Université d'Evry Val d'Essonne

Optimal scheduling and control for distributed real-time systems

---

Ordonnancement et contrôle optimal des systèmes temps-réel répartis

THÈSE

Présentée et soutenue publiquement le 20 novembre 2006

pour l'obtention du

Doctorat de l'Université d'Evry Val d'Essonne

(Spécialité Automatique et Génie Informatique)

par

Mohamed El Mongi BEN GAID

Composition du jury :

– Florent CHAVAND	Président
– Karl-Erik ÅRZÉN	Rapporteur
– Daniel SIMON	Rapporteur
– Didier GEORGES	Examinateur
– Arben ÇELA	Co-directeur de thèse
– Yskandar HAMAM	Directeur de thèse

Thèse préparée au sein du laboratoire COSI – Groupe ESIEE Paris

Résumé

Le développement considérable des réseaux et de l'électronique embarquée fait apparaître de nouvelles applications, où la boucle de contrôle est soumise à des contraintes de communication et de calcul. En particulier, dans de nombreuses applications de commande embarquées et distribuées, les ressources de communication et de calcul sont limitées. Cette situation affecte plusieurs domaines, allant de robotique sous-marine au contrôle des clusters de satellites en passant par l'automobile. Le comportement de ces systèmes ne dépend plus seulement des propriétés du couple procédé/contrôleur, mais aussi des caractéristiques des moyens de communication et de calcul. Dans cette thèse, une approche intégrée du problème de la commande et de l'ordonnancement (mise à disposition des ressources de communication ou de calcul) des systèmes dynamiques distribués est étudiée. Les principaux résultats sont les suivants :

- Nous considérons d'abord le problème des limitations des ressources de communication. Nous adoptons un modèle où la commande et l'allocation des ressources de communication sont fortement liées. En interprétant ce modèle comme un modèle hybride, ayant deux types d'entrées : les entrées de commande et les entrées d'ordonnancement, nous formalisons et résolvons le problème de l'optimisation conjointe de la commande et de l'ordonnancement, au sens d'un critère quadratique.

- Nous motivons l'utilisation de la norme $\mathcal{H}_2$ comme un critère de synthèse d'ordonnancements optimaux hors-ligne, ne dépendant que des caractéristiques propres du système. Nous proposons une méthode pour l'optimisation conjointe de la commande et de l'ordonnancement hors-ligne au sens du critère $\mathcal{H}_2$.

- Nous proposons une approche permettant de déterminer en-ligne, en même temps, les valeurs optimales de la commande et l'ordonnancement, au sens d'un critère quadratique. La mise en oeuvre de cette méthode est cependant très coûteuse en ligne. C'est pour cette raison qu'un nouvel algorithme d'ordonnancement, nommé OPP est proposé. OPP permet d'affecter en-ligne les ressources de communication en utilisant l'état dynamique des systèmes commandés, tout en assurant leur stabilité et en garantissant l'amélioration des performances si des hypothèses simples sont vérifiées.

- En s'appuyant sur un modèle affiné des limitations des ressources de communication, où l'échange d'information est modélisé en *bits*, nous proposons une approche permettant l'affectation automatique de la précision de quantification ainsi que du taux de rafraîchissement des commandes, afin d'améliorer les performances de commande.

- Enfin, en utilisant un modèle fin prenant en compte l'exécution des tâches de commande et de l'algorithme d'ordonnancement, nous illustrons la généralisation de ces approches au problème de l'ordonnancement régulé des tâches de commande.

Mots-clés: Commande par réseau, Conception conjointe commande-ordonnancement, optimisation, systèmes hybrides, ordonnancement temps-réel.

Abstract

The considerable development of networks and embedded electronics generates new applications, where the control loop is subjected to communication and computation constraints. In particular, in many embedded and distributed control applications, communication and computation resources are limited. This situation affects several fields, ranging from the underwater robotics to the control of satellite clusters and the automotive industry. The behavior of these systems does not uniquely depend on the properties of the plant/controller couple, but also on the characteristics of the communication and computation means. In this thesis, an integrated approach for the problems of control and scheduling (allocation of communication or computational resources) of distributed control systems is studied. The main results are the following:

- We first consider the problem of communication resources limitations. We adopt a model, where control and communication resource allocation aspects are strongly dependent. By interpreting this model as a hybrid model, having two types of inputs: control inputs and scheduling inputs, we formalize and solve the problem of the joint optimization of control and scheduling, for a quadratic performance criterion.

- We motivate the use of the $\mathcal{H}_2$ norm as a design criterion for obtaining optimal off-line schedules, only depending on the intrinsic characteristics of the system. We propose a method for the joint control and off-line scheduling in the sense of the $\mathcal{H}_2$ criterion.

- We propose an approach that allows determining on-line, at the same time, the optimal values of both control and scheduling, in the sense of a quadratic cost function. The implementation of this method is however expensive on-line (in terms of computations). For that reason, an on-line scheduling algorithm, named OPP is proposed. While being based on a pre-computed optimal offline schedule, OPP makes it possible to allocate on-line the communication resources, based on the state of the controlled systems, ensuring the asymptotic stability of the controlled systems and a control performance improvement if some mild conditions are satisfied.

- Using a refined model of the communication resources limitations, where the exchange of information is modeled in *bits*, we propose an approach for automatically assigning the quantization precision and the update rate of the control signals, in order to improve the control performance.

- Finally, based on fine-grained model, taking into account both the execution of the control tasks and the scheduling algorithm, we generalize the previously described approaches to the problem of the feedback scheduling of control tasks.

Keywords: Control and scheduling co-design, hybrid systems, networked control systems, optimization, real-time scheduling.

Remerciements

Je tiens à exprimer ma reconnaissance et mes vifs remerciements à M. Arben Çela, qui a fait preuve tout au long de cette thèse d'une grande disponibilité et d'un grand enthousiasme pour ce travail. Je le remercie également pour les discussions scientifiques passionnantes que nous avons partagées et pour son soutien continu. Je témoigne également toute ma gratitude et mes vifs remerciements à M. Yskandar Hamam pour avoir accepté de diriger cette thèse. Son soutien continu et ses conseils avisés m'ont été d'une aide précieuse tout au long de ces trois années de travail. Je tiens à remercier chaleureusement tous les membres du jury : MM. Karl-Erik Årzén et Daniel Simon qui m'ont fait l'honneur d'avoir très aimablement accepté la tâche de rapporteur, et MM. Didier Georges et Florent Chavand, qui m'ont fait l'honneur de participer au jury de thèse respectivement en tant qu'examinateur et président. Je remercie M. Rémy Kocik pour ses critiques constructives tout au long de ce travail, avec une vision un peu plus informaticienne, et pour m'avoir donné l'opportunité de contribuer au projet RNTL Eclipse. Je remercie aussi M. Cosmin Ionete pour sa sincère collaboration durant son stage post-doctoral, et pour ses contributions au travail de notre groupe. Merci à tous les membres du laboratoire COSI ainsi qu'aux personnes avec qui j'ai eu l'opportunité de travailler, particulièrement à Mmes Yasmina Abdeddaim et Béatrice Bérard et à MM. Denis Bureau, Alain Carrière, Abdennasser Fakri, Arié Finkelstein, Rédha Hamouche, Eric Ledrap, Abdellatif Réama et Alban Roche pour leur aide que ce soit au niveau de mes activités d'enseignement et pour leurs conseils concernant mon travail. Mes remerciements vont également à Mmes Josiane Crété et Martine Elichabe et à M. Alberto Tiburcio pour leur assistance concernant les tâches administratives, et à Mmes Sylvie Camus et Sylvie Pepin-Lehalleur, pour le soutien qu'elles m'ont apporté au niveau de la documentation.

Contents

List of Figures

List of Tables

1 Introduction

A distributed control system is a control system having one or more control loops that are closed via a communication network. In these systems, the sensors and the actuators are situated in distant locations. This distribution is primarily related to the physical location of the system's components, which may be either sources of information (sensors) or means of action (actuators). In order to control such systems, the information that is provided by the sensors is transmitted to the controllers. Based on this received information, the controllers determine the corrective actions that allow imposing the desired performance. The control commands thus computed are sent to the actuators.

Nowadays, we observe a substantial increase in the use of *networks* in distributed control systems. Several factors explain this choice. The price of the hardware components (network nodes and cables) has continuously decreased over last years to make them more affordable. Consequently, the use of networks has become an economically conceivable solution in many application fields. Furthermore, the use of networks offers many technological advantages, such as the reduction of the obstruction (compared to the point-to-point wires), a better flexibility and modularity, an easier diagnosis and maintenance and finally, an improved reliability. As a consequence, networks are largely used in distributed control systems that belong to many application fields such as automotive industry, aeronautics, robotics or manufacturing. An example of architecture of a distributed control system using a communication network is given in Figure 1.1. In this figure, abbreviations **S**, **C** and **A** respectively represent sensor, controller and actuator nodes

Networks are generally classified into two main categories: *control networks* and *data networks*. The first category was developed in order to be employed in distributed control systems. The networks belonging to this category were designed to support a frequent and regular exchange of small-size data (measurements or control commands for example) that must satisfy strict temporal constraints. The network must guarantee an upper bound on the transmission delay of a given message. Various architectures are supported but the bus topology remains the most used one. CAN, TTP/C, or Profinet-DP are typical examples of control networks. The second category was designed in order to support an infrequent and bursty transfer of a large quantity of data, encapsulated in packets, and without any critical-

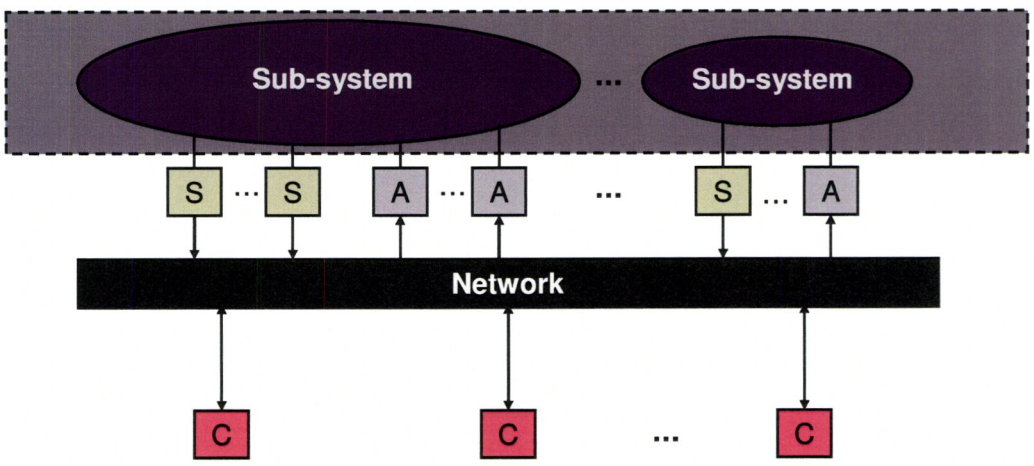


Figure 1.1: Architecture of a distributed control systems

ity. Due to these design constraints, the data-rate that is offered by data networks is much more important than that supported by control networks. In general, data networks (like Ethernet) can cover a wider geographical area than control networks.

The fast development of Internet and wireless networks, which have become inexpensive, ubiquitous and pervasive means of communication, represents an opportunity for the development of new distributed control applications. By allowing the number of theoretically assignable IP addresses to be equal to 3.4×10^{38} addresses (which makes approximately 6.5×10^{23} addresses per square meter of the earth surface, including the oceans), the designers of version 6 of the IP protocol, have probably anticipated this development.

However, the introduction of networks throws down new challenges. In fact, depending on the nature of the employed network, problems like bandwidth limitations, delays or information loss may appear. The controllers may be implemented on limited CPU and memory target platforms, which may be shared between several concurrent tasks. In these situations, the basic assumption, consisting on separating control, communication and computation problems, deserves a careful reexamination. In fact, the presence of the network considerably affects the behavior of the controlled system. The control system performance becomes strongly dependant on the network and computer nodes behaviors. For that reason, the comprehensive study of the distributed control system requires the integrated study of the control, the communication and the computation problems.

As a first example, consider pursuit evasion games [Sinopoli et al., 2003], where a team of pursuing robots must arrest a team of evaded robots, and minimize a given criterion, for example the evasion time. The communication between the different robots is a fundamental aspect of the operation of the system. Any disturbance affecting the communication deteriorates the global behavior of the system and degrades the performances expressed by the criterion. Taking into account the communication constraints becomes crucial for understanding the global operation of the system.

The second example refers to the problem of the distributed control of the automobile suspension [Chalasani, 1986]. The objective is to maximize passengers' comfort as well as road handling. This amounts to minimize the accelerations of the car body, and to maximize the tire-ground contact effort. The correct operation of this system requires the coordinated operation of the four actuators, which are located at the four corners of the vehicle. It is clear that without communication between these distributed components, the achievement of control objectives such as the minimization the roll acceleration is not possible any more. For that reason, the anti-roll bars are maintained when local control strategies are employed.

1.1 Motivations

With the new prospects that are offered by the diffusion of the communication means on the one hand, and the abilities of the control engineering and science to model and handle different categories of systems belonging to a broad range of application fields on the other hand, it is easy to envisions the development of new applications in the future, where the *dynamics* and the *information* will be strongly dependent. For that reason, the study of the integration of control, communication and computation was considered, in a recent report on the future of control [Murray et al., 2003], as a major challenge research direction.

The integrated study of control communication, and computation [Årzén et al., 2000] becomes more important when the communication bandwidth or the processing power is limited. Communication and computing resource limitations are generally presented as being a common characteristic to an important range of *embedded systems*. The term embedded system refers to an electronic system, which is in close relationship to a physical system. Embedded systems are reactive systems: they must correctly respond to the stimuli of their physical environment. They are characterized by a given degree of autonomy, which often appears in the autonomy of the computational resources, and less frequently in the autonomy of the energy supply.

1.1.1 Communication resources limitations

Communication resources limitations may have several reasons. These limitations may be caused by the signal's propagation medium. The underwater medium is an example [Sozer et al., 2000]. In this communication medium, the acoustic communications, which are preferred to the electromagnetic communications [Quazi and Konrad, 1982], offer data-rates varying between a few kbps for long-range communications (over several tens of kilometers), to a few hundred kbps for short-range communications (over several tens of meters) [Stojanovic, 1996]. Some other wireless networks present bandwidth limitations. In the Bluetooth networks for example, only one node can have access to the network every 1.25 ms.

In other situations, network nodes may be autonomous and battery-powered. The transmission of the control information through the network interface requires an important energetic power, particularly in wireless networks. In order to satisfy the lifetime requirements of the batteries, the power consumption that is due to the transmissions, must be limited. Consequently, the data-rate is limited.

The guarantee of a deterministic data transmission introduces technological constraints limiting the maximal possible data-rate. For example, consider the CAN networks [Rachid

and Collet, 2000]. In these networks, the maximal available data-rate depends on the length of the network cable. This data-rate is equal to 1 Mbps for cables whose length is less than 40 m. It goes down to 125 kbps when the cable length becomes equal to 130 m. This limitation is due to the relationship between the bit length on the network and the data-rate on the one hand, and to the used collision resolution mechanisms on the other hand (which require that all the nodes observe the same information during a predefined amount of time). In fact, collision resolution mechanisms require that the length of a bit must be greater than twice the distance between the two most distant network stations. Let R be the data rate, $\mathcal{D}_x$ the distance between the two most distant stations, t_{bit} the duration of a bit on the bus, c_{bit} its propagation velocity and l_{bit} its length. Knowing that $t_{bit} = \frac{1}{R}$, $l_{bit} = c_{bit}t_{bit}$, l_{bit} must verify $l_{bit} > 2\mathcal{D}_x$, which imposes that $R < \frac{c_{bit}}{2\mathcal{D}_x}$. Bandwidth limitations affect many areas such as embedded systems [Årzén and Cervin, 2005], formations control [Belanger et al., 2004], underwater robotics [Speranzon et al., 2004], haptics [Kuschel et al., 2006] or large arrays of micro-electro-mechanical systems (MEMS) [Berlin and Gabriel, 1997].

1.1.2 Computational resources limitations

Embedded systems are generally found in products that are designed for mass production. Their cost may represent a significant part of the cost of these products. Due to market requirements, and to a strong competition between the different manufacturers, their cost is subject to very strict constraints. This may be easily understood since economizing a few centimes in the cost of a hardware component that will be produced in million copies represents a very significant economy. Nowadays, a modern car contains more than 40 embedded microcomputers, called Electronic Control Units (ECUs). Control laws are still being computed using fixed point operations, on limited CPU power and memory microcontrollers. It's just recently that some automotive suppliers have begun to implement the engine controller, which represents the most complex controller of a modern car, on floating point microcontrollers.

1.2 Goals and contributions

When computation or communication resources are limited, they have to be exploited as efficiently as possible. In distributed information processing systems, the *scheduler* is the entity that is responsible for the allocation of communication or computation resources. Consequently, the efficient use of these resources amounts to the design of appropriate scheduling algorithms.

The objective of this thesis is to propose methods allowing a more effective exploitation of communication or computation resources in distributed embedded control systems. The proposed approach relies on two complementary ideas:

- to refine the dynamic model of the plant by taking into account the available communication and computation resources,
- to use performance criteria of controlled system for the joint synthesis of the control and scheduling.

This thesis tackles two problems: communication networks scheduling and single-processor scheduling.

1.2.1 Communication networks scheduling

In the first part, we consider a distributed control system. We focus on the information exchange between its controller and its distant actuators (Figure 1.2).

