal{P}'$ est enlevé de la liste $\mathcal{L}$ sans la procédure Séparation. L'algorithme se termine quand la liste $\mathcal{L}$ devient vide ou lorsqu'une solution sous-optimale avec une tolérance prédéfinie (définie par une précison ϵ spécifiée) est trouvée.

5.2.2.3 Application de la méthode Séparation et Evaluation au problème de recherche d'un (m,k)-pattern optimal

Considérons le système étudié dans ce chapitre et décrit par l'équation 4.1 avec $A = \begin{bmatrix} 0 & 1 \\ 0 & -12.6559 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1.9243 \end{bmatrix}$; v_c est un bruit blanc de covariance $R_c = \begin{bmatrix} 0 & 0 \\ 0 & 10^{-5} \end{bmatrix}$. Le système est échantillonné avec la période $h = 0.01s$. Pour diminuer la bande passante requise, la contrainte $(3,11)$-firm est appliquée au système. Pour déterminer le $(3,11)$-pattern optimal, on pose le problème d'optimisation suivant.

Problème (P1)

$$\begin{array}{rl} \text{Calculer les valeurs de} & F = (f_0, f_1, f_2) \\ \text{qui minimisent} & J_p(f_0, f_1, f_2) \\ \text{sous les contraintes} & f_0 + f_1 + f_2 = 11 \\ & f_i \geq 1 \text{ et } f_i \epsilon \mathbb{N}, \text{ pour } i \in \{0,1,2\} \end{array}$$

D'après l'approche proposée dans la section précédente, le problème *P1* est relaxé en remplaçant $f_i \epsilon \mathbb{N}$ par $f_i \epsilon \mathbb{R}^+$ et est résolu ensuite en appliquant les conditions de Kuhn Tucker, ce qui nous donne $f_0 = f_1 = f_2 = 3.67$. Puisque les valeurs prises par les variables f_i ne sont pas entières, l'approche subdivise le problème en deux sous-problèmes :

Problème (P2)

$$\begin{array}{rl} \text{Minimiser} & J_p(f_0, f_1, f_2) \\ \text{Sous les contraintes} & f_0 + f_1 + f_2 = 11 \\ & f_0 \leq 3 \\ & f_i \geq 1 \text{ et } f_i \epsilon \mathbb{R}^+ \text{pour } i \in \{0,1,2\} \end{array}$$

et

Problème (P3)

$$\begin{array}{rl} \text{Minimiser} & J_p\left(f_0, f_1, f_2\right) \\ \text{Sous les contraintes} & f_0 + f_1 + f_2 = 11 \\ & f_0 \geq 4 \\ & f_i \geq 1 \text{ et } f_i \epsilon \mathbb{R}^+ \text{pour } i \in \{0, 1, 2\} \end{array}$$

Cet étape est représentée par l'arbre de recherche illustré dans la figure 5.1.

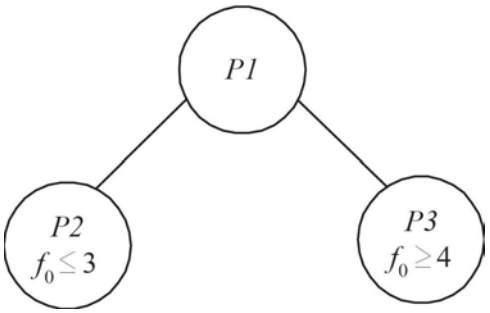

FIGURE 5.1 – L'arbre de recherche du problème $\mathcal{P}1$

La solution optimale du problème (*P1*) est nécessairement la solution de l'un de ces deux sous-problèmes. En appliquant les conditions de Kuhn Tucker aux problèmes (*P2*) et (*P3*), on obtient deux solutions :

$$f_0 \approx 3,\ f_1 = 4.15,\ f_2 = 3.85$$
$$f_0 = 4.03,\ f_1 = 3.32,\ f_3 = 3.65$$

qui correspondent respectivement au coût $J = 0.00103$ et $J = 0.0011$. Comme la deuxième solution donne une performance pire que la première, la valeur de f_0 est fixée à 3 et la branche du problème *P3* est supprimée. L'approche divise ensuite le problème *P2* en deux sous-problèmes :

Problème (P4)

$$\begin{array}{rl} \text{Minimiser} & J_p\left(f_0, f_1, f_2\right) \\ \text{Sous les contraintes} & f_0 + f_1 + f_2 = 11 \\ & f_0 = 3 \\ & f_1 \leq 3 \\ & f_i \geq 1 \text{ et } f_i = \mathbb{R}^+ \text{pour } i \in \{1, 2\} \end{array}$$

et

Problème (P5)

$$\begin{aligned} \text{Minimiser} \quad & J_p\left(f_0, f_1, f_2\right) \\ \text{Sous les contraintes} \quad & f_0 + f_1 + f_2 = 11 \\ & f_0 = 3 \\ & f_1 \geq 4 \\ & f_i \geq 1 \text{ et } f_i = \mathbb{R}^+ \text{pour } i \in \{1, 2\} \end{aligned}$$

Cette étape est représentée par l'arbre de recherche illustré dans la figure 5.2.

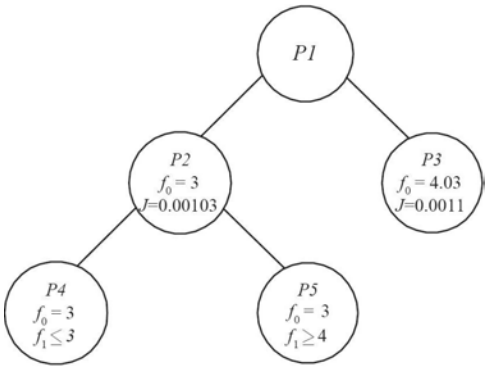

FIGURE 5.2 – L'arbre de recherche du problème $\mathcal{P}$

En appliquant les conditions de Kuhn Tucker aux problèmes *P4* et *P5*, on obtient deux solutions :

$$f_0 \approx 3,\ f_1 = 3.02,\ f_2 = 4.98$$
$$f_0 = 3,\ f_1 \approx 4,\ f_3 \approx 4$$

qui correspondent respectivement au coût $J_p = 0.0012$ et $J_p = 0.0011$. La branche du problème (P4) est ainsi supprimée et la solution optimale du problème original (*P1*) est $f_0 = 3$, $f_1 = 4$ et $f_3 = 4$. C'est-à-dire que le $(3, 11)$- pattern optimal est $\Pi = 10010001000$.

L'évolution de l'état du processus avec un $(3, 11)$-pattern arbitraire $\Pi = 100110000000$ et le $(3, 11)$-pattern optimal $\Pi = 10010001000$ sont illustrées respectivement par les figures 5.3 et 5.4. L'évolution de l'état du processus avec la contrainte (k, k)-firm (le cas idéal, tous les messages sont reçus par le contrôleur) est illustrée par la figure 5.5. Dans ces trois essais, la référence varie à l'instant 0 de la valeur 0 à la valeur 0.1.

En comparant les figures 5.3 et 5.4, on peut remarquer que la variance de la trace est plus faible avec le $(3, 11)$-pattern optimal que celle avec le $(3, 11)$-pattern arbitraire. La comparaison des coûts obtenus avec les deux $(3, 11)$-patterns confirme aussi l'amélioration de la performance. La figure 5.5 montre que dans le cas idéal, la trace est légèrement plus convergente que celle avec le $(3, 11)$-pattern optimal, et le coût est environ 19% de moins que celui avec le $(3, 11)$-pattern optimal. Néanmoins, avec 19% de dégradation de performance, le système sous la contrainte $(3, 11)$-firm diminue considérablement la bande passante requise : il n'utilise que 27% de la bande passante requise par le système sous la contrainte (k, k)-firm.

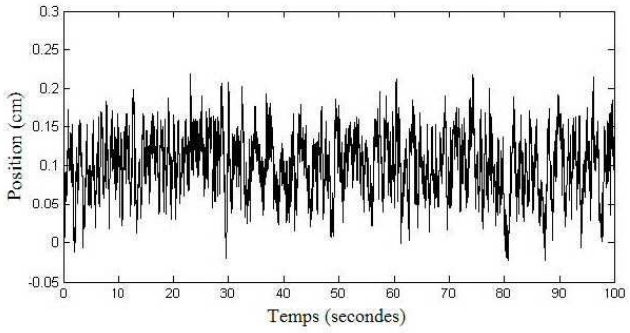

FIGURE 5.3 – Trace de position lorsque la référence varie à l'instant 0 de la valeur 0 à la valeur 0.1 sous la contrainte (3,11)-firm, avec $f_0 = 3$, $f_1 = 1$ et $f_2 = 7$; le coût $J = 15.94$

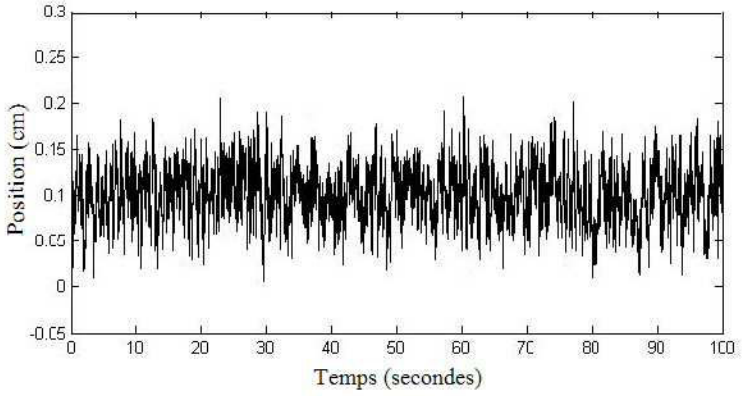

FIGURE 5.4 – Trace de position lorsque la référence varie à l'instant 0 de la valeur 0 à la valeur 0.1 sous la contrainte (3,11)-firm, avec $f_0 = 3$, $f_1 = 4$ et $f_2 = 4$; le coût $J = 12.76$

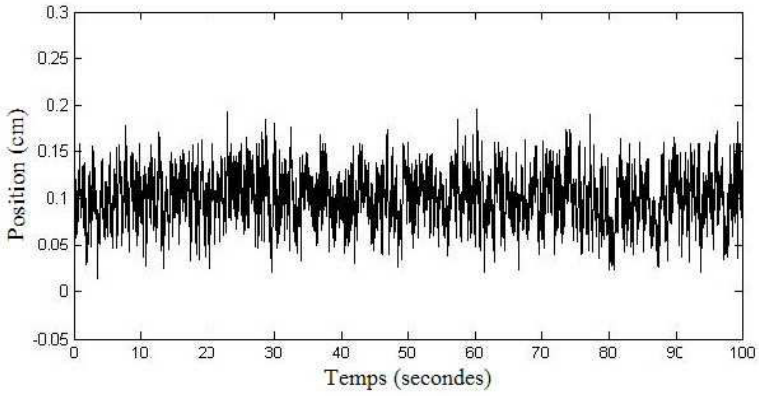

FIGURE 5.5 – Trace de position lorsque la référence varie à l'instant 0 de la valeur 0 à la valeur 0.1 sous la contrainte (k,k)-firm ; le coût $J = 10.7$

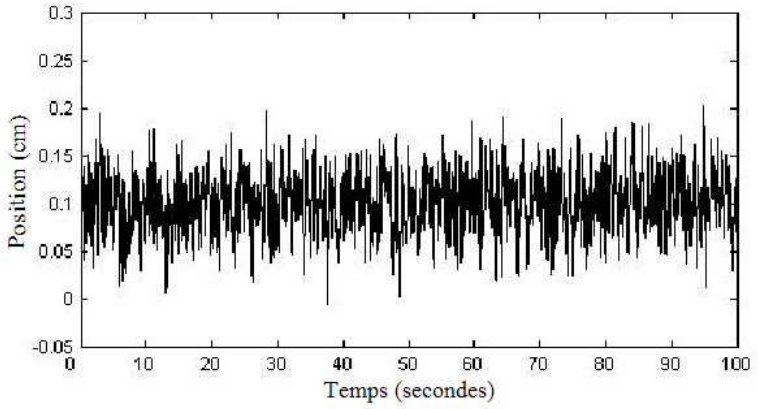

FIGURE 5.6 – Trace de position lorsque la référence varie à l'instant 0 de la valeur 0 à la valeur 0.1 avec la période d'échantillonnage $h = 0.036$; le coût $J = 12.75$

Période d'échantillonnage de base - h Une méthode alternative pour diminuer la bande passante est d'augmenter la période d'échantillonnage h du système. Par exemple, pour avoir la même bande passante que celle induite par la contrainte $(3, 11)$-firm, on peut passer la période de 0.01 seconde à 0.037 seconde. La figure 5.6 illustre la trace du processus avec la période d'échantillonnage 0.037 seconde. En comparant les figures 5.4 et 5.6, on peut remarquer qu'il n'y a pas une différence importante entre elles. Le coût avec la période d'échantillonnage 0.037 seconde est légèrement meilleur que celui avec le $(3, 11)$-pattern optimal, ce qui est évident car la solution optimale dans $\mathbb{R}$ du problème est $f_0 = f_1 = f_2 = 3.67$. Cependant, avec le mécanisme de rejet de messages, les inconvénients de l'allongement de période qu'on a cités au début du mémoire peuvent être évités.

5.2.2.4 Solution sous optimale dans $\mathbb{N}$ - Mots mécaniques

L'algorithme Séparation et Evaluation est très coûteux en temps de calcul. Dans cette sous-section, nous proposons une approche qui nous permet de déterminer un (m,k)-pattern sous-optimal en temps négligeable.

Par les conditions de Kuhn Tucker 5.13, on peut remarquer que la fonction de coût J est minimisé si $f_0 = f_1 .. = f_{m-1} = v$ avec $v \in \mathbb{R}^+$. C'est-à-dire que les rejets de messages sont répartis de manière absolument uniforme dans la séquence de messages. Bien que ce ne soit pas toujours possible car les variables f_i ne peuvent prendre que les valeurs entières, sous l'hypothèse que la fonction de coût $J_p\,(f_0, f_1.., f_{m-1})$ est convexe ou concave, on peut obtenir une performance sous-optimale si les variables f_i prennent des valeurs entières à proximité de v. Autrement dit, la performance est sous-optimale avec le (m,k)-pattern dont les lettres 0 sont réparties uniformément dans le mot.

Par des études sur la théorie des mots menées dans le cadre du stage de DEA, nous remarquons que les lettres 0 sont réparties uniformément et périodiquement dans le mot binaire nommé mot mécanique [Lothaire02]. Ainsi, le mot mécanique peut être utilisé pour

définir le (m,k)-pattern sous-optimal. Dans la suite, avant que nous donnions la définition du mot mécanique, présentons d'abord deux notations :

Soit w un mot binaire, la première lettre de w est notée $w(0)$ et on note par $w(n-1)$ la n^{ime} lettre de w. Si w est périodique de période T avec $T \in \mathbb{N}$, alors on a $w(i) = w(i+T)$ pour $i \geq 0$.

La pente d'un mot binaire w est définie par :

$$\alpha(w) = \frac{|w|_1}{|w|}$$

où $|w|_1$ est le nombre de 1 de w et $|w|$ sa longueur.

La définition du mot mécanique est donnée dans [Lothaire02] comme suit :

Définition 2 *La n^{ime} lettre du mot mécanique de pente α est calculée par :*

$$\lceil (n+1)\alpha \rceil - \lceil n\alpha \rceil \ \forall n \geq 0 \tag{5.14}$$

Le mot mécanique a la propriété suivante[Lothaire02] :

Lemma 3 *Soit w le mot mécanique supérieur de pente α. Si α est rationnel ($\alpha = \frac{p}{q}$), alors le mot est périodique de période q.*

Par définition, un mot mécanique de pente $\frac{m}{k}$ et de longeur k contient exactement m lettres 1, D'après le lemme 3, lorsque la pente d'un mot mécanique $\frac{m}{k}$ est rationnelle, ce mot est périodique de période k. En l'occurrence, il suffit d'utiliser la séquence de lettres dans la première période du mot mécanique de pente $\alpha = \frac{m}{k}$ pour définir le (m,k)-pattern.

Exemple 1 *Etant données les contrainte $(3,5)$-firm, $(3,10)$-firm, et $(7,11)$-firm, par 5.14, on obtient, pour la contrainte $(3,5)$, le $(3,5)$-pattern $\Pi = 11010$ car :*

$$\Pi(0) = \lceil (0+1)\tfrac{3}{5} \rceil - \lceil 0\tfrac{3}{5} \rceil = 1 \qquad \Pi(1) = \lceil (1+1)\tfrac{3}{5} \rceil - \lceil 1\tfrac{3}{5} \rceil = 1$$
$$\Pi(2) = \lceil (2+1)\tfrac{3}{5} \rceil - \lceil 2\tfrac{3}{5} \rceil = 0 \qquad \Pi(3) = \lceil (3+1)\tfrac{3}{5} \rceil - \lceil 3\tfrac{3}{5} \rceil = 1$$
$$\Pi(4) = \lceil (4+1)\tfrac{3}{5} \rceil - \lceil 4\tfrac{3}{5} \rceil = 0$$

De la même façon, on obtient le $(3,10)$-pattern $\Pi = 1001001000$ et le $(7,11)$-pattern $\Pi = 110101101010$.

5.2.2.5 Conclusion sur les méthodes

En utilisant l'algorithme Séparation et Evaluation, on arrive à déterminer le (m,k)-pattern optimisant la performance du système. L'inconvénient de cet algorithme est qu'il est très coûteux en temps de calcul, d'autant plus que l'algorithme calcule répétitivement les équations de Riccati et de Lyapunov qui sont aussi coûteuses en temps. En l'occurrence, l'approche proposée ci-dessus ne permet pas d'adapter dynamiquement le (m,k)-pattern en ligne en fonction du changement de la contrainte (m,k)-firm. Le mot mécanique permet de déterminer le (m,k)-pattern sous-optimal en temps négligeable, permettant ainsi d'adapter dynamiquement le (m,k)-pattern en ligne.

5.3 Conclusions

Dans ce chapitre, nous avons proposé dans un premier temps la méthode pour calculer un contrôleur optimal sous une politique spécifiée (contrainte (m,k)-firm et (m,k)-pattern) de rejets systématique d'échantillons. Puis, sachant que l'on peut disposer d'un tel contrôleur, exprimé sous la forme d'un gain différent pour chaque longueur de séquence de rejets d'échantillons, nous proposons deux méthodes pour, connaissant m et k, définir un (m,k)-pattern optimal. La première approche repose sur l'algorithme Séparation et Evaluation adapté à la recherche de solutions en nombres entiers. Pour diminuer la complexité du calcul, une solution basée sur la théorie des mots pour dériver le (m,k)-pattern sous-optimal est ensuite présentée.

Chapitre 6

Gestion de surcharge du processeur dans des applications de contrôle-commande centralisées

Dans ce chapitre, nous nous attachons au déploiement des applications qui intégrent un ensemble de systèmes à contrôler et pour lesquels, les contrôleurs sont implantés sur le même calculateur ; dans ce contexte, nous traitons le problème de l'ordonnancement de l'ensemble de tâches chargées de réaliser les algorithmes de contrôle des systèmes, considérés comme indépendants, dans une implémentation centralisée (partage d'un processeur). Nous considérons que l'ensemble et la configuration des tâches de contrôle peuvent varier dans le temps en fonction des changements de modes de marche, décidés, par exemple, par un algorithme de supervision. En particulier, un changement de mode de marche peut se concrétiser par :

- l'activation de nouvelles tâches,
- l'arrêt de certaines tâches,
- la modification du pire temps d'exécution (WCET) et / ou de la période d'activation de certaines tâches (résultant, par exemple, de la modification de l'algorithme de contrôle exécuté par la tâche).

