Institut National de Polytechnique de Lorraine
Département de formation doctorale en informatique — École doctorale IAE+M

Définition et gestion d'une qualité de service pour les applications temps réel

THÈSE

soutenue le 27 novembre 2003
pour l'obtention
du

Doctorat de l'Institut National de Polytechnique de Lorraine

(Spécialité Informatique)

par

Fabrice JUMEL

Composition du jury

Président :	Guy JUANOLE	Prof. à l'université Paul Sabatier,Toulouse
Rapporteurs :	Marie-Claude PORTMANN	Prof. à l'Ecole de Mines de Nancy (INPL)
	Francis COTTET	Prof. à l'ENSMA (Poitiers)
	Martin TÖRNGREN	Prof. au Royal Institut of Technology, Stockolm, Suéde
Examinateurs :	Françoise SIMONOT-LION	Prof. à l'Ecole de Mines de Nancy (INPL)
	Nicolas NAVET	Chargé de Recherche INRIA

Laboratoire LOrrain de Recherche en Informatique et ses Applications - UMR 7503

Définition et gestion d'une qualité de service pour les applications temps réel

JUMEL Fabrice

Résumé

Ce travail de thèse se situe dans le cadre de l'aide au développement des systèmes automatisés échantillonnés. L'objet principal de ces travaux consiste en l'évaluation des différents paramètres exprimant la qualité de régulation en fonction des caractéristiques de l'architecture opérationnelle. Deux approches possibles ont été étudiées : soit une étude analytique, soit une simulation basée sur des modèles. Différents modèles d'architecture opérationnelle implémentant les applications de contrôle-commande ont été proposés ou étudiés. Ils permettent d'évaluer l'influence d'une architecture sur l'évolution du système échantillonné et donc leur qualité de fonctionnement. Ces modèles permettent directement l'étude du fonctionnement nominal, ils ont aussi été adaptés de manière à proposer une méthode d'évaluation de la sûreté de fonctionnement de ces mêmes architectures en présence de défaillances transitoires. Enfin un ensemble de techniques d'ordonnancement, basées notamment sur des techniques de réservation " temps réel ", a été présenté de manière à améliorer la qualité de fonctionnement de ces applications.

Mots clefs : Systèmes échantillonnés, Temps réel, Ordonnancement, Sûreté de fonctionnement.

Abstract

The aim of this thesis is to provide some helps for the design of real time control system. The main contribution is to link the quality of the controlled system with the different parts of the control application's implementation. Both analytical analysis and simulation have been studied, using different models for the architecture of the control application. Those models allow a direct study of the nominal mode and have been adapted in order to propose some evaluation techniques for the safety of the control system in presence of transient faults. Finally, different scheduling techniques, especially based on "real time" resource reservation algorithms, have been presented in order to increase the quality of those applications.

Keywords : Control systems, Real time systems, Scheduling, Safety analysis .

Remerciements

L'achèvement d'une thèse est un événement important, et au combien attendu. Il n'est rendu possible que grâce à l'aide de nombreuses personnes. Je profite donc de l'occasion qui m'est donnée pour les en remercier même si cela ne sera jamais à la hauteur de ce que j'ai reçu.

Je tiens , en premier lieu, à remercier les rapporteurs de cette thèse qui ont su trouver le temps, malgré leur emploi du temps bien chargé, de lire ce manuscrit et d'y apporter un grand nombre de remarques.

Merci à Martin Törngren, Francis Cottet et Marie-Claude Portmann.

Un jury, c'est aussi un président. Merci à Guy Juanole d'avoir accepter cette responsabilité. Il a été à l'origine de mes travaux de thèse et c'est lui qui les conclura, je ne pouvais rêver d'un meilleur président de jury.

Une thèse n'est pas un travail solitaire, c'est avant tout celui d'une équipe, du directeur de thèse, du co-encadrant et du doctorant.

Merci à Françoise d'avoir su soutenir, à bout de bras, la conclusion de ce travail. Ses remarques, ses doutes et ses conseils avisés ont fait de ce travail ce qu'il est. Merci pour sa présence et sa patience et surtout pour le temps qu'elle a toujours su trouver pour m'encadrer malgré ses nombreuses prérogatives.

Merci à Nicolas pour sa patience et ses conseils. Il a toujours su être présent pour partager ses connaissances et son enthousiasme, il m'a permis de franchir les nombreux obstacles techniques qui se dressent sur la route de tous les chercheurs.

Ma thèse, c'est avant tout de longues années passées au sein de l'équipe TRIO. Je devrais plutôt parler de la famille TRIO. J'ai y trouvé un accueil parfait, pour résoudre les problèmes techniques et scientifiques mais surtout pour me donner un cadre de vie et de travail exceptionnel.

Merci à toutes les personnes qui y ont participé, des femmes de ménage toujours souriantes et communicantes, aux ingénieurs systèmes, en passant bien entendu par les services administratifs de l'ENSEM et du LORIA.

L'équipe TRIO, c'est avanttous ts membres permanents ou non , toujours prêts à s'entraider.

Merci à tous : Micheline, Josette, Martine, Jean-Pierre, Marcel, Michel, Xavier, Olivier, Gérald, Mathieu, Miguel, Paolo, Paolo (II), Raffaele, Tiziana, Emmanuel, Ricardo, Anis, Cédric, Raoul, Oracio, Gerardo, Mohamed, Li Ping, et j'en oublie forcément ...

Un merci tout particulier à tous ceux qui m'ont communiqué une partie de leurs connaissances et de leurs expériences :

- Jean-Pierre Thomesse dont j'ai toujours apprécié la justesse de ses remarques,
- Yé Qiong Song pour son aide sur les problèmes de modélisation,
- Jean-François Aubry et Jean-Marc Thiriet pour leur aide dans l'appréhension des problèmes de Sûreté de Fonctionnement,
- Benoit Iung pour ses conseils sur les problèmes d'automatisme,
- Sergio Junco pour sa participation active aux choix des études de cas présentées dans ce document.

Je remercie les nombreuses personnes qui m'ont aidé dans la finalisation de mes travaux et en particulier dans

la préparation de ma soutenance. J'en profite pour remercier les membres de l'équipe CITI de l'INSA de Lyon et tout particulièrement Abdelhakim, Jean Philippe, Isabelle et Anne.

Travailler, c'est bien, mais cela ne suffit pas au bonheur.

Merci à ma famille pour son soutien sans faille,

A mes parents qui ont toujours fait de leur mieux pour me permettre de travailler dans les meilleurs conditions matérielles,

A mes beaux-parents, pour les moments agréables que nous avons pu partager

Et bien sûr à ma femme Anna qui saura apprécier au moins autant que moi la fin de cette époque de " dur labeur ".

"Je tiens pour impossible de connaître un tout,
si je n'en connais pas les parties mais je tiens
aussi pour impossible de connaître les parties,
si je ne connais pas le tout de ces parties."

Pascal

Table des matières

Résumé **2**

Remerciements **3**

1 Problématique **17**
1.1 Définitions et concepts de base . . . 17
1.1.1 Système . . . 17
1.1.2 Régulation - asservissement . . . 18
1.1.3 Commande échantillonnée . . . 19
1.2 Applications numériques de contrôle-commande . . . 20
1.2.1 Systèmes échantillonnés et propriétés associées . . . 20
1.2.1.1 Choix de la fréquence d'échantillonnage . . . 20
1.2.1.2 Modes de marche . . . 21
1.2.2 Architectures de référence . . . 22
1.3 Objectifs de la thèse . . . 23

2 Critères de Qualité de Service pour la régulation **25**
2.1 Introduction . . . 25
2.2 Formalisation des critères de qualité propres à la régulation . . . 26
2.2.1 Notions élémentaires de Qualité de Service d'une régulation . . . 26
2.2.2 Critères de qualité vis à vis du rejet des perturbations . . . 28
2.3 Autres critères de qualité de Service d'utilisation courante . . . 28
2.4 Utilisation de critères intégraux pour définir la Qualité de Service . . . 29
2.4.1 Fonction de coût . . . 29
2.4.2 Critères les plus courants . . . 29
2.4.3 Généralisation . . . 30
2.5 Conclusions . . . 31

3 Modélisation des caractéristiques temporelles de l'implémentation des algorithmes de contrôle **33**
3.1 Modélisation sous forme de fonction d'états des retards et des gigues . . . 34
3.1.1 Modélisation du système régulé . . . 34
3.1.2 Formalisation des retards sur les signaux . . . 35
3.1.3 Formalisation des gigues . . . 36
3.2 Modélisation d'autres phénomènes liés aux échantillons . . . 37
3.2.1 Dérive de l'échantillonnage . . . 37
3.2.2 Retard supérieur à la période d'échantillonnage . . . 37

3.2.3 Perte de signaux . . . 37
3.2.4 Retards dans les systèmes Multiple Input - Multiple Output (MIMO) . . . 38
3.3 Modélisation simplifiée des retards . . . 38
3.3.1 Modélisation des retards par transformée de Laplace . . . 38
3.3.2 Modélisation des retards en transformée en z . . . 39
3.3.3 Comodélisation en transformée en z et transformée de Laplace . . . 40
3.4 Modélisation des caractéristiques temporelles sous forme de perturbations . . . 40
3.4.1 Retards au niveau de la commande . . . 40
3.4.2 Gigue sur l'acquisition . . . 41
3.5 Conclusions . . . 41

4 Caractériques temporelles d'une architecture informatisée 43
4.1 Introduction . . . 43
4.2 Vue comportementale détaillée : blocs et interactions . . . 44
4.2.1 Modélisation de la phase de calcul . . . 45
4.2.2 Règles d'activations des blocs . . . 46
4.2.3 Phase de lecture et écriture . . . 46
4.2.4 Coopération entre blocs . . . 47
4.2.4.1 Mode "Push" . . . 47
4.2.4.2 Mode "Pull" . . . 49
4.2.4.3 Mode "Pull entre port d'entrée et port de sortie" . . . 49
4.2.4.4 Mode "Push entre port d'entrée et port de sortie" . . . 50
4.2.4.5 Mode "Push avec activation périodique du port de sortie" . . . 52
4.2.4.6 Mode "Rendez-vous" . . . 52
4.2.4.7 Mode "Entrées multiples" . . . 53
4.3 Présentation de quelques architectures de contrôle-commande . . . 55
4.3.1 Architectures de références . . . 55
4.3.2 Exemples d'architectures élémentaires . . . 56
4.4 Prise en compte de l'Architecture Matérielle . . . 58
4.5 Conclusions . . . 60

5 Mesure de l'influence de l'implémentation sur la qualité de service de la régulation 61
5.1 Introduction . . . 61
5.2 Evaluation analytique . . . 62
5.2.1 Sensibilité aux retards . . . 62
5.2.1.1 Evaluation de l'influence d'un retard constant . . . 62
5.2.1.2 Evaluation de l'influence d'un retard variable . . . 64
5.2.1.3 Cas particulier de la gigue sur les dates d'acquisition . . . 65
5.2.2 Conclusions sur l'évaluation analytique de l'influence des fautes temporelles sur les propriétés d'un système . . . 66
5.3 Evaluation de l'influence d'une implantation par une approche simulation . . . 67
5.3.1 Principes de modélisation . . . 68
5.3.2 Modélisation à l'aide de retards précalculés - quelques exemples . . . 69
5.3.3 Exemple de modèles explicites des accès aux ressources . . . 71
5.4 Exemple de caractérisation de l'influence par simulation . . . 72
5.4.1 Métriques considérées . . . 73
5.4.2 Influence de l'implémentation . . . 74

5.4.2.1 Caractéristiques essentielles de l'architecture support . . . 74
5.4.2.2 Influence de la variabilité des temps de traitements . . . 75
5.4.2.3 Influence des choix des politiques d'ordonnancement . . . 78
5.4.2.4 Influence des modes de communication . . . 79
5.4.2.5 Adaptation au retard, mise en évidence des besoins de prédictabilité . . . 80
5.4.3 Conclusion . . . 82

6 Modélisation de la Sûreté de Fonctionnement d'une application de contrôle-commande 83
6.1 Etude de cas : " système de contrôle du niveau de liquide dans une cuve " . . . 85
6.1.1 Architecture Fonctionnelle: fonctions et flux de données . . . 86
6.1.2 Défaillance de l'information de commande vs défaillance du système. . . . 87
6.1.2.1 Réponse du système à une défaillance isolée de l'information. . . . 87
6.1.2.2 Défaillance du système . . . 87
6.1.3 Etude de fiabilité . . . 89
6.1.3.1 Modélisation . . . 89
6.1.3.2 Fiabilité de diverses architectures . . . 91
6.2 Extension de l'étude à un cas plus général . . . 92
6.2.1 Prise en compte de plusieurs défaillances globales . . . 93
6.2.2 Mise en évidence de la propagation des défaillances de l'information dans les modèles d'architectures informatisées . . . 93
6.2.3 Autres modèles d'occurrences des défaillances de l'information . . . 94
6.2.4 Extension de la fonction d'état d'erreur . . . 95
6.2.4.1 Etude de la boucle ouverte en présence d'une erreur permanente . . . 95
6.2.4.2 Rejet de l'erreur en boucle fermée . . . 96
6.2.4.3 Modélisation de la fonction d'état d'erreur . . . 97
6.3 Extension de la fonction d'état d'erreur à des systèmes sensibles aux retards . . . 100
6.4 Conclusion . . . 102

7 Techniques d'ordonnancement et prévisibilité 103
7.1 Tâches à gabarits . . . 104
7.1.1 Définition . . . 104
7.1.2 Adaptation de l'ordonnanceur . . . 105
7.1.3 Optimalité de la politique EDF . . . 107
7.1.3.1 Découpage de la tâche à gabarit en un ensemble de tâches . . . 107
7.1.3.2 Equivalence des trajectoires d'exécution entre les deux modèles . . . 108
7.1.3.3 Test de faisabilité d'un ensemble de tâches à gabarit. . . . 109
7.2 Techniques d'ordonnancement fluide . . . 109
7.2.1 Politiques à poids et à taux . . . 109
7.2.1.1 Politique à poids . . . 109
7.2.1.2 Politique à taux constant . . . 110
7.2.1.3 Quelques résultats d'optimalité des politiques à taux constants . . . 111
7.2.1.4 Conclusion sur les politiques à taux constant . . . 112
7.2.2 Mise en place de techniques de réservation temps réel . . . 112
7.2.2.1 Etat de l'art . . . 113
7.2.2.2 Modèle de réservation explicite . . . 114
7.2.2.3 Politiques de réservation . . . 116
7.2.2.4 Évaluation de performances des politiques de réservation explicite . . . 120

7.2.2.5 Conclusion . . . 125
7.3 Conclusion . . . 126

Annexe **140**

A Les formalismes des systèmes continus **141**
A.1 Représentation des systèmes continus . . . 141
A.1.1 Utilisation de la transformée de Laplace . . . 141
A.1.2 Représentation sous forme de matrice d'état . . . 143
A.1.3 Représentation fréquentielle . . . 143
A.1.4 Notions de pôles et de zéros . . . 144
A.2 Représentation des systèmes échantillonnés . . . 145
A.2.1 Transformée de Fourrier (ou en z) . . . 145
A.2.2 Représentation d'état . . . 147
A.2.3 Evolution du système entre deux échantillons . . . 147
A.3 Modélisation des non-linéarités . . . 148
A.4 Etude de la dynamique des systèmes usuels . . . 149
A.4.1 Etude de Système 1er ordre . . . 150
A.4.2 Etude de Système du $2^{ème}$ ordre . . . 150

B Architectures de commande **153**
B.1 Système en boucle ouverte . . . 153
B.2 Système en Boucle Fermée . . . 154
B.3 Différentes techniques . . . 155
B.3.1 Action en cascade . . . 155
B.3.2 Action en boucle de retour . . . 156
B.3.3 Techniques de commande optimale . . . 157

C Gestion de tâches "anytime" dans le cadre des applications temps réel **159**
C.1 Tâche temps réel flexible . . . 159
C.2 Extension du test de faisabilité "Earliest Deadline First" . . . 161
C.3 Politiques de distribution des tâches anytime reposant sur la politique EDF . . . 162
C.3.1 Politique de distribution à priorité . . . 163
C.3.2 Distribution équitable . . . 164
C.3.3 Distribution à poids . . . 164
C.4 Exemple . . . 164
C.5 Conclusion . . . 167

Table des figures

1.1 Définition d'une boucle de régulation 18
1.2 Exemple de système physique 18
1.3 Structure d'un système régulé 19
1.4 Exemple d'automate de commutation de mode de marche dans le cas du contrôle d'un moteur 21

2.1 Définition de propriétés sur l'évolution des grandeurs caractérisant le système 26
2.2 Gabarit admissible lié à un changement de mode de fonctionnement 27
2.3 Définition du temps de réponse 27
2.4 Gabarit admissible d'évolution lié à un rejet de perturbation 28

3.1 Les différents délais qui peuvent apparaître au cours de la création de la commande (sous hypothèse de relation directe entre acquisition et commande). CNA et CAN représente une conversion numérique vers analogique et réciproquement. 34
3.2 Représentation sous forme d'état du système réglé et du système de contrôle 35
3.3 Les différents délais qui peuvent apparaître au cours de la création de la commande (hypothèse de la présence d'un buffer sur les données) 41

4.1 Modélisation d'un bloc en SDL 45
4.2 Mode "Push" - niveau système 48
4.3 Mode "Push" - niveau processus 48
4.4 Mode "Pull" - niveau système 49
4.5 Mode "Pull" - niveau processus 49
4.6 Mode "Pull entre port d'entrée et port de sortie" - niveau système 50
4.7 Mode "Pull entre port d'entrée et port de sortie" - niveau processus 50
4.8 Mode "Push entre port d'entrée et port de sortie" - niveau système 51
4.9 Mode "Push entre port d'entrée et port de sortie" - niveau processus 51
4.10 Mode "Push avec activation périodique du port de sortie" - niveau système 52
4.11 Mode "Push avec activation périodique du port de sortie" - niveau processus 52
4.12 Mode "RendezVous" - niveau système 53
4.13 Mode "RendezVous" - niveau processus 53
4.14 Mode "Entrées multiples" - niveau système 54
4.15 Mode "Entrées Multiples" - niveau processus 54
4.16 Syntaxe de la représentation graphique des architectures de contrôle-commande 55
4.17 Architecture de référence "au plus tôt" 55
4.18 Architecture de référence "à retard", la loi de calcul travaille sur la donnée du cycle précédent dont elle dispose 56
4.19 Elaboration d'une commande périodique 56

4.20 Acquisition et actionnement périodiques . . . 57
4.21 Acquisition et commande périodiques . . . 57
4.22 Exemple de système MIMO . . . 57
4.23 Implantation centralisée du modèle "au plus tôt" . . . 58
4.24 Implantation distribuée autour du réseau CAN . . . 59
4.25 Implantation distribuée (2 stations connectées sur le réseau CAN et 2 stations reliées par une liaison série) . . . 59

5.1 Modèle générique d'un bloc . . . 70
5.2 Exemple élémentaire d'une régulation . . . 70
5.3 Modélisation d'un retard variable dans la boucle de régulation . . . 70
5.4 Modélisation d'une gigue sur l'aquisition des données . . . 71
5.5 Modélisation d'activations indépendantes du calcul de la loi de commande et de l'acquisition des données . . . 71
5.6 Modélisation de l'architecture de commande . . . 72
5.7 Modélisation des gestionnaires d'accès aux ressources . . . 73
5.8 Représentation schématique de la régulation du moteur . . . 74
5.9 Evolution du temps de réponse et du dépassement en fonction de la période d'échantillonnage . 75
5.10 Evolution de la sortie du système régulé pour différents temps de traitement . . . 76
5.11 Evolution du temps de réponse et du dépassement en fonction d'un retard constant . . . 76
5.12 Qualité de fonctionnement en présence de temps de traitement variable: jeux d'essais dans un intervalle borné de taille constante pour différentes moyennes . . . 77
5.13 Qualité de fonctionnement en présence de temps de traitement variable: jeux d'essais pour une même moyenne dans des intervalles bornés de taille différente . . . 78
5.14 Dates de début et de fin d'exécution des activités pour différentes politiques d'ordonnancement 79
5.15 Influence des politiques sur le temps de réponse de la régulation. Les politiques considérées sont FPP, FIFO, à poids de paramètres (400,200,100), à poids de paramètres (200,200,100) et à taux de paramètres 0.25 par tâche . . . 80
5.16 Évolution de la sortie du système, en présence d'un retard constant d'une période, avec et sans l'utilisation d'un prédicteur de Smith . . . 81
5.17 Intérêt du prédicteur de Smith suivant les politiques d'ordonnancement utilisées; sont étudiées la politique à taux constant de paramètre 0.25 et FPP . . . 81

6.1 Exemple d'architecture opérationnelle détaillée . . . 84
6.2 Causalité des défaillances . . . 85
6.3 Système de contrôle de niveau dans une cuve . . . 86
6.4 Chaîne de l'information . . . 86
6.5 Réponse à une "défaillance de l'information de commande " . . . 88
6.6 Réponse du système à une séquence d'ordres contenant des défaillances . . . 88
6.7 Diagramme de transition où 0 est l'état initial et 11 est l'état absorbant . . . 90
6.8 Mise en évidence des résultats obtenus . . . 92
6.9 Motifs élémentaires permettant de modélisation l'apparition et la propagation des défaillances dans une architecture de contrôle-commande . . . 94
6.10 Exemple de la modélisation d'un mécanisme de tolérance aux fautes, le bloc reste dans un état non défaillant tant qu'il n'a pas reçu consécutivement 3 échantillons défaillants . . . 95
6.11 Mise en évidence de l'erreur, sous forme d'une saturation de l'actionneur . . . 96
6.12 Réponse de divers système en présence d'une erreur permanente de commande . . . 96

6.13 Correction de l'erreur par le système en boucle fermée 96
6.14 Evolution du système en présence d'une erreur de commande dans le cas général 97
6.15 Différentes possibilités d'ouverture de la boucle et définition des entrées/ sorties sur le système 98
6.16 Modélisation sous forme de chaîne de Markov d'un système du premier ordre en boucle ouverte et boucle fermée 99
6.17 Evolution d'un premier ordre en fonction de l'état atteint en boucle ouverte (saturation de la commande) et en boucle fermée (fonctionnement correct) 99
6.18 Exemple de système régulé composé d'un système réglé du deuxième ordre et d'une loi de contrôle élémentaire correspondant à un correcteur proportionnel 99
6.19 Diagramme de transition où (0,0) est le seul état initial considéré et 11 est l'état absorbant (défaillance du système physique) correspondant à l'exemple d'un système du second ordre . . 100
6.20 Mise en évidence de la perte de capacité de vidange d'un système tout ou rien en présence de retard 101
6.21 Mise en évidence de la perte de capacité de vidange d'un système tout ou rien en présence de retard 101

7.1 Bornes sur la date de fin d'une activité "temps réel" 105
7.2 Définition d'un gabarit à partir des profils de charge et d'échéance 105
7.3 Automate de fonctionnement d'un ordonnanceur classique 106
7.4 Prise en compte du gabarit dans la définition de l'état des tâches 107
7.5 Découpage d'une tâche à gabarit 108
7.6 Trajectoire d'une tâche à poids 110
7.7 Placement d'une requête sur une table de réservation existante 115
7.8 Une réservation selon la politique à taux constant. 117
7.9 Une réservation selon la politique au plus tôt. 117
7.10 Placement d'une requête selon la politique de réservation à lissage de tache 118
7.11 Définition de la fonction de disponibilit du niveau de charge 1 jusqu' un niveau de charge c. . . 118
7.12 Le fait de trier l'intervalle et de connaître la longueur totale de l'ensemble des intervalles, permet de calculer en une étape la charge apportée entre le niveau courant et le niveau précedent pour l'ensemble des intervalles. 118
7.13 Exemple de réservation à lissage de charge 119
7.14 Définition de la fonction de disponibilité du niveau de charge 0 jusqu'à un niveau c 119
7.15 Placement d'une requête selon la réservation proportionnelle à la disponibilité. 120
7.16 Deux requêtes de réservation Ik et Ik+1 générées selon notre modèle de trafic où Dk est la différence entre la date de fin au plus tard et la date de début au plus tôt. 121
7.17 Taux de rejet en charge pour les politiques LC, PD, LT, TC et PTO pour une charge soumise de 50, 70 et 90 pour cent. Le taux de rejet pour TC est toujours nettement plus élevé. 122
7.18 Ecart absolu l'uniformité pour les politiques LC, PD, LT pour une charge soumise de 50, 70 et 90 pour cent. 123
7.19 Taux de rejet en charge pour les politiques LC, PD, LT, TC et PTO pour r=0.5,r=1 et r=2 (Bmax=5000 et Dmoy=2500, 5000 et 10000). 123
7.20 Ecart absolu l'uniformit pour les politiques LC, PD, LT, TC et PTO pour r=0.5,r=1 et r=2 (Bmax=5000 et Dmoy=2500, 5000 et 10000). 124

A.1 Présentation des transformées de Laplace de quelques signaux 142
A.2 Compositions de deux sous-systèmes 143
A.3 Exemple de diagramme de Bode d'un système élémentaire 144

A.4 Un exemple de la notion de dominance liée au placement des pôles 146
A.5 Définition d'une série échelon 146
A.6 Représentation schématique de la représentation d'état dans le cas des systèmes continu et discret 147
A.7 Caractéristique d'une saturation 148
A.8 Exemple de seuil 148
A.9 Phénoméne de quantification,lié en particulier à la représentation des valeurs 149
A.10 Représentation temporelle d'un signal retardé 149
A.11 Réponse d'un système du premier ordre à un échelon unité 150
A.12 Les différentes formes de réponses indicielles possibles pour un ssytème du premier ou second ordre 151
A.13 Réponse d'un système du second ordre à un échelon unité 151

B.1 Schéma générale d'une architecture de contrôle 153
B.2 Mise en place d'une boucle d'anticipation 154
B.3 Schéma d'un controlleur en boucle de retour 156
B.4 Schéma d'un controlleur par retour d'état 156
B.5 Utilisation du retour d'état quand l'état n'est pas directement accessible 157
B.6 Principe de fonctionnement du prédicteur de Smith 157
B.7 Utilisation d'un estimateur en boucle ouverte et en boucle fermée 158

C.1 Qualité du résultat en fonction du temps CPU alloué 160
C.2 Résumé de la définition des tâches de l'exemple 165
C.3 Intervalles critiques 165
C.4 Définition des matrices 166
C.5 Politique de distribution suivant les priorités 166
C.6 Politique de distribution suivant les priorités 166

Introduction

L'utilisation des systèmes automatisés est de plus en plus généralisée. On les retrouve dans tout appareil en interaction avec son environnement. Traditionnellement présents dans les usines, que ce soit au niveau du contrôle sur des systèmes manufacturiers ou au niveau du contrôle des procédés continus, ils occupent de plus en plus une place importante dans l'ensemble des systèmes mécatroniques (mécanique et électronique) et touchent actuellement la plupart des systèmes grand public. Tout système automatisé comprend un système (équipements qui assurent des fonctions de traitement et de transport de matières, de matériaux et / ou d'informations), un système de contrôle et un ou plusieurs utilisateurs. La qualité des systèmes automatisés repose d'une part sur leurs capacités de réaction à un environnement et / ou de réalisation de la mission qui leur est confiée et, d'autre part, sur la confiance que leurs utilisateurs peuvent mettre en eux.

Le but global des travaux présentés dans ce document a été de fournir des moyens et des méthodes pour assurer, d'une part, la qualité d'une implémentation du système de contrôle et, d'autre part, évaluer la qualité d'un système automatisé complet en fonction de cette implémentation.

Ce document se décompose en deux parties. L'objectif de la première partie est de recenser les concepts, moyens et techniques permettant de formaliser les liens entre les paramètres exprimant la qualité d'une régulation et les caractéristiques de l'architecture opérationnelle. Plus précisément, dans le chapitre 1, nous présentons les différentes notions nécessaires au développement de tels systèmes et les problèmes posés par leur cycle de conception. Nous rappelons rapidement, dans un premier temps, ce que recouvrent les concepts de système, de régulation et de commande échantillonnée. Puis, nous introduisons les problèmes liés à l'implémentation de ces systèmes ainsi que les questions auxquelles ce travail apporte des éléments de réponse. Le chapitre 2 décrit ce qui pourrait s'appeler la qualité de service au sens de l'automatique et fournit les différentes métriques de performance d'utilisation courante en automatique. L'implantation numérique des systèmes de contrôle-commande introduit des perturbations qui se manifestent comme des fautes temporelles vis-à-vis d'une spécification de lois de commande. Ces fautes peuvent, abstraction faite de leur origine, être prise en compte dans des études d'automatique. Les techniques usuelles font l'objet du chapitre 3. Pour comprendre l'influence des choix d'implémentation sur la qualité de service, il faut être capable de modéliser l'architecture opérationnelle et son influence sur l'évolution du système régulé. Pour ceci, nous montrons dans le chapitre 4 comment caractériser les fautes temporelles engendrées par l'implémentation et ce, sur la base de la notion d'échantillon.

La seconde partie du document contient les contributions propres de ce travail. A l'aide des différents modèles proposés dans la première partie, nous pouvons enfin faire le lien entre la qualité de la régulation et l'architecture opérationnelle choisie, c'est-à-dire, que nous pouvons évaluer les différents paramètres exprimant la qualité d'une régulation en fonction des caractéristiques de l'architecture opérationnelle. Pour cela deux approches sont possibles, une approche analytique et, lorsque celle-ci n'est plus possible en raison de

la complexité des systèmes, une approche par simulation de modèles. Ces deux approches ont été explorées et font l'objet du chapitre 5. Nous y présentons les principaux outils mathématiques nécessaires à une étude analytique ainsi que des modèles de simulation développés sous Matlab/Simulink ; les deux techniques permettent la prise en compte de la plupart des fautes temporelles induites par les performances de l'architecture opérationnelle. Leur exploitation se révèle possible dans certains cas pour lesquels des modèles fins à la fois du système régulé et de l'architecture support sont exigés. Un ensemble de résultats concernant l'influence des choix d'architecture est également donné dans ce chapitre . Nous avons appliqué les techniques analytiques de la co-modélisation à l'étude de la sûreté de fonctionnement des systèmes régulés. Nous avons ainsi pu proposer des outils de calcul de différents critères de fiabilité (comme le temps moyen avant défaillance) en fonction des différents choix d'implémentation qui sont faits, aussi bien au niveau des algorithmes de commande que des mécanismes de tolérances aux fautes (chapitre 6). Enfin, nous étudions le problème sous l'aspect des services qui pourraient être intégrés à une architecture informatique support afin d'optimiser la qualité d'une architecture opérationnelle de contrôle-commande. En particulier, dans le chapitre 7, nous introduisons une nouvelle forme d'ordonnancement des ressources, qui repose sur la notion de partage équitable et permet d'obtenir une plus grande prédictibilité sur les dates d'occurrence des événements de l'architecture opérationnelle significatifs pour la qualité de la régulation.

De nombreuses perspectives peuvent être dégagées de ces études, nous développerons en conclusion (chapitre 7.3) celles qui nous semblent les plus intéressantes au vu des résultats déjà obtenus.

Finalement, l'annexe A rappelle les concepts de base sur la modélisation de systèmes continus tandis que l'annexe B donne quelques exemples d'architectures canoniques de contrôle-commande. En annexe C figurent les résultats d'une étude menée en parallèle à cette thèse en coopération avec le projet MAIA du LORIA ; il s'agit de la spécification d'un contrôleur de tâches “anytime” temps réel.

Chapitre 1

Problématique

Dans ce chapitre, nous présentons les différentes notions nécessaires au développement de systèmes automatisés et les problèmes posés par leur cycle de conception. Nous rappelons rapidement, dans un premier temps, ce que recouvrent les concepts de système, de régulation et de commande échantillonnée. Puis, nous introduisons les problèmes liés à l'implémentation des systèmes automatisés ou, plus exactement de leur système de contrôle en identifiant particulièrement les contraintes qui sont associées.

1.1 Définitions et concepts de base

1.1.1 Système

Le système est l'objet d'étude de l'automatique. Un système se caractérise par des grandeurs qui peuvent être d'entrée, d'état ou de sortie. Les grandeurs d'entrée agissent sur le système. Sous leur effet, le système se modifie : une partie de ses caractéristiques internes varient (grandeurs d'état) et ses valeurs externes surveillées, dont les évolutions peuvent être désirées et que l'on appelle grandeurs de sortie ou réponses du système, changent. Parmi les grandeurs d'entrée, une partie est maîtrisée ou maîtrisable. En particulier, certaines sont volontairement modifiées afin d'obtenir un nouvel état du système : on parle alors de consignes (ou plus généralement de grandeurs d'actions) appliquées au système. Les autres grandeurs d'entrée modifient le système par des effets indésirables qui ne sont pas connus de manière déterministe et sur lesquelles on ne peut pas agir : on parle alors de grandeurs de perturbation (ou de bruit).

Les grandeurs de sortie sont donc les réponses extérieures du système aux sollicitations de l'entrée. Elles peuvent faire ou non l'objet d'une mesure, d'une surveillance ou d'un contrôle : on parle alors de régulation ou d'asservissement. Les grandeurs d'état tiennent compte de façon intrinsèque de l'évolution du système ; la connaissance de l'ensemble des valeurs d'état d'un système, à un instant donné, doit permettre de connaître son évolution future ; c'est pourquoi l'état du système est également appelé mémoire du système. Il existe bien entendu un certain nombre de relations entre l'état du système et sa sortie. En règle générale, les sorties sont des fonctions des variables d'état et leur choix dépend des besoins exprimés sur le système.

Un système peut posséder une seule grandeur d'entrée et de sortie (SISO simple input, simple output) où plusieurs (MIMO multiple input, multiple output). Un système SISO peut cependant posséder un grand nombre de grandeurs d'états. Dans le cas des systèmes SISO, on définit simplement les besoins car ils s'expriment sur les variations d'une grandeur de sortie unique alors que dans les systèmes MIMO, on les définit plus difficilement car il peut exister des besoins contradictoires entre différentes sorties.

La figure (1.2) présente un système composé d'un chariot mû par une force de traction (par exemple due à un moteur). L'entrée du système est représentée par la force de traction ; sa dynamique correspond à son

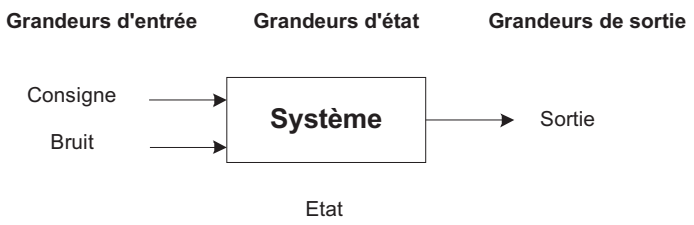


FIG. 1.1 – Définition d'une boucle de régulation

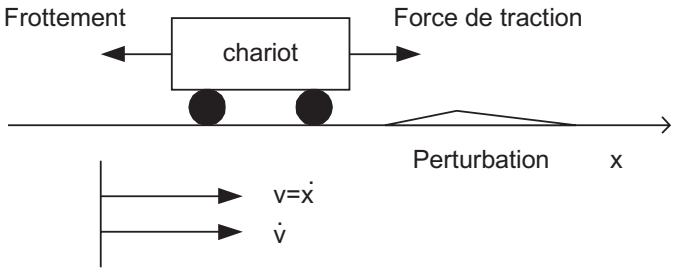


FIG. 1.2 – Exemple de système physique

déplacement. Ses variables d'états sont : sa vitesse, son accélération et éventuellement sa position. Si l'on considère que la sortie du système est la vitesse (tout dépend du but que l'on fixe au déplacement du chariot), alors le bruit peut correspondre à l'existence d'un relief qui perturbe les déplacements du chariot (une pente ou une bosse) .

1.1.2 Régulation - asservissement

La régulation consiste à maintenir un certain état du système soumis à l'évolution de l'environnement extérieur et l'asservissement à fournir au système la capacité de suivre au mieux les variations de la consigne qui lui est appliquée. Les systèmes sont généralement à la fois régulés et asservis. Le système que l'on régule est alors appelé *système réglé* (ou procédé) et sa régulation est rendue possible par l'introduction d'un autre système ayant une existence physique ou purement calculatoire, appelé *système réglant*. L'ensemble constitue le *système régulé*. Sa mise en place nécessite l'existence de moyens d'actions sur les grandeurs d'entrée du procédé (ou grandeurs d'actions) ou *actionneurs* qui permettent de convertir des données (dans notre cas, des ordres) sous une forme reconnue par le système réglé.

Pour obtenir la réponse désirée du système, notamment parce qu'il n'est pas possible de modéliser le système avec une précision suffisante, il est nécessaire de suivre les différentes sorties (ou états) en temps réel, afin d'ajuster en permanence les valeurs de commande appliquées. On utilise pour cela des *capteurs*, qui convertissent les grandeurs de sortie du système réglé (aussi appelées *grandeurs de réaction*) sous une forme utilisable par le système régulant (on parle alors de *grandeurs de mesure*).

Le but de l'automatique est de fixer les règles et les différentes méthodes pour construire le système régulant (aussi appelé *correcteur*) qui permet d'obtenir le comportement désiré du système réglé. Un panorama des techniques est fourni à l'annexe B.

Définir la qualité de service en automatique peut se faire en caractérisant l'ensemble des comportements admissibles du système réglé : réaction du système face à un bruit, évolution des différentes grandeurs dans le cadre du changement de consigne. Un système est dit en équilibre quand il n'y a plus de variation de ces grandeurs d'états : la sortie est alors constante. Un paramètre de qualité de service d'un système caractérise

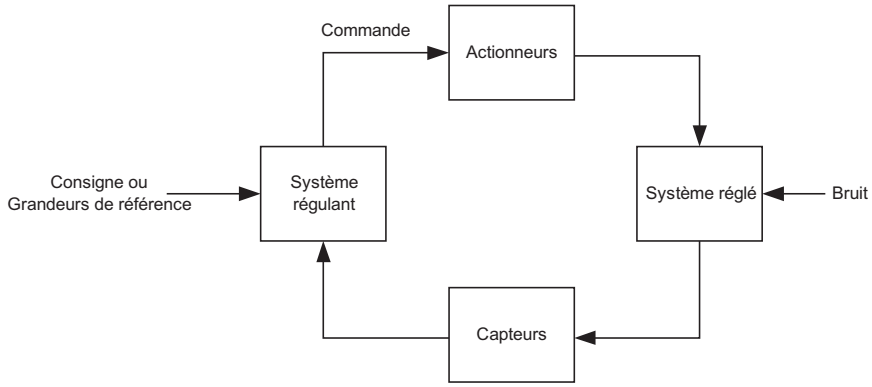


FIG. 1.3 – Structure d'un système régulé

notamment sa capacité à atteindre l'équilibre ou plus généralement à donner des indications sur la stabilité d'un système. Par exemple, dans le cas du changement de consigne, le système doit atteindre au bout d'un temps fini l'équilibre désiré (régime permanent). Deux paramètres sont importants. D'une part, il s'agit de prouver que le système est stable, c'est-à-dire de garantir qu'il atteint toujours un équilibre aprés une perturbation ou un changement de consigne et, donc, d'évaluer le temps mis pour atteindre l'équilibre. D'autre part, il s'agit de mesurer l'écart entre la valeur de sortie atteinte, à partir du point d'équilibre, et celle voulue, appelé *erreur statique* ; ceci permet de calculer la précision du système.

1.1.3 Commande échantillonnée

La réalisation des systèmes de contrôle commande, systèmes réglants, est actuellement majoritairement numérique. Ceci signifie qu'elle est implémentée sous la forme d'algorithmes (programmes) et de flux de données distribués sur une architecture de calculateurs munis de services exécutifs et de réseaux gérés par des protocoles. L'utilisation de l'informatique pour la réalisation de commande a apporté de nombreux avantages, au niveau du coût, de la flexibilité et des possibilités d'extension des fonctions du système réglant (supervision, aide à la maintenance, aide à la gestion qualité, ...).

Généralement, les systèmes à contrôler sont par nature continus, alors qu'une régulation "informatisée" est discrète. La connaissance des variables de sortie du système réglé par le système réglant ne peut alors se faire qu'à des instants donnés. Il est donc nécessaire de "discrétiser" le système continu, c'est à dire de l'*échantillonner*. L'échantillonnage d'un système consiste à procéder, en général, périodiquement à l'acquisition de l'état du système réglé et à la mise à jour correspondante de ses variables d'entrée. Notons que l'utilisation de commandes numériques, en raison des capacités et puissances de calcul disponibles et de la flexibilité apportée aux traitements, permet d'implémenter des algorithmes de commande plus complexes et, en particulier, plus fortement adaptatifs.

De manière générique, une architecture fonctionnelle de commande échantillonnée, vue indépendamment de son implémentation, comprend 3 types de fonctions, *Acquisition*, *Calcul* et *Action* où la fonction d'*Acquisition* met en oeuvre des capteurs, celle d'*Action*, des actionneurs et celle de *Calcul*, des unités de traitement d'information (processeurs, mémoire, ...). Capteurs et actionneurs assurent l'interface avec le système réglé (système physique).

Notons que les informations fournies par les capteurs nécessitent pour donner une image de l'état réel du système réglé un certain nombre de traitement : extraction des informations utiles qui sont souvent noyées dans du bruit et conversion des données dans un format utilisable par l'algorithme de contrôle.

Les systèmes échantillonnés forment le contexte spécifique de nos travaux. Plus particulièrement nous nous

intéressons à l'ensemble des activités (liées à l'acquisition d'informations sur le système réglé ou au calcul), aux propriétés qui sont attachées aux paramètres de qualité de service vues sous l'angle de l'automatique, et à l'impact de l'implantation du système réglant sous forme numérique sur ces propriétés.

1.2 Applications numériques de contrôle-commande

Les applications de contrôle-commande se décomposent en un certain nombre d'algorithmes de traitement du signal. A ces différents algorithmes sont associés des propriétés qui nécessitent une attention toute particulière dans le choix des architectures supports. Nous présentons dans cette partie, les propriétés importantes associées aux systèmes échantillonnées et les architectures de référence couramment utilisées.

1.2.1 Systèmes échantillonnés et propriétés associées

Les différentes lois et régles de commande dans le cas des systèmes échantillonnés se distinguent par leur capacité à obtenir le comportement désiré du système réglé, leur résistance à différents aléas, on parle alors de robustesse, le niveau nécessaire de finesse pour la modélisation du système, la complexité de l'implémentation ou la difficulté de paramétrer les algorithmes avant la mise en service. La réalisation du type de commande choisie passe par :

- La définition des prétraitements (filtrage, algorithmes d'analyse des données)
- L'identification et la spécification de l'algorithme de commande (qui calcule en fonction des entrées actuelles, de la consigne et de valeurs internes, la valeur à imposer sur les actionneurs qui agissent sur le système.)
- Les règles d'activation de l'algorithme de commande ou plus exactement la fréquence d'échantillonnage.

La réalisation du système réglant sous une forme échantillonnée doit intégrer le choix des actionneurs (imposé par la nature du système physique ou dépendant du type de commande choisie). Pour les travaux traités dans cette thèse, nous n'avons pas considéré le problème du choix des actionneurs (problème d'automatique pur). Par contre, l'interface que possède ces différents actionneurs et les mécanismes mis en oeuvre pour transmettre les ordres sont importants dans l'étude que nous avons conduite. Cette remarque est bien entendue valable pour les capteurs qui produisent les différents échantillons de données.

Dans la suite, nous identifions deux points clés dans la conception d'un système réglant, à savoir comment choisir les régles d'activation des traitements (la fréquence d'échantillonnage) et comment identifier les modes de marche du système.

1.2.1.1 Choix de la fréquence d'échantillonnage

La définition des règles d'activation est liée à la notion d'échantillonnage du système. Les lois de commande les plus courantes, reposent sur une propriété d'activation *périodique*. La connaissance de la fréquence correspondante influe fortement le choix des paramètres de la loi de commande. Un écart, même faible, vis à vis de la fréquence désirée peut s'avérer préjudiciable. Le choix de cette fréquence d'utilisation, est donc un des premiers problèmes que doit résoudre l'automaticien, il est complexe car il nécessite la connaissance :

- de la dynamique du système physique
- de la dynamique des capteurs
- des perturbations
- des évolutions possibles de la consigne que l'on impose au système

Le sous-échantillonnage se révèle catastrophique. Comme l'indique le théorème de Shannon [Goodwin *et al.*, 2001], il est impossible de reconstruire correctement un signal si l'échantillonnage n'est pas suffisamment rapide. Si l'on considère une décomposition du signal sous forme d'une série de fonctions sinusoïdales (de

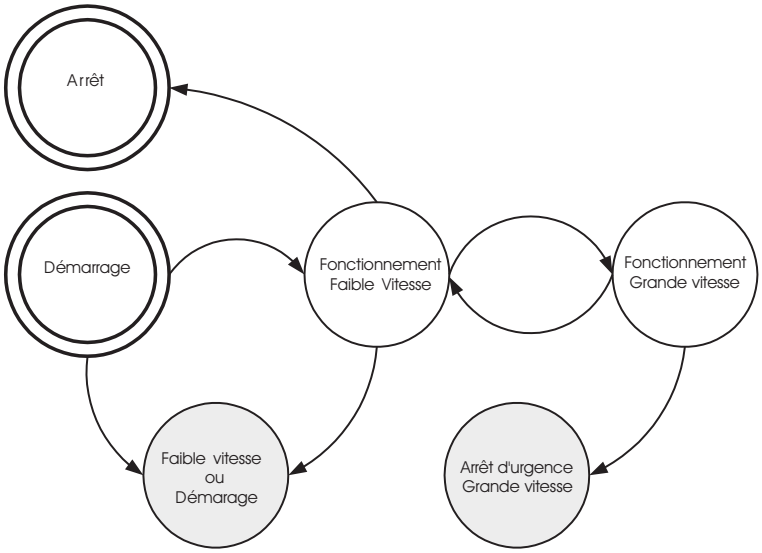


FIG. 1.4 – Exemple d'automate de commutation de mode de marche dans le cas du contrôle d'un moteur

périodes différentes), il est impossible de retrouver les caractéristiques liées à une fréquence f si la fréquence d'échantillonnage est inférieure à $2f$.

On déduit en fait de cette propriété une règle pour choisir la fréquence d'échantillonnage. Elle doit être au moins deux fois supérieure à la plus grande fréquence significative f_p. Soit une propriété attendue de la fréquence d'échantillonnage :

$$f > 2f_p$$

Dans la pratique, il est souvent difficile de connaître les fréquences propres et utiles du système, on utilise donc une règle plus simple reposant sur la réponse du système en boucle ouverte, définie par rapport à une consigne de référence, qui traduit la dynamique du système. Cette règle définie par Astrom [Ogata, 1987], s'énonce ainsi :

La période d'échantillonnage doit être 4 à 5 fois inférieure au temps de "montée" attendu du système.

L'intérêt de cette règle est qu'une fois connue la dynamique attendue du système réglé, on en déduit sa fréquence d'échantillonnage. On peut aussi déduire de cette régle, que le nombre d'échantillon nécessaire au système pour trouver un nouvel équilibre est de l'ordre de la dizaine d'échantillons.

1.2.1.2 Modes de marche

Les modes de marche s'apparentent, pour un système donné, à des phases de fonctionnement qui nécessitent la mise en place de techniques différentes pour les contrôler (cf [Gehin and Bayart, 1995]). Par exemple, dans le cas du contrôle d'un moteur, on pourrait identifier des phases de mise en route, d'arrêt, d'utilisation à faible vitesse et d'utilisation à très grande vitesse ; la spécification du fonctionnement de l'algorithme de contrôle se trouve simplifiée par l'introduction des modes de marche.

Ainsi la définition des différents modes s'accompagne de celle des enchaînements possibles entre modes. Par exemple, aller du mode de fonctionnement "à vitesse élevée" à la phase d'arrêt, passe forcément par le fonctionnement "à faible vitesse". De telles règles se modélisent globalement par un automate, dit automate de commutation des modes de marche (cf. 1.4).

1.2.2 Architectures de référence

Les applications numériques de contrôle-commande sont implémentées sur une architecture informatique support, désignées sous le terme d'*Architecture Matérielle*, et comprenant différents types d'équipements (calculateurs, capteurs, actionneurs) connectés entre eux par l'intermédiaire de réseaux de communication et gérés par des systèmes d'exploitation, des protocoles et, éventuellement, des intergiciels ("middleware"). Cette Architecture Matérielle supporte une *Architecture Logicielle*, à savoir les activités de traitement, d'échange d'informations ainsi que le contrôle de ces activités réalisant les algorithmes de contrôle-commande, leurs interactions et leur règles d'activation. La distribution, le placement et la configuration d'une Architecture Logicielle sur une Architecture Matérielle est appelée *Architecture Opérationnelle*.

Dans certains cas, pour la réalisation de l'Architecture Matérielle, du matériel spécifique (automates programmables, superviseur, ...) est utilisé comme, par exemple, pour les systèmes manufacturiers [Leigh, 1985]. Dans certains domaines, on procède directement à l'implémentation des régulations sur des supports banalisés (micro-contrôleurs, microprocesseurs banalisés), comme c'est le cas dans le contexte des systèmes embarqués (cf [Bonnet, 1999]). Les services fournis par une Architecture Matérielle sont :

- des services de calcul,
- des services de communication,
- des services de mémorisation,
- des services de gestion du temps et de contrôle des activités (tâches et messages).

La conception d'une Architecture Opérationnelle est liée d'une part, aux besoins exprimés dans le cahier des charges et, d'autre part, à des contraintes diverses (coût, réutilisation, standard d'entreprise, ...). La distribution de certains traitements peut être imposée en raison, par exemple, de la localisation de capteurs ou d'actionneurs sur le système physique. Le choix entre une architecture opérationnelle centralisée (l'ensemble des traitements est réalisé sur le même calculateur) ou distribuée (les algorithmes de contrôle sont répartis sur différents calculateurs) dépend, entre autres, de contraintes de coût et/ou de sûreté de fonctionnement). Plus généralement, la qualité d'une architecture opérationnelle dépend de :

- la puissance et la capacité du matériel utilisé (réseaux, calculateurs ...),
- les systèmes d'exploitation, les protocoles et, éventuellement, les intergiciels,
- la répartition des traitements (algorithmes) sur un ou plusieurs sites,
- la configuration des traitements, des échanges et le contrôle de leurs activations.

Deux services de l'Architecture Matérielle induisent *directement* des *caractéristiques temporelles* de l'application de contrôle-commande. Il s'agit, d'une part, des services de calcul (processeurs, co-processeurs, pipeline, mémoire cache, ...) qui associent un temps d'exécution à un algorithme de traitement et, d'autre part, des services de communication, qui associent un temps de transmission à un échange d'informations entre activités situées sur des sites distants. Dans le cas du temps de transmission et d'autant plus dans le cas d'exécution d'une activité, on ne peut généralement que déterminer une borne supérieure, le plus souvent pessimiste ; on parle alors de pire temps d'exécution ou Worst Case Execution Time (WCET - cf. par exemple [Colin and Puaut, 2000; Cottet *et al.*, 2000]).

Par ailleurs, le contrôle des activités impacte également, de façon indirecte, les performances temporelles d'une application. Cette influence est bien entendu liée au comportement global de l'architecture opérationnelle. On parlera, par exemple, de temps de réponse d'une tâche exécutant un algorithme de l'application (resp. d'un temps de réponse d'un message) pour désigner le délai qui sépare l'activation de la tâche (resp. l'émission du message) de sa fin d'exécution (resp. de la réception du message par le calculateur destinataire). Ces temps de réponse dépendent des stratégies d'ordonnancement locales et des protocoles utilisés ainsi que des mécanismes de contrôle de flux éventuellement présents dans un intergiciel.

Enfin, le matériel informatique support est lui-même caractérisé par une fiabilité exprimée à l'aide de différentes métriques ([Villemur, 1988]).

En conclusion, la réalisation d'une application numérique de contrôle-commande passe par la conception d'une architecture opérationnelle validée. Cette validation revient à vérifier que les choix faits pour l'architecture opérationnelle (matériels informatiques support, distribution, modes de contrôle des activités -périodique vs événementiel-, configuration des priorités des activités, ...) respecte les propriétés exprimées de manière plus ou moins formelles par l'automaticien (cf. section 1.2.1). Il s'agit, par exemple, de vérifier que les instants d'activation des traitements respectent la règle définie par Astrom ou le théorème de Shannon, que les données consommées sont suffisamment fraîches, ... Cette validation nécessite un modèle pertinent de l'architecture opérationnelle ainsi que des propriétés à vérifier.

1.3 Objectifs de la thèse

Le développement des applications numériques de contrôle-commande relève naturellement de la conception des systèmes temps-réels. Les modèles spécifiés par un automaticien sont généralement idéaux, c'est-à-dire qu'ils reposent sur de fortes hypothèses : les temps de traitement sont supposés nuls et la synchronisation des activités est supposée parfaite (échanges synchrones, pas de gigues, pas de retard). Or, vis-à-vis de ces hypothèses, la réalisation d'une Architecture Opérationnelle fait apparaître des fautes temporelles (gigues, retard, pertes, ...) dont certaines dégradent le fonctionnement du système. Se posent alors certaines questions comme :

- Quelle est l'influence des performances du support matériel et de la distribution et donc, des caractéristiques temporelles des activités, sur le fonctionnement du système ?
- Comment choisir les caractéristiques comportementales et temporelles attendues de l'architecture support ?
- Comment minimiser les fautes temporelles susceptibles de dégrader le fonctionnement du système et quels mécanismes utiliser pour ce faire ?

Le but du travail présenté dans ce document est de fournir des éléments de réponse à ces différentes questions et ainsi d'améliorer la conception des applications numériques de contrôle-commande.

Chapter 2

Critères de Qualité de Service pour la régulation

2.1 Introduction

L'objectif global d'une application de régulation est de permettre au système réglé de suivre la consigne désirée même en présence de perturbations. Cet objectif se décline sous forme de fonctionnalités ainsi que de propriétés attendues sur ces fonctionalités et / ou de performances qui peuvent leur être associées. Les principales fonctionnalités sont :

- sa mission première, à savoir, le suivi de la consigne et ce, avec une efficacité maximum,
- rejeter les perturbations qui peuvent influer sur le fonctionnement du système,
- assurer un fonctionnement sécuritaire du système régulé, c'est-à-dire minimiser le risque d'atteindre un état qui mette en danger son environnement.

Plus précisément, à chacune de ces fonctionnalités, on peut associer des *caractéristiques* mesurables qui permettent d'évaluer la qualité du système régulé et, par conséquence, du système réglant. L'évolution du système peut être modélisée par l'évolution de grandeurs le caractérisant. Le schéma 2.1 présente un exemple[1] d'une grandeur caractéristique C ainsi que de propriétés attendues sur cette grandeur :

- la première propriété spécifie que le système est en fonctionnement normal à un instant donné, si, à cet instant, $|C - C_{ref}| \leq \frac{\varepsilon}{2}$ où C_{ref} est la valeur de la caractéristique correspondant à la consigne courante et ε une tolérance donnée,
- la seconde propriété spécifie que le système est sécuritaire si et seulement si $C \leq C_{limite}$; cette propriété signifie, par exemple, que le système n'est plus sécuritaire (c'est-à-dire, intuitivement, plus contrôlable) dès que la valeur de la caractéristique C dépasse un certain seuil C_{limite}.

De plus, ainsi que nous le montrons dans la suite de ce chapitre, l'évolution de ces grandeurs caractéristiques peut être liée à des paramètres de performances temporelles. L'ensemble des grandeurs caractéristiques et des paramètres de performances associés définit des métriques de la qualité d'une régulation.

Nous présentons dans la suite les différentes métriques d'usage courant associées à la qualité de la régulation et en proposons d'autres qui reposent sur des critères intégraux (c'est à dire définis par l'intégration dans le temps d'une grandeur caractéristiques) et dont l'utilisation est plus simple. Cette formalisation qui est faite de manière générique est fondamentale dans cette étude car elle offre un cadre unique pour comparer des architectures opérationnelles implémentant des systèmes de régultation qui peuvent utiliser des techniques de régulation différentes.

[1] Le système correspondant à ce schéma sera présenté au chapitre 6 Une interprétation concrète des seuils C_{limite} et C_{max} en sera alors donnée.

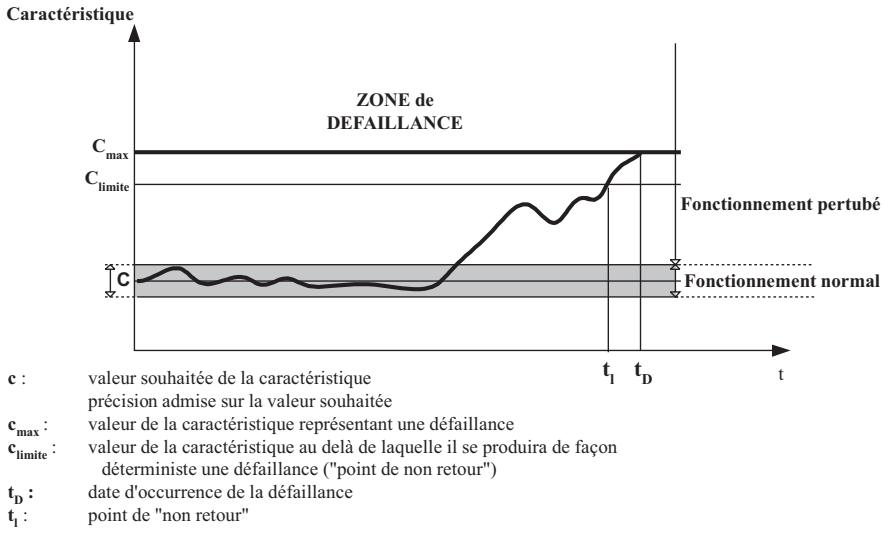


FIG. 2.1 – Définition de propriétés sur l'évolution des grandeurs caractérisant le système

2.2 Formalisation des critères de qualité propres à la régulation

Différentes définitions de la qualité de service d'une régulation sont utilisées ou peuvent être définies, certaines reposent directement sur la notion de gabarit, (ensemble des trajectoires admissibles), alors que d'autres font référence aux techniques mises en oeuvre dans certains types de régulations (comme l'utilisation de critères intégraux).

2.2.1 Notions élémentaires de Qualité de Service d'une régulation

Suivre une consigne dans les meilleures conditions est, ainsi que nous l'avons vu dans le paragraphe 2.1, une des fonctionnalités essentielles à assurer. Le terme "meilleures conditions" est lié, par exemple, à la caractéristique de *stabilité* du système; évaluer la stabilité du système revient à évaluer quantitativement sa capacité à retrouver un équilibre sous des sollicitations données. Différentes métriques de stabilité peuvent être utilisées, en fonction de la norme considérée. Par exemple, la norme la plus utilisée est $H\infty$ (pour toute entrée bornée, la sortie du système doit être bornée). On dira que la *propriété de stabilité* est assurée si la valeur obtenue par ces métriques respecte une propriété donnée.

Dans le cas de la commutation d'un mode de fonctionnement donné à un autre, par exemple, lors d'un changement de consigne, on peut étudier les propriétés du "transitoire" qui représente le comportement du système entre le moment où il quitte son premier mode de fonctionnement et celui où il se stabilise sur le second. A chaque mode est associé une propriété de stabilité qui peut être définie par un gabarit correspondant à un comportement admissible du système réglé (par exemple, à la réponse admissible du système en présence de bruit). Dans le cas où le système peut être amené à changer de zone de fonctionnement, on définit de la même manière un gabarit admissible au cours de cette évolution (voir figure 2.2) .

La notion de gabarit n'étant pas facilement exploitable. On préfére se référer à deux critères de performances, le temps de réponse et le dépassement.

- **Le *dépassement*** définit l'écart maximal constaté au cours du régime transitoire. Il s'exprime généralement sous forme d'un pourcentage $D\%$ défini par rapport à la différence entre la nouvelle consigne C_2 et l'ancienne consigne C_1. Une propriété sur cette caractéristique est : à tout instant de la zone transitoire, la valeur de la grandeur observée S est telle que $\frac{|S-C_1|}{|C_2-C_1|} \leq D\%$ ($D\%$ est d'ordinaire de 10 ou 20%).
- **Le *temps de réponse*** définit le temps nécessaire au système pour retrouver l'équilibre. On peut ca-

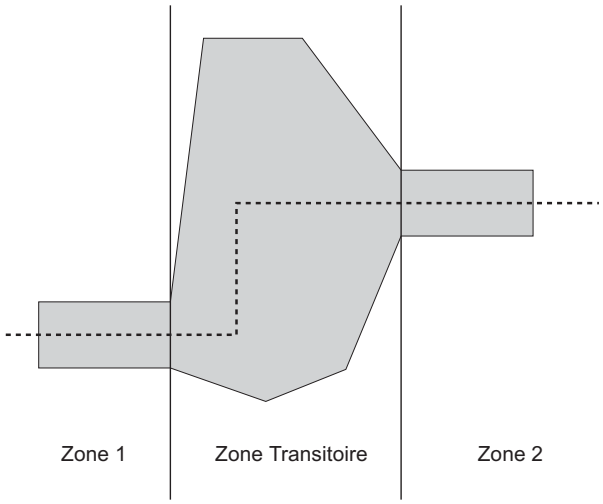


FIG. 2.2 – Gabarit admissible lié à un changement de mode de fonctionnement

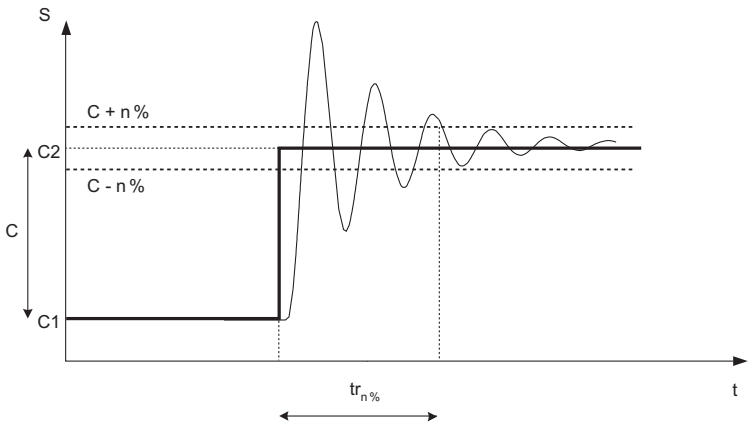


FIG. 2.3 – Définition du temps de réponse

ractériser le "temps de réponse à $n\%$", comme le temps nécessaire avant que le système ne quitte plus l'équilibre souhaité, ou plus précisément comme le temps avant que la valeur de la caractéristique C ne s'éloigne plus de plus de $n\%$ (vis à vis de $C_2 - C_1$) de la valeur C_2 correspondant à la consigne à atteindre (voir figure 2.3) . En raison de l'existence des bruits, et pour des raisons de dynamique, il est souvent impossible de se stabiliser exactement sur une valeur. On définit donc l'entrée dans un régime stationnaire par la garantie d'une borne sur l'écart entre la valeur souhaitée et celle obtenue. Généralement, on considère des temps de réponse pour $n = 5\%$ ou $n = 10\%$.