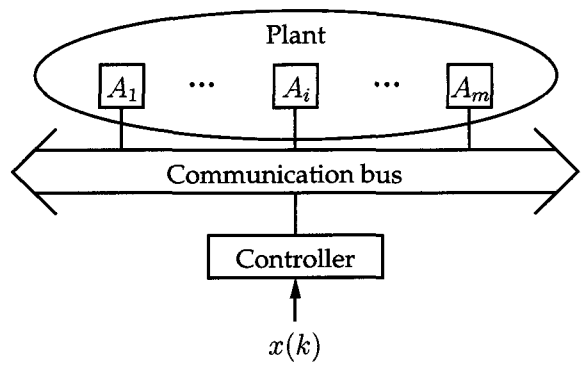


Figure 1.2: A distributed control system with constraints affecting the transmission of the control commands to the m distributed actuators $A_1, \ldots, A_m$

The main contributions are summarized as follows:

Formulation and solving of the optimal integrated control and scheduling problem

We adopt a model from [Hristu, 1999], where control and communication resource allocation aspects are strongly dependent. By interpreting this model as a hybrid system [Bemporad and Morari, 1999], having two types of inputs: *control inputs* and *scheduling inputs*, we formalize and solve the problem of the joint optimization of control and scheduling, using a quadratic cost function as performance criterion. This cost function represents the design criterion of the optimal controller, for a classical model (which does not take into account the communication constraints). The study of the properties of optimal schedule, through some selected numerical examples, shows that it is strongly dependent on the controlled dynamic system state. This dependence offers prospects for the improvement of the performances in terms of quality of control, by the use of on-line scheduling algorithms, which are based on the knowledge of the system state. However, this dependence shows that it is necessary to find other performance metrics for the synthesis of optimal off-line schedules.

Optimal integrated control and off-line scheduling in the sense of the $\mathcal{H}_2$ norm

We motivate the use of the $\mathcal{H}_2$ norm as a design criterion for obtaining optimal off-line schedules, only depending on the intrinsic characteristics of the system. We propose a method for the joint control and off-line scheduling in the sense of the $\mathcal{H}_2$ criterion. We show that this problem may be decomposed into two sub-problems, which may be solved separately. The first sub-problem aims at determining the optimal off-line scheduling in the sense of the $\mathcal{H}_2$ criterion and may be solved using the branch and bound method. The second sub-problem aims at determining the optimal control gains and may be solved using the periodic optimal control theory.

The use of the model predictive control as a means for the joint optimal control and on-line scheduling

We propose an approach that allows determining on-line, at the same time, the optimal values of both control and scheduling, in the sense of a quadratic cost function. This approach relies on the use of the model predictive control (MPC) technique, which was applied in the past for the control of hybrid systems. We illustrate the performance improvements, in terms of quality of control, which are brought by this approach, compared to static approaches, where the used scheduling is pre-computed off-line. We also state the stability conditions of the predictive controller.

Reduction of the computational complexity of the controller/scheduler

The major disadvantage of the model predictive control technique is that it requires the solving of an optimization problem, which is rather expensive on-line (in terms of computations). For that reason, an on-line scheduling algorithm, called OPP for *optimal pointer placement* is proposed. While being based on an off-line pre-computed optimal schedule, OPP makes it possible to allocate on-line the communication resources, based on the state of the controlled dynamic systems. It is shown that under mild conditions, OPP ensures the asymptotic stability of the controlled systems and enables in all the situations the improvement of the control performance compared to the basic static scheduling. OPP is applied in a typical example of a distributed control system: the car active suspension controller.

Trading quantization precision for sampling rates

We extend the model that was first considered in order to take into account quantization related aspects. Consequently, the communication constrains are modeled at the bit level, in bits per second. In general, increasing the sampling frequency improves the disturbance rejection abilities whereas increasing the quantization precision improves the steady state precision. However, when the bandwidth is limited, increasing the sampling frequency necessitates the reduction of the quantization precision. In the opposite, augmenting the quantization precision requires the lowering of the sampling frequency. Based on these observations, we propose an approach allowing the dynamical on-line assignment of sampling frequencies and control inputs quantization. This approach, which is based on the model predictive control technique, enables to choose the sampling frequency and the quantization levels of control signals from a predefined set, in order to optimize the control performance.

1.2.2 Control tasks scheduling

In the second part, we tackle the problem of the single-processor scheduling of control tasks. Two sub-problems are dealt with:

The joint off-line optimization of control and scheduling

We illustrate the required steps allowing to generalize the previous results on the optimal integrated control and off-line scheduling in the sense of the $\mathcal{H}_2$ performance index to control tasks having different execution times and being executed "in parallel" with other non-control tasks.

On-line scheduling of control tasks

The previously considered model is refined in order to deal with the problem of the plant state-feedback scheduling of control tasks. The execution cost of the scheduling algorithm is explicitly taken into account. We propose a method allowing the reduction of the feedback scheduler execution cost. Based on a fine grained and comprehension implementation model, which was simulated using the tool TRUETIME [Andersson et al., 2005], we evaluate the suggested on-line scheduling approach.

1.3 Outline and publications

The following of this document is organized as follows.

In Chapter 2, the basic concepts of the real-time scheduling theory are first reviewed. Then, an overview of the state of the art of the integrated approaches for control and resource allocation in distributed embedded control systems is given.

Chapter 3 describes the abstract model of a distributed embedded control system, with communication resources constraints that was adopted in this thesis. The main theoretical results, which were established previously in the literature, are reviewed.

In Chapter 4, the problem of the optimal integrated control and scheduled is formalized and solved. The analytic solution of the simpler problems of the optimal control for a fixed scheduling over finite and infinite horizons is stated.

In Chapter 5, we motivate the use of the $\mathcal{H}_2$ performance index in order to derive optimal off-line schedules. The problem of the joint optimization of control and off-line scheduling in the sense of the $\mathcal{H}_2$ performance index is then formalized and solved.

Chapter 6 focuses on the problem of the optimal integrated control and on-line scheduling of networked control systems. A model predictive control-based approach is first proposed. Then, the OPP scheduling algorithm is introduced and its main properties are stated. Finally, the OPP algorithm is applied to a distributed car suspension control system.

In Chapter 7, the considered model of distributed embedded control systems is extended, to take into account quantization aspects. An approach for the dynamical on-line assignment of sampling frequencies and control inputs quantization is proposed.

In Chapter 8, a model describing the single-processor execution of control tasks is proposed. The previously established results are refined in order to be applicable to the problems of the real-time scheduling of control tasks.

Chapter 9 constitutes the conclusion of this document.

Publications

The different chapters of this document are based on the following publications.

- M-M. Ben Gaid, A. Cela and Y. Hamam. Optimal Real-Time Scheduling of Control Tasks with State Feedback Resource Allocation. In revision, *IEEE Transactions on Control Systems Technology*, 2006. **(Chapter 8).**
- M-M. Ben Gaid, A. Cela and Y. Hamam. Optimal Integrated Control and Scheduling of Networked Control Systems with Communication Constraints: Application to a Car Suspension System. *IEEE Transactions on Control Systems Technology*, 14 (4), pp. 776-787, 2006. **(Chapters 3, 4 and 6).**

- M-M. Ben Gaid and A. Cela. Trading Quantization Precision for Sampling Rates in Networked Systems with Limited Communication. In *Proceedings of the 45th IEEE Conference on Decision and Control*. San Diego, California, USA, December 2006. **(Chapter 7).**

- M-M. Ben Gaid, A. Cela, Y. Hamam and C. Ionete. Optimal Scheduling of Control Tasks with State Feedback Resource Allocation. In *Proceedings of the 2006 American Control Conference*. Minneapolis, Minnesota, USA, June 2006. **(Chapters 5 and 8).**

- M-M. Ben Gaid, A. Cela and Y. Hamam. Optimal Integrated Control and Scheduling of Systems with Communication Constraints. In *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*. Seville, Spain, December 2005. **(Chapters 4 and 6).**

- M-M. Ben Gaid and A. Cela. Model Predictive Control of Systems with Communication Constraints. In *Proceedings of the 16th IFAC World Congress on Automatic Control*. Prague, Czech Republic, July 2005. **(Chapter 6).**

2

Resource allocation in distributed embedded control systems

2.1 Introduction

Traditionally, control design problems are decoupled from software design and implementation considerations. This separation of concerns allowed the control and computer science communities to focus on specific problems, and led to the development that we are familiar with nowadays. However, this separation of concerns relies on the fact that these two fields use very simplified models of their interface with each other [Årzén et al., 2000]. In fact, control designers disregard the characteristics of the implementation and the available computational and communication resources. Real-time designers on the other hand see the control loop as a periodic task with a hard deadline, which have sometimes to fulfill data dependency constraints (especially when the sensing and actuation are distant). Recently, researchers from the two communities have shown that if more elaborate models are used, significant improvements in terms of implementation efficiency and quality of control may be achieved.

This chapter is organized into two parts. The first part presents the state of the art of the real-time scheduling theory, focusing on the most used results in distributed embedded control applications. Real-time single-processor scheduling problems are first presented. Then, the problem of ensuring real-time networked communications is dealt with. We put the emphasis on the different methods for managing the access concurrency, which have a determinant impact on the guarantee of deterministic real-time communications. Finally, an overview of the problem of guaranteeing end-to-end real-time constraints in distributed systems is given. In the second part, we present a state of the art of the new approaches, that are based on more elaborate models, and that take into account both the dynamic nature of the controlled systems and also some characteristics of their implementation. Various problems and models were tackled in the literature. We propose a classification of these different approaches and illustrate the different problems and models that were addressed.

2.2 Real-time scheduling theory

2.2.1 Real-time single-processor scheduling

In real-time processing systems, the processor is a resource that is shared between various concurrent *tasks*. A task is a sequence of instructions that are intended to be executed by the processor. The service that is delivered by a task may be performed several times during the lifetime of the application. That's why a task may be "instantiated" several times in the form of *jobs* or *task instances*. Jobs or task instances represent the execution flow that corresponds to the effective execution of the task code.

Events characterizing the lifetime of a job

A job or task instance is characterized by the following temporal parameters:

- its *release time*: the time instant at which the scheduler is requested to execute the job, which have just become ready to run,
- its *start time*: the time instant at which the job starts its execution,
- its *preemption times*: time instants when the scheduler suspends the execution of the job on behalf of other jobs having a more important priority,
- *resumption times*: time instants at which the execution of the job is resumed after a preceding preemption,
- its *completion time*: time instant at which the job finishes its execution,
- its *absolute deadline*: time instant before which the job should have terminated.

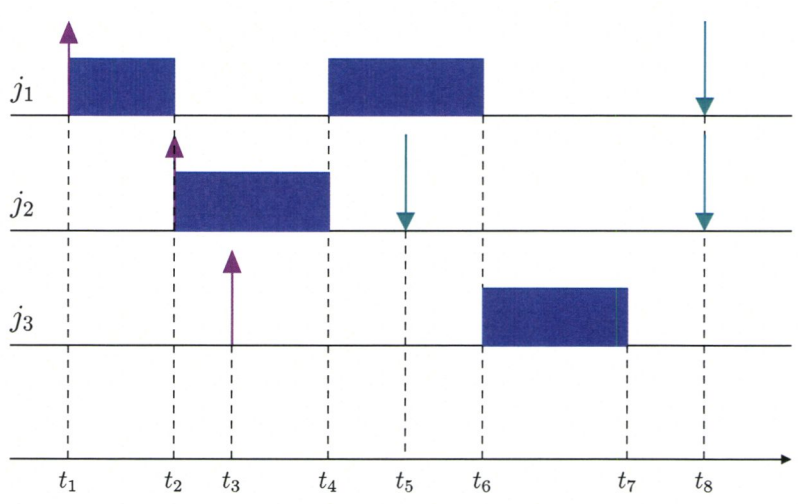


Figure 2.1: Single-processor scheduling of three tasks

Figure 2.1 illustrates the single processor scheduling of three jobs j_1, j_2 and j_3. In this figure:

- instants t_1, t_2 and t_3 represent the respective release time instants (depicted by up arrows) of jobs j_1, j_2 and j_3,

- instants t_1, t_2 and t_6 represent the respective start times of jobs j_1, j_2 and j_3,
- instants t_2 and t_4 represent respectively preemption and resumption times of job j_1,
- instants t_6, t_4 and t_7 represent the respective completion time instants of jobs j_1, j_2 and j_3,
- instants t_5 and t_8 represent the respective absolute deadlines (depicted by down arrows) of jobs j_1 and of j_2 and j_3.

Task model

A real-time task $\tau^{(i)}$ is characterized by:

- its worst case execution time (WCET) $c^{(i)}$,
- the activation law of its jobs: the jobs of a given task may be activated periodically with a period $T^{(i)}$, sporadically with a minimum inter-arrival time or aperiodically if no temporal constraints are imposed on the activation of its jobs,
- its relative deadline $D^{(i)}$: the time interval between the release time of the job and its absolute deadline.

A real-time task is called periodic, sporadic or aperiodic according to the activation law of its jobs.

Scheduling algorithms classification

This paragraph describes the commonly used terminology for classifying the scheduling algorithms.

- **Preemptive/Non-preemptive**: A scheduler is called *preemptive* if it is able de suspend a running task on behalf of other tasks that have more important priorities. It is called *non-preemptive* in the opposite case. Preemption is supported by the majority of real-time operating systems. In the opposite, the scheduling of packets in networks is always non-preemptive.
- **Off-line/On-line**: In *off-line* scheduling algorithms, the sequencing of the tasks to be executed is described at design time, as a *schedule* or *execution plan*. Consequently, the scheduler is simply a sequencer, which executes the different tasks according to the schedule that was pre-computed off-line. The execution order of the different tasks is then identical to that specified in the execution plan. In practice, the schedule is executed in a repetitive way. The period of repetition is called *major cycle* or *hyperperiod*. In general, the schedule describes the start time instants of the different tasks instances, and possibly, their preemption and resumption time instants, which are expressed as a function of an elementary time unit, called *minor cycle* of the schedule. In the opposite, in *on-line* scheduling algorithms, the choice of what task to execute is determined at runtime by the scheduler. When activated, the scheduler performs a given processing in order to determine the next tasks to execute. In most cases, this processing amounts to the comparison of the priorities of the ready tasks. These priorities may be fixed in the case of *fixed-priority* scheduling algorithms or dynamic (i.e. adjustable at runtime) in the case of *dynamic scheduling algorithms*.

Schedulability analysis

Real-time scheduling theory aims at providing the sufficient conditions (and preferably the necessary and sufficient conditions) guaranteeing that a task set (which is defined by a given model) will respect its real-time constraints (which are defined by the assigned deadlines). An important theoretical tool that it provides is the *schedulability analysis*. Schedulability analysis is performed off-line, in order to ensure that the scheduling of a task set (which satisfies a given model), using a given scheduling algorithm, ensures the respect of the tasks deadlines. In the following, we present the fundamental results that are related to the preemptive real-time scheduling of periodic tasks whose relative deadlines are equal to their periods, by fixed-priority and dynamic-priority scheduling algorithms.

Consider a task set containing $\mathcal{N}$ independent tasks $\tau = \left(\tau^{(1)}, \ldots, \tau^{(\mathcal{N})}\right)$. Each task $\tau^{(i)}$ is characterized by its WCET $c^{(i)}$, its period $T^{(i)}$ and its relative deadline $D^{(i)} = T^{(i)}$. The task set τ is said to be *schedulable* by a given scheduling algorithm if all the jobs of all the tasks forming τ finish their execution before their absolute deadlines.

- **Fixed-priority scheduling:** In this scheduling policy, a fixed priority $p^{(i)}$ is assigned to each task $\tau^{(i)}$. At each instant, the scheduler executes the ready task whose priority is the most important. Since preemption is authorized, if during the execution of a given task $\tau^{(i)}$, another task $\tau^{(j)}$ that has a more important priority becomes ready, then task $\tau^{(i)}$ is preempted and the processor is allocated to task $\tau^{(j)}$. In 1973, Liu and Layland [Liu and Layland, 1973] have shown that the optimal priority assignment is given by the *rate monotonic* (RM) rule (ie. the smaller the period is, the higher is the assigned priority). Thus, the rate monotonic scheduling algorithm is a fixed-priority scheduling algorithm where priorities are assigned according to the rate monotonic rule. The optimality of a real-time scheduling algorithm was defined by Liu and Layland by its ability to schedule a given task set, which verifies a given task model, such that the predefined real-time constraints are met. A fixed-priority (resp. dynamic-priority) scheduling algorithm is optimal (for a given task model) in the sense that *any task set (satisfying the task model) that is not schedulable under this algorithm will not be schedulable under any other fixed-priority (resp. dynamic-priority) scheduling algorithm*. A sufficient schedulability condition using RM is

$$U = \sum_{i=1}^{\mathcal{N}} \frac{c^{(i)}}{T^{(i)}} \leq \mathcal{N}(2^{1/\mathcal{N}} - 1).$$

The necessary and sufficient schedulability condition by RM requires the analysis of the maximum response time $R^{(i)}$ of each task (i.e. the maximum among the response times of its jobs) [Joseph and Pandya, 1986]. The response time of a job is defined by the duration between its release time and its completion time. $R^{(i)}$ is given by

$$R^{(i)} = c^{(i)} + \sum_{j \in hp(i)} \left\lceil \frac{R^{(i)}}{T^{(j)}} \right\rceil c^{(j)}.$$

where $hp(i)$ is the set of tasks which have priority over $\tau^{(i)}$. The task set τ is schedulable by RM if and only if $R^{(i)} \leq D^{(i)}$, for all $i \in \{1, \ldots, \mathcal{N}\}$.