Nous proposons une architecture d'ordonnancement pour la manupulation de tels ensembles de tâches. À chaque changement de mode, en tenant compte du nouvel ensemble de tâches et des nouveaux paramètres de configuration des tâches, un *régulateur* de tâches, que nous spécifions, détermine en-ligne une stratégie qui repose sur la contrainte (m,k)-firm. Cette stratégie vise à rejeter sélectivement les instances des tâches afin que la configuration soit ordonnançable, sous la politique d'ordonnancement choisie, et que la performance de contrôle de l'application soit maintenue à niveau global optimal.

6.1 Introduction

Considérons une application de contrôle-commande évolutive constituée d'un ensemble de boucles de contrôle. Le processus, dans chaque boucle de contrôle, est contrôlé par un contrôleur dédié qui est implémenté sous la forme d'une tâche temps réel responsable d'effectuer le calcul de la loi de commande. Les caractéristiques du système ainsi que du contrôleur choisi imposent des contraintes sur les durées limites des périodes d'échantillonnage et sur la "régularité" de cet échantillonnage dans le temps. De plus, l'implantation centralisée de tous les contrôleurs sur une ressource de calcul limitée pose le problème d'ordonnançabilité des tâches. L'étude de l'ordonnançabilité de tâches de contrôle considérées comme des tâches temps réel dur (en particulier, chaque instance de chaque tâche doit respecter l'échéance imposée sur sa fin d'exécution) conduit généralement à surdimensionner les ressources nécessaires, et, donc, à une performance non-optimale, au sens du déploiement, due à l'usage inefficace de la ressource. Ce problème est encore plus important dans le cas des applications étudiées dans ce chapitre pour lesquelles le nombre de tâches et les caractéristiques des tâches sont variables d'un mode de marche à l'autre. De plus, la relation performances/ordonnancement d'une loi de commande est mal connue, surtout quantitativement, et l'affectation des paramètres d'ordonnancement repose généralement sur des résultats non exhaustifs et non généralisables d'expériences et de simulations [Simon05a]. Ainsi l'ordonnancement faisable produit par la solution classique en relaxant les contraintes temporelles conduit vraisemblablement à la performance non optimale de l'application.

En pratique, beaucoup de systèmes de contrôle-commande peuvent être conçus de manière à rester robustes sous la variation des paramètres de la tâche implémentant le contrôleur tels que le temps d'exécution, la période d'échantillonnage etc. De plus, il est, pour certains systèmes (voir chapitre précédent), possible de compenser en ligne ces variations en recalculant, par exemple, les paramètres du contrôleur. Suite à ces constats, nous proposons d'implanter une approche d'ordonnancement adaptative qui utilise les informations courantes de l'application pour ajuster correctement les paramètres d'ordonnancement (les périodes par exemple) des tâches afin de réaliser une meilleure performance globale dans la limite de la ressource disponible.

La conception optimale d'une telle approche d'ordonnancement adaptatif repose sur la conception conjointe des contrôleurs et de l'ordonnancement (voir chapitre 1). Parmi les travaux précédents, citons par exemple les approches représentatives proposées dans [Eker00, Cervin02] qui consistent à ajuster la période des tâches au moment du changement de configuration de l'application afin que la performance globale soit maintenue optimale et l'ordonnançabilité des tâches soit respectée. Cependant, ainsi que nous l'avons vu dans 1.3.3, l'implantation de ces approches basées sur le réglage de périodes est parfois problématique. Dans la suite des études menées aux chapitres précédents, nous étudions ici l'applicabilité du modèle de contraintes (m,k)-firm pour traiter ce problème. L'objectif de l'étude menée dans ce chapitre est d'implanter une approche adaptative qui, au moment du changement de

configuration de l'application (changement de mode de marche), ajuste les contraintes (m,k)-firm des tâches de contrôle afin que la performance de contrôle globale de l'application soit maintenue à un niveau optimisé et que l'ordonnançabilité des tâches soit garantie.

Ce chapitre est organisé comme suit. L'architecture du système considéré dans cette étude est décrite dans la section 2. La section 3 traite de l'ordonnançabilité d'un ensemble de tâches sous contraintes (m,k)-firm. La description formelle du problème traité est présentée dans la section 4. La section 5 donne l'algorithme heuristique pour calculer la contrainte (m,k)-firm pour chaque tâche. Un exemple numérique de l'approche d'ordonnancement proposée est présenté dans la section 6. Enfin, la conclusion et les perspectives sont données dans la section 7.

6.2 Architecture de système

Dans cette étude, nous considérons une architecture de système qui est illustrée par la figure 6.1. Les contrôleurs qui commandent chacun un processus sont implémentés sous forme de tâches temps réel que nous appelons tâches de contrôle. Un processeur unique est partagé par toutes les tâches de contrôle. Nous supposons que la fonction de supervision est implantée sur un processeur dédié. Le rôle de cette fonction est de détecter les instants où :

- certains processus doivent être contrôlés d'une manière différente ce qui implique un changement d'algorithme de contrôle et, donc, au niveau d'une tâche, par exemple, un pire temps d'exécution différent ou une période différente ;
- un processus n'a plus besoin d'être contrôlé ;
- un processus devient actif et nécessite ainsi l'activation du contrôleur associé.

Grâce à cette fonction de supervision, il est possible d'avoir, à ces instants de changement de mode de marche, la connaissance de la configuration courante de l'application en termes de nombre et de paramètres des tâches actives. Au moment d'un changement de la configuration, une fonction particulière, appelée *régulateur* des tâches, est activée et reçoit, de la fonction de supervision, les informations sur la nouvelle configuration de tâches. A partir de ces informations, le régulateur des tâches détermine l'ensemble des paramètres d'ordonnancement sous contraintes (m,k)-firm (identification du paramètre m et du (m,k)-pattern) de chaque tâche pour la nouvelle configuration et les transmet à l'ordonnanceur. Ce dernier applique alors la stratégie transmise (rejets de certaines instances de tâches).

6.3 Ordonnançabilité des tâches sous contraintes (m,k)-firm

Le (m,k)-pattern optimisant la performance de contrôle n'est pas nécessairement optimal dans le sens que les instances de la tâche peuvent ne pas être ordonnançable. En effet, il est prouvé dans [Koren96, Quan00] que le problème de déterminer s'il existe un (m,k)-pattern pour chaque tâche sous contrainte (m,k)-firm tel que l'ensemble des tâches soit ordonnançable, ainsi que le problème de déterminer l'ordonnançabilité d'un ensemble des tâches sous

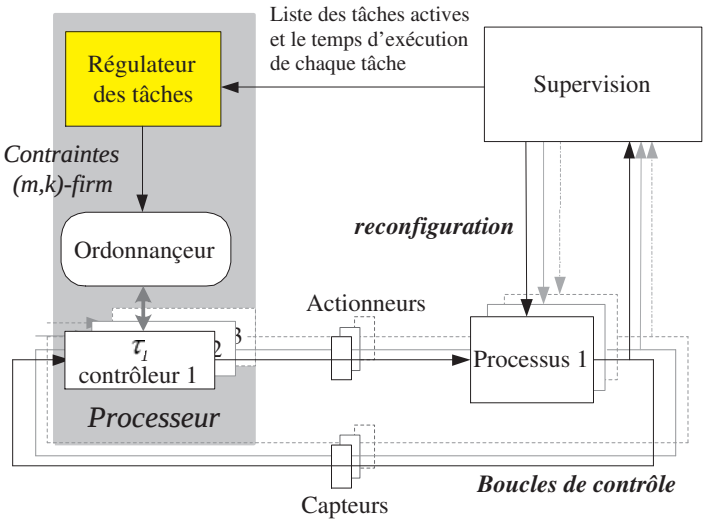

FIGURE 6.1 – L'architecture globale du système

contraintes (m,k)-firm étant donné les (m,k)-pattern sont NP-difficiles que ce soit sous l'algorithme d'ordonnancement à priorités dynamiques ou sous l'algorithme d'ordonnancement à priorités statiques.Mais lorsque le calcul du (m,k)-pattern de chaque tâche s'effectue selon une stratégie explicite, il est parfois possible de déterminer l'ordonnançabilité des tâches. C'est le cas lorsque les (m,k)-patterns sont déterminés par les mots mécaniques. De plus, dans le chapitre 5, nous avons montré que la performance de contrôle du système de contrôle-commande est sous-optimale lorsque le (m,k)-pattern est un mot mécanique. En raison de ces observations, nous adoptons, dans le cadre du partage de processeur, la méthodologie proposée dans le paragraphe 5.2.2.3 du chapitre 5pour définir le (m,k)-pattern d'une tâche sous contrainte (m,k)-firm.

Dans la suite, nous considérons que les tâches de contrôle partageant le processeur sont ordonnancées selon la politique d'ordonnancement préemptif à priorités fixes. Une conditon d'ordonnançabilité sous une telle politique d'ordonnancement sera donnée.

Si les (m,k)-patterns des tâches sont des mots mécaniques, alors l'ordonnançabilité des tâches peut être déterminée explicitement selon le théorème suivant :

Théorème 7 *[Ramanathan99, Jia05a] Soit un ensemble de tâches τ_1, τ_2... τ_n, tel que la tâche τ_i est plus prioritaire que la tâche τ_j si $i < j$. Chaque tâche τ_i est décrite par h_i sa période d'échantillonnage considérée sans rejet (période de base), C_i, son pire temps d'exécution et (m_i, k_i) la contrainte (m,k)-firm qui lui est appliquée. On définit les termes suivants :*

$$R_{ij} = \left\{ \left\lfloor l\frac{k_j}{m_j} \right\rfloor h_j \mid \forall l \in \mathbb{N} \text{ et tel que } \left\lfloor l\frac{k_j}{m_j} \right\rfloor h_j < h_i \right\}$$

$$R_i = \bigcup_{j=1}^{i-1} R_{ij}$$

$$
\begin{aligned}
n_j(t) &= \left\lceil \frac{m_j}{k_j} \left\lceil \frac{t}{h_j} \right\rceil \right\rceil \\
W_i(t) &= C_i + \sum_{j=1}^{i-1} n_j(t) C_j
\end{aligned}
$$

Si pour tout $1 \leq i \leq n$, $\min_{r \in R_i} \frac{W(r)}{r} \leq 1$, *alors la contrainte* (m,k) *peut être respectée pour toutes les tâches.*

Noter que la condition ci-dessus est nécessaire et suffisante lorsque toutes les tâches sont synchrones (c'est-à-dire que elles sont démarrées en même temps). Si les tâches ne sont pas synchrones, la condition devient suffisante.

Malheureusement, cette condition d'ordonnançabilité ne peut pas être utilisée directement dans le routine d'optimisation que nous développons dans la suite à cause du non-déterminisme sur le temps t dans le terme $\min_{r \in R_i} \frac{W(r)}{r} \leq 1$. Pour pallier ce problème de non-déterminisme, nous donnons ici une condition suffisante.

Par la suite, les instances correspondant aux lettres 1 du (m,k)-pattern seront appelées instances critiques, celles correspondant aux lettres 0 seront appelées instances optionnelles.

Théorème 8 *[Ramanathan99, Jia05a] Soit un ensemble de tâches* $\tau_1, \tau_2 \ldots \tau_n$, *tel que la tâche* τ_i *est plus prioritaire que la tâche* τ_j *si* $i < j$. *On pose*

$$
n_{ij} = \left\lceil \frac{m_j}{k_j} \left\lceil \frac{h_i}{h_j} \right\rceil \right\rceil \tag{6.1}
$$

Si pour tout $1 \leq i \leq n$, $C_i + \sum_{j=1}^{i-1} n_{ij} C_j \leq h_i$, *alors la contrainte* (m,k) *peut être respectée pour toutes les tâches.*

Démonstration : Supposons que les tâches sont toutes démarrées à l'instant 0 (c'est-à-dire que les tâches sont synchrones). D'après le lemme 2, le nombre d'instances critiques dans l'intervalle de temps $[0,t]$ est maximal pour t quelconque. La première instance de la tâche τ_i pour $1 \leq i \leq n$ subit ainsi la plus grande interférence d'exécution engendrée par les tâches plus prioritaires. Alors si la première instance de chaque tâche respecte son échéance, toutes les autres instances des tâches respecteront leurs échéances et les tâches seront ordonnançables qu'elles soient synchrones ou pas.

Par le lemme 1, le nombre d'instances critiques de la tâche τ_i est donné par l'équation 6.1. Le terme $\sum_{j=1}^{i-1} n_{ij} C_j$ donne la quantité de travail engendré par les tâches plus prioritaires que la tâche τ_i avant l'instant h_i. Ainsi si l'inéquation $C_i + \sum_{j=1}^{i-1} n_{ij} C_j \leq h_i$ est vérifiée, alors le travail engendré par les tâches plus prioritaires ainsi que celui engendré par la tâche τ_i dans l'intervalle de temps $[0, h_i]$ peuvent être terminés avant l'instant h_i, et la première instance de la tâche τ_i respecte son échéance. Si pour tout $1 \leq i \leq n$, on a $C_i + \sum_{j=1}^{i-1} n_{ij} C_j \leq h_i$, alors la première instance de chaque tâche respecte son échéance, et la contrainte (m,k)peut être respectée par toutes les tâches.

□

Noter que cette condition, qui est suffisante dans le cas général, devient nécessaire et suffisante si la période de la tâche τ_i pour $1 \leq i \leq n$ est multiple de la période de la tâche τ_j pour tout $1 \leq j < i$ dont la période est plus petite que celle de τ_i.

6.4 Architecture d'ordonnancement - Formulation du problème

Nous cherchons à implanter un régulateur qui, au moment du changement de configuration, calcule les contraintes (m,k)-firm des tâches de contrôle de sorte que la performance de contrôle globale de l'application soit maintenue à un niveau optimisé tout en garantissant l'ordonnançabilité des tâches de contrôle.

En utilisant les résultats qu'on a obtenus dans le chapitre 4, nous supposons que le paramètre k_i de chaque tâche τ_i a été déterminé pour assurer la stabilité du système dans le cas d'un pattern $(1, k_i)$ et qu'il reste constant pour chaque mode d'exécution de la tâche. Le paramètre m_i est choisi en ligne dans l'intervalle $[1..k_i]$. En particulier, pour chaque tâche de contrôle τ_i, chaque valeur possible de m_i, $m_{i,j}$, est associée à un niveau de performance de contrôle, noté $v_{i,j}$, de la boucle de contrôle correspondante. Nous supposons que plus la valeur de $v_{i,j}$ est grande, meilleure est la performance du contrôle. On note, néanmoins, que l'augmentation de la valeur de m_i n'améliore pas nécessairement la performance de contrôle. Nous supposons, de plus, que seules les valeurs de m_i qui donnent une performance supérieure à un seuil donné sont considérées.

Considérons maintenant que m_i peut prendre l_i valeurs différentes, notées $m_{i,j}$ pour $j \in [1..l_i]$ avec $l_i \leq k_i$. Les valeurs de $m_{i,j}$ sont ordonnées ainsi : $m_{i,j'}$ avant $m_{i,j"}$ $(j^{'} < j^{''})$ si $v_{i,j'} < v_{i,j''}$. L'objectif du régulateur des tâches est de trouver, pour toutes les tâches τ_i, $1 \leq i \leq n$, une valeur $j_i \in [1..l_i]$, c'est-à-dire m_{i,j_i} afin que $\sum_{i=1}^{i=n} v_{i,j_i} \geq \sum_{i=1}^{i=n} v_{i,k_i}$ | $\forall k_i \epsilon [1..l_i]$, $k_i \neq k_{j_i}$ et que la condition d'ordonnançabilité énoncée par le théorème 7 soit vérifiée, ce qui peut être formulé par le problème d'optimisation suivant :

Problème 3 *Déterminer l'ensemble* $\{x_{i,1}, x_{i,2}, .., x_{i,l_i}\}$ *pour chaque tâche* τ_i, $i = 1, .., n$ *qui maximise*

$$\sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{i,j} v_{i,j} \tag{6.2}$$

avec $x_{i,j} \in \{0, 1\}$, $\sum_{j=1}^{l_i} x_{i,j} = 1$, $i = 1, .., n$, $j = 1, .., l_i$
sous la contrainte :

$$C_i + \sum_{j=1}^{i-1} \left\lceil \frac{\sum_{p=1}^{l_i} x_{jp} m_{jp}}{k_j} \left\lceil \frac{T_i}{T_j} \right\rceil \right\rceil C_j \leq T_i \quad i = 1, .., n \tag{6.3}$$

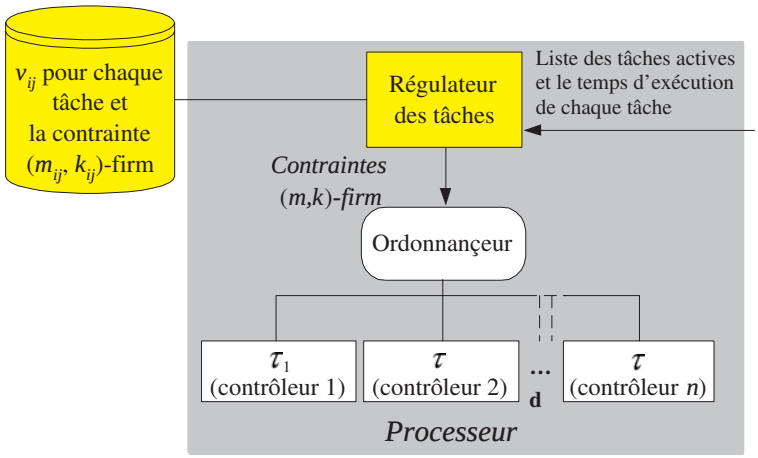

FIGURE 6.2 – L'architecture d'ordonnancement

Si la performance de contrôle est représentée par le coût LQ (linéaire quadratique), alors plus ce coût est petit, meilleure est la performance de contrôle. Le problème d'optimisation est donc un problème de minimisation. Néanmoins, ce dernier peut être facilement transformé en problème de maximisation 3 en remplaçant le coût par sa valeur opposée.

L'architecture d'ordonnancement est illustrée par la figure 6.2 qui détaille la figure 6.1. A un changement de configuration de l'application, le régulateur des tâches reçoit les informations sur le temps d'exécution courant C_i de chaque tâche τ_i et calcule la nouvelle contrainte (m,k)-firm pour chaque tâche en résolvant le problème d'optimisation 3. Puis les tâches sont ordonnancées par l'ordonnançeur selon les contraintes (m,k)-firm définies auparavant.

Le niveau de performance de contrôle $v_{i,j}$ correspondant à une contrainte $(m_{i,j},k_i)$-firm de la tâche τ_i peut être déterminé en ligne ou hors ligne. Mais souvent le temps du calcul en ligne de la performance de contrôle est très important pour des critères de performance usuels (par exemple, le coût LQ décrit par la fonction 5.9) relativement au temps d'exécution de la tâche de contrôle elle-même. En conséquence, les $v_{i,j}$ sont calculés hors ligne dans notre approche et conservés dans une table qui est consultée par le régulateur des tâches au moment de la résolution du problème d'optimisation 3, c'est-à-dire, à chaque changement de mode de marche.