Une définition des propriétés attendues dans le cadre du changement de consigne en régulation est donc donnée par l'ensemble des quadruplets $(C_i, C_c, tr_{n\%}, D_{\%})$ où C_i et C_c correspondent à l'état d'origine (consigne initiale) et au changement de consigne et $D_{\%}$ et $tr_{n\%}$, aux bornes supérieures imposées au dépassement et au temps de réponse.

En considérant que l'état initial de la consigne n'a aucune incidence sur la qualité demandée, on peut se ramener à l'ensemble des triplets $(C_c, tr_{n\%}, D_{\%})$ équivalent au couple $(tr_{n\%}(C_c), D_{\%}(C_c))$. De plus, en considérant que les valeurs de C_i et C_c sont toujours proches, les fonctions $tr_{n\%}(C_c)$ et $D_{\%}(C_c)$ peuvent alors être considérées comme constantes. Une propriété de qualité peut alors, selon une pratique courante, être exprimée sous ces hypothèses en bornant supérieurement les éléments du couple $(tr_{n\%}, D_{\%})$.

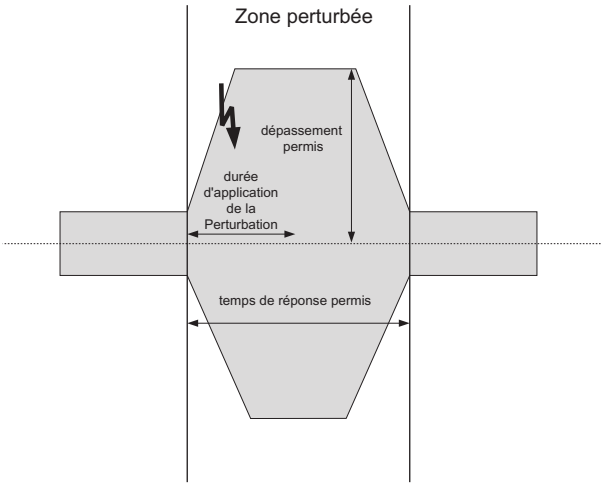


FIG. 2.4 – Gabarit admissible d'évolution lié à un rejet de perturbation

2.2.2 Critères de qualité vis à vis du rejet des perturbations

Dans un premier temps, évaluer la réaction à des perturbations passe par l'analyse faite précédemment (définition de propriétés autour de l'équilibre désiré). Cette définition est utile dans le cas où les perturbations sont permanentes et de nature aléatoire (en particulier existence d'un bruit blanc). Par contre dans le cas où les perturbations sont de nature sporadique, il est plus naturel de les considérer comme une entrée et de définir un gabarit de fonctionnement similaire au cas du changement de consigne. De la même manière, on peut donc définir un dépassement admissible et un temps de réponse avant disparition de l'influence de la perturbation.

Le dépassement dans le cas du rejet de perturbation se donne soit de manière absolue soit comme un facteur vis à vis de l'influence de la perturbation en absence de la régulation. Une propriété communément attendue sur ce dépassement est d'obtenir une réduction d'un facteur 5 ou 10 sur cette influence (cf [Goodwin *et al.*, 2001]).

2.3 Autres critères de qualité de Service d'utilisation courante

La notion de Qualité de Service fondée sur le dépassement et le temps de réponse se révèle difficile à utiliser, par exemple, dans le cas de systèmes MIMO (en particulier si le système possède plusieurs grandeurs de sortie de même importance). En outre, comme l'objectif des régulations est souvent double (suivi de changement de consigne et rejet des perturbations), cette définition ne prend en compte qu'un aspect des besoins. D'autres critères de qualité sont utilisés et permettent de quantifier de manière plus globale la qualité de la régulation. Ces notions ont été introduites simultanément aux lois de commandes, ce qui explique leur grande hétérogénéité.

Dans le cas où le système régulant est créé par étude des représentations fréquentielles du système, la *marge de phase* correspond à la capacité du système bouclé à tendre rapidement vers un équilibre. La marge de phase peut donc servir de critères de comparaison entre différentes régulations (par exemple dans les travaux de Isabelle Blum et Guy Juanole du LAAS cf [Blum and Juanole, 1999; Juanole and Blum, 1999; Blum, 2000])

De manière équivalente, les techniques courantes cherchent à ramener la dynamique du système à une dynamique qui offre le meilleur *rapport dépassement/temps de réponse*. La qualité de service dans le cadre de ce type d'approche est simple à définir puisqu'il suffit de rechercher l'écart entre le système obtenu et le système idéalement désiré (cf Annexe A). Par exemple, dans le cas où la régulation consiste en une identification à un deuxième ordre caractérisé par $\xi = 0.7$, on peut utiliser comme critère de qualité la différence entre le ξ

obtenu et le ξ désiré et fixer comme propriété attendue une borne sur les valeurs de ξ admissibles. Ce critère est intéressant car il correspond à la fois à la capacité d'asservissement et de régulation de la commande.

2.4 Utilisation de critères intégraux pour définir la Qualité de Service

2.4.1 Fonction de coût

Dans le cas où l'algorithme de commande cherche à minimiser une fonction de coût en présence de contraintes, cette *fonction de coût* apparaît comme une définition naturelle de la qualité de service de la régulation et l'on peut fixer une borne sur la qualité admissible. La fonction de coût se met généralement sous une forme linéaire (cf [Goodwin *et al.*, 2001]) :

$$J = \frac{1}{2}\sum (xQx^T + uRu^T)$$

où x correspond aux états du système et u à la commande.

Ce critère fait apparaître une qualité de la régulation qui a deux composantes, l'une représente l'évolution interne du système (terme en Q) et s'apparente à la notion de gabarit présentée précédemment et l'autre représente un facteur énérgétique (terme en R). En effet, l'action sur le système a souvent un coût énérgétique important, dans le cas d'un véhicule ou d'un four par exemple, on comprend aisément qu'il faut fournir de l'énergie pour intervenir sur la dynamique. Cette fonction de coût permet donc de tenir compte de cet aspect dans la définition de la solution optimale de l'évolution du système. Cette définition a été rendue obligatoire dans le cadre des commandes par optimisation, car de manière triviale, la meilleure commande au niveau des performances est celle qui nécessite une énergie infinie. Pour s'en convaincre, il suffit d'imaginer un véhicule qui puisse s'arrêter et atteindre sa vitesse maximale quasi instantanément. C'est en utilisant ces deux modes, que l'on peut l'amener à un endroit précis en un minimum de temps, mais bien entendu une telle propriété nécessiterait l'utilisation d'une énergie immense pour combattre l'inertie du système.

2.4.2 Critères les plus courants

L'utilisation d'une fonction de coût (comme introduite en commande optimale), peut être généralisée à l'ensemble des systèmes. Nous présentons ici, les différents critères dans le cas où le système est un système SISO (une seule entrée et une seule sortie) mais les critères peuvent être étendus au cas MIMO, comme cela est fait pour définir les fonctions de coût évoquées précédemment.

Il est très intéressant de disposer d'une définition de la qualité d'après une grandeur réelle unique ; on peut alors établir une relation d'ordre qui permet toujours de comparer la qualité de deux régulations. Le cas du temps de réponse est intéressant mais il ne prend en compte qu'une des deux caractéristiques essentielles de la qualité attendue. Il est possible d'utiliser des métriques de qualité plus fines tenant compte de ces deux critères. La qualité peut se définir sur les variations de l'écart (temps de réponse et dépassement). Nous considérerons ci-dessous un ensemble de critères de comparaison intégraux basés sur l'écart.

On peut donner un poids égal à tous les instants t :

– l'intégrale des écarts absolus (IAE, Integral of Absolute Error)

$$IAE = \int_0^T |\varepsilon(t)|dt$$

– l'intégrale des écarts au carré (ISE, Integral of square error) qui donne plus d'importance au grand écart

$$ISE = \int_0^T \varepsilon^2(t)dt$$

On peut aussi considérer que les écarts sont plus préjudiciables avec le temps : cela correspond plus à la décomposition sous forme d'un dépassement au début du changement de consigne suivi d'une décroissance exponentielle vers un équilibre. On fait donc apparaître un facteur temps dans l'intégrale qui est lié au changement de mode de la consigne.

– l'intégrale de la valeur absolue de l'écart par le temps (ITAE, Integral of absolute error time) qui donne à l'écart une importance croissante avec le temps

$$ITAE = \int_0^T t|\varepsilon(t)|dt$$

– l'intégrale du carré l'écart par le temps (ITSE, Integral of square error * time) qui donne plus d'importance aux grands écarts et de façon croissante avec le temps

$$ITSE = \int_0^T t\varepsilon^2(t)dt$$

Ces différents critères sont déjà considérés dans de nombreuses études de régulation, ce qui montre leur intérêt pratique (cf [Kocick and Sorel, 2000; Lavarenne and Sorel, 1997; Martí *et al.*, 2002b; Cervin *et al.*, ; Redell, 1996; Nilsson, 1996; Nilsson *et al.*, 1998; Yook *et al.*, 2000] pour des approches de comodélisation et dans un cadre plus général [Otanez *et al.*, 2002; Martí *et al.*, 2002a]). Pour pouvoir les utiliser, il faut encore fixer une valeur au temps T qui apparaît comme borne de l'intégrale. Comme on considère que les régulations sont proches de vérifier le gabarit souhaité, on peut considérer une remise à zéro du temps aprés chaque changement de consigne et utiliser ces critères uniquement dans le cas où le système est dans une zone transitoire entre deux changements de consigne. En particulier, il est intéressant de considérer ces critères entre l'instant t du changement et $t + n \cdot tr_{5\%}$ avec $n = 2$ ou 3, par exemple.

Il est possible de fixer des propriétés attendues sur ces critères, il s'agit là aussi de bornes qui peuvent éventuellement être fonction du niveau de perturbation. Cette propriété correspondrait à une propriété de fonctionnement moyen. Dans la pratique cela est rarement le cas, par contre, on peut chercher à obtenir le meilleur algorithme de commande qui minimise un critère donné. C'est par exemple un moyen couramment utilisé pour fixer les paramètres d'une régulation PID (cf [Nilsson *et al.*, 1999]).

2.4.3 Généralisation

Les critères intégraux proposés sont intéressants pour modéliser la qualité de service obtenue sur le système physique. Ils reposent sur la différence dans le temps entre la réponse souhaitée et la réponse obtenue et peuvent prendre en compte facilement l'existence de plusieurs variables controlées et l'évolution dans le temps des besoins. En particulier, il est possible de modéliser la notion de temps de réponse et de dépassement dans le cas d'un changement de consigne de manière simple et naturelle. On modélise donc les différents modes de fonctionnement d'une régulation qui peuvent être de l'ordre d'un asservissement (suivie d'un changement de consigne) ou d'une régulation proprement dite (rejet des perturbations).

La proposition générale est la suivante :

On choisit un critère, pour le fonctionnement constant ($f_C(\varepsilon) = |\varepsilon|$ ou $f_C(\varepsilon) = \varepsilon^2$) et le fonctionnement transitoire (défini sur une durée T_T en cas de changement de consigne, $f_T(\varepsilon, t) = t|\varepsilon|$ ou $f_T(\varepsilon, t) = t\varepsilon^2$). On peut alors définir un critère pour la qualité de service de la régulation :

On note $\{t_i\}$ les instants de changement de consignes (on suppose, pour séparer facilement les zones de fonctionnement, que $|t_{i+1} - t_i| < T_T$; si ce n'est pas le cas, on peut adapter les équations en utilisant un T_T fonction de i).

Soit T_f la durée de fonctionnement considérée du système. La qualité de la régulation peut alors se définir simplement par :

– en fonctionnement constant :

$$QdS = \int_0^{T_f} \delta(t) \cdot f_C(\varepsilon) dt$$

avec $\delta(t) = 0$ si $t \in [t_i, t_i + T]$ et 1 sinon.

– en fonctionnement transitoire :

$$QdS = \int_0^{T_f} \overline{\delta}(t) \cdot f_T(\varepsilon, t - t_i(t)) dt$$

avec $\overline{\delta}(t) = 1$ si $t \in [t_i, t_i + T]$ et 0 sinon.

et $t_i(t) = t_i$ tel que $t_i \geq t$ et $t < t_i$

La définition de la qualité du système pendant la durée de vie du contrôle peut être vue comme une combinaison linéaire de ces deux critères :

$$QdS = \int_0^{T_f} \alpha(\overline{\delta}(t) \cdot f_T(\varepsilon, t - t_i(t))) + \beta(\delta(t) \cdot f_C(\varepsilon)) dt$$

où α et β représentent les poids relatifs donnés entre les régimes transitoires et les régimes permanents.

Dans le cas MIMO (plusieurs sorties contrôlées), on peut étendre la définition précédente, en utilisant un vecteur des écarts ε qui représentent l'écart au cas idéal sur l'ensemble des sorties.

Si la distinction qui est faite dans ce critère entre le mode de marche normal et le mode transitoire n'a aucun intérêt, on se contente d'utiliser un critère qui ne tient pas compte des transitoires comme l'IAE ou l'ISE. Inversement, on peut s'intéresser uniquement au régime transitoire (ce qui est généralement fait) et dans ce cas on peut utiliser des critères tel que l'$ITAE$ ou l'$ITSE$.

2.5 Conclusions

Il existe de nombreuses métriques permettant de décrire la qualité de la régulation. Dans le phase de définition des régulations, les automaticiens fixent les comportements admissibles et cherchent à optimiser le critère qui leur semble le plus important. On définit généralement deux types de propriétés, les premières concernent le fonctionnement normal (pour une consigne constante) et les secondes, la phase transitoire liée au changement de consigne.

Une propriété sur le fonctionnement normal peut se définir comme une zone de fonctionnement autour de l'équilibre désiré. Le comportement admissible en présence d'un changement de consigne se définit sous la forme d'un dépassement admissible et du temps de réponse maximal admis.

On peut associer à ces propriétés, des propriétés de sûreté correspondant à des bornes sur certaines grandeurs qui peuvent conduire à une mise en danger du sytème ou à sa destruction. Ces propriétés sont définies par l'automaticien qui s'assure que son algorithme de contrôle peut assurer le fonctionnement requis dans le cadre d'un fonctionnement idéal du système et de l'implémentation de la loi de commande.

La validation du respect des propriétés dans le cadre d'un fonctionnement non idéal se révèle difficile et n'est géneralement pas réalisé. Les automaticiens travaillent cependant sur la robustesse de la loi de commande, c'est à dire qu'il choisisse la loi de commande pour qu'elle résiste au mieux aux différents aléas.

Parallélement à cette recherche de garantie sur le fonctionnement du système, l'automaticien a généralement comme véritable objectif la minimisation d'un critère donné. Ce critère peut être :

– une fontion de coût (cf 2.4.1) définie sous forme d'un critère intégral,

– le temps de réponse,

– le dépassement.

L'automaticien cherche généralement à obtenir de son application de commande, le meilleur fonctionnement possible vis à vis d'un ou plusieurs critères. Il est cependant obligé de fixer des contraintes sur des grandeurs caractéristiques importantes vis-à-vis des différentes phases de fonctionnement.

Notre objectif est de pouvoir comprendre comment les choix effectués au cours de l'implémentation peuvent influencer le fonctionnement du système et de donner à l'automaticien la possibilité d'effectuer les meilleurs choix. Idéalement, il faut donc être capable d'apporter une réponse aux deux questions qui se posent :

– Comment dériver des propriétés attendues sur les métriques de performance de la régulation, les contraintes à fixer sur l'architecture informatique support du contrôle-commande ?
– Comment choisir les différents paramètres de l'implémentation de manière à optimiser le critère de fonctionnement choisi ?

La partie suivante essaye de proposer un ensemble de méthodes et d'outils de modélisation pour répondre à ces questions.

Chapitre 3

Modélisation des caractéristiques temporelles de l'implémentation des algorithmes de contrôle

Les problèmes dus à l'implantation sous forme numérique d'un système réglant peuvent être intégrés par l'automaticien lors de la spécification de lois de commande. Dans ce chapitre, nous montrons les principales techniques utilisées et proposons leur analyse .

Lors de la définition de lois de commandes sont faites un certain nombre d'hypothèses :

- la première porte sur la linéarité supposée des systèmes, hypothèse sur laquelle repose la plupart des techniques de commande (voir annexe B),
- les études n'intégrent pas la capacité de saturation possible sur les actionneurs et des non linéarités dues aux phénomènes de seuil ou d'hysteresis (voir annexe A),
- les études ne prennent pas en compte certaines perturbations, comme le bruit sur les signaux récupérés au niveau des capteurs.

Néanmoins, dans le cas où la loi de commande est implémentée de manière numérique, certains problèmes doivent être maîtrisés :

- l'échantillonnage des données empêche le contrôle du système entre deux instants d'échantillonnage,
- la précision des calculs influe sur la précision des informations manipulées ; il est nécessaire de prendre en compte les erreurs sur ces informations,
- le (ou les) échantillons utilisés par une loi de commande ne sont, dans la réalité, pas contemporains de l'activation de la loi en raison de temps de traitement et transmission incontournables.

L'influence d'une implémentation apparaît donc sous forme d'*écarts temporels* et de *fautes de valeurs*. Nous faisons l'hypothèse, dans la première partie de ce travail jusqu'au chapitre 5, qu'il n'y a pas fautes de valeurs dues à un calcul (algorithme prouvé, précision des calculs suffisante). Par contre, les caractéristiques temporelles de l'implémentation peuvent éventuellement être considérées et analysées comme des fautes de valeurs (cf 3.1.2) mais, en les considèrant uniquement comme telles, on risque de masquer leurs causes et, en particulier, leurs lois d'arrivée.

Un écart temporel signifie que les commandes calculées sont appliquées au mauvais moment et / ou sur des données qui n'ont pas été produites au bon moment. Ces deux phénomènes bien connus des automaticiens s'expriment usuellement sous forme de retards et de gigues sur l'acquisition. Nous conserverons donc cette terminologie dans la suite de ce chapitre.

Le schéma (3.1) présente les différentes écarts temporelles (vis à vis du cas idéal) qui peuvent apparaître,

liées à des processus de calcul ou de transmission de signaux ou d'informations. On peut y suivre le devenir des signaux et en particulier le lien entre les signaux discrets et continus qui coexistent dans les systèmes échantillonnés. Les écarts apparaissent sous forme de *retard* (temps de traitement) ou de *gigues* (les dates d'occurences des évènements ne sont pas celles qui sont prévues, ce phénomène est particulièrement fréquent dans le cas du processus d'acquisition).

3.1 Modélisation sous forme de fonction d'états des retards et des gigues

La représentation des systèmes continus sous forme de matrice d'état (cf Annexe A) présente le maximum de flexibilité pour modéliser l'ensemble des phénomènes qui peuvent influer sur l'évolution du système régulé.

3.1.1 Modélisation du système régulé

Le système physique à réguler (système réglé) se modélise naturellement à l'aide d'une représentation d'état (cf 3.2), c'est à dire que l'on fait apparaître dans la modélisation, les grandeurs internes qui influent sur l'évolution du système. Une équation matricielle permet alors de décrire l'évolution du systéme :

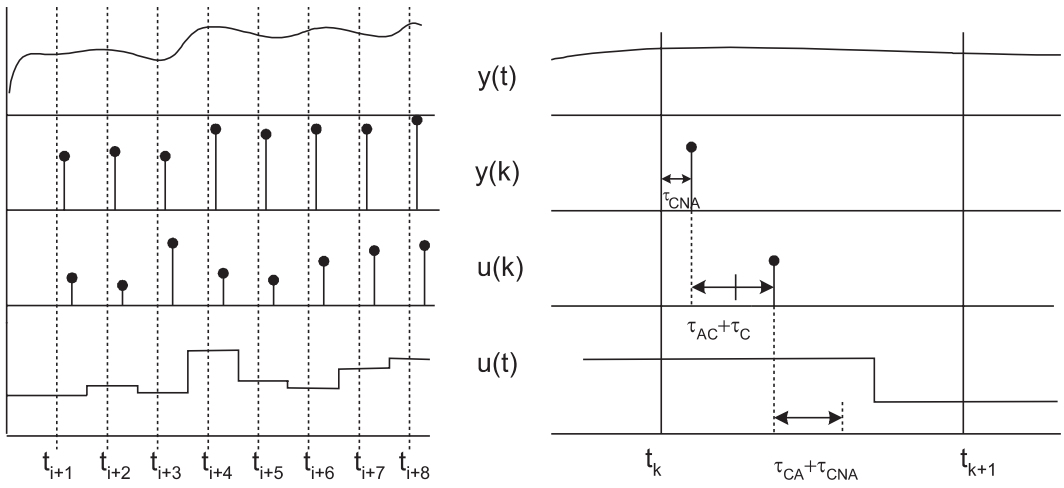


FIG. 3.1 – Les différents délais qui peuvent apparaître au cours de la création de la commande (sous hypothèse de relation directe entre acquisition et commande). CNA et CAN représente une conversion numérique vers analogique et réciproquement.

$$\left\{ \begin{array}{l} x'(t) = A(t)x(t) + B(t)u(t) \\ y(t) = C(t)x(t) + D(t)u(t) \end{array} \right\}$$

x représente l'état du système, u le vecteur de commande appliqué au système, les matrices A et B la modélisation du système réglé et les matrices C et D servent à décrire les sorties du système. (cf annexe A)

Le système de contrôle élabore une commande $u(t)$ à partir d'informations récupérées sur le système réglé (généralement la sortie $y(t)$ ou l'état $x(t)$). Plus précisément, l'algorithme de contrôle travaille sur des échantillons ; à l'étape n° k, il produit des échantillons de commande ($u_c(k)$), à partir des échantillons en provenance du système physique ($y(k)$ ou $x(k)$). Dans le cas idéal, les instants de contrôle sont périodiques de période T et l'étape n°k correspond à l'instant t_k, où $t_k = t_{k-1} + T$. La commande appliquée au système est alors constante sur $[t_{k-1}, t_k[$.

On peut modéliser cet algorithme sous forme d'une fonction (dans le cas où seul l'état x est utilisé) :

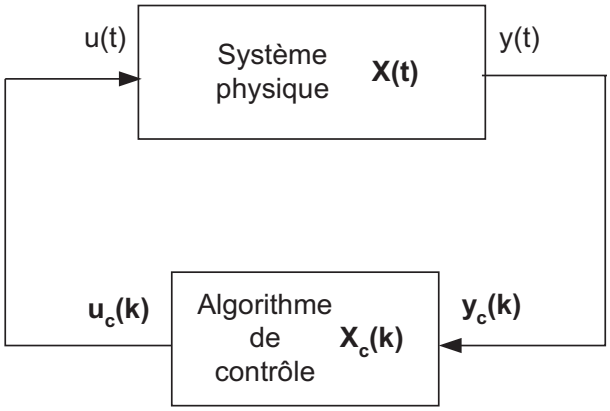


FIG. 3.2 – Représentation sous forme d'état du système réglé et du système de contrôle

$$u_c(k) = f(x(k), ..., x(k-l))$$

Cette fonction associe la sortie du système de contrôle à un certain nombre des entrées présentes et passées. Dans le cas où l'algorithme conserve en mémoire des informations sur le système (comme une intégrale de certaines grandeurs), il est plus simple de considérer une représentation d'état qui permet de modéliser l'existence d'une mémoire interne (noté $x_c(k)$ à l'étape k) au sein du système de contrôle.

On peut alors représenter la commande appliquée par l'algorithme de contrôle grâce à un système d'équation :

$$\left\{ \begin{array}{c} x_c(k+1) = f(x_c(k), x(k+1)) \\ u_c(k) = g(x_c(k), x(k)) \end{array} \right\}$$

Dans le cas où la loi de commande est définie linéairement, on obtient une représentation matricielle de la forme :

$$\left\{ \begin{array}{c} x_c(k+1) = Ax_c(k) + Bx(k+1) \\ u_c(k) = Cx_c(k) + Dx(k) \end{array} \right\} \tag{3.1}$$

En l'absence de décalages temporelles, la commande est constante au cours d'un cycle $[t_k, t_{k+1}[$, la commande appliquée en entrée du système physique $u(t)$ est donnée par :

$$u(t) \quad = \quad u_c(k) \text{ pour } t \in [t_k, t_{k+1}] \text{ avec } t_k = k \cdot T$$

3.1.2 Formalisation des retards sur les signaux

La modélisation d'un retard sur un signal continu, peut être effectuée en utilisant la définition mathématique naturelle des retards. Si on considére un système qui retarde un signal d'une durée τ, la modélisation du système se fait en liant l'entrée du système (e) à la sortie (s) :

$$s(t) = e(t - \tau)$$

La nature des signaux échangés dans les systèmes informatisés n'est pas continue mais discrète. Ainsi, la sortie du système (s_k) est la même que l'entrée (e_k) mais la date de création t_{s_k} de s_k est décalée de τ par rapport à la date de création t_{e_k} de e_k.

$$(s_k, t_{s_k}) = (e_k, t_{e_k} + \tau)$$

Ce type de modélisation discrète est nécessaire dans le cas où l'on s'intéresse au devenir des signaux à l'intérieur du système et à la conséquence de ces retards à l'intérieur du système informatisé. Cette approche sera développée dans la suite (voir Chapitre 4).

L'acquisition périodique sur $y_c(t)$ produit des échantillons $y_c(k)$ qui arrivent avec un retard τ_{AC} (de l'Acquisition vers le Calcul) au niveau du contrôleur. La loi de commande nécessite un certain temps de calcul τ_C afin de produire la commande $u_c(k)$. Il faut alors un temps τ_{CA} pour modifier la commande appliquée au système au niveau des actionneurs. Vis-à-vis du système idéal sans retard[1], le signal de commande se retrouve retardé, à l'étape k de $\tau_k = \tau_{AC} + \tau_C + \tau_{CA}$ (C'est à dire du temps nécessaire à amener les informations acquises sur le système (τ_{AC}), du temps nécessaires à l'élaboration de la commande (τ_C) et du temps à amener la commande sur les actionneurs (τ_{CA}). Cette décomposition n'a de sens que dans le cadre d'architecture élémentaire, en particulier si chaque échantillon de données amène à la création d'un échantillon de commande.Pour les autres cas, le chapitre 4 présente une modélisation permettant de prendre en compte des architectures plus complexes.

Il est donc possible de modéliser l'ensemble des retards en considérant un retard unique avant l'action sur le système. A l'étape k, en présence d'un retard, la commande est appliquée sur l'actionneur une fois reçue, c'est à dire avec un retard τ_k et ce jusqu'à ce que la nouvelle commande arrive

$$u_c(t) \quad = \quad u_c(k) \text{ pour } t \in [t_k + \tau_k, t_{k+1} + \tau_{k+1}] \tag{3.2}$$

Si les retards sont constants, la commande se retrouve décalée uniformément vis-à-vis du cas idéal (indicé *ideal*) :

$$u_c(t) = u_c^{ideal}(t - \tau)$$

L'écriture matricielle du traitement (3.1) est toujours possible. Les dates d'acquisition sont toujours t_k par contre la commande est appliquée avec un certain retard (voir 3.2).

3.1.3 Formalisation des gigues

La loi de commande repose sur la connaissance d'un certain nombre de valeurs qui caractérisent le système. Théoriquement, les échantillons doivent être acquis périodiquement $u(k) = u(t_k)$ mais il peut exister une gigue J_k par rapport à ce cas idéal, l'acquisition se fait alors à $t_k + J_k$ (la gigue J_k peut être positive ou négative).

$$u(k) = u(t_k + J_k)$$

On peut remarquer que la notion de gigue ne peut être définie que vis à vis de l'acquisition sur une grandeur continue. Cette gigue peut avoir une autre conséquence sur les signaux qui transitent dans l'architecture informatisée. En effet les dates de création des échantillons ne sont plus celles qui étaient prévues. La modélisation discrète de la gigue est donc de la forme :

$$(u_k, t_{s_k}) = (u(t_k + J_k), t_k + J_k)$$

Si J_k est constante, il est intéressant de remarquer que l'acquistion est périodique et que ce décalage sera

[1] Les schémas 3.1 et 3.3 présentent la décomposition du retard. Les temps de conversion des signaux Analogique -> Numérique et Numérique -> Analogique ont aussi étés représentés. Dans le cas d'une architecture retardée, le premier temps de conversion n'intervient pas car il est pris en compte dans l'attente du cycle suivant.

a priori sans conséquence. La modélisation d'une gigue n'a donc de sens que si J_k varie dans le temps et que la moyenne de la distribution est nulle. Dans le cas contraire, il faut reconsidérer la date du premier échantillonnage, et l'on peut se ramener à une gigue dont la moyenne est nulle.

La modélisation matricielle du traitement permet de prendre en compte la gigue, la date d'acquisition, à l'étape k, étant $t_k + J_k$. On a donc $y_c(k) = y_c(t_k + J_k)$ et en présence d'un retard τ_k,

$$u_c(t) = u_c(k) \text{ pour } t \in [t_k + J_k + \tau_k, t_{k+1} + J_{k+1} + \tau_{k+1}]$$

3.2 Modélisation d'autres phénomènes liés aux échantillons

3.2.1 Dérive de l'échantillonnage

Il peut arriver que la fréquence d'échantillonnage ne corresponde pas à ce qui est prévu. La modélisation du phénomène dans le domaine temporel ferait apparaître une gigue qui tend vers l'infini, ce qui n'a aucun sens. Il est donc préférable de représenter une dérive dans le domaine fréquentiel, c'est à dire de redéfinir les échantillons, sans chercher à les comparer temporellement au cas attendu.

On a théoriquement

$$x(k) = x(t_k)$$

avec $t_k = t_{k-1} + T$

Une dérive de l'échantillonnage se traduit par une création des échantillons à des instants différents :

$$x(k) = x(t'_k)$$

avec $t'_k = t'_{k-1} + T'_k$.

On peut faire apparaître directement cette variation dans la modélisation sous forme d'état du système. Il suffit de remplacer T par $T_k^{'}$ dans la définition des instants d'échantillonnage.

3.2.2 Retard supérieur à la période d'échantillonnage

Dans la plupart des études ([Lian *et al.*, 1999; Cervin *et al.*, 2003a; Martí *et al.*, 2002b, ...]), les retards de l'implémentation sont considérés inférieurs à la période d'échantillonnage ($\tau < T$). Il peut cependant être nécessaire de disposer de la modélisation d'un retard supérieur à une période notamment dans le cas d'architectures informatiques supports complexes et distribuées. Cette modélisation ne pose aucun problème avec la représentation matricielle, si le retard est constant ou qu'il n'existe aucun chevauchement des échantillons. Dans le cas contraire, il faut prévoir des mécanismes qui gèrent l'existence de plusieurs échantillons présents pour un seul attendu et qui permettent de choisir un échantillon suivant une service de datation par exemple (le plus récent ou le plus ancien), ou suivant leur ordre d'arrivée (le premier ou le dernier arrivé par exemple).

3.2.3 Perte de signaux

La prise en compte de la perte de signaux pose le même problème que ci-dessus. Il faut connaître la valeur qui sera appliquée sur les consignes ; en général, la valeur précédente est maintenue. Il n'est cependant pas impossible d'imaginer une valeur par défaut qui serait utilisée s'il n'y a pas de nouvelle consigne arrivée avant une certaine date. On peut modéliser le phénomène mais cela nécessite un modèle d'occurences des pertes. Le chapitre 6 qui traite des problèmes de fiabilité offre un certain nombre de réponses aux problèmes de la perte

ou de la dégradation des messages de commande. Il est aussi possible de consulter les travaux de Shin qui font réference en la matière (cf. [Shin and Kim,])

3.2.4 Retards dans les systèmes Multiple Input - Multiple Output (MIMO)

Nous avons présenté la modélisation des retards et des gigues dans un système SISO (une entrée et une sortie). Dans les cas plus complexes, le système peut posséder un grand nombre d'entrées et de sorties (système MIMO) sur lesquelles les mêmes phénomènes peuvent apparaître. S'il y a plusieurs sorties, on considère un vecteur de sorties (dont les composantes sont indicés par j), les fréquences d'utilisation et les retards peuvent être différents sur les coordonnées. La modélisation au niveau des retards est la même que précédemment mais les retards peuvent différer sur chacun des actionneurs :

$$u_{C,j}(t) = u_{C,j}^{ideal}(t - \tau_j)$$

De même, on peut représenter l'existence d'une gigue différente sur les acquisitions utilisées pour élaborer la loi de commande.

$$x_j(k) = x_j(t_k + J_{k,j})$$

Le plus complexe est alors de modéliser la loi de commande, la forme matricielle nécessite une loi d'activation unique. Si l'implémentation réelle de cette loi n'est pas activée une fois par cycle, il faut décomposer l'algorithme de commande en différentes fonctions qui échangent des données entre elles et avec le système physique. Il faut alors expliciter les règles d'activation, de communication et d'échanges de ces différentes fonctions ce qui nécessite un langage de description détaillée de l'implémentation. Les principes d'une telle modélisation sont présentés dans le chapitre suivant (cf 4).

3.3 Modélisation simplifiée des retards

La forme matricielle du système physique et du système de contrôle permet de modéliser les phénomènes qui peuvent influer sur le bon fonctionnement du système régulé. Cette écriture peut être utilisée pour formaliser mathématiquement tous les caractéristiques temporelles de l'implémentation qui peuvent influer sur le fonctionnement du sytème. Par contre, son étude analytique, même si elle reste possible, n'est pas simple d'utilisation. Nous présentons au chapitre 6 un exemple d'une étude analytique dans le cas d'une évaluation de métriques de sûreté de fonctionnement.

Les représentations des systèmes en z et en p permettent également de modéliser les systèmes échantillonnés et sont d'utilisation plus aisée (voir Annexe A). Par contre, elles ne permettent pas de prendre en compte toutes les formes d'écart temporel. Nous présentons dans la suite, les cas où cela reste possible.

3.3.1 Modélisation des retards par transformée de Laplace

La représentation de Laplace (cf Annexe A) permet de modéliser sous forme de fonction de transfert les systèmes régulés. Dans le cas d'un retard constant, la transformée de Laplace du retard est :

$$e^{-\tau p}$$

La représentation en p concerne les systèmes continus ; dans le cas des systèmes échantillonnés, il faut faire apparaître le phénomène de quantification à l'aide d'un bloqueur dont une forme possible :

$$\frac{1 - e^{-Tp}}{p}$$

L'action du bloqueur, primordiale dans la représentation d'un système échantillonnée, se formalise de manière simplifiée sous forme d'un retard :

$$e^{-\frac{T}{2}p}$$

Les retards non constants ne sont pas directement pris en compte sous forme de transformée en p car le calcul direct (au niveau du produit de convolution servant à la définition de la transformée) n'est alors plus possible.

Le sytème de contrôle, étant par nature discret, il n'est pas possible de le modéliser en p. Dans le cas où l'algorithme de contrôle discret a été obtenu à partir d'un contrôleur continu, il est cependant enviseageable de conserver le modèle du contôleur continu comme une approximation du contrôleur discret. On comprend bien qu'il est cependant impossible d'obtenir des conclusions sur le fonctionnement du système réel. Cette approche a cependant le mérite de permettre d'obtenir facilement des indications sur l'influence possible de l'introduction des retards. Une telle approche a été utilisée dans de nombreuses études qui cherchaient avant tout à prouver que le retard engendré par l'implémentation peut être dangereux, et pouvait donc se contenter d'une telle approximation (cf par exemple [Kocick and Sorel, 2000; Blum and Juanole, 1999; Cervin *et al.*,]).

3.3.2 Modélisation des retards en transformée en z

La transformée en z est la représentation naturelle des systèmes échantillonnés, elle permet de modéliser à la fois la loi de commande et le système physique sous l'hypothèse que les instants d'échantillonnage et de commande sont synchronisés et périodiques. La représentation du système physique est discrétisée, ce qui simplifie la modélisation en nécessitant plus l'utilisation d'un bloqueur (inclus implicitement dans la transformée en z). La notation en z permet de lier des échantillons, ainsi l'opérateur z^{-1}correspond à un décalage dans les échantillons utilisés. Les échantillons sont obtenus par une acquisition périodique, l'opérateur z^{-1} peut être vu comme un retard d'une période T.

La modélisation d'un retard T est donc z^{-1}

La représentation d'un retard inférieur à T n'est pas possible, car par définition de la transformée en z, le temps n'a pas de sens entre deux échantillons. On peut par contre modéliser des retards supérieurs à T, à condition que ce retard soit un multiple de la période, on obtient alors simplement le retard global en composant en série les différents retards. Ainsi un retard de k périodes se modélise par :

$$z^{-k}$$

Pour représenter des retards inférieurs à la période, la technique la plus utilisée consiste à considérer une modélisation en z avec une période T' telle que T la vrai période d'échantillonnage soit un multiple de T'. On peut, alors, formaliser des retards qui sont des multiples de T'et pas seulement de T. La thèse de Nilsson de Lund présente ces différentes techniques ([Nilsson *et al.*, 1998]).

Il est aussi possible de considérer des versions adaptées des transformées en z, mais leur utilisation n'est pas toujours simple à mettre en oeuvre (en particulier, parce que les principaux outils de validation ne les prennent pas en compte). On peut cependant citer la transformée en z modifiée ([Borne *et al.*, 2000]) ou encore la transformée en z non entière ([Bailey and Swarztrauber, 1990]). Cependant, la prise en compte des

retards non constants reste délicate et, même si différentes propositions ont été faites, elles restent difficilement exploitables de manière analytique.

En conclusion, la modélisation sous forme de transformée en z du système régulé est intéressante mais n'autorise pas de considérer une grande variété de modèle d'implémentation. L'intéret principal de la méthode est qu'elle permet de conclure rapidement dans le cas d'un retard constant équivalent à une période. Il est communément admis que, si il n'y a pas de dégradation importante de la qualité dans ce cas, il y a de grande chance que pour tout retard inférieur à une période, il n'y ait pas (cf [Ogata, 1987]). Cette remarque est généralement vraie, il est cependant possible de créer des cas où elle ne fonctionne pas, il n'est donc pas possible de se limiter à une telle approche dans le cas général.

3.3.3 Comodélisation en transformée en z et transformée de Laplace

La modélisation continue ne permet pas de définir correctement l'algorithme de contrôle et la modélisation discrète n'est pas assez fine pour rendre compte des évolutions possibles du système physique en présence de retards inférieurs à la période d'échantillonnage. Par contre, rien n'empêche de considérer le système physique sous forme continue (transformée de Laplace notée p) et le système de contrôle sous forme discrète (Transformée en z notée z) (cf.). Autant, l'exploitation analytique devient complexe (comme dans le cas matricielle), autant l'approche par simulation peut donner des résultats exploitables.

Le logiciel Matlab/Simulink permet cette *comodélisation*. Le modèle obtenu n'est cependant pas aussi riche que le modèle matriciel car il est impossible de remettre en cause la périodicité des traitements effectués modélisés en z. Par contre, il est possible d'introduire des retards au niveau des signaux continus. Le système de contrôle interagit uniquement avec le système physique par l'intermédiaire de deux signaux continus ($y(t)$ pour l'acquisition et $u(t)$ pour la commande). Il est donc possible d'injecter des retards sur ces deux signaux. (voir schéma 3.2).

Le retard au niveau de l'acquisition permet de créer une gigue sur les dates d'acquisition. L'algorithme de contrôle produit périodiquement la commande mais celle-ci est calculée à partir de données produites avec un certain retard. Le retard au niveau de la commande permet de retarder l'application de la commande, ce qui correspond plus à la définition de retard introduite initialement. On peut donc utiliser cette représentation pour simuler toutes formes de gigues et de retards. Cette modélisation est d'autant plus intéressante que Matlab/Simulink permet de générer des retards continus variables. Il devient alors possible de simuler une gigue variable sur l'acquistion et un retard variable sur la commande. La seule restriction à l'utilisation des retards variables réside dans le problème des chevauchements (cf 3.2.2), Matlab/Simulink ne permet pas directement de régler ce type de problème, il faut donc éviter qu'il puisse se produire pour garantir la cohérence du modèle. Le plus simple pour cela est de limiter les retards à une période au maximum, le chevauchement n'est alors plus possible.

3.4 Modélisation des caractéristiques temporelles sous forme de perturbations

Dans cette approche, il faut quantifier la différence entre les signaux qui devraient être présents et les signaux qui existent. La perturbation peut exister au niveau des actionneurs ou au niveau de l'acquisition (cf. [Åström and Wittenmark, 1997]).

3.4.1 Retards au niveau de la commande

Dans le cas de l'action $u(t)$, on doit faire la différence entre $u_C(t) - u(t)$.

sur $[t_k, t_{k+1}[$ l'écart est donné simplement par

$$P(t) = 1_{[t_k, t_k+\tau_k[} \cdot (u_k - u_{k-1})$$

La perturbation est donnée par la différence $(u_k - u_{k-1})$ qui est difficile à évoluer a priori en raison de la boucle de retour. Le pire cas est facile à définir puisqu'il est lié aux capacités minimum et maximum d'utilisation d'un actionneur. Connaissant la dynamique de la loi de commande et du système contrôlé, on peut avoir une idée plus précise de l'évolution dans le temps de la perturbation. En particulier, il est possible de connaître la perturbation à très court terme, si on connait u_k, u_{k-1}et τ_k on peut connaître précisément à l'instant t_k la perturbation entre $t_k$et t_{k+1}.

La modélisation des retards de commande sous forme de perturbation est intéressante, si on possède des techniques de contrôle permettant d'en amoindrir les effets. Cependant, il est difficile de les quantifier à long terme en raison de leur caractère aléatoire. Par contre, à très court terme, il est possible de connaître précisément la perturbation qui va être engendrée par ce retard.

3.4.2 Gigue sur l'acquisition

De la même manière, on peut assimiler la gigue sur l'acquisition à une perturbation qui est plus difficilement calculable car il n'y a aucune raison de connaître la bonne valeur de l'acquisition (contrairement à la commande). On peut cependant avoir une idée de l'erreur qui est faite, en connaissant le niveau de bruit et la dynamique du système et ainsi obtenir un intervalle sur l'écart de l'acquisition. Cette fourchette dépend de la gigue ; plus la gigue est importante et plus la fourchette obtenue sera grande. On ne peut pas modéliser cette perturbation avec exactitude mais l'on peut éventuellement modéliser le cas le plus probable ou le pire cas au sens des conséquences qu'il peut avoir sur le système.

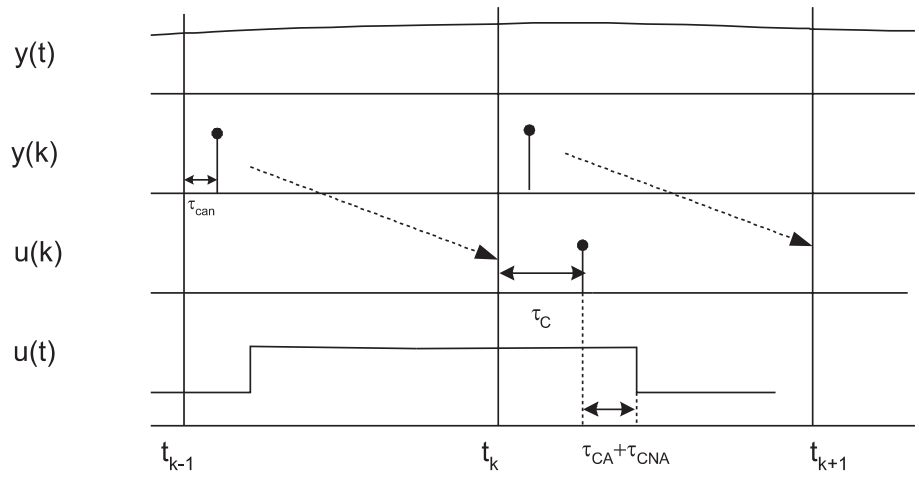


FIG. 3.3 – Les différents délais qui peuvent apparaître au cours de la création de la commande (hypothèse de la présence d'un buffer sur les données)

3.5 Conclusions

Dans ce chapitre, nous avons présenté les différentes techniques de modélisation qui permettent la prise en compte des écarts temporels, liés à l'implantation des systèmes de contrôle-commande, dans l'étude d'un système régulé. La formulation générale matricielle fait apparaître un système hybride (défini en combinant un comportement continu du système réglé et un comportement discret de la loi de commande). Ce modèle

est le plus général et permet de prendre en compte la totalité des aléas de fonctionnement du système régulé. Il est cependant difficilement utilisable au niveau analytique. Nous avons donc, ensuite, étudiés des modèles simplifiés, qui, par contre, ne permettent pas de traduire toutes les formes d'implantation.

Notons néanmoins que les services fournis par le support exécutif doivent assurer que les hypothèses sur les caractéristiques temporelles établies lors de l'analyse des lois de commande sont bien vérifiées en ligne. Par exemple, il s'agit de d'offrir des mécanismes de synchronisation et de "timers" qui garantissent des dates d'occurrences d'événements correspondant à une hypothèse de retard constant ou des services de gestion d'horloge globale et de signalisation qui permettent à l'algorithme implantant une loi de commande de connaitre le retard de l'échantillon courant, dans le cas de la prise en compte de retards variables.

Sur la base des modèles et techniques présentées dans ce chapitre (formulation générales et / ou modèles simplifiés), nous pouvons développer des méthodes d'évaluation de l'influence proprement dite des choix d'implantation. C'est ainsi que, dans le chapitre 5, nous donnons les principaux résultats analytiques disponibles sur la caractérisation de cette influence et proposons un simulateur hybride permettant de tester simplement le fonctionnement d'un système régulé utilisable dans les cas où une étude analytique se révèle impossible à conduire.

Enfin, une étude analytique précise de cette influence dans le cadre de la sûreté de fonctionnement est faite dans le chapitre 6. Elle permet d'exprimer la fiabilité du système régulé en fonction des caractéristiques d'une architecture opérationnelle supportant l'application de régulation.

Chapitre 4

Caractériques temporelles d'une architecture informatisée

4.1 Introduction

Nous venons de présenter comment la présence des choix d'implémentation peut être modélisé pour prendre en compte son influence sur la dynamique du système régulé. Nous avons montré en particulier comment modéliser cette influence par l'introduction de retards et de gigues. Nous avons également constaté qu'il était nécessaire d'avoir une évaluation précise des caractéristiques temporelles de l'implémentation des applications de contrôle-commande. Il s'agit, dans cette section, de définir le lien entre l'architecture opérationnelle implantant le système et les caractéristiques temporelles des retards et des gigues présentés précédemment.

Nous exploitons par la suite essentiellement la notion d'*échantillon* et considérons que les différents acteurs du système, nommés blocs dans la suite, communiquent par des flux de données composés d'une suite d'échantillons. Ceux-ci correspondent à l'émission récurrente de la valeur courante d'une donnée. Un flux est donc une suite d'échantillons caractérisant une certaine grandeur.

Nous considérons que pour les systèmes étudiés, les échantillons sont produits périodiquement ou sporadiquement. Ils sont indicés suivant l'ordre de leur production. Certaines activités produisent des échantillons (source), d'autres les utilisent (puits) et les dernières consomment des échantillons pour en produire d'autres.

Un échantillon s_n peut être défini par un triplet $T_{S,n} = (S, n, s(n))$ où S est le producteur du signal, n l'indice et $s(n)$ la valeur portée par l'échantillon d'indice n. A tout échantillon, est associé :

- une date de production $Prod(s_n)$ par le producteur unique S,
- des dates de consommation $Cons^B(s_n)$ où B est un bloc (plusieurs blocs peuvent être consommateurs du même flux de données)

Dans un système de contrôle-commande, il existe deux classes de flux de données :

- ceux transportant les données de commande issues d'activités implantant les algorithmes de calcul de consignes à appliquer aux actionneurs,
- ceux transportant des données de mesures (en provenance de capteurs) et qui seront utilisés pour les activités citées ci-dessus.

Les données de commandes appliquées aux actionneurs influent sur l'évolution du système réglé. Ces échantillons sont donc les plus importants vis à vis du système réglé (car ils influent sur sa dynamique) et le but du système de commande est de les produire.

Dans le travail présenté dans ce document, l'objectif de la modélisation de l'architecture informatique est

l'évaluation des dates d'occurences des événements significatifs du système régulé. Ces événements significatifs -ou *critiques*- correspondent aux actions qui ont une influence sur l'évolution du système physique. Il s'agit, donc particulièrement des événements de production des échantillons correspondant aux données de mesures et de ceux correspondant aux consommations de données de commandes.

Pour suivre les occurrences de ces événements *critiques*, il faut être capable d'une part, d'évaluer leurs dates d'occurences $\delta_{S,n}$ et, d'autre part, de déterminer si les valeurs associées aux échantillons correspondants sont correctes et, le cas échéant, de pouvoir quantifier l'erreur sur ces valeurs.

Dans le cas de systèmes complexes, l'évaluation des dates d'occurrence des événements critiques nécessite de travailler sur un modèle qui fournit une abstraction pertinente de l'ensemble de l'architecture d'implémentation, c'est-à-dire du modèle de l'*Architecture Opérationnelle* (modèle de la distribution et de la configuration d'une *Architecture Logicielle*, représentant les algorithmes de traitement -lois de commande, filtrage, ...- et leurs échanges sur une *Architecture Matérielle* composée de calculateurs munis de systèmes d'exploitation et connectés sur des réseaux de communication). Les principes de cette modélisation reposent, d'une part, sur une représentation du *comportement* (ensemble des règles d'activation et d'utilisation des échantillons) et, d'autre part, sur les c*aractéristiques temporelles* associées (temps de traitement, aléas temporels vis à vis des règles d'activation ...).

4.2 Vue comportementale détaillée : blocs et interactions

La partie comportementale de l'Architecture Logicielle contient la définition des différents *blocs* de traitements et les comportements associés :

- les algorithmes associés aux différents traitements,
- la spécification des données consommées et produites par ces algorithmes,
- les règles d'activation et d'utilisation des blocs.

ainsi que les interactions entre blocs.

Dans le cas des architectures de contrôle-commande, différents types de blocs de traitement peuvent être définis qui correspondent :

- aux traitements d'acquisition (implantés sur et / ou par des capteurs),
- aux traitements d'application des commandes (implantés sur et / ou par des actionneurs),
- aux algorithmes de calcul des commandes,
- aux algorithmes de traitement du signal (filtrage, ...).

Ces blocs interagissent entre eux en s'échangeant (ou partageant) des informations (ou échantillons). Ils sont éventuellement synchronisés entre eux (en fonction de leurs règles d'activation spécifiques). Le schéma 4.17 présente un exemple de système défini sous forme de blocs, qui sera détaillé dans la suite du chapitre. Les relations d'activation et les échanges de messages apparaissent explicitement sur le schéma.

Pour fournir une abstraction du comportement d'un bloc, nous identifions, pour chaque bloc :

- une phase de lecture d'échantillon (ou d'acquisition sur le système physique pour un traitement d'acquisistion),
- une phase de calcul (élaboration de la valeur d'un échantillon),
- une phase d'écriture d'échantillon (ou d'action sur le système physique pour un traitement d'application de commande),
- des règles d'activation de ces différentes phases.

Ce fonctionnement est modélisé simplement en SDL[1] à l'aide du schéma suivant (cf 4.2.1)

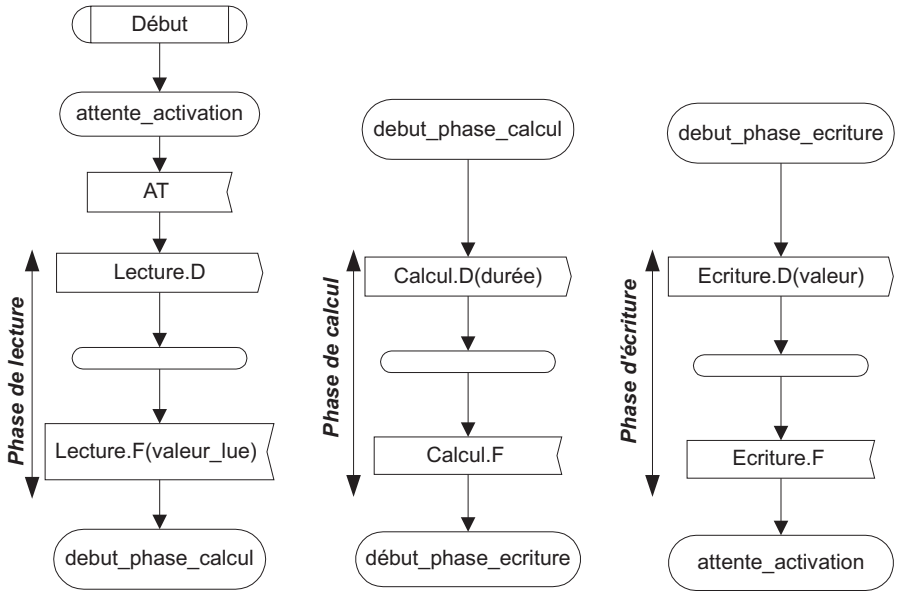


FIG. 4.1 – Modélisation d'un bloc en SDL

Notons que les phases d'écriture et de lecture peuvent être bloquantes, en fonction de mécanismes de coopération entre blocs. La modélisation de ces mécanismes repose sur la notion de *port de communication* et de *mode de coopération* notions qui sont définies dans la section suivante. Cette abstraction est similaire à celle qui a été proposée pour le langage de description d'architectures logicielles CLARA, développé à l'IRCCyN dans l'équipe Temps Réel (cf. [Durand, 1998]). Enfin, la phase de calcul utilise la ressource processeur de la station supportant le bloc pour le temps T spécifié. Un ordonnanceur gère l'accès au processeur. La phase de calcul est donc également bloquée en attente du signal Calcul.F.

4.2.1 Modélisation de la phase de calcul

Pour modéliser la phase de calcul d'un bloc, nous faisons l'hypothèse que les traitements sont déterministes c'est-à-dire que sous les mêmes conditions en entrée, ils fournissent les mêmes résultats en sortie. On peut donc représenter la phase de calcul sous forme d'une fonction F qui lie, à l'étape n, la valeur $s(n)$ de l'échantillon s_n d'une donnée résultat s aux valeurs de certains échantillons d'un ensemble de données d'entrée $\{e^j\}$. Nous désignons par la suite par E^j les échantillons de la donnée e^j utilisés dans le calcul.

$$s = F(E^1, ...E^l)$$

Plus précisément, pour décrire cette fonction, nous considérons un état interne à la fonction (x) qui contient toutes les informations nécessaires, liées au passé, pour définir les sorties à partir des données courantes et de l'état interne.

$$\left\{ \begin{array}{c} x(n) = G(E^1(n), .., E^l(n), s(n-1), x(n-1)) \\ s(n) = F(x(n), E^1(n), ..., E^l(n)) \end{array} \right\}$$

où $E^j(n) = \{e^j(i), .., .e^j(i+k_j)\}$ représentent l'ensemble des échantillons de la donnée d'entrée e^j disponibles entre la création de $s(n-1)$ et celle de $s(n)$.

[1] Le langage SDL (Specification and Description Language) permet de décrire les systèmes sous forme d'une description hiérachique de blocs échangeant des signaux. Le comportements des différents blocs est défini de manière formelle à l'aide d'automates communiquants.

Usuellement, les fonctions sont linéaires, la fonction de transfert d'un bloc, réduit à une entrée et une sortie, s'écrit alors de manière simplifiée grâce à l'opérateur de décalage q (cf[2]) :

$$H(q) = \frac{s(q)}{e(q)} = \frac{a_{N_e} q^{N_e} + ... + a_0 q^0}{b_{N_s} q^{N_s} + ... + b_0 q^0}$$

avec a_i et b_i appartenant à $\mathbb{R}$ et b_0 différent de 0. N_e et N_s représentent respectivement les nombres d'échantillons nécessaires de la donnée d'entrée e et de la donnée de sortie s. Ceci correspond à une relation linéaire sur les échantillons :

$$s(n) = \frac{1}{b_0}(-b_1 s(n-1) + ... + (-b_k) \cdot s(n - N_s) + a_0 e(n) + ... + a_n e(n - N_e))$$

La modélisation des fonctions de calcul sert à évaluer les propriétés importantes à respecter par l'échantillon s_n, à savoir :

- quelles données d'entrée servent dans le calcul de $s(n)$?
- quels sont les échantillons utiles pour ce calcul (échantillons de s et échantillons des données d'entrée) ?
- quelle est leur influence ?

4.2.2 Règles d'activations des blocs

Afin d'avoir une abstraction significative pour l'évaluation de propriétés de comportement d'une architecture opérationnelle, la modélisation des calculs permettant l'obtention d'échantillons de sortie d'un bloc doit être complétée par une spécification des règles d'activation des blocs qui appartiennent à l'une des deux classes suivantes :

- périodique,
- événementielle ; l'occurrence de cet événement peut, entre autres, être liée à l'arrivée d'une donnée (ou plus exactement, d'un échantillon d'une donnée) sur un port d'entrée du bloc (techniques de modélisation employées dans le langage CLARA, [Durand, 1998]).

Ces régles d'activation sont issues de la spécification fonctionnelle du système ; ces spécifications sont décrites pour chaque bloc, indépendamment des régles d'activation fournies pour les autres blocs. Elles décrivent donc une propriété imposée au comportement de l'architecture opérationnelle.
De manière générique, pour chaque bloc, on représente la synchronisation d'un bloc sur sa demande d'activation sous la forme d'attente de l'événement A_T dans le diagramme SDL de la figure 4.1.

4.2.3 Phase de lecture et écriture

La phase de lecture modélise l'activité d'un bloc pour réaliser l'entrée d'un échantillon de donnée d'entrée. Cette phase sera représentée de manière générique par la partie identifiée "Phase de lecture" dans le diagramme SDL de la figure 4.1. Elle se décrit par :

- l'émission d'un signal Lecture.D (demande de lecture),
- suivie d'une attente de l'arrivée du signal de fin de lecture Lecture.F (notons que ceci implique que la lecture peut être bloquante) ; la date de consommation de l'échantillon lu de la donnée d'entrée est celle de la réception du signal Lecture.F.

[2]La fonction de transfert des systèmes échantillonnés est généralement donnée à l'aide de la transformée de Fourrier (noté en z, voire Annexe A). La notation en q est équivalente ($H(q) = H(z^{-1})$) mais ne considère que des échantillons consécutifs et ne nécessite pas l'hypothèse d'échantillonnage périodique parfait.

De même, la phase d'écriture correspond à l'activité du bloc pendant qu'il réalise la sortie d'un échantillon d'une donnée de sortie. Elle se décrit, toujours de manière générique en SDL (voir partie identifiée "Phase d'écriture" dans la figure 4.1) et suivant le même principe :

- l'émission d'un signal Ecriture.D (demande d'écriture),
- suivie d'une attente de l'arrivée du signal de fin d'écriture Ecriture.f (l'écriture peut, elle aussi, être bloquante) ; la date de production de l'échantillon écrit de la donnée de sortie est celle de l'émission du signal Ecriture.D.

4.2.4 Coopération entre blocs

Le mode de transmission d'échantillons entre deux blocs relève d'un mode de coopération entre ces blocs, assimilable à la notion de *connecteur* utilisée dans les langages de description d'architecture ([Honeywell, 1998; Balarin, 1999; Durand, 1998]). Un mode permet de décrire les interactions et synchronisations entre la phase d'écriture d'un bloc et la phase de lecture d'un autre bloc lorsqu'il existe, au niveau fonctionnel, un flux de données les reliant. Chaque mode est représenté par au moins un *port de sortie* synchronisé sur la phase d'écriture du bloc émetteur et un *port d'entrée* synchronisé sur la phase de lecture du bloc destinataire. Dans une architecture opérationnelle, la transmission propre d'un échantillon repose sur des *services de transmission* interagissant avec port de sortie et port d'entrée. Ils sont implantés, par exemple, par les services de communication d'un exécutif multi tâches (boîte aux lettres, rendez-vous, ...) ou par les mécanismes protocolaires d'un réseau dans le cas où les blocs émetteur et destinataire sont distants.

La lecture ou l'écriture sur un port peut être bloquante ou non. Le port d'entrée (resp. le port de sortie) répond à la demande de lecture (resp. d'écriture) du bloc. Il reçoit (resp. émet) lui-même des données en provenance (resp. à destination) des autres blocs et peut posséder son propre mode d'activation (périodique ou événementiel).

Nous présentons ci-dessous quelques modes de coopération classiquement utilisés dans une version simplifiée. La mémorisation, le cas échéant, d'échantillons relève d'un buffer à une place et écrasement ; d'autres types de mémorisation sont modélisables (FIFO, LIFO, ... avec ou sans écrasement). Nous utilisons le formalisme SDL pour les représenter et, à chaque fois, nous donnons la vision "système" mettant en évidence les canaux d'interaction entre les blocs, les ports et le service de transmission ainsi que la vision "processus" des ports d'entrée et de sortie. Les interactions entre un bloc et un port de sortie (resp. entre un bloc et un port d'entrée) se font directement par l'intermédiaire des signaux SDL Ecriture.D (resp. Lecture.D) émis par le bloc et attendu par le port de sortie (resp. le port d'entrée) et Ecriture.F (resp. Lecture.F) émis par le port de sortie (resp. le port d'entrée) et attendu par le bloc. Le signal Ecriture.D est valué par la valeur de l'échantillon émis. Le signal Lecture.F est valué par la valeur de l'échantillon transmis au bloc destinataire et, pour des raisons d'évaluation du modèle, par son rang d'émission et son rang de consommation (on peut ainsi suivre les échantillons entre deux blocs et évaluer quels sont les échantillons perdus).

4.2.4.1 Mode "Push"

Dans ce mode, un bloc, émetteur d'échantillon (Bloc 1), produit des échantillons suivant sa propre loi d'activation (signal AT1). A chaque signal Ecriture.D reçu, le port de sortie du Bloc 1 fait appel (signal Request) au service de transmission pour acheminer l'échantillon sur le port d'entrée du bloc destinataire (Bloc 2). Le service de transmission signale (selon sa dynamique propre) au port d'entrée du Bloc2 (signal Ind) qu'un échantillon est arrivé. Le port d'entrée active alors le Bloc 2 (signal AT2) et attend que celui-ci soit prêt à lire (signal Lecture.D). L'échantillon est alors transmis au Bloc 2 (signal Lecture.F) et la coopération entre les deux blocs est terminée pour cet échantillon. L'activation du bloc 2 dépend donc de l'activation du bloc 1. Les figures 4.2 et modélisent ce mode de coopération.