- **Dynamic-priority scheduling:** In this scheduling policy, the priority $p^{(i)}$ that is assigned to task $\tau^{(i)}$ may vary over time. Liu and Layland proved that the optimal dynamic priority assignment policy consists on assigning the most important priority to the task that is the closest to its deadline. This priority assignment rule is called Earliest Deadline First (EDF). The necessary and sufficient schedulability condition under EDF is simpler than that established for RM and is given by

$$\mathcal{U} = \sum_{i=1}^{\mathcal{N}} \frac{c^{(i)}}{T^{(i)}} \leq 1.$$

Real-time scheduling theory progressed substantially since the fundamental article of Liu and Layland, to take into account the problems involving the scheduling of sporadic and aperiodic tasks, the scheduling of tasks whose deadlines are lower or higher than their periods, the protected access to shared resources, the precedence constraints and the non-preemptive scheduling. A detailed description of the fundamental results concerning these problems may be found in [Krishna and Shin, 1997, Buttazzo, 1997, Liu, 2000, Burns and Wellings, 2001, Decotigny, 2003].

2.2.2 Real-time medium access control in communication networks

Ensuring real-time communications is the responsibility of the entire communication stack. Nevertheless, the crucial role falls on the MAC (medium access control) sub-layer of the layer 2 of the OSI model [Zimmermann, 1980]. The MAC sub-layer has the responsibility of managing the access to the communication medium, which may be shared between several nodes of the network. In real-time networks, there are several access protocols. The most deployed ones are:

TDMA

The TDMA (time division multiple accesses) protocol makes it possible to statically divide, the available bandwidth, in the temporal domain, between several competing nodes. In this protocol, each node knows exactly the moments when it is allowed to transmit over the network. In consequence, each node must transmit during the *time slot* which is allocated to it, called *TDMA slot*. In this way, collisions are avoided. The time slots are predetermined off-line. Their sequencing has a periodic structure. The minimal sequence of time slots allowing to describe the sequencing of the time slots of the various nodes is called *TDMA round* (Figure 2.2).

The TDMA protocol may be implemented in a centralized or a distributed way. In centralized implementations, a master node has the responsibility of triggering the communications of the other slave nodes by the transmitting a synchronization signal. The major disadvantage of this approach is that a breakdown of the master node leads to a total breakdown of the network. The distributed implementations require the establishment of a sufficiently precise global time in all the nodes of the network, which requires the use of clock synchronization algorithms. The TDMA protocol is the cornerstone the mobile communications GSM protocol. The TTP/C communication protocol [TTTech, 2003] manages the concurrent access to the communication medium using a distributed implementation of

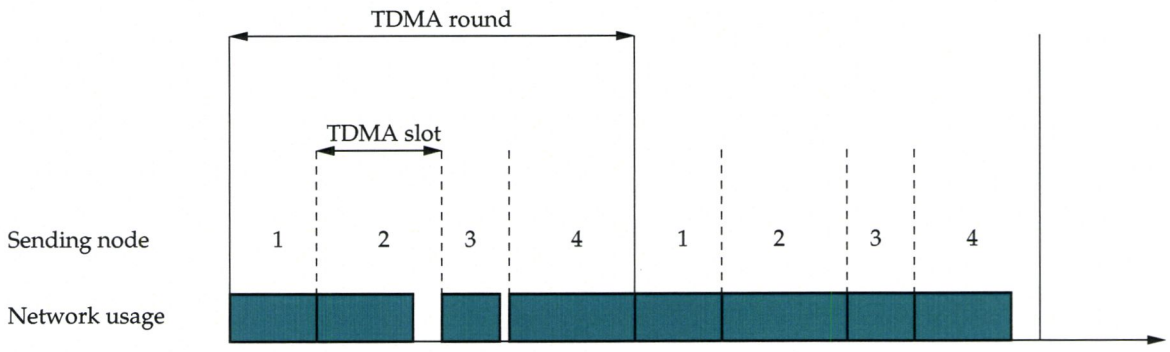


Figure 2.2: Bandwidth sharing using the TDMA protocol and used terminology

the TDMA access protocol. In order to ensure a global clock, the FTA (Fault-Tolerant Average) algorithm [Kopetz and Ochsenreiter, 1987] is used to ensure the clock synchronization of the different network nodes.

Token-Bus

The Token-Bus access protocol was specified by the IEEE (IEEE standard 802.4) and by the ISO (ISO standard 8802.4). It represents the cornerstone of many communications protocols that are employed in industrial fieldbusses [Thomesse, 1998], like Profibus [Bajic and Bouard, 2002], ControlNet [ControlNet International, 1999], MAP [MAP/TOP Users Group, 1988] and ProfiNet [Bouard, 2005].

In this protocol, the network nodes are logically organized in a ring topology: each node knows its logical predecessor and its logical successor. The access arbitration is performed by the circulation of the *token* between the nodes. At any given moment, only one node has the token. The possession of the token gives to the possessing node the right to transmit over the network. The node that takes possession of the token can start transmitting over the network. It must pass the token to its successor if the time it has held the token reaches a limit or if it finishes transmitting before the expiry of this duration. If a node that does not have any information to transmit, receives the token, then it transmits it directly to its successor node. Concerning the real-time properties of this protocol, the analysis of the worst-case response times of Profibus and ControlNet messages are respectively given in [Tovar and Vasques, 1999] and [Lian et al., 2001]. The worst-case response time of a message is defined as the duration between the moment the transmission is requested and the moment when the message is received by the destination process.

CSMA/CA

The CSMA/CA (carrier sense multiple access with collision avoidance) access protocol is used in many networks such as CAN, DeviceNet or IEEE 802.11 wireless networks. In this access model, each message is characterized by a unique priority. Since the shared communication medium can transmit only one message at a time, each node that wishes to transmit a message must initially check whether the network is free (by sensing the network to find whether or not a carrier signal is being transmitted). If the network is free, then the node can start transmitting. However, it is possible that other nodes start transmitting at the same

time, because they have detected at the same time that the communication medium has become free. In this situation, the transmission of the highest priority message is continued; the other messages with lowest priorities are discarded. By this way, collisions are avoided.

CAN networks [ISO, 1993, Rachid and Collet, 2000] use the CSMA/CA protocol in order to manage the concurrent access to the shared bus. Their implementation of CSMA/CA relies on a bit synchronization mechanism at the bus level. In CAN networks, each message is characterized by a unique identifier. Furthermore, no node is particularly addressed: all the sent messages are broadcasted to all the other network nodes. When several nodes are emitting and if at least one node sends one a '0' (called dominant level in CAN terminology), then all listening network nodes will detect a '0' at the same time, even if there are other nodes which have transmitted a '1'. Reciprocally, when all the transmitting nodes send a '1' (called recessive level), all the listening nodes will detect a '1'. The CAN bus behaves like a logical AND gate. Since the identifier field is located at the beginning of the frame (the most significant bit is first coded, the highest priority is 0), and the binary synchronization is implemented on the bus, when a collision occurs, the different nodes directly compare the identifiers of their messages to the resulting logical level at the bus. A node that detects that the resulting level is dominant ('0') whereas it has sent a '1', knows that it has tried to send a message whose priority is lower than another message and must consequently stop transmitting. Figure 2.3 describes a collision between two messages, which were sent at the same moment by node 1 and node 2. Node 2 stops transmitting when it detects that it has a lower priority than node 1 (because it has sent a '1' and the resulting level on the bus was a '0').

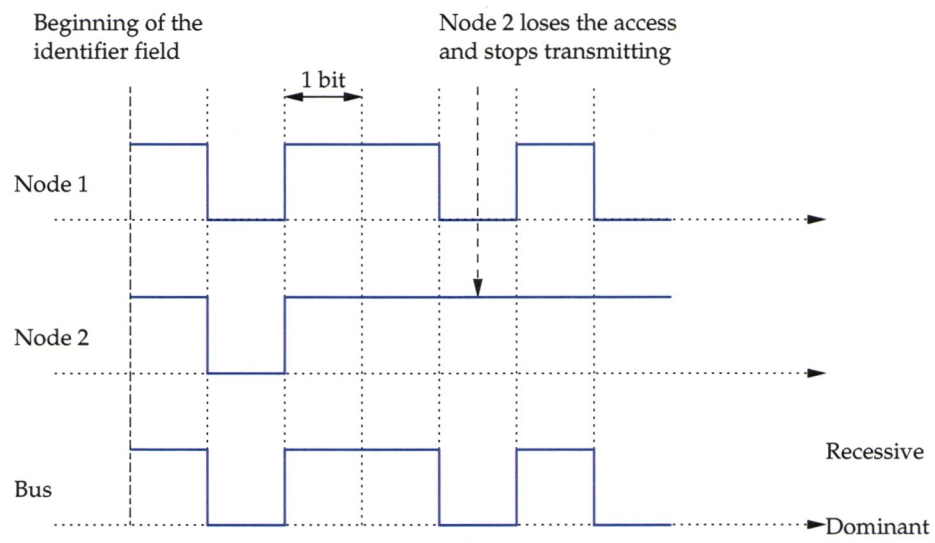


Figure 2.3: Illustration of the bus arbitration mechanism in CAN networks

CAN protocol is a deterministic protocol. Consequently, it is possible to compute an upper bound over messages response times. A CAN bus may be seen as a non-preemptive scheduler. The computation of the worst-case response time of fixed-priority messages have been tackled in [Tindell and Burns, 1994, Tindell et al., 1995]. The evaluation of a RM priority assignment policy was also addressed in these references. The use of EDF scheduling in CAN networks was studied in [Di Natale, 2000].

The binary synchronization over the bus, which is necessary to collisions arbitration,

introduces a relationship between the maximum data-rate and the length of the network cable. In fact, to use a data-rate of 1 Mbps, the maximum length of the cable must be less than 40 m. If it is necessary to use a cable whose length is greater than 620 m, then the maximum data-rate falls down to 100 kbps.

Note that in data networks, like Ethernet or the Internet, the CSMA/CD (carrier sense multiple access with collision detection) protocol is the most used medium access protocol. The fundamental difference between CSMA/CD and CSMA/CA resides in the collision arbitration mechanism. In fact, in the CSMA/CD protocol, the nodes that generate a collision are able to detect this collision and to stop their transmission during a random duration. For that, reason, it is impossible to bound messages response times in networks employing the CSMA/CD access protocol. In wireless networks, such as IEEE 802.11 networks, the CSMA/CA protocol is employed because it is not possible to detect the collisions (and thus to deploy the CSMA/CD protocol).

2.2.3 Real-time scheduling of distributed systems

Real-time multiprocessor scheduling problems are still not as well understood as real-time single-processor scheduling problems; the most obscure points are related to the schedulability analysis [Sha et al., 2004]. In this field, the impact of the Dhall and Liu's paper [Dhall, 1978] on real-time multiprocessor scheduling theory was equivalent to the impact of the Liu and Layland's paper on real-time single-processor scheduling theory [Liu and Layland, 1973].

Multiprocessor scheduling algorithms may be classified into two categories:

- *partitioned scheduling*, where each task is assigned to only one processor,
- *global scheduling*, where all the tasks compete for the use of all the processors.

The problem of the optimal partitioning of tasks among processors is NP-complete [Garey and Johnson, 1979]. For that reason, simulated annealing or branch and bound-based heuristics were proposed to tackle this problem. The use of these heuristics relies on the modeling the scheduling problem as an optimization problem. A detailed presentation of these approaches is given in [Attiya, 2004]. An outline of the most important results concerning multiprocessor schedulability analysis may be found in [Sha et al., 2004].

The tasks that may be located on the different processors may have data-dependencies and thus exchange messages via a communication medium. The communication medium may be seen as "a processor" that only supports the non-preemptive scheduling. Among the tools for off-line partitioned scheduling generation for tasks with precedence, on distributed architectures, one may cite the tool Syndex [Sorel, 2004, Kocik and Sorel, 1998], which is based on the *Adéquation Algorithme Architecture* (for efficient matching of the algorithm on the architecture) approach [Grandpierre et al., 1999]. In Syndex, the architecture and the algorithm are described by two graphs. The algorithm graph is a direct acyclic graph, where the vertices represent the operations to be performed and where the edges represent the data-dependencies between the operations. The architecture graph describes the available parallelism as well as the communication possibilities between the various processors. Based on these two graphs, and possibly on placement constraints which may be specified by the user, Syndex uses the greedy list scheduling algorithm [Yang and Gerasoulis, 1993, Kwok and Ahmad, 1999] to synthesize the scheduling of the operations on the different architecture elements.

2.3 Integrated approaches for control and resource allocation

In the previous paragraph, we have described some important results of the real-time scheduling theory. Disregarding the nature of the considered applications, these results mainly addressed the problem of communication or computation resource allocation, in order to respect strict temporal constraints. In this paragraph, we outline other approaches, which jointly consider the problems of control and communication or computation resources allocation. First, we review some problems and methods for the adaptive sampling. Second, we give an outline of the methods allowing to jointly considering the problems of control and communication resource allocation. Finally, we review the state of the art of the methods for the joint control and computational resource allocation in embedded systems.

2.3.1 Adaptive sampling of control systems

The analysis of asynchronously sampled systems was undertaken at the end of the fifties. The research works, which were performed at the sixties and at the beginning of the seventies focused on SISO systems. In our best knowledge, the first adaptive sampling method was proposed by Dorf *et al.* [Dorf et al., 1962]. The proposed adaptive sampler changes the sampling frequency according to the absolute value of the first derivative of the error signal. Using analog simulations, the authors have shown that this method reduces between 25% and 50% of the number of required samples, with respect to the periodic sampling, given the same response characteristics. Other approaches were proposed thereafter, in particular those of [Gupta, 1963a, Gupta, 1963b, Tomovic and Bekey, 1966a, Tomovic and Bekey, 1966b, Mitchell and McDaniel Jr., 1969]. The evaluation of these approaches and their comparison to the periodic sampling was performed in [Smith, 1971]. Simulations have shown that these adaptive sampling methods are not always better than periodic sampling, especially where the input is subject to unknown disturbances. Remarking that the methods of [Dorf et al., 1962] and [Mitchell and McDaniel Jr., 1969] are closely related [Hsia, 1972], Hsia proposed a generic approach allowing to derive adaptive sampling laws [Hsia, 1974].

2.3.2 Allocation of communication resources: the "per symbol" paradigm

In this paradigm, the information exchange is modeled at the symbol level. The quantization of measurements and control commands is thus implicitly taken into account. The general model is given in Figure 2.4.

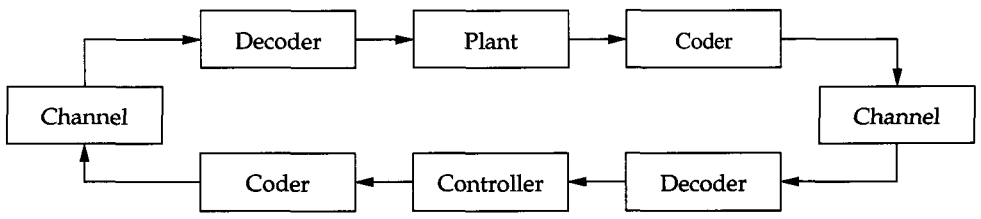


Figure 2.4: General model of the information flow in a control system whose control loop is closed through finite bandwidth communication channels

In this model, the communication channel can transmit at most R bits per time unit. Because of these resource limitations, measurements and control commands must be encoded

(as a flow of symbols) before their transmission and decoded at their reception. Various coding techniques may be employed. A fundamental question is to determine the necessary and/or sufficient data-rate allowing the existence of a coder, a decoder and a controller that achieve the stabilization of the system. Many contributions have tried to bring more insight into this fundamental question, by treating various models of resource limitations.

In [Delchamps, 1990], Delchamps has shown that it is impossible to asymptotically stabilize a discrete-time unstable LTI system, whose output passes through a quantizer having a finite number of quantization levels. In this setting, it is necessary to introduce and use other stability concepts, like practical stability.

The problem of state estimation, in presence of state and measurement noise, was studied in [Wong and Brockett, 1997]. In the considered model, the state observer is situated at the same location as the plant. However, the controller is located at a distant place. Consequently, the observations must be sent to the controller through a finite bandwidth communication channel. This problem was shown to be different from the classic estimation and vector quantization problems. The concept of *finitely recursive coder-estimator sequence* was then introduced. Necessary conditions as well as sufficient conditions, which are related to the stability and the convergence of various coding-estimation algorithms, were stated. These conditions connect the network data-rate to the dynamical characteristics of the plant.

In [Wong and Brockett, 1999], Wong and Brockett introduced the concept of *containability*, as a weaker stability condition, to tackle the problem of the stabilization of networked systems through limited capacity communication networks, where the values of the measurements (which are received by the controller) and the controls (which are sent to the plant) belong to a finite set of values $\mathcal{J}$ (because of the quantization which is induced by the limited data-rate communication channel). Considering continuous-time LTI systems, which are impulsively controlled, they proved that a necessary condition to ensure the containability is $e^{\frac{2}{R}tr(A_c)} \leq |\mathcal{J}|$ where $|\mathcal{J}|$ is the size of the alphabet, $\frac{1}{R}$ is the transmission duration of one bit and A_c is the state matrix. They also proved that if the initial condition of the system lies in a bounded set, then a memoryless coding and control is sufficient to ensure the containability, if some conditions affecting the data-rate are met.

In [Nair and Evans, 2000], Nair and Evens considered a class of discrete-time, linear, time-varying and infinite-dimensional plants. No process or measurement noise affects the considered plants. The initial state is the realization of a random variable. Communications constraints only affect the sensors-to-controller link. The controller is directly connected to the actuators. The considered problem is the synthesis of a coder (on the sensors-to-controller link) and of a controller that minimize a cost function of the state over finite and infinite horizons. The cost function over the finite horizon is the m^{th} output moment. A coder/controller scheme was proposed. Under some technical assumptions, which are related to the probability density function of the initial state, to some conditions depending on the size of the alphabet, to the data-rate and to the plant dynamics, a necessary and sufficient optimality condition of the proposed coder/controller was established. A necessary and sufficient condition for the existence of a coder/controller that asymptotically stabilizes the system (in the sense that the m^{th} output moment converges to zero over an infinite horizon) was stated. In the special case where the plant is invariant, unstable and finite-dimensional, this last condition simplifies to $R > log_2|\lambda|$ where λ is the unstable open-loop pole with the largest magnitude.