6.5 Résolution du problème d'optimisation

Dans cette section, nous allons d'abord montrer que le problème d'optimisation 3 est un problème NP-difficile, puis, en nous basant sur l'algorithme proposé dans [Khan02], nous proposerons un algorithme heuristique de complexité acceptable pour une résolution approchée, en ligne, du problème.

Le problème d'optimisation 3 relève du problème du "Sac-à-dos Multiple-Choix Multidi-

mensionnel" (noté MMKP pour *Multiple-Choice Multi-Dimension Knapsack Problem*) qui est décrit ci-dessous.

Supposons que il y a n groupes d'articles. Le groupe i contient l_i articles. L'article j du groupe i est associé d'une valeur $v_{i,j}$, et nécessite les ressources représentées par le vecteur $r_{i,j} = (r_{i,j_1}, r_{i,j_2}, .., r_{i,j_m})$. La quantité des ressources disponibles est donnée par $R = (R_1, R_2, .., R_m)$. Le MMKP revient à sélectionner exactement un article de chaque groupe de telle manière que la valeur totale des articles sélectionnés soit maximisée sous contrainte des ressources disponibles. Mathématiquement, le MMKP est exprimé comme suit :

$$\begin{aligned} \text{Maximiser} \quad & \sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{i,j} v_{i,j} \\ \text{Sous contraintes} \quad & \sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{i,j} r_{i,j} \leq R_k,\ k = 1, .., m \\ & \sum_{j=1}^{l_i} x_{i,j} = 1,\ i = 1, .., n,\ j = 1, .., l_i \\ \text{Avec} \quad & x_{i,j} \in \{0, 1\} \end{aligned}$$

Le MMKP est l'une des variantes les plus difficiles à résoudre du problème du sac à dos en variables binaires (noté généralement 01KP) [Pisinger95, Martello90]. En effet, le problème a été prouvé NP-difficile au sens fort [Parra-Hernandez05]. La plupart des algorithmes proposés dans la littérature, par exemple celui basé sur la technique de séparation et évaluation avec la programmation linéaire [Khan98] pour dériver la solution optimale, et l'algorithme heuristique proposé dans [Parra-Hernandez05] pour trouver la solution approchée, ont une complexité temporelle très importante et sont ainsi inappropriés pour les applications à pouvoir de décision en temps réel.

Heureusement, dans [Khan02], l'auteur a développé un algorithme heuristique nommé HEU capable de dériver la solution sous-optimale en temps acceptable dans le contexte d'application temps réel. Pour notre problème d'optimisation, l'algorithme HEU peut être décit comme suit.

- La première phase consiste à trouver une solution faisable, c'est à dire à sélectionner une valeur $m_{i,j}$ pour chaque tâche τ_i tout en satisfaisant la contrainte d'ordonnançabilité 6.3.
- La deuxième phase sert à améliorer itérativement la solution courante en remplaçant, pour chaque tâche τ_i, le $m_{i,j}$ initialement choisi par $m_{i,j'}$ correspondant à la meilleure performance $v_{i,j'}$ en maintenant la condition d'ordonnançabilité 6.3 vérifiée.
- Dans le cas où la solution courante ne peut plus être améliorée à cause de la violation de la condition d'ordonnançabilité 6.3, l'algorithme passe à sa troisième phase qui engage une procédure itérative pour améliorer la solution courante. La procédure remplace d'abord $m_{i,j}$ choisi d'une certaine tâche τ_i par $m_{i,j'}$ avec $j < j^{'}$ (ce qui rend les tâches non-ordonnançables), puis remplace $m_{i',j}$ choisi de la tâche $\tau_{i'}$ pour $i \neq i^{'}$ par $m_{i'j'}$ avec $j > j^{'}$(qui correspond à une pire performance $v_{i',j'}$). Si la solution obtenue est

faisable, l'algorithme revient à la deuxième phase avec la solution courante et répète les opérations mentionnées ci-dessus, sinon, la solution courante est retournée comme la solution finale.

Dans [Khan02], l'exactitude et l'efficacité de l'algorithme sont justifiées par une série d'expérimentations : la valeur de la solution heuristique dépasse en moyenne 94% de la valeur de la solution optimale ; le temps de calcul sur un processeur Pentium 3 à 700 MHz est moins de 5 millisecondes pour une configuration de 10 ressources et de 50 groupes possédant chacun 10 articles, et de moins de une seconde si le nombre de groupes ne dépasse pas 350. Ces caractéristiques de l'algorithme heuristique rendent possible l'implémentation du régulateur des tâches pour un nombre limité de tâches. Le régulateur appelle, à chaque changement de mode de marche, en ligne, la routine d'optimisation réalisant cet algorithme pour dériver les nouvelles contraintes (m,k)-firm selon la configuration courante de l'application afin de maintenir une bonne performance de l'application dans la limite de la ressource disponible.

Malgré les intérêts de l'algorithme HEU, mentionnés ci-dessus, celui-ci nécessite d'être adapté à notre modèle afin d'avoir un bon fonctionnement. Dans notre configuration de problème, les valeurs $m_{i,j}$ et $v_{i,j}$ pour $j \in [1..l_i]$ sont rangées en ordre croissant pour chaque tâche τ_i. En conséquence, contrairement à l'algorithme HEU qui sélectionne avant tout une solution faisable en testant successivement un certain nombre de solutions susceptibles d'être faisables, l'algorithme HEU modifié choisit la plus petite valeur $m_{i,0}$ pour chaque tâche τ_i (si cette solution n'est pas faisable, il n'existe donc pas de solution faisable pour le problème d'optimisation 3). De plus, l'algorithme HEU cherche à trouver, après sa première phase, une meilleure solution nécessitant moins de ressource que la précédente ; cependant, une telle solution n'existe pas dans notre configuration de problème car l'augmentation de la valeur de m_i augmente nécessairement la quantité de la ressource nécessaire. Cette observation nous permet de supprimer une procédure de recherche non-profitable de l'algorithme HEU. L'algorithme HEU modifié est donné par l'algorithme 3. Le remplacement de $m_{i,j}$ par $m_{i,j'}$ est nommé un échange. Un échange conduisant à une meilleure performance représente une amélioration, tandis qu'un échange dégradant la performance est appelé une dégradation. Un échange est dit faisable si la solution après l'échange est faisable, sinon il est appelé infaisable.

Evaluation de la complexité de l'algorithme HEU modifié.

Pour simplifier l'analyse de complexité, nous supposons que $l_1 = l_2 ... = l_n$. Dans la première phase de l'algorithme, la solution est directement choisi comme $\rho = (1, 1, .., 1)$, ce qui permet de réduire la complexité temporelle de la première phase de l'algorithme original à $O\left(n^2\right)$ contre $O\left(n^2 (l-1)^2 (n-1)\right)$. Dans les autres phases, les modifications que nous avons faites ne changent pas la complexité temporelle dans le pire cas qui reste à $O\left(n^2 (l-1)^2 (n-1)\right)$. Comme la phase 3 consiste à retourner à la phase 2 une solution temporaire lorsque celle-ci est faisable, la complexité globale des phases 1 et 2 donne ainsi la complexité de l'algorithme qui est donc $O\left(n^2 (l-1)^2 (n-1)\right)$.

Algorithm 3 Algorithme de détermination des contraintes (m,k)-firm des tâches

Symboles et formalisation :

n : le nombre des tâches

m_i et k_i : la contrainte (m_i, k_i)-firm de la tâche τ_i

m_{ij} : le j^{ime} valeur possible de m_i

l_i : le nombre des valeurs possibles de m_i

T_i : la période de la tâche τ_i

C_i : le temps d'exécution de la tâche τ_i

$\rho = (\rho_1, \rho_2, .., \rho_n)$ le vecteur de solution représentant la solution courante, où ρ_i donne l'indice j de la valeur choisie de m_i pour la tâche τ_i avec $j \in [1..l_i]$

$\rho|Z$: le vecteur de solution après l'échange Z de ρ

L : le vecteur de ressource qui donne la quantité de la ressource nécessaire courante.

$U(\rho)$: la performance de contrôle globale de l'application obtenu avec la solution ρ, c'est-à-dire $U(\rho) = \sum_{i=1}^{n} v_{i\rho_i}$

$X = (i,j)$: représente un échange où m_{ij} est choisi comme la valeur de m_i à la place de $m_{i\rho_i}$

Phase 1 : initialisation

Pose $\rho_i = 1$ pour tout $i \in [1..n]$

Calculer $L = \begin{pmatrix} C_1 \\ C_2 + \sum_{j=1}^{1} \left\lceil \frac{m_{j1}}{k_j} \left\lceil \frac{h_2}{h_j} \right\rceil \right\rceil C_j \\ \vdots \\ C_n + \sum_{j=1}^{n-1} \left\lceil \frac{m_{j1}}{k_j} \left\lceil \frac{h_n}{h_j} \right\rceil \right\rceil C_j \end{pmatrix}$

Chercher α tel que $\frac{L_\alpha}{h_\alpha} = \max_{i=1,2,..,n} \frac{L_i}{h_i}$

Si $\frac{L_\alpha}{h_\alpha} \leq 1$, passer à la **Phase 2**, sinon "pas de solution faisable" - ***Fin***

Phase 2 : améliorer itérativement la solution avec les améliorations faisables

Poser $\triangle\alpha(\rho,i,j) = \frac{\sum_{s=i+1}^{n} \left(\left\lceil \frac{m_{i\rho_i}}{k_i} \left\lceil \frac{h_s}{h_i} \right\rceil \right\rceil C_i - \left\lceil \frac{m_{ij}}{k_i} \left\lceil \frac{h_s}{h_i} \right\rceil \right\rceil C_i \right) L_s}{|L|}$ et $\triangle p(\rho,i,j) = \frac{v_{i\rho_i} - v_{ij}}{\triangle\alpha(\rho,i,j)}$

Chercher une amélioration faisable $X' = (\delta, \eta)$ qui maximise $\triangle p(\rho,i,j)$

Si une telle amélioration existe et $\triangle p(\rho,i,j) > 0$, alors poser $\rho = \left(\rho|X'\right)$ et revenir début de ***Phase 2*** ; sinon passer ***Phase 3***.

Phase 3 : améliorer itérativement la solution avec les améliorations faisables suivies d'un ou plusieurs dégradations

Etape 3.1

Poser $\triangle t(\rho,i,j) = \sum_{s=i+1}^{n} \frac{\left\lceil \frac{m_{i\rho_i}}{k_i} \left\lceil \frac{h_s}{h_i} \right\rceil \right\rceil C_i - \left\lceil \frac{m_{ij}}{k_i} \left\lceil \frac{h_s}{h_i} \right\rceil \right\rceil C_i}{h_s - L_s}$ et $\triangle p'(\rho,i,j) = \frac{v_{i\rho_i} - v_{ij}}{\triangle t(\rho,i,j)}$

Chercher une amélioration $Y = \left(\delta', \eta'\right)$ maximisant $\triangle p'(\rho,i,j)$

Poser $\rho' = (\rho|Y)$

Etape 3.2

Poser $\triangle t'(\rho,i,j) = \sum_{s=i+1}^{n} \frac{\left\lceil \frac{m_{i\rho_i}}{k_i} \left\lceil \frac{h_s}{h_i} \right\rceil \right\rceil C_i - \left\lceil \frac{m_{ij}}{k_i} \left\lceil \frac{h_s}{h_i} \right\rceil \right\rceil C_i}{L_s}$ et $\triangle p''(\rho,i,j) = \frac{v_{i\rho_i} - v_{ij}}{\triangle t'(\rho,i,j)}$

Chercher une dégradation $Y = \left(\delta'', \eta''\right)$ maximisant $\triangle p''(\rho,i,j)$ tel que $U\left(\rho'|Y'\right) > U(\rho)$

Si Y' est trouvé et $\left(\rho'|Y'\right)$ est faisable, poser $\rho = \left(\rho'|Y'\right)$ et revenir ***Phase 2***

Si Y' est trouvé et $\left(\rho'|Y'\right)$ n'est pas faisable, poser $\rho' = \left(\rho'|Y'\right)$ et revenir à l'étape ***3.2***

Si Y' n'est pas trouvé, la solution est ρ - ***Fin***

6.6 Exemple numérique

Dans cette section, nous présentons un exemple numérique pour illustrer l'approche d'ordonnancement proposée. Nous considérons le système de chariot que nous avons utilisé dans la section 4.1.2 comme le processus à contrôler, et nous étudions le problème de contrôler simultanément quatre chariots, $Chariot_1$, $Chariot_2$, $Chariot_3$, $Chariot_4$ qui peuvent être arrêtés ou démarrés à tout instant de la vie du système. Les contrôleurs des chariots sont implémentés par quatre tâches de contrôle qui partagent un processeur pour leur exécution, posant ainsi le problème de surcharge du processeur. Pour garantir l'ordonnançabilité des tâches tout en maintenant une bonne performance de contrôle globale, nous appliquons l'approche adaptative présentée dans ce chapitre qui consiste à sélectivement rejeter les instances des tâches selon la contrainte (m,k)-firm qui leur est attribuée. Afin de montrer l'avantage de notre approche, nous avons également réalisé une simulation lorsque le système de tâche est ordonnancé selon une priorité fixe (allocation de priorité réalisée selon l'algorithme RM - Rate Monotonic - sans rejet d'instances) ; la comparaison des résultats de simulation obtenus avec les deux approches d'ordonnancement est ensuite présentée.

6.6.1 Processus et contrôleurs

Le modèle à temps continu de $Chariot_i$ est donné par :

$$dx = \begin{bmatrix} 0 & 1 \\ 0 & \frac{-11.4662}{M_i} \end{bmatrix} x dt + \begin{bmatrix} 0 \\ \frac{1.7434}{M_i} \end{bmatrix} u dt + dv_c$$

où M_i est la masse du chariot ; v_c est un bruit blanc de covariance $R_c = \begin{bmatrix} 3.24 & -1.8 \\ -1.8 & 1 \end{bmatrix}$. Les quatre chariots, $Chariot_1$, $Chariot_2$, $Chariot_3$, $Chariot_4$ ont des masses différentes : $M_1 = 1.5$, $M_2 = 1.2$, $M_3 = 0.9$, $M_4 = 0.6$, exprimées en kilo. Le contrôleur de $Chariot_i$ est noté $Controleur_i$. La période de $Controleur_i$ est notée h_i avec $h_1 = 0.007$, $h_2 = 0.0085$, $h_3 = 0.01$, $h_4 = 0.0115$.

La performance de contrôle, au niveau de chaque contrôleur, est mesurée en utilisant le coût LQ suivant :

$$J = \lim_{N\to\infty} \frac{1}{N} E\left(\int_0^N x^T(t) Q x(t) + u^T(t) R u(t) dt \right) \tag{6.4}$$

avec $Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ et $R = 0.00006$.

Les tâches de contrôle implémentant les contrôleurs sont ordonnancées selon un algorithme à priorité fixe où l'allocation de priorité est obtenue par *Rate Monotonic*. Puisque la tâche ayant la plus grande période possède la priorité la plus basse, le rejet selectif de instances de cette tâche selon une contrainte (m,k)-firm n'a aucune influence sur l'ordonnançabilité des

autres tâches plus prioritaires. En conséquence, la tâche de contrôle implémentant $Chariot_4$ (qui possède la période la plus grande) est exécutée sans rejet d'instances, autrement dit, elle s'exécute sous la contrainte (k_4, k_4)-firm. Le paramètre k_i de la contrainte (m_i, k_i)-firm pour les autres systèmes de chariot est choisi en utilisant l'approche proposée dans le chapitre 4 afin de garantir la stabilité du système. Nous obtenons ainsi une valeur de k_i plus grande que 10 pour $Chariot_1$, $Chariot_2$. Pour faciliter la démonstration des résultats de simulation, nous rendons la valeur de k_i inférieure à 10 pour les deux contrôleurs sans affecter la stabilité du système. Finalement, pour $Chariot_1$, $Chariot_2$, $Chariot_3$, le paramètre k_i est respectivement 5, 8 et 10, et la valeur de m_i peut varier entre $[1..k_i]$.

Les paramètres du contrôleur sous la contrainte (m,k)-firm minimisant le coût LQ 6.4 ainsi que le coût minimal correspondant sont calculés selon l'approche présentée dans la section 5.1.1. Durant le fonctionnement du contrôleur, lorsque la contrainte (m,k)-firm associée est modifiée, les paramètres du contrôleur sont ré-évalués selon la nouvelle contrainte (m,k)-firm attribuée pour maintenir la performance de contrôle à un bon niveau autant que possible. Afin d'obtenir une adaptation rapide au changement de contrainte (m,k)-firm, les paramètres du contrôleur pour chaque contrainte (m,k)-firm possible sont calculés hors-ligne et stockés dans un table.

6.6.2 La configuration d'expérimentation

Le modèle de simulation est créé en utilisant MATLAB/Simulink et la boîte à outils TrueTime [Cervin03].

Le temps d'exécution, C_i, de chaque tâche de contrôle est fixé à 3 ms, et celui du régulateur des tâches est fixée à 2.5 ms dans cette expérimentation (le temps d'exécution du régulateur des tâches peut varier en fonction du nombre des tâches dans le système et du paramètre k de chaque tâche). Le régulateur des tâches possède la priorité la plus haute et les priorités des tâches de contrôle sont attribuées selon l'algorithme RM. Dans tout ce qui suit, on considère que les tâches sont à échéance sur requête (échéance relative égale à la période).

La scénario de la simulation est comme suit (un mode de marche initial et deux changements de mode de marche, respectivement en $t = 1$ et $t = 2$) :

- A l'instant $t = 0$, $Controleur_1$ et $Controleur_2$ sont activés, tandis que $Controleur_3$ et $Controleur_4$ sont à l'état arrêt ;
- A $t = 1$, $Controleur_4$ est activé ;
- A $t = 2$, $Controleur_3$ est activé.

La performance de contrôle de $Controleur_i$ est représentée par le coût qui a la forme suivante :

$$J_i(t) = \int_0^t x^T(s)Qx(s) + u^T(s)Ru(s)ds \tag{6.5}$$

Le coût 6.5 est calculé pour chaque contrôleur à chaque unité de temps. Ainsi une bonne performance de contrôle est représentée par une augmentation "lissée" de la fonction 6.5.

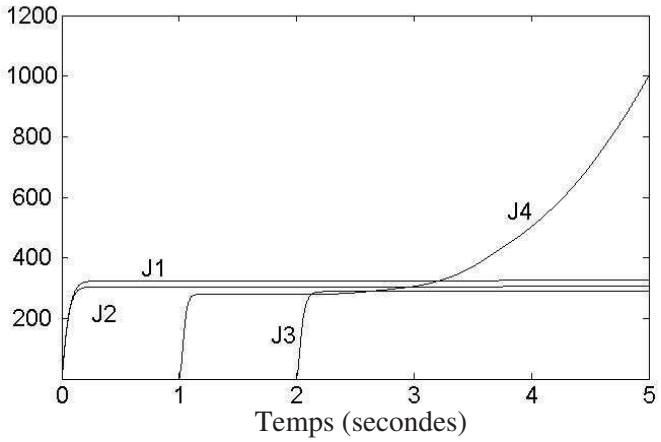

FIGURE 6.3 – Coûts J_i, pour chacun des quatre contrôleurs, avec l'approche d'ordonnancement classique

Pour comparer les résultats obtenus avec différentes approches d'ordonnancement, les quatre chariots sont soumis à une séquence de bruits blancs identiques pour chaque simulation.