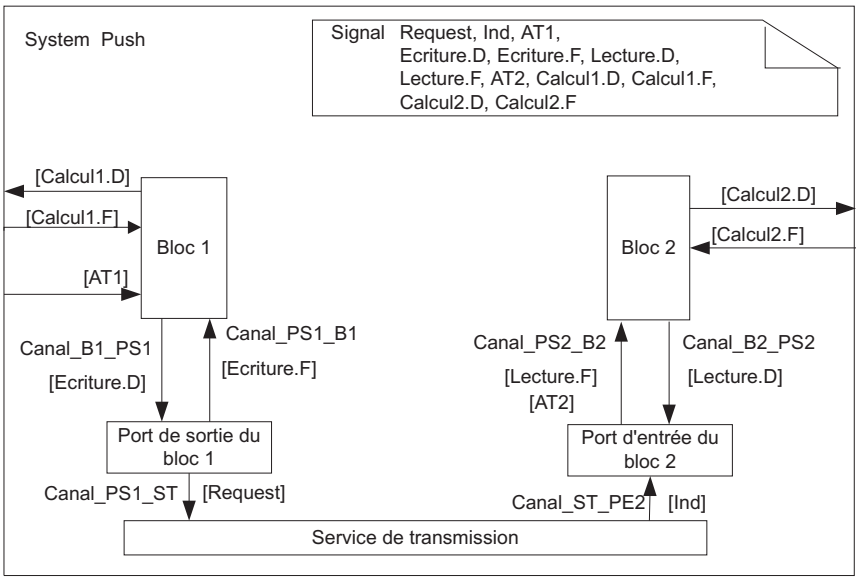


FIG. 4.2 – Mode "Push" - niveau système

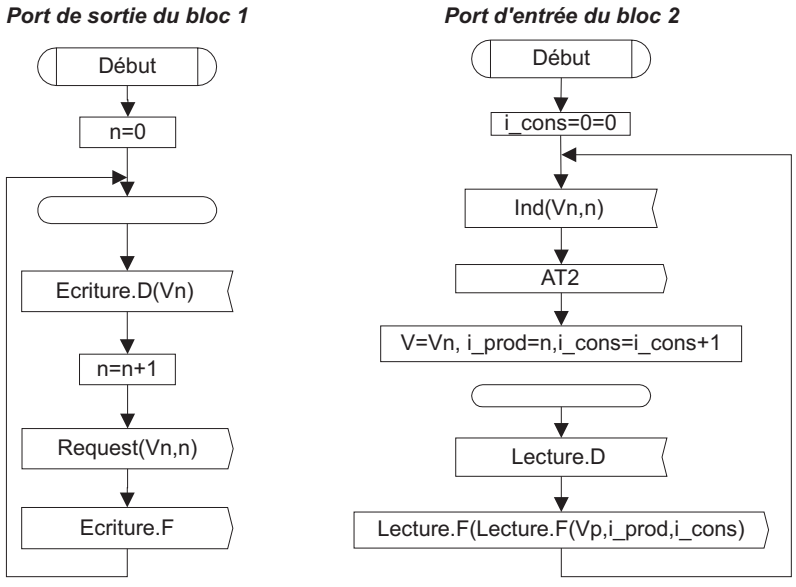


FIG. 4.3 – Mode "Push" - niveau processus

4.2.4.2 Mode “Pull”

Il s’agit en fait d’un mode Client / Serveur où le Bloc 1 émetteur d’échantillon est serveur et le Bloc 2, destinataire est client. Dans ce cas, l’activation du Bloc 1 dépend de la demande de lecture du Bloc 2. Les figures 4.4 et 4.5 représentent cette coopération.

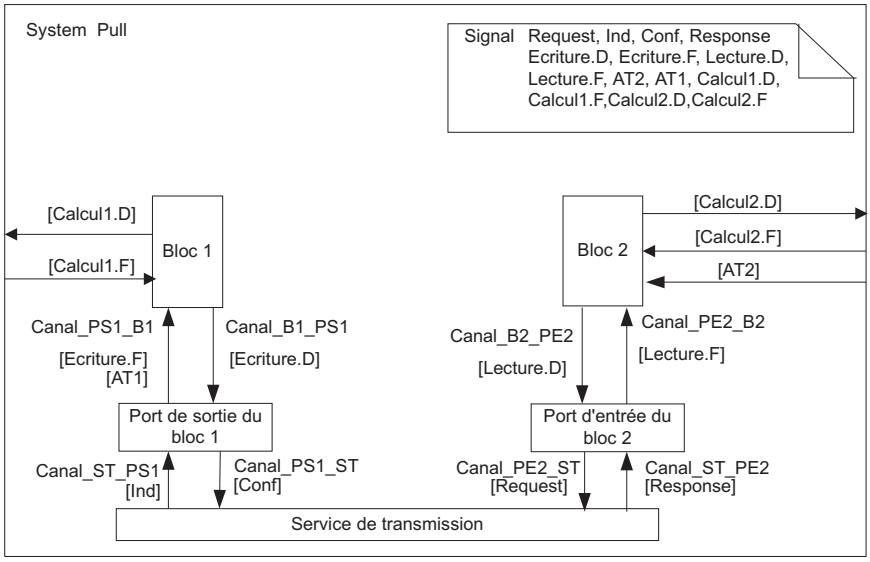


FIG. 4.4 – Mode "Pull" - niveau système

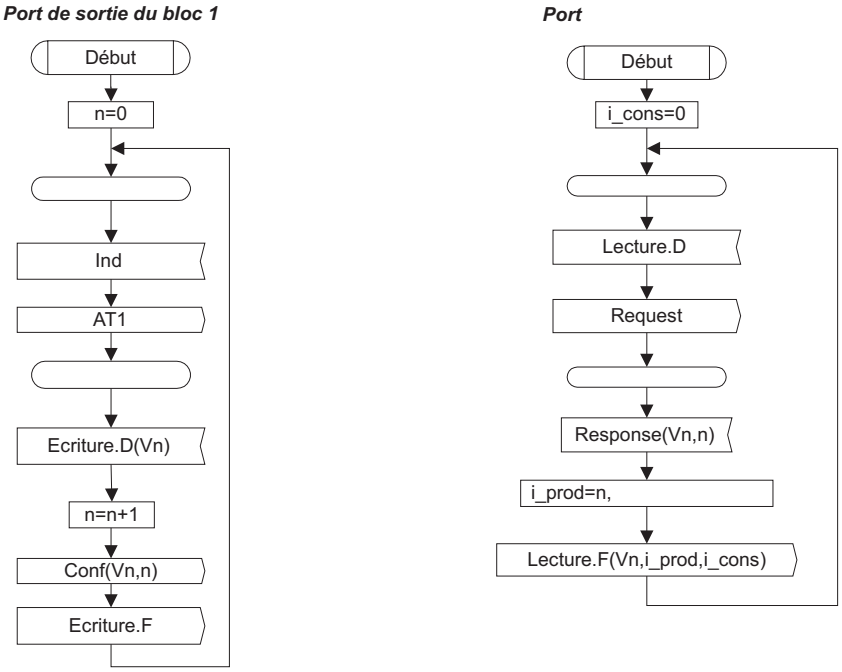


FIG. 4.5 – Mode "Pull" - niveau processus

4.2.4.3 Mode “Pull entre port d’entrée et port de sortie”

Un modèle Client / Serveur est fourni entre le port d’entrée (client) du Bloc 2 destinataire de l’échantillon et le port de sortie (serveur) du Bloc 1 émetteur de l’échantillon. Dans ce cas, Bloc 1 et Bloc 2 sont activés

indépendamment (signaux AT1, AT2). Ce mode est modélisé dans les figures 4.6 et 4.7.

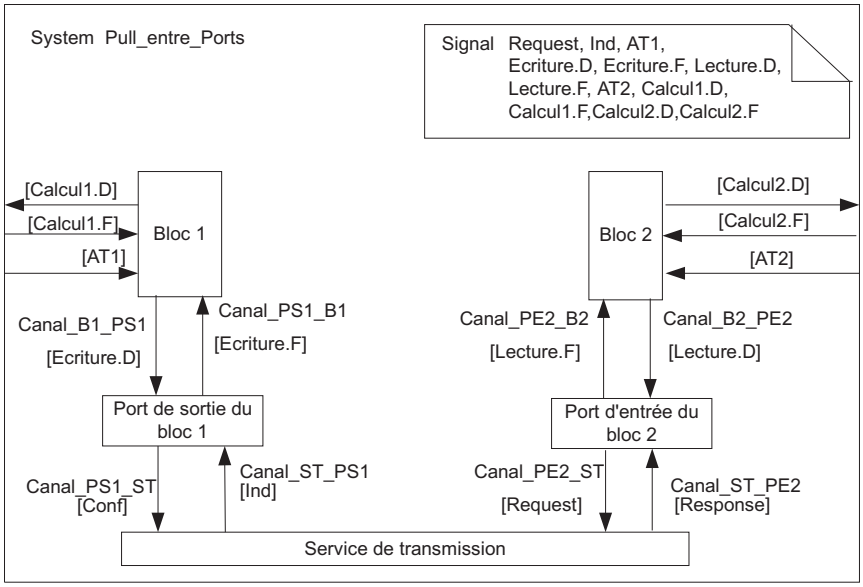


FIG. 4.6 – Mode "Pull entre port d'entrée et port de sortie" - niveau système

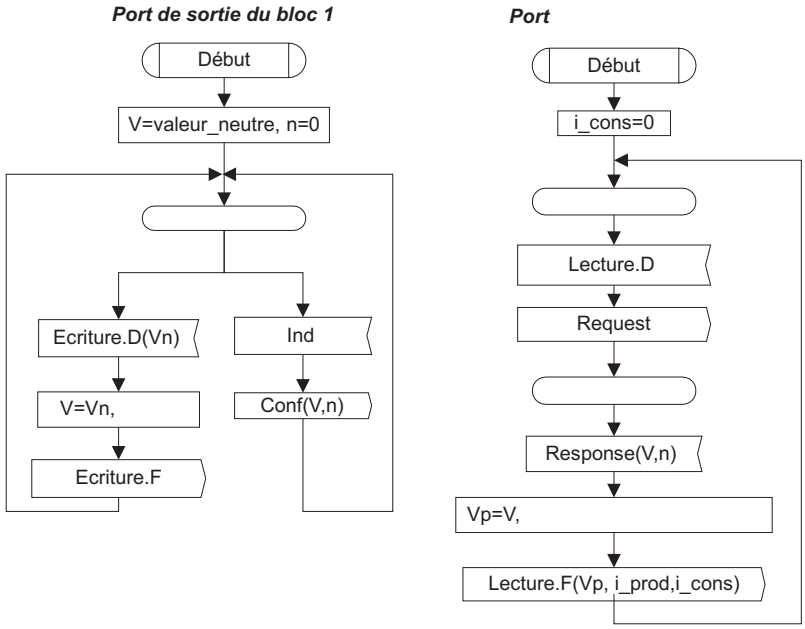


FIG. 4.7 – Mode "Pull entre port d'entrée et port de sortie" - niveau processus

4.2.4.4 Mode "Push entre port d'entrée et port de sortie"

Les activations de chaque bloc se font indépendamment. Le port d'entrée du Bloc 2 mémorise tout échantillon transmis, via le service de transmission, par le port de sortie du bloc 1. Il fournit le dernier échantillon transmis à chaque de mande de lecture du Bloc 2 (voir figures 4.8 et 4.9)

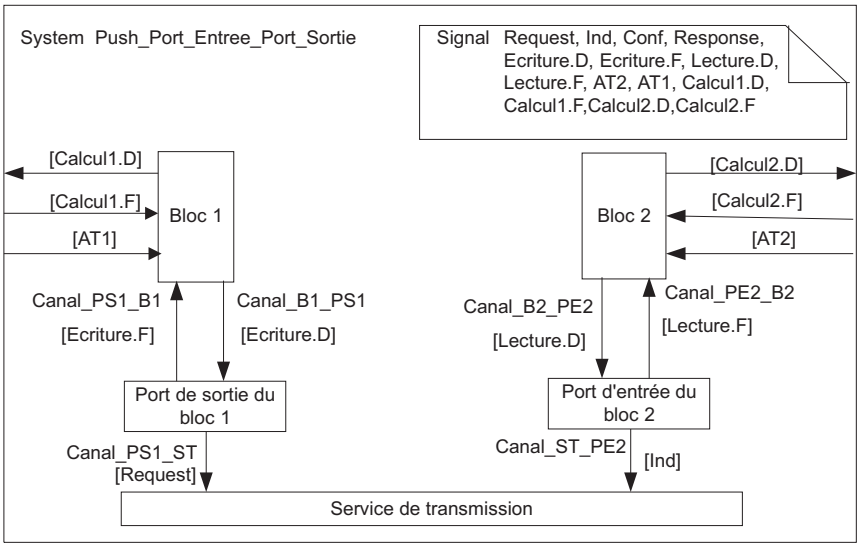


FIG. 4.8 – Mode "Push entre port d'entrée et port de sortie" - niveau système

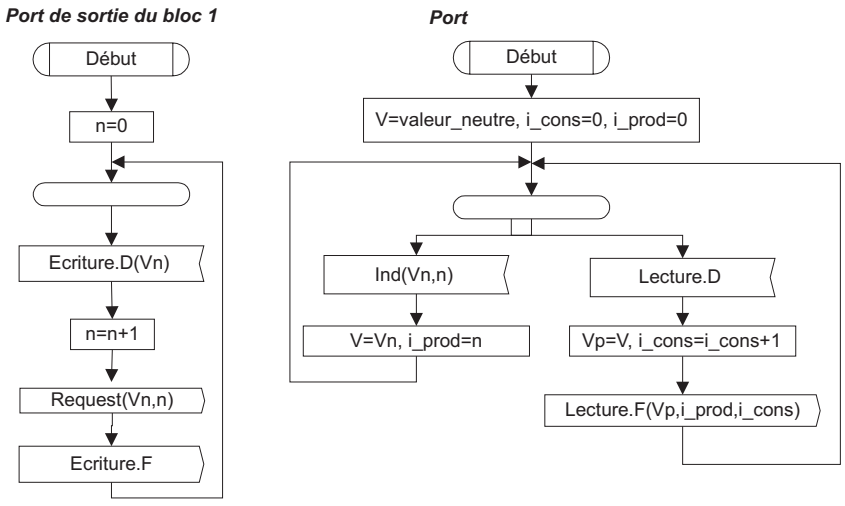


FIG. 4.9 – Mode "Push entre port d'entrée et port de sortie" - niveau processus

4.2.4.5 Mode “Push avec activation périodique du port de sortie”

Ce mode peut abstraire le fonctionnement d'un middleware qui réalise la construction de messagerie (frame packing) ainsi que l'émission automatique des trames ([Santos-Marquez *et al.*, 2003; Graefe and McKenna, 1993]). Dans ce cas, à chaque demande d'écriture du Bloc 1, son port de sortie mémorise l'échantillon et périodiquement, transmet un échantillon mémorisé (le dernier, dans la version illustrée aux figures 4.10 et 4.11) via le système de transmission, au port d'entrée du Bloc 2. Celui-ci le conserve et, dans la version modélisée, restitue le dernier échantillon mémorisé lors d'une demande de lecture du Bloc 2.

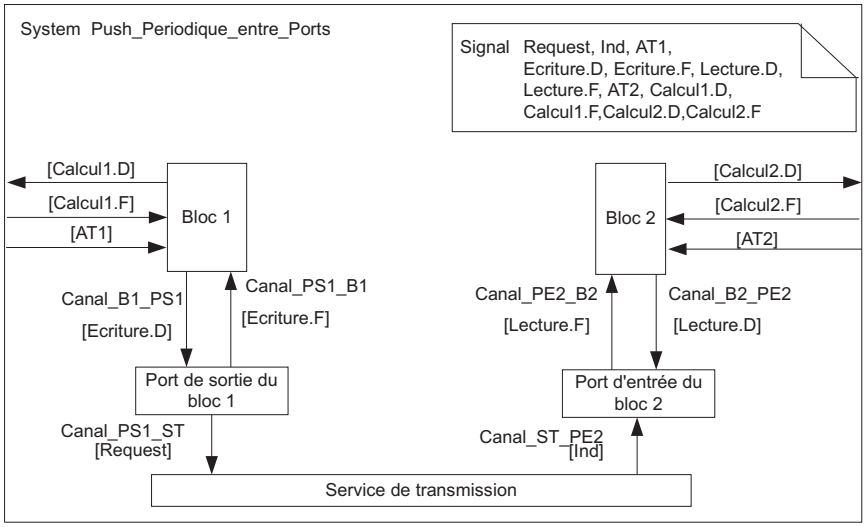


FIG. 4.10 – Mode "Push avec activation périodique du port de sortie" - niveau système

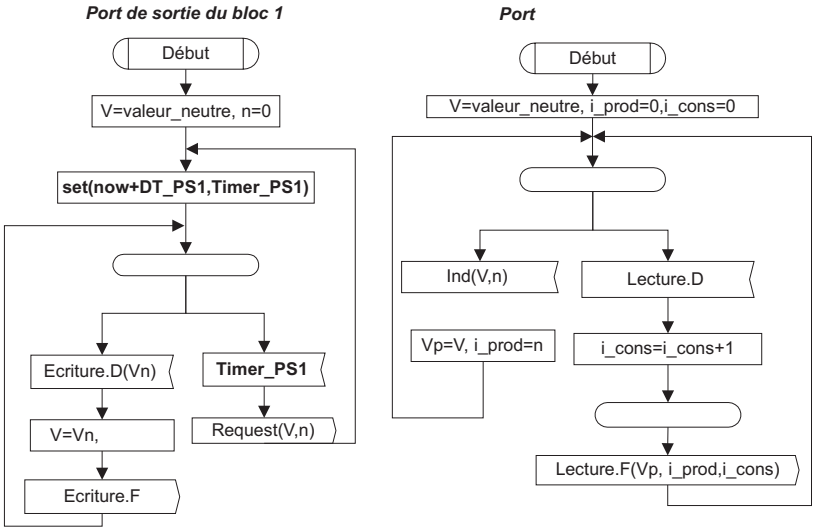


FIG. 4.11 – Mode "Push avec activation périodique du port de sortie" - niveau processus

4.2.4.6 Mode “Rendez-vous”

Chaque bloc est activé indépendamment. Les ports d'entrée et de sortie attendent respectivement les demandes d'écriture et de lecture des blocs 1 et 2. Un automate assure la coopération de type rendez-vous entre une demande de lecture et une demande d'écriture (attente des deux signaux Request1 et Request2)

puis émet les deux signaux Response1 et Response2. Cet automate représente, dans ce cas, le service de transmission. Ce modèle est présenté dans les figures 4.12 et 4.13.

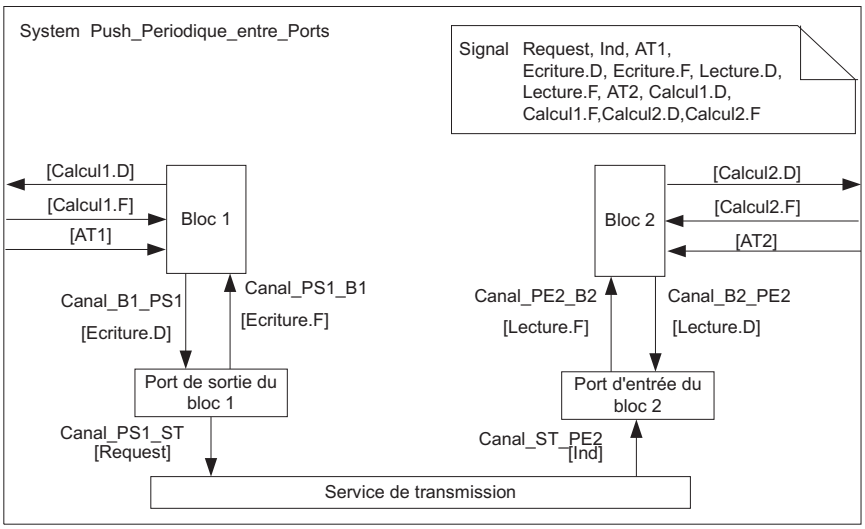


FIG. 4.12 – Mode "RendezVous" - niveau système

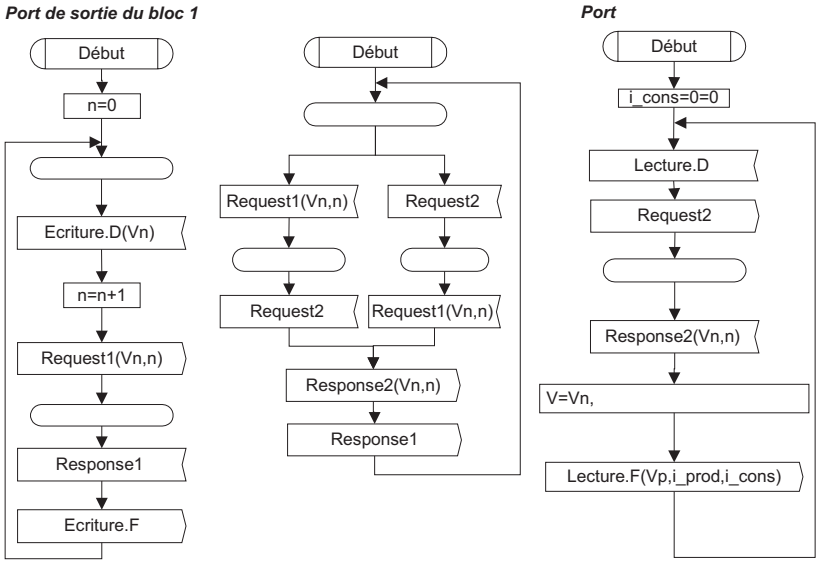


FIG. 4.13 – Mode "RendezVous" - niveau processus

4.2.4.7 Mode “Entrées multiples”

Le modèle illustré dans les diagramme des figures 4.14 et 4.15 représente un des modes possibles lorsqu'un destinataire consomme des données venant de plusieurs émetteurs. Plusieurs blocs sont producteurs d'échantillons, sous des règles d'activation indépendantes, à destination d'un unique bloc destinataire qui, lors d'une demande de lecture (Lecture.D) reçoit le dernier échantillon des données correspondant à chaque bloc émetteur.

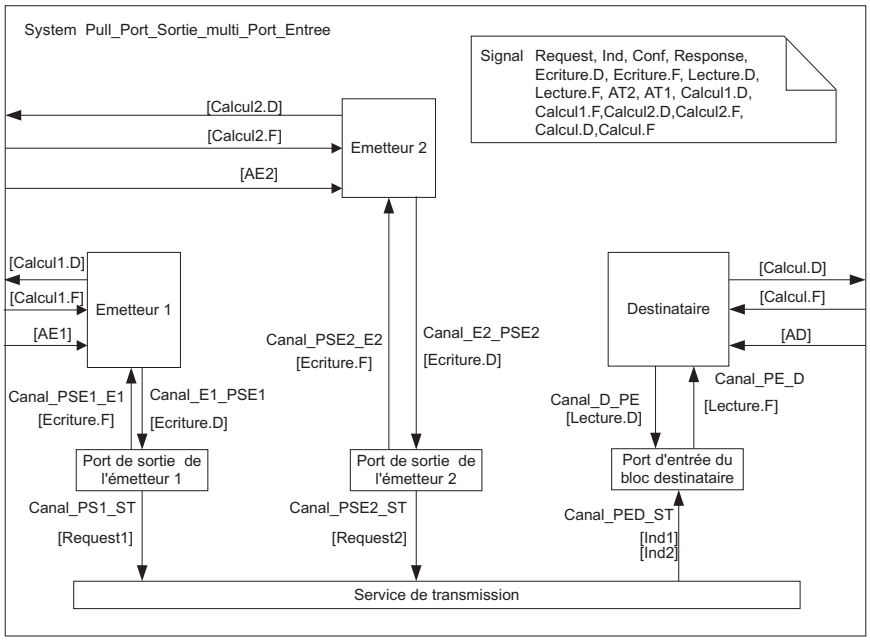


FIG. 4.14 – Mode "Entrées multiples" - niveau système

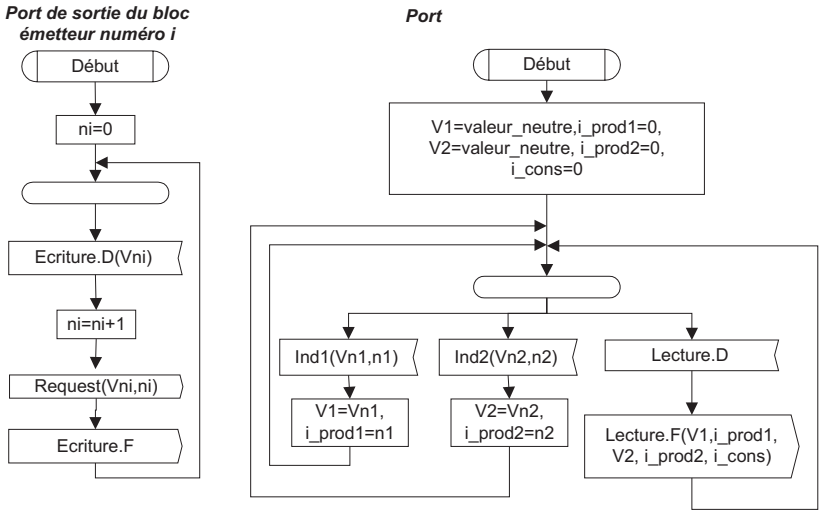


FIG. 4.15 – Mode "Entrées Multiples" - niveau processus

4.3 Présentation de quelques architectures de contrôle-commande

Une architecture est définie à partir d'un ensemble de blocs obéissant à des lois d'activation et interagissant entre eux par des modes de coopération. Nous proposons une représentation graphique de ces architectures en utilisant les modes de coopération présentés ci-dessus et en identifiant les types d'activation de chaque bloc. La syntaxe graphique est fournie à la figure 4.16.

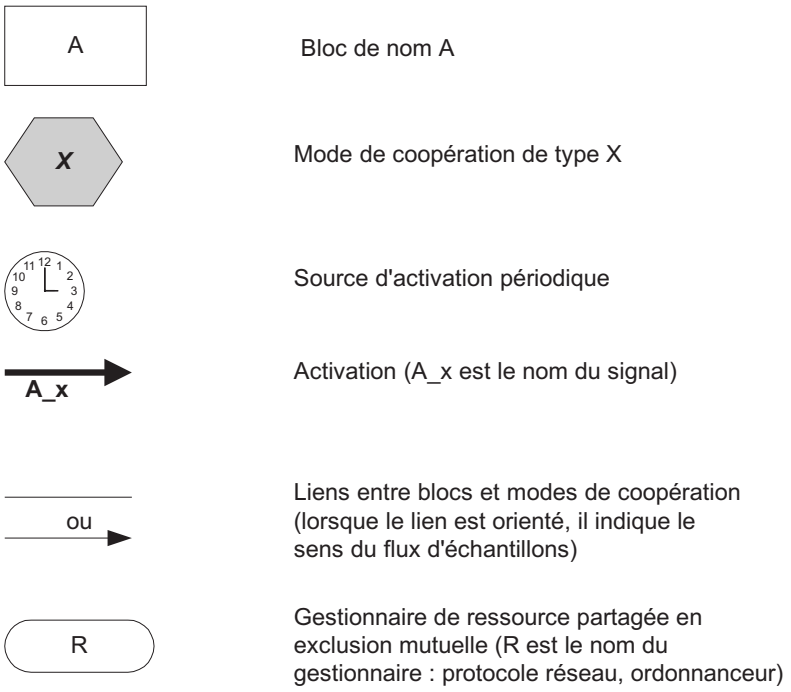


FIG. 4.16 – Syntaxe de la représentation graphique des architectures de contrôle-commande

4.3.1 Architectures de références

Les exemples figurant dans cette section donnent une représentation de l'architecture fonctionnelle, sans référence à un gestionnaire de ressource réseau ou processeur ; ceux-ci seront introduits à la section 4.4.

La première architecture de référence est dite "**au plus tôt**". Elle est synchronisée sur le capteur. Le capteur émet des informations de manière périodique. L'échantillon est envoyé à la commande qui émet à son tour un échantillon de commande à l'actionneur. Le schéma 4.17 présente cette architecture, on voit que les activations du calcul et de l'actionneur sont faites sur réception des messages.

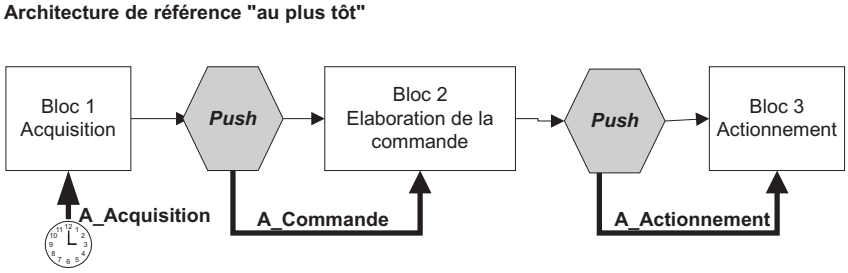


FIG. 4.17 – Architecture de référence "au plus tôt"

La seconde architecture est dite **“à traitement retardé**” (figure 4.18). Dans ce cas, l'acquisition et la commande sont synchronisées sur la même horloge. Par contre l'algorithme de calcul a été défini pour travailler sur la donnée reçue au cycle précédent. L'hypothèse sous-jacente est qu'un échantillon ne peut être reçu avec plus d'un cycle de retard.

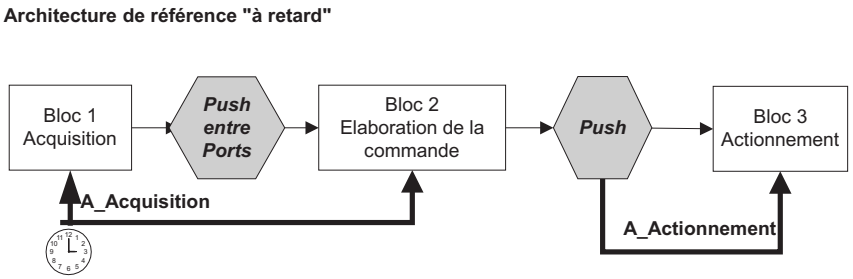


FIG. 4.18 – Architecture de référence "à retard", la loi de calcul travaille sur la donnée du cycle précédent dont elle dispose

4.3.2 Exemples d'architectures élémentaires

La première architecture (4.19) fait apparaître un mode de coopération de type client/serveur entre l'acquisition et l'élaboration de la commande. Le capteur devient alors un serveur d'information et la loi de commande en est le client. Dans cette architecture tous les traitements sont activés par la loi de commande, elle-même étant activée périodiquement.

La seconde architecture (4.20) présente une activation de la loi de commande sur réception de message en provenance du bloc d'acquisition. La différence par rapport à l'architecture “au plus tôt” réside dans l'activation périodique du bloc d'actionnement et un mode client/serveur entre bloc d'élaboration de la commande et bloc d'actionnement.

Enfin, la dernière architecture (4.21) fait apparaître des activations périodiques indépendantes pour les blocs d'acquisition et d'élaboration de la commande, tandis que le bloc d'actionnement est activé par l'arrivée d'un échantillon.

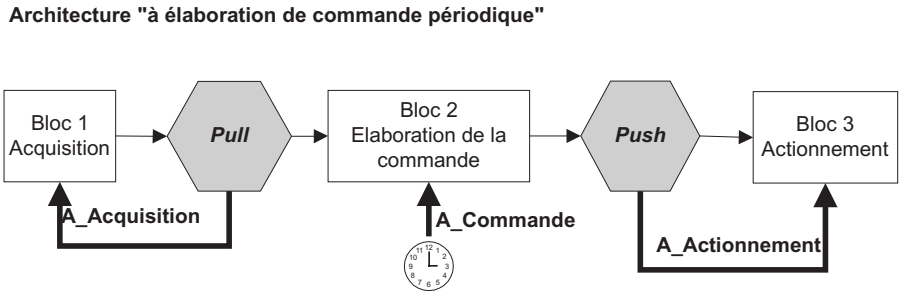


FIG. 4.19 – Elaboration d'une commande périodique

Enfin, la dernière architecture (4.22) correspond à un système MIMO (pour Multiple inputs/Multiple outputs). Elle utilise également les modèles présentés dans la section précédente.

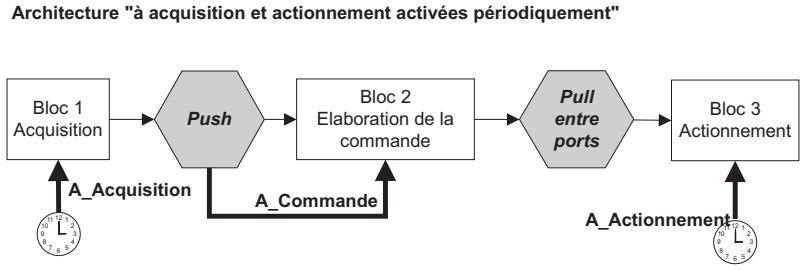


FIG. 4.20 – Acquisition et actionnement périodiques

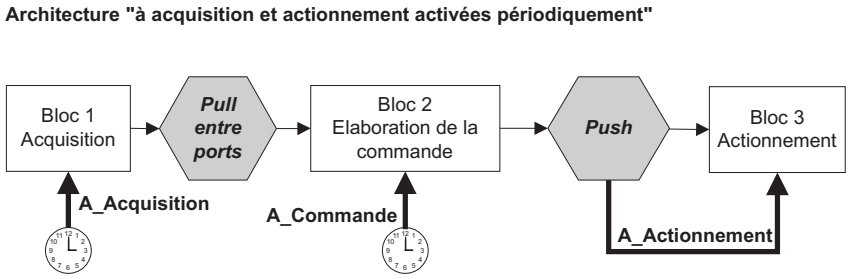


FIG. 4.21 – Acquisition et commande périodiques

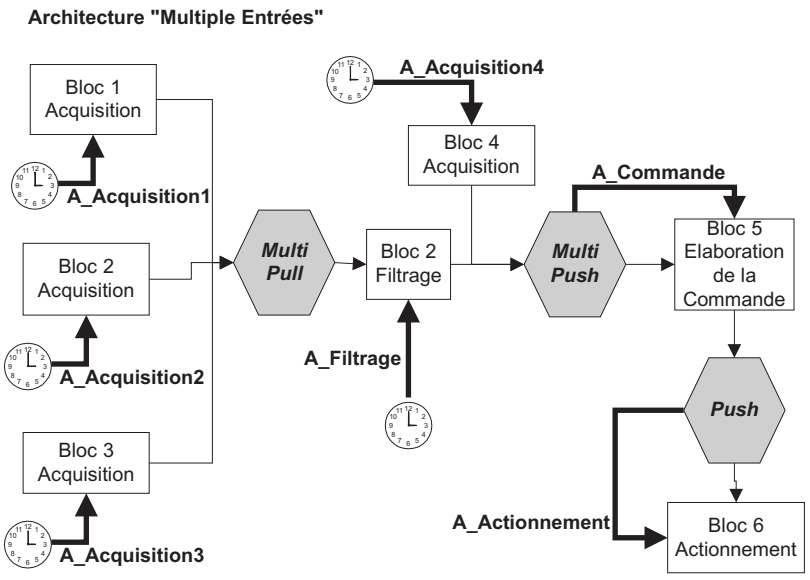


FIG. 4.22 – Exemple de système MIMO

4.4 Prise en compte de l'Architecture Matérielle

Les modèles d'architectures précédents ne font pas apparaître de consommation de temps dans chaque entité présente. Il s'agit de modèle du comportement et des interactions entre entité.

Pour évaluer les dates d'occurrence des événements liés à chaque échantillon (production / consommation), il est nécessaire de prendre en compte le temps physique. Ceci se fait sous deux formes :

- le temps de consommation des ressources (temps d'occupation d'un processeur par la partie traitement d'un bloc, temps de transmission sur un réseau au niveau du service de transmission)
- le temps de blocage pour l'accès à une ressource ; ces temps viennent des stratégies d'ordonnancement locales ou des protocoles d'accès au médium de communication.

Dans le cas des systèmes d'exploitation temps réel, un ordonnanceur décide, suivant une politique donnée et les paramètres des différentes activités à executer, comment attribuer le processeur aux activités le demandant. Une grande partie des politiques utilisées reposent sur l'utilisation de priorité. Le calcul des dates d'utilisation du processeur pour une activité donnée, nécessite donc la connaissance de l'ensemble des activités susceptibles d'utiliser le processeur. La modélisation de l'ordonnanceur peut être faite grâce à un automate gérant les différentes demandes de traitement. De nombreuses études montrent comment procéder à une telle modélisation ([Trinquet, 2000; Castelpietra *et al.*, 1999a; Jumel, 1999; Colin and Puaut, 2000; Trinquet and Elloy, 1999; Puaut, 2001; Castelpietra *et al.*, 1999b]). De la même manière, un automate peut décrire la statégie d'accès au médium de communication.

Nous pouvons alors modéliser les architectures opérationnelle de contrôle-commande correspondant aux architectures fonctionnelles présentées dans la section précédente. Par exemple, à la figure 4.23, nous proposons une version mono-processeur de l'architecture "au plus tôt" de la figure 4.17 où le gestionnaire de la ressource processeur est représenté.

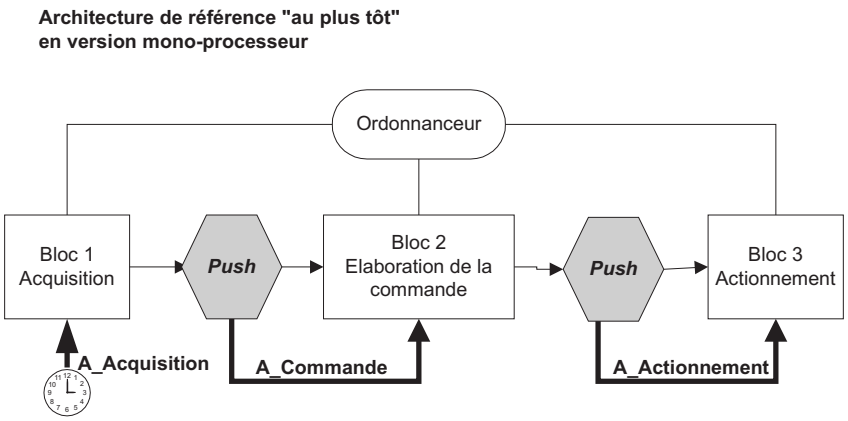


FIG. 4.23 – Implantation centralisée du modèle "au plus tôt"

Les figures 4.24 et 4.25 montrent respectivement une implantation distribuée des architectures fonctionnelles dites "à élaboration de commande périodique" et "à acquisition et actionnement activées périodiquement" des figures 4.19 et 4.20.

La quantification de la durée d'un traitement peut être obtenue, par test, par simulation, ou encore, grâce à des techniques d'analyse détérministes ou probabilistes (cf [Martí *et al.*, 2002b; Liu, 2000; Sanfridson, 2000; Puaut, 2001; Cottet *et al.*, 2000; Stankovic *et al.*, 1998; Lian *et al.*, 1999; Song *et al.*, 1999; Castelpietra *et al.*, 2002]). Cette caractéristique peut être exprimées de nombreuses manières (cf [Vega, 1996]). Les plus courantes sont sous forme :

- de bornes

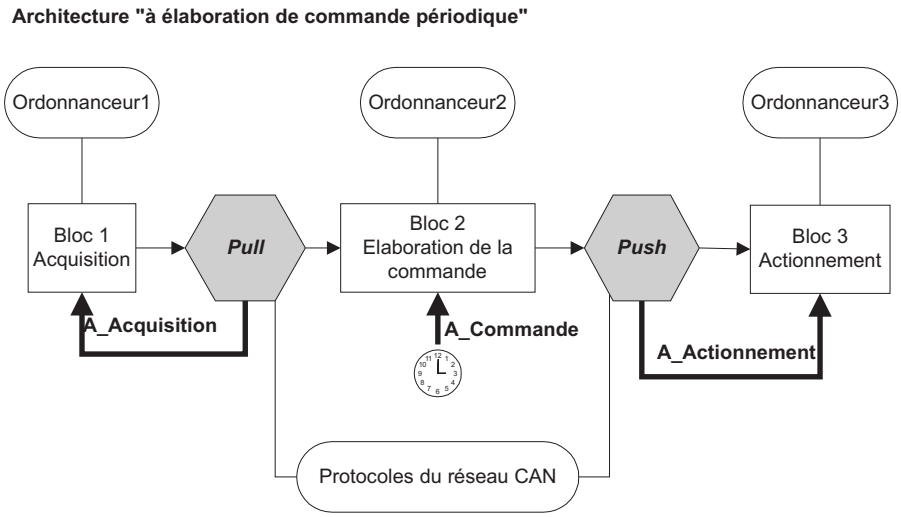


FIG. 4.24 – Implantation distribuée autour du réseau CAN

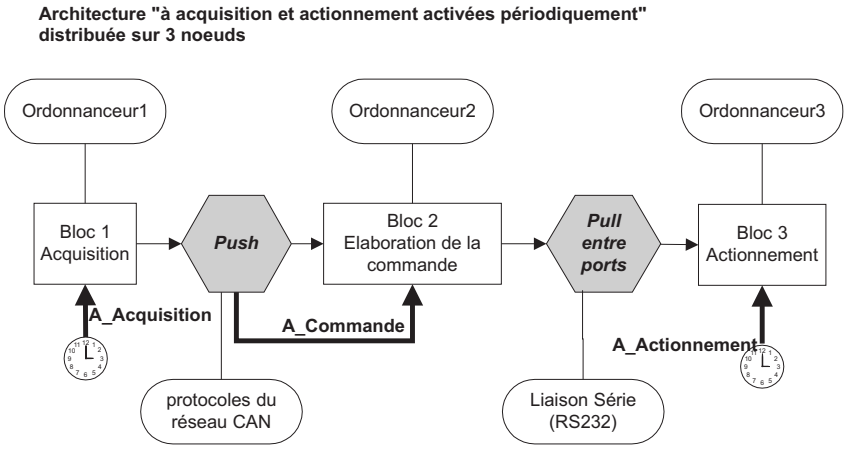


FIG. 4.25 – Implantation distribuée (2 stations connectées sur le réseau CAN et 2 stations reliées par une liaison série)

– de distributions

Les bornes sont simples à déterminer, il s'agit d'obtenir les temps de traitement maximum et minimun qui peuvent être obtenus pour un service et une architecture donnée. La détermination de bornes liées au temps de traitement est une obligation dans le cas des systèmes temps réel (cf. [Liu, 2000; Stankovic *et al.*, 1998]). Ces bornes sont susceptibles d'être fonction de paramètres extérieurs comme la charge ou le temps. Les modèles sont, dans ce cas, plus difficiles à obtenir mais la précision est alors bien plus grande (c'est à dire que l'on donne les meilleures bornes possibles) (cf. [Migge, 1999; Redell and Sanfridson, 2002; Redell and Törngren, 2002]). La connaissance de bornes, même si elle est nécessaire, ne suffit pas à caractériser les temps de traitement. Des propriétés complémentaires peuvent être données, comme la moyenne, la médiane, les écarts types, qui sont des valeurs généralement calculées par tests sur des systèmes existants. Ils peuvent aussi posséder des propriétés remarquables comme l'existence de motifs qui se répéte périodiquement. On peut également modéliser l'évolution dans le temps de certains temps de traitement. Ceci correspond notamment à l'existence d'une contrainte sur la différence maximale de temps de traitements entre deux occurences successives.

4.5 Conclusions

Modéliser les propriétés temporelles des architectures de contrôle-commande numériques est une activité complexe qui doit prendre en compte deux aspects : d'une part, les caractéristiques matérielles impactant les durées d'exécution des activités (temps de calcul nécessaire à un algorithme, temps de transmission d'un message sur un médium de communication...) et, d'autre part, les règles d'activation des différents activités (en particulier des traitements) ; l'ensemble permet d'évaluer les dates d'occurrences des événements critiques associés au système régulé.

Dans ce chapitre, nous avons identifié les points importants de cette modélisation. En particulier, nous nous sommes particulièrement intéressés à la représentation de règles d'activations complexes qui sont classiques dans l'étude des applications temps réel et qui, par contre, sont souvent occultées lors de l'analyse de performances des applications de contrôle-commande. Les concepts de base sont centrés sur la notion d'information et, plus spécifiquement, sur celle d'échantillon.

On peut constater que, en utilisant ces principes, la modélisation, en vue d'une analyse hors ligne, de l'architecture opérationnelle d'un système de contrôle-commande, quelle que soit sa complexité (centralisée ou distribuée) est toujours possible car elle fournit les moyens d'observer les événements de création (production) et utilisation (consommation) des échantillons. Seuls ces événements ont une pertinence pour le type d'étude considéré. Sur la base de cette modélisation, dans la suite du document, nous montrons comment mesurer, quantifier, analyser, évaluer l'influence des choix d'implémenation sur la qualité de la régulation dans les chapitres 5 et 6.

Chapitre 5

Mesure de l'influence de l'implémentation sur la qualité de service de la régulation

5.1 Introduction

Dans ce chapitre, nous montrons comment évaluer dans la pratique, quantitativement, l'influence d'une implémentation sur la qualité d'un système. Les problèmes engendrés vont de la dégradation de la qualité d'une régulation à la perte de la contrôlabilité du système (c'est-à-dire à l'instabilité du système controlé).

Pour comprendre le phénomène, il suffit de se placer au niveau des actionneurs, on applique sur le système réglé une certaine commande. Cette commande est obtenue à partir d'une représentation de l'état du système. Cet état peut être une représentation exacte du système, dans le cas de commande par retour d'état, ou approchée, par exemple dans le cas des commandes PID (Proportional Integrator Derivator) où l'on travaille uniquement sur la différence entre la consigne et la sortie du sytème.

L'existence de retard ou de gigue sur les informations manipulées dans le système réglant entraîne une incohérence dans la représentation du système, interne aux algorithmes de commande. Dans le cas d'un retard, l'état considéré correspond à un état passé du système. Pendant ce laps de temps, que l'on qualifie de "temps mort" en automatique, le système a évolué et la commande n'est donc pas adaptée à l'état actuel du système. De la même manière, l'existence d'une gigue sur les dates d'acquisition des grandeurs peut entraîner une incohérence similaire. En effet, la définition de la loi commande repose sur des calculs qui peuvent être assimilés à des calculs approchés de dérivées ou de primitives. Il est donc important que les intervalles de temps qui séparent les acquisitions soient conformes aux hypothèses sous-jacentes aux approximations sans quoi les résultats des calculs seraient incohérents.

Nous présentons, dans la première section de ce chapitre, comment par une analyse mathématique, il est possible de réaliser cette évaluation. Cette méthode repose néanmoins sur de fortes hypothèses. Aussi, lorsque ces hypothèses ne sont pas garanties, il est nécessaire de mettre en oeuvre des simulations. Matlab/Simulink est un outil fortement utilisé dans la communauté de l'automatique, aussi, avons-nous proposé une technique de modélisation sous Matlab/Simulink permettant de prendre en compte une architecture d'implantation du système réglant dans la modélisation du système régulé. Ce simulateur est présenté à la section 5.4 tandis que, dans la section 5.4, son application à des exemples traditionnellement étudiés en automatique montre quels types de résultats il est possible d'obtenir. Notons que quelle que soit la technique utilisée (analyse mathématique ou simulation, seule la caractérisation de dates d'occurrences d'événements significatifs est utile pour mesurer l'influence de l'implémentation sur la qualité du système réglé. De plus, dans ce qui suit, nous ne prenons en compte que les fautes dues aux fautes temporelles des événements liés aux échantillons "transitant"

dans le système (production, consommation d'échantillon, cf. chapitre 4).

5.2 Evaluation analytique

L'exploitation analytique des modèles de l'architecture support (chapitre 4) permet d'obtenir des propriétés sur les dates d'occurence des évènements critiques (acquisitions et commandes) qui vont influer sur le système réglé. Nous considérons comme seules fautes possibles l'existence de fautes temporelles liées aux échantillons qui transitent dans le système. Dans le cas où les mécanismes de coopération n'engendrent pas la perte des données à l'intérieur du système de contrôle (sous forme d'un écrasement dans le buffer), ces différentes fautes vont se traduire par l'existence d'un phénomène de retard sur la commande et/ou d'un retard et d'une gigue sur les acquisitions (voir chapitre 3).

5.2.1 Sensibilité aux retards

De nombreuses études sur la commande des systèmes dits "à retard" existent dans la littérature ([Martí *et al.*, 2001b; Martí *et al.*, 2001a; Ray, 1994; Kim *et al.*, 1996; Lee *et al.*, 1994; Akian *et al.*, 1998; Hristu, 2000; Moon *et al.*, 1998, ...]). L'objectif de ces travaux est généralement de proposer de nouvelles lois de commande adaptées à la présence de retard. Par contre, notre objectif est différent, puisqu'il s'agit de comprendre l'influence des retards dûs à une implémentation pour un système et une commande donnée et donc de fournir des moyens pour permettre à un concepteur de spécifier une (des) loi (s) de commande et une architecture d'implémentation garantissant une qualité spécifiée du système.

Dans cette section, nous distingons retards constants et retards variables car les outils mathématiques d'analyse ne sont pas identiques. Comme présenté dans le chapitre 3, les retards constants sont très simples à modéliser et correspondent à un phénomène de temps mort. Au contraire, même si la présence d'un retard variable peut être modélisé (cf. section 3.1.2), son influence se révèle beaucoup plus délicate à étudier (cf. section 5.2.1.2).

5.2.1.1 Evaluation de l'influence d'un retard constant

Le fonctionnement d'un système en présence d'un retard constant peut être évalué analytiquement si toutes les paramètres du modèle sont calculables. Comme indiqué en Annexe A, la fonction de transfert (définie à partir de la transformée de Laplace) qui décrit un système bouclé est de la forme :

$$G(p) = \frac{G_{cd}(p)}{1 + G_{cd}(p)G_{cr}(p)}$$

où $G_{cd}(p)$ est la fonction de transfert de la chaîne directe et $G_{cr}(p)$ celle de la chaîne de retour. En présence d'un retard τ_{cd} dans la chaîne directe et τ_{cr} dans celle de retour. La fonction de transfert du système à retard est :

$$G(p) = \frac{G_{cd}(p) \cdot e^{-\tau_{cd} \cdot p}}{1 + G_{cd}(p)G_{cr}(p)e^{-(\tau_{cd}+\tau_{cr}) \cdot p}}$$

Connaissant l'entrée appliquée, on peut en déduire l'évolution du système par une inversion de la transformée de Laplace. On obtient alors une représentation analytique de l'évolution du système au cours du temps, notée $y(t)$. Dans le cas des systèmes échantillonnés, un raisonnement analogue peut être fait sur la transformée de Fourrier du système mais en considérant uniquement des retards multiples de la période.

La représentation d'état des systèmes échantillonnés en présence de retard (3.1) peut servir également à obtenir la réponse du système mais l'analyse est décomposée en une partie discrète pour chaque échantillon

et une partie continue entre deux échantillons ; la forme de la réponse, bien que plus complexe, reste analysable.

De la même manière, les différents critères de qualité peuvent être calculés. Néanmoins, il est impossible de les présenter sous des formes closes. Cependant, des résultats intéressants sur la stabilité des systèmes ont été montrés et peuvent être utilisés dans notre approche. L'annexe A montre notamment que la stabilité des systèmes est liée au dénominateur de la fonction de transfert. On peut donc en conclure immédiatement que la stabilité est fonction du *retard maximal* ($\tau_{cd} + \tau_{cr}$). Ce résultat n'est pas aussi intuitif que l'on pourrait le croire, il signifie que dans le cas des systèmes linéaires (seuls représentables sous la forme proposée) la place des retards à l'intérieur de la chaîne de retour et de la chaîne directe n'influe pas sur la stabilité. En particulier, si les retards se décomposent en de nombreux retards à différents niveaux du système, seul le retard global résultant a un impact sur la stabilité (c'est-à-dire la somme des retards obtenue par propriété de commutation dans le cas des systèmes linéaires).

Inversement, il est possible d'en déduire *le plus grand retard admissible* vis à vis de la propriété de stabilité. Le critère le plus adapté est la *marge de phase* (liée à la représentation fréquentielle des systèmes). La marge de phase doit être supérieure à une certaine valeur ϕ_{min}[1] pour garantir la stabilité. La présence d'un retard constant agit directement sur la marge de phase (ϕ_{marge}). On a donc une relation directe entre stabilité et retard constant.

Soit $G(p)$ la fonction de transfert d'un système sans retard. La marge de phase ϕ^G_{marge} du système G est obtenue à partir du diagramme de bode à la fréquence de coupure w_c (qui correspond à un gain unitaire). La marge de phase du système G retardé de τ (de fonction de transfert $G(p)e^{-\tau p}$) est simplement donnée par :

$$\phi^{G,\tau}_{marge} = \phi^G_{marge} - w_c \cdot \tau$$

On peut donc en déduire le retard maximal toléré avant la perte de stabilité :

$$\phi^{G,\tau}_{marge} \geq \phi_{min}$$

implique

$$\tau \leq \frac{\phi^G_{marge} - \phi_{min}}{w_c}$$

On en déduit donc le retard maximal admissible :

$$\tau_{max} = \frac{\phi^G_{marge} - \phi_{min}}{w_c}$$

Nous avons donc un moyen rapide pour calculer le plus grand retard constant admissible. La technique présente cependant 2 inconvénients :

- Par définition, le calcul n'a de sens que si le retard est constant, cette étude n'est donc pas suffisante pour conclure à une borne maximale sur les retards admissibles. En effet, dans le cas général, les retards dûs à l'implantation ne sont pas constants et cette étude ne permet pas de conclure sur la stabilité du système.
- La fonction de transfert $G(p)$ doit correspondre au système régulé en boucle ouverte (c'est-à-dire correcteur inclu). Dans le cas qui nous intéresse le correcteur est n'est pas continu (algoritme de commande), il n'est donc pas possible de l'inclure dans $G(p)$ sans procéder à une importante simplification en le transformant en un correcteur continu équivalent.

La marge de phase est cependant très intéressante car elle permet de connaître la *forme de la réponse du*

[1] Par définition, $\phi_{min} = 0$ degrée. Cependant pour plus de sécurité, on fixe une marge de sécurité comme par exemple $\phi_{min} = 50$ degrées.

système, elle peut donc aussi servir de critère de qualité équivalent au dépassement. Ce critère a, en particulier, été choisi comme critère de qualité principal dans une étude dont cette thèse peut être vue comme une continuation (thèse de I. Blum, cf. [Juanole and Blum, 1999]).

Si l'on considére un autre critère comme le temps de réponse dans le cas du changement de consigne, où un critère intégral de qualité sur le fonctionnement moyen du système (comme par exemple l'IAE, cf. 2), il est très difficile de lier analytiquement la qualité obtenue à la présence de retard. Par contre, on peut supposer l'existence d'une dégradation progressive des performances du système avec l'augmentation du retard.

5.2.1.2 Evaluation de l'influence d'un retard variable

L'étude de l'influence d'un retard variable est beaucoup plus délicate à mettre en oeuvre. La modélisation d'état (cf. section 3.1) permet de prendre en compte des retards variables (elle s'applique, évidemment également aux retards constants). Les modèles simplifiés par contre ne le permettent plus.

Si les retards sont variables, ils correspondent pour le système régulé à la présence d'une suite de retards $\{\tau_k\}$ appliqués aux différents instants t_k. On cherche donc à caractériser la qualité du système vis à vis de cette suite. Dans le cas où cette suite est parfaitement connue (connaissance de chaque valeur de la suite), il est possible de trouver la réponse du système en l'absence de perturbation et d'en déduire la zone de fonctionnement possible pour un modèle donné de perturbations. Les valeurs des différentes métriques de qualité peuvent alors être calculées.

Si cette suite ne peut pas être connue à l'avance, certaines propriétés caractéristiques de cette suite peuvent être utilisées pour évaluer les évolutions possibles du système. Intuitivement les propriétés importantes sont :

- les bornes (min et max) des valeurs de la suite,
- la moyenne des valeurs de la suite.

Ces propriétés peuvent être qualifiées de propriétés stationnaires car elles sont définies pour la totalité de la suite. Par contre, elles ne permettent pas de modéliser les fluctuations des valeurs. Or, un changement de consigne se stabilise en un nombre déterminé de cycles de traitement. La qualité obtenue dans ce cas ne dépend donc pas de propriétés stationnaires mais de propriétés définies sur une fenêtre temporelle restreinte. Or même si le nombre de cycles à considérer est faible, le nombre de combinaisons possibles est trop élevé pour permettre une étude exhaustive. En effet, si le nombre de valeurs possibles des retards est petit, par exemple 10 valeurs, le nombre de combinaisons possibles sur cinquante cycles atteint néanmoins 10^{50}. Même si, pour chacune de ces combinaisons, il est très simple de calculer la qualité obtenue, il est impossible de tester tous les cas.

On pourrait être tenté de conclure rapidement en considérant que le pire cas est obtenu pour un retard constant maximal sur la fenêtre temporelle, mais ce n'est pas forcément vrai (cf. l'une des premières études sur le sujet faite à Lünd ([Andreff, 1994]) et les travaux de thèse de Martin Törngren [Törngren, 1995]).

Comme dans le cas des retards constants, on dispose malgré tout d'un certain nombre de résultats. Une première famille de résultats ([Ayyagari and Ray, 1993; Ray, 1994; Chapellat and Dahleh, 1992; Park *et al.*, 1997; Kim *et al.*, 1996; Hristu, 2000; Moon *et al.*, 1998; Moon and Kwon, 1996]) permet de conclure sur la stabilité du système en connaissant les *bornes de la suite*. Il s'agit de conditions suffisantes mais non nécessaires. C'est à dire que l'on peut prouver que le système régulé est stable pour $\tau_k \in [\tau_{min}, \tau_{max}]$ (avec généralement $\tau_{min} = 0$). Par contre, il est plus difficile de prouver que ces bornes sont les meilleures possibles et il est possible que la suite quitte ces bornes sans que le système régulé devienne instable. Il est à noter que le calcul de ses bornes, basé sur une représentation matricielle continue du système se révèle complexe et pose le même problème évoqué dans le cas constant : *il est nécessaire de disposer d'une modélisation continue de l'ensemble du système régulé.*

Dans le cas où la suite possède un *motif qui se répéte périodiquement*, on dispose également de résultats spécifiques. En effet la définition récursive des algorithmes de contrôle, permet d'obtenir dans ce cas des

propriétés de calcul intéressantes et, par conséquent, d'identifier des critères de stabilité plus précis (cf. [Moon and Kwon, 1996]).

De manière générale, les différents résultats concernant la stabilité des systèmes hybrides variant dans le temps (constitués d'un système continu en interaction avec un système discret) peuvent être appliqués pour donner des conditions de stabilité (cf. [Park *et al.*, 1995]). Les techniques proposées sont cependant difficilement exploitables même si elles constituent des outils indispensables pour envisager une étude analytique précise.

Si l'on définit par hypothèse que la *suite des retards suit une certaine distribution de probabilités*, il est possible d'en déduire des conditions de stabilité ou même d'évaluer des critères de qualité (cf. [Cervin *et al.*, 2003a]). En effet, des propriétés probabilistes peuvent être étudiées dans un cadre stationnaire, elles permettent donc d'obtenir des résultats équivalents à la prise en compte du cas constant. Si l'on exclut le cas de certaines distributions aux propriétés de calcul remarquables (comme les processus de poissons), les calculs probabilistes sont très complexes à finaliser, même sous forme numérique. Cependant, l'inconvénient majeur des distributions probabilistes est qu'elles constituent une modélisation approchée de la suite et non une propriété vérifiable de cette dernière. Par contre, par construction, elles vérifient des propriétés statistiques (bornes et moyenne) et permettent de générer aisément des suites ayant des propriétés attendues (en vue, par exemple, de procéder à des simulations).

La modélisation des retards sous forme de perturbations est une solution pour obtenir plus simplement des résultats ; on peut en effet quantifier les *pires perturbations possibles correspondant à ces retards*, il suffit alors de chercher à caractériser la capacité du système à rejeter les perturbations (cf. section 3.4). il est intéressant dans ce cas de séparer la composante fixe des retards de la partie variable (cf. section 3.1.2). On garde ainsi la propriété mathématique essentielle des retards constants, tout en simplifiant l'étude de la partie variable en la considérant comme une perturbation.

5.2.1.3 Cas particulier de la gigue sur les dates d'acquisition

Les événements d'acquisition de données sur le procédé physique peuvent être décalés dans le temps par rapport à leur date prévue. Ce décalage peut relever de 3 formes :

- une variation de la période réelle d'échantillonnage, dont l'évolution est connue (cf. section 3.2.1),
- un retard constant sur les dates prévues d'acquisition,
- une gigue, c'est-à-dire un retard variable sur les dates prévues d'acquisition (cf. section 3.1.3).

Notons, qu'intuitivement, l'influence de ces fautes temporelles sur les propriétés du système dépend de l'écart entre la période réelle T' et la période de référence T. Deux phénomènes peuvent influer sur le fonctionnement du système régulé :

1. l'influence de T sur la loi de commande lorsqu'elle est calculée en fonction de T ; par exemple, une loi en T^2 est sensible à une erreur sur T et on risque, alors, de perdre tout l'intérêt de la loi de commande et de rendre le système instable.
2. même si la loi de commande reste correcte (cas d'un contrôleur proportionnel), si la période T' est trop grande, d'après le théorème de Shannon, on risque de perdre la capacité de reconstruire l'état du système et donc la stabilité de celui-ci (cf. section 1.2.1).

L'étude exacte de l'influence d'une gigue sur le fonctionnement du système régulé est du même ordre de difficulté que l'étude du fonctionnement en présence de retards variables. Il faut être capable de quantifier l'incohérence qu'elle génére à l'intérieur des algorithmes de commande et les études à mener sont donc très dépendantes des définitions de ces lois de commande. Dans le cas d'un correcteur proportionnel, l'influence est minime. Par contre dans le cas d'un calcul reposant sur le temps comme la dérivée, l'influence de la gigue peut être catastrophique. Nous illustrons comment évaluer cette influence dans l'exemple élémentaire présenté ci-dessous.