In [Brockett and Liberzon, 2000], Brockett and Liberzon proposed the idea of "zooming"

as a means for ensuring the asymptotic stability of continuous-time and discrete time system whose control loops are closed through a finite bandwidth communication network. The zooming technique consists on changing the sensitivity of the quantizer over the time, based on the available quantized measurements. The relationship between performance and complexity of the quantized stabilization using the zooming technique were studied in [Fagnani and Zampieri, 2004].

In [Elia and Mitter, 2001], Elia and Mitter studied the problem of the stabilization of single-input linear systems whose measurements and control commands are quantized. By first considering quantizers with a countable number of levels, and supposing the exact knowledge of the state, they proved that the coarsest quantizer that allows the quadratic stabilization of a discrete-time single-input LTI system, is logarithmic, and may be computed by solving a special LQR problem. The state-feedback control problem as well as the state observation problem were solved within this theoretical framework. These results were thereafter extended to the continuous-time single-input and periodically sampled linear systems. The expression of the optimal sampling period (for the suggested quantizers) was established. It only depends on the sum of the unstable eigenvalues of the continuous system. This approach was finally extended to address quantizers with a finite number of levels.

In [Tatikonda and Mitter, 2004], Tatikonda and Mitter considered discrete-time LTI systems. The control loop is closed through a limited capacity communication channel. Consequently, before their transmission, the measurements are quantized and encoded in symbols by a coder. At their reception, a decoder reconstructs a state estimate that will be used by the controller, which is directly connected to the plant. Two types of coders were studied:

- class 1 coders, which know past measurements, past controls and past transmitted symbols that were sent over the channel,
- class 2 coders, which only know past measurements.

Stabilization and asymptotic observability properties were studied. It was shown that a necessary conditions for the existence of coders and decoders making it possible to guarantee these two properties is given by $R > \sum_{\lambda(A)} \max\{0, log\,|\lambda(A)|\}$, the sum is over the eigenvalues of the state matrix A. This necessary condition is independent from coder classes and becomes sufficient if the whole state is measured and if class 1 coders are used.

2.3.3 Allocation of communication resources: the "per message" paradigm

The different approaches that may be related to the "per message" paradigm are motivated by the fact that in all communication networks, protocol frames contain fields with fixed and incompressible length. These fields include, for example, the identifier field, the CRC (Cyclic Redundancy Check) field, which is used by error detection algorithms. For example, in CAN networks, the minimal length of the fixed protocol fields is 47 bits (the length can be more important because of the bit-stuffing mechanism [Rachid and Collet, 2000]). A measure that is encoded in 12 bits and sent in a CAN message only represents 20% of the size of the message. In Bluetooth networks, the minimal size of the data field is 368 bits. If an information whose size is less than 368 bits is to be transmitted, then padding by '0' bits must be carried out.

The various approaches, which are related to the "per message" paradigm, may be classified into two categories:

- the decentralized minimization of the network bandwidth usage. The basic idea is that each node tries to locally minimize its bandwidth consumption,
- the scheduling of the concurrent access to the network. In these approaches, a more global view of the application, of its distribution and of the network access mechanism is taken into account. The information transmission over the network is managed by taking into account static characteristics (the model) or instantaneous information (the state) of the controlled dynamic system.

Minimization of the network bandwidth usage

A research direction that may be linked to the "per message" paradigm is the *model based control*, which was studied in [Yook et al., 2002, Montestruque and Antsaklis, 2003, Montestruque and Antsaklis, 2004, Hespanha and Xu, 2004a, Hespanha and Xu, 2004b]. The basic idea of this approach is to use local open-loop observers in order to reduce the required communications between the sensors and the controller.

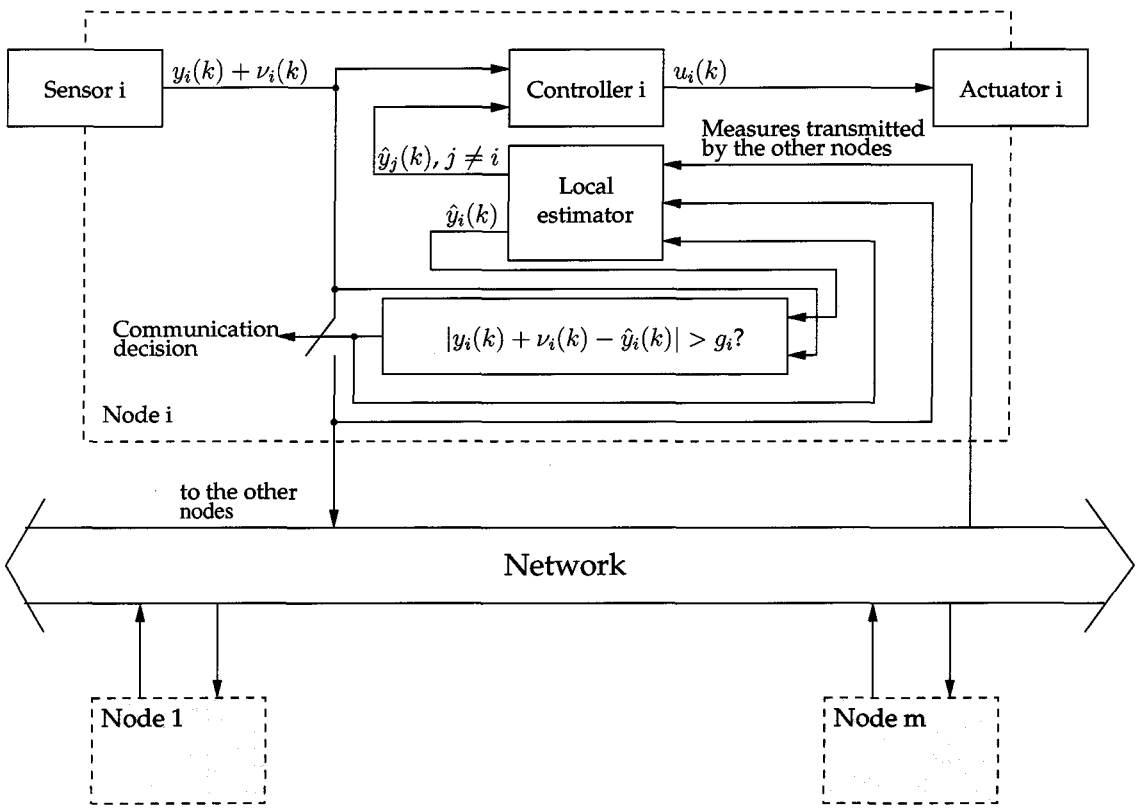


Figure 2.5: Global architecture and model of the i^{th} node according to the approach of [Yook et al., 2002]

In [Yook et al., 2002], a method allowing the minimization of the required communications in some particular distributed control applications was proposed. This method addresses multivariable discrete-time LTI systems that are implemented according to a specific distributed architecture and controlled using output-feedback. The global distributed system has n states, m inputs and m outputs, such that the i^{th} input $u_i(k)$ and the i^{th} output

$y_i(k)$ are co-located at the same network node, which is also equipped with a computer (Figure 2.5). The measurements, which are provided by the m sensors, are corrupted by a measurement noise $\nu(k)$. The m nodes are connected through a perfect network (without delays, nor losses of information). Each node contains an estimator that estimates at the same time its own output as well as the outputs of the other nodes. The estimated outputs $\hat{y}_j(k)$, $j \neq i$ are then used by the i^{th} node, for the computation of the control $u_i(k)$, instead of the measured outputs $y_j(k) + \nu_j(k)$, $j \neq i$, whose use would require a transmission of their values over the network. By this way, an important communication saving is achieved, at the price of a more important computational load at the computer nodes. This approach relies on the fact that all the estimators compute identical values of the estimated outputs $\hat{y}_j(k)$. If the estimator of the i^{th} node realizes that $|y_i(k) + \nu_i(k) - \hat{y}_i(k)| \geq g_i$, where g_i is a fixed threshold, then it broadcasts to the other nodes the value of its measurement $y_i(k) + \nu_i(k)$, enabling them to update the state of their estimators. By this way, $|y(k) + \nu(k) - \hat{y}(k)|$ is limited by $g = [g_1, \ldots, g_m]^T$. Finally, a result allowing the choice of H to guarantee a maximum degradation of $\varepsilon\%$ compared to the ideal system (without networked communication) was stated. The experimental validation of the method was carried out on a two axis contouring system. The experimental results have shown that only 12% of the communication is necessary in order to guarantee a maximum degradation of 1%, assuming a model uncertainty of 20%.

A similar approach was studied in [Montestruque and Antsaklis, 2003]. In the considered architecture, the controller is directly connected to the plant. A perfect network connects the sensors to the controller. The sensors periodically transmit (each T_s time instants) the measurements to the controller, which is provided with an open-loop state-observer. The estimated state is then used for the computation of the control commands. At the reception of a message from the sensors, the state of the open-loop observer is updated. The necessary and sufficient stability conditions of this particular model of networked control systems were stated, and generalized to take into account output feedback. Sufficient stability conditions when T_s is time-varying but bounded were presented in [Montestruque and Antsaklis, 2004].

In [Hespanha and Xu, 2004a], a similar architecture was studied. In the considered model, the plant is disturbed by a zero-mean Gaussian white noise. Instead of sending the measurements (of the full state) when the prediction error exceeds a threshold [Yook et al., 2002] or periodically [Montestruque and Antsaklis, 2003], the transmission of measurements is performed using predefined communication logics. The study and the evaluation of different communications logics (stochastic and deterministic) was performed in [Hespanha and Xu, 2004a]. By modeling the problem as jump-diffusion process, sufficient conditions for the boundedness of the finite moments of the estimation error were established for the considered logics. Considering a long term average cost, penalizing at the same time the estimation error and the transmission rate, the expression of the optimal communication logics was given in [Hespanha and Xu, 2004b].

Medium access scheduling

The experimental study of communication networks characteristics was performed in [Nilsson, 1998] and [Lian et al., 2001]. Studying the main characteristic of ControlNet, DeviceNet and EtherNet networks, Lian *et al.* [Lian et al., 2001] have shown that the transmission time of a message (i.e. the time the message spends on the physical link from the source to the destination) in the most used networks may be neglected. The delays occurring in networked control loops are mainly due to the *contention* between the different messages which are sent

by the nodes of the network. The most efficient way of reduction of these delays is the design and use of appropriate message scheduling strategies.

These results show the practical importance of the study of the medium access control as well as scheduling algorithms. The problems of the concurrent access to shared communication resources were studied these last years within various theoretical frameworks and with various modeling assumptions. We present thereafter a brief summary of the approaches taking into account explicitly the concurrent access to the communication network. These contributions were classified into three categories, according to the class of the used scheduling algorithms: off-line scheduling, on-line scheduling of the sensors-to-controller link and the on-line scheduling of the controller-to-actuators link.

- **Off-line scheduling:** The problem of optimal control and off-line scheduling of the controller-to-actuators link was studied in [Rehbinder and Sanfridson, 2004]. In the proposed model, the control commands are sent to the actuators through a shared TDMA bus. At each slot, only one control command can be sent, the remaining commands for the other actuators are held constant. The choice of which actuator to update at each slot was handled using the notion of communication sequence [Brockett, 1995]. Only periodic communication sequences were considered. A quadratic cost function is associated to each communication sequence, corresponding to the worst-case initial condition and worst-case sequence permutation. Control commands and periodic communication sequences are obtained through the solving of a complex combinatorial optimization problem. The problem of the optimal control and scheduling in the sense of LQG was introduced and developed in [Lincoln and Bernhardsson, 2002]. The relaxed dynamic programming method was applied for its resolution leading to a more efficient search heuristics. A heuristic approach for the problem of the optimal control and off-line scheduling of the sensors-to-controller link in the sense of the $\mathcal{H}_\infty$ performance index was proposed in [Lu et al., 2003].

- **On-line scheduling of the sensors-to-controller link:** The scheduling of sensor measures was studied in [Walsh and Ye, 2001]. The addressed configuration consists of a continuous-time plant where the controller is directly connected to the actuators. The network only connects the sensors to the controller. The notion of MATI (maximum allowable transfer interval) was introduced, and represents the upper bound on the time between two consecutive sensor messages transmissions that guaranties the stability of the plant. The MATI is defined for a given scheduling algorithm. A new on-line scheduling algorithm, called MEF-TOD (maximum error first - try once discard), was introduced. In this dynamic priority on-line scheduling algorithm, the priority of a sensor message depends on the error of the measure that it carries; smaller the error is, lower is the assigned priority. The error is defined as the weighted absolute value of the difference between the value of the current measure and the value of the last transmitted measure. If a node fails to send a message, then this message is discarded (dropped from the queue). The authors stated sufficient stability conditions, involving the value of the MATI, which ensure the stability of the system, when the MEF-TOD algorithm and a round robin like static scheduling algorithm are used. These results are based on the perturbation theory and are very conservative. This approach was generalized to non-linear systems in [Walsh et al., 2001]. The practical implementation of the MEF-TOD algorithm was considered in [Walsh et al., 2002]. This implementation,

which was performed on CAN networks, is mainly based on the nondestructive bitwise arbitration of CAN technology to dynamically to encode the dynamic priorities. The effects of the quantization of priorities were experimentally studied. Sufficient input/output $\mathcal{L}_p$-stability results for a class of network scheduling protocols, including MEF-TOD and static scheduling algorithms were stated and illustrated in [Nešic and Teel, 2004]. These results considerably reduces the conservativeness of the results that were initially stated in [Walsh and Ye, 2001]. The application of the Rate Monotonic scheduling algorithm to the networked control systems was investigated in [Branicky et al., 2002].

- **On-line scheduling of the controller-to-actuators link:** On-line scheduling of control commands to the actuators was studied in [Palopoli et al., 2002a]. In the proposed model, it is assumed that every slot, only one command vector can be sent to an actuator group, the other control vectors are set to zero. The stabilization is achieved using a model predictive controller, which calculates on-line the appropriate control law and the allocation of the shared bus. The cost function used by the MPC calculates a weighted sum of the infinity norms of the states and the control commands over a specified horizon. The optimization problem solved at each step by the MPC algorithm was proven to be equivalent to the generalized linear complementarity problem (GLCP) [Ye, 1993]. The same architecture is considered in [Goodwin et al., 2004]. The considered model assumes that it is possible to send only one message during one sampling period. The actuators, which do not receive their control inputs, maintain constant the last received ones. The control commands are quantized with a fixed precision. The expression of the optimal model predictive controller, in the sense of a quadratic cost function, was established.

2.3.4 Allocation of computational resources

Optimal control and monoprocessor scheduling

The problem of the optimal selection of control tasks periods subject to schedulability constraints was addressed in [Seto et al., 1996]. Assuming that the discrete-time control laws are designed in the continuous-time domain and then discretized, the notion of performance index was introduced. The performance index quantifies the performance of the digitalized control law at a given sampling frequency. In most control applications, the performance index is minimal when the continuous-time control law is used and increases (i.e. degrades) as the sampling frequency is decreased (note that for some control systems this relationship is more complicated, as illustrated in [Eker, 1999]). Considering this important class of control applications, the problem of the optimal sampling frequency assignment for a set of control tasks consists on minimizing a weighted sum of the performance indices of the considered control tasks subject to schedulability constraints. In [Rehbinder and Sanfridson, 2000], the optimal off-line scheduling of control tasks in the sense of LQG was considered, assuming that all the control tasks have the same constant execution time. The resolution of this problem was performed using the exhaustive search method, which limits the application of this approach to applications with a limited number of tasks. Similarly, the problem of the optimal monoprocessor scheduling of control tasks in order to optimize a robustness metric (i.e. the stability radius) was treated in [Palopoli et al., 2002b, Palopoli et al., 2005].

Scheduling of control tasks in environments with variable computing workload

It is well-known that worst-case analysis techniques [Sha et al., 2004] may be used in order to guarantee the deadlines of tasks with variable but bounded execution times. However, when the average execution time is smaller than the worst-case execution time (WCET), these techniques lead to an oversized design. Recently, new approaches were proposed in order to handle variations in tasks execution times and system overload more efficiently than worst-case analysis techniques, among them the feedback scheduling [Lu et al., 2002, Cervin et al., 2002, Robert et al., 2005, Xia and Sun, 2006] and the elastic task model [Buttazzo et al., 2002].

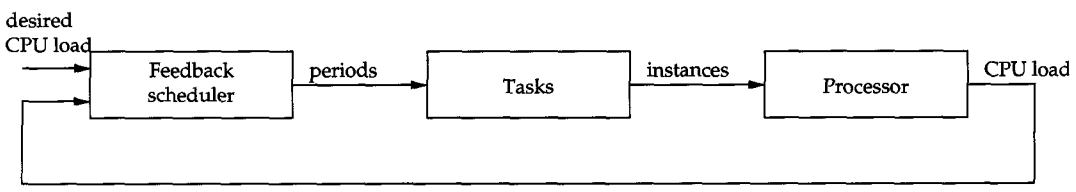


Figure 2.6: General model of a feedback scheduler

Feedback scheduling is a control theoretical approach to real-time scheduling of systems with variable workload. The feedback scheduler may be seen as a "scheduling controller" that receives filtered measures of tasks execution times and acts on tasks periods in order to minimize deadline misses. The application of feedback scheduling to robot control was experimentally evaluated in [Simon et al., 2005].