6.6.3 Résultats de simulation

Dans cette section, les résultats de simulation obtenus avec l'approche d'ordonnancement classique (Priorité fixe sans rejet) ainsi que l'approche d'ordonnancement adaptative proposée sont présentés.

6.6.3.1 L'approche d'ordonnancement classique

Les coûts de $Controleur_1$, $Controleur_2$, $Controleur_3$ et $Controleur_4$, notés respectivement J_1, J_2, J_3 et J_4 sont montrés sur la figure 6.3.

Les vues détaillées de l'ordonnancement aux instants $t = 1$ et $t = 2$ sont présentées respectivement sur la figure 6.5 et la figure 6.6. Pour ceci, nous utilisons quelques conventions de représentation :

– L'état de chaque tâche peut prendre trois valeurs comme illustré sur la figure 6.4 :

1. Exécutée : en cours d'exécution sur le processeur (par exemple, les états entre $t2$ et $t3$, $t5$ et $t6$) ; Dans ce cas la tâche figure en niveau haut ;
2. Préemptée : l'exécution est préemptée, en attente de la disponibilité du processeur (par exemple, les états entre $t1$ et $t2$, $t5$ et $t6$) ; la tâche figure en niveau médian sur les figures ;
3. Non-activée : il n'y a pas d'instance à exécuter (par exemple, les états avant $t1$, entre $t3$ et $t4$, après $t6$) ; la tâche figure en niveau bas sur les figures.

– Les flêches indiquent l'arrivée des instances.

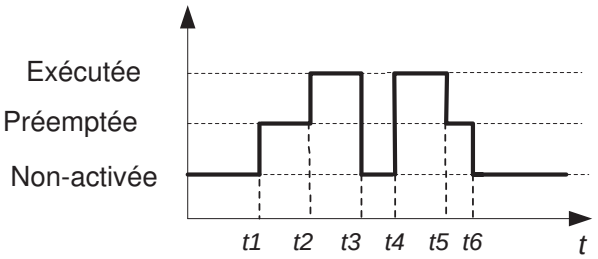

FIGURE 6.4 – Représentation de l'état de tâche

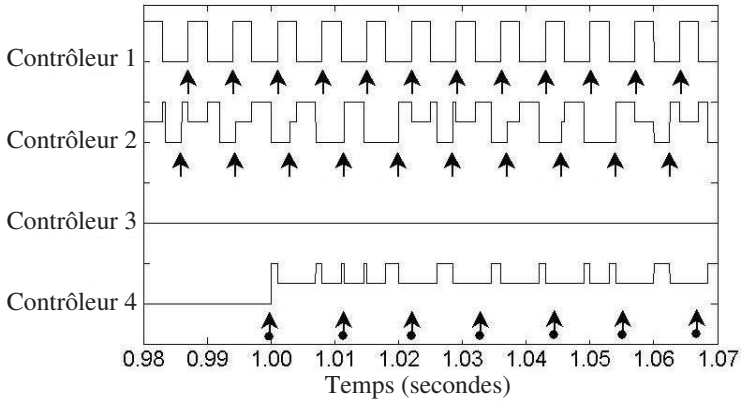

FIGURE 6.5 – Détail du comportement des tâches associées à chaque contrôleur, à $t = 1$, avec l'approche d'ordonnancement classique (le temps est en abscisse et l'état des tâches en fonction du temps est représenté selon la convention de la figure 6.4)

– Les flêches avec un point rond représentent l'arrivée des instances qui manquent leur échéance dans le déroulement de l'application.

Analyse des chronogrammes illustrés par les figures 6.5 et 6.6

– Durant la simulation, $Controleur_1$ et $Controleur_2$ sont ordonnancés sans dépassement d'échéance. Leurs coûts augmentent lentement en fonction du temps, et les deux chariots contrôlés fonctionnent bien.
– A $t = 1$, $Controleur_4$ est activée. Avec l'interférence d'exécution induite par $Controleur_1$ et $Controleur_2$, les échéances des instances de $Controleur_4$ sont toutes manquées. Cependant, malgré, d'une part, le retard, pour chaque instance, de la date de début d'exécution effective par rapport à la date d'activation prévue et, d'autre part, de la fin effective d'exécution de chaque instance par rapport à l'échéance, la performance de contrôle de $Controleur_4$ reste acceptable avant l'activation de $Contrleur_3$ (à $t = 2$).
– A $t = 2$, $Controleur_3$ est activée. L'exécution de $Controleur_4$ est complètement interrompue à cause de l'interférence d'exécution induit par $Controleur_1$, $Controleur_2$ et $Controleur_3$. $Chariot_4$ devient en conséquence instable (ce qui est représenté par une

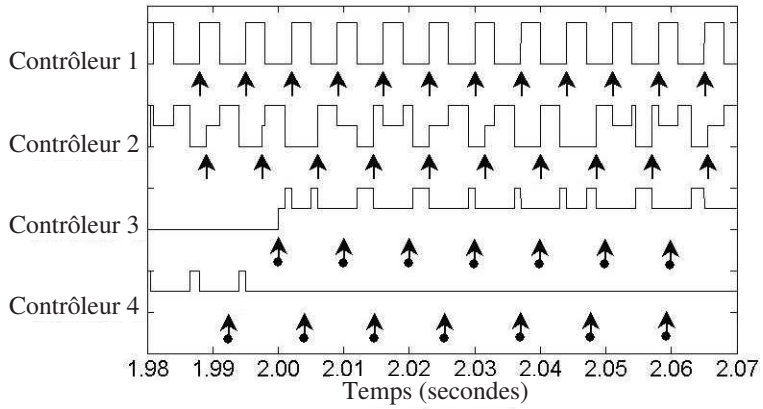

FIGURE 6.6 – Détail du comportement des tâches associées à chaque contrôleur, à $t = 2$ avec l'approche d'ordonnancement classique (le temps est en abscisse et l'état des tâches en fonction du temps est représenté selon la convention de la figure 6.4)

forte augmentation du coût pour $t > 2$). Noter que malgré une interférence d'exécution induite par $Controleur_1$ et $Controleur_2$ (retards analogues aux retards précédents vus pour $Controleur_4$ en $t = 1$), la performance de $Controleur_3$ ne se dégrade pas aussi rapidement que celle du $Controleur_4$.

6.6.3.2 L'approche d'ordonnancement adaptative

Nous avons mis en oeuvre l'approche d'ordonnancement adaptatif (c'est-à-dire calculé à chaque changement de mode de marche) sur la même application. Les résultats de simulation obtenus sont présentés dans ce paragraphe. Les conventions de représentation sont les même que dans la section précédente. De plus, les flêches grises signifient que les instances activées aux instants indiqués sont classées comme instances optionnelles et ne seront pas exécutées.

Les coûts des quatres contrôleurs sont illustrés par la figure 6.7. Les détails de comportement de l'ordonnancement autour des instants de changement de mode de marche ($t = 1$ et $t = 2$) sont illustrés par les figures 6.8 et 6.9.

- A $t = 0$, seul $Controleur_1$ et $Controleur_2$ sont activées. Comme les deux tâches sont ordonnançables sans rejet, les (m,k)-contraintes associées sont donc la contrainte (k,k)-firm.
- A $t = 1$, un changement de configuration des tâches est détecté et le régulateur des tâches est immédiatement activé. A $t = 1.025$, la contrainte (m,k)-firm de $Controleur_2$ est ajustée à la contrainte $(4,8)$-firm et celle de $Controleur_2$ reste à la contrainte (k,k)-firm. La condition de surcharge est ainsi éliminée.
- A $t = 2$ se répète le même scénario, où les contraintes (m,k)-firm de $Controleur_1$, $Controleur_2$ et $Controleur_3$ sont ajustées respectivement la contrainte $(2,5)$-firm, $(4,8)$-firm et $(3,10)$-firm. La contrainte (m,k)-firm de $Controleur_4$ reste toujours à la contrainte

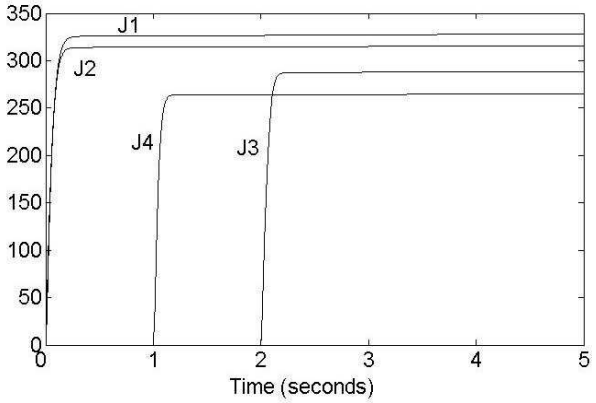

FIGURE 6.7 – Coûts J_i, pour chacun des quatre contrôleurs, avec l'approche d'ordonnancement adaptatif

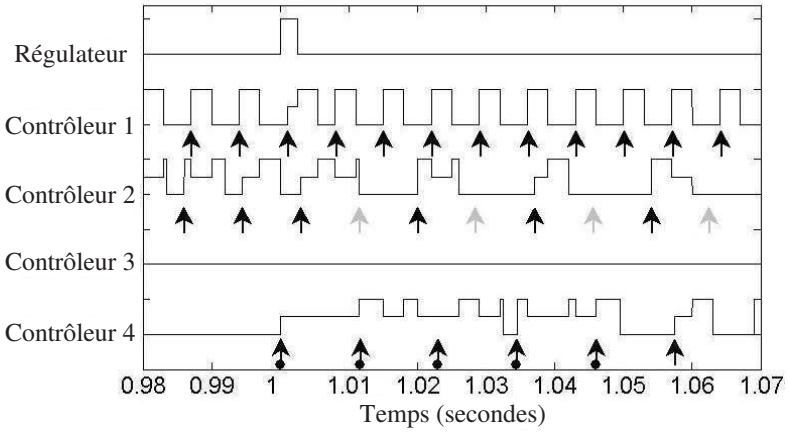

FIGURE 6.8 – Détail du comportement du système à $t = 1$ avec l'approche d'ordonnancement adaptative (le temps est en abscisse et l'état des tâches en fonction du temps est représenté selon la convention de la figure 6.4)

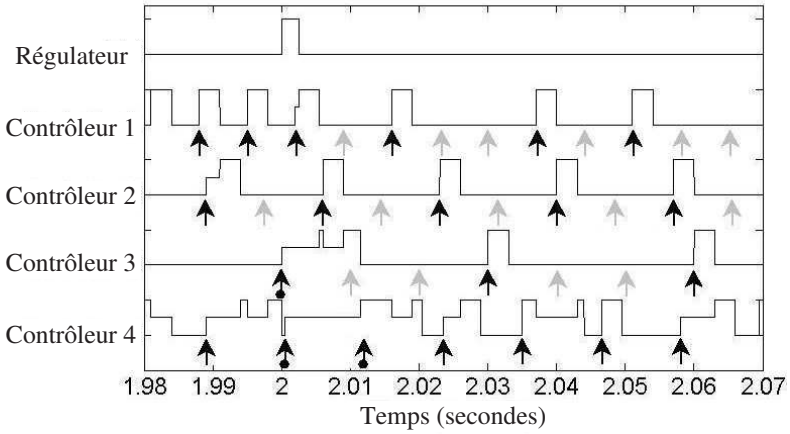

FIGURE 6.9 – Détail du comportement du système à $t = 1$ avec l'approche d'ordonnancement adaptative (le temps est en abscisse et l'état des tâches en fonction du temps est représenté selon la convention de la figure 6.4)

(k, k)-firm (se référer au paragraphe 6.6.1). Noter que le non-respect de l'échéance des instances obligatoires, après l'activation de la nouvelle tâche, est dû à la surcharge passagère qui disparaît, cependant, très rapidement après la déduction des nouvelles contraintes (m,k)-firm.

La comparaison des résultats de simulation obtenus avec les deux approches d'ordonnancement nous montre que, en appliquant l'approche d'ordonnancement adaptative proposée, la performance globale des contrôleurs est beaucoup améliorée relativement à l'approche d'ordonnancement sans mécanisme de gestion de surcharge adaptatif : $Controleur_4$ reste stable et le coût de $Controleur_3$ et $Controleur_4$ ne dépasse pas 300 à $t = 5$.

6.7 Conclusion

Dans ce chapitre, nous considérons une application de contrôle-commande évolutive constituée d'un ensemble de boucles de contrôle. Chaque contrôleur est implémenté sous la forme d'une tâche temps réel responsable du calcul de la loi de commande. Toutes les tâches partagent un processeur pour leur exécution, et nous étudions le problème d'ordonnancement de ces tâches de contrôle. En raison de l'échec des approches d'ordonnancement classiques (sans rejet) face à l'incertitude de configuration de l'application, nous proposons une approche d'ordonnancement adaptative basée sur le modèle de contrainte (m,k)-firm, qui consiste à sélectivement rejeter les instances des tâches afin de garantir l'ordonnançabilité des tâches tout en maintenissant la performance de contrôle globale de l'application à un bon niveau.

Comparée avec les approches d'ordonnancement non-adaptatif, l'approche proposée permet une adaptation dynamique de la stratégie d'ordonnancement en fonction de la configuration courante de l'application, garantissant le respect des contraintes temporelle dans le cas de

ressource limitée et satisfaisant les exigences de performance des applications évolutives. De plus, l'approche proposée permet de surmonter les inconvénients, mentionnés dans la section 1.3.3, des approches d'ordonnancement adaptatives basées sur le réglage de périodes. Concrètement, contrairement aux techniques de réglage de périodes (par exemple, celle proposée dans [Eker00]), l'approche proposée ne dépend pas du type de la fonction représentant la performance de contrôle ni d'hypothèses sur les caractéristiques de cette fonction. Par exemple, dans le cas où la performance de contrôle est représentée par une fonction de coût LQ, il n'est pas nécessaire que cette fonction soit convexe, ni même approchable par une fonction convexe ainsi que cela est exigé pour le réglage de périodes. Enfin, dans le cas de tâches soumises à des contraintes de précédence, l'adaptation des périodes de toutes les tâches impliquées dans cette relation de précédence, n'est pas nécessaire pour maintenir un échange efficace d'informations entre ces tâches, car les périodes nominales des tâches, les h_i restent constantes durant toute la vie de l'application.

Conclusion et perspectives

1 Travaux réalisés

L'objectif de cette thèse est de proposer des méthodes de conception qui permettent de réaliser la meilleure performance de contrôle des applications évolutives sous contraintes de ressources de calcul ou de communication. Notons que c'est aussi équivalent à dire qu'on vise à maintenir la meilleure performance de contrôle possible (ou avec dégradation contrôlée) en cas de surcharge des ressources. Pour atteindre cet objectif le système doit offrir des services adaptatifs et capable de fournir des niveaux de QdS (Qualité de Service) appropriés. Une solution classique revient à définir des contraintes temporelles strictes et à mettre en oeuvre la preuve de l'ordonnançabilité (de taches et messages) dans le pire cas. Cette méthode conduit généralement à surdimensionner les ressources nécessaires. Les mêmes principes, dès lors qu'ils intègrent les dégradations de performances dues à des perturbations de l'environnement, peuvent conduire à des solutions extrêmes dont l'implantation serait irréaliste. Une solution qui repose sur l'évaluation des performances probabilistes du système support permet d'éviter un sur-dimensionnement coûteux, mais ne fournit qu'une garantie en moyenne ou, au mieux, probabiliste. Ceci s'avère insuffisant pour les applications de contrôle-commande. Face à ces solutions, le modèle de contraintes (m,k)-firm (au moins m parmi k instances consécutives quelconques doivent respecter leurs échéances), apparaît comme une technique intéressante. L'expression des contraintes et la définition de niveaux de QdS selon ce modèle est plus générale que la formulation précédente tout en l'incluant. Cette thèse a exploré la possibilité d'appliquer le modèle (m,k)-firm pour permettre à une application de contrôle temps réel de respecter ses contraintes temporelles et de satisfaire les exigences de performance de contrôle par la mise en oeuvre de mécanismes adaptatifs en-ligne.

Pour que les techniques proposées soient pertinentes, les travaux sont menés en se basant sur un nouveau modèle d'étude de commande et d'ordonnancement : la conception conjointe de commande et d'ordonnancement qui consiste à effectuer la conception de commande et d'ordonnancement conjointement au lieux de les traiter comme deux aspects isolés : d'une part, les contrôleurs sont conçus en tenant compte des caractéristiques temporelles d'ordonnancement impliqué par le partage de la ressource, et d'autre part, les tâches ou messages sont ordonnancés sans ignorer les conséquences sur la performance de contrôle.

Le travail de cette thèse s'est déroulé en deux phases. La première phase consiste à étu-

dier l'intégration du modèle (m,k)-firm dans les applications de contrôle-commande centralisées ou distribuées autour d'un réseau de communication. Nous avons, dans un premier temps, analysé l'impact des rejets de messages selon le modèle (m,k)-firm sur un système de contrôle-commande mono-variable. La stabilité et l'optimisation des performances de contrôle du système sous la contrainte (m,k)-firm ont été systématiquement analysées. Puis l'extension de l'étude appliquée aux systèmes multi-variables a été conduite. Nous avons montré comment déterminer la contrainte (m,k)-firm pour que la stabilité du système puisse être garantie, et que la distribution des rejets des échantillons dans la séquence des échantillons exerce une influence notable sur la performance du système. A partir de ces résultats, nous avons proposé une approche qui détermine, étant donné la contrainte (m,k)-firm, la distribution optimale des rejets des échantillons. A la fin de cette étape, le problème d'ordonnancement des messages sous contraintes (m,k)-firm a été traité. Après avoir montré que les problèmes d'ordonnancement autour du modèle (m,k)-firm sont tous NP-difficiles, deux conditions suffisantes sont données pour déterminer l'ordonnançabilité des messages sous contraints (m,k)-firm. Dans la deuxième phase, nous considérons l'application de contrôle-commande dans laquelle le processeur est partagé par un ensemble de tâches de contrôle, et nous avons proposé une approche d'ordonnancement qui intègre de façon coordonnée les caractéristiques qui expriment la qualité de contrôle au sens de l'automatique, et les paramètres de l'ordonnancement temps-réel des tâches. L'intérêt de cette approche coordonnée réside essentiellement dans la minimisation des ressources nécessaires tout en garantissant la qualité de contrôle requise. Les résultats obtenus de la première phase du travail constituent les bases théoriques pour le déroulement de la seconde phase de la thèse. C'est ainsi qu'une politique de gestion en-ligne basée sur le modèle (m,k)-firm a été conçue afin de garantir la qualité des différents systèmes de contrôle dans la limite de la ressource disponible.

Cette politique est clairement une approche de conception conjointe car nous régulons à la fois le gain des contrôleurs et l'ordonnancement (ceci en terme d'adaptation de contraintes (m,k)-firm de chaque tâche de contrôle, bien que l'algorithme lui-même reste toutefois un ordonnancement à priorité fixe sous contrainte (m,k)-firm).

2 Aspect d'implémentation de solutions

Le travail mené dans cette thèse permet d'optimiser la performance de contrôle des applications temps-réel tolérant les pertes/rejets par la régulation conjointe de contraintes (m,k)-firm et du gain des contrôleurs. L'ensemble de propositions a été validé analytiquement et par simulation sous Matlab/Simulink et en utilisant aussi le paquet logiciel associé TrueTime quand c'est nécessaire. Néanmoins, la mise en oeuvre du mécanisme de régulation est souvent soumise à certaines contraintes d'implémentation, parmi lesquelles les plus importantes sont les suivantes :

– Les caractéristiques du temps d'exécution de l'algorithme de contrôle. Le pire temps d'exécution est-il prédictible sur un OS choisi ? Quelle sera la variation du temps d'exé-

cution ? Sachant qu'en analyse et simulation nous avons supposé un temps d'exécution constant et que chaque changement de configuration d'une tâche est connu par un superviseur.