Considérons un système embarqué dans un équipement mobile et supposons qu'une des lois de commande ait, à chaque activation, à déterminer la vitesse de cet équipement. Le système dispose pour cela d'un capteur de position qui fournit la position X. Dans la spécification de la loi de commande, les acquisitions sont considérées périodiques de période T ; $X_{k-1},\ X_k$, représentent donc $X(t_{k-1}),\ X(t_k)$ où $t_k - t_{k-1} = T$. La vitesse v calculée dans cette loi de commande, par exemple par une approximation du premier ordre, est :

$$v_k = \frac{X_{k+1} - X_k}{T}$$

Il est facile de voir que si l'acquisition de la valeur de la position est mal échantillonnée, les deux valeurs $X_{k-1},\ X_k$ ne correspondent plus à $X(t_{k-1}),\ X(t_k)$. La valeur calculée de la vitesse est donc erronnée. Supposons que la gigue sur l'acquisition de la valeur de position soit bornée par une propriété de fraîcheur du type :

$$\forall k\ J_k \in [J_{min}, J_{min} + F] \tag{5.1}$$

et que la vitesse de l'équipement soit constante sur une période et égale à v. La formule de calcul de la vitesse utilisée dans la loi de commande est donc :

$$v = \frac{X_{k+1} - X_k}{T}\ (\forall k)$$

La vitesse mesurée dépend par contre de la gigue :

$$v^k_{mesure} = \frac{x(t_{k+1} + J_{k+1}) - x(t_k + J_k)}{T} = v \cdot (1 + \frac{J_{k+1} - J_k}{T}) \tag{5.2}$$

On peut déduire des équations (5.1) et (5.2) la borne sur l'erreur relative qui est commise sur la vitesse en présence d'une gigue sur l'acquisition :

$$\frac{|v_{reelle} - v_{mesuree}|}{v_{mesuree}} < \frac{F}{T}$$

5.2.2 Conclusions sur l'évaluation analytique de l'influence des fautes temporelles sur les propriétés d'un système

Nous avons dans le paragraphe précédent montré qu'il était possible dans certains cas d'avoir une expression analysable mathématiquement donnant une caractéristique de qualité d'un système en fonction des caractéristiques temporelles de l'implémentation. L'étude analytique est généralement difficellment réalisable.

Cependant, même dans le cas où l'on sait garantir mathématiquement la stabilité, on peut néanmoins être en présence d'une dégradation importante du fonctionnement. Cette dégradation ne peut pas toujours, elle, être évaluée de manière analytique. Par exemple, on peut théoriquement calculer une trajectoire de poursuite (comportement du système en présence d'une consigne évoluant dans le temps) en l'absence de perturbation et ainsi en déduire des caractéristiques (bornes, intervalles) définissant la qualité. Malheureusement, la trajectoire ne peut généralement pas être exprimée sous une forme close. Cette difficulté augmente dés qu'on prend en compte les perturbations en raison de leur caractère aléatoire et difficilement modélisable.

Deux autres faits rendent l'étude analytique des systèmes difficile à mettre en oeuvre :

- l'existence d'une boucle de retour,
- la complexité de certains algorithmes de contrôle.

La boucle de retour joue un rôle fondamental dans le bon fonctionnement du système. Cependant, sa présence peut amplifier considérablement l'influence d'un phénomène a priori minime. Il suffit pour s'en convaincre de considérer le cas le plus connu : l'effet LARSEN où le fait de renvoyer la sortie d'un système audio sur son entrée

provoque très rapidemement la saturation du système (bruit insupportable dans ce cas). Ainsi, l'existence d'une boucle de retour peut faire apparaître un phénomène déstabilisant majeur dans le cas où les signaux utilisés ont des caractéristiques mathématiques spécifiques (périodicité et retard constant en particulier). Enfin, certains algorithmes de contrôle peuvent s'avérer difficilement analysables. C'est le cas des commandes prédictives et adaptatives pour lesquelles la capacité d'adaptation des algorithmes est équivalente à la présence d'une boucle de retour à l'intérieur de la loi de commande. Sous des conditions particulières, l'algorithme peut alors devenir instable. Pour prendre en compte l'existence de tels phénomènes, une analyse fine des algorithmes et des signaux utilisés doit être faite.

De plus, même si certains résultats analytiques sont disponibles, ils permettent donc essentiellement d'obtenir des critères de stabilité. Ces critères correspondent à des conditions *suffisantes* de stabilité mais ne permettent pas de conclure au mauvais fonctionnement du système en cas de non-respect des critères.

Pour toutes ces raisons, une approche par simulation se révèle donc un complément nécessaire afin de mieux comprendre les différentes formes de cette influence. Elle permet en particulier l'étude du fonctionnement du système régulé vis à vis de différentes métriques de qualité sans nécessiter l'adaptation des modèles initiaux traditionnellement développés par un automaticien.

5.3 Evaluation de l'influence d'une implantation par une approche simulation

Comme annoncé dans l'introduction, nous avons développé une technique reposant sur le langage et l'outil Matlab/Simulink de la société MathWorks pour construire des modèles de systèmes intégrant les caractéristiques de l'architecture opérationnelle implantant le système réglant.

Matlab/Simulink est un outil offrant un éditeur graphique de modèles de systèmes continus et échantillonnés ainsi qu'une machine d'exécution permettant la simulation de ces modèles. Ce logiciel offre de nombreuses fonctionnalités :

- une approche reposant sur les signaux,
- un grand nombre de modèles prédéfinis, les "toolboxes",
- un vrai langage de programmation qui permet de définir d'autres composants,
- la possibilité d'encapsuler des modèles les uns dans les autres (hiérarchie de composants).

La machine d'exécution Simulink permet de simuler l'évolution des systèmes continus représentés sous forme d'équations différentielles ou sous une forme équivalente comme la représentation de Laplace (transformée en p, cf annexe A). Il permet aussi de simuler les systèmes échantillonnés représentés sous forme de transformées en z.

Comment à l'aide de ces techniques usuelles représenter l'implémentation du système réglant ? Ainsi que nous l'avons vu, l'effet d'une architecture informatique support peut être interprété de deux manières : soit comme des perturbations, soit comme des retards dans les modèles. Si la définition des perturbations dans un modèle Matlab/Simulink ne pose pas de problèmes nouveaux, l'intégration de phénomènes de retard peut se révéler plus problématique. Dans le paragraphe suivant, nous donnons les principes généraux d'une modélisation Matlab/Simulink intégrant les performances d'architectures informatique support et introduisons deux niveaux d'abstraction pour les prendre en compte. Puis dans les deux paragraphes suivants, nous fournirons respectivement des exemples de modélisation aux deux niveaux d'abstraction proposés.

5.3.1 Principes de modélisation

Considèrons des systèmes échantillonnés dont le contrôle est fait de manière périodique (ce qui est le cadre de notre étude). Il serait alors logique d'utiliser la représentation en z des systèmes et d'y injecter les retards. Cette approche est cependant rarement possible car la représentation en z suppose que les différents systèmes soient parfaitement périodiques. Cela signifie que l'ensemble des échanges de signaux est synchronisé sur la période d'utilisation et que l'évolution des systèmes n'est considéré qu'aux instants périodiques. Ainsi l'intégration de retard dans un modèle en z n'est possible que si les retards sont des multiples de la période d'échantillonnage. Or, les retards sont généralement plutôt inférieurs aux périodes d'échantillonnage, il est donc difficile de se contenter d'une définition en z de la commande.

En fait, ainsi que nous l'avons vu précédemment (cf. chapitre 3), il existe deux façons de modéliser le système réglé de manière à intégrer ces retards :

- Intégrer dans le modèle une fréquence d'observation multiple de la fréquence réelle d'échantillonnage et faire apparaître les retards comme des multiples de la période d'observation. Il faut alors adapter les traitements de manière à ce qu'ils correspondent bien aux traitements réels définis sur le système et conformes à la fréquence d'échantillonnage spécifiée.

- Utiliser un modèle continu des systèmes physiques et modéliser les lois de commandes sous forme de transformée en q. C'est-à-dire sous forme d'un opérateur qui travaille sur les différents échantillons reçus sans avoir besoin de considérer une hypothèse de périodicité sur la production et l'utilisation des échantillons.

Ces deux techniques ont déjà été proposées (cf. thèse de Nilsson à Lünd [Nilsson *et al.*, 1998] pour une étude comparative détaillée) : la première permet plus facilement une étude analytique en considérant la notion de discrétisation, la seconde est plus intéressante car elle fait apparaître de manière bien distincte la partie physique du système et la partie de contrôle et permet donc une plus grande liberté de modélisation des retards.

Plusieurs simulateurs ont été développés [Henriksson *et al.*, 2002; Cervin *et al.*, 2003b; Nilsson, 1996; Nilsson *et al.*, 1998; Elkhoury and Törngren,] pour simuler l'influence de l'implémentation (tout d'abord au cours des travaux Törngren puis les simulateurs plus aboutis RTkernel, True time & Jitterbug, ...). RTkernel ([Wittenmark *et al.*, 1998]), qui était le seul disponible au cours de l'étude, ne permet que la prise en compte des retards liés à l'existence d'un ordonnanceur ; il ne permet pas de manipuler explicitement la notion d'échantillon. Un nouveau simulateur, "True time", offre lui plus de possibilités (mise en série d'ordonnanceurs) mais cache la loi de commande à l'intérieur des ordonnanceurs ce qui ne donne pas un modèle très lisible, ni très flexible. C'est pourquoi, nous avons développé notre propre technique de modélisation qui permet de faire apparaître le modèle usuel de l'automaticien (c'est-à-dire système physique + loi de contrôle) mais aussi de rendre compte de manière naturelle les retards engendrés par l'implémentation. Cette technique repose donc sur une approche modulaire intégrant des composants propres à l'automatique et d'autres propres à l'implémentation. Cette décomposition permet de mettre en évidence des choix qui peuvent être fait à différents niveaux de l'architecture à l'aide du même formalisme. Les caractéristiques principales en sont :

- modélisation des systèmes physiques sous forme de modèles continus ,

- définition des lois de commande sous forme récurente (notation en q ou en k) ,

- manipulation explicite des échantillons avec leurs dates d'occurences de production et d'utilisation,

- possibilité de modifier simplement les modèles de régulation développés sous SIMULINK pour analyser l'influence des choix d'implémentation.

Une grande partie des fautes temporelles dues à l'implantation peuvent être modélisées en utilisant les blocs prédéfinis de Matlab/Simulink. Il s'agit en particulier de certains retards et des gigues que l'on peut représenter par l'utilisation conjointe des transformée de Fourrier et de Laplace (cf. annexe A).

Cependant, des fautes temporelles touchant certaines propriétés de fonctionnement (comme la périodicité des algorithmes de traitement) ne peuvent être représentées ainsi. Il en est de même des fautes qui pourraient être liées à la distribution des algorithmes de traitement Ainsi, un grand nombre de phénomènes ne peuvent pas être modélisables en utilisant les composants prédéfinis de Matlab/Simulink. Pour résoudre ce problème, nous avons défini des blocs qui permettent l'activation des traitements sur évènement (voir schéma 5.1). Un bloc posséde une entrée pour les données et une pour les évènements activant ce bloc. De la même manière, le bloc peut produire des données en sortie et y associer des évènements. Les différents blocs évènementiels proposés sont :

- Le bloqueur : c'est le bloc fondamental qui discrétise un signal continu suivant les évènements en entrée. Sur l'occurrence d'un événement d'entrée, la valeur du flux d'entrée est collectée. Elle est retransmise en sortie associée à un événement qui pourra servir à activer les blocs suivants.

- Le bloc fonction de transfert en q : il modélise un algorithme de traitement défini sur les échantillons. La fonction de transfert est la même que celle qui est définie par l'automaticien en z. La notation en q est juste utilisée pour rappeler que la propriété de périodicité du traitement n'est plus conservée et que la fonction de transfert est maintenant définie sur les échantillons quelles que soient leurs dates de production.

- Les blocs retard constant et retard variable : comme leur nom l'indique, ils modélisent des retards spécifiant que la donnée en entrée est transmise avec un certain retard ; ils produisent ainsi des événements de sortie décalés de la durée du retard par rapport aux événements d'entrée. Ce retard peut être géré de manière aléatoire ou en utilisant des données externes pouvant être produites par exemple par un simulateur d'architecture opérationnelle (cf. chapitre 4) utilisé de manière externe.

- Le bloc perte : il modélise l'occurrence de pertes de données entre deux blocs suivant un certain motif qui se reproduit périodiquement ou qui est généré aléatoirement de façon à satisfaire une contrainte sur la moyenne. Ce bloc est utile pour comprendre l'influence des pertes sur l'évolution du système (cf. le chapitre 6 pour un développement plus détaillés sur les notions de pertes) .

Chacun de ces blocs est paramétrable. Nous proposons, sur la figure 5.1, une représentation générique d'un bloc conformément à la notation graphique Matlab. Notons que pour chaque bloc nous identifions les données d'entrée et de sortie ainsi que les événements d'activation du bloc en entrée et de production de la donnée de sortie. Cette notation permet de représenter à la fois la transformation des informations à l'intérieur du système et les règles de contrôle sur ces flux, qu'elles proviennent de la loi de commande elle-même (par exemple, traitement périodique) ou de l'accès aux ressources de l'architecture informatique support (accès à la ressource processeur, accès au réseau, ...) ainsi que nous l'avons proposé au chapitre 4. Enfin, par cette méthode, il est possible de représenter l'influence de l'architecture informatique support du système réglant à plusieurs niveaux d'abstraction : simple retard précalculé ou retard calculé en cours de simulation, comme dans True Time. Par contre, elle offre l'avantage de séparer clairement le problème d'automatique du problème de l'implantation du système.

5.3.2 Modélisation à l'aide de retards précalculés - quelques exemples

Le schéma (5.2) représente un exemple élémentaire de modèle de régulation ; le système physique est modélisé en utilisant les blocs habituels de Matlab / Simulink (sous forme de transformé en p), par contre le système de

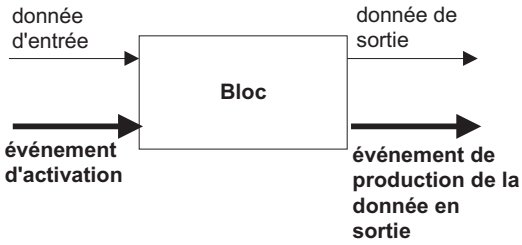


Figure 5.1: Modèle générique d'un bloc

contrôle est modélisé à partir des blocs événementiels proposés. On voit que le bloc de calcul de la commande ($H(q)$) à appliquer au système est activé sur réception d'un événement produit par le bloqueur, lui-même étant activé périodiquement (échantillonnage périodique du système physique).

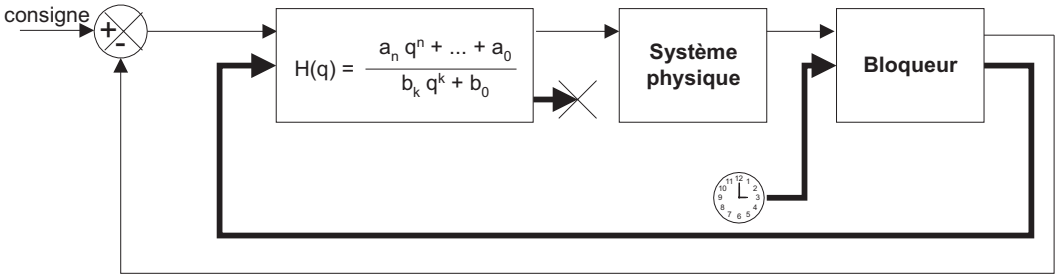


Figure 5.2: Exemple élémentaire d'une régulation

Le schéma (5.3) montre comment modéliser un retard variable au niveau de la chaîne de contrôle. Dans le cas où le traitement n'est effectué que dans un seul bloc (c'est-à-dire, qu'implicitement, on ne considère pas de distribution du traitement sous la forme de blocs activés indépendamment les uns des autres), on peut considérer l'ensemble des retards au niveau de l'acquisition des données.

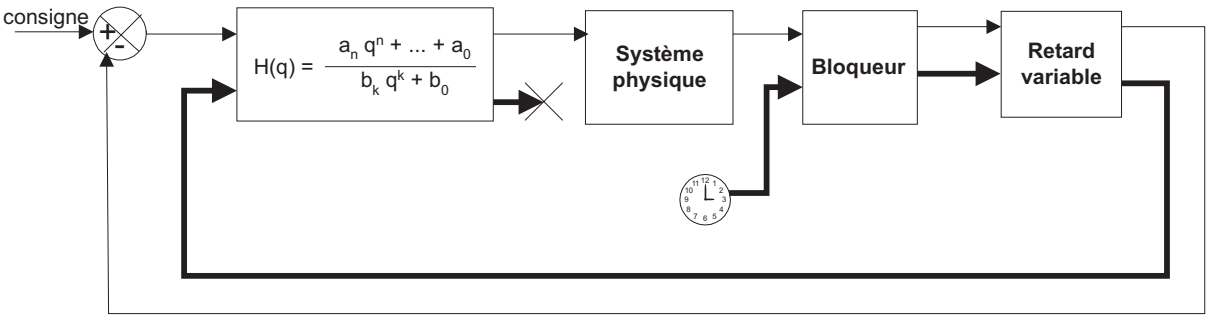


Figure 5.3: Modélisation d'un retard variable dans la boucle de régulation

Le schéma (5.4) modélise un phénomène de gigue au niveau de l'acquisition. Cette gigue est représentée en utilisant un retard variable au niveau de l'horloge qui active le bloqueur.

Enfin le schéma 5.5 présente un exemple où les activations de l'acquisition et du traitement ne sont pas synchronisées sur la même horloge (d'où un écrasement possible de valeurs). En effet, il est courant de procéder à une acquisition des données à une fréquence plus élévée que celle d'activation de la loi de commande et de

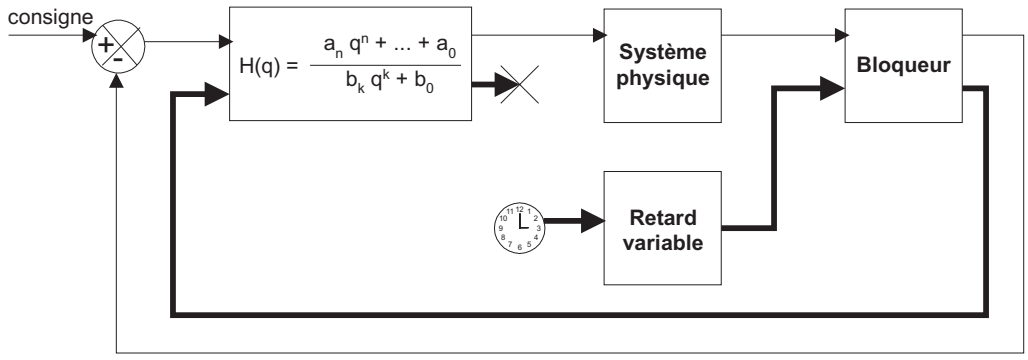


Figure 5.4: Modélisation d'une gigue sur l'aquisition des données

calculer ainsi la commande avec la donnée la plus récente.

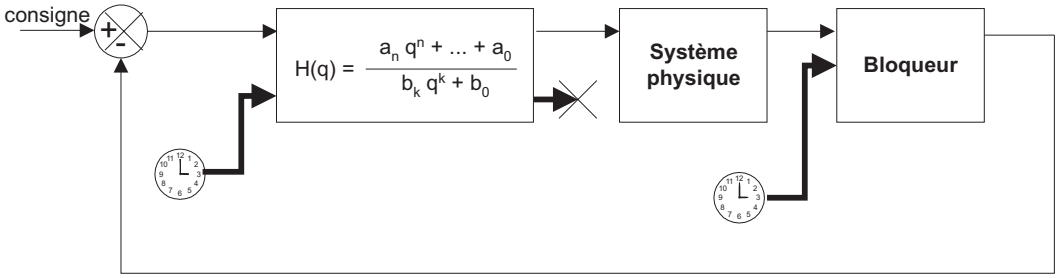


Figure 5.5: Modélisation d'activations indépendantes du calcul de la loi de commande et de l'acquisition des données

On peut remarquer que dans les exemples illustrés aux figures 5.2, 5.3, 5.4, le mode de coopération implicite entre Bloqueur et loi de commande, ou entre Bloqueur et Retard, ou entre Retard et loi de commande est de type "pull entre blocs". Pour ne pas alourdir la représentation graphique Matlab/Simulink, nous ne les représentons uniquement par des liens entre événements de sortie d'un bloc et événement d'entrée d'un autre bloc. Dans le cas de l'exemple de la figure 5.5, le mode de coopération entre Bloqueur et loi de commande est de type "pull entre ports". De la même manière, il est implicitement illustré par le comparateur du modèle Simulink.

5.3.3 Exemple de modèles explicites des accès aux ressources

Dans le cas où l'on désire analyser de façon plus précise l'impact d'une politique d'accès aux ressources ainsi que la configuration de tâches et/ou de messages au niveau de l'implantation, il est nécessaire d'introduire des blocs supplémentaires. Nous proposons donc :

- Bloc Ordonnanceur CPU : il modélise un ordonnanceur des traitements sur une ressource processeur. Ce bloc est connecté aux blocs classiques précédemment cités par des événements et des données. Il reçoit un événement de demande de traitement et en donnée d'entrée, le temps supposé d'exécution connu (par exemple, le pire temps d'exécution - WCET -) ainsi que d'autres caractéristiques éventuelles (priorité du traitement, ...). Il calcule suivant une politique d'ordonnancement les dates de fin des traitements qui généreront des événements qui seront utilisés pour activer d'autres blocs. Les politiques EDF et FIFO ont été implantées pour ce bloc.

- Bloc Ordonnanceur Réseau : il représente la gestion des accès à une ressource réseau. De la même manière, que pour le bloc Ordonnanceur CPU, il reçoit un événement de demande de transmission et en donnée d'entrée, le temps de transmission supposé connu (ou la taille utile du message) et des caractéristiques éventuelles des messages. Il calcule les dates de fin de transmission qui généreront des événements d'activation d'autres blocs. Un protocole de communication à priorités a été implanté pour régler les conflits d'accès au réseau.

Les schémas (5.6) et (5.7) montrent un exemple typique d'utilisation d'un ordonnanceur sur la ressource réseau et sur la ressource CPU. Pour plus de clarté, nous présentons le modèle en deux parties qui formeraient sous Matlab / Simulink un modèle unique. La première partie introduit les blocs traditionnels du modèle de commande tandis que la deuxième partie montre l'intégration des blocs de gestion des ressources réseaux et processeur. Dans l'exemple, nous supposons que la transmission des données entre le Bloqueur d'acquisition et la loi de commande, ainsi qu'entre la loi de commande et le processus physique se font via un même réseau de communication. Par ailleurs, la loi de commande s'exécute sur un processeur partagé avec d'autres applications non représentées ici. Les liens sont indiqués par les motifs Neti_e, Neti_s, CPUi_e et CPUi_s. Ceux-ci contiennent l'événement et, si nécessaire, la donnée associée ainsi que décrit précédemment.

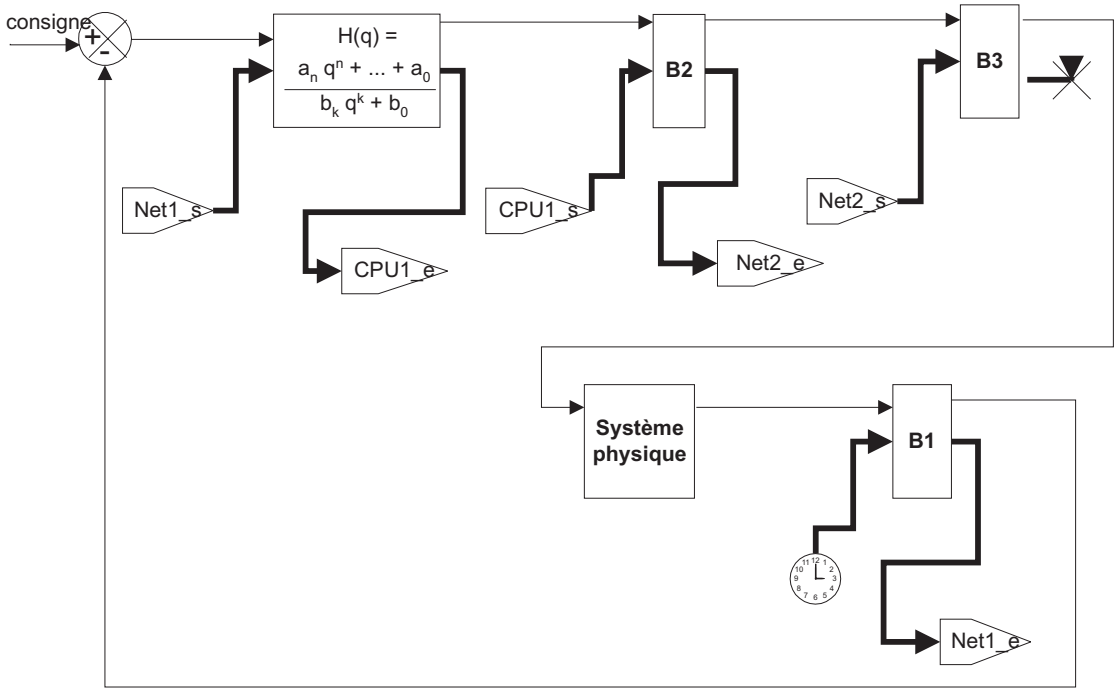


Figure 5.6: Modélisation de l'architecture de commande

5.4 Exemple de caractérisation de l'influence par simulation

L'exemple étudié est tiré d'un des ouvrages de référence des systèmes échantillonnés écrit par K. Ogata ([Ogata, 1987]. L'exemple concerne la régulation en position d'un moteur à courant continu, pouvant par exemple contrôler le déplacement d'un mobile sur une crémaillère. Le système est composé:

- d'un moteur qui constitue le système réglé (ou système physique),
- des algorithmes de commande.

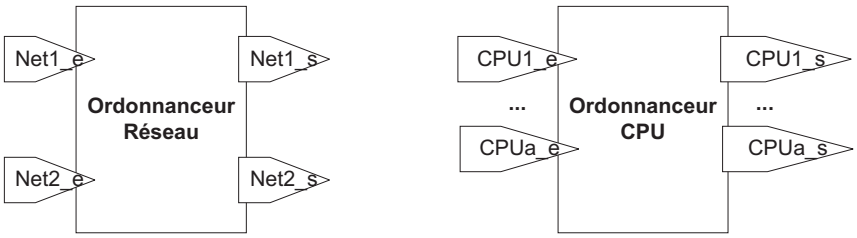


Figure 5.7: Modélisation des gestionnaires d'accès aux ressources

Le moteur est modélisé par sa fonction de transfert qui lie la tension appliquée à la position du moteur:

$$\frac{1}{p \cdot (p+2)}$$

Un certain nombre de perturbations écarte le moteur de son fonctionnement idéal, nous les avons modélisées sous forme de bruits blancs. La commande est un correcteur en boucle direct (cf. schéma 5.8 et annexe B). La période choisie pour le contrôle est de $0.2s$. Ce correcteur a été calculé par placement de pôle sur le système continu équivalent puis a été discrétisé (voir le livre de Ogata p335 [Ogata, 1987])

Le controlleur est défini par sa fonction de transfert en z :

$$\frac{13.57(z-0.6703)}{(z-0.2644)}$$

ce qui correspond à un algorithme de traitement de la forme (cf 4.2.1):

$$c(n) = c(n-1) * 0.2644 + 13.57 * (cn(n) - p(n)) - 9.096 * (cn(n-1) - p(n-1))$$

où $c(.)$ est la commande, $cn(.)$ est la consigne et $p(.)$ est la position.

L'existence de fautes temporelles vis à vis du cas idéal peut être modélisée par :

- une gigue ou un retard sur les signaux produits et utilisés,
- un retard sur la commande,
- une autre fréquence d'utilisation de la loi de commande que celle définie théoriquement.

De manière à simuler les conséquences sur le fonctionnement du système régulé, nous avons développé un modèle de simulation du moteur qui permet d'injecter différents modèles de fautes.

5.4.1 Métriques considérées

Le but de la simulation est d'étudier l'influence des caractéristiques de l'implémentation sur le bon fonctionnement des applications de contrôle-commande. Le chapitre 2 présente les différentes métriques d'usage courant pour décrire la qualité de la régulation.

Nous avons choisi pour décrire le comportement du système en asservissement de considérer le temps de réponse et le dépassement. Ces critères sont dans le cas des système SISO les plus importants. L'automaticien fixe en général une borne admissible sur ces deux critères[2]. La meilleure qualité est obtenu pour un temps de réponse et un dépassement les plus bas possibles. Généralement, on fixe le dépassement admissible (par exemple à 20%) et l'on cherche à minimiser le temps de réponse.

[2]La définition mathématique de ces critères est donnée dans le chapitre 2.

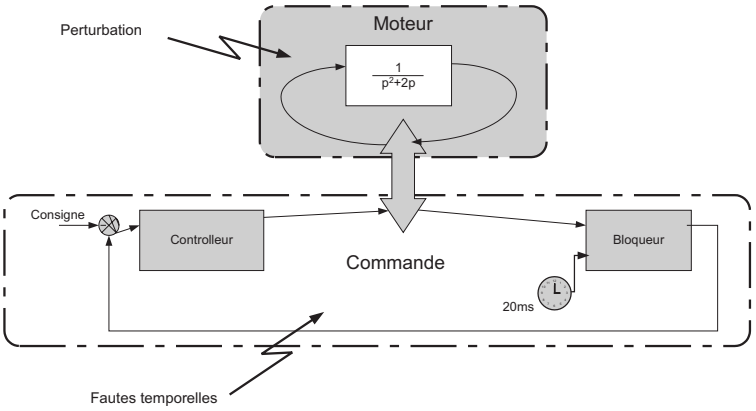


Figure 5.8: Représentation schématique de la régulation du moteur

Le critère intégral (IAE) est utilisé pour décrire la capacité de régulation du système en absence de changement de consigne. Il permet simplement de quantifier la qualité obtenue. Dans le cas d'un système parfait, en absence de changement de consigne, on devrait obtenir une valeur nulle de ce critère. Plus le fonctionnement constant est perturbé plus l'IAE obtenu sera élevé. Si l'on ramène l'IAE à la durée de simulation considérée, on obtient une mesure de l'écart moyen vis à vis du régime constant souhaitée.

5.4.2 Influence de l'implémentation

L'implémentation d'un système de contrôle commande fait apparaître différentes formes de fautes temporelles (cf. chapitre 3). Ces fautes sont dérivées de la modélisation du support (cf. chapitre 4). L'important est de connaître, les dates associées aux événements critiques (acquisition et commande sur le système physique). Les caractéristiques fondamentales de l'architecture support sont:

- la fréquence d'utilisation,
- le temps de traitement.

Cependant, certaines caractéristiques annexes peuvent engendrer l'existence d'une gigue sur les acquisition ou d'une variabilité des temps de traitement et donc influer sur le fonctionnement du système. L'existence de mécanismes de coopération, en particulier, peut avoir une influence non négligeable.

5.4.2.1 Caractéristiques essentielles de l'architecture support

Un algorithme de contrôle commande est défini pour fonctionner à une fréquence d'utilisation précise. Le non respect de cette fréquence peut engendrer une perte de qualité importante (cf. section 5.2.1.3). Pour illustrer ce phénomène, nous présentons l'étude de l'influence du changement sur le contrôle de vitesse du moteur (période d'utilisation prévue 0.2s).

Le schéma 5.9 présente les résultats obtenus pour un niveau de perturbation donné relativement faible et peu influant.Le dépassement évolue presque linéairement avec la période car le système est laissé sans contrôle durant un temps plus grand. Dans ce cas, la période de 0.2s n'a aucune propriété particulière et il semble plus intéressant d'utiliser l'algorithme de contrôle à une fréquence la plus élevée possible. L'évolution du temps de réponse est par contre plus complexe, pour une période plus courte (fréquence plus élevée) que 0.2s la qualité se dégrade de manière linéaire. Pour une période plus grande elle s'améliore très légérement avant de se dégrader considérablement. La loi de commande est en fait adaptée à une fréquence de 0.2s. Si cette fréquence est trop éloignée, la perte de qualité est très importante. Le saut constaté s'explique par le besoin d'une oscillation

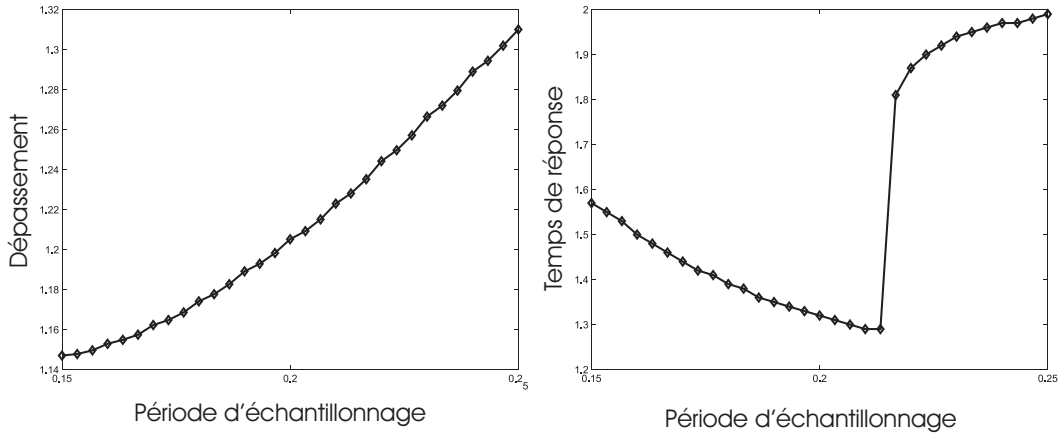


Figure 5.9: Evolution du temps de réponse et du dépassement en fonction de la période d'échantillonnage

supplémentaire pour entrer dans le gabarit (cf. remarque pour le temps de traitement). Un faible écart vis à vis de la fréquence est peu préjudicable sauf si l'on dépasse le niveau du saut ou dans ce cas la perte est très importante. La connaissance de ces caractéristiques permet de procéder à la meilleur implantation de la périodicité du traitement dans le cas où la valeur attendue ne peut être fournit et aussi de quantifier la perte qui peut être obtenue.

Le temps de traitement associé à la commande est la deuxième propriété la plus importante de l'architecture support. Dans le cas d'une implémentation élémentaire (monotâche, électronique programmable ...), ce temps peut être considéré comme constant. Il se modélise sous forme *d'un retard* au sens de l'automatique, le schéma 5.11 représente l'évolution des critères de qualité associés au changement de consigne en fonction de ce temps de traitement. La première courbe à gauche représente le dépassement, on remarque une augmentation linéaire du dépassement jusqu'à un certain retard puis un pic. Ce pic correspond à la perte de stabilité du système (sous forme de sinusoïde d'amplitude augmentant exponentiellement et aménant à la saturation ou la destruction du système) et correspond au retard critique du système (cf. section 5.2.1.2). La courbe du milieu représente le temps de réponse moyen. Il se dégrade avec l'augmentation du temps de traitement (comme le dépassement). Cette dégradation ne se fait plus de manière linéaire mais sous forme d'escalier. Ce résultat s'explique par la nature oscillante du système, le temps de réponse est lié à la dernière oscillation hors gabarit (cf. figure 5.10). Il y a donc un changement important du temps de réponse dès que la stabilisation dans le gabarit nécessite une oscillation de plus. La connaissance de ce type de caractéristique peut se révéler très pertinente. Sur cet exemple, la différence de qualité obtenue est faible pour un passage du temps de traitement de 0.05s à 0.1s. La dernière courbe représente l'évolution de l'IAE (cf. section 5.4.1) en absence de changement de consigne et représente donc la capacité du système régulé à rejeter les perturbations. On remarque ici que la qualité semble évoluer suivant trois pentes différentes. La difficulté de rejeter les perturbations s'accentue donc très fortement avec l'augmentation du temps traitement.

5.4.2.2 Influence de la variabilité des temps de traitements

Dans le cas des architectures informatisées, les temps de traitement sont rarement constants. En effet, le temps de calcul effectif peut être variable (dépendant par exemple des valeurs des entrées) mais surtout, l'accès aux ressources peut se faire en concurence avec d'autres activités. La variabilité des temps de traitements peut donc être très importante, de quelques pourcents à un facteur 10 ou 20 dans le cas de l'ordonnancement d'activités périodiques sur un CPU. La question se pose alors de comprendre quelle va être l'influence de cette variabilité. Nous avons présenté (cf. section 5.2.1.2) quelques résultats analytiques disponibles, ils sont cependant peu nombreux et permettent essentiellement de se prononcer sur la stabilité du système. Nous utilisons donc les

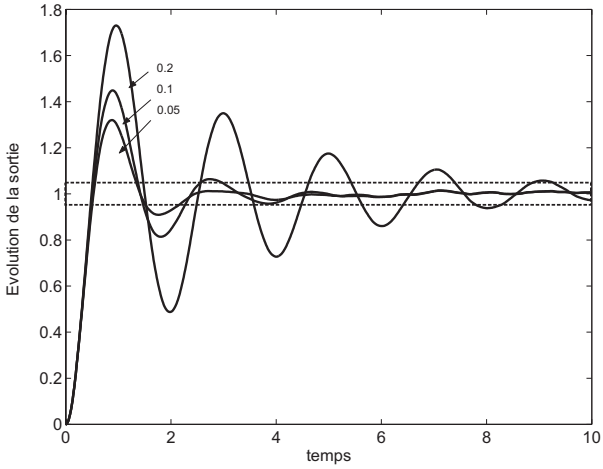


Figure 5.10: Evolution de la sortie du système régulé pour différents temps de traitement

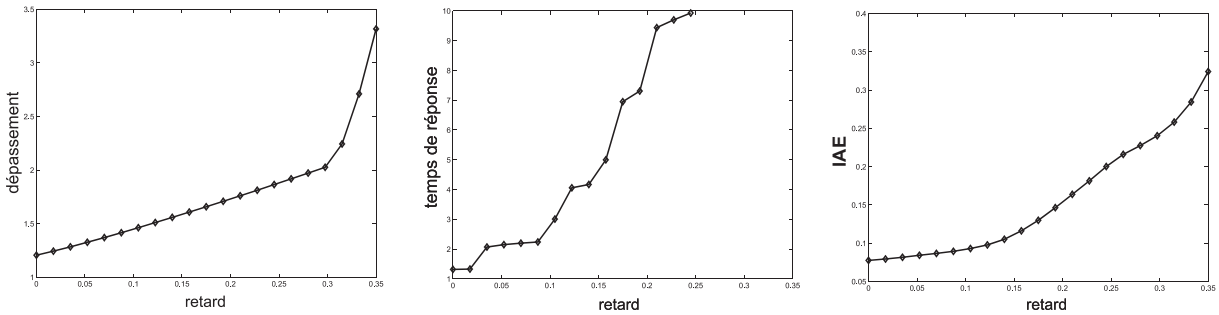


Figure 5.11: Evolution du temps de réponse et du dépassement en fonction d'un retard constant

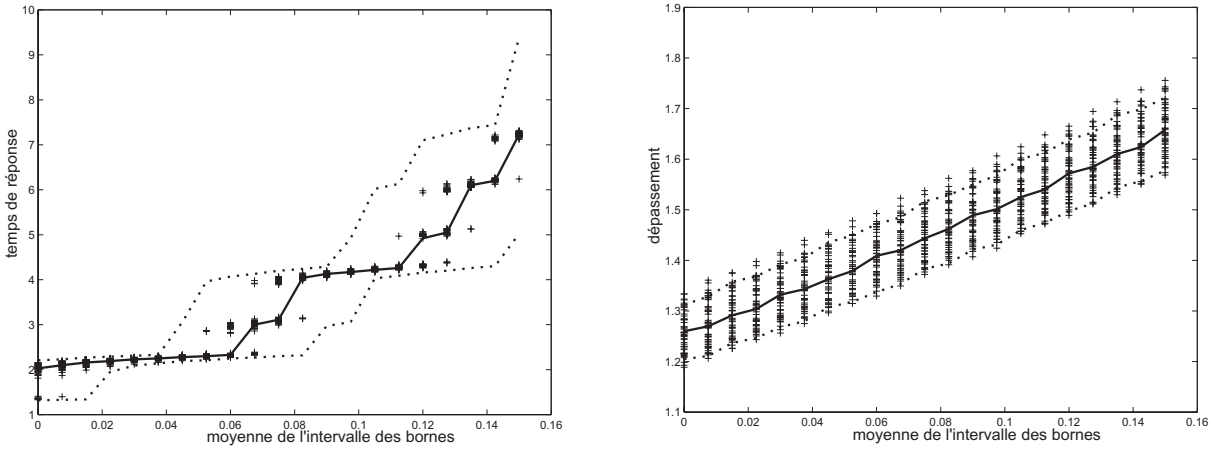


Figure 5.12: Qualité de fonctionnement en présence de temps de traitement variable: jeux d'essais dans un intervalle borné de taille constante pour différentes moyennes

techniques de simulation de manière à comprendre l'influence de la variabilité des temps de traitement. Nous avons choisi deux exemples élémentaires pour montrer les caractéristiques essentielless de cette variabilité.

Exemple 1 Le schéma 5.12 représente l'évolution de la qualité de la régulation du moteur (temps de réponse et dépassement) en fonction du temps de traitement. Le temps de traitement est tiré aléatoirement (suivant une loi uniforme) entre tmoy +/- 0.025 s, tel que le temps moyen varie de 0.05s à 0.195s. Les courbes en pointillé représentent la qualité obtenue pour un temps de traitement constant maximal et minimal dans chaque intervalle considéré. Une centaine de scénari aléatoires (qui correspondent chaqu'un à 50 cycles de 0.2s) ont été testés pour chaque intervalle, le résultat d'une simulation est indiquée par une croix qui représente la qualité obtenue pour un changement de consigne.

Les résultats obtenus sont intéressants, le temps de réponse est compris dans l'enveloppe donnée par un temps de traitement constant minimal et maximal. On remarque que les temps de réponse forment des amas autour de valeurs particulières. Ces valeurs pour chaque intervalle correspondent au nombre d'oscillations avant stabilisation (voir remarque précédente pour le cas constant). L'évolution du dépassement fait apparaître que la qualité dans le cas variable peut être meilleure ou moins bonne que ce que l'on pense intuitivement être le meilleur cas et le pire cas (représentés en pointillé). Ce phénomène est encore plus visible dans le deuxième exemple.

Exemple 2 Dans cet exemple, la moyenne est toujours identique mais l'on considère des intervalles (qui bornent les temps de traitement) de plus en plus grand (de 0.004s à 0.2s). Pour chaque intervalle, on crée une centaine de scénari (les temps de traitements sont tirés aléatoirement dans l'intervalle). Pour chaque scénario créé, la qualité obtenue est représentée par une croix. Les courbes en pointillé représentent la qualité obtenue pour un temps de traitement constant maximal et minimal dans chaque intervalle considéré. Le phénomène présenté sur le dépassement dans l'exemple 1 est toujours présent mais s'amplifie considérablement avec l'augmentation de l'intervalle. Le dépassement peut donc être beaucoup plus important que celui donné par le pire temps constant (jusqu'à 30% de plus sur cet exemple). Le temps de réponse évolue dans le cas variable sous forme d'amas comme précédemment. Par contre, quand la taille de l'intervalle augmente, on remarque que le temps de réponse reste relativement faible (inférieur au cas moyen obtenu avec un retard constant correspondant à la moyenne de l'intervalle cf. 5.13). Cela signifie que l'existence sporadique de temps de traitement très grand n'a pas toujours d'influence sur le temps de réponse et que les propriétés les plus importantes sont plutôt la moyenne et la borne inférieur des temps de traitement.

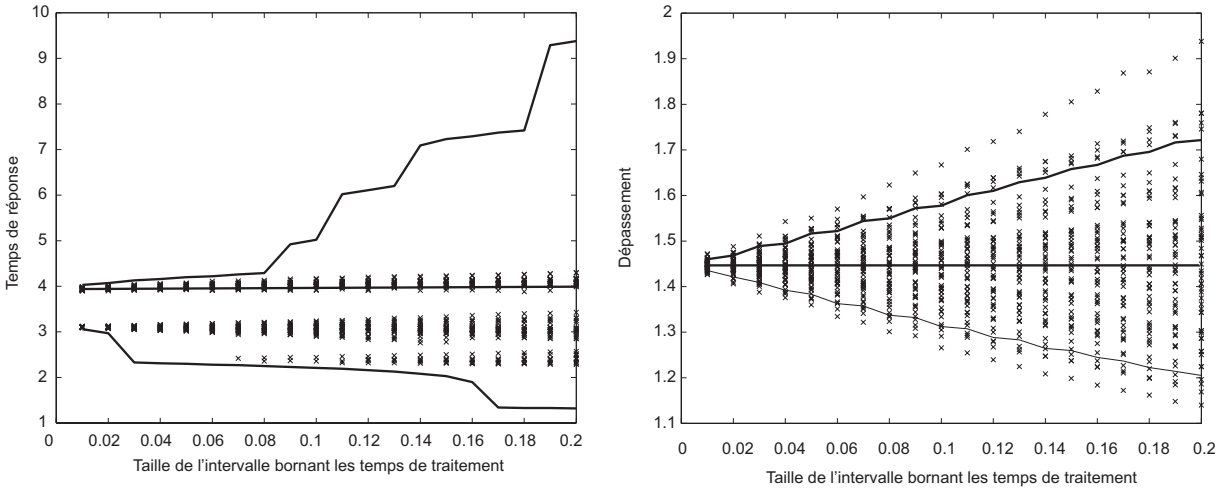


Figure 5.13: Qualité de fonctionnement en présence de temps de traitement variable: jeux d'essais pour une même moyenne dans des intervalles bornés de taille différente

5.4.2.3 Influence des choix des politiques d'ordonnancement

Dans le cas d'utilisation du partage des ressources réseaux et CPU entre plusieurs activités, le choix des politiques d'ordonnancement peut considérablement modifier la qualité des différentes activités. Pour illustrer le phénomène, on considère 3 régulations de 3 moteurs identiques. Chaque régulation se fait par une tâche périodique T_i ($i \in \{1,2,3\}$). Nous faisons l'hypothèse que les 3 tâches ont les mêmes caractéristiques:

- la date d'activation initial: 0,
- la période: 0.2s,
- le temps d'exécution: 0.05s.

Les acquisitions sont réalisées périodiquement aux instants $t_i = 0.2 \cdot i$ et le seul événement critique correspond à l'application de la commande à la fin du traitement de la tâche. Considérons en premier lieu une politique à priorité fixe (FPP Fixed Priority Preemptive) : la tâche T_1 est plus prioritaire que T_2 qui elle même est plus prioritaire que T_3. En raison de la simultanéité des débuts d'exécution, nous obtenons un motif périodique élémentaire pour les exécutions (cf. schéma 5.14). Les dates de fin de traitements sont respectivement (0.05,0.10,0.15) pour les tâches T_1, T_2, T_3.

La seconde politique considérée est de type FIFO (premier arrivé, premier sorti), sans autre hypothèse cette politique peut conduire au cas précédent ou à une indétermination sur l'ordre de passage qui se traduit par une distribution aléatoire des temps de traitements dans $\{0.05, 0.10, 0.15\}$. Nous présentons au chapitre 7 la notion de politiques à poids et à taux et y renvoyons le lecteur pour une description de leur fonctionnement. Nous avons appliqué ces politiques qui amènent à un motif déterministe périodique (en absence de tout autre trafic). Nous avons choisis de représenter les résultats obtenus pour des poids identiques et des poids différents (400,200,100) et (200,200,100) et pour un politique à taux de paramètres 0.25 par tâche (confère les études menés en Italie par Buttazo pour plus de détails [Abeni *et al.*,]). Le schéma 5.14 donne la forme de l'allocation de la ressource dans le temps par l'ordonnanceur à poids avec la répartition (400,200,100).

Le schéma 5.15 montre les temps de réponse obtenus pour ces différentes configurations. La meilleure qualité est obtenue pour la tâche la plus prioritaire avec un ordonnancement à priorité. La politique à taux dégrade au maximum toutes les régulations et semble, à première vue, peu pertinente. La politique à poids est une alternative au cas à priorité, on obtient une qualité satisfaisante pour les tâches d'un poids de 100 et 200. Dans le cas où deux tâches se partagent le même poids, le résultat obtenu est très intéressant car les

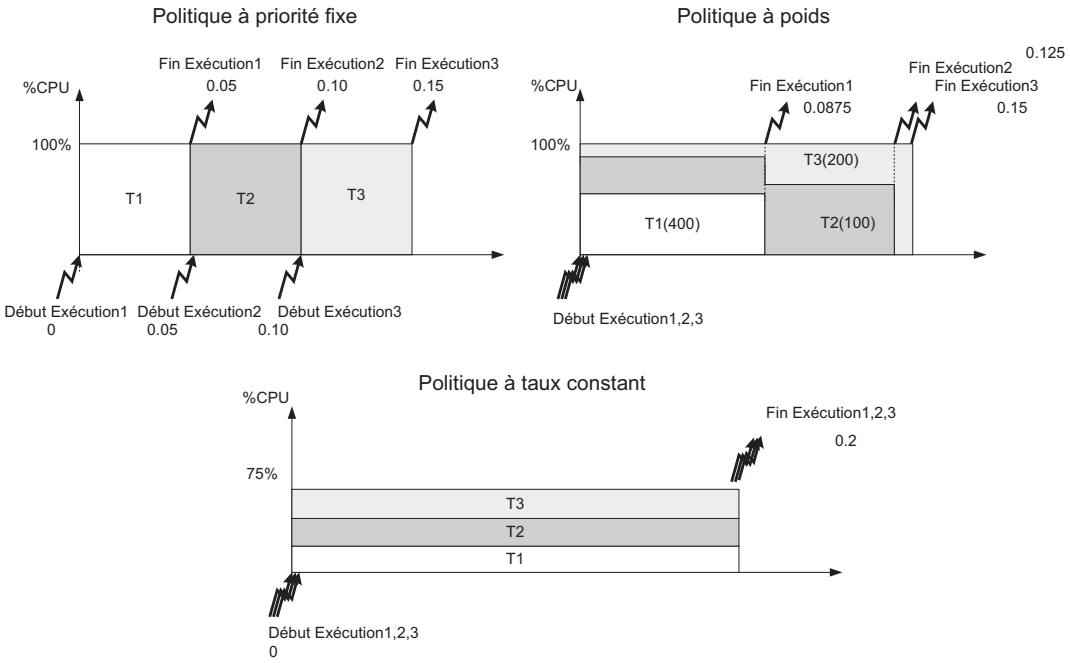


Figure 5.14: Dates de début et de fin d'exécution des activités pour différentes politiques d'ordonnancement

deux tâches de plus haut poids obtiennent la même qualité (on peut augmenter cette qualité en augmentant ce poids car la troisième tâche ne peut pas se terminer avant 0.15s et nous obtenons donc le même temps de réponse pour cette dernière).

L'intérêt premier de ces illustrations est de montrer que l'on peut évaluer l'influence des choix d'ordonnancement et que les différences peuvent être importantes. Le choix d'une politique d'ordonnancement de ressources est donc important vis à vis de la qualité des régulations. Nous présentons dans le paragraphe 5.4.2.5 les résultats obtenus lorsque l'on adapte la loi de commande aux caractéristiques de l'ordonnanceur. Ces résultats probants montrent l'intérêt des politiques qui permettent de connaître précisément les temps de traitement à l'avance.

5.4.2.4 Influence des modes de communication

Nous avons accordé une grande importance à la modélisation des modes de communication de l'architecture opérationnelle support (cf. chapitre 4). En effet, ils conditionnent des propriétés importantes sur les événements critiques, en particulier d'acquisition. La distinction la plus importante est liée à la notion de mode Push et de mode Pull (cf. section 4.2.4). Dans le premier cas (mode Push), le producteur d'échantillon est indépendant du consommateur alors que dans l'autre (mode Pull), il est subordonné au consommateur selon une coopération de type client/serveur. Dans le cas du Pull, la date de début d'exécution correspond donc à la date d'acquisition (retardée légérement du temps de traitement associé). Dans le cas du Push, il n'y a plus simultanéité entre la création et l'utilisation des données.

Dans la section précédente, le cas utilisé par défaut correspond à un mode Push tel que le producteur émet aux dates d'acquistion idéales. Le tableau suivant présente un comparatif pour les politiques à priorité et FIFO des résultats obtenus précédemment comparé à l'utilisation du mode Pull.

	FPP-Prio. 1	FPP-Prio. 2	FPP-Prio. 3	FIFO
Moyenne du gain en qualité en utilisant Pull au lieu de Push (critère: temps de réponse du système)	X	2,3%	2,65%	3%

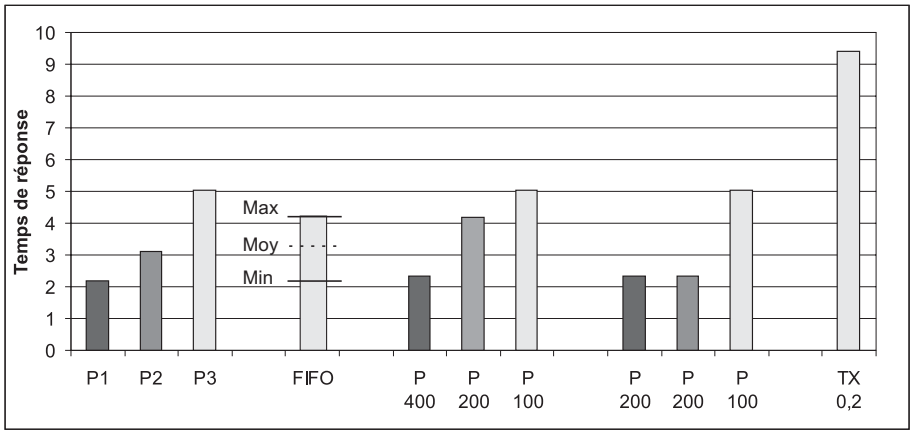


Figure 5.15: Influence des politiques sur le temps de réponse de la régulation. Les politiques considérées sont FPP, FIFO, à poids de paramètres (400,200,100), à poids de paramètres (200,200,100) et à taux de paramètres 0.25 par tâche

Sur cet exemple, la qualité obtenue est toujours meilleure dans le cas du mode Pull. En particulier dans le cas FIFO, même si le gain est faible au moyenne, il peut être supérieur à 10% sur cerains scénarii. Il n'est cependant pas possible de généraliser la comparaison. Dans le cas du calcul de la vitesse d'un mobile, le mode Push donne un meilleur résultat car il ne crée pas de gigue sur l'acquistion au contraire du mode Pull (cf. section 5.2.1.3). Ces exemples montrent surtout que les modes de coopération ont une influence sur la qualité de fonctionnement du système et que l'on peut l'estimer, en particulier grâce aux techniques de simulation proposées dans ce chapitre.

5.4.2.5 Adaptation au retard, mise en évidence des besoins de prédictabilité

Les politiques d'ordonnancement étudiées ont été utilisées en absence de tout autre trafic temps réel. L'existence d'un autre trafic peut influer sur les dates de fin des traitements qui sont alors soumis à un certain aléa. Dans le cas des politiques à priorité, même si l'on garantit le respect de l'échéance sur requête, on peut avoir un une variabilité importante. Dans le cas de la politique à taux, par définition, la variabilité est très faible. La présence d'autres tâches ne peut avoir que très peu d'influence car l'ordonnancement est garantie avec une faible granularité.

Nous présentons en annexe quelques possibilités d'adaptation de la commande à la connaissance du temps de traitement. Par exemple, il est possible d'utiliser un prédicteur de Smith pour compenser la présence d'un retard constant (d'une période dans notre exemple). Nous avons appliqué cette technique à l'exemple proposé de manière à compenser le retard constant. Le schéma 5.16 montre l'évolution de la réponse avec et sans correction dans le cas d'un ordonnancement à taux constant avec, en plus du retard constant, une variation possible du temps de traitement de 1%. On remarque le gain énorme apporté par le prédicteur. Le schéma 5.17 présente un comparatif des différents temps de réponse avec et sans utilisation du prédicteur de Smith. Les meilleurs résultats sont obtenus en compensant la politique à taux constant. En effet, cette dernière permet par construction de donner un temps de traitement constant et multiple de la période pour les traitements ce qui permet de les compenser à l'aide du prédicteur de Smith. L'utilisation sur les différentes priorités montrent que le prédicteur de smith n'a d'intérêt que si les temps de traitement sont proches de la période (cas de la priorité 3). Couplé au prédicteur de Smith, la politique à taux constant devient donc très pertinente et permet d'obtenir une qualité équivalente pour toute les activités à l'activité la plus prioritaire dans l'utilisation d'une politique à priorité. Cet exemple montre donc tout l'intérêt d'une propriété de prévisibilité sur les temps de traitement, dont la mise en place fait l'objet du chapitre 7.

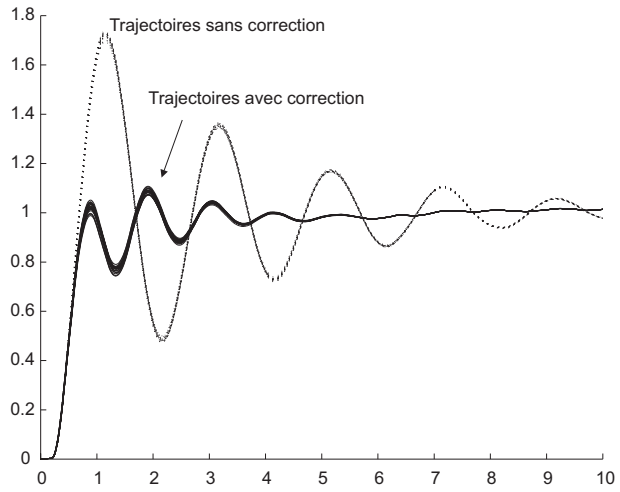


Figure 5.16: Évolution de la sortie du système, en présence d'un retard constant d'une période, avec et sans l'utilisation d'un prédicteur de Smith

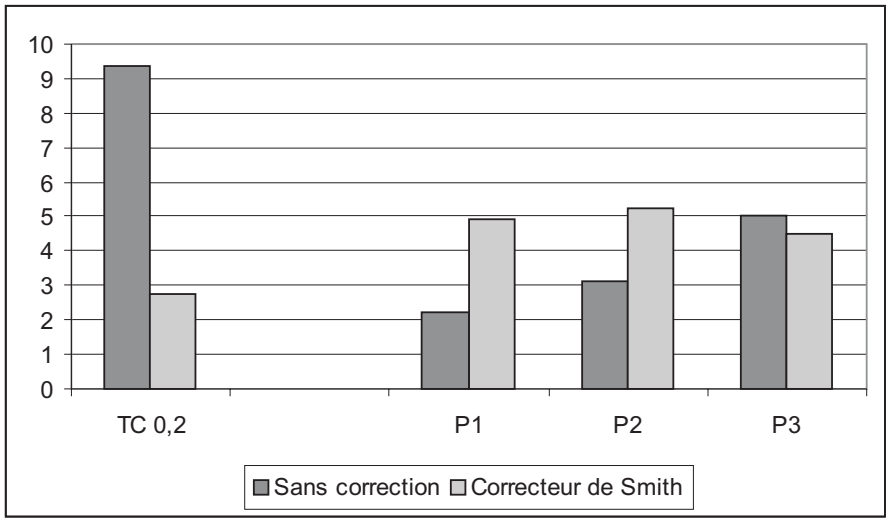


Figure 5.17: Intérêt du prédicteur de Smith suivant les politiques d'ordonnancement utilisées; sont étudiées la politique à taux constant de paramètre 0.25 et FPP

5.4.3 Conclusion

Ce chapitre présente différentes techniques qui permettent d'évaluer l'influence des choix d'implémentation sur une régulation. Même si certains résultats analytiques sont disponibles, ils permettent difficilement de faire le lien avec les critères de qualité d'usage courant. L'approche par simulation s'avère donc une alternative intéressante. La mise en place d'un simulateur se révèle cependant complexe et nécessite une compréhension fine des différents formalismes propres à l'automatique, en particulier transformée de Laplace et de Fourrier mais aussi représentation d'état.

La mise en place d'un outil de simulation a été rendu nécessaire mais ne constitue pas une finalité. Il permet de montrer que les relations qui lient les caractéristiques de l'architecture opérationnelle (en particulier temporelles) et l'évolution d'un sytème régulé sont complexes et peu intuitives. Cela permet en particulier de mieux comprendre quels sont les vrais besoins d'une régulation vis à vis des caractéristiques des supports d'exécutions des algorithmes de commande. Deux résultats intéressants sont apparus lors de l'étude de notre exemple de régulation d'un moteur. Tout d'abord, l'évolution de la qualité (en particulier le temps de réponse du système) ne suit pas un profil linéaire. Il est donc important de connaître ce profil si l'on veut améliorer la qualité d'une régulation. Ensuite, la variabilité des temps de traitement peut avoir des conséquences très diverses comme par exemple:

- une dégradation de la qualité plus importante que ce que l'on pourrait intuitivement considérer comme le pire cas,
- l'influence relativement faible des bornes sur les temps de traitement vis à vis de la moyenne.

Il est donc nécessaire de faire attention dans la définition des contraintes (en particulier temporelles) liées aux algorithmes de traitement et de valider les hypothèses faites grâces à des techniques équivalentes à celle proposée dans ce chapitre. Enfin, cette étude nous a permis de montrer l'intérêt de politiques d'ordonnancement plus prévisibles dans le domaine des applications temps réel de contrôle-commande par opposition aux politiques actuelles qui favorisent la faisabilité du système ou la minimisation des temps de réponse.

Chapter 6

Modélisation de la Sûreté de Fonctionnement d'une application de contrôle-commande

Dans ce chapitre, nous proposons une méthode d'évaluation de la Sûreté de Fonctionnement des Architectures Opérationnelles de Contrôle-Commande. Pour cela, nous définissons la notion de défaillances sur les informations manipulées par l'application de contrôle-commande, puis, à partir d'un exemple simple, nous introduisons d'une part, la notion d'état d'erreur et, d'autre part, un mode d'évaluation de la fiabilité d'une architecture. Nous présentons enfin l'extension de l'étude à un cas plus général.

La méthode que nous développons dans la suite de cette section, montre comment les choix

- de support matériel (avec ou sans redondance),
- de la distribution d'une architecture fonctionnelle sur une architecture support,
- de modes de coopération (client/serveur, introduction de mémoires partagées ou de files d'attente),
- de mécanismes réalisant les règles de contrôle des activités (périodiques ou sur événement),

peuvent avoir une influence sur le risque de défaillance du système global.

De plus, nous évaluons ce même risque en présence des solutions de tolérance aux fautes spécifiées au niveau de l'architecture fonctionnelle (comme le sur-échantillonnage). Ces différents choix peuvent être exprimés plus facilement à l'aide du formalisme présenté au chapitre 4. La figure 6.1 présente un exemple d'utilisation de cette représentation, qui permet de mieux comprendre les règles de création et d'utilisation des données utilisées au cours du fonctionnement du système. Dans l'exemple considéré, le bloc 2 (élaboration de la commande) contrôle l'ensemble des traitements. Il est activé périodiquement et récupére par un pull les données de l'acquisition (communication client/serveur) et par un push l'actionnement (activation sur réception du message). Le schéma fait aussi apparaître les architectures supports (ici 3 ordonnanceurs de ressource CPU et un réseau CAN) .