In the elastic task model [Buttazzo et al., 2002], a periodic task set containing $\mathcal{N}$ tasks may be seen as a sequence of $\mathcal{N}$ linear springs. In this model, the utilization factor of a task is analogous to the spring's length. Tasks may change their utilization rate in order to handle overload conditions, which may occur, for example, if a new task is admitted to the computing system. In order to ensure the schedulability of the task set, tasks are compressed or decompressed. In [Liu et al., 2000], the elastic task model was applied to the scheduling of control tasks with variable execution times. The use of this method permits the application of the approach of [Seto et al., 1996] in order to find the optimal tasks periods based on tasks average execution times (instead of their worst-case execution times), leading to an improvement of the control performance. Buttazzo *et al.* [Buttazzo et al., 2004] generalized this approach to take into account the degradations that may occur to the control system if its control task that was designed to work at a given rate runs at another rate. The analytical expressions of the performance degradations were given. A compensation method was proposed. This method allows to trade-off the performance degradations and the required memory space (which is needed to store the parameters of the pre-computed control laws that will be used by the compensation algorithm).

However, all these described approaches are mainly based on the assumption that control performance is a convex function of the sampling period. In reality, as illustrated in [Martí et al., 2002] (and further in Chapter 4), the quality of control is also dependent on the dynamical state of the controlled system (in equilibrium, in transient state).

2.4 Conclusion

This chapter introduced the basic concepts, the terminology and the state of the art of communication and computation resource allocation approaches in distributed embedded control systems. To this end, the basic concepts of the real-time scheduling theory were overviewed, focusing primarily on the hard real-time scheduling of tasks on processors and messages on deterministic networks. An outline of the state of the art of the approaches for the integrated control and communication/computation resource allocation was given, providing an overview of the tackled problems and the provided solutions.

3

Resource-constrained systems

3.1 Introduction

In this chapter, we present our abstract view of a distributed embedded control system operating under communication constraints. This abstract view is described by the class of *computer-controlled systems*, which was introduced by Hristu in [Hristu, 1999]. This class allows to model, in a finely-grained and abstract way, the impact of the resource limitations on the behavior of the controlled system. In this thesis, we will rather use the term of *resource-constrained systems* to refer to this class of systems. After the introduction of the used notation, we present the framework of the *mixed logical dynamical* (MLD) systems, which represents a modeling framework for hybrid systems, and which was introduced by Bemporad and Morari in [Bemporad and Morari, 1999]. We show that resource-constrained systems may be modeled in the MLD framework. We propose a systematic approach allowing establishing the MLD model of a resource-constrained system. Finally, we review the main theoretical results that are related to resource-constrained systems, and which were already established in the literature. These results include the problems of stabilization, tracking, reachability and observability.

3.2 Definitions and notations

Let X be a vector of $\mathbb{R}^n$. The 2-norm or Euclidian norm of X, denoted $\|X\|_2$, is defined by

$$\|X\|_2 = \sqrt{\sum_{i=1}^{n} X_i^2}.$$

The ∞-norm of X, denoted by $\|X\|_\infty$, is given by

$$\|X\|_\infty = \max_{i \in \{1,\ldots,n\}} |X_i| .$$

Let $[A]_{1\leq i\leq n,1\leq j\leq m}$ be a $n \times m$ matrix. $[A]_{ij}$ denotes the element of A that is situated at the i^{th} row and j^{th} column. Let $X_1, \ldots, X_n$ be a collection of matrices. $\mathrm{Diag}(X_1, \ldots, X_n)$ denotes the block-diagonal matrix defined by

$$\mathrm{Diag}(X_1, \ldots, X_n) = \begin{bmatrix} X_1 & & \\ & \ddots & \\ & & X_n \end{bmatrix}.$$

Let I be an interval of $\mathbb{R}$. We denote by $\mathcal{L}_2^n(I)$ the vector space of functions $f : I \longrightarrow \mathbb{R}^n$ that are square integrable on I, i.e.

$$\left(\int_I \|f(t)\|_2^2\, dt\right)^{1/2} < \infty.$$

For a continuous signal f in $\mathcal{L}_2^n(I)$, the $\mathcal{L}_2$ norm of f, denoted by $\|f\|_{\mathcal{L}_2}$, is defined by

$$\|f\|_{\mathcal{L}_2} = \left(\int_I \|f(t)\|_2^2\, dt\right)^{1/2}.$$

In a similar way, in discrete-time, let I be a discrete interval of $\mathbb{Z}$, we denote by $l_2^n(I)$ the vector space of $\mathbb{R}^n$-valued sequences $(f_k)_{k\in I}$ verifying

$$\left(\sum_{k\in I} \|f_k\|_2^2\, dt\right)^{1/2} < \infty.$$

For a discrete-time signal f in $l_2^n(I)$, the l_2 norm of f, denoted by $\|f\|_{l_2}$, is defined by

$$\|f\|_{l_2} = \left(\sum_{k\in I} \|f_k\|_2^2\right)^{1/2}.$$

Finally, we note by $0_{n,m}$ (resp. $1_{n,m}$) the matrix having n rows and m columns whose elements are equal to zero (resp. equal to 1).

3.3 Mixed-logical dynamical systems

Mixed logical dynamical (MLD) systems, are a class of hybrid systems, which was introduced by Bemporad and Morari in [Bemporad and Morari, 1999]. This framework was proposed in order to allow the modeling and the control of a class of systems where dynamics interact in tightly coupled way with logic and heuristic rules. The MLD framework provides a general model, which generalizes various existing models such as linear hybrid systems, linear constrained systems, finite-state machines...

The logical aspects may be formulated in the propositional logic. The fundamental idea of the MLD framework is to represent these propositional formulas by equivalent linear constraints, involving continuous and/or Boolean variables. In this way, it is possible to represent many hybrid systems by linear dynamic equations whose variables, which may be at continuous and/or Boolean, are subjected to linear inequality constraints. Note that the

idea of representing propositional formulas by equivalent linear inequalities was developed in the past in the field of artificial intelligence, and more specifically for inference engines. Using this correspondence, the proof of a theorem is reduced to the search of the existence of a feasible solution of an optimization problem. In fact, by associating a Boolean variable to a propositional formula, such that the truth-value of the formula is equal to the value of the variable, the propositional calculus operations like conjunction, disjunction, negation, implication or equivalence may be translated into equivalent linear programs.

Let X_1 and X_2 be two propositional formulas and α_1, α_2 two associated Boolean variables verifying

$$X_1 \text{ (resp. } X_2) \text{ is True if and only if } \alpha_1 = 1 \text{ (resp. } \alpha_2 = 1),$$

and equivalently

$$X_1 \text{ (resp. } X_2) \text{ is False if and only if } \alpha_1 = 0 \text{ (resp. } \alpha_2 = 0).$$

It is easy to verify the correspondence between the propositional formulas and the conjunction of linear inequalities that are described below.

$$\begin{array}{rll} X_1 \vee X_2 & \text{is equivalent to} & \alpha_1 + \alpha_2 \geq 1, \\ X_1 \wedge X_2 & \text{is equivalent to} & \alpha_1 = 1, \alpha_2 = 1, \\ \neg X_1 & \text{is equivalent to} & \alpha_1 = 0, \\ X_1 \Longrightarrow X_2 & \text{is equivalent to} & \alpha_1 - \alpha_2 \leq 0, \\ X_1 \Longleftrightarrow X_2 & \text{is equivalent to} & \alpha_1 - \alpha_2 = 0. \end{array}$$

Following the same reasoning, it is also possible to translate terms involving at the same time continuous and Boolean variables into linear inequalities. Let $f : \mathbb{R}^m \longrightarrow \mathbb{R}$ be a linear function. Let $\mathcal{X}$ be a bounded set. Assume that

$$U = \max_{x \in \mathcal{X}} f(x),$$

and

$$L = \min_{x \in \mathcal{X}} f(x).$$

Morari and Bemporad have shown that the product $\alpha f(x)$, for $x \in \mathcal{X}$, where α is a Boolean, may be replaced by a conjunction of linear constraints, if the auxiliary variable $y = \alpha f(x)$ is introduced. It is easy to verify that the equality $y = \alpha f(x)$ is equivalent to

$$\begin{array}{rcl} y & \leq & U\alpha \\ y & \geq & L\alpha \\ y & \leq & f(x) - L(1-\alpha) \\ y & \geq & f(x) - U(1-\alpha). \end{array}$$

This translation will be of great importance when dealing with the problem of the joint optimization of control and scheduling.

In [Bemporad and Morari, 1999], many other useful translations are reviewed. Using these transformations, many hybrid systems may be handled in the MLD framework. The

general state-model of a MLD system is given by the following equations

$$\begin{aligned} x(k+1) &= A_k x(k) + B_{1_k} u(k) + B_{2_k} \alpha(k) + B_{3_k} \xi(k) && \text{(3.1a)} \\ y(k) &= C_k x(k) + D_{1_k} u(k) + D_{2_k} \alpha(k) + B_{3_k} \xi(k) && \text{(3.1b)} \\ E_{2_k} \alpha(k) + E_{3_k} \xi(k) &\leq E_{1_k} u(k) + E_{4_k} x(k) + E_{5_k}, && \text{(3.1c)} \end{aligned}$$

where

$$x = \begin{bmatrix} x_c \\ x_l \end{bmatrix}, x_c \in \mathbb{R}^{n_c}, x_l \in \{0,1\}^{n_l}, n = n_c + n_l$$

is the state, which contains n_c real-valued components and n_l Boolean-valued components,

$$y = \begin{bmatrix} y_c \\ y_l \end{bmatrix}, y_c \in \mathbb{R}^{p_c}, y_l \in \{0,1\}^{p_l}, p = p_c + p_l,$$

and

$$u = \begin{bmatrix} u_c \\ u_l \end{bmatrix}, u_c \in \mathbb{R}^{m_c}, u_l \in \{0,1\}^{m_l}, m = m_c + m_l$$

are respectively the output and the command input of the system, also containing both continuous and Boolean components. $\alpha \in \mathbb{R}^{q_l}$ and $\xi \in \mathbb{R}^{q_c}$ are respectively continuous and Boolean auxiliary variables.

It may be remarked that the inequality (3.1c) might be satisfied for many values of $\alpha(k)$ and $\xi(k)$. For that reason, the determination of $x(k+1)$ and $y(k)$ may be non-unique. A condition that is generally imposed on MLD systems is to be *well posed*. This condition guaranties that once $x(k)$ and $u(k)$ are assigned, then $x(k+1)$ and $y(k)$ are uniquely determined, which authorizes the definition of trajectories in the state or output space. In general, MLD models that are obtained from a real physical system are well posed. The formal definition of a well posed MLD system may be found in [Bemporad and Morari, 1999].

3.4 MLD modeling of resource-constrained systems

Consider the continuous-time LTI plant described by

$$\begin{aligned} \dot{x}_c(t) &= A_c x_c(t) + B_c u_c(t) + W_c w_c(t) && \text{(3.2a)} \\ y_c(t) &= C_c x_c(t) && \text{(3.2b)} \end{aligned}$$

where $x_c(t) \in \mathbb{R}^n$, $u_c(t) \in \mathbb{R}^m$, $y_c(t) \in \mathbb{R}^p$ and $w_c(t) \in \mathbb{R}^r$ represent respectively the state, the command input, the output and the disturbance input. The plant is controlled by a discrete-time controller, with sampling period T_s. The plant (3.2) and the controller are connected through a limited bandwidth communication bus. At each sampling instant $t = kT_s$ ($k \in \mathbb{N}$), the bus can carry at most b_r measures and b_w control commands, with $b_r \leq p$ and $b_w \leq m$. The input to the plant is preceded by a zero-order holder, which maintains the last received control commands constant until new control values are received. Let $u(k)$ be the input of the zero-order holder at instant kT_s, then its output is given by

$$u_c(t) = u(k) \quad \text{if } kT_s \leq t < (k+1)T_s. \qquad \text{(3.3)}$$

First, we assume that $W_c = 0$. Let $x(k) = x_c(kT_s)$ and $y(k) = y_c(kT_s)$ be respectively the sampled values of the state and the output. A discrete-time representation of the plant (3.2) at the sampling period T_s is given by

$$x(k+1) = Ax(k) + Bu(k) \tag{3.4a}$$
$$y(k) = Cx(k) \tag{3.4b}$$

where $A = e^{A_c T_s}$, $B = \int_0^{T_s} e^{A_c \tau} B_c d\tau$ and $C = C_c$.

We will assume, throughout this thesis, that the pairs (A, B) and (A, C) are respectively reachable and observable. These assumptions are systematically satisfied if the pair (A_c, B_c) is reachable, the pair (A_c, C_c) is observable and the sampling period T_s *non pathological*. A sampling period is said pathological if it causes the loss, for the sampled-data model, of the reachability and observability properties, which were verified by the continuous model before its discretization. In [Kalman et al., 1963], Kalman *et al.* have proved that the set of pathological sampling period is countable, and uniquely depends on the eigenvalues of the state matrix A_c. Consequently, in order to avoid the loss of reachability and observability, which may be caused by the sampling, it is sufficient to choose T_s outside this set.

Communication constraints may be formally described by introducing two vectors of Booleans $\sigma(k) \in \{0,1\}^{b_r}$ and $\delta(k) \in \{0,1\}^{b_w}$, defined for each sampling instant k.

Definition 3.1. The vector $\sigma(k)$ defined by

$$\begin{cases} \sigma_i(k) = 1 & \text{if } y_i(k) \text{ is read by the controller at instant } k, \\ \sigma_i(k) = 0 & \text{otherwise.} \end{cases}$$

is called sensors-to-controller scheduling vector at instant k

Definition 3.2. The vector $\delta(k)$ defined by

$$\begin{cases} \delta_i(k) = 1 & \text{is } u_i(k) \text{ updated at instant } k, \\ \delta_i(k) = 0 & \text{otherwise.} \end{cases}$$

is called controller-to-actuators scheduling vector at instant k

The vector $\sigma(k)$ indicates the measures that the controller may read at instant k. In a similar way, the $\delta(k)$ indicates the control inputs to the plant that the controller may update at instant k. The introduction of the scheduling vectors allows to model in a simple way the communication constraints. The limitations that affect the transmission of the measures to the controller may be described by the following inequality:

$$\sum_{i=1}^{p} \sigma_i(k) \leq b_r. \tag{3.5}$$

In a similar way, the limitations concerning the sending of the control commands to the actuators may be modeled by

$$\sum_{i=1}^{m} \delta_i(k) \leq b_w. \tag{3.6}$$

The last received control inputs (through the communication bus) are kept constant. Consequently, if a control input is not updated at the k^{th} sampling period, then it is maintained constant. This assertion may be modeled by the logic formula:

$$\delta_i(k) = 0 \Longrightarrow u_i(k) = u_i(k-1). \tag{3.7}$$

The plant, the analog-to-digital and digital-to-analog converters, the communication bus and the controller are schematically depicted in Figure 3.1. In this figure, $\eta(k) \in \mathbb{R}^{b_r}$ represents the vector of partial measurements that the controller receives (through the communication bus) at the sampling period k. In a similar way, vector $v(k) \in \mathbb{R}^{b_w}$ represents the vector of partial control commands that the controller may send to the actuators (through the limited bandwidth communication bus) at the sampling period k. Blocks D/A and A/D respectively represent the digital-to-analog and analog-to-digital converters. The controller may also assign the values of the sensors-to-controller scheduling vector ($\sigma(k)$) as well as the controller-to-actuators scheduling vector ($\delta(k)$).

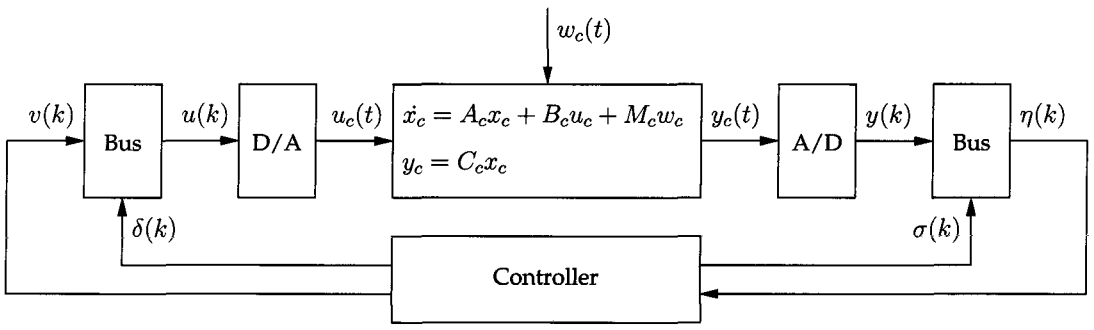


Figure 3.1: Schematic representation of a resource-constrained system

Knowing $v(k)$ and the relation (3.7), $u(k)$ is given by

$$\begin{cases} u_i(k) = v_j(k) & \text{if } \delta_i(k) = 1 \text{ and } \sum_{l=1}^{i} \delta_l(k) = j, \\ u_i(k) = u_i(k-1) & \text{otherwise.} \end{cases} \tag{3.8}$$

It may be easily verified that if v_{j_1} and v_{j_2} are mapped to respectively u_{i_1} and u_{i_2}, than $(j_1 < j_2)$ implies that $(i_1 < i_2)$.

The mapping (3.8) may be written in matrix form. Let $[D_\delta(k)]_{1 \le i \le m, 1 \le j \le b_w}$ the matrix defined by

$$\begin{cases} [D_\delta(k)]_{ij} = 1 & \text{if } \delta_i(k) = 1 \text{ and } \sum_{l=1}^{i} \delta_l(k) = j, \\ [D_\delta(k)]_{ij} = 0 & \text{otherwise,} \end{cases}$$

and

$$E_\delta(k) = \begin{bmatrix} 1 - \delta_1(k) & & \\ & \ddots & \\ & & 1 - \delta_m(k) \end{bmatrix},$$

then

$$u(k) = D_\delta(k)v(k) + E_\delta(k)u(k-1).$$

Conversely, knowing the control input $u(k)$ and the scheduling decision $\delta(k)$, the vector of control commands $v(k)$ that were sent through the bus may be determined.