- Optimisation en ligne : Quelles sont les informations qui peuvent être mesurées en ligne, et quelle est la consommation supplémentaire de la ressource partagée due à l'exécution du processus d'optimisation ? C'est aspect a été supposé négligeable dans notre analyse et simulation
- Les caractéristiques de la ressource partagée : Est-il possible d'adapter les paramètres des contrôleurs aux changement de paramètres d'ordonnancement des tâches (ou messages) sans avoir recours à des mécanismes complexes ? Est-il faisable d'implémenter le processus d'optimisation sur tout type de plate-forme (OS et protocoles) ?

Bien que certaines de ces questions relèvent du problème de l'ingénieurie, elles méritent toutes d'être discutées afin de voir l'implémentabilité de nos propositions.

Les études menées dans cette thèse supposent que les informations nécessaires (surtout les caractéristiques des tâches) à la procédure d'optimisation sont disponible (via un superviseur). En cas de partage d'un processeur, la procédure d'optimisation pourra être réalisée sous la forme d'une tâche, et s'exécute avec d'autres tâches dans le noyau. Il est aussi possible de réattribuer les paramètres des contrôleurs (le gain par exemple) au cas d'un changement de paramètres d'ordonnancement pour avoir une meilleur performance. Par contre, si la ressource partagée est le réseau, le rejet de messages selon la contrainte (m,k)-firm se fait sur un noeud de communication (e.g. routeur, switch), dans lequel l'implantation de la procédure d'optimisaton est souvent difficile. Ainsi dans cette étude, les messages sont soumis aux contraintes (m,k)-firm précalculées et sont ordonnancés par un algorithme d'ordonnancement basé sur la contrainte (m,k)-firm (e.g. EFP) sans l'intervention de la procédure d'optimisation. Cette stratégie ne représente toutefois pas de conservatisme car en dehors de la période de surcharge, la performance du système pourra être améliorée par le traitement des messages optionnels. Ce point reste néanmoins à démontrer. Une possibilité est de simuler le cas où les messages optionnels dans EFP peuvent être transmis si la ressource n'est pas surchargée pour voir comment ils contribuent à l'amélioration de performance de contrôle.

Une autre contrainte d'implémentation en cas de partage du réseau est liée à l'architecture de la boucle de contrôle qui est illustrée par la figure 1.4. Le réseau est utilisé pour connecter d'une part les capteurs et le contrôleur et d'autre part le contrôleur et les actionneurs, la surcharge peut survenir des deux côtés suivant le changement de configuration de l'application. Intuitivement, le mécanisme de rejet selon la contrainte (m,k)-firm doit s'appliquer sur les deux côtés pour diminuer la bande passante requise et ainsi éviter la surcharge. Néanmoins, le bon fonctionnnement de l'application (e.g. stabilité) s'appuie sur une connaissance précise des pertes de messages sur le réseau, le mécanisme de communication entre les noeud de communication où s'implémente la contrainte (m,k)-firm est ainsi nécessaire. L'implémentation de tel mécanisme de communication nécessite des dispositifs de réseau plus performants, et de la programmation supplémentaire, ce qui rend la conception et l'implémentation de l'applica-

tion plus complexe et coûteux. Par conséquence, le mécanisme de rejet de message ne se fait dans cette étude que sur le réseau entre les capteurs et le contrôleur, et nous imposons une réservation complète de la bande passante sur le réseau entre le contrôleur et les actionneurs, autrement dit, les messages contenant les commandes sont associés à la contrainte (k,k)-firm.

Un pas vers une implémentation réelle a été réalisé récemment dans le cadre d'un stage d'ingénieur. Il s'agit d'implémenter l'exemple traité dans le chapitre 6 sous Orccad (http ://sed.inrialpes.fr/Orccad/), et ceci dans le cadre du projet ANR ARA Safe_NECS (http ://safe-necs.cran.uhp-nancy.fr/) auquel cette thèse contribue. Un autre exemple du contrôle adaptatif du quadrotor du drone (plate-forme de démonstration du projet Safe_NECS) sous contrainte (m,k)-firm est en cours d'étude. Les résultats attendus devraient confirmer à la fois la validité et la généralité de nos propositions.

3 Perspectives

Les perspectives ouvertes par cette thèse concernent tout d'abord l'amélioration de l'algorithme heuristique pour la déduction des contraintes (m,k)-firm. Premièrement, à cause de la complexité importante des problèmes d'ordonnancement autour du modèle de contraintes (m,k)-firm, le problème de déterminer les contraintes (m,k)-firm ainsi que les (m,k) patterns correspondants pour un ensemble de tâches de sorte que la performance globale de l'application soit optimale sous contrainte de l'ordonnançabilité est prohibitivement complexe. Grâce aux résultats obtenus dans la première phase du travail, nous avons pu fixer la stratégie de définition de (m,k) pattern en utilisant la théorie des mots mécaniques, ce qui a permis de simplifier la complexité du problème et de nous concentrer sur le calcul des contraintes (m,k)-firm. Avec la théorie des mots mécaniques, nous proposons une condition d'ordonnançabilité suffisante qui est utilisée par l'algorithme d'optimisation pour dériver les contraintes (m,k)-firm. Nous avons montré que des conditions d'ordonnançabilité plus serrées existent lorsque les rejets d'instances (ou messages) se font selon des stratégies spécifiques. Ainsi, la définition de nouvelles conditions d'ordonnançabilité plus appropriées permettra d'obtenir des solutions plus pertinentes. Sur l'aspect d'ordonnancement, une piste intéressante à explorer consiste à mener une étude de comparaison entre un (m,k)-pattern et le modèle de tâches à offset (ou transaction), puis voir la possibilité d'appliquer des techniques d'analyse d'ordonnançabilité de ce dernier afin d'obtenir des conditions suffisantes encore plus serrées [Traore07]. Deuxièment, cet algorithme a été basé sur l'algorithme HEU qui a été conçu pour résoudre le problème MMKP (Multiple-Choice Multi-Dimension Knapsack Problem). Il nous semble que les proprétés de l'ordonnancement sous contraintes (m,k)-firm peuvent faciliter la résolution du problème MMKP. Ainsi un autre point intéressant serait d'exploiter les possibilités d'adapter l'algorithme d'optimisation selon les propriétés de l'ordonnancement sous contraintes (m,k)-firm pour obtenir de meilleures solutions.

Une deuxième perspective intéressante sera l'étude des conditions de déclenchement de reconfiguration (adaptation de gain et de (m,k)-pattern). En effet, dans notre étude nous avons

supposé que la condition déclencheur de la reconfiguration est le changement de configuration d'une tâche (exemple : activation d'une nouvelle tâche). Dans la pratique ce déclencheur pourra provenir du changement notable de l'état du procédé, d'une consigne de reconfiguration suite au diagnostic d'un défaut, etc. Un autre problème lié est la détermination du moment de changement car on pense qu'il serait parfois préférable du point de vue de stabilité, de ne changer la configuration qu'à des instants discrets avec une période minimale de kh. Il serait donc intéressant d'évaluer la pertinence des conditions de déclenchement de reconfiguration. D'ailleurs, un premier un travail dans ce sens a été réalisé récemment [Felicioni08] dans lequel nous avons considéré la variation de l'état du procédé comme le déclencheur de la reconfiguration. En plus le problème de stabilité du contrôleur avec le changement de (m,k)-pattern est aussi étudié. La stabilité asymptotique peut être assurée dès lors qu'on trouve une fonction quadratique commune de Lyapunov pour tous les (m,k)-patterns possibles.

Annexe A

Compléments au chapitre 2

Description de tâche

Une tâche est en charge de fournir un des services de l'application. Elle correspond aux exécutions d'une séquence d'opérations donnée sur le processeur.

Dans le modéle canonique du domaine, le service fourni par la tâche peut être rendu plusieurs fois au cours de la vie du système (e.g. la procédure de calcul de la loi de commande est périodique durant toute la vie du procédé physique à contrôler). Ce qui signifie que la séquence d'opérations d'une tâche peut être ré-exécutée plusieurs fois. Pour cette raison, chacune de ces exécutions de la séquence d'opérations est considérée individuellement sous la forme d'instance.

Plusieurs instances de plusieurs tâches sont susceptibles de s'exécuter sur un même processeur, mais, par définition, ce dernier ne peut en exécuter qu'une seule à la fois. Une instance est donc associée à une structure de données qui renferme en particulier son état d'exécution. Le systéme d'exploitation s'occupe entre autres de faire passer les instances d'un état à un autre suivant le diagramme de transition des instances (la figure A.1 propose un exemple), en même temps qu'il choisit d'exécuter une instance plutôt qu'une autre. Généralement, le diagramme d'état des instances fait intervenir les états suivants :

- *ready* : instance prête à démarrer, c'est-à-dire pas encore démarrée, ou dont l'exécution a été interrompue pour laisser la place à une autre instance ;
- *running* : instance en cours d'exécution sur le processeur ;
- *blocked* : instance en attente de la disponibilité d'une ou plusieurs ressources ;
- *sleeping* : instance en sommeil, en attente d'un événement de réveil ;
- *stopped* : instance terminée.

La vie d'une instance peut également être représentée suivant un chronogramme comme l'exemple illustré par la figure A.2. Sur le chronogramme, les caractéristiques temporelles suivantes d'une instance peuvent être identifiées :

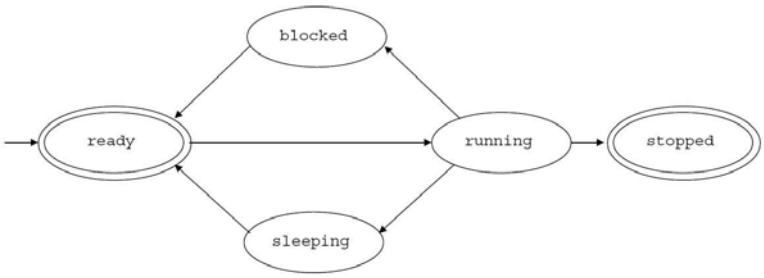

FIGURE A.1 – Diagramme d'état des instances

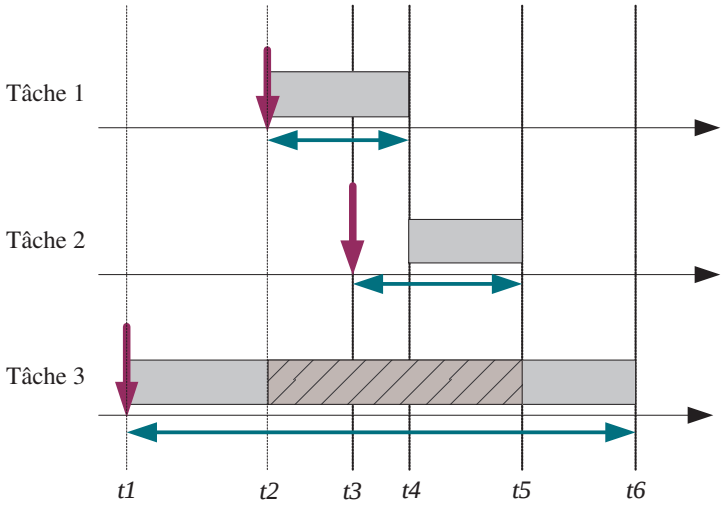

FIGURE A.2 – Evénement au cours de la vie d'une instance

- date d'activation : l'instance est créée et prête à être exécutée sur le processeur (correspond aux instants $t1$, $t2$ et $t4$ pour respectivement tâche 3, tâche 1 et tâche 2) ;
- date de démarrage : l'instance commence à être exécutée sur le processeur pour la première fois (correspond aux instants $t1$, $t2$ et $t4$ pour respectivement la tâche 3, tâche 1 et tâche 2) ;
- dates de préemption : l'instance est momentanément interrompu au profit d'une autre tâche plus prioritaire (correspond à l'instant $t2$ pour la tâche 3) ;
- dates de reprise : Le processeur reprend l'exécution de l'instance, là où elle avait été précédemment préemptée (correspond à l'instant $t4$ pour la tâche 3) ;
- date de terminaison : La fin d'exécution de l'instance sur le processeur (correspond aux instants $t4$, $t5$ et $t6$ pour respectivement la tâche 1, tâche 2 et tâche 3) ;
- temps de réponse : L'écart entre la date de terminaison et la date d'activation d'un travail (correspond aux intervalles de temps entre les instants $t1$ et $t6$, $t2$ et $t4$, $t4$ et $t5$ pour respectivement la tâche 3, tâche 1 et tâche 2).

Caractéristiques de tâche

Le système d'exploitation doit connaître les caractéristiques temporelles des instances de chaque tâche, formant une partie du modèle de tâche, puisqu'il doit garantir que les contraintes temporelles sont respectées. Dans le modèle de tâche figurent habituellement les caractéristiques temporelles suivantes :

- *Le temps d'exécution* : il s'agit du temps d'exécution d'une instance de la tâche sur le processeur. En général, il s'agit d'un temps d'exécution au pire-cas (ou WCET pour Worst Case Execution Time en anglais), c'est à dire un majorant sur tous les temps d'exécution possibles de toutes les instances de la tâche. Cette donnée constitue un paramètre d'entrée important pour les méthodes et outils de vérification du respect des échéances.
- *La période* : Les instances de la tâche sont créés de façon rigoureusement périodique (il existe d'autres types de répartition dans le temps des dates d'activation des instances de la tâche. Pourtant, dans cette étude, nous ne nous intéressons qu'à la répartition périodique, car en pratique, les systèmes de contrôle commande sont souvent périodiques). La période de la tâche indique le temps entre deux dates d'activation consécutives.
- *L'échéance* : imposée par la contrainte temporelle, il s'agit de la date à laquelle les instances doivent être terminées, relativement à leur date d'activation.

Annexe B

Stabilité et Contrôle optimal de systèmes hybrides stochastiques à sauts markoviens

B.1 Modèle de système

Considérons le système linéaire à temps invariant décrit par les équations suivantes :

$$\begin{aligned} x_{i+1} &= Ax_i + Bu_i \\ y_i &= Cx_i \end{aligned} \tag{B.1}$$

où $x_i \in \mathbb{R}^n$ est le vecteur d'état, $u_i \in \mathbb{R}^m$ est le vecteur des entrées et $y_i \in \mathbb{R}^p$ représente le vecteur des sorties. $A \in \mathbb{R}^{n \times n}$ est la matrice d'état du système linéaire, $B \in \mathbb{R}^{n \times m}$ est la matrice d'entrée et $C \in \mathbb{R}^{p \times n}$ est la matrice de sortie.

Sous la configuration d'implantation évoquée dans la section 2, nous considérons que les messages contenant le vecteur d'état sont rejetés sélectivement selon une contrainte (m,k)-firm par l'algorithme d'ordonnancement DBP pour diminuer la bande passante requise par le système. Le processus de rejet de messages peut être représenté par une chaîne de Markov. La figure B.1 donne un exemple de diagramme d'état transitions sous contrainte (2,3)-firm, où un état de la chaîne de Markov est représenté par le (m,k)-pattern courant qui représente les trois derniers états de réception de message, et p_i est la probabilité de transition d'un pas du processus markovien. Par convention, le déplacement des bits dans un (m,k) pattern se fait de droite vers la gauche.

Supposons qu'étant donné une contrainte (m,k)-firm, la chaîne de Markov possède n états. On numérote les états (qui sont représentés chacun par un (m,k) pattern) de la chaîne de Markov par les nombres entiers de 1 à n. La distribution de probabilité peut être ainsi représentée par une matrice de transition dont le $(x, y)^{me}$ élément vaut

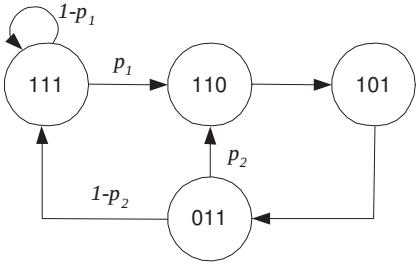

FIGURE B.1 – Diagramme d'état-transition avec la contrainte (2,3)-firm

$$\Pr\left[x_i = x \mid x_{i-1} = y\right] = p_{xy}$$

avec $p_{xy} \geq 0$ et $\sum_{j=1}^{n} p_{ij} = 1$ pour $i \in [1, 2., n]$.

Notons y_{ci} le vecteur des sorties reçu par le contrôleur, on a :

$$y_{ci} = \theta_i y_i + (1 - \theta_i)\, y_{c(i-1)} \tag{B.2}$$

où $\theta_i = 1$ si le i^{ime} message est reçu par le contrôleur, sinon $\theta_i = 0$.

En introduisant le vecteur d'état augmenté $\bar{x}_i \in \mathbb{R}^{n+p}$:

$$\bar{x}_i = \left[x_i^T\, y_{c(i-1)}^T\right]^T$$

on obtient le système à temps variant intégrant l'effet de rejet de messages :

$$\begin{array}{rcl} \overline{x}_{i+1} & = & \bar{A}_{\theta_i}\bar{x}_i + \overline{B}u_i \\ y_{ci} & = & \overline{C}_{\theta_i}\overline{x}_i \end{array} \tag{B.3}$$

avec

$$\bar{A}_{\theta_i} = \begin{bmatrix} A & 0 \\ \theta_i C & (1-\theta_i)\, I \end{bmatrix} \tag{B.4}$$

$$\overline{B} = \begin{bmatrix} B \\ 0 \end{bmatrix} \tag{B.5}$$

$$\overline{C}_{\theta_i} = \begin{bmatrix} \theta_i C & (1-\theta_i)\, I \end{bmatrix} \tag{B.6}$$

Puisque le processus de rejet de messages est un processus markovien, le système décrit par les équations B.3 est ainsi un système hybride stochastique à sauts markoviens. Dans les sections suivantes, nous allons d'abord introduire une condition suffisante et nécessaire pour déterminer la stabilité du système, puis en supposant que le contrôleur a connaissance de l'état courant du système dans la chaîne de Markov, nous présentons une approche pour concevoir le contrôleur optimal.

B.2 Stabilité de système

Le problème de stabilité du système décrit par les équations B.3 a été étudié dans [Costa93] où une condition suffisante et nécessaire est donnée pour déterminer la stabilité du système. Dans la suite, nous présentons d'abord des définitions de stabilité préliminaires qui proviennent de [Ji91], puis nous donnons la condition de stabilité.