Cette analyse repose sur deux hypothèses. La première est de considérer que les lois de commande sont correctes et pertinentes pour le contrôle du système physique en l'absence de défaillance du système . La deuxième est qu'il n'y a pas de causes de défaillances du système physique commandé autres que celles provenant du système de contrôle (comme par exemple la rupture d'un élément mécanique) . Nous étudions, alors, les causes de défaillances du système physique dans la chaîne de traitement de l'information. La figure 6.2 donne une

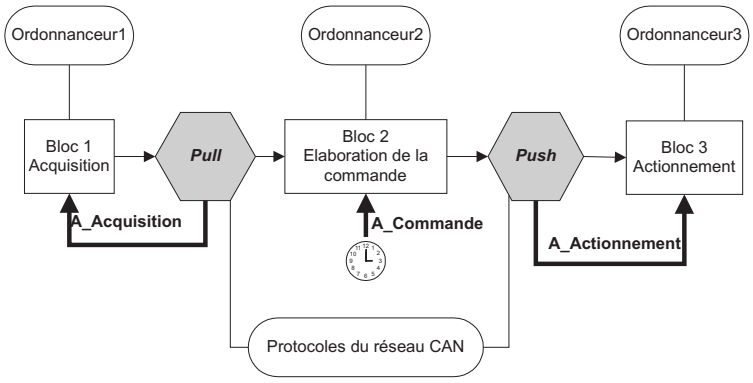


Figure 6.1: Exemple d'architecture opérationnelle détaillée

représentation schématique d'une Architecture Opérationnelle de contrôle-commande hiérarchisée par niveau de service. Elle montre les liens qui existent entre une faute à un niveau de service donné et une défaillance du système global. Notons que ce schéma ne fait pas apparaître les fautes latentes de conception (erreur sur un mot mémoire, erreur de codage, erreur de dimensionnement de piles, ...). De plus, à chaque niveau "de service", nous faisons apparaître dans ce schéma, quelques mécanismes qui peuvent être mis en place pour éviter la propagation de la faute initiale. Il s'agit, par exemple, d'une prévention de fautes par utilisation d'un médium de communication tolérant de fortes perturbations électromagnétiques, d'une redondance active de services avec vote majoritaire, d'un sur-échantillonnage des données, ...

Nous proposons par la suite une méthode qui, connaissant la loi d'apparition d'une faute dont la conséquence est une "défaillance" sur une information, permet d'évaluer le risque d'avoir une défaillance au niveau du système global. Le terme "défaillance" sur une information, exprime le fait que la valeur de l'information consommée à un instant par une activité n'est pas cohérente avec l'état instantané du système. Par exemple, dans le cas des systèmes échantillonnés, pour un échantillon donné, une telle défaillance peut provenir d'une défaillance d'un capteur (information altérée voire non produite), d'un processeur indisponible (information non produite "à temps") ou d'une faute de transmission de l'information (information altérée ou information transmise "trop tard"). La "défaillance" de l'information de commande peut alors éloigner le système de son état normal désiré et, compte tenu des lois de commandes implantées, de le conduire à la défaillance du système global.

La défaillance du système physique s'apparente à un dépassement sur une grandeur caractéristique. Les mesures de fiabilité sont donc liées à l'évaluation du risque de ce dépassement (cf. chapitre 2).

Nous avons choisi de considérer deux métriques de fiabilité d'usage courant:

- le temps moyen d'atteinte de la défaillance (cf. section 6.1.3.1),
- la probabilité de défaillance en 1 heure.

La première métrique est d'utilisation très courante et correspond à la moyenne du temps de fonctionnement avant défaillance. On peut alors comparer ce temps avec la durée de vie de l'installation et ainsi estimer si la disponibilité du système est suffisante. La seconde permet de connaître la probabilité de défaillance (dépassement sur la grandeur critique) sur 1 heure de fonctionnement. Cette métrique est intéressante car elle permet de donner une propriété d'intégrité du système sous forme de niveau SIL (Safety Integrity Level). Les niveaux SIL ont été initialement définis pour décrire le niveau d'intégrité d'un système de contrôle ([IEC, 1997]) mais peuvent être étendus à un système complet (qui est alors considéré comme un système critique). Le tableau suivant présente les niveaux d'intégrité et les propriétés à garantir sur le fonctionnement du système:

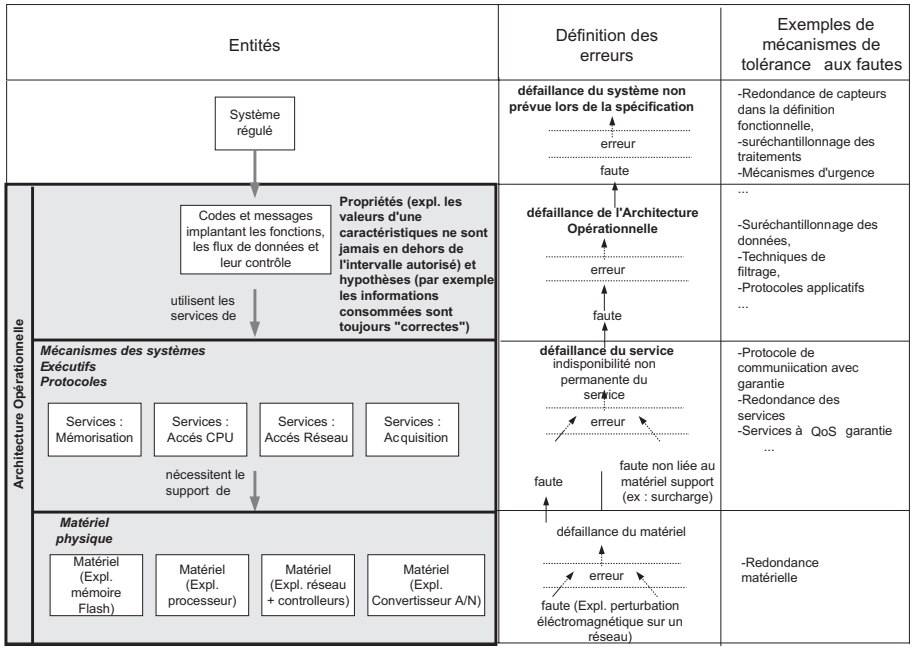


Figure 6.2: Causalité des défaillances

Niveau d'intégrité	Probabilité de défaillance en 1 Heure
SIL0	si inconnue ou supérieure à 10^{-5}
SIL1	de 10^{-6} à 10^{-5}
SIL2	de 10^{-7} à 10^{-6}
SIL3	de 10^{-8} à 10^{-7}
SIL4	de 10^{-9} à 10^{-8}

Le but de cette étude est de montrer comment calculer ces métriques de fiabilité dans le cas d'un système régulé et ainsi de pouvoir quantifier l'apport en terme de sûreté de fonctionnement des différents choix effectués au cours de l'implémentation.

6.1 Etude de cas : " système de contrôle du niveau de liquide dans une cuve "

Ce système étudié est très simple avec l'objectif d'introduire clairement les concepts utilisés. Le système est constitué d'une cuve de hauteur H_{max}. L'ensemble du dispositif est représenté à la figure 6.3. L'objectif du système de commande est de maintenir le niveau du liquide dans la cuve à une valeur donnée H_0. La cuve sert à des opérations annexes qui sont ici considérées comme des perturbations, c'est-à-dire comme des évènements non contrôlables (noté Perturbation sur le schéma 6.3). Le liquide est incompressible, le modèle considéré de la cuve est un intégrateur, c'est- à-dire que le niveau d'eau dans la cuve à chaque instant est proportionnel à la quantité de liquide entrant moins la quantité de liquide sortant. La quantité de liquide est proportionnelle à la hauteur du liquide mesurée par un capteur de niveau. L'action sur la cuve est faite par un système composé d'une vanne et d'une pompe. La loi de contrôle est de type " plus ou moins à seuil ". Le correcteur utilise l'écart entre la consigne et le niveau actuel pour décider de la valeur de la commande à appliquer. A chaque période d'échantillonnage, trois cas peuvent se présenter:

- le niveau est égal à H_0,aucune action n'est demandée,

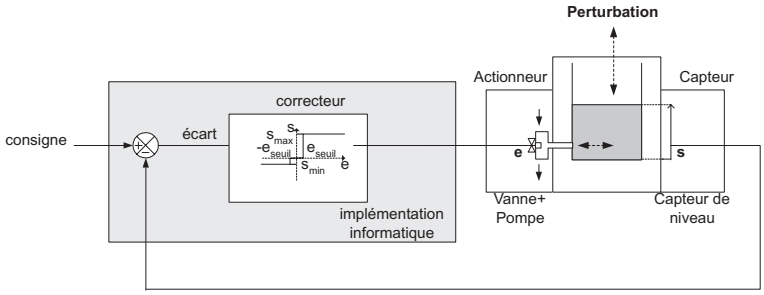


Figure 6.3: Système de contrôle de niveau dans une cuve

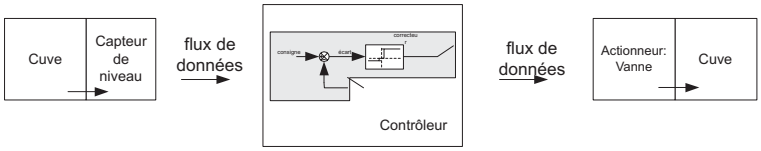


Figure 6.4: Chaîne de l'information

- le niveau est supérieur à H_0, une consigne de vidange doit être appliquée à l'actionneur; nous considérons que la pompe retire une quantité de liquide Q_v pendant la période d'échantillonnage δt,

- le niveau est inférieur à H_0, une consigne de remplissage doit être envoyée à l'actionneur ; de la même manière que pour une vidange, nous considérons, que la pompe introduit une quantité de liquide Q_R pendant la période d'échantillonnage δt.

Nous supposons, de plus, que $Q_v < Q_R$ (dans le cas contraire les risques de défaillances sont infimes). Par la suite, nous travaillons avec les hauteurs de liquide H_v et H_R proportionnelles respectivement à Q_v et Q_R avec $N = [H_R/H_V]$.

La seule défaillance du système physique que nous prenons en compte est le débordement de la cuve, c'est-à- dire le cas où le niveau de liquide dans la cuve H dépasse un niveau H_{max}.Cette étude peut cependant être étendue à d'autre type de dysfonctionnement.

6.1.1 Architecture Fonctionnelle: fonctions et flux de données

L'élaboration de la commande peut être vue comme une chaîne de traitement de l'information constituée de 3 fonctions :

- une fonction acquisition de l'état du système grâce à un capteur,

- une fonction d'élaboration de la commande à appliquer à l'actionneur, consigne calculée selon la loi de commande choisie et à partir de l'état du système à contrôler,

- une fonction actionnement qui modifie la dynamique du système par la commande d'un actionneur (vidange / remplissage / aucune action).

Les flux de données entre ses différentes fonctions véhiculent des informations qui doivent être cohérentes avec l'état du système et la stratégie de commande vue ci-dessus. Le schéma 6.4 montre les flux de données dans le cas de l'exemple traité.

6.1.2 Défaillance de l'information de commande vs défaillance du système.

Nous introduisons la notion de "défaillance de l'information", liée généralement à une violation des règles de contrôle sur les flux de données correspondants. En règle générale, nous considérons qu'il y a "défaillance de l'information de commande" lorsque la consigne appliquée par l'actionneur est incohérente avec l'état courant du système. Nous dirons, alors, que cette consigne est erronée; elle peut l'être pour plusieurs raisons:

- suivant la distribution qui est faite de l'architecture fonctionnelle réalisant l'application de contrôle-commande, cette consigne peut, par exemple, être acheminée jusqu'à l'actionneur via un réseau de communication; elle est alors erronée si la trame qui la contient est corrompue ou si cette trame ne parvient pas à l'actionneur assez tôt et qu'il utilise alors la valeur précédemment transmise.
- dansle cas où la valeur acquise au début de la période d'échantillonnage et qui sert au calcul de la commande est erronée.

Nous pouvons, alors, définir les défaillances de l'information pour l'exemple de régulation de niveau dans une cuve d'eau. Ainsi que nous l'avons signalé précédemment, nous supposons, dans un premier temps, que les perturbations sont nulles. En effet l'existence des perturbations peut être intégrée dans la définition de la défaillance (sous forme d'une notion de défaillance probable pour un niveau de perturbation donné) et n'intervient, donc, pas directement dans l'étude. A tout instant, nous nous placerons donc dans le pire cas.

La seule défaillance considérée au niveau du système global est le débordement de la cuve et, pour cette présentation, nous ne nous intéressons, donc, qu'aux "défaillances de l'information de commande" qui augmentent le niveau de liquide. Deux situations sont donc possibles :

- la consigne envoyée à l'actionneur demande un remplissage alors qu'il faudrait vidanger ou ne rien faire,
- la consigne envoyée à l'actionneur demande de "ne rien faire" alors qu'il faudrait vidanger.

Le pire cas est donné par la première situation. La seule défaillance de l'information considérée est donc une commande de remplissage au moment et dans l'état où il faudrait vidanger ou ne rien faire.

6.1.2.1 Réponse du système à une défaillance isolée de l'information.

Le système de contrôle ayant pour rôle de maintenir la hauteur de liquide au niveau H_0, il doit appliquer des ordres de vidange. La figure 6.5 représente la réponse du système à une défaillance unique de la commande. A l'occurrence de cette défaillance, la consigne que reçoit l'actionneur est une commande de vidange. Donc, pendant une période d'échantillonnage, δt, le niveau de la cuve augmente de H_R. Si lors de l'échantillonnage suivant, la défaillance n'est plus présente, l'actionneur recevra une commande de vidange et ce jusqu'à ce que la hauteur dans la cuve soit revenue au niveau H_0. Comme la capacité de vidange est plus faible que la capacité de remplissage, le système requiert N pas d'échantillonnage pour retrouver cette position d'équilibre H_0 (où $N = [H_R/H_V]$).

6.1.2.2 Défaillance du système

Nombre de défaillances admissibles consécutivement. Dans le cas du système considéré (un intégrateur), on obtient facilement la réponse du système pour Nc défaillances consécutives de l'information (en considérant toujours le pire cas). Le temps avant retour à l'équilibre est donc $N.Nc.\delta t$ et la hauteur maximale atteinte après la dernière défaillance est de $Nc.H_R + H_0$. Le nombre de défaillances consécutives limites avant défaillance du système physique (c'est-à-dire débordement, dans ce cas) est $N_{\text{ecl}} = (H_{max} - H_0)/H_R$.

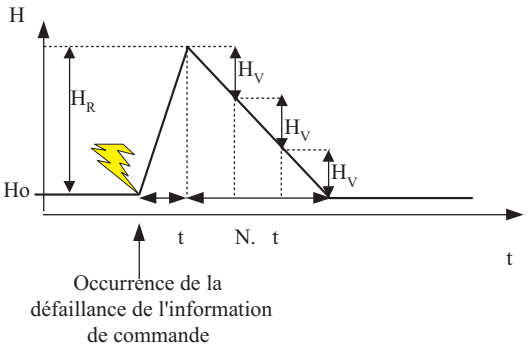


Figure 6.5: Réponse à une "défaillance de l'information de commande "

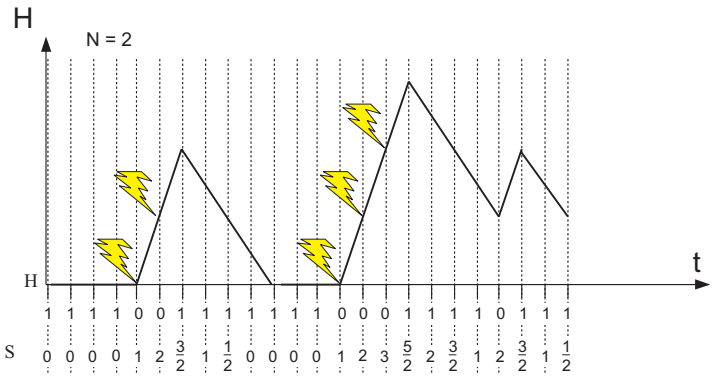


Figure 6.6: Réponse du système à une séquence d'ordres contenant des défaillances

Evolution admissible du système en présence de "défaillances de l'information de commande". Nous cherchons alors à caractériser toutes les séquences de commandes qui mènent à une défaillance du système physique. Pour ceci, nous définissons une fonction qui lie le niveau atteint dans la cuve à la séquence de commande .

Définition d'une séquence. Nous considérons toujours un seul type d'erreur : un ordre de remplissage non attendu. Soit ε_i l'état de la commande pour l'échantillon (ou le pas) n°i : s'il y a défaillance de l'information de commande $\varepsilon_i = 0$ sinon $\varepsilon_i = 1$. Une séquence est une suite $(\varepsilon_i, \varepsilon_{i+1} ... \varepsilon_j, \varepsilon_{j+1})$ d'informations de commande relatives aux échantillons $i, i+1, ..., j, j+1$. La suite (11011111111) correspond à une séquence de 2 échantillons sans erreurs, 1 défaillance de l'information de commande, et 8 échantillons sans erreurs.

Réponse du système à une séquence. Le niveau H_i après i échantillons est donné par :

$$H_i = S_i . H_R$$

où

$$S_i = S_{i-1} - \frac{1}{N}\varepsilon_i \delta(S_{i-1}) + (1 - \varepsilon_i)$$

avec

$$\begin{aligned} \delta(x) &= 0 \,\text{si}\, x = 0 \\ &= 1 \,\text{si}\, x \neq 0 \end{aligned}$$

S_i représente une mesure de l'écart à l'équilibre H_0 dû à la présence d'une ou plusieurs défaillances de l'information de commande dans une séquence. On peut considérer que S_i caractérise, à tout instant, " l'état d'erreur " du système (voir figure 6.6). Dans l'exemple, une défaillance de l'information de commande fait monter l'état d'erreur de 1 et il faut N pas d'échantillonnage sans défaillance pour revenir à l'équilibre. La défaillance du système global correspondant à $H > H_{max}$ apparaît pour $Si > Smax$ avec $Smax = \frac{H_{max}}{H_r}$.

6.1.3 Etude de fiabilité

La connaissance de l'architecture informatique support, associée à un modèle d'apparition des fautes sur cette architecture, peut être utilisée pour évaluer les risques de défaillances, soit, sur l'exemple de la cuve d'eau, le risque de dépassement du niveau acceptable maximal dans la cuve S_{max} , soit tel que nous l'avons indiqué précédemment, le risque que S_i devienne supérieur à S_{max} . Nous considérons le cas où chaque consigne reçue par l'actionneur a la même probabilité $1 - p$ d'être erronée (et donc p d'être correcte). Dans la suite, nous fixons les différents paramètres de la cuve ainsi:

- $N = 3$ (nombre de consignes sans erreur pour revenir à l'équilibre); une consigne erronée fait monter le niveau de la cuve de 3 unités alors qu'une consigne correcte le fait baisser de 1 unité si le niveau mesuré était supérieur à H_0,
- S_{max} correspond à l'état absorbant (noté 11 sur le schéma).

6.1.3.1 Modélisation

L'évolution du niveau de la cuve d'eau (S_i) peut être modélisée par une chaîne de Markov, son espace d'état étant discret et toute l'information importante pour l'évolution future de la cuve étant contenue dans la valeur

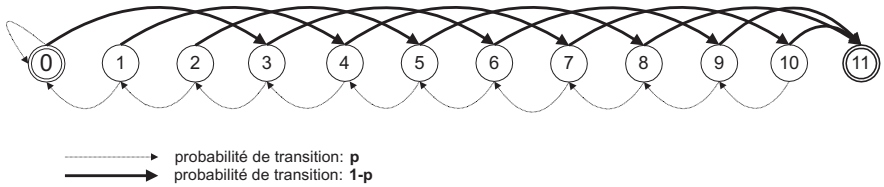


Figure 6.7: Diagramme de transition où 0 est l'état initial et 11 est l'état absorbant

de l'état courant. Nous choisissons cette chaîne en temps discret avec comme instants d'observation les dates successives d'application de la consigne. Les états de la chaîne sont les différents niveaux possibles de la cuve, dans notre exemple de l'état 0, le point d'équilibre (hauteur de la cuve H_0), à l'état 11, le point de défaillance (débordement). Cet état particulier est un état absorbant qu'il est impossible de quitter une fois atteint ce qui correspond au fonctionnement réel du système (le niveau de la cuve n'est jamais supérieur à H_{max}). Le diagramme de transitions est représenté à la figure 6.7.

La matrice de transitions de la chaîne de Markov est:

$$P = \begin{array}{c|cccccccccc} P_{i,j} & 0 & 1 & 2 & 3 & 4 & 5 & .. & 8 & 9 & 10 & 11 \\ \hline 0 & p & 0 & 0 & 1-p & 0 & 0 & .. & 0 & 0 & 0 & 0 \\ 1 & p & 0 & 0 & 0 & 1-p & 0 & .. & 0 & 0 & 0 & 0 \\ 2 & 0 & p & 0 & 0 & 0 & 1-p & .. & 0 & 0 & 0 & 0 \\ . & . & . & . & . & . & . & .. & . & . & . & . \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 & .. & 0 & 0 & 0 & 1-p \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & .. & p & 0 & 0 & 1-p \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & .. & 0 & p & 0 & 1-p \\ 11 & 0 & 0 & 0 & 0 & 0 & 0 & .. & 0 & 0 & 0 & 1 \end{array}$$

A partir de cette matrice, il est possible d'effectuer un certain nombre de calcul pour évaluer la fiabilité du système. En particulier, P^n nous donne la probabilité d'être dans l'état défaillance après n consignes en partant de l'état d'équilibre 0. Il est également possible de calculer le temps moyen pour atteindre l'état de défaillance ainsi que sa variance.

On note N_i la variable aléatoire qui donne le premier temps d'atteinte de l'état de défaillance en partant de n'importe quel état i. L'ensemble $\mathcal{T}$ est l'ensemble des états transitoires (tous sauf l'état absorbant). A l'aide d'une analyse classique en "un pas", on obtient:

$$N_i = \begin{cases} \gamma_i + N_j, & \text{avec probabilité } \sum_{j\in\mathcal{T}} P_{j,i}, \\ \gamma_i, & \text{avec probabilité } P_{i,11} \end{cases}$$

avec $\gamma_i = 1$ si $i \neq 11$ et 0 sinon. On peut en dériver l'espérance de N_i:

$$\begin{aligned} E[N_i] &= P_{i,11}E[\gamma_i] + \sum_{j\in\mathcal{T}} P_{i,j}E[\gamma_i + N_j] \\ &= \gamma_i + \sum_{j\in\mathcal{T}} P_{i,j}E[N_j] \end{aligned}$$

Cet ensemble de 12 équations linéaires se résout aisément à l'aide du logiciel Maple. Nous considérons que le système est remis en marche à l'état 0 après une défaillance; $E[N_0]$ est donc le temps moyen d'atteinte de la défaillance en partant de l'état initial.

D'une façon similaire, il est possible de calculer la variance des temps d'atteinte de l'état de défaillance qui est par définition $V[N_i] = E[N_i^2] - E[N_i]^2$. On a:

$$N_i^2 = \begin{cases} (\gamma_i + N_j)^2, & \text{avec la probabilité } \sum_{j\in\mathcal{T}} P_{i,j}, \\ \gamma_i^2, & \text{avec la probabilité } P_{i,11} \end{cases}$$

L'espérance de N_i^2 est:

$$\begin{aligned} E[N_i^2] &= P_{i,11}E[\gamma_i^2] + \sum_{j\in\mathcal{T}} P_{i,j}E[(\gamma_i + N_j)^2] \\ &= \gamma_i^2 + \sum_{j\in\mathcal{T}} P_{i,j}E[(N_j + \gamma_i)^2] \\ &= \gamma_i + \sum_{j\in\mathcal{T}} P_{i,j}E[N_j^2] + 2\sum_{j\in\mathcal{T}} P_{i,j}E[N_j]\gamma_i \end{aligned}$$

Après résolution de cet ensemble d'équations, la variance des temps d'atteinte de l'état de défaillance à partir de l'état 0 est $V[N_0] = E[N_0^2] - E[N_0]^2$.

6.1.3.2 Fiabilité de diverses architectures

En utilisant ces résultats, nous pouvons mesurer l'influence de choix de conception sur la fiabilité du système (la fiabilité correspond à l'aptitude d'un système à accomplir sa fonction). Cinq architectures sont évaluées (cf. 6.8):

- l'architecture 1 est le système tel que décrit au paragraphe 6.1.1 avec $N = 3$, $S_{max} = 11$ et une probabilité d'erreur sur la consigne de 5% ($p = 0.95$). La fréquence des consignes est de une par minute,
- l'architecture 2 est identique au cas 1 mais la probabilité d'erreur est divisée par 5 ($p = 0.99$) ce qui correspond par exemple à l'utilisation d'un support de transmission blindé (cf. [Barrenscheen, 1997] pour des mesures sur la robustesse relative de différents support),
- l'architecture 3 est identique au cas 1 mais la fréquence des consignes est doublée (toutes les 30 secondes). La quantité d'eau injectée lors d'un ordre de remplissage étant divisée par 2, cela revient à fixer $S_{max} = 22$,
- l'architecture 4 est identique au cas 1 mais une même consigne est transmise deux fois sous une hypothèse de *fail-silence* [Poledna *et al.*, 2000]: une donnée reçue est nécessairement bonne au niveau de sa valeur mais la donnée peut-être occasionnellement manquante. Une erreur se produit sur la consigne lorsque les deux messages qui la contiennent sont corrompus. Sous l'hypothèse d'erreurs indépendantes, la probabilité d'un tel événement est p^2, soit ici 0.0025,
- l'architecture 5 est obtenue par modification de la capacité de récupération du système contrôlé, on augmente H_v (en changeant la loi de commande ou si nécessaire les actionneurs). On diminue ainsi le rapport N, on prend ici N=2 au lieu de N=3 pour l'architecture de référence 1. Il ne faut plus que 2 échantillons au système contrôlé pour corriger un ordre défaillant.

Nous avons choisi de comparer les différentes architectures vis à vis de deux métriques de fiabilité:

- le temps moyen d'atteinte de la défaillance (cf. 6.1.3.1),
- la probabilité de défaillance en 1 heure (qui sert de référence pour la définition des niveaux d'intégrité SIL cf. [IEC, 1997]).

Architecture	Espérance du temps d'atteinte de la défaillance	Variance du temps d'atteinte de la défaillance	Probabilité de défaillance en 1 heure	Safety Integrity Level
Cas 1 : Architecture de référence	20 jours	20 jours	0.19e-02	-
Cas 2 : La probabilité d'erreurs de transmissison diminue fortement	42 années	42 années	0.24e-05	SIL1
Cas 3 : La fréquence du contrôle est doublée	227 années	227 années	0.44e-06	SIL2
Cas 4: Architecture "fail silence"	11000 années	11000 années	0.89e-08	SIL4
Cas 5 : Modification de la capacité du systÈme controlé à récupérer une erreur	13 années	13 années	0.68e-05	SIL1

Figure 6.8: Mise en évidence des résultats obtenus

La première architecture sert de référence et ne possède aucun mécanisme de tolérances aux fautes. Les architectures 2, 3 et 4 peuvent être vues comme une extension de l'architecture 1 avec la mise en oeuvre d'une technique de tolérances aux fautes (cf. figure 6.2).

Les différentes techniques apportent un gain certain au niveau de la fiabilité du système global. On remarque en particulier que les solutions qui prennent en compte la possibilité de défaillance au niveau de la conception de l'architecture fonctionnelle et/ou opérationnelle (sur-échantillonnage et garantie de "fail silence") donnent de meilleurs résultats que des solutions de bas niveaux vis-à-vis des fautes primaires (blindage du câble). Les gains obtenus en utilisant des architectures possédant des tolérances aux fautes sont très importants (passage d'un niveau SIL0 à SIL4 grâce à l'architecture "fail silence"). Des solutions simples et peu onéreuses comme le sur-échantillonnage sont également intéressantes (passage de SIL0 à SIL2).

Enfin cette étude montre que, dans le cadre des systèmes échantillonnés, de nombreux éléments interviennent dans la fiabilité du système global. Ainsi, les différentes techniques qui ont été utilisées pour augmenter la fiabilité de l'architecture 1 se placent à des niveaux de conception très différents:

- l'amélioration du matériel en fonction des niveaux de perturbations rencontrés (Architecture 2),
- le sur-échantillonnage (Architecture 3), qui est une solution de haut niveau et qui doit être définie dans l'Architecture Fonctionnelle,
- la mise en place d'une architecture tolérante aux fautes (Architecture 4), qui relève d'une caractéristique désirée de l'Architecture Opérationelle support,
- la définition d'une loi de commande qui prenne en compte les risques de défaillances de l'information en plus des critères habituels liés à la dynamique du système physique (Architecture 5).

6.2 Extension de l'étude à un cas plus général

Nous avons présenté sur un exemple, comment étudier la notion de fiabilité sur un système simple possédant une boucle de régulation. Nous allons essayer dans cette partie, de montrer les possibilités d'extension de l'approche proposée aux systèmes classiquement utilisés.

Le calcul de la fiabilité repose sur le choix d'un modèle de défaillance du système global/défaillance de l'information, ce modèle se décompose en 4 points:

- la modélisation de la défaillance globale du système (débordement pour la cuve),
- la modélisation des défaillances de l'information et de leurs éventuelles propagations ,
- le lien entre l'apparition de la défaillance globale et l'existence de défaillance de l'information, rendu explicite par la fonction d'état d'erreur,
- la modélisation de l'occurrence des défaillances de l'information.

6.2.1 Prise en compte de plusieurs défaillances globales

L'extension du modèle de défaillance du système global peut être réalisée simplement; il suffit de lister l'ensemble des grandeurs caractéristiques du système (dérivées incluses) et de déterminer l'ensemble des défaillances qui correspondent à un dépassement sur une de ces grandeurs (ou sur une combinaison). La forme de l'étude précédente est inchangée mais il existe alors différents états de défaillances.

Différents types de défaillances de l'information peuvent aussi être considérés, comme par exemple l'existence de valeurs totalement erronées mais aussi de valeurs légèrement erronées. La difficulté réside alors dans le choix des paramètres de ces modèles qui nécessite une étude préalable de l'architecture opérationnelle choisie en terme de risque de défaillances (l'étude [Gaujal and Navet, 2001] présente un exemple des relations entre une architecture opérationnelle et l'apparition de défaillances de l'information). Il est aussi possible de faire apparaître des erreurs liées à l'architecture de contrôle, c'est à dire de prendre en compte la notion de valeur par défaut ou de réutilisation de la dernière valeur reçue en cas d'absence de nouvelle valeur attendue.

6.2.2 Mise en évidence de la propagation des défaillances de l'information dans les modèles d'architectures informatisées

Nous cherchons à modéliser l'existence de défaillances de l'information. Les seules défaillances qui nous intéressent concernent les échantillons qui transitent dans le système. Les deux principaux cas à considérer sont:

- le fait que des échantillons transmis ou utilisés portent des valeurs erronées,
- l'absence de production d'échantillons attendus.

L'intérêt de modéliser ces défaillances est de permettre l'étude de leur propagation au sein de l'architecture. Il est important de pouvoir étudier l'influence des choix d'architectures sur la propagation ou la non propagation d'une défaillance au sein du système de contrôle (cf. [Shin and Kim, ; Cunha *et al.*, 2001] pour plus de détails). Cette modélisation permet aussi de valider des mécanismes de tolérance aux fautes.

Nous avons présenté les points importants de la modélisation de l'architecture informatisée d'une application de contrôle-commande (cf. chapitre 4). Nous pouvons compléter cette modélisation de propriétés liées à la propagation des défaillances et ainsi être capable de quantifier l'apparition de ces défaillances.

La modélisation des phénomènes de défaillance est facilitée en SDL car :

- le langage permet d'associer des valeurs aux signaux ,
- il est possible de définir des choix multiples dans le comportement d'un bloc.

On peut donc associer aux échantillons un état qui représente s'ils sont défaillants ou non. Cette valeur d'état peut être manipulée à travers un bloc et se propager entre les différents blocs. Le schéma 6.9 représente les motifs élémentaires permettant cette modélisation. Le premier motif (à gauche) représente l'apparition d'une

défaillance sur une information produite. Le symbole *"any"* permet de définir l'existence de deux comportements possibles du système, le message envoyé peut être correct ou incorrect. Le second motif définit la notion de défaillance du bloc. Dans cet exemple, le bloc est dans un état de défaillance dès qu'il reçoit un message défaillant. Il est bien entendu possible de modéliser des comportements plus complexes où le bloc ne passe pas aussi facilement en état de défaillance (existence de mécanismes de tolérances aux fautes). Le dernier motif lie la défaillance du bloc à la défaillance des informations produites.

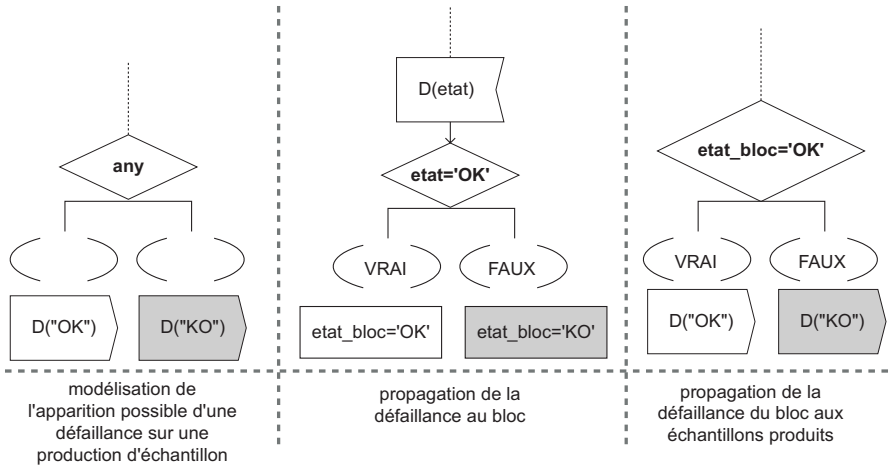


Figure 6.9: Motifs élémentaires permettant de modélisation l'apparition et la propagation des défaillances dans une architecture de contrôle-commande

Il est possible de décrire des règles plus complexes de propagation des défaillances. Le schéma 6.10 présente un exemple de modélisation d'un mécanisme de tolérance aux fautes. On considère que le bloc est non défaillant tant qu'il n'a pas reçu 3 échantillons défaillants consécutifs.

6.2.3 Autres modèles d'occurrences des défaillances de l'information

Le modèle d'occurrence des erreurs peut aussi être étendu; en effet, il est possible, sans remettre en question l'utilisation des chaînes de Markov (qui permettent de calculer facilement les critères choisis), de modéliser des rafales d'erreurs dans une fenêtre temporelle. Ce type de modèle est très intéressant pour modéliser des erreurs liées à des perturbations électromagnétiques dont les apparitions viennent d'une source qui peut avoir une durée de vie assez longue vis à vis des pas d'échantillonnage considérés. De tels modèles ont déjà été appliqués à l'étude [Navet *et al.*, 2000] et sont exploitables dans des modèles reposant sur des chaînes de Markov.

La généralisation de la méthode à des occurrences d'erreurs relevant de lois plus complexes reste possible mais nécessite l'utilisation de techniques probabilistes plus complexe. Il existe cependant une analogie entre le traitement des erreurs par un système bouclé et la notion de files d'attente qui permet de réutiliser certains résultats obtenus dans le cadre de cette théorie. De nombreux ouvrages sont consacrés à ce sujet (cf. par exemple [Kleinrock, 1975a; Kleinrock, 1975b; Gross and Harris, 1985; Mari and Schott, 2000]), et en particulier une des propriétés étudiée intéressante est liée à la saturation des files d'entrée des serveurs (comme [Schott and Louchard, 1997]). Si l'on considère le système régulé qui corrige les dépassements comme un serveur qui traite des erreurs, on remarque qu'il y a analogie entre la défaillance du système (vu comme un dépassement du seuil sur la grandeur controlée dû à une nouvelle erreur) et la saturation de la file d'attente. La difficulté réside alors dans la modélisation du temps de traitement du serveur qui doit correspondre à la dynamique du système physique et à sa capacité à corriger les dépassements. Une fois le modèle du serveur réalisé (serveur dont le temps de traitement est déterministe mais est lié à un état interne qui évolue en fonction des erreurs), on

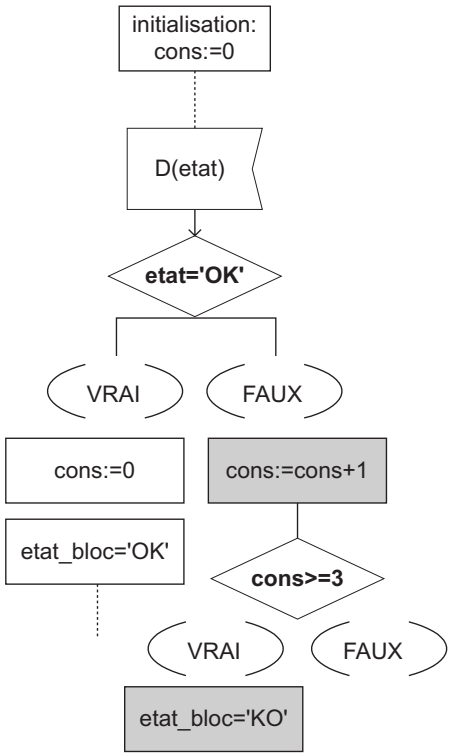


Figure 6.10: Exemple de la modélisation d'un mécanisme de tolérance aux fautes, le bloc reste dans un état non défaillant tant qu'il n'a pas reçu consécutivement 3 échantillons défaillants

peut ainsi appliquer les résultats existants. En effet, malgré la complexité du modèle de serveur, les résultats généraux développés sur la saturation des files d'entrée des serveurs (voir [Mari and Schott, 2000]) restent appliquables et l'on peut espérer obtenir des résultats équivalents à ceux obtenus dans la section précédente avec des modèles d'occurrences d'erreurs quelconques.

6.2.4 Extension de la fonction d'état d'erreur

6.2.4.1 Etude de la boucle ouverte en présence d'une erreur permanente

L'étude du système physique en boucle ouverte (cf. figure 6.11.) donne le nombre d'erreurs consécutives admissibles et une première idée de l'influence directe de l'erreur. Nous faisons l'hypothèse générale que la "pire" commande qui peut être appliquée au système (vis à vis de l'apparition de la défaillance) est une saturation de ses actionneurs. Nous ne considérons pas le phénomène de résonnance (une oscillation de forte amplitude) qui peut apparaître quand on applique à un système une commande périodique dont la fréquence correspond à la fréquence propre du système. Ce cas peut être étudié à part, une fois la fréquence de résonnance du système connue.

L'évolution du système en présence de rafales d'erreurs, permet de définir la granularité nécessaire à l'étude du système. En effet, si l'influence d'une erreur supplémentaire diminue énormément à proximité de la zone de défaillance (comme cela est possible dans le cas d'un système physique du premier ordre), il est nécessaire d'étudier le système avec une granularité très fine. Au contraire, dans le cas où l'influence relative d'une erreur est constante (cas de l'intégrateur), on peut utiliser une granularité plus grossière (cf. section 6.3).

Le système physique possède une certaine dynamique (évolution des grandeurs) en présence de cette erreur permanente, nous avons représenté (figure 6.12) les différentes possibilités de réponse du système. On remarque que, dans le cas des systèmes d'ordres supérieurs à un, le système tend vers un équilibre et qu'à l'approche de

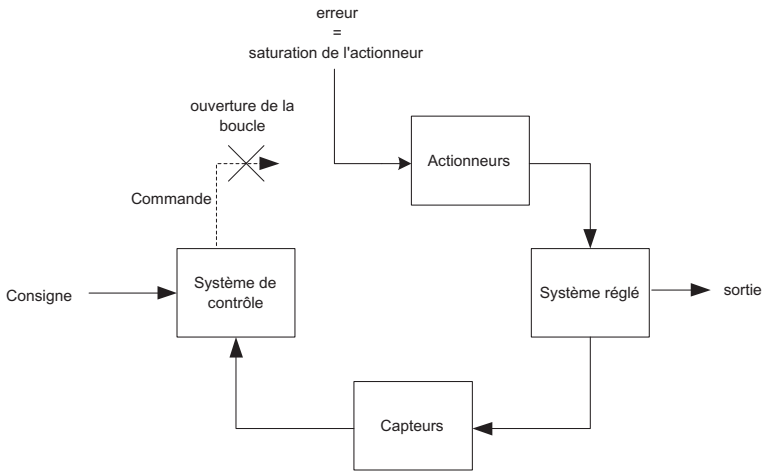


Figure 6.11: Mise en évidence de l'erreur, sous forme d'une saturation de l'actionneur

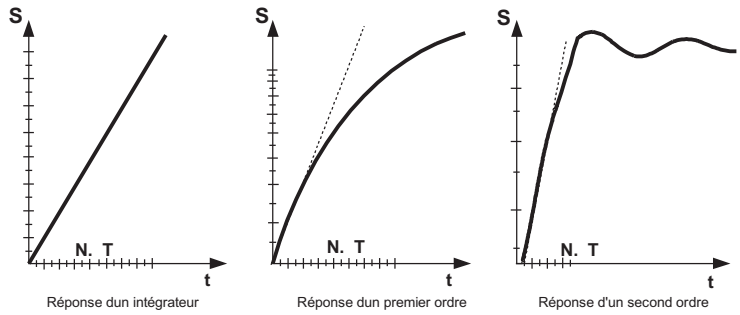


Figure 6.12: Réponse de divers système en présence d'une erreur permanente de commande

cet équilibre, l'influence d'une erreur supplémentaire diminue (voire s'annule dans le cas d'un second ordre). Le niveau de défaillance peut ne jamais être atteint (s'il est supérieur à l'équilibre), ou bien il faut un nombre d'erreurs consécutives très important (cas du premier ordre si le seuil de défaillance est légèrement en dessous de la réponse à l'infini en présence d'une erreur permanente).

6.2.4.2 Rejet de l'erreur en boucle fermée

La disparition de l'erreur se traduit par un fonctionnement correct de la boucle de contrôle. Le système est alors capable de corriger l'influence de l'erreur. Le schéma 6.13 représente la capacité du système bouclé à corriger une erreur.

Les systèmes bouclés ont souvent une capacité importante à corriger les erreurs. Un système bouclé d'ordre important (par exemple du deuxième ordre) corrige très rapidement une erreur mais surtout met un temps équivalent pour corriger une erreur ou une rafale d'erreurs. Il faut comprendre corriger comme la capacité à

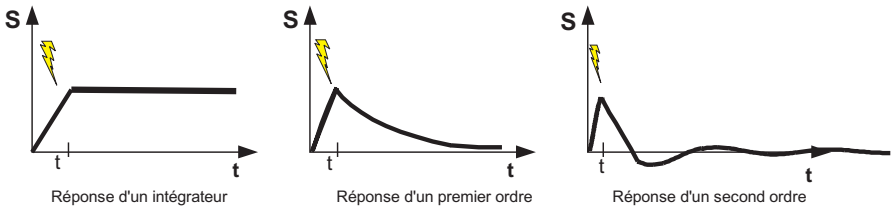


Figure 6.13: Correction de l'erreur par le système en boucle fermée

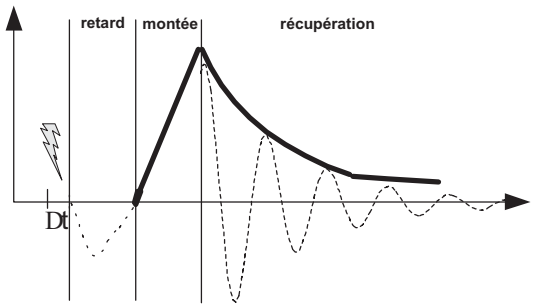


Figure 6.14: Evolution du système en présence d'une erreur de commande dans le cas général

ramener très rapidement la grandeur qui est en état d'erreur à la valeur souhaitée (consigne).

L'étude de la correction de l'erreur en boucle fermée fait apparaître que le système peut mettre plus ou moins de temps à corriger une erreur en fonction de sa provenance. En particulier, on distingue deux catégories de défaillance de l'information :

- Dans un cas, la commande a été correctement élaborée mais n'a pas été correctement appliquée sur le système. Dans ce cas, la défaillance se traduit par une évolution non désirée du système qui est corrigée par la boucle fermée.
- Dans l'autre cas, la loi de commande a produit une mauvaise information de commande car elle utilise des informations erronées.

Dans le deuxième cas, la correction de l'erreur peut prendre plus de temps car la loi de commande a un modèle interne erroné de l'évolution du système physique et utilisera des informations erronées pendant un certain temps. L'influence fine de ce phénomène dépend de l'algorithme de la loi de commande et de l'utilisation faite des valeurs passées. En général, les valeurs passées ont, dans le processus d'élaboration de la loi de commande, une importance moindre que les valeurs récentes. L'influence de l'erreur diminue donc avec le temps mais l'on peut voir apparaître un phénomène de retard entre l'occurrence d'une erreur et sa répercution sur l'évolution du système (cf. figure 6.14) .

Dans la section précédente, nous avons évalué des métriques de fiabilité d'un système élémentaire. Nous allons maintenant généraliser les techniques utilisées à des systèmes plus complexes.

6.2.4.3 Modélisation de la fonction d'état d'erreur

Pour modéliser le système, il faut connaître l'ordre de ce système au sens de l'automatique. Plus l'ordre d'un système est grand, plus il faut considérer de dérivées sur les grandeurs physiques pour modéliser son comportement.

La représentation d'état du système physique permet de représenter un système en faisant apparaître explicitement les différentes grandeurs qui interviennent dans sa dynamique (voir annexe A et chapitre 3).

$$\left\{ \begin{array}{l} X' = AX + BV \\ Y = CX + DV \end{array} \right\}$$

où X représente le vecteur d'états du système et Y la sortie du système (plusieurs sorties peuvent être considérées)

Dans le cas où le système est échantillonné (action périodique sur le système physique), on peut modéliser l'évolution du système aux instants d'échantillonnages définis de manière périodique. On a alors une représentation équivalente sous forme d'états, qui n'utilise plus le concept de dérivée mais d'échantillons:

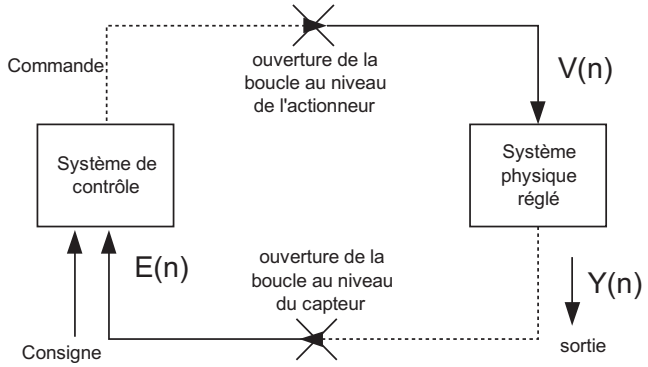


Figure 6.15: Différentes possibilités d'ouverture de la boucle et définition des entrées/ sorties sur le système

$$\left\{ \begin{array}{c} X(n+1) = AX(n) + BV(n) \\ Y(n+1) = CX(n+1) + DV(n+1) \end{array} \right\}$$

De la même maniére, on modélise la loi de commande, qui est un algorithme de calcul agissant sur les échantillons (cf. chapitre 3):

$$\left\{ \begin{array}{c} X_C(n+1) = A_C X_C(n) + B_C E(n) \\ V(n+1) = C_C X_C(n+1) + D_C E(n+1) \end{array} \right\}$$

où $E(n)$ représente les entrées de la loi de contrôle, $V(n)$ est la sortie de la loi de commande et correspond à la commande appliquée au système réglé, $X_C(n)$ est l'état interne de la loi de commande, qui correspond à la mémorisation de données passés nécessaires à l'élaboration de la loi de commande. Dans les systèmes les plus courants $E(n)$ est égale à la différence entre la consigne et la sortie du système.

Comme indiqué dans la section 6.2.4.2, la défaillance de l'information, même si elle conduit à la même défaillance de la commande, ne sera pas corrigée de la même façon si elle est apparue au niveau des actionneurs ou des capteurs (pour plus de details cf. [Vinter *et al.*, 2001; Askerdal *et al.*, 2002; Gäfvert *et al.*, 2003]) . L'injection de la défaillance de l'information, peut donc être faite au niveau de $V(n)$ (défaillance au niveau de l'actionneur) ou de $E(n)$ (la loi de commande travaille avec des données erronées). Dans le cas où la défaillance de l'information apparaît au niveau de l'actionneur, la modélisation de la faute peut être effectuée de manière équivalente à l'étude de la cuve d'eau (section 6.1.3.2).

En reprenant les notations définies précédemment ($\varepsilon_i = 0$ correspond à une défaillance de l'information), la modélisation du système en présence des défaillances de commande est simple, traduisant que l'entrée du système est soit donnée par la loi de commande en absence d'errreur, soit correspond à une saturation de l'actionneur en présence d'une défaillance de l'information de contrôle:

$$X(n+1) = AX(n) + B(\varepsilon_i V(n) + (1-\varepsilon_i) Sat_{actionneur})$$

($Sat_{actionneur}$représente la valeur de la saturation de l'actionneur, considérée comme la défaillance de l'information de contrôle)

L'évolution du système peut alors être modélisée en considérant l'état du système mais aussi, si nécessaire, l'état associé à la loi de commande noté X_c précédemment. Le calcul des critères de fiabilité présentés auparavant est toujours applicable, il suffit de projeter les états (à plusieurs dimensions) sur une seule dimension. Cette projection est toujours réalisable et peut être faite en appliquant un ordre lexicographique dur les différentes composantes de l'état.

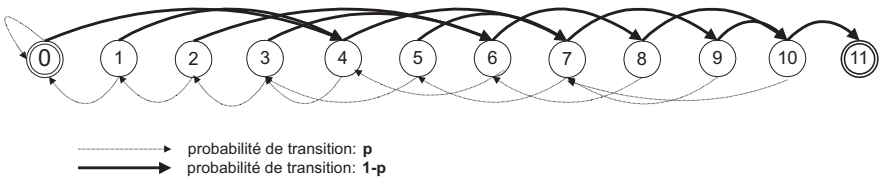


Figure 6.16: Modélisation sous forme de chaîne de Markov d'un système du premier ordre en boucle ouverte et boucle fermée

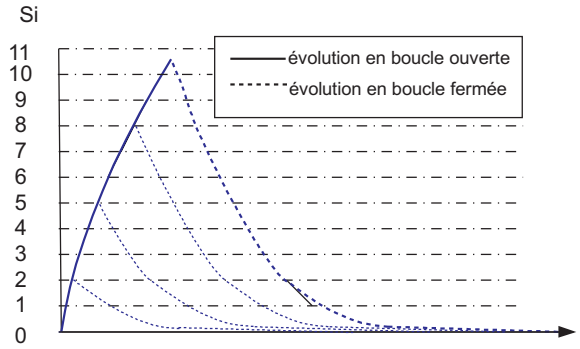


Figure 6.17: Evolution d'un premier ordre en fonction de l'état atteint en boucle ouverte (saturation de la commande) et en boucle fermée (fonctionnement correct)

Le premier exemple correspond à un système du premier ordre, la modélisation sous forme de chaîne de Markov (cf. figure 6.16) est équivalente à celle de la cuve d'eau mais l'évolution du système dépend de la valeur de la grandeur. La figure 6.17 présente quelques trajectoires possibles du système, on remarque que la trajectoire du système en boucle fermée ne dépend que de la valeur de la grandeur (propriété de superposition des différentes trajectoires).

Le deuxième exemple présente un cas qui correspond plus au cas général, où l'évolution du système dépend de plusieurs grandeurs d'états. La seule défaillance considérée correspond à un dépassement sur la grandeur de sortie (valeur 2.25, ce qui correspond à 11 avec la discrétisation choisie). Le schéma 6.18 présente le système considéré, composé d'un système physique du deuxième ordre et d'un correcteur proportionnel. En considérant un seul état de départ possible (O, O), la figure 6.19 montre la modélisation sous forme de chaîne de Markov de l'évolution de l'erreur en présence de défaillances de la loi de commande. Dans cet exemple, seuls les états accessibles à partir de l'état de départ considéré sont représentés. On peut, à partir de la chaîne de Markov modélisant le système, calculer les critères de fiabilité présentés précédemment.

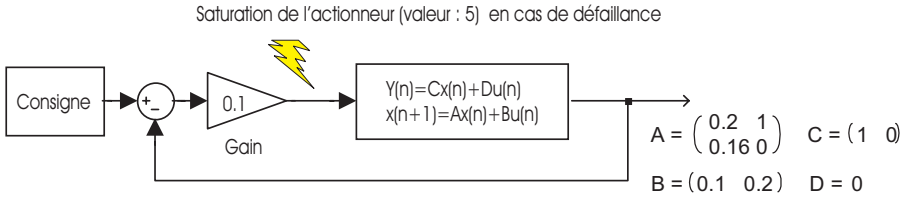


Figure 6.18: Exemple de système régulé composé d'un système réglé du deuxième ordre et d'une loi de contrôle élémentaire correspondant à un correcteur proportionnel

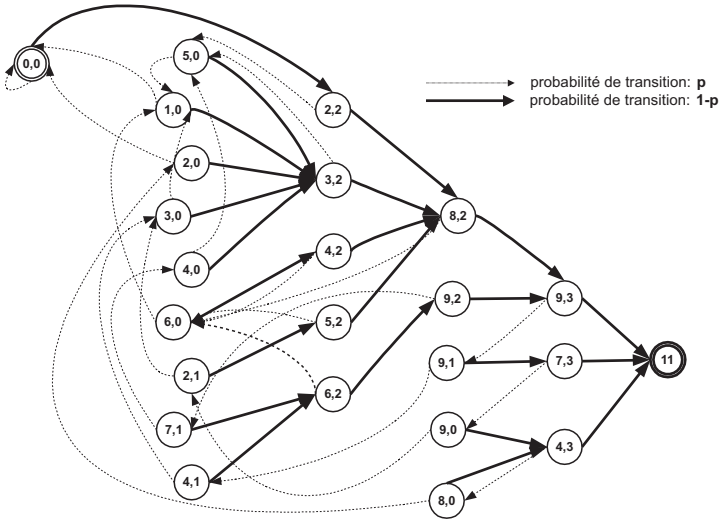


Figure 6.19: Diagramme de transition où (0,0) est le seul état initial considéré et 11 est l'état absorbant (défaillance du système physique) correspondant à l'exemple d'un système du second ordre

6.3 Extension de la fonction d'état d'erreur à des systèmes sensibles aux retards

L'étude précédente repose sur la notion de système échantillonné, c'est-à-dire que le contrôle est appliqué sur le système de manière périodique. Ainsi la notion de défaillance du système physique est liée à l'existence de défaillances sporadiques sur le contenu des informations. La même étude peut être adaptée dans le cas où la défaillance de l'information de commande correspond à un retard. Dans les systèmes commandés en tout ou rien (par exemple en éléctronique de puissance) qui doivent changer la commande très fréquemment pour empêcher la destruction du système le phénomène peut se produire. L'information de commande ne porte pas de valeurs, seules les dates d'application du changement de consigne permettent le contrôle du système. Dans ce cas la seule défaillance possible est une faute temporelle sur ces dates.

Considérons à nouveau notre cuve d'eau, on imagine que la commande génère des événements de vidange qui impliquent une vidange jusqu'au début la période suivante où a lieu automatiquement l'arrêt de la vidange. On considère que le système peut idéalement vidanger 50% du temps d'un cycle (de période T) pour un ordre de vidange arrivé à la date $t_k + \frac{T}{2}$. La capacité de vidange maximale correspondante est de H unités pour un cycle.

Si l'ordre est appliqué avec un retard τ_k (par hypothèse $\tau_k < \frac{T}{2}$), la capacité maximale de vidange sur le cycle est alors de $H \cdot (1 - \frac{\tau_k}{\frac{T}{2}})$ (voir figure 6.20).Avec un remplissage constant de R sur un cycle (perturbation due au fonctionnement), le système ne peut vidanger la totalité de la perturbation que si $H \cdot (1 - \frac{\tau_k}{\frac{T}{2}}) > R$, sinon le système a augmenté au cours du cycle de $(R - H \cdot (1 - \frac{\tau_k}{\frac{T}{2}}))$. En présence d'un tel remplissage, l'évolution du système apparaît en "dents de scie" (cf. figure 6.21). Il est possible de discrétiser l'évolution du système et de considérer uniquement les valeurs maximales s_{max} et minimales s_{min} atteintes au cours de la période.

On note :

$$\tau_{lim} = (1 - \frac{R}{H}) \cdot \frac{T}{2}$$

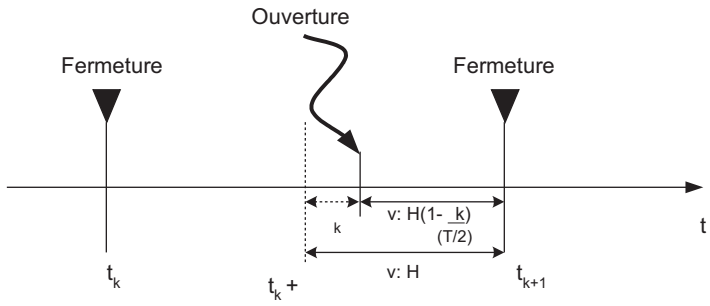


Figure 6.20: Mise en évidence de la perte de capacité de vidange d'un système tout ou rien en présence de retard

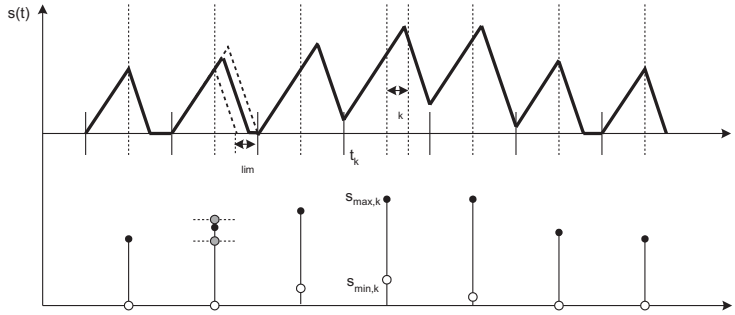


Figure 6.21: Mise en évidence de la perte de capacité de vidange d'un système tout ou rien en présence de retard

où τ_{lim} correspond au retard limite tel que le système ne peut plus se vidanger totalement au cours du cycle. Il est possible d'écrire simplement l'équation d'évolution en présence de retard:

$$s_{min,i} = Pos(s_{min,i-1} + \frac{H}{\frac{T}{2}}(\tau_{lim} - \tau_i)) \cdot \delta(s_{min,i-1}))$$

avec

$$\begin{aligned} \delta(x) &= 0 \,\text{si}\, x = 0 \\ &= 1 \,\text{si}\, x \neq 0 \end{aligned}$$

qui sert à définir que l'on ne vidange pas un système déjà vide, et

$$\begin{aligned} Pos(x) &= x \,\text{si}\, x > 0 \\ &= 0 \,\text{si}\, x \leq 0 \end{aligned}$$

qui sert à éviter de vidanger sur un cycle plus que le contenu de la cuve.

Cette équation d'évolution en présence de retard est de la même forme que dans le cas des pertes (cf. section 6.3). L'étude précédente peut donc être adapté au cas des retards. En particulier si l'on discrétise les différents retards qui peuvent apparaître, on peut modéliser le système sous forme de chaîne de Markov et en déduire des propriétés sur la sûreté de fonctionnement du système.

6.4 Conclusion

Nous avons présenté dans ce chapitre, une méthodologie d'étude de la sûreté de fonctionnement des applications de contrôle-commande. En effet, les systèmes régulés présentent la particularité de pouvoir supporter un grand nombre de défaillance de l'architecture support des algorithmes de commande. Les études usuelles de sûreté de fonctionnement des architectures informatisés ne sont donc plus adaptés et de nouvelles techniques doivent être développés (cf. [Gäfvert *et al.*, 2003]) . Nous formalisons ici le lien entre l'apparition de défaillances de l'architecture support et l'évolution du système régulé. Il devient ainsi possible de quantifier les risques d'apparition de défaillances majeures du système régulé vis à vis des défaillances des informations de commandes. Pour cela, nous avons introduit la notion de fonction d'état d'erreur qui permet de lier l'état du système et la défaillance des informations de commande. La connaissance de cette fonction permet à la fois de calculer des métriques de fiabilité du système régulé (cf. section 6.1.3.2 pour un exemple détaillé) mais aussi de proposer des mécanismes de tolérance aux fautes adaptés aux spécificités des applications de contrôle-commande (cf. conclusion).

Chapter 7

Techniques d'ordonnancement et prévisibilité

La spécification de lois de commande intégrant les paramètres temporels de son implantation nécessite de pouvoir caractériser à l'avance et avec précision les activités réalisant cette commande, notamment les temps de réponse. L'estimation des temps de réponse, appelés aussi temps de traitement par la suite, repose à la fois sur la connaissance de la charge de travail à effectuer et des règles d'accès aux ressources (CPU et réseau) c'est-à-dire la politique d'ordonnancement (cf 4.4). Le multiplexage des ressources dans le cadre des applications temps réel est assuré à l'aide par un ordonnanceur qui décide de l'activité (messages ou tâches) qui prend le contrôle de la ressource et de la durée de cette prise de contrôle.

Le plus souvent les systèmes temps réel utilisent des ordonnanceurs à base de priorité fixe (par exemple le protocole *CAN* [Paret, 1996] pour les réseaux ou la politique *Rate Monotonic* [Klein *et al.*, 1994] pour l'ordonnancement de tâche). L'ordonnanceur donne la ressource à l'activité la plus prioritaire (en permettant ou non la préemption de la ressource avant la fin de l'activité en cours de traitement).

L'ordonnancement temps réel repose sur la recherche de la garantie a priori du respect des échéances, c'est à dire la capacité de borner les temps de réponse. De nombreuses techniques permettent donc de calculer les pires temps de réponses ou des bornes pour un très grand nombre de politiques d'ordonnancement. Cependant, nous avons montré dans la section 5.4.2.5 que la connaissance d'une borne n'est pas suffisante si l'on veut compenser un temps de traitement non-nul.

Les techniques d'ordonnancement usuelles ne permettent pas de déterminer les temps de réponses avec précision. En effet, il faut connaître toutes les activités plus prioritaires (et moins prioritaires si la politique est non-préemptive) qui peuvent interférer sur l'activité considérée. Une grande incertitude existe car il suffit de peu de choses (par exemple un temps d'exécution moins long que prévu ou une gigue par exemple) pour qu'une activité s'exécute ou non avant celle étudiée. Il est donc très délicat de déterminer à l'avance les temps de traitements.