Let $[M_\delta(k)]_{1\leq i\leq b_w, 1\leq j\leq m}$ the matrix defined by

$$\begin{cases} [M_\delta(k)]_{ij} = 1 & \text{if } \delta_j(k) = 1 \text{ and } \sum_{l=1}^{j} \delta_l(k) = i, \\ [M_\delta(k)]_{ij} = 0 & \text{otherwise,} \end{cases}$$

then

$$v(k) = M_\delta(k)u(k). \tag{3.9}$$

A schematic representation illustrating this mapping is given in Figure 3.2.

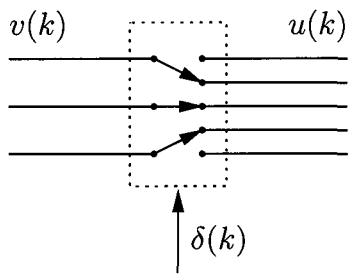


Figure 3.2: Schematic representation of the mapping of $v(k) \in \mathbb{R}^3$ to $u(k) \in \mathbb{R}^5$ corresponding to the scheduling vector $\delta(k)$ verifying $\delta_1(k) = 0$, $\delta_2(k) = 1$, $\delta_3(k) = 1$, $\delta_4(k) = 1$ and $\delta_5(k) = 0$

In the same way, the input $\eta(k)$ to the controller is defined by

$$\begin{cases} \eta_i(k) = y_j(k) & \text{if } \sigma_j(k) = 1 \text{ and } \sum_{l=1}^{j} \sigma_l(k) = i, \\ \eta_i(k) = 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

This mapping may be also written in matrix form. Let $[M_\sigma(k)]_{1\leq i\leq b_r, 1\leq j\leq p}$ the matrix defined by

$$\begin{cases} [M_\sigma(k)]_{ij} = 1 & \text{if } \sigma_j(k) = 1 \text{ and } \sum_{l=1}^{j} \sigma_l(k) = i, \\ [M_\sigma(k)]_{ij} = 0 & \text{otherwise.} \end{cases}$$

then we have the relation

$$\eta(k) = M_\sigma(k)y(k). \tag{3.11}$$

Equations (3.4), (3.5), (3.6), (3.8) and (3.10) describe a model where dynamics and plant performance are tightly coupled with the assignment of communication resources. In the particular case where $b_r = p$, $b_w = m$, $\sigma(k) = 1_{p,1}$ and $\delta(k) = 1_{m,1}$, for all $k \in \mathbb{N}$, this model coincides with the classical model of a sampled-data system. The presence of the communication bus moves the classical frontier between "the plant" and "the controller". In fact, for sampled-data systems, this frontier lies at the digital-to-analog and analog-to-digital converters. In the considered model, this frontier moves to the communication bus interface. We will call *resource-constrained system* the entity constituted by the sampled-data model of the plant and the communication bus. The formal definition of a resource-constrained system is given thereafter.

Definition 3.3. A resource-constrained system is a mixed logical dynamical system having three inputs: the command input $v(k)$, the scheduling vector of the sensors-to-controller link $\sigma(k)$ and the scheduling vector of the controller-to-actuators link $\delta(k)$. It has one output denoted $\eta(k)$. Its mathematical model is defined by:

- a recurrent equation (3.4) describing the sampled dynamics of the plant,
- inequality constraints (3.5) and (3.6) expressing the limitations of the communication medium,
- logic formulas (3.8) describing the mapping of the computed controller outputs $v(k)$ to plant inputs $u(k)$, knowing the scheduling decisions $\delta(k)$,
- logic formulas (3.10) describing the mapping of the sampled plant outputs $y(k)$ to the controller's inputs $\eta(k)$, knowing the scheduling decisions $\sigma(k)$.

The particularity of a resource-constrained system, compared to a sampled-data system, is that at each sampling period, it is important to determine:

- the measures that should be acquired (it is only possible to acquire at most b_r measures, defined by the scheduling function $\sigma(k)$),
- the control commands that should be applied (it is only possible to apply at most b_w control commands, defined by the scheduling function $\delta(k)$),
- the value of the applied control commands.

3.5 Notion of communication sequence

The notion of communication sequence was introduced by Brokett [Brockett, 1995] and generalized by Hristu [Hristu, 1999] in order to quantify the notion of *attention* [Brockett, 1997]. It characterizes the "allocation" of the bus resources to the different inputs and outputs of the system, i.e. the attention that must be given to each input and output. There are two types of communication sequences: finite communication sequences and periodic infinite communication sequences.

Definition 3.4 ([Hristu, 1999]). A finite communication sequence ρ^{N-1} of length N and width l is a finite sequence $\rho^{N-1} = (\rho(0), ..., \rho(N-1))$ of elements of $\{0,1\}^l$.

Definition 3.5 ([Hristu, 1999]). A periodic communication ρ^{T-1} sequence of period T and width l is an infinite sequence $\rho^{T-1} = (\rho(0), ..., \rho(T-1))$ of elements of $\{0,1\}^l$ verifying $\rho(k+iT) = \rho(k) \ \ \forall i \in \mathbb{N}$.

In resource-constrained system, a communication sequence is a sequence of scheduling vectors. We may distinguish between two types of communication sequences:

- sequences of sensors-to-controller scheduling vectors, which will be called *measurements communication sequence*,
- sequences of controller-to-actuator scheduling vectors, which will be called *commands communication sequence*.

Naturally, each element of a finite or periodic communication sequence must respect the communication and fairness constraints.

Definition 3.6 ([Hristu, 1999]). Let $\mathcal{S}$ be a resource-constrained system, characterized by its resource limitations b_r and b_w. The measurements communication sequence σ^{N-1} (resp. the commands communication sequence δ^{N-1}) is called *admissible* if

- $\forall k \in \mathbb{N}, 0 < \|\sigma(k)\|_2^2 \leq b_r$ (resp. $0 < \|\delta(k)\|_2^2 \leq b_w$) (non-surpassing of the bus capacity),
- $\text{Span}\{\sigma(0), \ldots, \sigma(N-1)\} = \mathbb{R}^p$ (resp. $\text{Span}\{\delta(0), \ldots, \delta(N-1)\} = \mathbb{R}^m$) (during any period N, each controller input (resp. plant input) is updated at least one time).

We introduce the notion of maximal communication sequence, in order to characterize the communication sequences that use all the available communication resources. This notion will help us to simplify the definition, formulation and solving the problems that will be tackled thereafter.

Definition 3.7. Let $\mathcal{S}$ be a resource-constrained system characterized by its resource limitations b_r and b_w. The measurements communication sequence σ^{N-1} (resp. the control communication sequence δ^{N-1}) is called *maximal* if

- $\forall k \in \mathbb{N}, 0 < \|\sigma(k)\|_2^2 = b_r$ (resp. $0 < \|\delta(k)\|_2^2 = b_w$) (usage of all the available communication resources).

3.6 State representation of resource-constrained systems

For predefined scheduling vectors σ and δ, a resource constrained system $\mathcal{S}$ may be viewed, between its input $v(k)$ and its output $\eta(k)$, as a linear time-varying system. Based on the previous definitions, and noting

$$\chi(k) = u(k-1)$$

and

$$\tilde{x}(k) = \begin{bmatrix} x(k) \\ \chi(k) \end{bmatrix},$$

the linear sampled-data and time-varying model of system $\mathcal{S}$ is given by

$$\tilde{x}(k+1) = \tilde{A}(k)\tilde{x}(k) + \tilde{B}(k)v(k) \tag{3.12a}$$
$$\eta(k) = \tilde{C}(k)\tilde{x}(k), \tag{3.12b}$$

where

$$\tilde{A}(k) = \begin{bmatrix} A & BE_\delta(k) \\ 0_{m,n} & E_\delta(k) \end{bmatrix},$$

$$\tilde{B}(k) = \begin{bmatrix} BD_\delta(k) \\ D_\delta(k) \end{bmatrix},$$

and

$$\tilde{C}(k) = M_\sigma(k) \begin{bmatrix} C & 0_{p,m} \end{bmatrix}.$$

3.7 Stabilization with limited resources

Given a resource-constrained system and a fixed periodic scheduling defined by two periodic admissible communication sequences $\delta^{T-1} = (\delta(0), \ldots, \delta(T-1))$ (commands communication sequence) and $\sigma^{T-1} = (\sigma(0), \ldots, \sigma(T-1))$ (measurements communication sequence), an interesting question is to determine whether it is possible of stabilize the discrete-time system (3.4), subject to the communication constraints (3.5), (3.6), (3.8) and (3.10), using a constant output feedback gain $\Sigma \in \mathbb{R}^{b_w \times p}$ such that $v(k) = \Sigma\bar{y}(k)$, where $\bar{y}(k) \in \mathbb{R}^p$ verifies $\bar{y}(k) = D_\sigma(k)\eta(k) + E_\sigma(k)\bar{y}(k-1)$.

Problem 3.1 ([Hristu, 1999]). Given a resource-constrained system $\mathcal{S}$, two admissible T-periodic communication sequences σ^{T-1} and δ^{T-1}, find a constant output feedback gain $\Sigma \in \mathbb{R}^{b_w \times p}$ that stabilizes system (3.4), under the communication constraints (3.5), (3.6), (3.8) and (3.10).

In [Hristu, 1999], Hristu proves that this problem is equivalent to the following NP-hard problem:

Problem 3.2 ([Hristu, 1999]). Given a collection of matrices $\mathcal{M}_i \in \mathbb{R}^{\beta \times \beta}$, $0 \leq i \leq i_{max}$ and scalars $\theta_1, ..., \theta_{i_{max}}$, find a stable element of the affine subspace:

$$\mathcal{M} = \mathcal{M}_0 + \sum_{i=1}^{i_{max}} \theta_i \mathcal{M}_i, \tag{3.13}$$

where $\beta = (2T^2 - T)n$ and $i_{max} = mp$.

The proof essentially exploits the periodicity of the communication sequences in order to represent the periodic discrete-time model by an equivalent discrete-time invariant model of higher dimension ($\beta = (2T^2 - T)n$). The procedure allowing obtaining this higher dimension invariant model (and to prove the equivalence) was called "extensification". Using this equivalence, the output-feedback stabilization of the resource-constrained system, based on two fixed T-periodic communication sequences σ^{T-1} and δ^{T-1}, amounts to find the scalars θ_i for which the spectral radius of the extensive form lies in the unit circle. A simulated annealing-based heuristic, aiming at minimizing the spectral radius of the extensive form, was proposed.

3.8 Trajectory tracking with limited resources

The N-steps output tracking problem was also studied in [Hristu, 1999]. The considered problem is a feed-forward control problem. All communication resources are allocated to the transmission of the control commands ($b = b_w$ and $b_r = 0$).

Problem 3.3 ([Hristu, 1999]). Given a resource constrained system $\mathcal{S}$ with $m \leq p$, an integer N, an admissible finite commands communication sequence δ^{N-1} of length N and width m, and a desired output $y_d \in \mathcal{L}_2^p([0, NT_s])$, find the optimal sequence of control commands $u^{N-1^*} = (u^*(0), ..., u^*(N-1))$ that minimizes $\|y - y_d\|_2$.

An analytic solution was proposed in [Hristu, 1999]. Let Υ be the operator defined by

$$\begin{aligned} \Upsilon : l_2^m(\{0,\ldots,N-1\}) &\longrightarrow \mathcal{L}_2^p[0,T] \\ u &\longmapsto y(t) = \Upsilon u \end{aligned}$$

such that

$$\Upsilon u = \sum_{k=0}^{N-1} \Theta_{T_s}(t - kT_s)u(k),$$

with

$$\Theta_{T_s}(t) = \begin{cases} \int_0^{\min(t,T_s)} C_c e^{A_c(t-\tau)} B_c d\tau & t \geq 0 \\ 0 & t < 0. \end{cases}$$

Let Υ^* be the adjoint operator of Υ in the sense of the Euclidian dot product. Υ^* is defined by

$$\begin{aligned} \Upsilon^* : \mathcal{L}_2^p[0,T] &\longrightarrow l_2^m(\{0,\ldots,N-1\}) \\ y &\longmapsto \Upsilon^* y \end{aligned}$$

such that

$$(\Upsilon^* y)(j) = \int_0^{NT_s} \Theta_{T_s}^T(t - jT_s)y(t)dt,\ j = 0,\ldots,N-1.$$

For the finite admissible communication sequence δ^{N-1}, let $\Omega(\delta)_{1\leq i\leq mN, 1\leq j\leq b_w N}$ the matrix defined by

$$\begin{cases} [\Omega(\delta)]_{ij} = 1 & \text{if } \left\lfloor \frac{i-1}{m} \right\rfloor \geq \left\lfloor \frac{j-1}{b_w} \right\rfloor, \delta(\left\lfloor \frac{j-1}{b_w} \right\rfloor)_{i-\left\lfloor \frac{i}{m+1} \right\rfloor m} = 1 \text{ and} \\ & \sum_{q=1}^{i-\left\lfloor \frac{i}{m+1} \right\rfloor m} \delta(\left\lfloor \frac{j-1}{b_w} \right\rfloor)_q = j - \left\lfloor \frac{j}{b_w+1} \right\rfloor b_w, \\ [\Omega(\delta)]_{ij} = 0 & \text{otherwise,} \end{cases}$$

then we have the following result.

Theorem 3.1 ([Hristu, 1999]). *The optimal solution u^* of Problem 3.3 is*

$$u^* = \left(\Omega(\delta)^T \Upsilon^* \Upsilon \Omega(\delta)\right)^{-1} \Omega(\delta)^T \Upsilon^* (y_d - y_{ic}),$$

where

$$y_{ic}(t) = C_c e^{A_c t} x(0) + \int_0^t C_c e^{A_c(t-\tau)} B_c u_{ic} d\tau$$

and u_{ic} is the initial conditions of the elements of $u(0)$ that are not updated at $t = 0$.

3.9 Reachability and observability with limited resources

Reachability and observability are structural properties of systems that are first imposed in control practice. Given a resource-constrained system $\mathcal{S}$, we may ask the following questions:

- If the discrete-time model (3.4) without communication constraints is reachable (resp. observable), what are the conditions allowing the resource-constrained system to keep these properties?
- Is it possible to construct periodic communication sequences guaranteeing these properties? If yes, under what conditions?

These questions were raised and answered by [Zhang and Hristu-Varsakelis, 2005] for a particular model of resource-constrained systems. In the considered model, if a control command is not received, then the corresponding actuator applies zero value, instead of maintaining constant the last received control command. The generalization of these results to resource-constrained systems, using zero-order holders (corresponding to the model that was considered in this thesis), was performed in [Ionete and Çela, 2006]. The resource-constrained system $\mathcal{S}$, viewed between its input $v(k)$ and its output $\eta(k)$ (as represented by equations (3.12)) being linear time-varying, it is necessary to use reachability and observability notions that are suitable for linear time-varying systems.

Definition 3.8 ([Rugh, 1996]). A linear discrete-time system is called l-step reachable (resp. l-step observable) if l is a positive integer such that the system is reachable (resp. observable) on $[i, i+l]$, for any i.

Theorem 3.2 ([Ionete and Çela, 2006]). *If A is invertible and the pair (A, B) is reachable. Then for any integer b_w such that $1 \leq b_w \leq m$, there exist integers $l, T > 0$ and a maximal T-periodic communication sequence of width m such that system (3.12) is l-step reachable.*

In practice, matrix A is obtained by digitalization of matrix A_c, then it is invertible. The proof of this theorem is obtained by construction. Consequently, it is possible to employ it in order to construct periodic communication sequences that guarantee the reachability. A similar result is obtained for the observability.

Theorem 3.3 ([Ionete and Çela, 2006]). *If A is invertible and the pair (A, C) is observable. Then for any b_r such that $1 \leq b_r \leq p$, there exist integers $l, T > 0$ and a maximal T-periodic communication sequence of width p such that system (3.12) is l-step observable.*

3.10 Conclusion

In this chapter, we have presented an abstract model of a distributed embedded control system operating under communication constraints. We have proposed a systematic approach allowing to formally describing it in the MLD framework. The originality of the considered model comes from the fact that communication resources allocation is viewed as an input, as well as the control commands. The interaction between control and scheduling is thus explicitly taken into account. We then reviewed the various theoretical results established

in the literature on this model and which include the problems of stabilization, of trajectory tracking, reachability and observability.

Thereafter, we will focus on the possibility that is offered by this model to assign at the same time the control and scheduling. In the next chapter, we will consider the problem of optimal integrated control and scheduling, disregarding practical implementation constraints. The resource-constrained systems being an extension of sampled-data systems, the problem of the joint optimization of control and scheduling may be seen like a natural extension of classical optimal control problems.

4

Optimal integrated control and scheduling of resource-constrained systems

4.1 Introduction

The general model of resource-constrained systems, as described in the previous chapter, allows the on-line assignment of the sensors-to-controller and the controller-to-actuators scheduling vectors. This assignment may be based on a pre-computed off-line schedule or on an on-line scheduling algorithm. The problem of the optimal integrated control and *off-line* scheduling of the sensors-to-controller link is the dual problem of the optimal integrated control and off-line scheduling of the controller-to-actuators link, and may be solved in a similar way. However, the problem of the optimal integrated control and *on-line* scheduling of the sensors-to-controller link is different from the problem of the optimal integrated control and on-line scheduling of the controller-to-actuators link, and its formulation and solving are extremely dependant on the used technology (possibility of on-line arbitration based on the comparison of messages identifiers like in CAN networks for example). Furthermore, it is not clear whether an on-line scheduling algorithm of the sensors-to-controller link will be better (from a control performance point of view) than a predefined off-line scheduling algorithm, especially, because the sensors are physically distributed whereas the optimal on-line assignment of the sensors-to-controller link requires the knowledge of all sensors values, which may not be possible due to the communication constraints. This latter question remains an open problem.