Définition 3 *Le système décrit par les équations B.3 avec* $u \equiv 0$ *est dit :*

1. mean-square stable *(MSS) si pour tout état intial* $(\overline{x}_0, \theta_0)$, $\lim_{i\to\infty} E\left[\|\overline{x}_i\|^2\right] = 0$;
2. *exponentiellement stable si pour tout état intial* $(\overline{x}_0, \theta_0)$, $E\left[\sum_{i=0}^{\infty} \|\overline{x}_i\|^2\right] < \infty$;
3. exponentially mean square stable *(EMSS) si pour tout état intial* $(\overline{x}_0, \theta_0)$, *il existe* $0 < \alpha < 1$ *et* $\beta > 0$ *tel que* $\forall i \geq 0, E\left[\|\overline{x}_i\|^2\right] < \beta\alpha^i \|\overline{x}_0\|^2$;
4. almost surely stable *si pour tout état intial* $(\overline{x}_0, \theta_0)$, $P\left[\lim_{i\to\infty} \|\overline{x}_i\| = 0\right] = 1$

Il est prouvé dans [Ji91] que pour le système décrit par les équations B.3, les trois premières définitions sont équivalentes et chacune implique *almost surely stable*. Le théorème obtenu dans [Costa93] qui donne les conditions nécessaires et suffisantes pour que le système soit MSS est énoncé comme suit :

Théorème 9 *Le système décrit par les équations B.3 est MSS si et seulement si il existe les matrices* $G_i > 0$ *pour* $i = 0, 1, .., N$ *qui satisfont l'une des conditions suivantes :*

1. $G_i - A_i^T \left(\sum_{j=1}^{N} p_{ij} G_j\right) A_i > 0$ *pour* $i = 0, 1, .., N$;
2. $G_j - A_j^T \left(\sum_{i=1}^{N} p_{ij} G_j\right) A_j^T > 0$ *pour* $j = 0, 1, .., N$;
3. $G_i - \sum_{j=1}^{N} p_{ij} A_j^T G_j A_j > 0$ *pour* $i = 0, 1, .., N$;
4. $G_j - \sum_{i=1}^{N} p_{ij} A_i G_i A_i^T > 0$ *pour* $i = 0, 1, .., N$.

B.3 Contrôle optimal

Considéron que la performance de contrôle du système B.3 est décrite par la fonction de coût suivante :

$$J = E\left(\sum_{i=0}^{N-1} \left(\overline{x}_i^T Q_{X_i} \overline{x}_i + 2\overline{x}_i^T M_{X_i} u_i + u_i^T R_{X_i} u_i\right)\right) \tag{B.7}$$

Le contrôleur optimal du système B.3 est celui qui minimise la fonction de coût B.7. En appliquant la programmation dynamique, la loi de commande optimale est dérivée dans [Blair75] comme suit :

$$u_i = -L_i x_i \quad i = 0, 1, 2... \tag{B.8}$$

où

$$L_i = \left(\bar{B}^T \sum_{j=1}^{n} K_{j,i+1} p_{X_i j} \bar{B} + R_{X_i} \right)^{-1} \left(\bar{B}^T \sum_{j=1}^{n} K_{j,i+1} p_{X_i j} \bar{A}_{v_i} + M_{X_i}^T \right) \tag{B.9}$$

où $K_{j,i+1}$ est obtenu de l'équation récurrente suivante :

$$\begin{array}{rcl} K_{X_N,N} & = & 0 \\ K_{l,i} & = & A_{v_l}^T \sum_{j=1}^{n} K_{j,i+1} p_{X_i j} \bar{A}_{v_l} + Q'_{X_i} \\ & = & -\left(B^T \sum_{j=1}^{n} K_{j,i+1} p_{X_i j} \bar{A}_{v_l} + M_{X_i}^T \right)^T \\ & & \times \left(\bar{B}^T \sum_{j=1}^{n} K_{j,i+1} p_{X_i j} \bar{B} + R'_{X_i} \right)^{-1} \\ & & \times \left(\bar{B}^T \sum_{j=1}^{n} K_{j,i+1} p_{X_i j} \bar{A}_{v_l} + M_{X_i}^T \right) \end{array} \tag{B.10}$$

avec $v_l = 1$ si la dernière lettre du (m,k) pattern à l'état l de la chaîne de Markov est '1', sinon $v_l = 0$.

La valeur minimale de la fonction de coût B.7 obtenue avec le contrôleur optimal B.8 est

$$J = \bar{x}_0^T K_{X_0,0} \bar{x}_0$$

Lorsque l'horizon temporel tend vers l'infini, $N = \infty$, en supposant que $K_{l,i} = K_{l,i+1} = K_{l,\infty}$ [3], l'équation B.9 devient :

$$L_i = \left(\bar{B}^T \sum_{j=1}^{n} K_{j,\infty} p_{X_i j} \bar{B} + R_{X_i} \right)^{-1} \left(\bar{B}^T \sum_{j=1}^{n} K_{j,\infty} p_{X_i j} \bar{A}_{v_i} + M_{X_i}^T \right)$$

avec

$$\begin{array}{rcl} K_{l,\infty} & = & A_{v_l}^T \sum_{j=1}^{n} K_{j,\infty} p_{X_i j} \bar{A}_{v_l} + Q'_{X_i} \\ & = & -\left(B^T \sum_{j=1}^{n} K_{j,\infty} p_{X_i j} \bar{A}_{v_l} + M_{X_i}^T \right)^T \\ & & \times \left(\bar{B}^T \sum_{j=1}^{n} K_{j,\infty} p_{X_i j} \bar{B} + R'_{X_i} \right)^{-1} \\ & & \times \left(\bar{B}^T \sum_{j=1}^{n} K_{j,\infty} p_{X_i j} \bar{A}_{v_l} + M_{X_i}^T \right) \end{array}$$

3. Dans [Chizeck86], des conditions nécessaires et suffisantes sont données pour déterminer si la solution converge lorsque l'horizon temporel tend vers l'infini. Cependant, il n'est pas facile de tester les conditions proposées. Dans [Ji88], les auteurs définissent les notions de la contrôlabilité et l'observabilité des systèmes hybrides linéaires d'une façon plus stricte, permettant ainsi d'effectuer des tests algébriques relativement simples pour déterminer la convergence de la solution.

Annexe C

Système monodimensionnel contrôlé en réseaux, expérimentations

Dans ces deux expériences, le processus à contrôler est représenté sous forme discrétisée, par $x_{i+1} = \alpha x_i + \beta u_i \quad i = 0, 1, ..,$ avec u_i et x_i respectivement l'entrée (la commande) et la sortie du système, α et β des constantes telles que $|\alpha| > 1$ et $\beta \neq 0$. Il est contrôlé en temps discret $z_i = \gamma y_i$ où γ est le gain et $y_i = x_i + p_i$ l'entrée du contrôleur. L'entrée du système est obtenue $u_i = z_i + q_i$. p_i et q_i sont des bruits blancs.

Les tableaux suivants donnent, pour chaque (m, k)-pattern admissible (couple (m, k) et pattern π), la valeur du gain optimal, γ_{opt} et la valeur correspondante de la borne supérieure de la variance de l'état du système, $M(\gamma_{opt}, \pi)$.

Etude de cas numéro 1

(1) Caractéristiques du système contrôlé :

- $\alpha = 3$, $\beta = 1$,
- les bruits blancs en entrée et sortie du système sont de variance 1 : $\sigma_p^2 = 1$ et $\sigma_q^2 = 1$.
- le contrôle du système doit respecter la contrainte suivante : la variance maximale supportée par ce système est $v_{max} = 10000$.

(2) Caractéristiques de l'architecture distribuée support :

- la mémoire disponible dans l'équipement gérant les rejets doit respecter une contrainte sur k, afin de pouvoir mémoriser (m, k)-pattern : $k \leq 10$,
- la bande passante réservée à l'application de contrôle distribuée impose une contrainte sur m et k : $\frac{m}{k} \leq 0.5$.

Les résultats sont présentés dans la table C.1. Ce tableau ne comprend que les patterns admissibles (répondant aux conditions précédentes) et tels qu'ils soient tous différents.

TABLE C.1 – Gain optimal, plus petite borne supérieure de la variance de l'état du système en fonction des (m,k)-pattern admissibles - Etude de cas numéro 1.

(m,k)	π	γ_{opt}	$M(\gamma_{opt},\pi)$		(m,k)	π	γ_{opt}	$M(\gamma_{opt},\pi)$
(1,2)	l0	-2,9667	9,0111E+01		(4,9)	ll00ll000	-2,8987	7,0802E+03
(1,3)	l00	-2,9956	8,1915E+02		(4,9)	ll0010100	-2,9696	8,1249E+02
(1,4)	l000	-2,9992	7,3810E+03		(4,9)	ll0010010	-2,9949	8,1924E+02
(2,4)	ll00	-2,8933	7,8596E+02		(4,9)	ll0001010	-2,8734	6,9924E+03
(2,5)	ll000	-2,9461	7,2122E+03		(4,9)	l01010100	-2,9696	8,1205E+02
(2,5)	l0100	-2,9692	8,1251E+02		(4,10)	lll0001000	-3,0000	7,3810E+03
(2,6)	l01000	-2,9800	7,3277E+03		(4,10)	ll01001000	-2,9962	7,3730E+03
(2,7)	l001000	-2,9962	7,3731E+03		(4,10)	ll01000100	-2,9654	7,3084E+03
(3,6)	lll000	-2,8533	6,9520E+03		(4,10)	ll00101000	-2,9692	7,3135E+03
(3,6)	ll0100	-2,9667	8,1193E+02		(4,10)	ll00100100	-2,9962	8,1911E+02
(3,6)	ll0010	-2,8600	7,7498E+02		(4,10)	ll00100010	-2,9962	7,3730E+03
(3,7)	ll01000	-2,9676	7,3085E+03		(4,10)	ll00010100	-2,9115	7,1052E+03
(3,7)	ll00100	-2,9954	8,1917E+02		(4,10)	ll00010010	-2,9038	7,0675E+03
(3,7)	ll00010	-2,9029	7,0724E+03		(4,10)	l0101010 00	-2,9654	7,3092E+03
(3,7)	l0101 00	-2,9676	8,1201E+02		(4,10)	l0101001 00	-2,9962	8,1911E+02
(3,8)	ll001000	-2,9968	7,3731E+03		(5,10)	llll001000	-2,9933	7,3783E+03
(3,8)	ll000100	-2,9359	7,1715E+03		(5,10)	llll000100	-2,9000	7,0202E+03
(3,8)	l0101000	-2,9680	7,3095E+03		(5,10)	lll0101000	-2,9667	7,3090E+03
(3,8)	l0100100	-2,9968	8,1912E+02		(5,10)	lll0100100	-2,9933	8,1971E+02
(3,9)	ll0001000	-3,0000	7,3810E+03		(5,10)	lll0100010	-2,9667	7,3083E+03
(3,9)	l01001000	-2,9956	7,3734E+03		(5,10)	lll00ll000	-2,8933	7,0739E+03
(3,9)	l01000100	-2,9733	7,3154E+03		(5,10)	lll0010100	-2,9733	8,1255E+02
(3,10)	l0100010 00	-3,0000	7,3810E+03		(5,10)	lll0010010	-2,9933	8,1970E+02
(3,10)	l00100100 0	-2,9969	7,3732E+03		(5,10)	lll0001100	-2,8133	6,8685E+03
(4,8)	lll01000	-2,9667	7,3083E+03		(5,10)	lll0001010	-2,8067	6,8472E+03
(4,8)	lll00100	-2,9933	8,1970E+02		(5,10)	ll01101000	-2,9667	7,3084E+03
(4,8)	lll00010	-2,8267	6,8860E+03		(5,10)	ll01100100	-2,9933	8,1970E+02
(4,8)	ll011000	-2,8400	6,9290E+03		(5,10)	ll01100010	-2,8267	6,9076E+03
(4,8)	ll0101 00	-2,9667	8,1200E+02		(5,10)	ll01011000	-2,8333	6,9200E+03
(4,8)	ll0100 10	-2,9667	8,1193E+02		(5,10)	ll01010100	-2,9667	8,1200E+02
(4,8)	ll001010	-2,8400	7,7068E+02		(5,10)	ll01010010	-2,9667	8,1200E+02
(4,9)	lll001000	-2,9949	7,3742E+03		(5,10)	ll01001100	-2,9667	8,1193E+02
(4,9)	lll000100	-2,8987	7,0216E+03		(5,10)	ll01001010	-2,9667	8,1193E+02
(4,9)	ll0101000	-2,9645	7,3094E+03		(5,10)	ll00110010	-2,8933	7,8589E+02
(4,9)	ll0100100	-2,9949	8,1924E+02		(5,10)	ll00101010	-2,8333	7,6906E+02
(4,9)	ll0100010	-2,9645	7,3086E+03					

Etude de cas numéro 2

(1) Caractéristiques du système contrôlé :

- $\alpha = 0,5117$, $\beta = 0,4883$,
- les bruits blancs en entrée et sortie du système sont de variance 1 : $\sigma_p^2 = 1$ et $\sigma_q^2 = 1$.
- le contrôle du système doit respecter la contrainte suivante : la variance maximale supportée par ce système est $v_{max} = 0,323$.

(2) Caractéristiques de l'architecture distribuée support :

- la mémoire disponible dans l'équipement gérant les rejets doit respecter une contrainte sur k, afin de pouvoir mémoriser (m, k)-pattern : $k \leq 10$,
- la bande passante réservée à l'application de contrôle distribuée impose une contrainte sur m et k : $\frac{m}{k} \leq 0.5$.

Les résultats sont présentés dans les tables C.2, C.4 et C.6. Ces tableaux ne comprennent que les pattern admissibles (répondant aux conditions précédentes) et tels qu'ils soient tous différents.

TABLE C.2 – Gain optimal, plus petite borne supérieure de la variance de l'état du système en fonction des (m, k)-pattern admissibles - Etude de cas numéro 2 (partie 1)

(m, k)	π	γ_{opt}	$M(\gamma_{opt}, \pi)$
(1,2)	l0	-0,2475	0,3174
(1,3)	l00	-0,2658	0,3216
(1,4)	l000	-0,1308	0,3227
(1,5)	l0000	-0,4505	0,3230
(2,4)	ll00	-0,2475	0,3214
(2,5)	ll000	-0,2646	0,3226
(2,5)	l0l00	-0,2646	0,3215
(2,6)	ll0000	-0,2658	0,3229
(2,6)	l0l000	-0,2658	0,3226
(2,7)	l0l0000	-0,1732	0,3229
(2,7)	l00l000	-0,1732	0,3227
(2,8)	l00l0000	-0,1308	0,3229
(2,9)	l00l00000	-0,1932	0,3229
(3,6)	lll000	-0,2475	0,3226
(3,6)	ll0l00	-0,2475	0,3215
(3,6)	ll00l0	-0,2475	0,3214
(3,7)	lll0000	-0,2473	0,3229
(3,7)	ll0l000	-0,2473	0,3226
(3,7)	ll00l00	-0,2473	0,3216
(3,7)	ll00l0l	-0,2473	0,3226
(3,7)	l0l0l00	-0,2473	0,3215
(3,8)	ll0l0000	-0,2972	0,3229
(3,8)	ll00l000	-0,2972	0,3226
(3,8)	ll000l00	-0,2972	0,3226
(3,8)	ll000l0l	-0,2972	0,3229
(3,8)	l0l0l000	-0,2972	0,3226
(3,8)	l0l00l00	-0,2972	0,3216
(3,9)	ll00l0000	-0,2658	0,3229
(3,9)	ll000l000	-0,2658	0,3226
(3,9)	ll0000l00	-0,2658	0,3229
(3,9)	l0l0l0000	-0,2658	0,3229
(3,9)	l0l00l000	-0,2658	0,3226
(3,9)	l0l000l00	-0,2658	0,3226

TABLE C.4 – Gain optimal, plus petite borne supérieure de la variance de l'état du système en fonction des (m, k)-pattern admissibles - Etude de cas numéro 2 (partie 2)

(m, k)	π	γ_{opt}	$M(\gamma_{opt}, \pi)$
(3,9)	l01000100	-0,2658	0,3226
(3,10)	l100010000	-0,2656	0,3229
(3,10)	l100001000	-0,2656	0,3229
(3,10)	l010010000	-0,2656	0,3229
(3,10)	l010001000	-0,2656	0,3226
(3,10)	l010000100	-0,2656	0,3229
(3,10)	l001001000	-0,2656	0,3226
(4,8)	llll0000	-0,2475	0,3229
(4,8)	lll01000	-0,2475	0,3226
(4,8)	lll00100	-0,2475	0,3216
(4,8)	lll00010	-0,2475	0,3226
(4,8)	ll011000	-0,2475	0,3226
(4,8)	ll010100	-0,2475	0,3215
(4,8)	ll010010	-0,2475	0,3215
(4,8)	ll001010	-0,2475	0,3214
(4,9)	lll010000	-0,2908	0,3229
(4,9)	lll001000	-0,2908	0,3226
(4,9)	lll000100	-0,1962	0,3226
(4,9)	lll000010	-0,1962	0,3229
(4,9)	l10110000	-0,2908	0,3229
(4,9)	l10101000	-0,2908	0,3226
(4,9)	l10100100	-0,2908	0,3216
(4,9)	l10100010	-0,2908	0,3226
(4,9)	l10011000	-0,2908	0,3226
(4,9)	l10010100	-0,2908	0,3216
(4,9)	l10010010	-0,2908	0,3216
(4,9)	l10001010	-0,2908	0,3226
(4,9)	l01010100	-0,2908	0,3216
(4,10)	lll10010000	-0,2646	0,3229
(4,10)	lll10001000	-0,2646	0,3226
(4,10)	lll10000100	-0,2646	0,3229
(4,10)	l101010000	-0,2646	0,3229
(4,10)	l101001000	-0,2646	0,3226
(4,10)	l101000100	-0,2646	0,3226
(4,10)	l101000010	-0,2646	0,3229
(4,10)	l100110000	-0,2646	0,3229
(4,10)	l100101000	-0,2646	0,3226
(4,10)	l100100100	-0,2646	0,3216

TABLE C.6 – Gain optimal, plus petite borne supérieure de la variance de l'état du système en fonction des (m, k)-pattern admissibles - Etude de cas numéro 2 (partie 3)

(m,k)	π	γ_{opt}	$M(\gamma_{opt}, \pi)$
(4,10)	1100100010	-0,2646	0,3226
(4,10)	1100010100	-0,2646	0,3226
(4,10)	1100010010	-0,2646	0,3226
(4,10)	1100001010	-0,2646	0,3229
(4,10)	1010101000	-0,2646	0,3226
(4,10)	1010100100	-0,2646	0,3216
(5,10)	1111010000	-0,2475	0,3229
(5,10)	1111001000	-0,2475	0,3226
(5,10)	1111000100	-0,2475	0,3226
(5,10)	1111000010	-0,2475	0,3229
(5,10)	1110110000	-0,2475	0,3229
(5,10)	1110101000	-0,2475	0,3226
(5,10)	1110100100	-0,2475	0,3216
(5,10)	1110100010	-0,2475	0,3226
(5,10)	1110011000	-0,2475	0,3226
(5,10)	1110010100	-0,2475	0,3215
(5,10)	1110010010	-0,2475	0,3216
(5,10)	1110001100	-0,2475	0,3226
(5,10)	1110001010	-0,2475	0,3226
(5,10)	1110000110	-0,2475	0,3229
(5,10)	1101101000	-0,2475	0,3226
(5,10)	1101100100	-0,2475	0,3216
(5,10)	1101100010	-0,2475	0,3226
(5,10)	1101011000	-0,2475	0,3226
(5,10)	1101010100	-0,2475	0,3215
(5,10)	1101010010	-0,2475	0,3215
(5,10)	1101001100	-0,2475	0,3215
(5,10)	1101001010	-0,2475	0,3215
(5,10)	1100110010	-0,2475	0,3214
(5,10)	1100101010	-0,2475	0,3214

Annexe D

Calcul de la dérivée de fonction de coût

Cette annexe est un complément du chapitre 5. Nous donnerons ici la dérivée de la fonction de coût 5.9.