Dans ce chapitre, nous proposons tout d'abord en section 7.1 un modèle de tâches qui en contraignant l'exécution induit plus de prévisibilité. Dans le même objectif, nous étudions en section 7.2.1.2 des politiques d'ordonnancement à taux constant qui permettent de rendre les temps de traitement prévisibles. Si les politiques à taux constant offrent la meilleur prévisibilité possible par construction, elles ont cependant comme important désavantage de limiter la charge du système qu'il est possible d'accepter (cf. test d'acceptation associé **??**) et conduisent donc à sa sous-utilisation. Nous proposons des techniques de réservation qui permettent une utilisation plus importante des ressources systèmes avec des temps de traitement toujours prévisibles. Leur désavantage est qu'il est possible qu'un certain nombre d'instances de tâches ne soient pas acceptées et donc, éventuellement, ne respectent pas leur échéance si elles sont exécutées. Dans le cadre des systèmes de

contrôle-commande, ce problème n'est cependant pas rédhibitoire car, comme nous l'avons montré au chapitre 5, la qualité d'un système n'est généralement que très faiblement diminuée par la présence sporadique de temps de traitement exceptionnellement longs ou de traitements manquants. La présentation des différentes politiques de réservation et leur étude font l'objet de la section 7.2.2.

7.1 Tâches à gabarits

De manière à augmenter la connaissance sur les dates de fin d'exécution, il est possible d'imposer des contraintes sur l'évolution du temps CPU alloué aux tâches dans le temps. Il faut pouvoir garantir le respect de ces propriétés par une politique d'ordonnancement adaptée. Nous proposons, dans cette section, d'associer aux tâches un ensemble d'échéances intermédiaires associées à la trajectoire du temps CPU alloué à la tâche. Le modèle de tâches correspondant est appelé "tâche à gabarits". Ce nouveau modèle de tâches nécessitera une modification des automates de fonctionnement associés à l'ordonnancement des tâches.

Comme nous le verrons, la politique EDF est optimale pour les tâches à gabarits, c'est-à-dire que si EDF ne permet pas de respecter les échéances d'une tâche à gabarit, aucune autre politique ne le peut. La politique EDF permet en outre de procéder hors ligne ou en ligne à des tests d'ordonnançabilité de complexité réduite [Stankovic *et al.*, 1998]. Il est ainsi possible de prouver le respect des contraintes temporelles des différentes tâches.

7.1.1 Définition

Dans le modèle de tâches temps réel classiques, on associe à chaque tâche, outre une date d'activation, une échéance sur la date de fin de traitement et un WCET qui est le plus grand temps CPU qui puisse être utilisé par la tâche. On peut en déduire un gabarit, c'est-à-dire une enveloppe, qui doit contenir toute trajectoire CPU associée à la tâche. Le schéma 7.2 présente le gabarit associé à une tâche temps réel de définition classique avec :

- une date de début A,
- une échéance D,
- un temps d'exécution C .

Une trajectoire définit l'évolution du temps CPU alloué à la tâche. Dans le cas où l'activité s'exécute sur un système monoprocesseur où le processeur fonctionne à vitesse constante, la trajectoire est une succession de zone où le CPU est alloué à 100% (correspondant à une droite de pente 1) ou à 0% (pente 0). Il est bien sur possible de borner le temps de traitement mais l'indétermination est grande car la date de fin dépend fortement des autres activités présentes (cf. 7.1).

Nous proposons d'encadrer l'évolution de l'exécution d'une tâche dans un "gabarit" de manière à accroître la prévisibilité sur les dates de fin de traitement. Pour cela, on associe à une tâche, à la fois un profil d'arrivée de travail (ou de charge) $C(t)$ et un profil d'échéance $D(t)$. De manière à simplifier la mise en place de mécanismes garantissant le respect de ses profils, nous ne considérons dans ce document que le cas où ses profils sont des "fonctions en escalier", c'est-à-dire constants par morceaux. Le profil $C(t)$ est de la forme :

$$C(t) = \sum_{i=1}^{m_C} C_i^{max} \cdot 1_{A_i \leq t < A_{i+1}}$$

- C_i^{max} représente la quantité maximale de travail qui peut être executée avant la date A_{i+1},
- A_1 correspond à la date de début d'exécution de la tâche avant laquelle aucune charge n'est demandée,

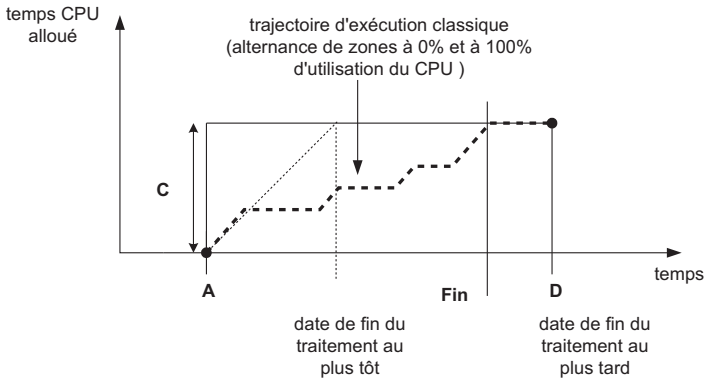


Figure 7.1: Bornes sur la date de fin d'une activité "temps réel"

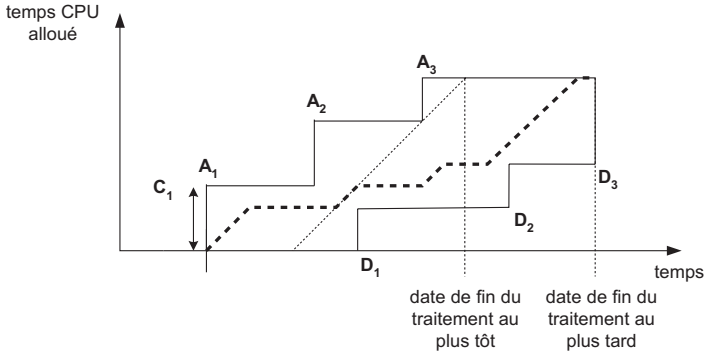


Figure 7.2: Définition d'un gabarit à partir des profils de charge et d'échéance

- A_{m_c+1} correspond à l'échéance finale de la tâche ($A_{m_c+1} = D_{m_D}$ - cf. suite),

De la même manière le profil $D(t)$ est de la forme :

$$D(t) = \sum_{i=1}^{m_D} C_i^{min} \cdot 1_{D_{i-1} \leq t < D_i}$$

- C_i^{min} représente la quantité de travail minimale qui doit être exécutée avant la date D_i,
- D_0 correspond à la date de début d'exécution de la tâche avant laquelle travail n'est demandée, c'est à dire A_1. On a donc par définition $C_0^{min} = 0$.

La mise en place d'un gabarit permet de donner plus de déterminisme sur l'évolution de l'exécution d'une tâche. En particulier ce gabarit peut réduire la taille de l'intervalle qui contiendra la date de fin d'exécution de l'activité. Elle reste cependant fortement dépendante des autres tâches en compétition pour la ressource. Il faut noter que l'utilisation de ces gabarits est d'autant plus intéressante si d'autres dates significatives existent à l'intérieur de l'activité car il devient possible de donner des garanties sur dates par un choix approprié de gabarit . L'utilisation de gabarits permet alors d'augmenter dans ce cas la prévisibilité sur l'ensemble du traitement.

7.1.2 Adaptation de l'ordonnanceur

L'utilisation de tâches gabarits va nécessiter une modification l'automate de fonctionnement de l'ordonnanceur de tâches. Dans un système classique, une tâche se trouve dans un de ces six états (cf. la figure 7.3 pour une représentation graphique):

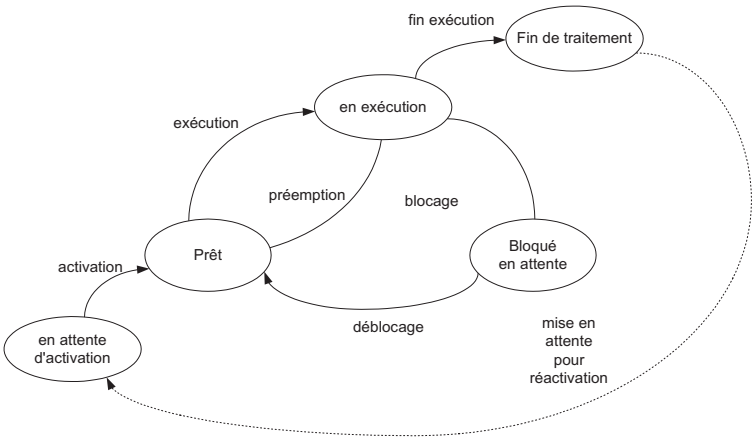


Figure 7.3: Automate de fonctionnement d'un ordonnanceur classique

- *Prêt* : l'ensemble des tâches qui ne sont pas en attente d'activation ou d'une ressource bloquante sont prêtes à accéder à la ressource CPU. C'est à l'ordonnanceur de décider à quelle tâche va être affecté le processeur.
- *En exécution* : la tâche qui gagne le contrôle du CPU passe en exécution. Elle ne quittera ce mode que si elle se termine, se retrouve en attente bloquante ou s'il elle se voit retirer le CPU par l'ordonnanceur.
- *Fin de traitement* : une fois terminée, la tâche passe en fin de traitement, elle disparaît du système où se met en attente d'une phase de réveil (tâche périodique).
- *Bloquée en attente* : au cours de son éxecution une tâche peut se retrouver en attente d'une ressource, elle se met donc en attente et libère le CPU.
- *En attente d'exécution* : ce cas particulier d'attente correspond à une attente d'activation pour une tâche recurrente comme une tâche périodique.

La charge de travail associée à la tâche n'est généralement pas connue de l'ordonnanceur. La charge de travail C, qui est le temps CPU nécessaire pour passer de l'état *Actif* à l'état *Fin de traitement*, peut cependant être estimé à l'avance. La connaissance de l'échéance des tâches n'est généralement pas nécessaire à l'ordonnanceur avec pour exceptions notables les politiques EDF ou Least Laxity First (LLF). Dans le cas des ordonnanceurs à priorités fixes, les échéances servent hors ligne à fixer des priorités.

Les tâches à gabarits que nous proposons nécessitent de pouvoir bloquer l'exécution d'une tâche avant sa fin. Soit $c(t)$ l'intégrale du temps CPU alloué à la tâche à l'instant t. Par définition des gabarits, il est imposé que

$$C(t) \geq c(t) \geq D(t)$$

De manière à garantir cette propriété, il faut :

- associer les échéances à l'évolution $c(t)$,
- interrompre l'éxecution de la tâche dès que $c(t) = C(t)$.

L'automate représenté sur la figure 7.4 intègre les modifications nécessaires. Les échéances sont conditionnées par la charge minimal à dépasser. Avec les notations précédentes, l'échéance initiale $D = D_1$, dès que $c(t) >$

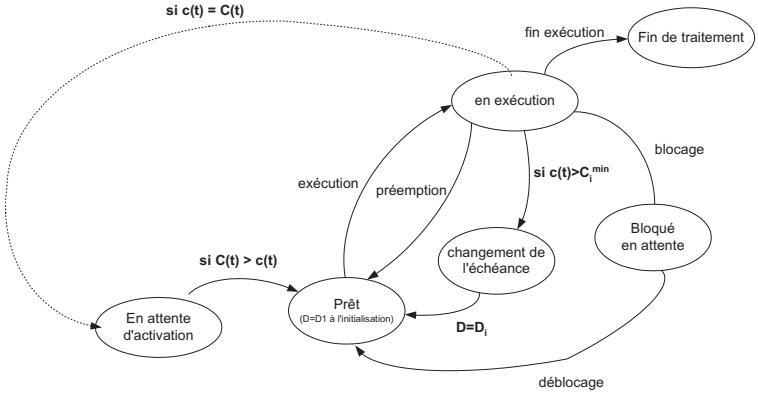


Figure 7.4: Prise en compte du gabarit dans la définition de l'état des tâches

c^i_{min}, l'échéance pour D_{i-1}est respectée, la nouvelle échéance à respecter est D_i avec une charge de travail c^{i+1}_{min}.

La tâche ne doit pas dépasser son gabarit, si le travail reçu $c(t)$ atteint la limite supérieure du gabarit $C(t)$, la tâche passe en attente de réactivation, c'est à dire dès que $C(t) > c(t)$. Elle est alors prête à accéder de nouveau au CPU.

7.1.3 Optimalité de la politique EDF

L'utilisation d'un ordonnanceur adapté permet de garantir un fonctionnement cohérent du système. Cela ne suffit cependant pas pour garantir qu'un ensemble de tâches à gabarit soit ordonnançable. L'utilisation de la politique EDF préemptive (avec la stratégie FIFO en cas d'égalité sur les échéances) comme politique d'ordonnancement sous-jacente est très intéressante car il est possible de montrer que cette politique est optimale, c'est à dire que si elle ne permet pas le respect de l'ensemble des gabarits des tâches, aucune autre politique ne le peut.

On considère un ensemble de tâches à gabarit $\{T_i\}$ auxquelles sont associés des gabarits $\{C_i(t), D_i(t)\}$. Il est bien connu que EDF est optimale dans le cas des tâches temps réel classiques préemptibles (cf. [Stankovic *et al.*, 1998]). Avec des tâches à gabarit, l'échéance peut évoluer au cours du temps. A chaque fois qu'une charge minimale, pour une échéance donnée est atteinte on passe à l'échéance suivante (cf 7.4). La politique EDF est toujours utilisable sans modification car les échéances sont toujours connues et il suffit de donner le CPU à la tâche dont l'échéance est la plus proche.

Pour prouver l'optimalité de la politique EDF pour les tâches à gabarit, une première étape est de montrer qu'il est possible de découper une tâche à gabarit en un ensemble de tâches "classiques" temps réel, c'est-à-dire avec une seule date d'arrivée du travail et une seule échéance.

7.1.3.1 Découpage de la tâche à gabarit en un ensemble de tâches

Une tâche à gabarit peut être découpée en sous-tâches de manière à ce que chaque sous tâche possède uniquement une échéance et une activation. Le schéma 7.5 présente ce découpage; du gabarit de la tâche T de manière on obtient une un ensemble de tâches $T_{i,j}$ où $T_{i,j}$ est défini par :

- une date d'activation A_i. Comme la fonction d'arrivée de travail C qui définit un gabarit est continue à droite, A_i est tel que $C(A_i) = a$, $C(A_i^+) = b$ avec $b > a$.
- une échéance D_j,
- une charge $C_{i,j}$.

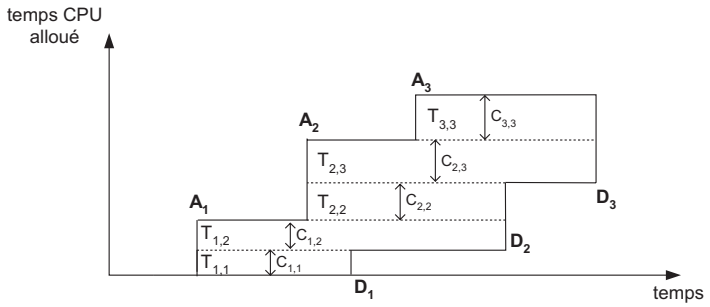


Figure 7.5: Découpage d'une tâche à gabarit

Dans cette définition i identifie une date d'arrivée de travail et j une date d'échéance, la première tâche créée par découpage étant $T_{1,1}$. Pour déterminer la quantité de travail de chaque sous-tâche (cf. figure 7.5), on applique les règles suivantes:

- Si $C_i^{max} > C_j^{min}$ alors $C_{i,j} = C_j^{min}$ et la tâche suivante à créer est de la forme $T_{i,j+1}$,
- Si $C_i^{max} > C_j^{min}$ alors $C_{i,j} = C_i^{max}$ et la tâche suivante à créer est de la forme $T_{i+1,j}$,
- Si $C_i^{max} = C_j^{min}$ alors $C_{i,j} = C_j^{min}$ et la tâche suivante à créer est de la forme $T_{i+1,j+1}$ ou nous sommes dans la situation où l'ensemble des tâches a déjà été créé pour couvrir le gabarit.

7.1.3.2 Equivalence des trajectoires d'exécution entre les deux modèles

On note $\mathcal{T}_G$ l'ensemble des trajectoires faisables avec une tâche définie selon son gabarit et l'on note $\mathcal{T}_D$ l'ensemble des trajectoires faisables avec la même tâche à gabarit découpée en de multiples tâches temps réel classiques. L'étape suivante de la preuve est de prouver que $\mathcal{T}_G \subseteq \mathcal{T}_D$. Ensuite nous montrerons que si la trajectoire EDF est faisable pour l'ensemble des tâches $T_{i,j}$ alors c'est une trajectoire faisable de la tâche à gabarit T. En raisonnant par contradiction, nous en déduirons que la trajectoire EDF est optimale pour la faisabilité pour les tâches à gabarit.

Etape 1: $\mathcal{T}_G \subseteq \mathcal{T}_D$. Considérons une trajectoire d'exécution faisable $c(t)$ d'une tâche à gabarit. Découpons cette trajectoire en les morceaux de trajectoires correspondant à chacun des $T_{i,j}$. On note $c_{i,j}(t)$ l'évolution du travail CPU réalisé exclusivement pour la tâche $T_{i,j}$ (i.e. $c_{i,j}(A_i) = 0$). Si la trajectoire $c(t)$ est faisable, c'est-à-dire respecte les contraintes du gabarit, alors pour chaque $T_{i,j}$, on a $c_{i,j}(D_j) = C_{i,j}$, donc la faisabilité est préservée si l'on découpe la tâche à gabarits en tâches classiques. On a donc nécessairement $\mathcal{T}_G \subseteq \mathcal{T}_D$.

Etape 2: si EDF est faisable pour l'ensemble des $T_{i,j}$ alors c'est une trajectoire faisable de la tâche à gabarit. Nous considérons ici la politique EDF où les égalités sont tranchées selon la stratégie FIFO. Remarquons qu'avec notre technique de découpage en tâches, il est impossible d'avoir une égalité à la fois sur la date d'arrivée et la date d'échéance (sinon les deux tâches n'en formeraient qu'une); la politique est donc non-ambigue. Sous cette politique et avec notre découpage, une tâche $T_{i,j}$ ne commence pas tant que la tâche qui la précéde dans le gabarit ne s'est pas entièrement terminée. L'ensemble des trajectoires des tâches $T_{i,j}$ permet donc d'obtenir (par construction de ces tâches par découpage) une trajectoire pour la tâche à gabarit T.

Etape 3: EDF est optimale. S'il existait une politique optimale autre que EDF sous le modèle à gabarit, alors cette politique serait également optimale sous le modèle de tâches classique car $\mathcal{T}_G \subseteq \mathcal{T}_D$. Or EDF est optimale avec le modèle de tâches classiques donc il ne peut exister mieux pour la faisabilité que EDF pour les tâches à gabarit. Comme l'ordonnancement sous EDF induit une trajectoire faisable pour les tâches à gabarit (étape 2), EDF est optimale pour ce modèle de tâche.

7.1.3.3 Test de faisabilité d'un ensemble de tâches à gabarit.

Le test de faisabilité d'EDF est conservé, il suffit de se ramener au découpage des tâches et d'appliquer les tests de faisbilité sur cette ensemble de tâches. Le test de faisabilité le plus connu [Stankovic *et al.*, 1998] est

$$\sum_i \frac{C_i}{T_i} \leq 1.$$

Ce test ne s'applique que dans le cas des tâches périodiques à échéances sur requêtes. Dans le cas des tâches à gabarit, on quitte le cadre de l'échéance sur requête car plusieurs échéances sont définies. Il faut donc utiliser le test plus général basé sur le facteur de charge U (cf. [Stankovic *et al.*, 1998]):

$$u = max(U(t_1, t_2)) \leq 1$$

Ce test est détaillé dans l'annexe C.

7.2 Techniques d'ordonnancement fluide

Dans la suite, nous considèrerons la ressource fluide: elle peut être partagée simultanément par plusieurs clients avec des niveaux de charge différents et sans granularité minimum de la charge. En pratique, il existe bien une granularité minimum pour l'accés à la ressource et donc un écart par rapport au cas fluide. De nombreuses études ont été faites quant à la définition de politiques d'ordonnancement approchant le cas fluide : on peut citer en particulier EEVDF (Earliest Eligible Virtual Deadline First) et les techniques de "Lottery Scheduling" [Regehr, 2003; Stoica *et al.*, 1996b; Waldspurger, 1995] dans le cadre de l'ordonnancement CPU et WFQ (Weighted-Fair Queing) [Demers *et al.*, 1990; Parekh and Gallager, 1993; Mowbray *et al.*, 1998] dans les réseaux. Ces politiques donnent des bornes sur l'écart par rapport au cas fluide et cet écart pourra être considéré dans la définition de la requête ou lors du test d'acceptation de la requête.

7.2.1 Politiques à poids et à taux

Nous introduisons dans cette partie les principales politiques d'ordonnancement fluide, c'est-à-dire les politiques à taux et à poids. Les politiques à taux sont celles qui nous intéressent tout particulièrement mais la présentation des politiques à poids est nécessaire à leur compréhension.

7.2.1.1 Politique à poids

Les politiques à poids sont d'usages courants dans le domaine des réseaux, en particulier pour les applications multimédias où l'utilisation des ressources se fait en mode "best effort" et où l'on veut garantir une certaine équité dans le partage des ressources. Dans le cas des politiques à poids, chaque activité A_i possède un poids w_i. La ressource (CPU ou réseau) est partagée de manière fluide (cf 7.2) entre les activités concurrentes proportionnellement à leur poids relatif. Dans un intervalle de temps où il n'y a aucune modification des poids relatifs des activités, cela revient à donner un taux constant de la ressource à chaque activité.

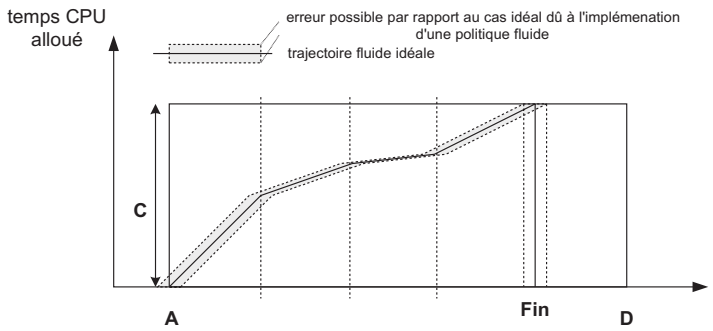


Figure 7.6: Trajectoire d'une tâche à poids

$$t_i = \frac{w_i}{\sum_k w_k} \tag{7.1}$$

La trajectoire d'exécution d'une activité est donc composée par un ensemble continu de droite (cf schéma 7.6) où chaque changement de pente correspond:

- à la disparition d'une tâche (fin d'exécution),
- à l'apparition ou l'activation d'une tâche,
- ou éventuellement, si cela est autorisé par la politique, au changement du poids d'une tâche.

Dans le cas du temps réel, l'utilisation des politiques à poids est peu répandue car il est difficile de faire le lien entre les poids choisis et le respect des échéances.

7.2.1.2 Politique à taux constant

Pour remédier à cette difficulté, il a été proposé de spécifier un taux d'utilisation minimum et non un poids pour le partage de ressources temps réel. Dans ce cas, chaque tâche temps réel se voit attribuer un taux garanti d'utilisation de la ressource qui reste constant. Pour permettre la cohabition entre tâche temps réel et tâche en best effort, il a été proposé d'implémenter cette politique à taux constant sur la base d'un ordonnanceur à poids . Dans cette situation, il suffit de calculer le poids de chaque tâche temps réel de manière à garantir un taux d'utilisation du CPU. Soit f_i, le taux fixé pour une tâche temps réels et w_i le poids de la tâche à instant t. Dans une zone où tout les poids sont fixes et où il n'y a pas de changement dans l'ensemble des tâches actives, le taux constant t_i, d'une tâche de poids w_i est $t_i = \frac{w_i}{\sum_k w_k}$ (cf 7.1). Inversement si on veut imposer un taux (soit $t_i = f_i$), il faut fixer le poids de la tâche i en fonction du poids de l'ensemble des tâches actives ($\sum_k w_k$). On en déduit le poids à donner à la tâche i :

$$w_i = \frac{t_i}{1 - t_i} \cdot \frac{\sum_{k \neq i} w_k}{} \tag{7.2}$$

L'adaptation du poids en fonction des modifications des tâches actives, permet de garantir le taux de fonctionnement souhaité f_i. Le détail de l'utilisation de cette technique peut être trouvé en particulier dans des études de Stoica et al ([Stoica *et al.*, 1996a; Goyal and Vin, 1997]). Cette implantation sur la base d'un ordonnanceur à poids ne se justifie que dans le cas où il existe un trafic à poids non temps réels permanent. Si ce n'est pas le cas, il suffit de fixer comme poids $w_i = f_i$ pour chaque tâche temps réel. Enfin les études menés en Italie autour de Buttazo ont montrés l'intéret de ce type de politique dans le cadre des applications de contrôle-commande (cf. [Abeni *et al.*,])

L'intérêt de cette technique est de permettre d'ordonnancer conjointement des tâches temps réel et des tâches sans contraintes de temps auxquelles on associe des poids. Pour cela Stoica et al proposent de fixer le taux pour les tâches temps réel à $\frac{C_i}{T_i}$. L'utilisation d'un tel taux avait déjà été proposée pour le multimédia avec un ordonnancement non-conservatif. Cela signifie que l'on fournit à chaque tâche exactement le taux demandé. Le CPU peut donc être oisif même si théoriquement du travail doit être effectué. Dans le cas d'utilisation de tâches périodiques à échéance sur requête, cette politique est optimale (sous hypothèse de fluidité) (cf. [Mercer *et al.*, 1994]).

7.2.1.3 Quelques résultats d'optimalité des politiques à taux constants

Nous rappelons dans ce paragraphe certains résultats concernant l'optimalité des politiques à taux constants publiés dans [Mercer *et al.*, 1994] et [Stoica *et al.*, 1996a; Goyal and Vin, 1997].

Tâches périodiques à échéances sur requête Soit un ensemble de tâches temps réel périodiques définies par :

- leur charge C_i,
- leur période T_i (l'échéance relative $\overline{D_i} = T_i$).

On donne à chaque tâche, le taux $f_i = \frac{C_i}{T_i}$. Dans le cas non-conservatif (*idling*), par définition toutes les instances j de la tâche i ont la même trajectoire d'exécution linéaire entre A_i et D_i:

$$c_i^j(t) = f_i \cdot t \tag{7.3}$$

Si le système est faisable, toutes les instances de tâches se terminent à D_i ($c_i^j(\overline{D_i}) = f_i \cdot T_i = C_i$) .

Dans le cas conservatif, la trajectoire d'exécution d'une instance n'est plus forcément constante (en particulier s'il n'y a qu'une tâche temps réel active, elle prend le contrôle du CPU à 100%). Cela ne remet en cause la faisabilité comme ce pourrait être le cas dans des politiques non-preemptives : dans le cas conservatif comme non conservatif, la politique à taux constants garantie est optimale pour les tâches périodiques à échéances sur requêtes et la condition de faisabilité nécessaire et suffisante est [Mercer *et al.*, 1994]:

$$\sum_i f_i \leq 1$$

avec $f_i = \frac{C_i}{T_i}$. Cette politique est donc dans ce cas aussi bonne que EDF et meilleure que les politiques à priorité fixe du point de vue faisabilité tout en donnant des garanties sur les dates de fin d'exécution.

Cas périodique avec échéance quelconque Nous montrons sur un contre-exemple que dans le cas d'échéances quelconques, l'optimalité n'est pas préservée. Soit un ensemble de tâches temps réel périodiques définies par :

- une charge C_i,
- une date d'activation initiale A_0,
- une période T_i,
- une échéance relative $\overline{D_i}$.

Nous allons montrer qu'il existe au moins une configuration ordonnançable sous EDF et non ordonnançable quels que soient les taux choisis. Considèrons deux tâches périodiques: T1 ($C_1 = 1, A_0 = 0, T_1 = 2, \overline{D_1} = 1$) et T2 ($C_2 = 1, A_0 = 0, T_2 = 2, \overline{D_2} = 2$). Cet ensemble est ordonnançable sous EDF (exécution de T1 puis de T2 etc ...). Il n'existe qu'un ordonnancement possible pour cette configuration, il faut que la tâche 1 prenne 100% du CPU dès son activation. Si on fixe un taux, ou un poids (non nul pour la tâche 2), la tâche 1 ne peut pas avoir 100% du CPU dès son activation et elle ne peut donc pas respecter son échéance. Les politiques à taux ou à poids ne sont donc pas optimales dans ce cas. Si on fixe un taux de $f_i = \frac{C_i}{D_i}$ pour chaque tâche, on peut montrer qu'on peut garantir l'ordonnancement tant que :

$$\sum_i f_i \leq 1.$$

Comme dans le cas EDF [Stankovic *et al.*, 1998], il n'existe pas de test de faisabilité nécessaire et suffisant dans le cas d'échéances inférieures aux périodes. En effet, le chevauchement des tâches peut entraîner des situations de surcharge liés aux relations entre les échéances et les activations. Il est cependant possible d'utiliser le test d'ordonnancement général présenté en annexe C basé sur le facteur de charge.

7.2.1.4 Conclusion sur les politiques à taux constant

Les politiques à taux constants sont très intéressantes. Avec un ensemble de tâches périodiques à échéances sur requêtes, la politique à taux constant est optimale, les dates de fin d'exécution sont connues à l'avance et le temps de traitement est constant. On peut donc essayer de corriger l'influence du temps de traitement sur une régulation grâce à des techniques de prédiction. Nous présentons en section 5.4.2.5, les gains qui peuvent être obtenus grâce à l'utilisation conjointe de techniques d'ordonnancement à taux et l'adaptation de l'algorithme de commande. Les résultats sont très significatifs et permettent de conclure que l'ordonnancement à taux peut être une alternative intéressante dans les applications temps réel de type contrôle-commande.

Dans le cas où les tâches quittent ce modèle périodique, ou lorsque les échéances ne sont plus égales aux périodes, il devient très difficile d'appliquer les techniques d'ordonnancement à taux constant de manière à augmenter la prévisibilité sans réduire considérablement l'ordonnançabilité du système. Le but de la section suivante est donc de présenter une alternative à ces problèmes d'ordonnancement en proposant des techniques d'ordonnancement fluide ne reposant plus sur les notions de poids ou de taux mais utilisant de la réservation de ressource en avance.

7.2.2 Mise en place de techniques de réservation temps réel

Dans cette partie, nous analysons la faisabilité de l'utilisation de mécanismes de réservation explicite de ressources dans le cadre des applications temps réel. De manière générale, la réservation d'une ressource $\mathcal{R}$ par une application se caractérise par une requête émise à un instant t, et demandant, pour les besoins de l'application, une qualité de service vis à vis de l'occupation de $\mathcal{R}$ entre les instants $t + a$ et $t + b$. L'analyse, à l'instant t, de cette requête conduit à son acceptation ou à son rejet (test d'acceptation). Une requête acceptée n'est plus remise en cause par les requêtes dont l'occurrence est postérieure à t. Le taux de rejet (nombre de requêtes refusées / nombre total de requêtes) est un élément significatif de la qualité d'une politique de réservation. Parmi les techniques employées, les méthodes de réservation implicite se bornent à effectuer un test d'acceptation tandis que les techniques de réservation explicite réalisent, en complément du test d'acceptation classique, un placement de l'activité sur la ressource sous l'hypothèse que la stratégie d'ordonnancement "en ligne" respecte ce placement. Les taux de rejet obtenus avec de tels mécanismes d'une politique de réservation explicite sont fortement liés à la stratégie de placement utilisée.

Les mécanismes de réservation ont été initialement développés pour répondre aux besoins des applications

multimedias. Les taux de rejet obtenus avec de tels mécanismes sont significatifs et inacceptables dans le cas du temps réel classique. Par contre, certaines applications, comme les systèmes de poursuite de cible ou de télésurveillance de processus complexes, relèvent à la fois du multimédia et du temps réel.

Nous nous intéressons donc particulièrement aux techniques de réservation explicite dans ce contexte d'applications multimédia et temps réel. Aussi, d'une part nous prenons en compte le débit souhaité sur un intervalle de temps donné, ce qui correspond à respecter une contrainte sur un taux moyen (caractéristique propre au multimédia). D'autre part, nous intégrons la notion de garantie capacitaire, qui s'exprime comme le respect de contraintes sur la ressource totale allouée pendant un intervalle de temps donné (caractéristique propre aux applications temps réel). Plus spécifiquement, nous étudions des politiques de réservation explicite qui favorisent une répartition uniforme dans le temps du travail demandé à chaque requête (multimédia) tout en assurant la garantie capacitaire (temps réel). La Section 7.2.2.1 présente un état de l'art des techniques de réservation implicite et explicite. Après avoir introduit le modèle et les notations utilisés dans la suite pour spécifier et analyser les politiques de réservation en Section 7.2.2.2, nous étudions dans la Section 7.2.2.3 les politiques de réservation usuelles et proposons de nouvelles politiques adaptées au besoin identifié précédemment. Dans la Section 7.2.2.4, nous évaluons par simulation les performances des politiques proposées vis-à-vis de leur qualité de service avec comme métriques le taux de rejet et l'écart à une répartition uniforme de la charge.

7.2.2.1 Etat de l'art

Le problème posé est donc de spécifier et d'analyser des stratégies d'ordonnancement qui garantissent les ressources demandées lors de requêtes faites par l'application. De nombreuses solutions existent pour résoudre un tel problème. Classiquement, dans le domaine du temps réel, les études sur l'ordonnancement proposent différentes techniques en fonction de la connaissance de l'application et des services offerts par le système d'exploitation ou le protocole de communication utilisé.

Une première classe de techniques repose sur une connaissance totale et a priori de l'application. Dans ce cas, il n'y a pas de test d'acceptation en ligne puisque la preuve d'ordonnançabilité est effectuée une fois pour toute hors ligne. Cette preuve repose sur le modèle des activités et la connaissance des mécanismes d'ordonnancement sous-jacents.

Dans cette classe de techniques, on peut parfois considérer des motifs périodiques d'exécution définis a priori par des algorithmes de placement qui garantissent le respect des contraintes dans le motif. Volcano et Syndex sont des représentants de cette catégorie ([Sorel, 1996; Graefe and McKenna, 1993]).

D'autres approches de la même classe n'effectuent pas de placement et s'appuient uniquement sur un ordonnanceur en ligne comme Rate Monotonic (RM) [Klein *et al.*, 1994; Jones *et al.*, 1997] ou Earliest Deadline First (EDF) [Stankovic *et al.*, 1998; Cottet *et al.*, 2000; Liu, 2000].

Une deuxième classe de techniques répond à l'impossibilité éventuelle de connaître et de modéliser a priori l'application. Dans ce cas, il faut développer des mécanismes en ligne pour traiter des demandes de ressources (requêtes) pendant l'exécution de l'application. Dans ce cadre, on rencontre les stratégies d'ordonnancement pour lesquelles un test d'acceptation est exécuté à l'arrivée de chaque requête. Lorsque ces politiques utilisent un ordonnanceur en ligne, aucun placement de la requête n'est effectuée mais dans le cas contraire un placement explicite est calculé. Ces solutions ont été particulièrement étudiées dans le cadre des réseaux en particulier dans le cadre d'ATM (Asynchronous Transfer Mode [Le Boudec, 1992]), et plus précisemment pour les services liés à CBR (Constant Bit Rate [Grossglauser and Keshav, 1996]). Citons en outre que des techniques similaires ont été développées pour la ressource CPU comme en particulier l'algorithme EEVDF (Earliest Eligible Virtual Deadline First [Stoica *et al.*, 1996b]).

Enfin, la requête peut anticiper les demandes de ressource comme par exemple dans le cadre du protocole ReRA (Resource Reservation in Advance [Berson and Lindell, 1997; Degermark *et al.*, 1995; Karsten *et al.*,

1999; Ross, 1995; Schill *et al.*, 1997; Schill *et al.*, 1998; Schelén and Pink, 1998; Wolf *et al.*, 1995; Lars and Ralf, 1997]). Les techniques précedemment présentées doivent alors permettre la gestion des demandes en avance. Notons que les protocoles de réservation comme RSVP (Resource ReSerVation Protocol [Schill *et al.*, 1997; Schill *et al.*, 1998]) et ReRA ([Lars and Ralf, 1997]) (qui en définit une extension pour des réservations en avance) n'offrent ni un test d'acceptation, ni un calcul de placement, mais les moyens à l'application de procéder à une demande de réservation dans un cadre distribué (mécanimes de signalisation).

Dans le cadre de la réservation explicite, la stratégie la plus usuelle est la politique à taux constant [Mowbray *et al.*, 1998; Basney, 2001] qui servira de référence dans cette étude. Cette politique, couramment utilisée dans le multimédia, garantit un débit constant de ressources à une application et est bien adaptée aux besoins définis en introduction car elle permet de donner des garanties capacitaires tout en assurant un débit constant (placement uniforme). Nous verrons par contre dans la Section 7.2.2.4 que son taux de rejet est beaucoup plus important que celui obtenu avec les politiques proposées en Section 7.2.2.3.

7.2.2.2 Modèle de réservation explicite

Nous présentons dans cette Section le modèle de réservation explicite qui est considéré et nous introduisons les notations nécessaires à la description des politiques de réservation. Cette étude est basé sur l'hypothèse de fluidité de la ressource (cf 7.2)

Requête de réservation Une requête de réservation I_k est définie par le quadruplet $\{A_k, D_k, C_k, t_k\}$ où :

- A_k est la date de début au plus tôt,
- C_k est la quantité de ressource à réserver,
- D_k est la date au plus tard à laquelle l'intégralité du travail doit avoir été réservée,
- t_k est la date d'arrivée de la requête au gestionnaire de la ressource. Les requêtes sont indicées selon cet ordre d'arrivée : $\forall\, I_i, I_j$ t.q. $i < j$ on a $t_i < t_j$.

Une fois une réservation acceptée, elle ne sera pas remise en cause par des réservations ultérieures et les instants d'allocation de la ressource sont définis par des "intervalles chargés". La Figure 7.7 illustre l'ajout d'une requête sur une table de réservation existante.

Intervalle chargé Un intervalle chargé L_i est défini par le couple $\{l_i, c_i\}$ où l_i est un intervalle de temps du type $[\mathcal{A}_i, \mathcal{D}_i[$ et c_i est le taux de charge de la réservation sur cet intervalle. Les intervalles chargés sont définis de telle façon qu'ils sont contigus (éventuellement de charge nulle) et ils sont indicés par leur ordre dans le temps.

La réservation d'une requête I_j engendre une réservation de c_i^j sur tout intervalle chargé l_i dans $[A_j, D_j[$. Les c_i^j étant constants, le taux de charge sur tout intervalle est également constant. La table de réservation est une suite d'intervalles chargés.

Table de réservation La table de réservation $\mathcal{R}$ est la séquence de tous les intervalles chargés : $\mathcal{R} = \{L_i\}_{i=1..m}$ où m est le nombre total d'intervalles avec la propriété que $\forall i, i+1$ on a $\mathcal{D}_i \leq \mathcal{A}_{i+1}$. On considérera la fonction $\mathcal{R}(t)$ associée à la table de réservation $\mathcal{R}$, qui à chaque t, associe un taux de charge $\mathcal{R}(t)$ tel que s'il existe i t.q. $t \in l_i$, $R(t) = c_i$ et 0 sinon. Cette fonction est une constante par morceau et est définie continue à gauche.

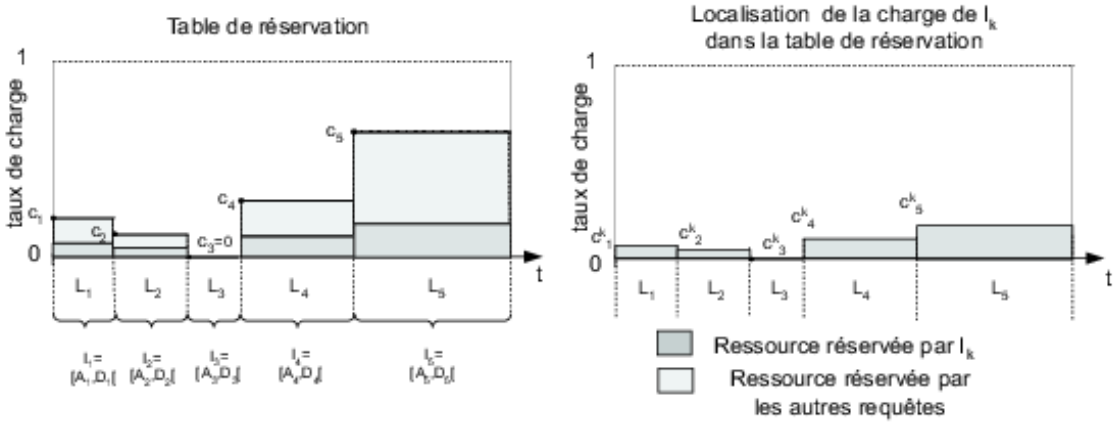


Figure 7.7: Placement d'une requête sur une table de réservation existante

On note $W(t_1, t_2)$ la charge, c'est à dire la quantité de ressource réservée, sur un intervalle $[t1, t2[$:

$$W(t1, t2) = \int_{t1}^{t2} R(t)dt \tag{7.4}$$

Le complémentaire de la charge est la fonction $\overline{W}(t1, t2)$ qui indique la disponibilité sur un intervalle $[t1, t2[$

$$\overline{W}(t1, t2) = (t2 - t1) - W(t1 - t2) \tag{7.5}$$

Politique de réservation Une politique de réservation est une fonction qui construit une nouvelle table de réservation à partir d'une nouvelle requête et de la table de réservation existante. On note $\mathcal{R}_k$ la table de réservation après traitement de la k^{ime} requête :

$$I_k, \mathcal{R}_{k-1} -> Politique -> \mathcal{R}_k, \text{ booleen}$$

La fonction renvoie un booléen pour indiquer si la requête a été acceptée ainsi que la nouvelle table de réservation qui sera inchangée si la requête a été rejetée. En pratique, une politique de réservation se décompose en deux phases: une phase d'acceptation et une phase de placement. La politique décide tout d'abord si elle accepte la réservation et le cas échéant, elle procède au placement de la charge de travail demandée par la requête dans la table de réservation selon sa politique propre.

Quelle que soit la politique de réservation, l'acceptation d'une requête I_k ne modifiera la charge que dans l'intervalle $[A_k, D_k[$. Pour définir les politiques de réservation, il nous suffit donc de considérer une restriction de la table à l'intervalle $[A_k, D_k[$. Cette restriction de la table de réservation s'écrit sous la forme d'une suite d'intervalles chargés contigus notée $\{L_i^k = \{l_i^k, c_i^k\}\}_{i=1..m_k}$. Selon la politique et selon les paramètres de I_k , il est possible que le placement d'une requête nécessite la création de nouveaux intervalles. L'ajout d'une réservation selon les politiques considérées dans cette étude crée au plus, deux intervalles. Nous définissons un opérateur d'insertion noté $\oplus$ qui prend comme opérande une table de réservation $\mathcal{R} = \{L_i\}_{i=1..m}$ et une date t et qui ajoute dans $\mathcal{R}$ l'intervalle crée par t :

$$\mathcal{R}^{'} = \mathcal{R} \oplus t \tag{7.6}$$

avec $\mathcal{R}^{'} = \{L_i^{'}\}_{i=1..m+1}$ où :

- $L_i^{'} = L_i \; \forall i \in 1..j-1$ avec j t.q. $A_j \leq t < D_j$,

- $L_j^{'} = \{[\mathcal{A}_j, t[, c_j\}$ et $L_{j+1}^{'} = \{[t, \mathcal{D}_j[, c_j\}$
- $L_{i+1}^{'} = L_i \, \forall i \in j+1..m$

Cette définition de la table de réservation sous forme d'intervalles chargés ne fait pas apparaître la quantité de ressource allouée pour chacune des requêtes sur $\mathcal{R}$. Au moment du placement d'une requête, on stocke l'allocation attribuée sur chacun des intervalles, connaissance qui est indispensable pour la phase où la requête est éligible (i.e. dans $[A_k, D_k[$).

Nous aurons besoin d'une mesure de la disponibilité sur la totalité de la table de réservation réduite notée $\overline{W_k}$:

$$\overline{W_k} = \sum_i (1 - c_i^k) \cdot (\mathcal{D}_i^k - \mathcal{A}_i^k) \tag{7.7}$$

et $\overline{W_k}(t_1, t_2)$ la disponibilité sur une portion $[t_1, t_2[$ de la restriction de la table de réservation.

7.2.2.3 Politiques de réservation

Dans cette section, nous présentons l'existant en matière de politiques de réservation ainsi que nos propositions nouvelles. Parmi l'existant, la politique de réservation à taux constant (TC) est sous-jacente à la très grande majorité des systèmes de réservation explicite dans les réseaux [Mowbray *et al.*, 1998; Basney, 2001] ou pour la réservation CPU [Kim and Nahrstedt, 2000]. Dans l'optique d'une répartition uniforme du travail, cette politique offre la meilleure QdS et elle servira de référence pour évaluer les performances de nos propositions dans la Section 7.2.2.4. Une autre politique dont l'utilisation a été suggérée dans les travaux de Basney liés à la co-allocation de ressources [Basney, 2001] est la politique "au plus tôt" (PTO). Cette politique sera un repère pour évaluer nos politiques vis-à-vis du taux de rejet. Enfin, nous présentons les politiques de réservation dites à lissage de tâches (LT), à lissage de charge (LC) et proportionnelle à la disponibilité (PD) que nous proposons comme alternative à TC.

Réservation à taux constant Cette politique, largement utilisée dans le cadre du multimédia, consiste à réserver, sur tout l'intervalle de la demande avec un taux constant (parfois aussi appelé bande passante constante dans les réseaux). Le taux de la requête est alors donné par $d = \frac{C_k}{D_k - A_k}$. La requête est acceptée s'il est possible de garantir le taux d sur toute la durée de la réservation. Le test d'acceptation de la requête I_k est :

$$\forall i \quad d \leq 1 - c_i^k \tag{7.8}$$

La charge c_i^k dans chaque intervalle l_i^k après acceptation de la requête I_k devient :

$$c_i^k = c_i^k + d \tag{7.9}$$

La figure 7.8 montre l'évolution de la table de réservation après acceptation d'une réservation à taux constant. La complexité de cette procédure de placement est celle d'un parcours de l'ensemble des intervalles.

Réservation au plus tôt En cas d'admission d'une requête I_k, la politique au plus tôt (PTO) charge la ressource au maximum depuis l'instant A_k et garantit ainsi la fin au plus tôt de la réservation. Une requête est acceptée s'il y a suffisamment de ressource disponible entre A_k et D_k, le test d'acceptation est donc $\overline{W_k} \geq C_k$. Remarquons que le test d'acceptation de TC ne suit pas cette règle car il ne suffit pas de disposer de suffisamment de ressource entre A_k et D_k, il faut également disposer d'un taux supérieur à d de ressource sur chaque intervalle l_i^k. Si une requête I_k est acceptée, il existe donc $t_k \in [A_k, D_k[$ tel que $\overline{W_k}(A_k, t_k) = C_k$. Le placement consiste tout d'abord à créer l'éventuel intervalle induit par t_k : $\mathcal{R} = \mathcal{R} \oplus t_k$. Ensuite, il faut

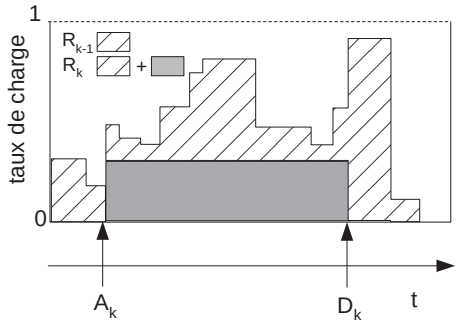


Figure 7.8: Une réservation selon la politique à taux constant.

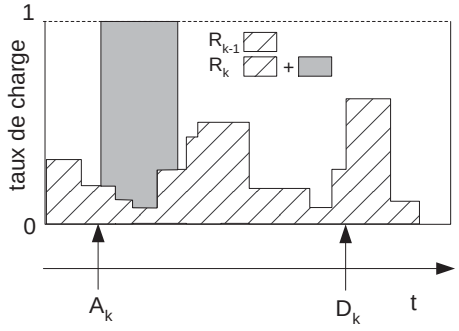


Figure 7.9: Une réservation selon la politique au plus tôt.

répartir la charge; la ressource sera utilisée à 100% à partir de A_k et ce jusqu'en t_k. La charge c_i^k dans chaque intervalle l_i^k après acceptation de la requête I_k devient donc simplement :

$$c_i^k = \begin{cases} 1 \, \forall \, i \;\; t.q. \;\; \mathcal{D}_i \leq t_k \\ c_i^k \text{ sinon} \end{cases}$$

La Figure 7.9 représente l'ajout d'une réservation selon la politique PTO sur une table de réservation existante. La complexité de cette procédure est linéaire en le nombre d'intervalles de la table.

Réservation à lissage de tâche Le principe de cette politique est de placer la charge uniformément sur tous les intervalles tant que cela est possible (i.e. $c_i^k < 1$). Une fois un intervalle complètement chargé, la politique continue à placer la charge uniformément sur les intervalles restants. La figure 7.10 montre un exemple de placement selon cette politique ou l'on voit que la charge sur l'interval central est plus faible que sur les autres intervalles car sa disponibilité était insuffisante.

Comme pour PTO, le test d'acceptation est $\overline{W}_k \geq C_k$. On note $\overline{W}_k^{\uparrow}(c)$ la disponibilité entre le niveau de charge 1 et le niveau de charge c entre les instants A_k et D_k (cf. Figure 7.11). Si une réservation I_k est acceptée alors $\overline{W}_k \geq C_k$ donc il existe $\rho_k \in [0,1]$ tel que $\overline{W}_k^{\uparrow}(\rho_k) = C_k$.

Dans la table de réservation restreinte à $[A_k, D_k[$, il n'y a pas de création de nouveaux intervalles. La charge de I_k est répartie, les c_i^k deviennent :

$$c_i^k = \begin{cases} 1 \text{ si } c_i^k \geq \rho_k \\ c_i^k + \rho_k \text{ sinon} \end{cases}$$

La difficulté d'implantation de la politique réside dans la calcul de ρ_k. Une façon de procéder est de trier par ordre décroissant les intervalles de la table restreinte selon leur niveau de charge. On répartit uniformément

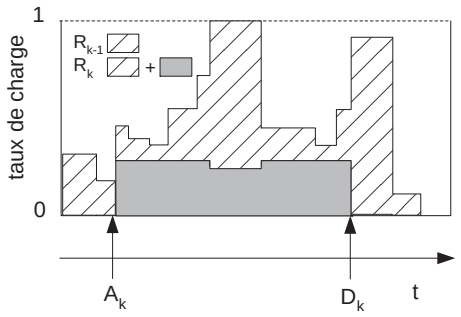


Figure 7.10: Placement d'une requête selon la politique de réservation à lissage de tache

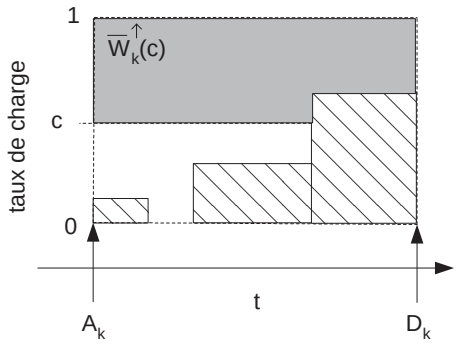


Figure 7.11: Définition de la fonction de disponibilit du niveau de charge 1 jusqu' un niveau de charge c.

jusqu'à saturation de intervalle le plus chargé, puis du second et ainsi de suite jusqu'à ce que la charge C_k soit attribuée. La Figure 7.12 illustre cette stratégie dont la complexité algorithmique est celle d'un tri plus d'un parcours de la liste des intervalles.

Réservation à lissage de charge Le but de cette politique est de minimiser la hauteur maximale de charge dans la table de réservation. Intuitivement, le placement d'une requête I_k peut être vue comme l'ajout d'une quantité de liquide C_k dans un récipient dont les bords seraient en A_k et D_k et dont le fond aurait comme relief la table de réservation avant l'arrivé de I_k (cf. Figure 7.13).

Le test d'acceptation est $\overline{W}_k \geq C_k$. On note $\overline{W}_k^{\downarrow}(c)$ la disponibilité jusqu'au niveau de charge c (en partant du bas donc) entre les instants A_k et D_k. Si une réservation I_k est acceptée alors il existe $\rho_k \in [0,1]$ tel que $\overline{W}_k^{\downarrow}(\rho_k) = C_k$.

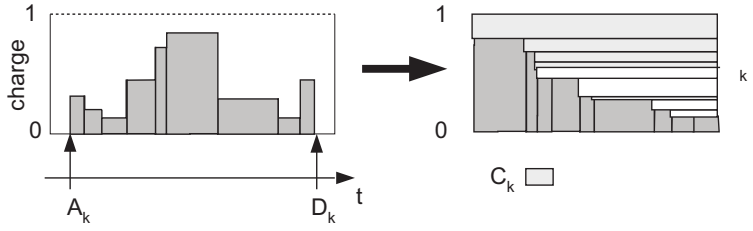


Figure 7.12: Le fait de trier l'intervalle et de connaître la longueur totale de l'ensemble des intervalles, permet de calculer en une étape la charge apportée entre le niveau courant et le niveau précedent pour l'ensemble des intervalles.

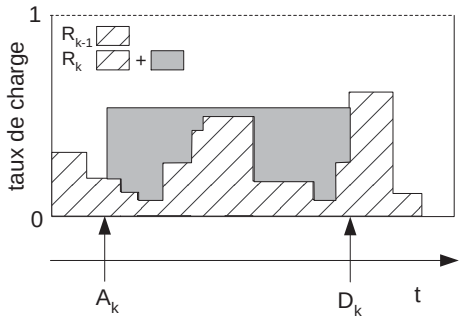


Figure 7.13: Exemple de réservation à lissage de charge

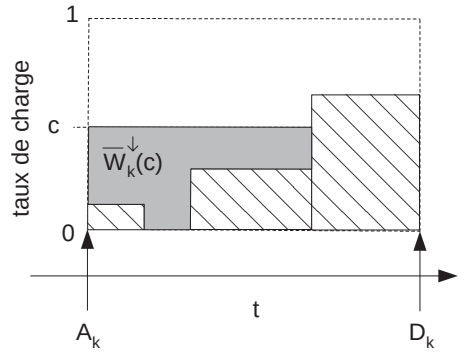


Figure 7.14: Définition de la fonction de disponibilité du niveau de charge 0 jusqu'à un niveau c

Dans la table de réservation restreinte à $[A_k, D_k[$ aucun nouvel intervalle ne se sera créé et les c_i^k deviennent :

$$c_i^k = \begin{cases} \rho_k \text{ si } c_i^k \leq \rho_k \\ c_i^k \text{ sinon} \end{cases}$$

La difficulté d'implémentation réside en le calcul de ρ_k t.q. $W_k^{\downarrow}(\rho_k) = C_k$. Une façon de procéder est de trier les intervalles selon leur charge (comme pour LT, cf. Figure 7.12) puis de faire monter la charge en partant de l'intervalle le moins chargé jusqu'à ce que les C_k unités de charge soient réparties. La complexité de ce placement est comme pour LT celle d'un tri et d'un parcours de $\mathcal{R}$.

Réservation proportionnelle à la disponibilité Cette politique repose sur le principe simple de répartir la charge proportionnellement à la disponibilité de chacun des intervalles. Un intervalle reçoit une part de la réservation proportionnelle à son taux de disponibilité. La Figure 7.15 montre un exemple de placement d'une requête selon la réservation proportionnelle à la disponibilité (PD).

Le test d'acceptation est $\overline{W}_k \geq C_k$. On note $\rho = \frac{C_k}{\overline{W}_k}$ la part de la disponibilité qui va être utilisée par la requête I_k sur $[A_k, D_k[$. Sur chaque intervalle, la charge ajoutée est proportionnelle à la disponibilité :

$$c_i^k = c_i^k + \rho\left(1 - c_i^k\right)$$

La complexité algorithmique de cette politique de placement est celle d'un parcours de la table de réservation restreinte.

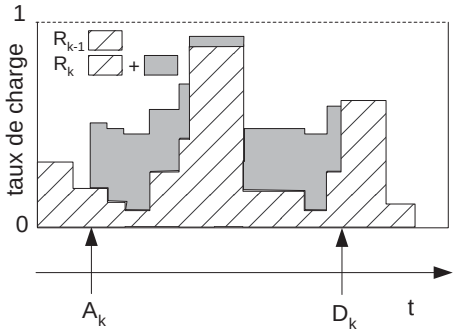


Figure 7.15: Placement d'une requête selon la réservation proportionnelle à la disponibilité.

7.2.2.4 Évaluation de performances des politiques de réservation explicite

Dans cette section, nous cherchons à évaluer les performances des politiques lissage de charge (LC), lissage de tâche (LT), proportionnelle à la disponibilité (PD) par rapport à la politique de référence à taux constant (TC). Ces politiques ont toute en commun, qu'en absence de charge déjà réservée, elles effectuent le même placement pour une réservation acceptée. Lorsqu'une charge pré-existe, ces politiques peuvent et vont généralement différer.

Métriques de performances Les critères de performances retenus pour évaluer les différentes politiques de réservation sont le taux de rejet, associé à une métrique de l'uniformité de la répartition obtenue de la charge de chaque requête.
Le taux de rejet peut s'exprimer soit par requêtes :

$$\text{taux de rejet en requêtes} = \frac{\text{Nombre de requêtes acceptées}}{\text{Nombre de requêtes soumises}}$$

soit vis à de la charge totale rejetée :

$$\text{taux de rejet en charge} = \frac{\text{Charge acceptée}}{\text{Charge soumise}}$$

Le critère choisi pour mesurer l'écart à l'uniformité, est l'écart absolu :

$$\text{écart absolu} = \frac{1}{2} \cdot \frac{\sum_{k \in \mathbf{A}} \left(\int_{A_k}^{D_k} \mid c^k(t) - d_k \mid \cdot dt \right)}{\sum_{k \in \mathbf{A}} C_k} \tag{7.10}$$

où $c^k(t)$ est la quantité de travail allouée à I_k à l'instant t, $d_k = C_k/(D_k - A_k)$ est le taux de répartition uniforme et $\mathbf{A}$ est l'ensemble des requêtes acceptées. Ce critère exprime le pourcentage de charge placée non-uniformément sur l'ensemble des requêtes acceptées. Nous nous intéressons à la prévisibilité sur les temps de traitement, nous aurions donc pu le prendre directement comme critère. Cependant, dans le cas où les temps d'exécution sont plus faibles que les temps réservés, correspondant aux Worst Case Execution Time, le temps de traitement calculé peut différer de façon importante de celui constaté (par exemple si la plus grande partie de la charge réservé l'est à proximité de la date d'activation). Le critère d'uniformité est un bon indicateur de l'écart au cas idéal donné par la politique à taux constant. Il permet donc d'estimer la constance des temps de traitements sans avoir à les calculer explicitement.

Les politiques de placement sont évaluées relativement à cettes métrique de performances à l'aide d'un

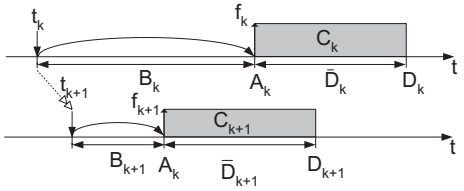


Figure 7.16: Deux requêtes de réservation Ik et Ik+1 générées selon notre modèle de trafic où Dk est la différence entre la date de fin au plus tard et la date de début au plus tôt.

modèle de trafic présenté dans le prochain paragraphe.

Modèle de trafic Nous considérons une ressource unique sur laquelle des activités génèrent des requêtes de réservation. L'ensemble de ces requêtes forme ce que nous appelons le trafic ou le flux de requêtes de réservation. Chaque requête est indicée selon sa date d'arrivée et est définie par le quadruplet (t_k, A_k, C_k, D_k) où t_k est la date d'arrivée (appelée également date de création de la requête), A_k est la date de début au plus tôt, C_k la quantité de ressource demandée et D_k la date de fin au plus tard. Le modèle de trafic génère un flux de requêtes selon quatre paramètres :

1. La valeur moyenne $\overline{D}_{moy}$ de la durée de l'intervalle $[A_k, D_k[$ notée $\overline{D}_k$. Dans nos expérimentations, $\overline{D_k}$ suit une loi de distribution uniforme entre $[0.5 \cdot D_{dmoy}, 1.5 \cdot D_{moy}]$.

2. Les bornes sur le taux de charge des réservations notées f_{min} et f_{max}. Le facteur de charge f_k suit une loi uniforme sur $[f_{min}, f_{max}]$, la quantité de ressource à réserver C_k est alors obtenue par

$$C_k = \overline{D_k} \cdot f_k$$

3. La borne sur le temps avant réservation noté B_{max}. Le temps avant réservation (connu sous le nom de *"booking time"*) $B_k = A_k - t_k$ est le temps qui sépare la création d'une requête et le début au plus tôt de la réservation de ressource. B_k suit une loi uniforme sur $[0, B_{max}]$.

4. La charge moyenne de la simulation ρ. Les interarrivées $(t_{k+1} - t_k)$ suivent une loi uniforme sur $[0, \alpha]$. La borne α du processus d'interarrivées est calculée en fonction de ρ et des autres paramètres. d'interarrivées :

$$\alpha = 2 \cdot \frac{T_{simulation} - \frac{B_{max}}{2} - \overline{D}_{moy}}{(\frac{T_{simulation} \cdot \rho}{\overline{D}_{moy} \cdot f_{moy}})} \tag{7.11}$$

   où le numérateur représente une estimation d'une borne sur t_k et le dénominateur du nombre de requêtes nécessaire pour arriver à un niveau de charge ρ.

Ce modèle nous permet de générer un trafic homogène (i.e. toutes les activités qui génèrent des requêtes ont des besoins similaires) pour lequel le taux de charge, la durée des réservations et le temps avant réservation sont décorrélés. Dans la suite, les simulations ont toutes été effectuées avec un minimum de 10000 requêtes et nous avons imposé un temps de simulation $(T_{simulation})$ supérieur à $30 \cdot B_{max}$ et $30 \cdot \overline{D}_{moy}$ pour limiter l'influence des zones transitoires de début et de fin de simulation.

Résultats de simulation Un scénario de simulation est défini par un modèle de trafic instancié et une politique de réservation. Le simulateur génère alors les requêtes selon le modèle défini et la gestion de la table de réservation est effectuée selon la politique choisie. En fin de simulation, les valeurs des métriques de performances sont calculées.