For that reason, in this chapter, as well as in Chapters 5 and 6, we will assume that the state of the plant is available to the controller at each sampling period. This assumption will allow us to mainly focus on the problem of the optimal integrated control and scheduling of the controller-to-actuators link. Note that assuming that the state of the plant is available to the controller at each sampling period does not necessarily mean that we are restricted to a particular architecture, but rather that state of the art methods (for example, an off-line scheduling) are deployed to obtain a satisfactory estimate of the state at the controller, using an adequate part of the bandwidth. Let $\mathcal{S}$ be a resource-constrained system verifying

this assumption. $\mathcal{S}$ verifies $p = n = b_r$, $\sigma(k) = 1_{n,1}, \forall k \in \mathbb{N}$ and $0 < b_w \leq m$. To simplify the notation, let $b = b_w$. In the model that will be considered in the following, the resource-constrained system $\mathcal{S}$ has two types of inputs: control inputs and scheduling inputs of the controller-to-actuators link. The fundamental problem that will be considered is the problem of the joint optimization of control and scheduling. This problem may be viewed as the generalization of optimal control problems for linear sampled-data system. It naturally appears as a hybrid problem, where continuous aspects (system dynamics) as well as logical aspects (scheduling) interact.

Using the basic results of optimal control theory, we first describe the solution of the optimal control problem given a fixed communication sequence, over finite and infinite horizons. Then, the problem of the optimal integrated control and scheduling of resource-constrained systems is formulated and solved. The formulation and solving of this problem are based on the existing theoretical tools from hybrid systems theory and especially the MLD framework [Bemporad and Morari, 1999], where this problem may be perfectly modeled. Finally, based on a numerical example, the solutions of this problem are studied.

4.2 Performance index definition

In order to quantify the "Quality" of the control and scheduling, a quadratic cost function is associated to system (3.2) (and implicitly to the resource-constrained system $\mathcal{S}$).

$$J_c(x_c, u_c, 0, T_f) = \int_0^{T_f} \left(x_c^T(t) Q_c x_c(t) + u_c^T(t) R_c u_c(t)\right) dt + x_c^T(T_f) S_c x_c(T_f) \tag{4.1}$$

where $T_f = NT_s$ and Q_c, R_c and S_c are positive definite matrices. These matrices define the design specifications of the ideal continuous-time controller. The sampled-data representation of the cost function $J_c(x_c, u_c, 0, T_f)$ at the sampling period T_s is

$$J(x, u, 0, N) = \sum_{k=0}^{N-1} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}^T & Q_2 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + x^T(N) Q_0 x(N). \tag{4.2}$$

The expressions of Q_1, Q_2, Q_{12} and Q_0 may be found in ([Åström and Wittenmark, 1997], pp. 411–412). In the following, it is assumed that Q, Q_2 and Q_0 are positive definite matrices, where

$$Q = \begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}^T & Q_2 \end{bmatrix}.$$

Note that this representation does not involve any approximation and is exact. Using this representation, the inter-sample behavior is taken into account.

4.3 Optimal control over a finite horizon for a fixed communication sequence

The problem of the optimal control, over a finite-time N, for a fixed admissible and maximal communication sequence δ^{N-1}, may be formulated as follows.

Problem 4.1. Given an initial state $x(0)$ and a final time N, find the optimal control sequence $v^{N-1} = (v(0), \ldots, v(N-1))$ that minimizes the cost function

$$J(x,u,0,N) = \sum_{k=0}^{N-1} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}^T & Q_2 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + x^T(N) Q_0 x(N),$$

subject to

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ u_i(k) &= v_j(k) \qquad \text{if } \delta_i(k) = 1 \text{ and } \sum_{l=1}^{i} \delta_l(k) = j \\ u_i(k) &= u_i(k-1) \qquad \text{otherwise.} \end{aligned}$$

In order to solve this problem, we re-consider the state representation (3.12) of system $\mathcal{S}$, which was established in Section 3.6 of Chapter 3. For a given maximal controller-to-actuators scheduling δ, system $\mathcal{S}$ is described by the state equation (3.12a). The cost function $J(x,u,0,N)$ may be re-written in the form

$$J(x,u,0,N) = J(\tilde{x},v,0,N) = \sum_{k=0}^{N-1} \begin{bmatrix} \tilde{x}(k) \\ v(k) \end{bmatrix}^T \tilde{Q}(k) \begin{bmatrix} \tilde{x}(k) \\ v(k) \end{bmatrix} + \tilde{x}^T(N) \tilde{Q}_0 \tilde{x}(N),$$

where

$$\tilde{Q}(k) = \begin{bmatrix} Q_1 & Q_{12}E_\delta(k) & Q_{12}D_\delta(k) \\ E_\delta^T(k)Q_{12}^T & E_\delta^T(k)Q_2E_\delta(k) & E_\delta^T(k)Q_2D_\delta(k) \\ D_\delta^T(k)Q_{12}^T & D_\delta^T(k)Q_2E_\delta(k) & D_\delta^T(k)Q_2D_\delta(k) \end{bmatrix}$$

and

$$\tilde{Q}_0 = \begin{bmatrix} Q_0 & 0_{n,m} \\ 0_{m,n} & 0_{m,m} \end{bmatrix}.$$

Problem 4.1 is equivalent to the optimal control problem of a discrete linear time-varying system, described by

Problem 4.2. Given an initial state $x(0)$ and a final time N, find the optimal control sequence $v^{N-1} = (v(0), \ldots, v(N-1))$ that minimizes the cost function

$$J(\tilde{x},v,0,N) = \sum_{k=0}^{N-1} \begin{bmatrix} \tilde{x}(k) \\ v(k) \end{bmatrix}^T \tilde{Q}(k) \begin{bmatrix} \tilde{x}(k) \\ v(k) \end{bmatrix} + \tilde{x}^T(N) \tilde{Q}_0 \tilde{x}(N)$$

subject to

$$\tilde{x}(k+1) = \tilde{A}(k)\tilde{x}(k) + \tilde{B}(k)v(k). \tag{4.3}$$

Problem 4.2 is a classical optimal control problem of a discrete linear time-varying system. Different methods for its resolution were developed in control textbooks, like [Åström and Wittenmark, 1997] for example. Let

$$\tilde{Q}_1(k) = \begin{bmatrix} Q_1 & Q_{12}E_\delta(k) \\ E_\delta^T(k)Q_{12}^T & E_\delta^T(k)Q_2E_\delta(k) \end{bmatrix},$$

$$\tilde{Q}_{12}(k) = \begin{bmatrix} Q_{12}D_\delta(k) \\ E_\delta^T(k)Q_2D_\delta(k) \end{bmatrix},$$

and

$$\tilde{Q}_2(k) = D_\delta^T(k)Q_2D_\delta(k).$$

The solution of this problem closely depends on the solution of an algebraic equation, involving the variable $\tilde{S}(k)$ and described by

$$\begin{aligned} \tilde{S}(k) = \tilde{A}^T(k)\tilde{S}(k+1)\tilde{A}(k) + \tilde{Q}_1(k) - \left(\tilde{A}^T(k)\tilde{S}(k+1)\tilde{B}(k) + \tilde{Q}_{12}(k)\right) \\ \times \left(\tilde{B}^T(k)\tilde{S}(k+1)\tilde{B}(k) + \tilde{Q}_2(k)\right)^{-1} \left(\tilde{B}^T(k)\tilde{S}(k+1)\tilde{A}(k) + \tilde{Q}_{12}^T(k)\right) \end{aligned} \tag{4.4}$$

under the terminal condition

$$\tilde{S}(N) = \tilde{Q}_0. \tag{4.5}$$

Equation (4.4) is the discrete algebraic Riccati equation associated to Problem 4.2. Knowing that $\tilde{Q}_0$ is semi definite positive and $\tilde{Q}_2(k)$ is definite positive (because $D_\delta(k)$ is injective when δ is a maximal admissible communication sequence), than this equation admits a unique positive semi-definite solution. Consequently, Problem 4.2 admits a unique solution [Åström and Wittenmark, 1997] defined by

$$v(k) = -\tilde{K}(k)\tilde{x}(k) \tag{4.6}$$

with

$$\tilde{K}(k) = \left(\tilde{Q}_2(k) + \tilde{B}^T(k)\tilde{S}(k+1)\tilde{B}(k)\right)^{-1} \left(\tilde{B}^T(k)\tilde{S}(k+1)\tilde{A}(k) + \tilde{Q}_{12}^T(k)\right) \tag{4.7}$$

4.4 Optimal control over an infinite horizon for a fixed communication sequence

Consider a T-periodic maximal communication sequence δ^{T-1} defined by

$$\delta^{T-1} = (\delta(0), \cdots, \delta(T-1))$$

and verifying $\delta(k+T) = \delta(k)$. Assume furthermore that $\delta \in \mathcal{S}^c$, where $\mathcal{S}^c$ is the set of communication sequences that guarantee the reachability of system (3.12). The periodicity of the communication sequence induces the periodicity of the resource-constrained system $\mathcal{S}$. As a result, matrices $\tilde{A}(k)$, $\tilde{B}(k)$ and $\tilde{Q}(k)$ satisfy $\tilde{A}(k+T) = \tilde{A}(k)$, $\tilde{B}(k+T) = \tilde{B}(k)$ and $\tilde{Q}(k) = \tilde{Q}(k+T)$.

Let ι be a discrete time instant and H a positive integer, assume that $N = \iota + HT$ and consider the optimal control problem:

$$\begin{cases} \min\limits_{v} J(\tilde{x}, v, \iota, N) = \sum\limits_{k=\iota}^{N-1} \begin{bmatrix} \tilde{x}(k) \\ v(k) \end{bmatrix}^T \tilde{Q}(k) \begin{bmatrix} \tilde{x}(k) \\ v(k) \end{bmatrix} + \tilde{x}^T(N)\tilde{Q}_0\tilde{x}(N) \\ \text{subject to} \\ \tilde{x}(k+1) = \tilde{A}(k)\tilde{x}(k) + \tilde{B}(k)v(k). \end{cases} \tag{4.8}$$

As illustrated in [Bittanti et al., 1991], a time invariant reformulation of the optimal control problem (4.8) may be obtained using the lifting technique. The time invariant reformulation may be seen as a down sampled representation of system (3.12) with periodicity T,

having an augmented input vector. In the following, the formulation of the time invariant representation is described.

Let Φ be the transition matrix associated with the state matrix $\tilde{A}$. Φ is defined by

$$\begin{cases} \Phi(l,s) = \tilde{A}(l-1)\tilde{A}(l-2)\cdots\tilde{A}(s) & \text{if } l > s \\ \Phi(l,l) = I_{n+m}. \end{cases}$$

Let Γ the matrix defined for $s < l < s+T$ by

$$\Gamma(l,s) = \left[\Phi(l,s+1)\tilde{B}(s) \;\; \Phi(l,s+2)\tilde{B}(s+1) \;\; \cdots \;\; \Phi(l,l)\tilde{B}(l-1) \;\; \underbrace{0_{n+m,b} \;\; \cdots \;\; 0_{n+m,b}}_{T-l-s}\right]$$

and for $s = l$ by

$$\Gamma(s,s) = [0_{n+m,b} \;\; \cdots \;\; 0_{n+m,b}].$$

Let

$$\bar{x}_\iota(q) = \tilde{x}(\iota + qT)$$

and

$$\bar{v}_\iota(q) = \begin{bmatrix} v(\iota+qT) \\ \vdots \\ v(\iota+(q+1)T-1) \end{bmatrix}$$

than for $0 \leq i \leq T$:

$$\tilde{x}(\iota+qT+i) = \Phi(\iota+i,\iota)\bar{x}_\iota(q) + \Gamma(\iota+i,\iota)\bar{v}_\iota(q).$$

In particular, let

$$\bar{A}_\iota = \Phi(\iota+T,\iota)$$

and

$$\bar{B}_\iota = \Gamma(\iota+T,\iota)$$

then the following relation is obtained:

$$\bar{x}_\iota(q+1) = \bar{A}_\iota\bar{x}_\iota(q) + \bar{B}_\iota\bar{v}_\iota(q).$$

Let $\Lambda(i)$ the matrix defined for $0 \leq i < T$ by

$$\Lambda(i) = \left[\underbrace{0_{b,b} \;\; \cdots \;\; 0_{b,b}}_{i} I_b \underbrace{0_{b,b} \;\; \cdots \;\; 0_{b,b}}_{T-i-1}\right],$$

then the cost function may be written

$$J(\tilde{x},v,\iota,N) = J(\bar{x}_\iota,\bar{v}_\iota,0,H) = \sum_{q=0}^{H-1} \begin{bmatrix} \bar{x}_\iota(q) \\ \bar{v}_\iota(q) \end{bmatrix}^T \bar{Q}_\iota \begin{bmatrix} \bar{x}_\iota(q) \\ \bar{v}_\iota(q) \end{bmatrix} + \bar{x}_\iota^T(H)(\bar{Q}_\iota)_0\bar{x}_\iota(H)$$

where

$$\bar{Q}_\iota = \sum_{i=0}^{T-1} F^T(i)\tilde{Q}(\iota+i)F(i)$$

$$F(i) = \begin{bmatrix} \Phi(\iota+i,\iota) & \Gamma(\iota+i,\iota) \\ 0_{b,n+m} & \Lambda(i) \end{bmatrix}$$

and $(\bar{Q}\iota)_0 = \tilde{Q}_0$. Finally, the following optimal control problem is obtained

$$\begin{cases} \min\limits_{\bar{v}_\iota} J(\bar{x}_\iota, \bar{v}_\iota, 0, H) = \bar{x}_\iota^T(H)(\bar{Q}\iota)_0\bar{x}_\iota(H) + \sum\limits_{q=0}^{H-1} \begin{bmatrix} \bar{x}_\iota(q) \\ \bar{v}_\iota(q) \end{bmatrix}^T \bar{Q}_\iota \begin{bmatrix} \bar{x}_\iota(q) \\ \bar{v}_\iota(q) \end{bmatrix} \\ \text{subject to} \\ \bar{x}_\iota(q+1) = \bar{A}_\iota\bar{x}_\iota(q) + \bar{B}_\iota\bar{v}_\iota(q). \end{cases} \tag{4.9}$$

The corresponding discrete algebraic Riccati equation is given by

$$\begin{aligned} \bar{S}_\iota(q) = \bar{A}_\iota^T\bar{S}_\iota(q+1)\bar{A}_\iota + \bar{Q}_{\iota_1} - \left(\bar{A}_\iota^T\bar{S}_\iota(q+1)\bar{B}_\iota + \bar{Q}_{\iota_{12}}\right) \\ \times \left(\bar{B}_\iota^T\bar{S}_\iota(q+1)\bar{B}_\iota + \bar{Q}_{\iota_2}(q)\right)^{-1}\left(\bar{B}_\iota^T\bar{S}_\iota(q+1)\bar{A}_\iota + \bar{Q}_{\iota_{12}}^T\right) \end{aligned} \tag{4.10}$$

with the terminal condition

$$\bar{S}_\iota(H) = \left(\bar{Q}_\iota\right)_0 \tag{4.11}$$

where

$$\bar{Q}_\iota = \begin{bmatrix} \bar{Q}_{\iota_1} & \bar{Q}_{\iota_{12}} \\ \bar{Q}_{\iota_{12}}^T & \bar{Q}_{\iota_2} \end{bmatrix}.$$

The relationship between the solutions of Riccati equations (4.4) and (4.10), which are respectively associated to optimal control problems (4.8) and (4.9), is formalized by the following result.

Lemma 4.1 ([Bittanti et al., 1991]). *If $\bar{S}_\iota(H) = \tilde{S}(\iota+NT) = \tilde{Q}_0$, then $\bar{S}_\iota(q) = \tilde{S}(\iota+qT)$ for all $q \leq H$.*

This result follows from the fact that optimal control problems (4.8) and (4.9) are coincident. By imposing the terminal condition $\bar{S}_\iota(H) = \tilde{S}(\iota+NT) = \tilde{Q}_0$, called *periodic generator*, the optimal costs must be identical, which implies that the solutions of the Riccati equations (4.4) and (4.10) must be the same. As a result, when $H \to +\infty$, $\bar{S}_\iota(q)$ converges to a constant solution $\bar{S}_\iota$. Consequently, $\tilde{S}(k)$ converges to a periodic solution, defined by $\tilde{S}(k) = \bar{S}_{(k \mod T)}$. This periodic solution may be obtained by solving the algebraic Riccati equation associated with problem (4.9) when $H \to +\infty$ and for $\iota \in \{1,\ldots,T\}$. The optimal control gains may then be deduced from the relation (4.7) and are described by the sequence $\left(\tilde{K}(0),\ldots,\tilde{K}(T-1)\right)$.

This formulation will be of practical importance at the next chapters when the problems of the optimal integrated control and scheduling of resource-constrained systems will be tackled.

4.5 Finite-time optimal integrated control and scheduling

In this paragraph, the problem of the finite-time optimal control and scheduling is formulated and translated into the mixed-integer quadratic programming (MIQP) formulation. We assume in the following that $u(k) = 0$ and $v(k) = 0$ for $k < 0$, and that control commands $u(k)$ and $v(k)$ are subject to saturation constraints

$$L_i \leq u_i(k) \leq U_i \tag{4.12}$$

$$L_i \leq v_i(k) \leq U_i \tag{4.13}$$

where $L_i < 0$ and $U_i > 0$.