Rappelons pour mémoire la fonction de coût stationnaire suivante :

$$J_p(f_0, f_1, .., f_{m-1}) = \sum_{i=0}^{m-1} (trS_{i+1}(f_0, f_1, .., f_{m-1}) R_1(f_i)) + \sum_{i=0}^{m-1} \bar{J}_i(f_0, f_1, .., f_{m-1}) \quad \text{(D.1)}$$

Soit $F = \{f_i\}_{i=0,1..m-1}$, la dérivée par rapport à f_i au premier ordre de la fonction D.1 pour $i \in [0, m-1]$ est :

$$\frac{dJ_p(F)}{df_i} = \sum_{i=0}^{m-1} tr \frac{dS_{j+1}(F)}{df_i} R_1(f_i) + trS_{i+1} \frac{dR_1(F)}{df_i} + \frac{d\bar{J}_i(F)}{df_i} \quad \text{(D.2)}$$

Dans la suite, nous montrons comment sont calculés les terms $\frac{dS_{j+1}(F)}{df_i}$, $R_1(f_i)$, $\frac{dR_1(f_i h)}{df_i}$ et $\frac{d\bar{J}_i(F)}{df_i}$ de la fonction D.2.

D.1 Calcule de $\frac{dR_1(f_i h)}{df_i}$ et $\frac{d\bar{J}_i(F)}{df_i}$

Les dérivées de $R_1(f_i)$ et $\bar{J}_i$ en fonction de f_1 sont calculées directement des équations 4.4 et 5.3, on obtient :

$$\frac{dR_1(f_i h)}{df_i} = he^{Af_i} R_c e^{A^T f_i} \quad \text{(D.3)}$$

$$\frac{d\bar{J}_i(F)}{df_i} = tr(QR_1(f_i) h) \quad \text{(D.4)}$$

D.2 Calcule de $R_1(f_i)$

Pour calculer l'intégrale $R_1(f_i) = \int_0^t e^{As} Q e^{A^T s} ds$, considérons le système linéaire de [Loan78] :

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -A & Q \\ 0 & A^T \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{D.5}$$

qui a la solution suivante :

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \exp\left(\begin{bmatrix} -A & Q \\ 0 & A^T \end{bmatrix} t \right) \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} \tag{D.6}$$

Soit

$$\begin{bmatrix} \Psi_{11} & \Psi_{12} \\ \Psi_{21} & \Psi_{22} \end{bmatrix} = \exp\left(\begin{bmatrix} -A & Q \\ 0 & A^T \end{bmatrix} t \right)$$

alors on a :

$$x_1(t) = \Psi_{11} x_1(0) + \Psi_{12} x_2(0) \tag{D.7}$$
$$x_2(t) = \Psi_{21} x_1(0) + \Psi_{22} x_2(0) \tag{D.8}$$

De D.5 et D.6, on a respectivement

$$\dot{x_1}(t) = -A x_1(t) + Q x_2(t) \tag{D.9}$$

et

$$x_2(t) = e^{A^T t} x_2(0) \tag{D.10}$$

on a donc

$$\dot{x_1}(t) = -A x_1(t) + Q e^{A^T t} x_2(0) \tag{D.11}$$

L'intégration de l'équation D.11 nous donne

$$x_1(t) = e^{-At}x_1(0) + e^{-At}\int_0^t e^{As}Qe^{A^Ts}dsx_2(0) \tag{D.12}$$

En identifiant les termes dans les équations D.7 et D.8 par les équations D.12 et D.10, on obtient

$$\begin{aligned}\Psi_{11} &= e^{-At}\\ \Psi_{12} &= e^{-At}\int_0^t e^{As}Qe^{A^Ts}\\ \Psi_{21} &= 0\\ \Psi_{22} &= e^{A^Tt}\end{aligned}$$

il vient ainsi :

$$R_1(t) = \int_0^t e^{As}Qe^{A^Ts}ds = \Psi_{22}^T\Psi_{12}$$

D.3 Calcule de $\frac{dS_{j+1}(F)}{df_i}$

De 5.4 et 5.5, on a

$$\left(\Gamma_i^T S_{i+1}\Gamma_i + R_i^{'}\right)L_i = \left(\Gamma_i^T S_{i+1}\Phi_i + M_i^T\right) \tag{D.13}$$

et

$$S_i + L_i^T\left(\Gamma_i^T S_{i+1}\Gamma_i + R_i^{'}\right)L_i = \left(\Phi_i^T S_{i+1}\Phi_i + Q_i^{'}\right)^T \tag{D.14}$$

En groupant les équations D.13 et D.14 dans une matrice, on a

$$\begin{bmatrix} S_i + L_i^T G_i L_i & L_i^T G_i \\ G_i L_i & G_i \end{bmatrix} = \begin{bmatrix} \Phi_i^T \\ \Gamma_i^T \end{bmatrix} S_{i+1} \begin{bmatrix} \Phi_i & \Gamma_i \end{bmatrix} + Q_{i,d} \tag{D.15}$$

où $G_i = \Gamma_i^T S_{i+1}\Gamma_i + R_i^{'}$ avec

$$Q_{i,d} = \begin{bmatrix} Q_{i,1d} & Q_{i,2d} \\ Q_{i,3d} & Q_{i,4d} \end{bmatrix} = \int_0^{f_i h} e^{\sum^t} dt$$

où

$$e^{\sum^t} = \exp\left(\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} t\right) = \begin{bmatrix} \Phi(t) & \Gamma(t) \\ 0 & I \end{bmatrix}$$

Ensuite, on différencie l'équation D.15 en fonction de f_i et on obtient :

$$\begin{bmatrix} 0 & 0 \\ \frac{dL_i}{df_i} & 0 \end{bmatrix}^T \begin{bmatrix} S_i & 0 \\ 0 & G_i \end{bmatrix} \begin{bmatrix} I & 0 \\ L_i & I \end{bmatrix} + \begin{bmatrix} I & 0 \\ L_i & I \end{bmatrix}^T \begin{bmatrix} \frac{dS_i(F)}{df_i} & 0 \\ 0 & \frac{dG_i}{df_i} \end{bmatrix} \begin{bmatrix} I & 0 \\ L_i & I \end{bmatrix} =$$
$$\frac{dQ_{i,d}}{df_i} + \begin{bmatrix} \frac{d\Phi_i^T}{df_i} \\ \frac{d\Gamma_i^T}{df_i} \end{bmatrix} S_{i+1} \begin{bmatrix} \Phi_i & \Gamma_i \end{bmatrix} + \begin{bmatrix} \Phi_i^T \\ \Gamma_i^T \end{bmatrix} \frac{dS_{i+1}(F)}{df_i} \begin{bmatrix} \Phi_i & \Gamma_i \end{bmatrix} + \begin{bmatrix} \Phi_i^T \\ \Gamma_i^T \end{bmatrix} S_{i+1} \begin{bmatrix} \frac{d\Phi_i}{df_i} & \frac{d\Gamma_i}{df_i} \end{bmatrix}$$

En regroupant les termes, il vient :

$$\begin{bmatrix} 0 & \frac{dL_i^T}{df_i} G \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \frac{dS_i(F)}{df_i} & 0 \\ 0 & \frac{dG_i}{df_i} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ G\frac{dL_i}{df_i} & 0 \end{bmatrix} =$$
$$\begin{bmatrix} \Phi_i^T - L_i^T \Gamma_i^T \\ \Gamma_i^T \end{bmatrix} \frac{dS_{i+1}(F)}{df_i} \begin{bmatrix} \Phi_i - \Gamma_i L_i & \Gamma_i \end{bmatrix} + \begin{bmatrix} I & 0 \\ -L_i & I \end{bmatrix}^T \overline{w_i} \begin{bmatrix} I & 0 \\ -L_i & I \end{bmatrix} \tag{D.16}$$

où

$$\overline{w_i} = \frac{dQ_{i,d}}{df_i} + \begin{bmatrix} \frac{d\Phi_i^T}{df_i} \\ \frac{d\Gamma_i^T}{df_i} \end{bmatrix} S_{i+1} \begin{bmatrix} \Phi_i & \Gamma_i \end{bmatrix} + \begin{bmatrix} \Phi_i^T \\ \Gamma_i^T \end{bmatrix} S_{i+1} \begin{bmatrix} \frac{d\Phi_i}{df_i} & \frac{d\Gamma_i}{df_i} \end{bmatrix}$$

De D.16, on arrive à obtenir $\frac{dS_i(F)}{df_i}$:

$$\frac{dS_i(F)}{df_i} = \Psi_i^T \frac{dS_{i+1}(F)}{df_i} \Psi_i + \begin{bmatrix} I & -L_i^T \end{bmatrix} \overline{w_i} \begin{bmatrix} I \\ -L_i \end{bmatrix}$$

où $\Psi_i = \Phi_i - \Gamma_i L_i$.

Par la même approche, on obtient aussi :

$$\frac{dS_j(F)}{df_i} = \Psi_{j+1}^T \frac{dS_{j+1}(F)}{df_i} \Psi_{j+1} \quad \text{pour mod}\left(\frac{j}{m}\right) \neq i$$

Les termes $\frac{dQ_{i,d}}{df_i}, \frac{d\Phi_i}{df_i}$ et $\frac{d\Gamma_i}{df_i}$ dans $\overline{w_i}$ sont calculés comme suit :

$$\begin{aligned} \frac{d\Phi_i}{df_i} &= hA\Phi_i \\ \frac{d\Gamma_i}{df_i} &= hA\Gamma_i + hB \\ \frac{dQ_{i,d}}{df_i} &= he^{\sum^T f_i h} \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} e^{\sum f_i h} = h \begin{bmatrix} \Phi_i & \Gamma_i \\ 0 & I \end{bmatrix}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} \Phi_i & \Gamma_i \\ 0 & I \end{bmatrix} \end{aligned}$$

$\overline{w_i}$ peut maintenant être écrit comme suit :

$$\begin{aligned}\overline{w_i} &= h\begin{bmatrix}\Phi_i & \Gamma_i\\ 0 & I\end{bmatrix}^T\begin{bmatrix}Q & 0\\ 0 & R\end{bmatrix}\begin{bmatrix}\Phi_i & \Gamma_i\\ 0 & I\end{bmatrix}\\ &+\begin{bmatrix}(hA\Phi_i)^T\\ (hA\Gamma_i+hB)^T\end{bmatrix}S_{i+1}\begin{bmatrix}\Phi_i & \Gamma_i\end{bmatrix}+\begin{bmatrix}\Phi_i^T\\ \Gamma_i^T\end{bmatrix}S_{i+1}\begin{bmatrix}Ah\Phi_i & hA\Gamma_i+hB\end{bmatrix}\end{aligned}$$

Pour raccourcir $\frac{dS_i(F)}{df_i}$, on pose :

$$\begin{aligned}W_i &= \begin{bmatrix}I & -L_i^T\end{bmatrix}\overline{w_i}\begin{bmatrix}I\\ -L_i\end{bmatrix}\\ &= h\begin{bmatrix}\Psi_i^T & -L_i^T\end{bmatrix}\begin{bmatrix}\Phi_i & \Gamma_i\\ 0 & I\end{bmatrix}\begin{bmatrix}\Psi_i\\ -L_i\end{bmatrix}\\ &\quad+\left(h\Psi_i^TA^T-hL_i^TB^T\right)S_{i+1}\Psi_i+\Psi_i^TS_{i+1}\left(hA\Psi_i-hBL_i\right)\end{aligned}\tag{D.17}$$

et il vient :

$$\frac{dS_i(F)}{df_i}=\Psi_i^T\frac{dS_{i+1}(F)}{df_i}\Psi_i+W_i$$

En résumé, $\frac{dS_{j+1}(F)}{df_i}$ est calculé à partir de l'équation récurrente suivante :

$$\frac{dS_j(F)}{df_i}=\left(\Phi_{j+1}-\Gamma_{j+1}L_{j+1}\right)^T\frac{dS_{j+1}(F)}{df_i}\left(\Phi_{j+1}-\Gamma_{j+1}L_{j+1}\right)\quad\text{pour mod}\left(\frac{j}{m}\right)\neq i$$

avec

$$\frac{dS_i(F)}{df_i}=\left(\Phi_{i+1}-\Gamma_{i+1}L_{i+1}\right)^T\frac{dS_{i+1}(F)}{df_i}\left(\Phi_{i+1}-\Gamma_{i+1}L_{i+1}\right)+W_i$$

où W_i est caluclé selon D.17. Jia05b Jia06a Jia06b Jia07a Jia07b Jia08

Table des figures

Liste des tableaux

Bibliographie

[Anderson99] Don Anderson. *FireWire system architecture (2nd ed.) : IEEE 1394a*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[Andrew02] Tanenbaum Andrew. *Computer Networks*. Prentice Hall Professional Technical Reference, 2002.

[Astrom97] K. J. Astrom and B. Wittenmark. *ComputerControlled Systems*. Prentice Hall, 1997.

[Baruah90] S. K. Baruah, A. K. Mok, and L. E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *IEEE Real-Time Systems Symposium*, pages 182–190, 1990.

[BenGaid05] M-M. Ben Gaid, A. Çela, and Y. Hamam. Optimal integrated control and scheduling of systems with communication constraints. In *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*, Seville, Spain, December 2005. To appear.

[BenGaid06a] M-M. Ben Gaid and Y. Çela, A.and Hamam. Optimal integrated control and scheduling of networked control systems with communication constraints : Application to a car suspension system. *IEEE Transactions on Control Systems Technology*, 14(4) :776–787, 2006.

[BenGaid06b] M. Ben Gaid. *Optimal Scheduling and Control for Distributed Real-Time Systems*. Ph.d. thesis, Universit¨¦ d'Evry Val d'Essonne, 2006.

[Bittanti91] S. Bittanti, P. Colaneri, and G. De Nicolao. *The Riccati Equation*, chapter The periodic Riccati equation, pages 127–162. Springer-Verlag, Berlin, 1991.

[Blair75] W. P. Jr. Blair and D.D. Sworder. Feedback control of a class of linear discrete systems with jump parameters and quadratic cost criteria. In *Int. J. Control*, volume 21, pages 838–841, 1975.

[Borchers94] B. Borchers and J.E. Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programming. In *Computers and Operations Research*, number 21, pages 359–367, 1994.

[Branicky00] M.S. Branicky, S.M. Phillips, and W. Zhang. Stability of networked control systems : Explicit analysis of delay. In *in Proc. American Control Conf*, pages 2352–2357, Chicago, Jun 2000.

[Bur94] A. Burns. Preemptive priority based scheduling : An appropriate engineering approach. In Prentice Hall, editor, *Advances in RealTime Systems*, 1973.

[Cervin02] A. Cervin, J. Eker, B. Bernhardsson, and K. E. Årzén. Feedbackfeedforward scheduling of control tasks. In *Real-Time Syst*, volume 23, pages 25–53, 2002.

[Cervin03] A. Cervin, D. Henriksson, Bo. Lincoln, J. Eker, B. Bernhardsson, and K. E. Årzén. How does control timing affect performance ? *IEEE Control Systems Magazine*, 23(3) :16–30, June 2003.

[Chan95] H. Chan and U. Ozguner. Closed loop control of systems over a communications network with queues. In *International Journal of Control*, volume 62, pages 493–510, 1995.

[Chizeck86] H.J. Chizeck, A.S. Wilsky, and D. Castanon. Discrete-time markovian jump linear quadratic optimal control. In *Int. J. Control*, volume 43, pages 213–231, 1986.

[Coffman76] E. G. Coffman and Jr. Introduction to deterministic scheduling theory. In *Computer and Job-Shop Scheduling Theory*, Wiley, New York, 1976.

[Costa93] O. L. do Valle Costa and M. D. Fragoso. Stability results for discrete-time linear-systems with markovian jumping parameters. In *Journal of Mathematical Analysis and Applications*, volume 179, pages 154–178, Oct 1993.

[Dakin65] R. J. Dakin. A tree search algorithm for mixed-integer programming problems. *Computer Journal*, (8) :250–255, 1965.

[Duranand86] M. A. Duranand I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Math Programming*, (36) :307.

[Eker00] J. Eker, P. Hagander, and K. E. Årzén. A feedback scheduler for real time control tasks. In *Control Engineering Practice*, pages 1369–1378.

[Felicioni08] F. Felicioni, N. Jia, F. Simonot, and Y.Q. Song. Optimal on-line (m,k)-firm constraint assignment for real-time control tasks based on plant state information. In *13th IEEE ETFA*, Hamburg, Germany, 2008.

[Fletcher94] R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Math Programming*, (66) :327, 1994.

[Freeman63] H . FREEMAN. *Discret time Systems.* John Wiley, 1963.

[Garfinkel72] R. S. Garfinkel and G. L. Nemhauser. *Integer Programming.* John Wiley, New York, 1972.

[Geoffrion72] A. M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, (10(4)) :237–260, 1972.

[George95] L. George, P. Muhletahler, and N. Rivierre. Optimality and nonpreemptive real-time scheduling revisited. Technical report 2516, INRIA, 1995.

[George96] L. George, N. Rivierre, and M. Spuri. Preemptive and non-preemptive real-time uniprocessor scheduling. Technical report 2966, INRIA Rocquencourt, Sep 1996.

[Gerber95] R. Gerber, S. Hong, and M. Saksena. Guaranteeing realtime requirements with resourcebased calibration of periodic processes. In *IEEE Trans on Software Engineering*, number 21 :7.

[Gupta04] Vijay. Gupta, Demetri. Spanos, Babak. Hassibi., and Richard M. Murray. Optimal LQG control across a packet-dropping link. In *Submitted, IEEE Transactions on Automatic Control*, 2004.

[Gupta05] Vijay Gupta, Demetri Spanos, Babak Hassibi, and Richard M. Murray. On LQG control across a stochastic packet-dropping link. In *Submitted, 2005 American Control Conference (ACC)*, 2005.

[Gupta85] O. K. Gupta and V. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, (31(12)) :1533–1546, 1985.

[Gustafsson91] Kjell Gustafsson and Per Hagander. Discrete-time LQG with cross-terms in the loss function and the noise description. Technical report tfrt-7475, Department of Automatic Control, Lund University, Sweden, April 1991.

[Hadjicostis02] C. Hadjicostis and R. Touri. Feedback control utilizing packet dropping network links. In *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, NV, 2002.

[Hahn67] W. Hahn. Stability of motion. Berlin, 1967.

[Halevi88] Y. Halevi and A. Ray. Integrated communication and control systems : Part i-analysis. *Journal of Dynamic Systems, Measurement, and Control*, pages 367–373, 1988.

[Hamdaoui94] M. Hamdaoui and P. Ramanathan. A service policy for real-time customers with (m,k)-firm deadlines. In *Proceedings of the Fault-Tolerant Computing Symposium*, pages 196–205, 1994.

[Hamdaoui95] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m,k)-firm deadlines. In *IEEE Transactions on Computers*, pages 1443–1451, Dec 1995.

[Hamdaoui97] M. Hamdaoui and P. Ramanathan. Evaluating dynamic failure probability for streams with (m, k)-firm deadline. In *IEEE Transactions on Computers*, number 46(12), pages 1325–1337, Dec 1997.

[Hassibi99] A. Hassibi, S. Boyd, and J. How. Control of asynchronous dynamical systems with rate constraints on events. In *in Proc. IEEE Conf. Decision and Control*, pages 1345–1351, Phoenix, AZ, Dec 1999.