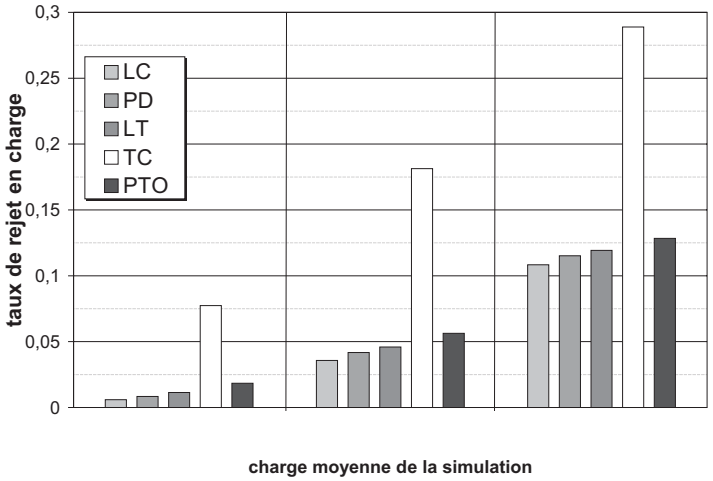


Figure 7.17: Taux de rejet en charge pour les politiques LC, PD, LT, TC et PTO pour une charge soumise de 50, 70 et 90 pour cent. Le taux de rejet pour TC est toujours nettement plus élevé.

Dans tous les scénarios considérés, le facteur de charge d'une requête a été tiré aléatoirement dans l'intervalle $[0.1, 0.2]$ (paramètres f_{min} et f_{max}). Ces facteurs de charge relativement élevés modélisent des besoins fréquemment rencontrés dans le cadre de l'ordonnancement temps réel de tâches ou pour la transmission de données sur des réseaux temps réel. Si les facteurs de charge considérés sont très faibles, toutes les requêtes sont de petite taille et les quatres politiques en concurrence se comportent de façon très similaire car il reste généralement suffisamment de ressource pour placer chacune des requêtes soumises et ce jusqu'à des taux d'utilisation de la ressource proches de 1. Des facteurs de charge très élevés (i.e $> 40\%$) n'ont pas été considérés car il ne correspondent pas aux besoins usuels des applications dans lesquelles la réservation est envisagée.

Nous avons considéré deux classes de trafic relativement au rapport $\overline{D}_{moy}/B_{max}$.

Nous évaluerons tout d'abord l'influence d'une montée de la charge sur les performances des politiques en concurrence. Puis nous ferons varier le rapport $\overline{D}_{moy}/B_{max}$ pour estimer l'influence de la forme des requêtes sur le comportement des politiques. Dans la suite, les résultats présentés sont indépendants de l'unité de temps choisie.

QdS et charge soumise Les expérimentations présentées dans ce paragraphe ont été réalisées pour une charge totale soumise de 50, 70 et 90 % et avec $B_{max} = 10000$, $D_{moy} = 500$ (requêtes fortement enchevêtrées).

Les résultats vis-à-vis du taux de rejet sont excellents : les taux de rejet en charge obtenus avec les trois politiques proposées sont de 2.4 fois (LT à 90 % de charge) à 12,8 fois (LC à 50% de charge) inférieurs aux taux de rejet de la politique de référence TC (cf. Figure 7.17). Les gains vis-à-vis du taux de rejet exprimés en pourcentage de requête sont du même ordre de grandeur. Les taux de rejet sont relativement proches entre LT, LC et PD mais les plus faibles sont obtenus avec la politique LC, puis en second avec PD, quelle que soit la charge. Remarquons également que nos politiques se comportent ici mieux que la politique "au plus tôt" (PTO) qui pourtant n'a pas cet objectif supplémentaire de répartir la charge uniformément.

Relativement à l'écart absolu à l'uniformité (cf. Equation 7.10), on remarque que les performances de LT et PD sont comparables avec une charge non-uniformément répartie oscillant entre 1% (LT à 50% de charge) et 9% PD (à 90% de charge) comme cela est montré sur la Figure 7.18. Néanmoins, LT est toujours sensiblement meilleure avec une charge mal placée ne dépassant pas 4.6%. On peut estimer que les performances de PD et LT sont très satisfaisantes vis-à-vis de ce critère. LC, qui privilégie la ressource au détriment des requêtes, se comporte logiquement moins bien avec une charge mal placée relativement importante et variant très peu selon la charge (entre 18% et 20% dans nos conditions d'expérimentation). La politique de référence TC n'est

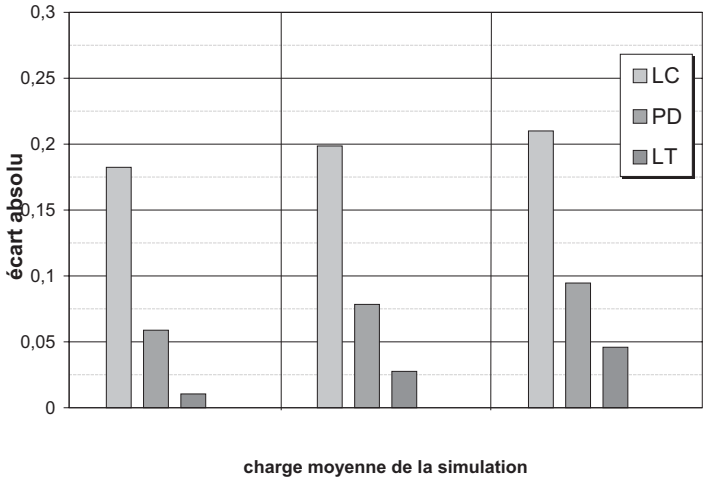


Figure 7.18: Ecart absolu l'uniformité pour les politiques LC, PD, LT pour une charge soumise de 50, 70 et 90 pour cent.

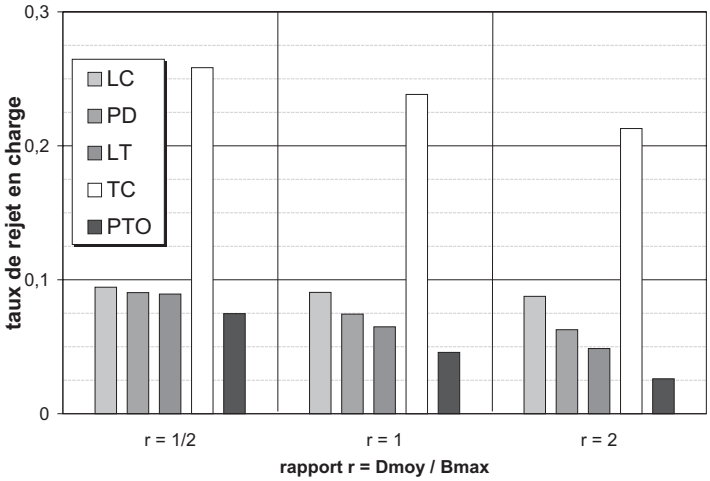


Figure 7.19: Taux de rejet en charge pour les politiques LC, PD, LT, TC et PTO pour r=0.5,r=1 et r=2 (Bmax=5000 et Dmoy=2500, 5000 et 10000).

pas représentée sur la Figure 7.18, son écart à l'uniformité étant toujours nul. A titre de comparaison, pour PTO la charge mal placée est de 85%.

Influence de la forme des requêtes Notre objectif est ici d'étudier comment les caractéristiques des requêtes influent sur les performances des politiques. Nous ferons varier le rapport $r = \overline{D}_{moy}/B_{max}$ jusqu'à nous situer clairement dans la seconde classe de trafic identifiée (requêtes très peu imbriquées). La charge moyenne de simulation est constante à 90% pour toutes les expérimentations de ce paragraphe.

Sur la figure 7.19 on observe que les politiques LC, PD et LT se comportent toujours nettement mieux que la politique à taux constant (TC) en termes de taux de rejet. Le gain varie ici d'un facteur 2.4 (LC pour $r = 2$) à un facteur 4.4 (LT pour $r = 2$). En considérant les expérimentations du paragraphe précédent pour lesquelles on avait $r = 0.05$, on observe que les performances relatives des politiques LC, PD, LT et PTO s'inversent au fur et à mesure que r devient grand. Ce comportement est lié à une raréfaction du nombre de chevauchements des réservations, c'est à dire une situation où globalement, l'ordre des demandes de début de réservation (A_k) suit l'ordre des créations t_k (cf. paragraphe 7.2.2.4). Dans ce cas, il n'y a plus interêt à

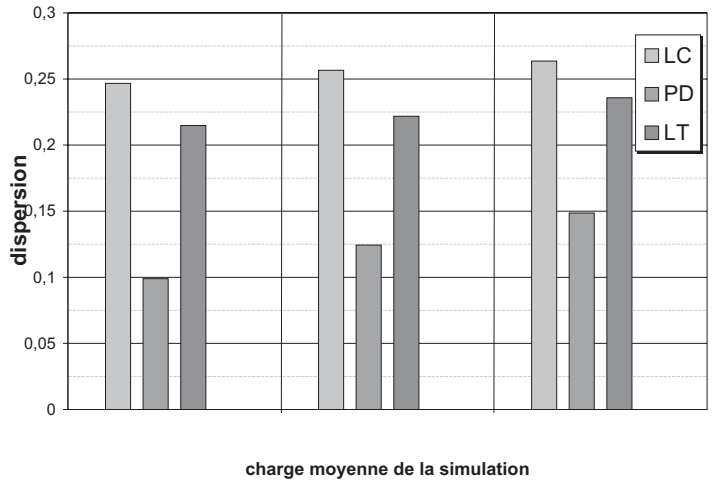


Figure 7.20: Ecart absolu l'uniformit pour les politiques LC, PD, LT, TC et PTO pour r=0.5,r=1 et r=2 (Bmax=5000 et Dmoy=2500, 5000 et 10000).

répartir uniformément la charge d'une requête I_k mais à réserver au plus tôt car la probabilité qu'une requête arrivant plus tard demande de la ressource avant A_k est faible. Ceci explique les très bonnes performances de PTO vis-à-vis du taux de rejet lorsque r croit. Si PD et LT sont meilleures que LC, c'est que leurs placements sont plus proches de celui fait par PTO ce qui explique cette inversion dans les performances relatives des trois politiques proposées par rapport aux résultats du paragraphe 7.2.2.4. Néanmoins, le gain par rapport à la politique de référence reste toujours très important et comparable à celui obtenu dans les expérimentations du paragraphe précédent dans lequel le gain à 90% de charge se situait entre un facteur 2.4 et 2.6 (cf. Figure 7.17).

En ce qui concerne le critère d'écart absolu à l'uniformité (cf. Figure 7.20), les performances relatives de LC, PD et LD sont identiques à celles du paragraphe 7.2.2.4 et ces expérimentations nous font penser qu'elles sont peu dépendantes des caractéristiques du trafic. On observe par contre que plus r augmente, plus l'écart absolu croit et ce, pour chacune des politiques. Notons que PTO qui, pour ce trafic peu enchevêtré, est la meilleure politique envisagée en termes de taux de rejet ne devient pas pour autant une solution à notre problème car la charge non-uniformément répartie est de 85%.

Analyse des résultats [1]

Les politiques LT, PD et LC tendent à répartir la charge uniformément mais sous ces politiques, et contrairement à la politique de référence TC, une requête I_k sera toujours acceptée si il y a plus de C_k unités de ressource entre A_k et D_k. Le fait que le test d'acceptation soit identique pour ces trois politiques est une explication des taux de rejet similaires.

En acceptant une distorsion dans la répartion uniforme de la charge, il était logique que LT, PD, et LC se comportent favorablement en terme de taux de rejet vis-à-vis de TC. Néanmoins, le gain est considérable (d'un facteur 2.3 à un facteur 12 pour le trafic du paragraphe 7.2.2.4 et de 3.3 à 7.5 pour le trafic du paragraphe 7.2.2.4) alors que les pertes dans l'uniformité de l'allocation sont relativement faibles pour PD et surtout pour LT.

Les fortes différences sur le taux de rejet entre les deux classes de trafic identifiées nous incite à penser que les performances des politiques en termes de taux de rejet sont très dépendantes du trafic et notamment par le fait que l'ordre de la séquence des A_k suit l'ordre de la séquence de t_k. Une bonne connaissance des caractéristiques du trafic d'entrée semble donc nécessaire si l'on veut pouvoir évaluer les performances des

[1] Une étude plus détaillée de ces politiques de réservation a déjà été publié dans la revue TSI numéro spécial "temps réel" (cf. [Jumel *et al.*, 2002]).

politiques vis-à-vis de ce critère.

Les écarts entre les différentes politiques vis-à-vis de l'uniformité de la charge s'expliquent par leur définition même et le fait qu'elles répartissent plus ou moins bien la charge dans la table de réservation. En ce qui concerne l'écart absolu, la politique LC donne toujours de meilleurs résultats, suivie de PD puis de LT et cet ordre reste identique pour des trafic de caractéristiques très différentes.

7.2.2.5 Conclusion

La politique "à taux constant" (TC) a en pratique pour inconvénient majeur de refuser un pourcentage important de requêtes soumises et donc de ne pas utiliser la ressource au maximum atteignable du taux de charge.

En acceptant une dégradation dans la répartion uniforme de la charge, nous avons défini trois politiques de réservation qui offrent des compromis différents entre utilisation de la ressource et uniformité des activités s'exécutant sur cette ressource (apportant ainsi la constance des temps de traitement). Nous avons évalué leurs performances par simulation relativement au taux de rejet des requêtes et à l'uniformité de la répartition de la charge en définissant deux métriques de performance qui sont l'écart absolu à l'uniformité et la dispersion de la charge mal-répartie. En pratique, il est possible de communiquer à l'application le résultat de la réservation qui décidera si la qualité est suffisante.

Parmi les politiques proposées, la politique dite "à lissage de charge" (LC) a les moins bonnes performances vis-à-vis de l'uniformité de la répartition de la charge. Elle minimise par contre le taux de rejet dans le cas où l'ordre des demandes de réservation est décorrélé de l'ordre des dates de début au plus tôt. Néanmoins plus le trafic perd cette caractéristique, plus les taux de rejet augmentent. En partant de ce constat, nous pensons qu'il est possible de concevoir des politiques visant exclusivement à minimiser les taux de rejet qui utiliseraient une connaissance sur certaines caractéristiques du trafic d'entrée. C'est une des perspectives de cette étude.

Les politiques "proprotionnelle à la disponibilité" (PD) et "à lissage de tâches" (LT) sont les plus performantes vis-à-vis de l'uniformisation de la charge. Clairement, les meilleurs résultats en termes de pourcentage de charge non-uniformément répartie sont obtenus par LT, la contrepartie étant des taux de rejet légérement supérieurs. La politique PD offre toujours un bon compromis rejet / uniformité, en particulier elle est la meilleure pour le critère de dispersion de la charge mal-répartie pour tous les trafics expérimentés dans cette étude. De plus, sa complexité algorithmique est la plus faible des politiques en compétition.

Le modèle de trafic que nous avons utilisé, même si il répond au standard développé autour des travaux sur la réservation de ressources en avance (RERA) pourrait être adaptée de manière à mieux correspondre aux spécificités des besoins des applications de contrôle-commande. Les conclusions obtenues seront a priori peu changeantes même si l'existence d'un flux de réservation périodique peut améliorer les performances de la politique TC "à taux constant" (cf. remarque sur l'optimalité dans le cas à échéances sur requêtes dans le paragraphe 7.2.1.3).

L'ordonnancement de ressources basé sur la réservation explicite est particulièrement intéressant dans le cadre de la réservation de ressources en série. En effet, les placements explicites de ressources sur un site, peuvent être transférés de proche en proche au cours de la phase de signalisation de la réservation (comme c'est déjà le cas des délais dans RSVP). Le mécanisme qui gère les réservations en avance (par exemple une extension ReRA) peut alors fournir des garanties exprimées de bout en bout, tout en respectant d'éventuelles contraintes de précédence (éventuellement définies de manière fluide). Maintenant que nous disposons de politiques de placement monosite, il faut définir des règles d'utilisation en série de ces politiques. En particulier, il serait intéressant de définir des politiques de lissage sur plusieurs ressources simultanément.

7.3 Conclusion

Nous avons cherché dans ce chapitre à améliorer la prévisibilité des temps de traitements. Deux techniques ont été proposées, la première définit un gabarit d'exécution sur les tâches alors que l'autre repose sur les techniques de réservation.

La réservation de ressource à taux constant, qu'elle soit implicite comme dans les politiques à taux ou explicite avec la politique TC, permet de rendre constant les temps de traitement mais a le désavantage important de ne pas permettre un taux d'utilisation des ressources très important ce que limite son intérêt dans le cadre d'une utilisation pour des applications temps réel. D'autres techniques de réservation explicites ont donc été proposées qui permettent de connaître à l'avance les temps de traitement tout en offrant de meilleurs résultats sur l'utilisation de la ressource. En particulier, la politique "proportionnelle à la disponibilité" semble offrir un compromis très intéressant, entre taux d'utilisation et uniformité du placement (ce qui implique une propriété de constance sur l'allocation de la ressource en plus de la prévisibilité).

Une des perspectives de recherche est l'extension de ces politiques de réservation dans un cadre distribué afin de permettre une réservation de bout en bout pour toutes les activités d'une même transaction. Outre la prévisibilité, certaines applications de contrôle-commande expriment des besoins en termes de flexibilité de l'ordonnancement. En particulier, l'annexe C présente une famille de politiques d'ordonnancement conçues pour des tâches qui peuvent s'adapter à la quantité de ressource dont elles disposent (tâche *anytime*, cf. [Jumel and Simonot-Lion, 2002] pour plus de détails).

Conclusions et perspectives

Le but du travail présenté dans ce document est la proposition de techniques et outils permettant d'améliorer le développement des systèmes numériques de contrôle-commande, et plus particulièrement des systèmes échantillonnés dans le contexte de la régulation, en intégrant dans l'analyse, les caractéristiques relevant de son implantation sur un support informatique induisant inévitablement des délais de traitement et de transmission d'information.

Une contribution significative de ce travail a été d'établir une méthode pour exprimer et évaluer la qualité obtenue par un système régulé en fonction de caractéristiques d'implantation du système de contrôle.

Pour atteindre cet objectif, nous nous sommes intéressés à la définition précise des besoins d'une régulation ainsi qu'à l'identification des critères de qualité associés. Puis, nous avons fait le point sur les techniques de modélisation des systèmes échantillonnés et continus de considérer les performances de l'architecture opérationnelle supportant les algorithmes de contrôle. En particulier, nous avons présenté un modèle général prenant en compte les différentes formes d'implantation. Dans ce modèle, une architecture opérationnelle apparaît comme génératrice de fautes temporelles sur les occurrences de signaux utilisés par le système de contrôle (retards et de gigues). La connaissance précise des caractéristiques de ces fautes, nécessite la prise en compte des événements de création et utilisation des données. Nous avons présenté les points importants de cette modélisation qui repose sur la notion d'échantillons de données.

Pour évaluer des architectures opérationnelles de lois de commande dans les cas où une étude analytique ne peut être conduite en raison de la complexité du système et / ou de son implantation, nous avons réalisé un outil de simulation. Cet outil, qui repose sur Matlab / Simulink, permet d'étudier l'influence des caractéristiques de l'architecture comme les politiques d'ordonnancement des accès aux ressources ou les modes de coopération entre activités. Son utilisation a permis, en particulier, d'analyser l'influence de phénomènes complexes, comme la variabilité des temps de traitement. Nous avons obtenu des résultats intéressants qui contredisent des idées intuitives sur les besoins des applications de contrôle commande, résultats parmi lesquels on peut citer notamment :

- La qualité d'un système de contrôle est liée aux propriétés en moyenne des temps de traitement plutôt qu'à l'existence d'une borne maximale (échéance).
- La qualité peut être plus mauvaise pour des temps de traitements variables que pour des temps de traitement constants plus grands.

De ces constats, nous avons centré notre étude sur deux voies :

- l'identification des propriétés essentielles au bon fonctionnement d'une application de contrôle commande,
- la spécification de techniques qui en augmentant la prévisibilité des temps de traitements permettent d'augmenter la qualité de la régulation.

La première voie a débouché sur la mise en place d'un ensemble de techniques associées à l'évaluation de la sûreté de fonctionnement des systèmes régulés. Le principal résultat est constitué par une méthodologie

de calcul de la fiabilité d'un système régulé en fonction de son architecture opérationnelle. Elle permet en particulier de quantifier l'apport réel des différents mécanismes de tolérances aux fautes. Cette étude nous a conduit à proposer la définition de contraintes de sûreté qui définissent le nombre minimal de traitements corrects à garantir en cas de fonctionnement dégradé de l'architecture support.

La seconde voie s'est concrétisée par la spécification de deux classes de techniques permettant d'augmenter la prévisibilité des temps de traitement. Dans l'une, on enrichit le modèle de tâche de nouvelles contraintes exécutives alors que dans l'autre on définit des techniques d'ordonnancement par réservation qui permettent de garantir par construction les dates de fin de traitement.

Perspectives

A l'issue de ces travaux, différentes perspectives peuvent être dégagées à plus ou moins long terme.

Tout d'abord, l'implantation d'un ordonnancement fluide est problématique et engendre l'apparition d'une surcharge ("overhead") importante vis à vis des temps de traitement. Même si cet overhead peut être intégré dans la définition des temps d'exécution, une phase d'implantation sur des systèmes réels serait nécessaire afin de valider l'intérêt de ces techniques.

Les techniques de réservation proposée pourraient être utilisées dans d'autres domaines, et en particulier dans les grappes de calcul. En effet, ces techniques peuvent apporter une solution là où les techniques classiques ont du mal à gérer la distribution des traitements avec des contraintes temporelles de bout en bout. En particulier, elles permettent de connaître à l'avance les possibilités d'exécution en parallèle et en série sur plusieurs composants (processeurs et réseaux).

La méthodologie de calcul de la fiabilité des systèmes régulés mérite d'être intégré à un outil d'analyse de la sûreté de fonctionnement. La recherche de la fonction d'état d'erreur associée à un système régulé permettrait d'affiner considérablement la connaissance des véritables besoins en termes de sûreté de fonctionnement. Elle permettrait, en outre, de proposer l'analyse du fonctionnement d'un système en présence de fautes transitoires et non uniquement permanente comme cela est traditionnellement le cas.

La principale perspective de nos travaux réside dans la définition de contraintes sur le fonctionnement des architectures informatiques associées. L'étude de fiabilité présentée au chapitre (6) repose sur la fonction d'état d'erreur qui permet de connaître l'évolution du système en présence d'erreurs. Il est ainsi possible de déterminer la pire évolution possible du système en présence de défaillances de la commande. Cette connaissance peut aussi être appliquée à la gestion des situations d'urgence. En effet, en cas de surcharge du système informatisé [2], il est important de disposer au mieux des ressources disponibles. Pour cela, il est possible d'utiliser, en ligne, une estimation de la fonction d'état d'erreurs. La gestion des ressources disponibles pourrait ainsi se faire en utilisant les fonctions d'état d'erreurs de différentes régulations partageant les mêmes ressources. La connaissance de la fonction d'état d'erreur permettrait de connaître la pire évolution possible des différentes régulations en absence de commande. Il serait ainsi possible, sans même s'intéresser aux données en provenance des capteurs (qui peuvent ne pas être disponible), de déterminer quelles sont les régulations dont la situation est la plus critique et de privilégier l'attribution des ressources en fonction de cette criticité.

La fonction d'état d'erreur permet de lier directement l'existence de traitements corrects et l'apparition de la défaillance majeure du système régulé. Elle est donc le meilleur indicateur possible de criticité de la régulation vis à vis d'absence de traitement. Elle peut être utilisée pour élaborer des modes de marche dégradés en donnant les contraintes sur les traitements minimaux à effectuer. En effet, plutôt que d'utiliser directement la fonction d'état d'erreur, on peut en dériver une contrainte sur les traitements à garantir de manière à éviter l'apparition de la défaillance du système régulé, par exemple sous la forme de contraintes

[2] Elle peut être due à la perte des capacités de traitement (liée à une défaillance du matériel, à la présence de perturbations éléctromagnétiques ...) ou à l'existence d'activités non prévues dans le dimensionnement des ressources.

(m,k)-firm [Ramanathan, 1999]. C'est à dire sous la forme d'une exigence de m traitements corrects sur k traitements consécutifs. Une perspective importante consiste à définir des moyens pour spécifier la contrainte (m,k)-firm (en particulier les valeurs de m et de k) qui assurent la qualité de service suffisante requise par une régulation.

Des contraintes similaires doivent être proposés vis à vis des temps de traitement Les études par simulation nous ont montré la répercussion importante de la variabilité des temps de traitement sur la qualité de fonctionnement du système. Une suite à nos travaux est l'identification de la fonction qui lie qualité et variabilité. Nous pourrons alors fixer des contraintes dont nous validerons l'intérêt analytiquement (sous forme de garantie sur les critères de Qualité de Services). Globalement ces travaux permettront :

- de proposer une méthodologie précise de spécification des contraintes d'échéances sur des événements pour assurer le respect de contraintes sur des critères de qualité de service des systèmes régulés (comme le temps de réponse à un changement de consigne).
- de définir un ensemble de politiques en ligne (sous forme de (m,k) firm, par exemple) pour assurer le fonctionnement optimal du système global (et pas seulement de son fonctionnement en mode dégradé) et optimiser l'utilisation de l'Architecture Opérationnelle.

Certaines perspectives présentées ci-dessus sont en cours d'étude. Afin de conforter leur intérêt, nous nous proposons de les appliquer dans le domaine des systèmes X-by-Wire (systèmes embarqués sans redondance mécanique embarqués dans l'automobile) pour la spécification de services exécutifs intégrant des mécanismes de tolérance aux fautes validés au niveau applicatif.

Index

Architecture Matérielle, 58
Asservissement, 18
ATM Asynchronous Transfer Mode, 113

CBR Constant Bit Rate, 113
commande optimale, 157
continu, 143
Coopération, 47

EEVDF Earliest Eligible Virtual Deadline First, 113

Fréquence d'échantillonnage, 20

Hypothése de fluidité, 109

Matrice d'état, 143

Non-linéarités, 148

Politique à taux constant, 110
Politiques à poids, 109
Politiques de réservation, 109
Prédicteur de Smith, 156
Pull, 49
Push, 47

Régulation, 18
ReRA Ressource Reservation in Advance, 113
RSVP Resource ReSerVation Protocol, 113

Système, 17
Système 1er ordre, 150
Système échantillonné, 20
Système du $2^{\text{ème}}$ ordre, 150

Tâches à gabarits, 104
Transformée de Fourrier, 145
Transformée de Laplace, 141

Bibliography

[Abeni *et al.*,] L. Abeni, G. Lipari, and G. Buttazzo. Constant bandwidth vs proportional share resource allocation. In *Proceedings of IEEE International Conference on Multimedia Computer and Systems 1999 (ICMCS 99)*, Juin.

[Akian *et al.*, 1998] M. Akian, P. Bliman, and M. Sorine. P.i. control of nonlinear oscillations for a system with delay. Technical Report 3422, INRIA, Mai 1998.

[Andreff, 1994] N. Andreff. *Robustness to jitter in real-time systems.* PhD thesis, University of Lünd, Novembre 1994.

[Årzén, 1999] K.-E. Årzén. A simple event-based PID controller. In *Preprints 14th World Congress of IFAC*, Beijing, Chine, 1999.

[Askerdal *et al.*, 2002] Ö. Askerdal, Gäfvert M., Hiller M., and Suri N. A control theory approach for analyzing the effects of data errors in safety-critical control systems. In *Proceedings of Pacific Rim International Symposium on Dependable Computing (PRDC'02)*, Tsukuba, Japon, Décembre 2002.

[Åström and Wittenmark, 1997] K. J. Åström and B. Wittenmark. *Computer Controlled Systems: Theory and Design Third Edition.* Prenctise Hall, New Jersey, 2 edition, 1997.

[Ayyagari and Ray, 1993] A. Ayyagari and A. Ray. Modelling and analysis of a data communication protocol for integrated control of advanced aircraft. *Computer Communications*, 16(6):350–365, 1993.

[Bailey and Swarztrauber, 1990] D. H. Bailey and P. N. Swarztrauber. The Fractional Fourier Transform and Applications. Technical Report RNR-90-004, Berkley, Mail Stop T27-A, Moffett Field,CA 94035, Avril 1990.

[Balarin, 1999] F. Balarin. Polis: A design environment for control-dominated embedded systems, 1999.

[Barrenscheen, 1997] G. Barrenscheen. Analysis of the physical CAN bus layer. In *Proceedings of the 4th International CAN Conference*, pages 201–209, Octobre 1997.

[Basney, 2001] J. Basney. *Network and CPU Co-Allocation in High Throughput Computing Environments.* PhD thesis, University of Wisconsin-Madison, 2001.

[Bennett and Zhang, 1997] J. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. *IEEE/ ACM Transactions on Networking*, 5(5):675–689, 1997.

[Berson and Lindell, 1997] S. Berson and R. Lindell. An architecture for advance reservations in the internet, 1997.

[Blum and Juanole, 1999] I. Blum and G. Juanole. Comparing the networks CAN and ARINC 629 CP with respect to the quality of the service provided to an automatic control application. In *Fieldbus and Technologie Conference*, Septembre 1999.

[Blum, 2000] I. Blum. *Qualité de service dans les réseaux locaux industriels : modélisation et évaluation. Lien avec les performances d'applications de contrôle-commande.* PhD thesis, Université Paul Sabatier, Janvier 2000.

[Bonnet, 1999] C. Bonnet. *Architectures de référence Introduction aux systèmes temps réel.* Technip, 1999. Collection pédagogique de télécommunications.

[Borne *et al.*, 2000] P. Borne, C. Sueur, and P. Vanheeghe. *Automatique des systèmes échantillonnés.* Technip, 2000. Collection Sciences et Technologies.

[Braden *et al.*,] R. Braden, L. Zhang, L. Berson, L. Herzog, and S. Jamin. Resource reservation protocol (RSVP) - version 1 functional specification, RFC 2205.

[Castelpietra *et al.*, 1999a] P. Castelpietra, F. Simonot-Lion, and Y-Q. Song. Rapport intermédiaire du contrat carosse. Technical report, INPL, LORIA TRIO, Mai 1999.

[Castelpietra *et al.*, 1999b] P. Castelpietra, F. Simonot-Lion, Y.-Q. Song, and M. Attia. Microscopic modeling of support system for in-vehicle embedded systems. *IFIP Distributed and parallel embedded systems*, Janvier 1999.

[Castelpietra *et al.*, 2002] P. Castelpietra, F. Simonot-Lion, Y-Q. Song, and M. Attia. Analysis and simulation methods for performance evaluation of a multiple networked embedded architecture. *IEEE Transactions on Industrial Electronics*, Décembre 2002.

[Cervin *et al.*,] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K. Årzén. Jitterbug and TrueTime: Analysis Tools for Real-Time Control Systems, booktitle = Proceedings of the 2nd Workshop on Real-Time Tools, year = 2002, pages = , month = Juin,.

[Cervin *et al.*, 2003a] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K. Årzén. Analysis and simulation of controller timing. *IEEE Control Systems Magazine Electronics*, Décembre 2003.

[Cervin *et al.*, 2003b] A. Cervin, D. Henriksson, B. Lincoln, and K. Årzén. Analysis and Simulation of Controller Timing. In *Proceedings IEEE Control Systems Magazine*, Décembre 2003.

[Chapellat and Dahleh, 1992] H. Chapellat and M. Dahleh. Analysis of time-varying control strategies for optimal disturbance rejection and robustness, 1992.

[Chavez, 2000] L. Chavez. *Qualité de Service et Ordonnancement dans les systèmes de Communication Temps Réel.* PhD thesis, INPL, Octobre 2000.

[Colin and Puaut, 2000] A. Colin and I. Puaut. Worst case execution time analysis for a processor with branch prediction. *Real-Time Systems*, 18(2/3):249–274, 2000.

[Cottet *et al.*, 2000] F. Cottet, J. Delacroix, L. Kaiser, and Z. Mammeri. *Architectures de référence Introduction aux systèmes temps réel.* Hermes, 2000.

[Courrier, 1998] M. Courrier. *Modélisation de composants matériels et exécutifs en vue de la validation d'architectures opérationnelles par évaluation de performances.* PhD thesis, INPL, Novembre 1998.

[Cunha *et al.*, 2001] J. Cunha, R. Maia, Rela, and J. M. Silva. A study of failure models in feedback control systems. In *Proceedings of the International Conference on Dependable System and Network*, Goteborg, Suéde, Juillet 2001.

[Degermark *et al.*, 1995] M. Degermark, T. Kohler, S. Pink, and O. Schelen. Advance reservations for predictive service. In *Network and Operating System Support for Digital Audio and Video*, pages 3–15, 1995.

[Delgrossi and Berger,] L. Delgrossi and O. Berger. Internet stream protocol version 2 (st2), rfc 1819.

[Demers *et al.*, 1990] A. Demers, S. Keshav, and Shenker S. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, 2(3):157–173, Septembre 1990.

[Dilger *et al.*, 1998] E. Dilger, T. Füher, B. Müller, and S. Poldena. The X-by-Wire concept : Time triggered information exchange and fail silence support by new system service. In *Proceedings of SAE Internatinal Congress and Exhibition*, Décembre 1998.

[Durand, 1998] E. Durand. *Description et vérification d'architecture d'application temps réel : CLARA et les réseaux de Petri temporels.* PhD thesis, Université de Nantes, Septembre 1998.

[Eker, 1999] J. Eker. *Flexible Embedded Control Systems. Design and Implementation.* PhD thesis, Department of Automatic Control, Lund Institute of Technology, Sweden, Décembre 1999.

[Elkhoury and Törngren,] J. Elkhoury and M. Törngren. Towards an toolset for architectural design of distributed real-time control systems. In *Proceedings of IEEE Real-time Systems Symposium 2001 (RTSS 2001)*, Décembre.

[Gaujal and Navet, 2001] B. Gaujal and N. Navet. Fault confinement mechanisms on CAN : Analysis and improvements. In *Proceedings of the 14th FeT IFAC Conference*, Novembre 2001.

[Gehin and Bayart, 1995] A.L. Gehin and M. Bayart. Operating modes of a distributed intelligent automated production system. In *Proceedings of IEEE int. Systems, Man and Cybernetics 1995*, Vancouver, Canada, Octobre 1995.

[Gäfvert *et al.*, 2003] M. Gäfvert, B. Wittenmark, and Ö. Askerdal. On the effect of transient data errors in controller implementations. In *Proceedings of American Control Conference 2003 (ACC'03)*, Denver, Etats Unis, Juin 2003.

[Goodwin *et al.*, 2001] G. Goodwin, S. Graebe, and M. Salgado. *Control System Design.* Prenctice Hall International, 2001.

[Goyal and Vin, 1997] P. Goyal and H. Vin. Generalized guaranteed rate scheduling algorithms: a framework. *IEEE/ACM Transactions on Networking*, 5(4):561–571, 1997.

[Graefe and McKenna, 1993] G. Graefe and W. McKenna. The volcano optimizer generator: Extensibility and efficient search. In *ICDE*, pages 209–218, 1993.

[Gross and Harris, 1985] D. Gross and C.M. Harris. *Fundamentals of Queing Theory 2nd Edition.* Wiley, 1985.

[Grossglauser and Keshav, 1996] M. Grossglauser and S. Keshav. On CBR service. In *Proceedings of IEEE Conference on Computer Communication (INFOCOM 1996) (1)*, pages 129–137, San Fransisco, Mai 1996.

[GuO, 2000] *Networks with advance reservations: The routing perspective*, 2000.

[Henriksson *et al.*, 2002] D. Henriksson, A. Cervin, and K.-E. Årzén. TrueTime: Simulation of control loops under shared computer resources. In *Proceedings of the 15th IFAC World Congress on Automatic Control*, Barcelona, Spain, Juillet 2002.

[Herdman, 1995] R. Herdman. Distributed simulation of combat. Technical Report OTA-BP-ISS-151, Princeton, Septembre 1995.

[Hiller, 2002] M. Hiller. *A Software Profiling Methodology for Design and Assessment of Dependable Software.* PhD thesis, University of Technology of Chalmers, Octobre 2002.

[Honeywell, 1998] Honeywell. Metah user's manual, 1998.

[Hristu, 2000] D. Hristu. Stabilization of LTI systems with communication constraints. In *Proceedings of American Control Conference (ACC 2000)*, Janvier 2000.

[IEC, 1997] IEC. Norme iec 61508, 1997.

[Jacobson, 2002] J. Jacobson. Project palbus validation of dependable bus system., 2002.

[Jones *et al.*, 1997] Michael B. Jones, Daniela R., and Rosu M. CPU reservations and time constraints: Efficient, predictable scheduling of independent activities. In *Symposium on Operating Systems Principles*, pages 198–211, 1997.

[Juanole and Blum, 1999] G. Juanole and I. Blum. Influence de fonctions de base (communication-ordonnancement) des systèmes distribués temps-réel sur les performances d'applications de contrôle-commande. In *7ème Colloque Francophone sur l'Ingénierie des Protocoles (CFIP)*, Avril 1999.

[Jumel and Simonot-Lion, 2002] F. Jumel and F. Simonot-Lion. Management of 'anytime' tasks in 'real time's applications. In *Procedings of XIV Workshop on Supervising and Diagnostics of Machining Systems*, Karpacz, Pologne, Mai 2002.

[Jumel *et al.*, 2002] F. Jumel, N. Navet, and F. Simonot-Lion. Nouvelles politiques pour la réservation explicite de ressources en avance. *Technique et Science Informatiques, Numéro spécial sur le temps réel*, Août 2002.

[Jumel, 1999] F. Jumel. *Génération automatique de modèles d'application Temps-Réel.* PhD thesis, LORIA/INPL, Septembre 1999.

[Karsten *et al.*, 1999] M. Karsten, N. Berier, L. Wolf, and R. Steinmetz. A Policy-Based Service Specification for Resource Reservation in Advance. In *Proceedings of the International Conference on Computer Communications (ICCC'99), Tokyo, Japan*, pages 82–88, Septembre 1999.

[Kim and Nahrstedt, 2000] K. Kim and K. Nahrstedt. A resource broker model with integrated reservation scheme. In *IEEE International Conference on Multimedia AND Expo (II)*, pages 859–862, 2000.

[Kim and Shin,] H. Kim and K.G. Shin. On the maximum feedback delay in a linear/nonlinear control system with input disturbances caused by controller-computer failures. *IEEE Trans. on Control Systems Technology.*

[Kim *et al.*, 1996] Y. Kim, H. Kwon, and H. Park. Stability and scheduling method for network-based control systems. In *Proceedings of the 22nd IEEE International Conference on Industrial Electronics, Control, and Instrumentation*, Mai 1996.

[Klein *et al.*, 1994] M. H. Klein, T. Ralya, B. Pollak, R. R. Obenzaand, and M. G. Harbour. *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems.* Kluwers Academic Publishers, 1994.

[Kleinrock, 1975a] L. Kleinrock. *Queueing systems : Vol. 1 : Computer applications.* Wiley, 1975.

[Kleinrock, 1975b] L. Kleinrock. *Queueing systems : Vol. 2 : Computer applications.* Wiley, 1975.

[Kocick and Sorel, 2000] R. Kocick and Y. Sorel. De la modélisation à la réalisation : réduction du cycle de développement des applications temps réel distribuées. In *Proceedings of the 8th conference on Real-time and embedded systems*, Paris, France, Mars 2000.

[Laprie, 1995] J.-C. Laprie. *Guide de la sûreté de fonctionnement.* Cépadués-Editions, 1995.

[Lars and Ralf, 1997] C. W. Lars and S. Ralf. Concepts for resource reservation in advance. *Multimedia Tools and Applications*, 4(3):255–278, 1997.

[Lavarenne and Sorel, 1997] C. Lavarenne and Y. Sorel. Modèle unifié pour la conception conjointe logiciel-matériel. *Revue Traitement du Signal, Numéro spécial Adéquation Algorithme Architecture*, 14(6), Août 1997.

[Le Boudec, 1992] J. Y. Le Boudec. The asynchronous transfer mode: a tutorial. *Computer Networks and ISDN Systems*, 24:279–309, 1992.

[Lee *et al.*, 1994] Y. Lee, Y. Kim, and W. Kwon. Generalized predective control for time-varying systems. In *Proceedings of the Asian Control Conference*, Tokyo, Japon, Juillet 1994.

[Leigh, 1985] J. R. Leigh. *Applied Digital Control : Theory Design and Editorial.* Prentice Hall International, 1985.

[Lian *et al.*, 1999] F. Lian, J. Moyne, and D. Tilbury. Performance evaluation of control networks: Ethernet, controlnet, and devicenet, 1999.

[Liu, 2000] J. Liu. *Real-Time Systems.* Prentice Hall International, 2000.

[Mari and Schott, 2000] J.-F. Mari and R. Schott. *Probabilistic AND Statistical Methods in Computer Science.* Kluwer Academic, 2000.

[Martí *et al.*, 2001a] P. Martí, J. M. Fuertes, and G. Fohler. An integrated approach to real-time distributed control systems over fieldbuses. In *Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA2001)*, pages 39–48, Antibes, France, Octobre 2001.

[Martí *et al.*, 2001b] P. Martí, J. M. Fuertes, G. Fohler, and K. Ramamritham. Jitter compensation for real-time control systems. In *Proceedings IEEE Real-Time Systems Symposium*, pages 39–48, London, UK, Dec 2001.

[Martí *et al.*, 2002a] P. Martí, G. Fuertes, J. M.AND Fohler, and K. Ramamritham. Improving quality-of-control using flexible timing constraint : Metric and scheduling issues. In *Proceedings Real-Time Systems Symposium*, pages 91–100, Austin, Texas, Décembre 2002. IEEE.

[Martí *et al.*, 2002b] P. Martí, R. Villà, J. M. Fuertes, and G. Fohler. Control loop performance analysis over networked control. In *Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society*, pages 39–48, Sevilla, Espagne, Novembre 2002. IEEE.

[Martí *et al.*, 2002c] P. Martí, R Villà, J. M. Fuertes, and G. Fohler. A discrete-time controller design method to tolerate non-equidistant sampling and actuation. Technical Report ESAII-RR-00-17, IT Lünd, Janvier 2002.

[Mercer *et al.*, 1994] C. Mercer, S. Savage, and H. Tokuda. Processor capacity reserves: Operating system support for multimedia applications. In *International Conference on Multimedia Computing and Systems*, pages 90–99, Mai 1994.

[Migge, 1999] J. Migge. *Scheduling of recurrent tasks on one processor: A trajectory based Model.* PhD thesis, Université Nice Sophia-Antipoli, Novembre 1999.

[Moon and Kwon, 1996] Y. Moon and K. Kwon. Robust stability of linear systems with structured delayed perturbations. In *Proceedings of Sensign Instrument Control Engineering (SICE'96)*, Tottori, Japan, Juillet 1996.

[Moon *et al.*, 1998] Y. S. Moon, P. Park, and W. Kwon. Delay-dependent robust stabilization of uncertain time-delay systems. In *Proceedings of IFAC conference System Structure AND Control*, Nantes, France, Juillet 1998.

[Mowbray *et al.*, 1998] M. Mowbray, G. Karlsson, and T. Kohler. Capacity reservation for multimedia traffics. *Journal of Distr. Syst. Eng.*, Août 1998.

[Navet *et al.*, 2000] N. Navet, Y.-Q. Song, and F. Simonot. Worst-case deadline failure probability in real-time applications distributed over can controller area network). *Journal of Systems Architecture*, 46(7), Août 2000.

[Nilsson *et al.*, 1996] J. Nilsson, B. Bernhardsson, and B. Wittenmark. Stochastic analysis of real-time systems with random time delays. In *Proceedings of IEEE Real-Time Systems*, pages 267–272, San Fransisco, USA, Juin 1996.

[Nilsson *et al.*, 1998] J. Nilsson, B. B. Bernhardsson, and B. Wittenmark. Stochastic analysis and control of real-time systems with random time delays. *Automatica*, 34(1):57–64, Août 1998.

[Nilsson *et al.*, 1999] J. Nilsson, B. Bernhardsson, and B. Wittenmark. Pi and pid controller tuning rules for time delay processes. In *Proceedings of the Irish Signals and Systems Conference*, Galway, Ireland, juin 1999.

[Nilsson, 1996] J. Nilsson. *Analysis and Design of Real-Time ystems with Random Delays.* PhD thesis, IT Lünd, Mai 1996.

[Nilsson, 1998] J. Nilsson. *Real-Time Control Systems with Delays.* PhD thesis, IT Lünd, Février 1998.

[O'Dwyer, 1996] A. O'Dwyer. Time delayed process model parameter estimation: A classification of techniques. Technical Report 4583, Dublin institute of Technology, Janvier 1996.

[Ogata, 1987] K. Ogata. *Discrete-time control systems.* Prenctise Hall, 1987.

[Otanez *et al.*, 2002] G. Otanez, R. Moyne J., and Tilbury D. Using deadbands to reduce communication in networked control systems. In *Proceedings of the 2002 American Control Conference*, Anchorage, Alaska, Mai 2002.

[Parekh and Gallager, 1993] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Trans. on Networking*, 1(3):344–357, Juin 1993.

[Paret, 1996] D. Paret. *Le bus CAN Controller Area Network.* Dunod, 1996.

[Park *et al.*, 1995] H. Park, S. Kim, and W. Kwon. Model and stability of hybrid system. In *Proceedings of DIMACS Workshop on Hybrid System*, NJ, USA, Mai 1995.

[Park *et al.*, 1997] P. Park, Y. Soo Moon, and W. Kwon. Output feedback linear quadratic regulation for input-delayed systems. In *Proceedings of 2nd Asian Control Conference*, Seoul, Corée, Juillet 1997.

[Poledna *et al.*, 2000] S. Poledna, A. Burns, A. Wellings, and P. Barrett. Replica determinism and flexible scheduling in hard real-time dependable systems. *IEEE Transactions on Computers*, 49(2):100–111, 2000.

[Puaut, 2001] I. Puaut. *Supports d'exécution à temps de réponse contrant tolérants aux fautes, Habilitation à diriger des recherches.* PhD thesis, Université de Rennes I, Novembre 2001.

[Ramanathan, 1999] P. Ramanathan. Overload management in real-time control applications using (m,k) firm guarantee. *IEEE Transaction on parallel and distributed systems*, 10(6), 1999.

[Ray, 1994] A. Ray. Output feedback control under randomly varying distributed delays. *Journal of Guidance, Control and Dynamics*, 17(4), 1994.

[Redell and Sanfridson, 2002] O. Redell and M. Sanfridson. Best-case response time analysis of fixed priority scheduled tasks. In *Proceedings of Int. Euromicro Conference on Real-Time Systems*, Vienne, Autriche, juin 2002.

[Redell and Törngren, 2002] O. Redell and M. Törngren. Calculating exact worst case response times for static priority scheduled tasks with offsets and jitter. In *Proceedings of Eighth IEEE Real-Time and Embedded Technology AND Applications Symposium*, Jose, California, Septembre 2002.

[Redell, 1996] O. Redell. Global scheduling in distributed real-time computer systems, an automatic control perspective. Technical report, Dept. of Machine design KTH, Janvier 1996.

[Regehr, 2003] J. Regehr. Some guidelines for proportional share CPU scheduling in general-purpose operating systems. In *Work in progress session of the 22nd IEEE Real-Time Systems Symposium (RTSS 2001)*, London, UK, juillet 2003.

[Richard, 2001] P. Richard. On the complexity of scheduling real-time tasks with self-suspension on one processor. In *Proceedings of Int. Euromicro Conference on Real-Time Systems*, Porto, Portugal, Décembre 2001.

[Ross, 1995] K. Ross. *Multiservice Loss Models for Broadband Telecommunication Networks.* Springler-Verlag, 1995.

[Russell and Zilberstein, 1991] S. Russell and S. Zilberstein. Composing real-time systems. In John Mylopoulos and Ray Reiter, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence (IJCAI-91)*, pages 212–217. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991.

[Sanfridson, 1999] M. Sanfridson. Survey of techniques for handling timing problems in distributed control, 1999.

[Sanfridson, 2000] M. Sanfridson. Timing problems in distributed real-time computer control systems. Technical Report TRITA-MMK 2000:11, IT Lünd, Novembre 2000.

[Santos-Marquez *et al.*, 2003] R. Santos-Marquez, N. Navet, and F. Simonot-Lion. Frame packing under real-time constraints. In *Proceedings of the 5th Conference on Fieldbus Technology (Fet'2003)*, Aveiro, Portugal, 2003.

[Schelén and Pink, 1998] O. Schelén and S. Pink. Resource sharing in advance reservation agents. *Journal of High Speed Networks, Special issue on Multimedia Networking*, 7(3-4), 1998.

[Schill *et al.*, 1997] A. Schill, S. Kuhn, and F. Breiter. Resource reservation in advance in heterogeneous networks with partial ATM infrastructures. In *Proceedings of Computer Communication Conference (INFOCOM 1997)*, pages 611–618, 1997.

[Schill *et al.*, 1998] A. Schill, F. Breiter, and S. Kuhn. Design and evaluation of an advance reservation protocol on top of rsvp. In *Proceedings of the 4th International Conference. (IFIP 1998)*, pages 23–40. Chapman AND Hall, 1998.

[Schott and Louchard, 1997] R. Schott and C. Louchard, G. Kenyon. Data structures' maxima. *SIAM journal on computing*, 26(4), 1997.

[Sha *et al.*, 1996] L. Sha, R. Rajkumar, and M. Gagliardi. Evolving dependable real-time systems. In *IEEE Aerospace Applications Conference. Proceedings*, pages 335–46, Aspen, Etats Unis, Février 1996. IEEE New York, NY, USA.

[Shin and Kim,] K.G. Shin and H. Kim. Derivation and application of hard deadlines for real-time control systems.

[Smith, 1959] O. Smith. A controller to overcome dead time. *ISA journal*, 6(2), 1959.

[Song *et al.*, 1999] Y.-Q. Song, F. Simonot-Lion, and N. Navet. De l'évaluation de performances du système de communication à la validation de l'architecture opérationnelle. In *Ecole d'été temps réel*, Poitier, France, Septembre 1999.

[Sorel, 1996] Y. Sorel. Real time embedded image processing applications using the a3 methodology. In *Proceedings of IEEE International Conference on Image Processing (ICIP'96)*, Lausanne, Suisse, Septembre 1996.

[Stankovic *et al.*, 1998] J. Stankovic, M. Spuri, K. Ramamritham, and G. Buttazzo. *Deadline Scheduling for real-time systems: EDF and related algorithms.* Kluwers Academic Publisher, 1998.

[Stoica *et al.*, 1996a] I. Stoica, H. Abdel-Wahab, and K. Jeffay. On the duality between resource reservation and proportional share resource allocation. Technical Report TR9619, University of California, Novembre 1996.

[Stoica *et al.*, 1996b] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. Baruah, J. Gehrke, and C. Plaxton. A proportional share resource allocation algorithm for real-time, time-shared systems. In *IEEE Real-Time Systems Symposium*, Décembre 1996.

[Trinquet and Elloy, 1999] Y. Trinquet and J.-P. Elloy. Les systèmes d'exploitation temps réel : les principes. *Techniques de l'Ingenieur*, R508, Février 1999.

[Trinquet, 2000] Y. Trinquet. *Noyau d'exécutifs pour la commande de systèmes.* traité EGEM, 2000.

[Törngren *et al.*, 2001] M. Törngren, R. Elkoury, J. Sandfridson, and M. Redell. Modelling and simulation of embedded computer control systems : Problem formulation. Technical Report ISSN 1400-1179, KTH Stockholm, Septembre 2001.

[Törngren, 1995] M. Törngren. *Modelling and Design of Distributed real time control applications.* PhD thesis, KTH, Dept of Machine Design, Juillet 1995.

[Törngren, 1996] J. Törngren, M. Wikander. A decentralization methodology for real-time control applications. *Journal of Control Engineering Practice*, 4(2), Février 1996.

[Vega, 1996] I. Vega. *Modèles de Coopération et de Communication entre Processus Temps Réel Répartis.* PhD thesis, INPL, Septembre 1996.

[Villemur, 1988] A. Villemur. *Sûreté de fonctionnement des systèmes industriels.* Editions Eyrolles, 1988.

[Vinter *et al.*, 2001] J. Vinter, J. Aidemark, P. Folkesson, and J. Karlsson. Reducing critical failures for control algorithms using executable assertions and best effort recovery. In *Proceedings of the International Conference on Dependable System and Network (DSN'01)*, Goteborg, Suéde, Juillet 2001.

[Waldspurger, 1995] C. A. Waldspurger. Lottery and stride scheduling: Flexible proportional-share resource management. Technical Report MIT/LCS/TR-667, MIT, 1995.

[Wittenmark *et al.*, 1998] B. Wittenmark, B. Bastian, and J. Nilsson. Analysis of time delays in synchronous and asynchonous control loops. In *Proceedings of 37th IEEE Conference on Decision and Control*, Tampa, Finlande, Décembre 1998.

[Wolf *et al.*, 1995] L. C. Wolf, L. Delgrossi, R. Steinmetz, and S. Schaller. Issues of reserving resources in advance. *Lecture Notes in Computer Science*, 1018:28–48, 1995.

[Yao *et al.*, 1995] F. Yao, A. Demers, and S. Shenker. A scheduling Model for reduced CPU energy. In *Proceedings of 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, Octobre 1995.

[Yook *et al.*, 1998] J. Yook, D. Tilbury, and N. Soparkar. Design methodology for distributed control systems to optimize performance in the presence of time delays. In *Proceedings of the American Control Conference*, Chicago, Illinois, Juin 1998.

[Yook *et al.*, 2000] J. Yook, D. Tilbury, and N. Soparkar. A design methodology for distributed control systems to optimize performance in the presence of time delays. In *Proceedings of the American Control Conference*, Chicago, Illinois, Juin 2000.

[Zhang *et al.*, 2001] W. Zhang, M. S. Branicky, and S. M. Philips. Stability of networked control systems. *IEEE Control System Magazine,*, 21(1):84–99, 2001.

[Zilberstein and Russel, 1996] S. Zilberstein and S. Russel. Optimal composition of real-time systems. *Journal of Artificial Intelligence*, 82(1-2), 1996.

Annexe A

Les formalismes des systèmes continus

L'automatisation des systèmes, en particulier des systèmes continus, repose sur l'utilisation d'outils mathématiques spécifiques, de type traitement du signal, pour décrire ces systèmes et résoudre les problèmes de leur contrôle. Les phénomènes physiques qui régissent les systèmes continus sont généralement d'ordre mécanique, électrique, thermique et se représentent sous forme d'équations différentielles. Par exemple, si on considère un système chargé du déplacement d'un chariot, les équations générales de la mécanique régissent sa dynamique.

Soit v la vitesse, considérée ici comme grandeur descriptive du système, u la force de traction, b le coefficient de frottement (force de frottement : $-b.v$) et m la masse, la dynamique du système est donnée par le système suivant (relation fondamentale de la mécanique) :

$$\left\{ \begin{array}{c} mv' + bv = u \\ y = v \end{array} \right\}$$

Il est, en général, nécessaire de limiter la complexité des modèles, afin de pouvoir les utiliser. En particulier, plusieurs techniques, allant dans ce sens, reposent sur l'hypothèse que les systèmes sont régis par un système d'équations différentielles linéaires (il est cependant possible de prendre en compte l'existence éventuelle de quelques non linéarités comme une saturation ou un phénomène d'hystérésis cf A.3). De plus, la plupart de ces techniques supposent que les coefficients considérés sont invariants dans le temps. Dans le cas contraire, il faut utiliser des méthodes spécifiques, ou considérer plusieurs modes de fonctionnement pour chacun desquels, on peut garantir que les coefficients sont constants.

A.1 Représentation des systèmes continus

A.1.1 Utilisation de la transformée de Laplace

L'étude des systèmes se fonde essentiellement sur l'étude des systèmes d'équations différentielles à coefficients constants. La résolution des équations différentielles linéaires se ramenant à des fonctions exponentielles, on utilise la transformée de Laplace qui permet de simplifier considérablement les calculs. Cette fonction est définie pour des signaux mais peut s'appliquer également à la description des systèmes

La transformée de Laplace d'un signal $s(t)$ est donnée par

$$S(t) = \int_0^{+\infty} s(t)e^{-pt}dt$$

avec $s(t) = 0$ pout $t < 0$ (sous l'hypothèse, garantie dans le cas des systèmes considérés, que cette intégrale converge). Le schéma A.1 présente les transformées de Laplace associées à quelques signaux d'utilisation

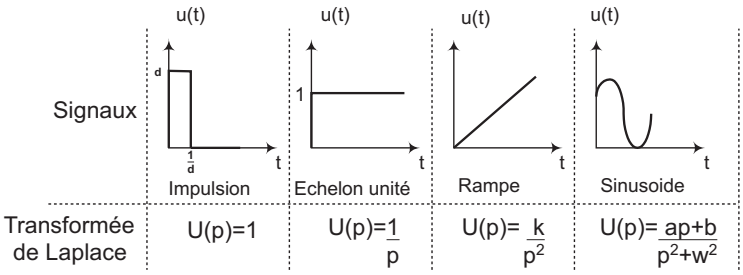


FIG. A.1 – Présentation des transformées de Laplace de quelques signaux

classiques.

A partir des transformées de Laplace des signaux d'entrée et de sortie, on peut obtenir une représentation du système :

$$H(p) = \frac{S(p)}{E(p)}$$

Les systèmes sont des systèmes physiques ; il doivent donc respecter le principe de causalité. La sortie du système ne peut pas dépendre d'une entrée future du système. Cette remarque a des conséquences importantes sur la forme des fonctions de transfert possibles. Il est intéressant de voir que $H(p)$ est indépendant de E et S et donne une définition générale du système. Seule la condition initiale qui n'apparaît pas dans la notation en p, manque pour définir complètement le système.

Il existe un lien entre cette transformée et les opérateurs de dérivation et d'intégration. Par exemple, le système suivant $s(t) = \int e(t)dt$ est défini par la fonction de transfert :

$$H(p) = \frac{1}{p}$$

En effet, si l'on considére une entrée en échelon (transformée $\frac{1}{p}$), la sortie obtenue est une rampe (transformée $\frac{1}{p^2}$) d'où $H(p) = \frac{1}{p}$

Même si la transformée de Laplace d'un signal ne peut être rigoureusement considérée comme un opérateur, on peut voir que $\frac{1}{p}$ correspond à un intégrateur et p à une dérivation.

Le système cité précédemment (chariot) a donc une fonction de transfert qui peut être obtenue à partir des équations différentielles qui régissent sa dynamique

d'où

$$H(p) = \frac{Y(p)}{U(p)} = \frac{1}{(mp + b)}$$

Une propriété très intéressante des fonctions de transfert est liée à la grande simplicité dans la mise en cascade ou en parallèle de différents systèmes ; la fonction de transfert du système composé est alors obtenue simplement par un produit (composition en série) ou une somme (composition en parallèle) des fonctions de transferts des sous-systèmes (voir figure A.2).

Il est toujours possible d'inverser une transformée de Laplace et de connaître la réponse du système à un certain signal d'entrée.

$$f(t) = \int_{c-j\infty}^{c+j\infty} F(p)e^{pt}dt$$

Connaissant la fonction de transfert d'un système et la transformée de Laplace du signal d'entrée, on obtient par un calcul simple la transformée de Laplace du signal de sortie, on peut alors obtenir la sortie du système

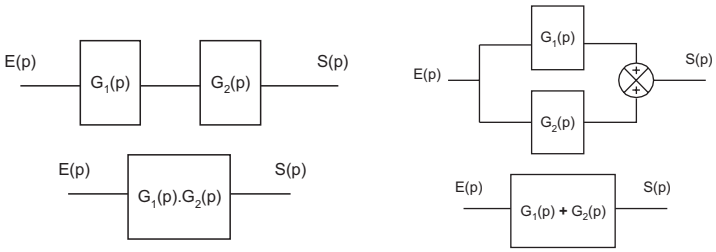


FIG. A.2 – Compositions de deux sous-systèmes

grâce à des abaques ou par calcul.

On peut identifier la fonction de transfert d'un système, soit par expérimentation, soit par calcul et modélisation des propriétés physiques. Souvent la recherche expérimentale donne de très bons résultats, surtout lorsque l'on cherche à obtenir un modèle simplifié du système.

Une bonne façon théorique d'obtenir la fonction de transfert d'un système est de regarder la réponse du système à une impulsion. En effet, comme la transformée d'une impulsion est une constante, on obtient directement la fonction de transfert du système. Dans la réalité, cette technique n'est pas applicable car la création d'une impulsion en entrée nécessite une énergie infinie, On préfère donc étudier la réponse à un échelon qui donne déjà des informations utiles sur le système.

A.1.2 Représentation sous forme de matrice d'état

Cette représentation particulièrement adaptée aux systèmes MIMO (plusieurs entrées et plusieurs sorties) peut aussi être utilisée dans le cadre des systèmes SISO (une entrée et une sortie).

On considère le système mis sous la forme suivante :

$$\left\{ \begin{array}{l} x'(t) = A(t)x(t) + B(t)u(t) \\ s(t) = C(t)x(t) + D(t)u(t) \end{array} \right)$$

où x représente le vecteur d'état du système, u le vecteur d'entrée et

A, B, C, D les matrices associées au système (elles sont constantes sous une hypothèse de linéarité mais peuvent ne pas être constantes)

La réponse du système est obtenue grâce à :

$$x(t) = e^{At}x(0) + \int_0^t e^{At-\tau}Bu(\tau)d\tau$$

On peut calculer à partir de ces matrices, certaines propriétés fondamentales du système comme

– l'atteignabilité, mesure de l'ensemble des états atteignables

– la commandabilité, représentation des capacités de commande du systéme

– l'observabilité, capacité de pouvoir calculer l'état interne du système à partir de ses sorties.

A.1.3 Représentation fréquentielle

Une autre technique de modélisation repose sur l'utilisation de la réponse d'un système à un ensemble de fonctions sinusoïdales, en particulier, sur l'étude du régime permanent obtenu pour des sollicitations d'entrée sinusoïdales de fréquences différentes .

Dans le cas des systèmes linéaires, on obtient une réponse sinusoïdale, avec un déphasage et un gain qui peuvent varier en fonction de la fréquence.

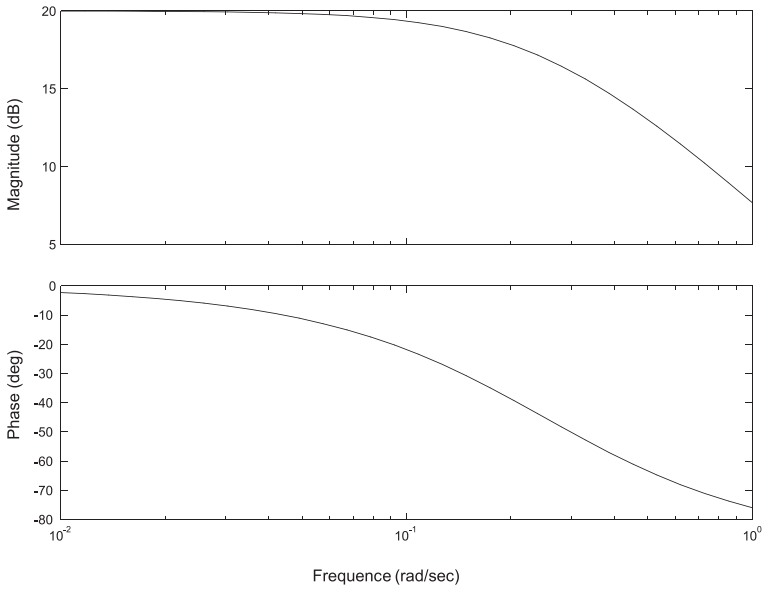


FIG. A.3 – Exemple de diagramme de Bode d'un système élémentaire

Ces courbes peuvent aussi être directement obtenues à partir des fonctions de transfert en posant $p = jw$ (qui correspond à la dérivée dans le cas d'une fonction sinusoïdale) et en étudiant le gain et le déphasage apportés en fonction de w.

Inversement, la connaissance de cette fonction (diagramme de Nyquist ou de Bode) permet de se ramener à la fonction de transfert. En fait, comme elle contient les mêmes informations, des techniques ont étés développées pour déduire directement les propriétés désirées de leur étude, sans passer par le calcul explicite de la fonction de transfert.

On peut même ramener l'étude d'un système dont on connaît la fonction de transfert à l'étude de son diagramme de Bode ou de Nyquist. Ces techniques ont été très développées dans le passé car elles permettent une résolution graphique des problèmes (sous forme d'abaques) très utile en l'absence des puissants moyens de calculs qui sont disponibles aujourd'hui.

A.1.4 Notions de pôles et de zéros

Propriétés des pôles et des zéros

On considère la fonction de transfert G d'un système linéaire, elle se met sous la forme d'une fraction rationnelle.

$$G(p) = \frac{A(p)}{B(p)}$$

On décompose le système en fraction rationnelle,

$$G(p) = \alpha \frac{\prod_{j=1}^{n}(p - z_j)}{\prod_{i=1}^{m}(p - p_i)}$$

$z_j$zéros

p_i pôles

Pour des raisons de causalité on a toujours $m \geq n$

Si on considère un échelon en entrée, la sortie est donnée par

$$S(p) = \frac{G(p)}{p}$$

soit

$$S(p) = \alpha \frac{\prod_{j=1}^{n}(p - z_j)}{p \cdot \prod_{i=1}^{n}(p - p_i)}$$

La décomposition en éléments simples en absence de pôles multiples donne

$$S(p) = \frac{a}{p} + \sum_i \frac{b_i}{p - r_i} + \sum_k \frac{c_k}{p - (\tau_k + j\omega_k)}$$

d'où

$$S(t) = au(t) + \sum_i b_i e^{r_i t} + \sum_l e^{\tau_l}(A_l cos(w_l t) + B_l sin(w_l t))$$

où $u(t)$ correspond à un échelon.

On comprend alors que le système ne tend pas vers un équilibre s'il possède une exponentielle avec un argument positif.

On en déduit le critère de stabilité suivant :

un système est stable s'il ne possède pas de pôles à partie réelle positive.

On utilise souvent ce critère sans calculer explicitement les racines, à l'aide de techniques graphiques ou qui reposent directement sur l'étude des coefficients des polynômes.

On remarque d'autre part que les réponses possibles d'un système linéaire sont principalement une superposition d'exponentielles, de sinusoïdes, et de sinusoïdes amorties exponentiellement. L'influence de chaque composante est liée à la proximité de l'argument exponentielle de 0. Ainsi les pôles dont la partie réelle est la plus proche de 0 constituent les pôles dominants du système. Le système suit le comportement de ces pôles dominants, les autres influant moins sur la réponse du système.

Le théorème de la décomposition des fractions rationnelles, permet de conclure que toute fraction rationnelle à coefficient réels est décomposable en fractions élémentaires du premier ordre et du deuxième ordre et éventuellement d'une partie polynomiale (impossible ici pour des raisons de causalité).

Pour des raisons de dominance, l'étude des systèmes se ramène donc généralement à l'étude d'un système du premier ordre ou du second ordre (cf. A.4).

A.2 Représentation des systèmes échantillonnés

A.2.1 Transformée de Fourrier (ou en z)

Un système numérique travaille sur un signal discret, c'est à dire une série de nombres.

Un formalisme fondé sur le même principe que la transformée en p existe pour les signaux échantillonnés, il s'agit de la transformé en z définie sur les séries.

Soit s une série :

$$s = \{s_0, s_1, ..., s_n\}$$

sa transformée en z est donnée par :

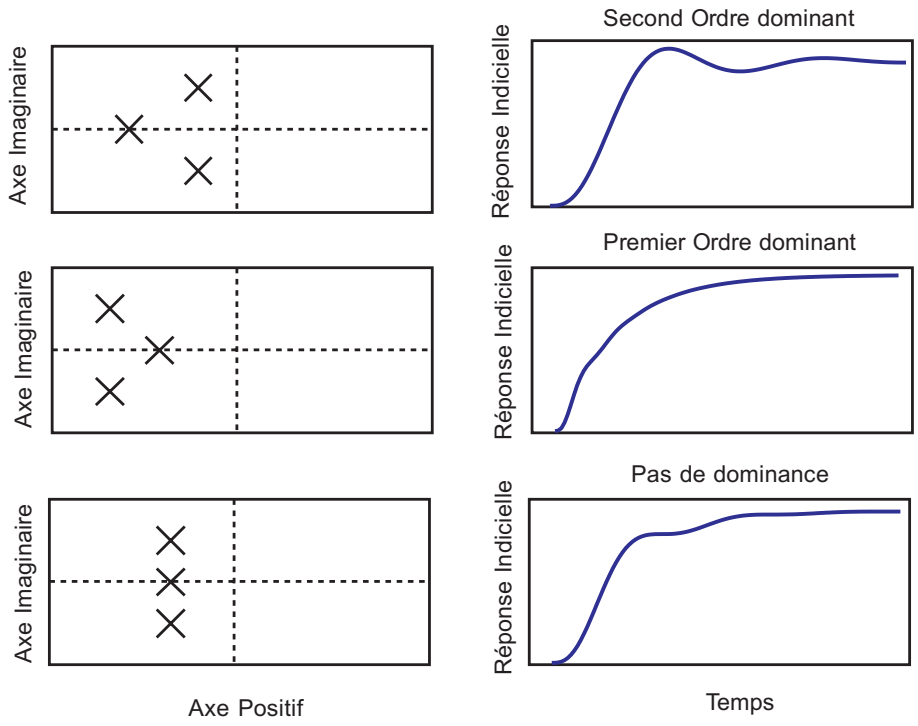


FIG. A.4 – Un exemple de la notion de dominance liée au placement des pôles

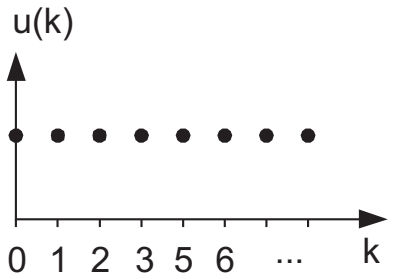


FIG. A.5 – Définition d'une série échelon

$$S(z) = \sum_{k=0}^{n} s_k z^{-k}$$

Par exemple dans le cas d'un échelon ($u_k = 1$ pour$k \geq 0$)

$$U(z) = \sum_{k=0}^{n} z^{-k} = \frac{1}{1 - z^{-1}} = \frac{z}{z - 1}$$

Dans le cas où la série correspond à l'échantillonnage d'un signal continu, z est lié à la fréquence d'échantillonnage par :

$$z = e^{pt}$$

Comme précedemment (A.1.1), on peut définir la transformée en z d'un système par le rapport :

$$G(z) = \frac{S(z)}{E(z)}$$

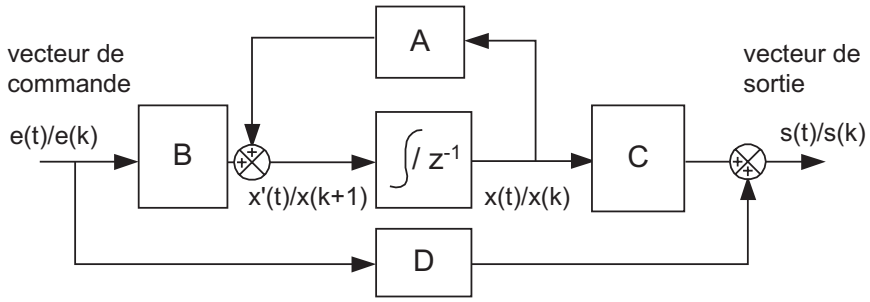


FIG. A.6 – Représentation schématique de la représentation d'état dans le cas des systèmes continu et discret

Dans le cas des systèmes linéaires, on obtient une représentation sous forme de fraction rationnelle

La dynamique du système est une fois de plus liée aux pôles et au zéros de la fonction de transfert .

A.2.2 Représentation d'état

On peut représenter un système discret sous une représentation d'état comme dans le cas des systèmes continus. La notion de dérivée est alors remplacée par la notion de décalage sur les échantillons

$$\left\{ \begin{array}{l} x(k+1) = A(k)x(k) + B(k)u(k) \\ s(k+1) = C(k)x(k) + D(k)u(k) \end{array} \right\}$$

L'évolution du système discret est alors donnée par :

$$x(k_0) = A^{k_0}x(0) + \sum_{i=0}^{k_0-1} A^{k_0-1-i}Bu(i)$$

A.2.3 Evolution du système entre deux échantillons

La représentation discrete des systèmes peut être obtenue directement par l'étude des signaux discrets où à partir de la repésentation continue du système .

Si on considère un système continu, dont la représentation est :

$$\left\{ \begin{array}{l} x'(t) = Ax(t) + Bu(t) \\ s(t) = Cx(t) + Du(t) \end{array} \right\}$$

sa représentation échantillonné sous forme d'état est de la forme :

$$\left\{ \begin{array}{c} x(k+1) = \Phi(t_{k+1}, t_k)x(k) + \Gamma(t_{k+1}, t_k)u(k) \\ s(k) = Cx(k) + Du(k) \end{array} \right\}$$

Le lien entre les 2 représentations se déduit de l'évolution du système continu en boucle ouverte entre 2 échantillons d'où le calcul des coefficients de la représentation d'état discrète.

$$\left\{ \begin{array}{c} \Phi(t_{k+1}, t_k) = e^{A(t_{k+1}-t_k)} \\ \Gamma(t_{k+1}, t_k) = \int_0^{t_{k+1}-t_k} e^{A\tau}Bd\tau \end{array} \right\}$$

Dans le cas des systèmes échantillonnés, on considére que les échantillons sont synchronisés sur la fréquence d'échantillonnage, il existe donc une relation temporelle importante entre les différents échantillons.

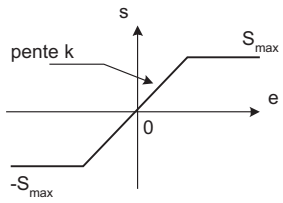


FIG. A.7 – Caractéristique d'une saturation

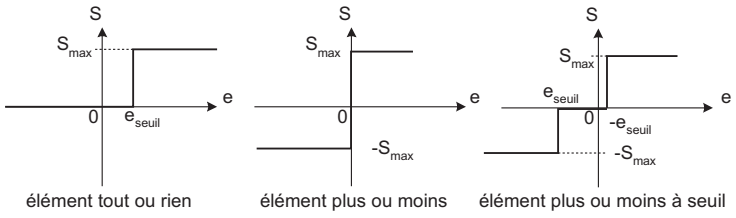


FIG. A.8 – Exemple de seuil

A.3 Modélisation des non-linéarités

Comme nous l'avons déjà indiqué, les techniques proposées, bien que fondées sur une hypothèse de linéarité des systèmes, peuvent prendre en compte l'existence de quelques non linéarités. Nous présentons ici quelques unes des non linéarités les plus fondamentales et inévitables dans l'étude des systèmes.

Saturation Le phénomène de saturation correspond à la limitation des valeurs de sortie d'un système. En particulier dans le cas de actionneurs, il existe une limite aux valeurs que peut prendre un signal de puissance en sortie d'un actionneur. Pour une vanne, par exemple, il existe 2 saturations correspondant à la fermeture complète et à l'ouverture complète.

Seuil Les seuils correspondent à l'existence de valeurs trop faibles pour avoir une influence (comme vaincre l'inertie du système), on parle aussi de zone morte. Dans le cas des actionneurs, certains systèmes ne peuvent prendre que deux valeurs, on parle alors d'actionneurs "tout ou rien" ou "plus ou moins". Dans certains cas , on a aussi trois états : plus, moins et zéro . La commande de tels actionneurs correspond alors à un système à seuil.

Quantification Le phénomène de quantification peut être vu comme une extension des systèmes "plus ou moins", à seuil, avec un grand nombre d'états possibles et un seuil entre chaque état.

Hystérésis L'hystérésis correspond à l'existence de réponses différentes pour une même entrée . La différence est liée au fait d'atteindre la valeur d'entrée de façon montante ou de façon descendante.

Retard Le retard correspond à un délai dans l'évolution du système. Le cas le plus simple de retard est lié à l'existence d'un délai de transport dans le système. L'évolution du systèmedépend alors de ce délai.

Bruit Le bruit n'est pas à proprement parlé une non linéarité mais il implique des écarts aux cas linéaires. Le bruit se caractérise par l'existence de paramètres variables, de variations dans les signaux, d'erreurs de mesure.

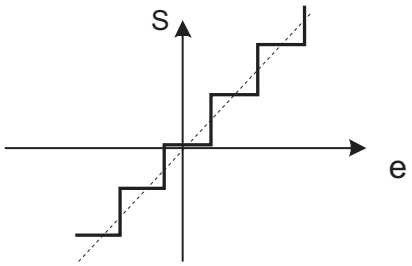


FIG. A.9 – Phénoméne de quantification,lié en particulier à la représentation des valeurs

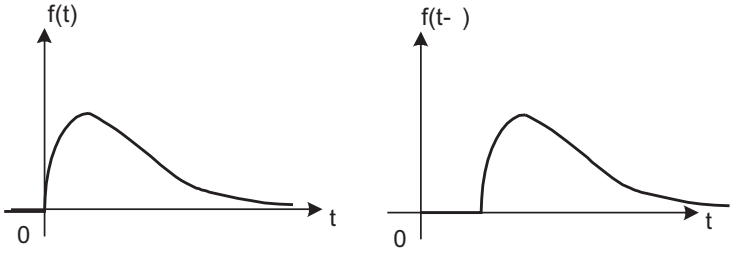


FIG. A.10 – Représentation temporelle d'un signal retardé

Remarques La plupart des non linéarités présentées sont très fréquentes. Certaines techniques de commande ont été prévues pour les prendre en compte. Dans les autres cas, il faut montrer que l'influence des non-linéarités est négligeable. La présence de ces non-linéarités peut être problématique aussi bien en boucle ouverte que fermée.

L'étude de la stabilité en présence de non linéarité fait l'objet de théories spécifiques comme par exemple la méthode du 1er harmonique (cf [Goodwin *et al.*, 2001]). On se contente en général d'étudier une loi de commande développée sans prendre en compte les non-linéarités et de montrer comment choisir les paramètres pour en minimiser l'influence. L'existence de non-linéarités conduit souvent à l'apparition de phénomènes gênants non désirés comme par exemple le phénomène de pompage qui correspond à une variation périodique de forte amplitude autour de la position d'équilibre.

A.4 Etude de la dynamique des systèmes usuels

Comme précisé en (A.1.4), l'évolution d'un grand nombre de systèmes se ramène à l'étude d'un système du premier ordre ou du second ordre. La connaissance de ces systèmes particuliers est donc essentielle pour comprendre les évolutions possibles des systèmes physiques. Les propriétés remarquables de systèmes du 1er ordre et du second ordre peuvent en outre être utiliser pour développer des techniques de contrôle ou ramener l'étude du comportement général du système à des cas particuliers.

Les systèmes du 1er ordre ont pour caractéristiques essentielles que leur comportement dépend directement de l'évolution d'une grandeur unique alors que pour les systèmes du second ordre, l'évolution est conditionnée par deux grandeurs (la grandeur de référence et sa dérivée). Cette remarque permet en particulier de comprendre que l'évolution d'un système du premier ordre ou du second ordre ne dépend que de son état courant (et pas de la façon dont il est arrivé à cet état). Cet état ne comporte qu'une grandeur pour les systèmes du premier ordre et deux pour ceux du second.

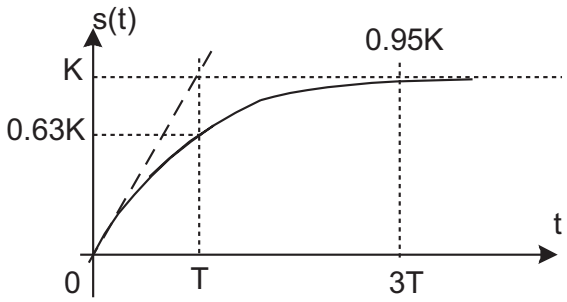


FIG. A.11 – Réponse d'un système du premier ordre à un échelon unité

A.4.1 Etude de Système 1er ordre

La fonction de transfert d'un premier ordre est de la forme

$$H(p) = \frac{K}{(1+Tp)}$$

La réponse à un échelon est donnée par A.11A.11 :

$$S(t) = K(1 - e^{-t/T})$$

K représente le gain du système et

T est la constante de temps du système, qui est une mesure de l'inertie du système.

Propriétés remarquables

- Le temps de réponse (défini sous forme d'un pourcentage vis à vis du régime stationnaire.) est donné simplement par $3 * T$ et ne dépend ni de K ni de la hauteur de l'échelon.
- L'asymptote à l'origine est une droite de pente $\frac{K}{T}$.

A.4.2 Etude de Système du 2ème ordre

Le Fonction de transfert sous forme normalisée est de la forme :

$$H(p) = \frac{w_n^2}{p^2 + 2\xi w_n p + w_n^2}$$

w_n est la fréquence propre du système et représente en particulier les différentes évolutions du système en fonction d'une excitation périodique.

ξ est le coefficient d'amortissement du système, qui caractérise l'importance des oscillations dans l'évolution du système

Un système du second ordre peut présenter des réponses trés diverses en fonction de ses paramètres :

Le cas le plus classique est une évolution oscillante amortie

Propriétés remarquables

- Le temps de montée (Le 1er instant où la sortie dépasse l'équilibre souhaité donné par une entrée indicielle)

$$t_p = \frac{\pi}{w_n\sqrt{1-\xi^2}}$$

- Le temps de réponse déduit en prenant en considération l'évolution pseudo-périodique

$$t_r = \frac{-ln(n\sqrt{1-\xi^2})}{\xi w_n}$$

- Le dépassement

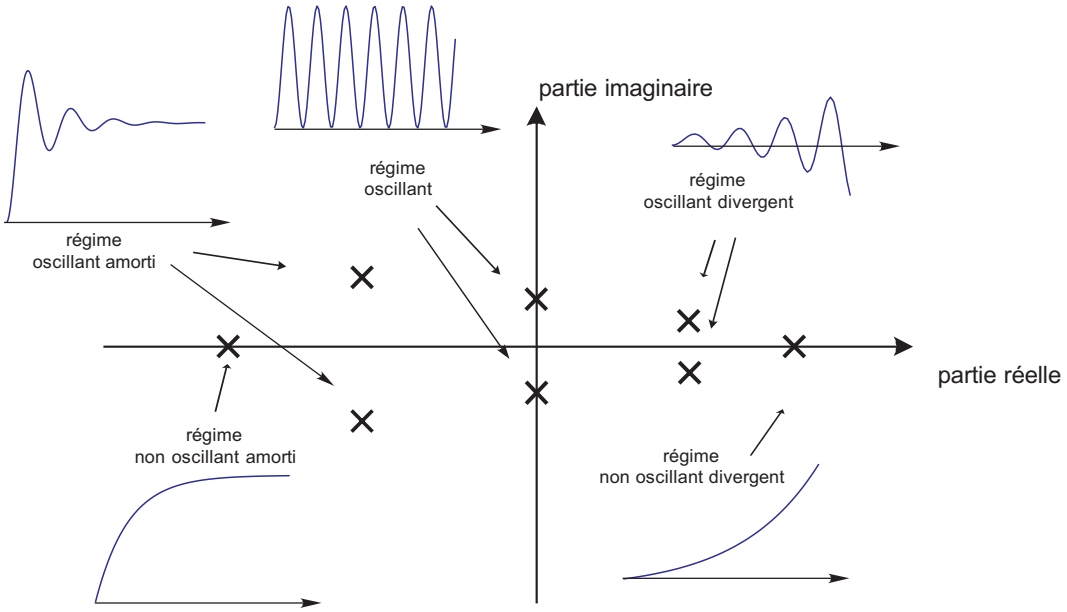


FIG. A.12 – Les différentes formes de réponses indicielles possibles pour un ssytème du premier ou second ordre

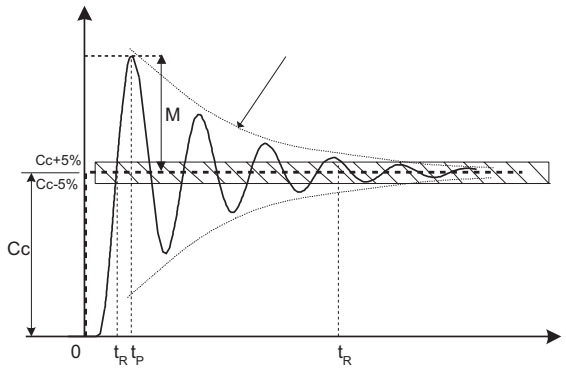


FIG. A.13 – Réponse d'un système du second ordre à un échelon unité

$$D_{\%} = e^{\frac{-\pi\xi}{\sqrt{1-\xi^2}}}$$

Annexe B

Architectures de commande

Les moyens permettant d'assurer la régulation ou l'asservissement des systèmes se décomposent en deux grandes familles reposant, d'une part sur la boucle ouverte et, d'autre part, sur la boucle fermée. Ces différents moyens peuvent être combinés en série (au même niveau d'intervention) ou hiérarchiquement. Des moyens spécifiques ont été développés pour le contrôle des systèmes dans des conditions particulières. Le schéma (B.1) présente la structure générale d'une boucle de contrôle, il est tout a fait possible de définir des contrôles qui n'utilisent que l'action corrective (Boucle fermée) ou que l'action directe (Boucle ouverte).

B.1 Système en boucle ouverte

Boucle directe La boucle directe correspond à l'ensemble des traitements qui permettent de transformer le signal de consigne en une commande adéquate à appliquer sur le système. La boucle directe la plus simple et la plus usuelle est une conversion proportionnelle de la consigne en une commande à appliquer sur les actionneurs.

Anticipation On peut suivre l'évolution de grandeurs qui interviennent avec un certain retard et adapter la commande vis à vis de cette évolution avant même que tout effet soit visible sur le sytème. L'anticipation est par exemple utilisée par l'automobiliste qui accèlére en apercevant une côte pour compenser la perte de vitesse qui résultera de la pente.

Prévision Les techniques de prévision se basent sur le présent et le passé du système pour évaluer son futur. Les techniques de prévision peuvent être déterministes (prise en compte d'une certaine inertie du système) ou probabilistes (estimation de l'évolution future des perturbations).

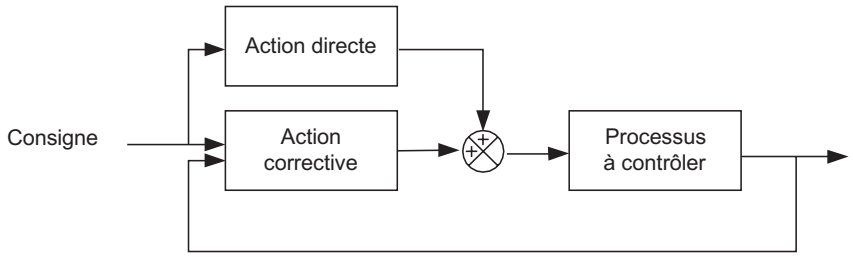


FIG. B.1 – Schéma générale d'une architecture de contrôle

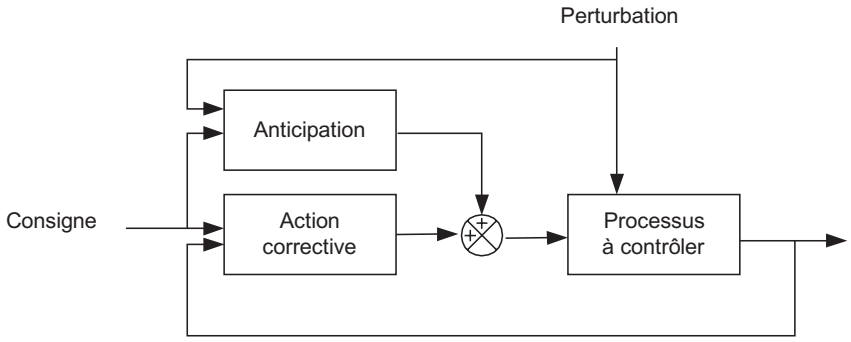


FIG. B.2 – Mise en place d'une boucle d'anticipation

B.2 Système en Boucle Fermée

Prétraitement Les prétraitements peuvent consister en la mise en place de filtre (pour supprimer le bruit sur les données) ou se révéler plus complexes et nécessiter énormément de puissance de calcul comme l'analyse de données en provenance d'une caméra.

Etude de la boucle fermée La boucle fermée constitue la base de la plupart des mécanismes de contrôle, il s'agit de boucler le système, c'est à dire de comparer la consigne et la sortie sur le système. On utilise donc l'écart entre la sortie désirée et la sortie obtenue en entrée du correcteur pour créer la commande qui est appliquée sur le système réglé. L'utilisation du bouclage procure de bonnes propriétés :

- Insensibilité au bruit ; cette propriété est très difficile à obtenir en boucle ouverte en raison de la difficulté d'une modélisation correcte du bruit.
- Insensibilité à des erreurs mineures de modélisation du système lors de la conception de la loi de commande.
- Dans certains cas, on peut rendre stable en boucle fermée, un système instable en boucle ouverte (en particulier si la stabilité est liée à la présence d'un intégrateur).
- Meilleure utilisation de la dynamique du système ; par comparaison à la boucle ouverte, on utilise une plage de valeurs de commandes bien plus grande, ce qui permet d'obtenir des temps de réponse bien plus faibles qu'en boucle ouverte (généralement un facteur supérieur à 5)

Par contre le bouclage peut entraîner l'apparition de nouveaux problèmes :

- Instabilité d'un système garanti stable en boucle ouverte. Cette notion est très importante, en particulier il faut comprendre que le bouclage n'implique pas la stabilité. Il ne suffit pas de chercher à rendre petit l'écart (sortie-consigne) pour assurer la stabilité, au contraire un système stable en boucle ouverte devient très facilement instable en boucle fermée.
- Apparition de dépassements ; le système répond plus vite mais en contrepartie, il peut dépasser la valeur de consigne voulue avant de se stabiliser.
- Enfin boucler un système ne suffit pas à supprimer l'existence possible d'une erreur statique (c'est-à-dire que la valeur obtenue n'est pas la valeur désirée).

Nous présentons ici la structure classique d'une régulation qui fait apparaître la notion de correcteur prenant en compte uniquement l'écart et non directement la consigne.

Le bouclage du système, se traduit simplement sur la fonction de transfert :

$$G(p) = \frac{G_{cd}(p)}{1 + G_{cd}(p)G_{cr}(p)}$$

Il faut noter que la fonction de transfert, lie l'entrée de consigne (et non l'écart) avec la sortie.

On remarque que le bouclage augmente l'ordre du système par la présence de la chaîne de retour.

D'après la remarque précédente sur les critères de stabilité (A.1.4), la stabilité est liée aux pôles de $G(p)$ c'est à dire aux racines de $1 + G_{cd}(p).G_{cr}(p)$.

Des techniques ont été développées pour conclure à la stabilité d'un système sans calculer explicitement les racines ou même la fonction de transfert en boucle fermée. Les plus utilisées sont :

– le critère de Routh

– l'utilisation du diagramme de Bode ou d'une variante (Nyquist, Black) (cf [Goodwin *et al.*, 2001]).

Le critère de Routh permet de déduire la stabilité du système à l'aide d'un calcul simple sur les coefficients du dénominateur. Le critère de Bode ou ses variantes donnent des techniques graphiques qui permettent d'étudier la stabilité de la boucle fermée en étudiant la réponse fréquentielle de la boucle ouverte.

L'étude de la réponse fréquentielle d'un système permet, en effet, de conclure à la stabilité du système. Il existe une relation entre le nombre de racines d'un polynôme et le nombre de quarts de tour apparents que fait sa représentation polaire autour de l'origine. On peut généraliser cette propriété à une fraction rationnelle. Le passage entre l'étude de la boucle ouverte et la stabilité de la boucle fermée est simple; il suffit d'étudier la boucle ouverte de représentation $A(p)B(p)$ autour du point $(0, -1)$, on obtient alors une étude des racines de $A(p)B(p) + 1$ qui est le dénominateur de la fonction de transfert en boucle fermée.

Ces quarts de tours impliquent donc une notion de déphasage. On définit la notion de marge de phase qui représente l'écart avec le premier quart de tour (qui implique une instabilité). De la même manière on définit la marge de gain comme le gain maximal que l'on peut ajouter en facteur dans la boucle ouverte, sans perdre la stabilité du système.

De la même manière que pour les pôles d'un système, il est possible de lier la dynamique du système à sa marge de phase. Ainsi, une marge de phase très importante peut être diminuée pour améliorer le temps de réponse du système en apportant un éventuel dépassement.

On peut remarquer également qu'il n'est pas possible de simplifier un pôle s'il entraîne une instabilité même si un zéro équivalent existe. Par contre, si le zéro n'entraîne pas d'instabilité, on peut l'annuler avec un zéro bien placé.

B.3 Différentes techniques

B.3.1 Action en cascade

La première classe de techniques correspond à la mise en cascade à la sortie du comparateur pour jouer sur l'écart. Malgré les apparences, le fait de ne considérer que l'écart pour calculer la commande à appliquer sur les actionneurs ne suffit pas à garantir la stabilité de système. De nombreuses techniques différentes appartiennent à cette classe. On trouve des techniques du monde continu, converties en numérique , ou des techniques "toutes numériques"

PID La plus connue de ces techniques est la régulation PID.

$$s(t) = K(e(t) + \frac{1}{T_i} \cdot \int^t e(s)ds + Td \cdot \frac{d(e(t))}{dt})$$

où K représente l'action proportionnelle du correcteur

T_i son action intégrale

et T son action dérivative.

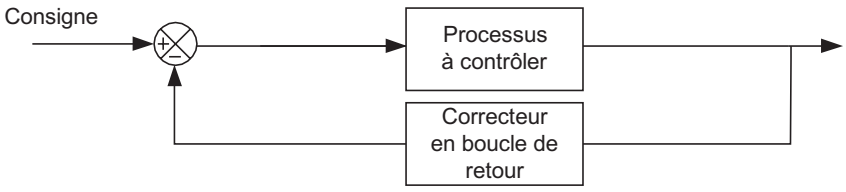


FIG. B.3 – Schéma d'un controlleur en boucle de retour

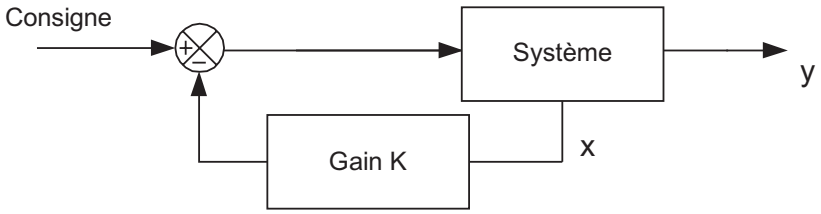


FIG. B.4 – Schéma d'un controlleur par retour d'état

On travaille sur l'écart au cas idéal (noté e) et on sépare l'action proportionnelle, intégrale et dérivative. L'intéret de cette technique est qu'elle est facilement compréhensible par tous, en particulier pour la façon de faire les réglages.

Correcteur de phase Ces techniques jouent directement sur la marge de phase (voir diagramme de Bode). On ajoute un correcteur qui augmente ou diminue la phase afin d'obtenir la réponse souhaitée du système.

B.3.2 Action en boucle de retour

Les techniques de correcteurs en cascade ne sont pas les seuls à être utilisées. On peut mettre en place une correction sur la boucle de retour. L'intérêt des ces techniques de "feedback" est qu'elles simplifient la mise en place de techniques plus efficaces

Retour d'état La technique du retour d'état est trés couramment utilisée. Une fois le système modélisé sous forme d'état, il s'agit de renvoyer cet état (constitué de plusieurs grandeurs) à partir de différents capteurs et de l'utiliser pour définir la commande (par produit matriciel). En apparence cette technique semble plus pauvre que la technique reposant sur le PID, pour laquelle on peut utiliser les dérivées et les intégrales de l'écart ; mais en fait, comme le vecteur d'état peut être composé d'un grand nombre de grandeurs, on a au moins la même puissance de commande qu'avec un PID.

Identification et Compensation Les différentes techniques présentées, doivent être paramétrées ; on peut utiliser des méthodes de réglages empiriques ou des méthodes reposant sur la définition des pôles et des zéros (A.1.4).

On trouve d'une part les techniques de compensation (ou placement de pôles) qui cherchent à minimiser l'influence destabilisatrice de certains pôles et à supprimer la lenteur apportée au système par d'autres.

Enfin les techniques d'identification, cherche à donner aux systèmes la dynamique souhaitée, en calculant les diverses corrections à apporter (à tout les niveaux dans le cas d'une régulation RST) afin d'obtenir une fonction de transfert du système régulé correspondant à celle choisie pour sa réponse idéale.

Prédicteur de Smith Le prédicteur de Smith est une forme particulière de compensation adaptée au retard. Son fonctionnement est relativement simple. En utilisant une modélisation précise du système réglé, il

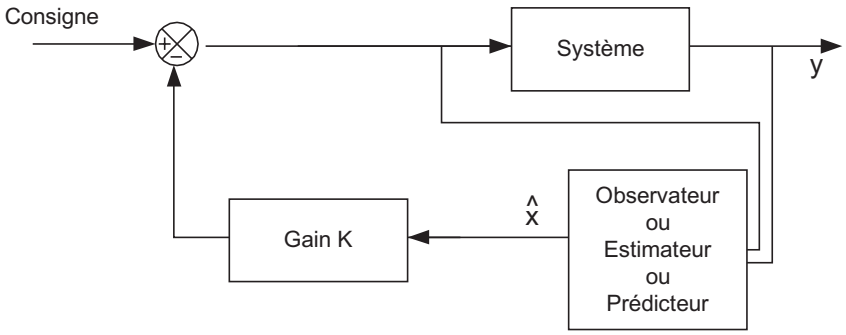


FIG. B.5 – Utilisation du retour d'état quand l'état n'est pas directement accessible

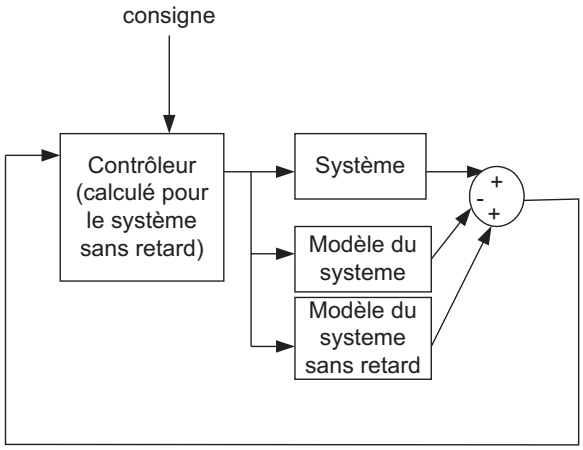


FIG. B.6 – Principe de fonctionnement du prédicteur de Smith

compense la présence du retard sur le suivie de consigne (cf [Smith, 1959; Ogata, 1987] pour plus de détails). Le schéma B.6 présente une application générique du prédicteur de Smith.

Observateur Nous avons présenté les techniques de contrôle par retour d'état. Dans certains cas, ces états ne sont pas atteignables à partir des capteurs disponibles, il est donc nécessaire de chercher à calculer cette état à partir des capteurs mais aussi d'une bonne connaissance du modèle du système. Le rôle de l'observateur est de donner, à partir des informations disponibles, l'état du système.

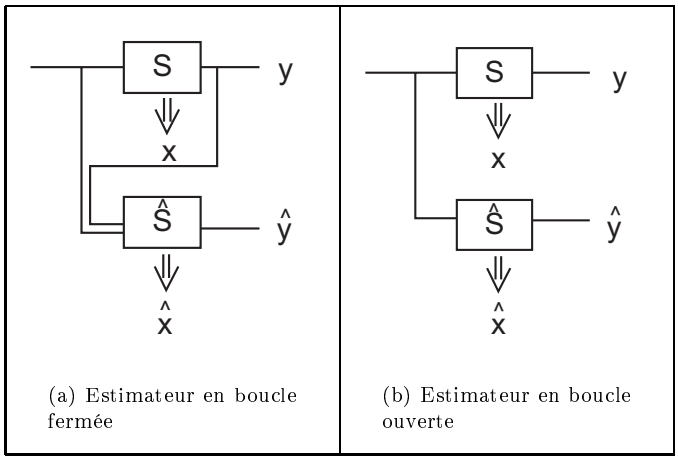
Estimateur d'état Il n'est pas toujours possible de reconstruire l'état du système ; dans ce cas, on peut essayer de l'estimer. L'observateur présenté précédemment utilise alors une boucle de retour pour se corriger. On utilise souvent une boucle de retour rapide (3 à 5 fois plus rapide que le système contrôlé)

B.3.3 Techniques de commande optimale

Optimisation quadratique LQ On aimerait disposer d'un contrôle qui permette de minimiser une fonction de coût de la forme

$$J = k_1 \cdot (tempsdemonte) + k_2(tempsderponse) + k_3(dpassment) + k_4(erreurstatique) + ...$$

mais c'est généralement impossible à analyser. On préfére donc se ramener à une fonction de coût qui permette un calcul rapide, d'où le fait d'utiliser un critère qui se définit de manière quadratique à l'aide de 2 matrices.



(a) Estimateur en boucle fermée

(b) Estimateur en boucle ouverte

FIG. B.7 – Utilisation d'un estimateur en boucle ouverte et en boucle fermée

$$J = \frac{1}{2}\sum (xQx^T + uRu^T)$$

x est le vecteur d'état du système

u est le vecteur de commande appliquée

Q représente la partie du critère liée au système (en particulier à sa dynamique)

R représente la partie énergétique du critère (en son absence, la meilleure commande à appliquer consommerait une énergie infinie)

Le système se modélisant sous forme d'état :

$$x(k+1) = Gx(k) + Hu(k)$$

$u(k)$ est obtenue par retour d'état, soit :

$$u(k) = -Kx(k)$$

Le but de la commande optimale est de calculer la valeur de K qui minimise J.

Commande optimale stochastique LQG Les principes sont les mêmes mais l'on prend en compte des modèles probabilistes dans la résolution du problème. Ils peuvent apparaître aussi bien au niveau du critère que dans la modélisation du système (estimations ou prévisions probabilistes).

Appendix C

Gestion de tâches "anytime" dans le cadre des applications temps réel

Un grand nombre d'applications dans le domaine du diagnostic, de la recherche de trajectoire ou de l'optimisation ([Russell and Zilberstein, 1991; Zilberstein and Russel, 1996]) utilisent de plus en plus des algorithmes "anytime" ou "gloutons" dont la qualité fournie croît avec le temps de calcul alloué. Deux types d'algorithmes "anytime" existent:

- A contrat
- Interruptible

Un algorithme interruptible peut être arrêté à n'importe quel instant et fournir un résultat alors que dans le cas des algorithmes à contrat, ce temps doit être connu avant le début de l'algorithme. L'utilisation de ce type d'algorithme fait apparaître des modes d'expression de contraintes temporelles supplémentaires à celles identifiées pour les tâches classiques (échéances, ...). Nous proposons dans cette étude, la définition d'une nouvelle entité appelée " Ressource Manager ". Son but est de partager le processeur entre les différentes tâches (non-anytime, anytime interruptible et anytime à contrat). De manière à respecter les contraintes temporelles, cet algorithme utilise le test d'acceptation générale de la politique Earliest Deadline First (EDF).

C.1 Tâche temps réel flexible

Les algorithmes "anytime" apportent de la flexibilité dans les systèmes. Ceci est essentiel dans de nombreux domaines (diagnostics, optimisation, contrôle de robot mobile) où la recherche de la solution optimale est impossible en raison du manque de temps. La qualité du résultat d'un algorithme anytime dépend du temps processeur qui lui est alloué. Dans le contexte de la recherche de trajectoire d'un robot mobile par exemple, la qualité du résultat augmente avec le temps de recherche de la solution. Cette fonction possède généralement une asymptote qui correspond à la solution optimale (qui ne peut être obtenue que pour un temps qui tend vers l'infini). La figure C.1 présente un exemple typique d'une telle fonction.

Deux types d'algorithmes anytime peuvent être considérés :

- Les algorithmes interruptibles : ils peuvent être arrêtés à n'importe quel instant et fournir un résultat. Ce type d'algorithme correspond au raffinement progressif de la qualité d'un résultat (par une approche dichotomique, par exemple). L'algorithme peut cependant nécessiter un temps minimum du processeur pour pouvoir délivrer un résultat, on parle alors de partie obligatoire.

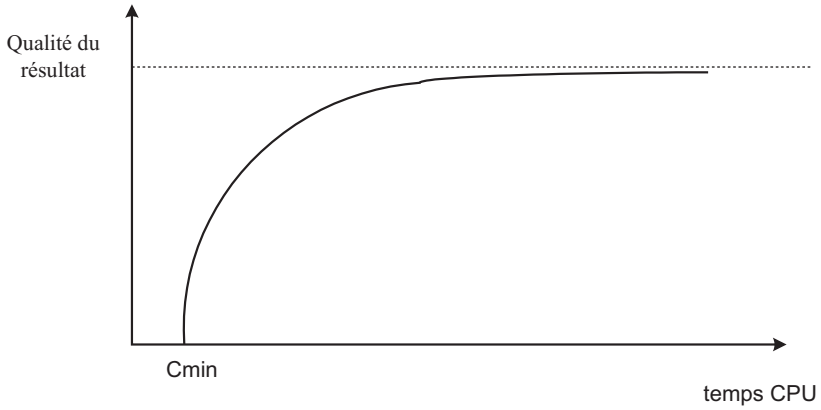


FIG. C.1 – Qualité du résultat en fonction du temps CPU alloué

- Les algorithmes à contrat : ils doivent connaître dès le début de l'exécution, le temps dont ils vont disposer. Ils ne pourront pas délivrer de résultats avant ce temps. Ces algorithmes correspondent typiquement au cas du parcours de graphe, où l'on définit au début de l'algorithme la profondeur des recherches.

Nous nous intéressons ici à l'utilisation des algorithmes "anytime" dans le contexte des applications temps réel, cela signifie que le temps de réponse de ces algorithmes doit respecter des contraintes de temps. Plus précisément, les tâches qui correspondent à l'exécution des algorithmes doivent se terminer avant une date donnée appelée échéance.

Dans le contexte du temps réel, chaque tâche T_i est définie par :

- une date de début A_i,
- une charge (temps CPU demandé) C_i,
- et une échéance D_i.

Un grand nombre de politiques ont été développées de manière à assurer le respect des échéances des différentes tâches. Généralement les résultats sont donnés pour un ensemble de tâches dont les activations sont périodiques.

Parmi les différentes politiques, EDF (Earliest Deadline First) est intéressante parce que son optimalité a été démontrée. Cette politique a une stratégie dynamique qui consiste à donner l'accès au processeur à la tâche dont l'échéance est la plus proche. Cette politique est optimale dans le cas des tâches préemptives (c'est à dire qui peuvent être interrompu en cours d'exécution.), c'est à dire que si la politique EDF ne permet pas de respecter les échéances, aucune autre ne le peut.

Ainsi, un des résultats principaux dans le cas périodique (à échéance sur requêtes) est qu'un ensemble de n tâches périodiques est ordonnançable si le facteur de charge est inférieur à 1 (cf [Stankovic *et al.*, 1998]).

$$\sum_i \frac{C_i}{T_i} \leq 1$$

régulation

Dans cette étude, nous proposons de montrer comment ordonnancer un ensemble de tâches temps réel et anytime.

Une tâche temps réel anytime est définie par :

- Une date de début A_i,
- Un temps minimal de calcul C_{min_i},
- Un temps maximal de calcul C_{max_i},
- Une échéance D_i,
- Sa capacité à être ou non interrompue

C_{min}définit la partie obligatoire de la tâche, c'est à dire qu'aucun résultat ne peut être délivré si le temps processeur consommé par la tâche est inférieur à ce temps.

C_{max} représente le temps processeur maximal utilisable par la tâche (obtention du meilleur résultat possible). Cette borne peut éventuellement être infinie.

Le problème d'ordonnancement des tâches anytime temps réel peut être formulé ainsi ;

Soit S_a un ensemble de tâches anytime et S_c un ensemble de tâches classiques sous contraintes d'échéance :

- Comment garantir le respect des échéances de toutes les tâches ?
- Suivant quel critère attribuer le processeur aux différentes tâches anytime?

C.2 Extension du test de faisabilité "Earliest Deadline First"

Le test de faisabilité présenté dans la section précédente n'est valable que dans le cas des tâches périodiques. Dans le contexte des tâches anytime, comme la recherche de trajectoires ou l'analyse de données complexes, les activités sont généralement apériodiques. La politique EDF est toujours optimale mais le test de faisabilité est plus complexe (cf [Stankovic *et al.*, 1998]). Il a déjà été appliqué dans le cadre de l'ordonnancement sous contrainte d'énergie.

On note $U(t_1, t_2)$ le facteur de charge défini par :

$$u(t_1, t_2) = \frac{S_{1..m}(t_1, t_2, t_2)}{t_2 - t_1}$$

où

$$S_{1..m}(t_1, t_2, d) = \sum_{i=1}^{m} C_i \cdot \infty_{t_1 \leq A_i < t_2} \cdot \infty_{D_i < d}$$

le test étendu de faisabilité de EDF devient :

$$u = max(u(t_1, t_2)) \leq 1$$

Les différents A_i et D_i apparaissent dans la définition des u. On les appelle donc les instants critiques du système en dehors desquels la fonction u est constante. Le test de faisabilité peut être réécrit en considérant uniquement les instants critiques soit :

$$u = max_{t_1, t_2 \in instants\ critiques}(u(t_1, t_2)) \leq 1$$

En considérant uniquement les combinaisons d'instants critiques, pertinents pour l'analyse, on réduit significativement la complexité de la formule. Ainsi, on évalue $u(I_1, I_2)$ uniquement dans les cas suivants :

- lorsque $I_1 < I_2$; dans les autres cas, $u(I_1, I_2) = 0$ et son évaluation est inutile,

- lorsque I_1 est la date d'activation (A_i) d'une tâche T_i et I_2 la date d'échéance (D_j) d'une tâche T_j et que D_i,A_j appartient à $[A_i, D_j]$; dans les autres cas, la charge induite par T_i où T_j n'a pas à être prise en compte car il est, alors, possible de trouver un autre couple (I_1, I_2), plus petit, contenant la même charge,

- lorsque $[I_1, I_2]$ correspond à l'occupation du processeur par au moins une tâche.

La restriction des intervalles critiques est illustrée à la figure C.3 vis à vis de l'exemple présenté à la section C.4.

En appliquant ces règles, le nombre d'intervalles critiques est compris entre 1 et n^2 (où n est le nombre de tâches).[1]

Dans le cas des tâches anytime, ce test peut être très intéressant. Si on considère un ensemble de n tâches conduisant à N intervalles critiques. Le test de faisabilité correspond à un ensemble de contraintes linéaires. On note CI_k un intervalle critique et ST_k l'ensemble des tâches T_i où à la fois A_i et D_i sont inclus dans CI_k. Pour chaque CI_k, la charge apportée par les différentes tâches doit être inférieure à la capacité L_k de l'intervalle. Pour chaque intervalle critique, la condition est donc de la forme :

$$\sum_{i \in ST_k} C_i \leq L_k$$

Pour l'ensemble des intervalles critiques, les contraintes peuvent être représentées sous forme matricielle.

On considère alors :

- Une matrice A où la ligne k représente la présence (1) ou l'absence (0) des différentes tâches T_i dans l'intervalle critique CI_k
- Un vecteur L tel que L_k est la longueur de l'intervalle CI_k
- Le vecteur C tel que C_i est la charge processeur de la tâche T_i

Le test de faisabilité pour déterminer si l'ensemble est ordonnançable sous EDF devient

$$A \cdot C \leq L$$

C.3 Politiques de distribution des tâches anytime reposant sur la politique EDF

Sur la base du test de faisabilité présenté dans la section précédente, nous avons développé différents algorithmes qui permettent de déterminer en ligne la quantité de ressources à allouer aux différentes tâches anytime. Nous considérons à la fois des tâches classiques et anytime. La partie obligatoire d'une tâche anytime est en fait équivalente à l'existence d'une tâche classique. Nous avons donc à partager l'accès au processeur entre des tâches classiques et des tâches anytime sans partie obligatoire. Nous pouvons représenter simplement les différentes contraintes.

Introduisons tout d'abord les notations :

- L : L_k est la longueur de l'intervalle CI_k
- C : C_i est la charge de la tâche T_i (tâche classique) ou la partie obligatoire de la tâche anytime
- CM : CM_i est la charge maximale que peut recevoir la tâche anytime T_i
- A : la ligne k de la matrice A représente la présence (1) ou l'absence (0) des différentes tâches T_i dans l'intervalle critique CI_k.
- X est le vecteur de charge qui est attribué comme partie optionnelle aux différentes tâches anytime.

[1] Dans nos expérimentations, nous n'avons jamais dépassé $\frac{n^2}{2}$.

On a :

$$A \cdot (C + X) \leq L$$

ce qui équivaut à :

$$A \cdot X \leq R$$

avec

$$R = L - A \cdot C$$

Nous proposons trois algorithmes élémentaires pour distribuer les ressources disponibles. La charge disponible apparaît dans le vecteur R. Ces algorithmes sont respectivement appelés distribution à priorité, distribution équitable et distribution à poids. Le but de la distribution à priorité est de donner les ressources aux tâches les plus prioritaires sous respect des contraintes d'échéances pour l'ensemble des tâches classiques et de la partie obligatoires des tâches anytime. La distribution équitable donne équitablement les ressources disponibles sous respect des contraintes d'échéances. La dernière politique est une adaptation de la distribution équitable mais elle permet de fixer des poids différents pour les différentes tâches anytime. Les trois politiques reposent sur le test de faisabilité présenté précédemment.

C.3.1 Politique de distribution à priorité

Cet algorithme comprend 2 étapes :

Premièrement, il faut définir l'ensemble des intervalles critiques et les contraintes associées, on obtient alors le vecteur de contrainte R.

$$R = L - A \cdot C$$

Ensuite, pour chaque partie optionnelle de la tâche T_p, de la plus prioritaire à la moins prioritaire :

a) On évalue les intervalles les plus critiques. Pour cela on parcourt les différents intervalles qui contiennent T_p et l'on cherche la contrainte la plus serrée CI_m.

$$m \text{ est tel que } R_m = min(R_1..R_N)$$

Dans cet intervalle critique CI_m, on détermine X_p qui maximise la charge de la tâche T_p dans CI_m tout en respectant la charge autorisée pour la tâche CM_p .

On a donc :

$$X_p = min(R_m, CM_p)$$

b) Il faut, dans un deuxième temps, prendre en compte la charge optionnelle de T_p. Il suffit pour cela de mettre à jour le vecteur R. On introduit pour cela le vecteur $V = (0..X_P..0)$ (charge attribuée à T_p), on a simplement :

$$R = R - V$$

La complexité de l'algorithme dépend de l'ordre des intervalles critiques et du nombre d'intervalles critiques concernés par chaque tâche. Si le nombre de tâches optionnelles est n_o, alors le pire cas est $o(n_o^2)$.

C.3.2 Distribution équitable

Le but de cette politique est de partager équitablement les ressources entre les différentes tâches anytime. Nous utilisons une technique qui a déjà été appliquée dans le cadre de l'économie d'énergie (cf [Yao *et al.*, 1995]) où des preuves d'optimalité peuvent être trouvées.

1) Comme pour l'algorithme precedent, la première étape consiste à évaluer les intervalles critiques et à calculer le vecteur R.

2) Tant que posssible, pour chaque tâche anytime T_p, et que pour chaque intervalle critique CI_k, on donne la même charge.

Pour cela, on introduit nt_k le nombre de tâches anytime non saturées présentes dans l'intervalle critique CI_k. On peut alors évaluer la charge maximale que l'on peut donner à chaque tâche :

$$d_l = max(\frac{R_k}{nt_k})$$

si $\forall z,\ d_l > CM_z$

Pour chaque tâche concernée,

$X_p = X_p + d_l$

$AT = AT - (T_p)$

Pour chaque intervalle critique CI_k où T_p est présent

$R_k = R_k - d_l$

sinon on détermine z tel que $CM_z = max_{T_p \in AP}(CM_p - X_p)$

Pour chaque tâche concernée,

$X_z = CM_z$

$AT = AT - T_z$

Pour chaque intervalle critique CI_k où T_z est présent

$R_z = R_z - CM_z$

La complexité peut être estimée en $0(nlog(n))$ si l'on représente les objets sous la forme d'un arbre bien balancé.

C.3.3 Distribution à poids

Les principes sont les mêmes que pour la distribution équitable en permettant la prise en compte de poids différents pour les différentes tâches.

C.4 Exemple

De manière à illustrer les trois politiques, nous considérons l'ensemble de tâches suivant :

3 tâches anytime T_1, T_2, T_3

- $A_1 = 2$, $A_2 = 6$, $A_3 = 4$ (elles commencent respectivement à 2, 6 et 4)
- $D_1 = 8$, $D_2 = 9$,$D_3 = 12$ (les échéances sont de 8, 9 and 12)
- $C_1 = 1$, $C_2 = 1$, $C_3 = 1$ (Le temps obligatoire d'exécution est de 1 pour chaque tâche)
- $CM1 = +\infty$, $CM2 = +\infty$, $CM3 = +\infty$ (Le temps d'exécution de chaque tâche n'est pas borné)

-1 tâche classique périodique T à échéance sur requête

- La première instance commence à 1,
- $Period = 6$ (La période est de 6 unités),
- Le temps d'exécution pour chaque instance est de 2 unités.

	tâche	A (date de début)	D (échéance)	C (ressources minimales)	CM (borne sur la partie optionnelle)
Tâches *Anytime*	T_1	2	8	1	$+\infty$
	T_2	6	9	1	$+\infty$
	T_3	4	12	1	$+\infty$
Tâches *Classiques*	T_4	1	7	2	X
	T_5	7	13	2	X

FIG. C.2 – Résumé de la définition des tâches de l'exemple

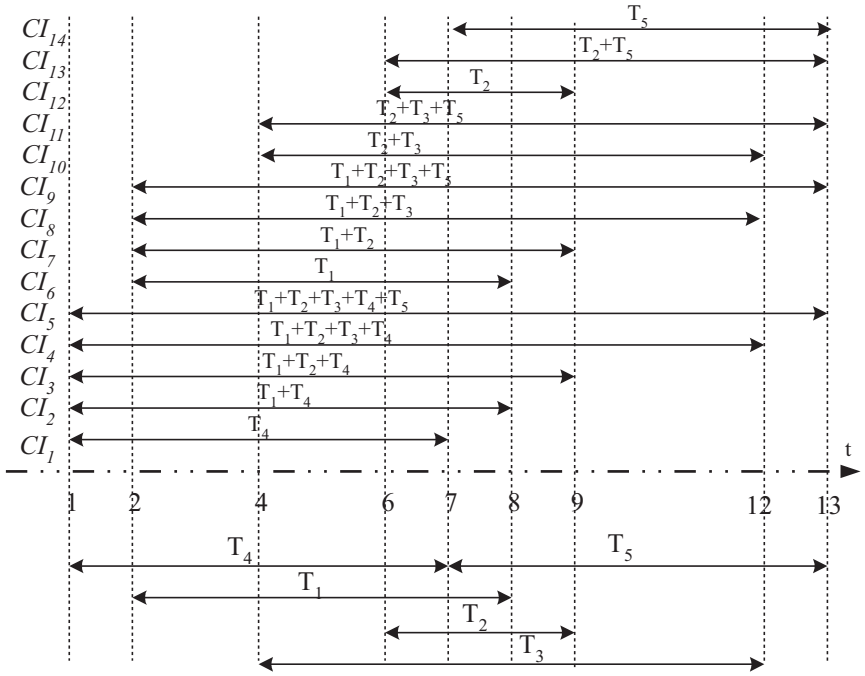


FIG. C.3 – Intervalles critiques

En fait dans cet exemple, nous avons besoin de distinguer plusieurs instances de la tâche périodique T. De manière à simplifier la lecture de l'exemple, nous nous limitons aux deux seules instances qui interviennent. On les notes T_4 et T_5 :

- $A_4 = 1$, $A_5 = 7$ (Les dates de début sont 1 et 7)
- $D_4 = 7$, $D_5 = 13$ (les échéances de la première et de la seconde instance sont respectivement 7 et 13)
- $C_4 = 2$, $C_5 = 2$ (temps d'exécution pour chaque instance)

Cet exemple est résumé dans la tableau C.2.

Au cours de la première étape, nous avons identifié 14 intervalles critiques montrés à la figure C.3

Pour chaque tâche anytime, nous voulons exposer comment définir le temps optionnel à allouer. Pour cela, nous cherchons à trouver X tel que $A \cdot X \leq L - A \cdot C$; vis à vis des définitions de la section précédente, A donne la présence des 5 tâches sur les 14 intervalles critiques, L la longueur de chaque intervalle et C la durée de la partie obligatoire de chaque tâche. A, L et C sont donnés à la Fig C.4.

Nous appliquons les trois politiques d'ordonnancement à cette configuration, nous obtenons pour chaque politique, une distribution des ressources disponibles sur les parties optionnelles des trois tâches anytimes (X_1, X_2, X_3). Les Tables C.5 et C.6 présentent les résultats obtenus. On peut noter que la quantité totale de charge attribuée est la même dans chaque cas (5 unités). Cela signifie que les différentes politiques maximisent l'utilisation des ressources.

Les résultats obtenus avec la politique à priorité pour les différentes configurations de priorités sont donnés

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad L = \begin{bmatrix} 6 \\ 7 \\ 8 \\ 11 \\ 12 \\ 6 \\ 7 \\ 10 \\ 11 \\ 8 \\ 9 \\ 3 \\ 7 \\ 6 \end{bmatrix} \quad C = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \end{bmatrix} \quad L - A.C = \begin{bmatrix} 4 \\ 4 \\ 4 \\ 6 \\ 5 \\ 5 \\ 5 \\ 7 \\ 6 \\ 6 \\ 5 \\ 2 \\ 4 \\ 4 \end{bmatrix}$$

(a) *(b)* **(c)** *(d)*

FIG. C.4 – Définition des matrices

Priorité (↘) **Charge**	T_1 / T_2 / T_3	T_1 / T_3 / T_2	T_2 / T_1 / T_3	T_2 / T_3 / T_1	T_3 / T_1 / T_2	T_3 / T_2 / T_1
X_1	4	4	0	0	2	0
X_2	1	1	5	5	1	3
X_3	0	0	0	0	2	2

FIG. C.5 – Politique de distribution suivant les priorités

à la table C.5.

Lorsque la tâche T_2 a la plus grande priorité, toute les ressources disponibles lui sont attribuées. Dans les autres cas, la ressource n'est pas entièrement attribuée à T_1 (respectivement T_3) car les contraintes d'échéances des différentes tâches ne pourraient plus être respectées. Le cas "priorité(T_3) > priorité(T_1) > priorité(T_2)" est un bon exemple de la distribution des ressources sous contraintes d'échéances, T_3 (de plus grande priorité) ne reçoit que deux unités, T_1 (deuxième priorité) reçoit une unité, et T_2 deux unités. Toutes les ressources sont utilisées suivant l'ordre des priorités.

La table C.6 présente les résultats obtenus avec la politique à distribution équitable et pour quelques configurations de poids différentes. La première configuration correspond au cas équitable (même poids pour toutes les tâches anytime). Dans ce cas toutes les tâches reçoivent la même quantité de ressources. Dans le

	Politique de distribution Equitable	Politique de distribution à poids				
Poids de : T_1		200	100	100	100	100
T_2		100	200	100	200	400
T_3		100	100	200	200	400
Charge						
X_1	1.66	2.5	1.25	1.5	1	0.6
X_2	1.66	1.25	2.5	1.5	2	2.4
X_3	1.66	1.25	1.25	2	2	2

FIG. C.6 – Politique de distribution suivant les priorités

deuxième et le troisième cas, un poids relatif de moitier est donné à une tâche. Elle reçoit alors la moitié de ressources par rapport aux deux autres. Dans le quatrième exemple (poids respectifs 100,100,200), la tâche T_3 a un poids double mais ne reçoit pas le double de ressources. Ce cas est similaire au cas précédent où "priorité(T_3) $>$ priorité(T_1)$>$ priorité(T_2)". En effet, la tâche T_1 ne peut pas recevoir plus de deux unités, l'équité de l'algorithme fait que, comme T_1 et T_2 ont le même poids, elles reçoivent la même quantité de ressources. Le dernier cas présente un exemple similaire, où deux tâches ont le même poids (400) mais ne peuvent pas recevoir la même quantité de ressources.

C.5 Conclusion

En utilisant une extension du test de faisabilité de la politique Earliest Deadline First, nous proposons un moyen simple de gérer un ensemble de tâche anytime dans un cadre temps réel. Cette approche a permis le développement de plusieurs algorithmes élémentaires de partage des ressources disponibles parmi les différentes tâches anytime en fonction de considération de priorité ou de poids.

Dans la première partie , nous avons fait une distinction entre les tâches à contrat ou non. Cette distinction n'a pas été utilisée dans nos exemples. Par contre si l'on veut appliquer la même technique en ligne, des tâches peuvent utiliser moins de ressources que leur WCET, on a donc des ressources supplémentaires à distribuer aux tâches anytime. Dans ce cas, seules les tâches interruptibles peuvent profiter, dés le début de leur exécution, des ressources nouvellement disponibles.

Une autre perspective de ces travaux porte sur l'intégration d'autres politiques que EDF. Nous proposons, en particulier, dans une autre étude [Jumel *et al.*, 2002], différentes politiques permettant d'accroître la flexibilité d'attribution des ressources en utilisant un mécanisme de réservation explicite fournissant de bons résultats en termes de taux d'acceptation.