[Henriksson05] D. Henriksson and A. Cervin. Optimal on-line sampling period assignment for real-time control tasks based on plant state information. In *Proceedings of the*

44th IEEE Conference on Decision and Control and European Control Conference ECC 2005, number 2005, Seville, Spain, Dec.

[Hetel06] Laurentiu Hetel, Jamal Daafouz, and Claude Iung. Stabilization of switched linear systems with unknown time varying delays. In *the 2nd IFAC Conf. on Analysis and Design of Hybrid Systems*, Alghero, Italy, Jun 2006.

[Jeffay91] K. Jeffay, D. F. Stanat, and C. U. Martel. On non-preemptive scheduling of periodic and sporadic tasks. In *Proceedings of the 12 th IEEE Symposium on Real-Time Systems*, pages 129–139, Dec 1991.

[Ji88] Y. Ji and H. J. Chezick. Controllability, observability and discrete-time markovian jump linear quadratic control. In *Int. J. Control*, volume 48, pages 481–498, 1988.

[Ji91] Y. Ji, H. J. Chizeck, X. Feng, and K. A. Loparo. Stability and control of discrete-time jump linear systems. In *Control Theory and Advanced Technology*, volume 7, pages 247–270, Jun 1991.

[Jia05a] N. Jia, H. Hyon, and Y.Q. Song. Ordonnancement sous contraintes (m,k)-firm et combinatoire des mots. In *13th International Conference on Real-Time Systems*, Paris, France, 2005.

[Jia05b] N. Jia, Y.Q. Song, and R.Z Lin. Analysis of networked control system with packet drops governed by (m,k)-firm constraint. In *The 6th IFAC International Conference on Fieldbus Systems and their Applications (FET'05)*, Mexique, 2005.

[Jia06a] F. Felicioni, N. Jia, Y.Q. Song, and F. Simonot-Lion. Impact of a (m,k)-firm data dropouts policy on the quality of control. In *6th IEEE International Workshop on Factory Communication Systems*, Turin, 2006.

[Jia06b] N. Jia, Y.Q. Song, and F. Simonot-Lion. Optimal lq-controller design and data drop distribution under (m,k)-firm constraint. In *4th Workshop on Advanced Control and Diagnosis (ACD'06)*, Nancy, France, 2006.

[Jia07a] N. Jia, Y.Q. Song, and F. Simonot-Lion. Feedback scheduling based on (m,k)-firm constraint model for the handling of a set of real-time controllers. In *15th International Conference on Real-Time and Network Systems (RTNS'07)*, Nancy, France, 2007.

[Jia07b] N. Jia, Y.Q. Song, and F. Simonot-Lion. Graceful degradation of the quality of control through data drop policy. In *European Control Conference 2007 (ECC'07)*, Kos, Greece, 2007.

[Jia08] F. Felicioni, N. Jia, Y.Q. Song, and F. Simonot-Lion. Optimal on-line (m,k)-firm constraint assignment for real-time control tasks based on plant state information. In *Emerging Technologies and Factory Automation (ETFA'08)*, Hamburg, 2008.

[Joseph86] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, (29(5) :390395,), 1986.

[Juanole07] Guy Juanole and Gérard Mouney. Using an hybrid traffic scheduling in networked control systems. presentation reunion safe–necs, Kos (Grèce), Jul 2007.

[Kailath99] T. Kailath. Linear estimation. 1999.

[Kaneko96] H. Kaneko, J. A. Stankovic, S. Sen, and K. Ramamritham. Integrated scheduling of multimedia and hard real-time tasks. Technical report umcs-1996-045, University of Massachusetts, Amherst, Dec 1996.

[Khan02] P. Khan, K. Li, E. Manning, and M. Akbar. Solving the knapsack problem for adaptive multimedia system. In *Studia Informatica*, volume 2, 2002.

[Khan98] S. Khan. *Quality Adaptation in a Multi-Session Adaptive Multimedia System : Model and Architecture.* Phd thesis, Department of Electrical and Computer Engineering, University of Victoria, May 1998.

[Kopetz92] H. Kopetz, G. Fohler, G. Grunsteidl, H. Kantz, G. Pospischil, P. Puschner J. Reisinger, R. Schlatterbeck, W. Schutz, A. Vrchotichy, and R. Zainlinger. The programmer's view of mars. In *IEEE Computer Society Press*, pages 223–226, Phoenix,Arizona, USA, Dec 1992.

[Koren96] G. Koren and D. Shasha. Skip-over : Algorithms and complexity for overloaded systems that allow skips. Technical report, New York, NY, USA, 1996.

[Kuhn51] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *2nd Berkeley Symposium*, pages 481–492, Berkeley, 1951.

[Kuo87] C. Kuo. Benjamin. *Automatic control systems (5th ed.).* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

[Land60] A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, (28(3)) :497–520, 1960.

[Lawrenz97] Wolfhard Lawrenz and W. Lawrenz. *Can System Engineering : From Theory to Practical Applications.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[Lemmon03] Michael Lemmon, Qiang Ling, and Yashan Sun. Overload management in sensor-actuator networks used for spatially–distributed control systems. In *SenSys '03 : Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 162–170, New York, NY, USA, 2003.

[Leyffer98] S. Leyffer. Integrating sqp and branch and bound for mixed integer noninear programming. In *submitted for publication*, 1998.

[Li02] Zhi Wang Shanbin Li and Youxian Sun. Fundamental problems of networked control system from the view of control and scheduling. In *28 th Annual Conference of IEEE Industrial Electronics*, pages 2503–2508, 2002.

[Li03] J. Li. Sufficient condition for guaranteeing (m,k)-firm real-time requirement under np-dbp-edf scheduling. Master report in computer science, LORIA, 2003.

[Lincoln00] B. Lincoln and B. Bernhardsson. Efficient pruning of search trees in lqr control of switched linear systems. In *Proceedings of the Conference on Decision and Control.*

[Lincoln01] B. Lincoln and A. Rantzer. Optimizing linear system switching. In *Proc. 40th IEEE Conference on Decision and Control*, Dec 2001.

[Lincoln02] B. Lincoln and Bo Bernhardsson. Lqr optimization of linear system switching. In *IEEE Transactions on Automatic Control*, Oct 2002.

[Lindsay99] W. Lindsay and P. Ramanathan. Dbp-m, a technique for meeting end-to-end (m,k)-firm guarantee requirements in point-to-point networks. In *IEEE Conference on Local networks*, pages 294–303, 1999.

[Ling02] Q. Ling and M. D. Lemmon. Robust performance of soft real-time networked control systems with data dropouts. In *Proceedings of 41st Conference on Decision and Control*, pages 1225–1230, Las Vegas, 2002.

[Ling03] Q. Ling and M. D. Lemmon. Soft real-time scheduling of networked systems with dropouts governed by a markov chain. In *In Proc. of the American Conference on Control*, 2003.

[Liou90] L. W. Liou and A. Ray. Integrated communication and control systems : Part iii–nonidentical sensor and controller sampling. In *Journal of Dynamic Systems, Measurement, and Control*, number 112, pages 357–364, 1990.

[Liu03] Donglin Liu, Michael D. Lemmon, and Qiang Ling. Firm real-time system scheduling based on a novel qos constraint. In *Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS03)*, pages 386–395, Mexico, 2003. Senior Member–Xiaobo Sharon Hu.

[Liu73] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1) :4661,, 1973.

[Loan78] C. Van Loan. Computing integral involving the matrix exponential. In *IEEE Transactions on Automatic Control*, number DAC 3E, pages 395–404, 1978.

[Lothaire02] M. Lothaire. Algebraic combinatorics on words. *Cambridge University Press*, 2002.

[Luck90] Rogelio Luck and Asok Ray. An observer-based compensator for distributed delays. *Automatica*, 26(5) :903–908, 1990.

[Luck94] R. Luck and A. Ray. Experimental verification of a delay compensation algorithm for integrated communication and control systems. In *International Journal of Control*, volume 59, pages 1357–1372, Jun 1994.

[Lyap07] A. M. Lyapunov. Probleme general de la stabilite du mouvement. In *Annales de la Faculte des sciences de Toulouse*, volume 9, pages 203–474, 1907. traduction en fraçais du mémoire Russe, Ob–shchaya Zadacha Ustoichivosti dvizheniya,

Kharkov, 1892, et d'une note, Comm. Soc. Math Kharkov, Vol. 3, pp. 265–272, 1983.

[Martello90] Silvano Martello and Paolo Toth. *Knapsack problems : algorithms and computer implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[Marti04] P. Marti, J. Yépez, M. Velasco, R. Villa, and J.M. Fuertes. Managing quality of control in network based control systems by controller and message scheduling co–design. In *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, volume 50, pages 1159–1167. ISSN : 0278–0046.

[Nabar91] S. Nabar and L. Schrage. Modeling and solving nonlinear integer programming problems. In *Presented at Annual AIChE Meeting*, Chicago, 1991.

[Nilsson98] J. Nilsson. *Real-Time Control Systems with Delays*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1998.

[Palopoli02] L. Palopoli, C. Pinello, A. SangiovanniVincentelli, L. ElGhaoui, and A. Bicchi. Synthesis of robust control systems under resource constraints. In *Proceedings of the Workshop on Hybrid Systems : Computation and Control*, 2002.

[Park81] P. C. Parks and V. Hahn. Stabilitats theorie. Berlin, 1981.

[Parra-Hernandez05] Rafael Parra-Hernandez and Nikitas J. Dimopoulos. A new heuristic for solving the multichoice multidimensional knapsack problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(5) :708–717, 2005.

[Pisinger95] D. Pisinger. *Algorithms for Knapsack Problems*. Ph.d. thesis, University of Copenhagen, 1995.

[Poggi03] E. M. Poggi, Y. Q. Song, A. Koubaa, and Z. Wang. Matrix-dbp for (m,k)-firm real-time guarantee. In *Conference of Real Time Systems, Paris (France)*, pages 457–482, 4 2003.

[Quan00] G. Quan and X. Hu. Enhanced fixed-priority scheduling with (m,k)-firm guarantee. In *Proceedings of the 21st IEEE Real-Time Systems Symposium*, pages 79–88, Orlando, Florida (USA), Nev 2000.

[Quesada92] I. Quesada and I.E. Grossmann. An lp/nlp based branch and bound algorithm for convex minlp optimization problems. *Computers and Chemical Engineering*, (16) :937–947, 1992.

[Ramanathan99] P. Ramanathan. Overload management in real-time control applications using (m,k)-firm guarantee. In *IEEE Transactions on Parallel and Distributed Systems*, volume 10, pages 549–559, Jun 1999.

[Ray87] A. RAY. Performance evaluation of medium access control protocols for distributed digital avionics. In *Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control*, number 109, pages 370–377, Dec 1987.

[Rehbinder00] H. Rehbinder and M. Sanfridson. Integration of offline scheduling and optimal control. In *Proceedings of the 12th Euromicro Conference on RealTime Systems*.

[Ryu97] M. Ryu, S. Hong, and M. Saksena. Streamlining realtime controller design : From performance specifications to endtoend timing constraints. In *Proceedings of the 3rd IEEE RealTime Technology and Applications Symposium*, pages 91–99, 1997.

[Ryu98] M. Ryu and S. Hong. Toward automatic synthesis of schedulable realtime controllers. In *Integrated ComputerAided Engineering*, number 5 :3, pages 261–277, 1998.

[Schinkel02] M. Schinkel, W. H. Chen, and Anders Rantzer. Optimal control for systems with varying sampling rate. In *Proceedings of the american control conference*, number 8-10, Anchorage, AK, May 2002.

[Seiler01] P. Seiler and R. Sengupta. Analysis of communication losses in vehicle control problems. In *Proc. Of ACC 2001, the 2001 American Control Conference*, pages 1491–1496, Arlington, VA, 2001.

[Seto96] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin. On task schedulability in real–time control system. In *In Proceedings of the 17th IEEE RealTime Systems Symposium*, pages 13–21, Washington, DC, Dec 1996.

[Seuret04] A. Seuret, M. Dambrine, and J.-P. Richard. Robust exponential stabilization for systems with time-varying delays. In *TDS04, 5th IFAC Workshop on Time Delay Systems*, Leuven, Belgium, Sep 2004.

[Seuret05] A. Seuret, E. Fridman, and J.-P. Richard. Sampled-data exponential stabilization of neutral systems with input and state delays. In *IEEE MED 2005, 13th Mediterranean Conference on Control and Automation*, Cyprus, 2005.

[Seuret06] A. Seuret, F. Michaut, J.-P. Richard, and T. Divoux. Networked control using gps synchronization. In *Proc. of ACC06, American Control Conf*, Mineapolis, USA, 2006.

[Simon05a] D. Simon and F. Benattar. Design of real-time periodic control systems through synchronisation and fixed priorities. In *Int. Journal of Systems Science*, volume 36, pages 57–76, 2005.

[Simon05b] Daniel Simon, David Robert, and Olivier Sename. Robust control/scheduling co-design : application to robot control. In *RTAS'05 11th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Francisco, CA, USA, Mar 2005.

[Spuri96] M. Spuri. Analysis of deadline scheduled real-time systems. Technical report 2772, INRIA Rocquencourt, Jan 1996.

[Stubbs96] R. A. Stubbs and S. Mehrotra. A branch and cut method for 0-1 mixed convex programming. In *presented at INFORMS Meeting*, Washington, 1996.

[Taha75] H. A. Taha. *Integer Programming-Theory, Applications and Computations.* Academic Press, London, 1975.

[Tin92a] K. Tindell. An extendible approach for analyzing fxed priority hard realtime tasks. Technical report ycs189, 1992.

[Tipsuwan03] Y. Tipsuwan and M. Y. Chow. Control methodologies in networked control systems. In *Control. Engineering Practice*, volume 11, pages 1099–1111, 2003.

[Tipsuwan99] Y. Tipsuwan and M.-Y. Chow. Fuzzy logic microcontroller implementation for dc motor speed control. In *The 25th Annual Conference of the IEEE, Industrial Electronics Society, IECON '99*, volume 3, pages 1271–1276, San Jose, CA, 1999.

[Traore07] K. Traore. *Analyse et validation des applications temps reel en presence de transactions : application au pilotage d'un drone miniature.* Ph.d. thesis, Universite de Poitiers, 2007.

[Walsh02] G. C. Walsh, H. Ye, and L. Bushnell. Stability analysis of networked control systems. In *IEEE Transactions on Control Systems Technology*, number 10(3) :438-46, 2002.

[Walsh99] G. Walsh, H. Ye, and L. Bushnell. Stability analysis of networked control systems. In *Proc. Amer. Control Conf*, pages 2876–2880, San Diego, CA, Jun 1999.

[Walsh99a] G. Walsh, O. Beldiman, and L. Bushnell. Asymptotic behavior of networked control systems. In *Proceedings of the 1999 IEEE international conference on control applications*, volume 2, pages 1448–1453, Kohala Coast, HI, 1999.

[Walsh99b] G. Walsh, O. Beldiman, and L. Bushnell. Error encoding algorithms for networked control systems. In *Proceedings of the 38th IEEE conference on decision and control*, volume 5, pages 4933–4938, Phoenix, AZ, 1999.

[West99] R. West and K. Schawn. Dynamic window-constrained scheduling for multimedia applications. In *6th IEEE International Conf. On Multimedia Computing and Systems*, 1999.

[Westerlund95] T. Westerlund and F. Pettersson. A cutting plane method for solving convex minlp problems. *Computers and Chemical Engineering*, (19) :S131–S136, 1995.

[Wilwert03] C. Wilwert, Y. Q. Song, F. Simonot-Lion, and T. Clément. Evaluating quality of service and behavioral reliability of steer-by-wire systems. In Calouste Gulbenkian Foundation, editor, *9th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 16–19, Lisbon (Portugal), Sep 2003.

[Xia04] F. Xia, Z. Wang, and Y. X. Sun. Simulation based performance analysis of networked control systems with resource contraints. In *The 30th annual conference of the IEEE industrial electronics society*, volume 3, pages 2946–2951, Busan, Korea, Nov 2004.

[Yuan88] X. Yuan, S. Zhang, L. Piboleau, and S. Domenech. Une methode d'optimisation nonlineare en variables mixtes pour la conception de procedes. *RAIRO*, (22) :331, 1988.

[Zhang01] W. Zhang, M. S. Brannicky, and S. M. Philips. Stability analysis of networked control systems. In *IEEE Control System Magazine*, number 21(1) :84-89, Feb 2001.

[Zhao99] Q. C. Zhao and D. Z. Zheng. Stable and realtime scheduling of a class of hybrid dynamic systems. In *Journal of Discrete Event Dynamical Systems*, 1999.

Résumé

Le cadre de ce travail est l'étude coordonnée de lois de contrôle et d'ordonnancement. Le premier objectif est de proposer et évaluer une approche de contrôle de la dégradation de la Qualité de Contrôle (QdC) par rejet sélectif d'instances de tâches ou de messages selon le modèle (m, k)-firm. Plus particulièrement, nous avons étudié l'impact de distribution de rejets sur la QdC d'une boucle de contrôle et, sur la base des résultats obtenus, nous avons spécifié une méthode de co-conception permettant de déterminer les paramètres (gain) optimaux de la loi de contrôle et les paramètres de la contrainte (m, k)-firm spécifiant le rejet sélectif d'instances. Cette proposition a été validée sur modèles à l'aide de techniques analytiques, par simulation ainsi que grâce à des expérimentations.

Notre deuxième objectif est d'étudier le problème de l'ordonnancement d'un ensemble de tâches temps réel réalisant chacune les algorithmes de contrôle dans une application centralisée évolutive. Nous proposons un mécanisme d'ordonnancement qui ajuste en ligne les contraintes (m, k)-firm des tâches suivant la configuration courante de l'application de manière à ce qu'un critère reflétant la performance globale de l'application soit optimal à tout instant.

Mots-clés: Temps Réel, (m, k)-firm, QdC, dégradation acceptable, ordonnancement, optimisation

Abstract

In this thesis, we study a coordinated approach for the design of control laws and scheduling parameters. The first objective is to propose and evaluate a technique to control the degradation of the quality of control (QoC) by selectively rejecting task or message instances according to the (m, k)-firm model. More specifically, we have studied the impact of the (m, k)-firm packet dropout policy on the QoC of a control loop. Based on the obtained results, we have specified a co-design method for determining the parameters (gain) of the optimal control law and the (m, k)-firm constraint specifying the selective rejet of instances. This proposal was validated by using analytical techniques, simulation and experimentation activities.

The second objective is to address the scheduling problem of a set of real-time tasks where each task implements a control law in a centralized scalable application. We proposed a scheduling mechanism which determines on line, according to the current system configuration, a (m, k)-constraint based scheduling strategy to apply to each task so that the criterion reflecting the overall performance of the application is optimal at all times.

Keywords: Real-time, (m, k)-firm, QoC, Graceful degradation, Scheduling, Optimization

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE
DU DOCTORAT DE L'INSTITUT NATIONAL
POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :
Madame Sylviane GENTIL, Professeur, INPG, Saint Martin d'Hères

Monsieur Daniel SIMON, Chargé de Recherche, INRIA Rhône Alpes-Alpes

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

Monsieur JIA Ning

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 54501
VANDŒUVRE CEDEX

à soutenir devant un jury de l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :

"conception conjointe optimisée de lois de contrôle et d'ordonnancement"

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : **« Informatique »**

Fait à Vandoeuvre, le 07 janvier 2009
Le Président de l'I.N.P.L.,
F. LAURENT

TEL. 33/03.83.59.59.59
FAX. 33/03.83.59.59.55