4.5.1 Problem formulation

The finite-time optimal control and scheduling problem may be formalized as follows:

Problem 4.3. Given an initial state $x(0)$ and a final time N, find the optimal control sequence $v^{N-1} = (v(0), \ldots, v(N-1))$ as well as the optimal communication sequence $\delta^{N-1} = (\delta(0), \ldots, \delta(N-1))$ which minimize the performance index

$$J(x,u,0,N) = \sum_{k=0}^{N-1} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}^T & Q_2 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + x^T(N) Q_0 x(N)$$

subject to

$$\begin{aligned} & x(k+1) = Ax(k) + Bu(k) \\ & \sum_{i=1}^{m} \delta_i(k) = b \\ & u_i(k) = v_j(k) \qquad \text{if } \delta_i(k) = 1 \text{ and } \sum_{l=1}^{i} \delta_l(k) = j \\ & u_i(k) = u_i(k-1) \qquad \text{otherwise} \\ & L_i \leq v_i(k) \leq U_i. \end{aligned}$$

The problem of finding the optimal control sequence v^{N-1} for a given fixed communication sequence δ^{N-1} is a quadratic programming (QP) problem. The number of possible communication sequences is finite. The resolution of Problem 4.3 may be reduced to the exploration of all the feasible maximal communication sequences and the solving of a QP problem for each fixed communication sequence. However, in practice, the number of feasible communication sequences grows exponentially with N, which means that exhaustive search may not be applied to problems with large values of N.

The solution of Problem 4.3 may be obtained through the solving of a simpler optimization problem, which may be seen as a constrained control problem, where the variables v^{N-1} are eliminated and the constraint (3.8) is replaced by (3.7). Let $u^{N-1} = (u(0), \cdots u(N-1))$, this problem may be stated as follows.

Problem 4.4. Given an initial state $x(0)$ and a final time N, find the optimal control sequence u^{N-1} as well as the optimal scheduling sequence δ^{N-1} that minimize the performance index

$$J(x,u,0,N) = \sum_{k=0}^{N-1} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}^T & Q_2 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + x^T(N) Q_0 x(N)$$

subject to

$$\begin{aligned} & x(k+1) = Ax(k) + Bu(k) \\ & \sum_{i=1}^{m} \delta_i(k) = b \\ & \delta_i(k) = 0 \Longrightarrow u_i(k) = u_i(k-1) \\ & L_i \leq v_i(k) \leq U_i. \end{aligned}$$

In order to solve this problem, it is necessary to translate the logical formula (3.7) into linear inequalities. The connective "$\Longrightarrow$" may be eliminated if (3.7) is rewritten in the equivalent form

$$u_i(k) - u_i(k-1) = \delta_i(k)u_i(k) - \delta_i(k)u_i(k-1). \tag{4.14}$$

However, equation (4.14) contains terms which are the product of logical variables and continuous variables. Using the procedure described in [Bemporad and Morari, 1999], this product is translated into an equivalent conjunction of linear inequalities. For example, let

$$\xi_i(k) = \delta_i(k)u_i(k). \tag{4.15}$$

Then (4.14) may be rewritten in the equivalent form

$$\begin{array}{l} \xi_i(k) \leq U_i\delta_i(k) \\ \xi_i(k) \geq L_i\delta_i(k) \\ \xi_i(k) \leq u_i(k) - L_i(1-\delta_i(k)) \\ \xi_i(k) \geq u_i(k) - U_i(1-\delta_i(k)). \end{array} \tag{4.16}$$

Note that the same procedure may be applied to

$$o_i(k) = \delta_i(k)u_i(k-1). \tag{4.17}$$

Let

$$\check{\Delta} = \begin{bmatrix} \delta(0) \\ \vdots \\ \delta(N-1) \end{bmatrix}$$

$$\check{U} = \begin{bmatrix} u(0) \\ \vdots \\ u(N-1) \end{bmatrix}$$

$$\check{X} = \begin{bmatrix} x(0) \\ \vdots \\ x(N) \end{bmatrix}$$

$$\check{\Xi} = \begin{bmatrix} \xi(0) \\ \vdots \\ \xi(N-1) \end{bmatrix}$$

$$\check{O} = \begin{bmatrix} o(0) \\ \vdots \\ o(N-1) \end{bmatrix}$$

and

$$\mathcal{V} = \begin{bmatrix} \check{\Delta} \\ \check{U} \\ \check{X} \\ \check{\Xi} \\ \check{O} \end{bmatrix},$$

-

then Problem 4.4 may be written in the form

$$\begin{cases} \min\limits_{\mathcal{V}} \frac{1}{2}\mathcal{V}^T\mathcal{F}\mathcal{V} + \mathcal{G}^T\mathcal{V} \\ \mathcal{A}\mathcal{V} \leq \mathcal{B} \\ \mathcal{V}_i \in \{0,1\}\ , \forall i \in \{1,\ldots,mN\}\,. \end{cases} \tag{4.18}$$

where the expressions $\mathcal{F}$, $\mathcal{G}$, $\mathcal{A}$ and $\mathcal{B}$ are defined in Appendix A.

Problem 4.3 is identical to problem 4.4 augmented with the additional constraint

$$v(k) = u^f(k)$$

where $u^f(k) \in \mathbb{R}^b$ the vector containing the b "free" elements of $u(k)$ (i.e. the elements of $u(k)$ whose indices i satisfy $\delta_i(k) = 1$), arranged according to the increasing order of their indices. As a consequence, the optimal solutions of Problem 4.3 may be deduced from the optimal solutions of Problem 4.4 using the mapping (3.9).

Problem (4.18) is a mixed-integer quadratic program. It may be solved using many efficient academic and commercial solvers. In this thesis, this problem was solved using the solver CPLEX, whose MIP solver is based on the branch and bound method.

4.5.2 The branch and bound method

The branch and bound method is a general search method for finding optimal solutions of discrete and combinatorial optimization problems. It was introduced in 1960 by Land and Doig [Land and Doig, 1960] for the solving of the traveling salesman problem. This method aims at exploring, in an intelligent way, the space of feasible solutions of the problem. That's why it may be classified among the *implicit enumeration* methods. In the following, the principles of this algorithm will be described in the case of a minimization. In order to simplify our explanation, we will suppose that considered problem admits at least one optimal solution. Of course, the other cases may be easily taken into account.

General concepts

As its name indicates it, this method is based on two complementary mechanisms: *brancing* and *bouding*.

- Branching makes it possible to decompose a given problem into subproblems (by adding additional constraints), such that the union of the feasible solutions of these subproblems forms a partition (in the worst case a covering) of the feasible solutions of the original problem. In this manner, the resolution of the original problem is reduced to the resolution of the subproblems obtained by its branching. Branching induces a hierarchical relation between the different subproblems. This relation may be described and represented using concepts from the graph theory. In fact, in the branch and bound method, branching is applied in a recursive way to the subproblems where it may be possible (at a given stage of the execution of the algorithm), to find an optimal solution of the initial problem. As a result, the subproblems obtained by branching may be seen as the *child nodes* of the problem to which branching was applied, which is called *parent node*. Thus, all these nodes form a rooted tree, whose *root node* represents the initial problem to solve. This tree is usually called *search tree* or *decision tree*.

- Bounding consists on computing an upper and a lower bound of the optimal solution of a given node. The bounding stage allows the branch and bound to avoid exploring the nodes where it is possible to certify that they do not contain any optimal solution. In fact, if the upper bound of a subproblem **A** is larger than the lower bound of another subproblem **B**, then the optimal solution cannot lie in the feasible set of solutions of subproblem **A**. For that reason, it becomes useless to branch node **A**. Node **A** is then *pruned*.

A node is called *solved* if an optimal solution of the associated subproblem was obtained. This may occur for example when the constraints that define it (and which were added progressively along the different branching steps), reduces its set of feasible solutions to a singleton. In other situations, branching may make the subproblem sufficiently simple to be solved by polynomial algorithms. The solved nodes are the *final nodes* or *leave nodes* of the decision tree. The algorithm finishes when all the nodes were either solved or pruned.

***Example* 4.1.** Figure 4.1 describes the search tree of problem $\mathcal{P}$ defined by

$$\mathcal{P} \begin{cases} \min\limits_{x} 4x_1^2 - 3x_2^2 \\ x_1 = 0 \\ x_1, x_2 \in \{0, 1\}. \end{cases}$$

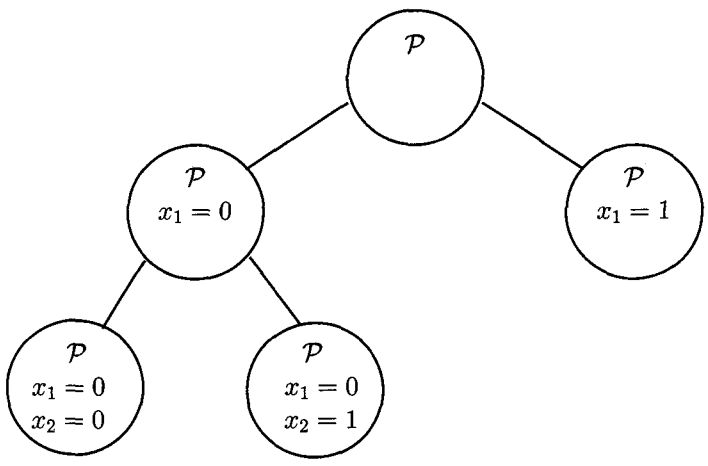


Figure 4.1: Search tree of problem $\mathcal{P}$

The branch and bound algorithm may be parameterized using the four following rules:

- *branching rules*, describing how to divide a given problem into subproblems,
- *bounding rules*, defining how to compute the upper and lower bounds of a given subproblem,
- *selection rules*, stating how to select the next problem to consider,
- *elimination rules*, describing how to recognize the subproblems that do not contain optimal solutions and that should be eliminated.

Application to mixed-integer programming

The application of the branch and bound method to the solving of mixed-integer nonlinear programs was proposed for the first time by Dakin [Dakin, 1965]. The resolution of mixed-integer quadratic programs was studied by many authors, for example [Lazimy, 1985, Flippo and Rinnoy Kan, 1990, Fletcher and Leyffer, 1998]. In the paper by Fletcher and Leyffer [Fletcher and Leyffer, 1998], the branch and bound method was applied for solving mixed-integer quadratic programs, allowing obtaining better experimental results than the other methods.

In this paragraph, we consider programs in the form:

$$\mathcal{P}\begin{cases} \min\limits_{\mathcal{V}} f(\mathcal{V}) = \frac{1}{2}\mathcal{V}^T\mathcal{F}\mathcal{V} + \mathcal{G}^T\mathcal{V} \\ \mathcal{A}\mathcal{V} \leq \mathcal{B} \\ \mathcal{V}_i \in \{0,1\}\,, \forall i \in I \end{cases}$$

where f is a positive semi-definite function and I is a set of indices indexing the Boolean components of $\mathcal{V}$. A basic branch and bound algorithm for solving program $\mathcal{P}$ is given in the following listing (Algorithm 4.1).

$\mathcal{L} := \{\mathcal{P}\}$;
$\mathcal{M} := \{\mathcal{P}\}$;
$U := +\infty$;
$L := -\infty$;
while $\mathcal{L} \neq \emptyset$ *or* $U - L > \varepsilon$ **do**
  select a node $\mathcal{Q}$ from list $\mathcal{L}$;
  remove the selected node $\mathcal{Q}$ from $\mathcal{L}$;
  solve the relaxed program $\mathcal{Q}'$ (bouding or evaluation of $\mathcal{Q}$) ;
  if $\mathcal{Q}'$ *admits an optimal solution* $x^*(\mathcal{Q}')$ *of cost* $J^*(\mathcal{Q}')$ *and* $J^*(\mathcal{Q}') < U$ **then**
    lowerbound$(\mathcal{Q}) := J^*(\mathcal{Q}')$;
    if $x^*(\mathcal{Q}')$ *is integer-feasible* **then**
      $U := J^*(\mathcal{Q}')$;
      $x^*(\mathcal{P}) := x^*(\mathcal{Q}')$;
      $J^*(\mathcal{P}) := J^*(\mathcal{Q}')$;
    else
      branch node $\mathcal{Q}$ and update list $\mathcal{L}$ with its child nodes ;
    endif
    if *all the brother nodes of* $\mathcal{Q}$ *were evaluated* **then**
      remove the father node of $\mathcal{Q}$ from $\mathcal{M}$;
      put $\mathcal{Q}$ and its brother nodes in $\mathcal{M}$;
      $L = \min\{\text{lowerbound}(\mathcal{Q}), \mathcal{Q} \in \mathcal{M}\}$;
    endif
  else
    lowerbound$(\mathcal{Q}) := +\infty$ (pruning of $\mathcal{Q}$);
  endif
endw

Algorithm 4.1: Basic branch and bound algorithm

In Algorithm 4.1, U represents the cost of the best feasible solution of problem $\mathcal{P}$ at a given stage of the execution of the algorithm and L a lower bound of the optimal cost of $\mathcal{P}$.

The set $\mathcal{M}$, used for the computation of L, represents the set of nodes that have been already evaluated (i.e. bounded) and that may contain the optimal solution. Problem $\mathcal{P}'$ denotes the problem obtained from $\mathcal{P}$ by relaxing the integrality constraints that are imposed on the variables indexed by I, i.e. replacing the constraints

$$\mathcal{V}_i \in \{0,1\} \ , \forall i \in I$$

by the constraints

$$\mathcal{V}_i \in [0,1] \ , \forall i \in I.$$

These variables are thus considered as continuous variables in $[0,1]$. $\mathcal{P}'$ is a quadratic program, and may be solved efficiently. Its solving is the most important part of the bounding or evaluation phase. In fact, the optimal cost of $\mathcal{P}'$ represents a lower bound of the optimal cost of $\mathcal{P}$.

The algorithm begins by solving $\mathcal{P}'$ and giving its solution $x^*(\mathcal{P}')$. If this solution respects the integrality constraints, than it is also an optimal solution for $\mathcal{P}$ and the algorithm terminates. Otherwise, there exists at least one non Boolean variable $\mathcal{V}_j$, $j \in I$ in $x^*(\mathcal{P}')$. The algorithm proceeds by branching problem $\mathcal{P}$ into two subproblems

$$\mathcal{Q}_1 \begin{cases} \min\limits_{\mathcal{V}} \frac{1}{2}\mathcal{V}^T\mathcal{F}\mathcal{V} + \mathcal{G}^T\mathcal{V} \\ \mathcal{A}\mathcal{V} \leq \mathcal{B} \\ \mathcal{V}_j = 0 \\ \mathcal{V}_i \in \{0,1\} \, , \forall i \in I_1 = I - \{j\} \end{cases}$$

and

$$\mathcal{Q}_2 \begin{cases} \min\limits_{\mathcal{V}} \frac{1}{2}\mathcal{V}^T\mathcal{F}\mathcal{V} + \mathcal{G}^T\mathcal{V} \\ \mathcal{A}\mathcal{V} \leq \mathcal{B} \\ \mathcal{V}_j = 1 \\ \mathcal{V}_i \in \{0,1\} \, , \forall i \in I_2 = I - \{j\} \end{cases}$$

that are added to list $\mathcal{L}$. The procedure is reiterated by the selection of a subproblem $\mathcal{Q}$ from $\mathcal{L}$, based on the predefined selection rules. It may be possible that the relaxed subproblem $\mathcal{Q}'$ does not have any feasible solution. In this situation, it is pruned from list $\mathcal{L}$ without being branched. The algorithm finishes when list $\mathcal{L}$ becomes empty or when a suboptimal solution with a predefined absolute tolerance (defined by ε) is found.

4.5.3 A numerical example

Consider the collection of 3 continuous-time LTI subsystems defined by

$$A_c^{(1)} = \begin{bmatrix} 0 & 130 \\ -800 & 10 \end{bmatrix} \quad B_c^{(1)} = \begin{bmatrix} 0 \\ 224 \end{bmatrix} \quad S^{(1)}$$

$$A_c^{(2)} = \begin{bmatrix} 0 & 14 \\ -250 & -200 \end{bmatrix} \quad B_c^{(2)} = \begin{bmatrix} 0 \\ 620 \end{bmatrix} \quad S^{(2)}$$

$$A_c^{(3)} = \begin{bmatrix} 0 & 0 & 0 & 100 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & -10 & 0 \\ 11.6 & 0 & 1.184 & 0 \end{bmatrix} \quad B_c^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 10 \\ 10.18 \end{bmatrix} \quad S^{(3)}$$

subsystems $S^{(1)}$ and $S^{(3)}$ are open-loop unstable in opposition to subsystem $S^{(2)}$ which is open-loop stable. The global system S composed by $S^{(1)}$, $S^{(2)}$ and $S^{(3)}$ may be described by the state matrix A_c and the input matrix B_c defined by

$$A_c = \text{Diag}\left(A_c^{(1)}, A_c^{(2)}, A_c^{(3)}\right)$$

and

$$B_c = \text{Diag}\left(B_c^{(1)}, B_c^{(2)}, B_c^{(3)}\right).$$

The global system S is controlled by a discrete-time controller executed at the sampling period $T_s = 2$ ms. The control commands are sent to the actuators through a bus with that can carry at most one control command every 2 ms ($b = b_w = 1$). Design criteria of the optimal continuous time controller for each subsystem are defined by matrices $Q_c^{(1)} = \text{Diag}(100, 10)$, $R_c^{(1)} = 1$, $Q_c^{(2)} = \text{Diag}(1000, 10)$, $R_c^{(2)} = 1$, $Q_c^{(3)} = \text{Diag}(1000, 1000, 1, 1)$ and $R_c^{(3)} = 1$. Desired closed loop specifications for the global system are described by the closed loop weighting matrices

$$Q_c = \text{Diag}\left(\mu_1 Q_c^{(1)}, \mu_2 Q_c^{(2)}, \mu_3 Q_c^{(3)}\right)$$

$$R_c = \text{Diag}\left(\mu_1 R_c^{(1)}, \mu_2 R_c^{(2)}, \mu_3 R_c^{(3)}\right)$$

and

$$S_c = Q_c$$

where μ_1, μ_2 and μ_3 are the weighting coefficients. In this example, μ_1, μ_2 and μ_3 were chosen equal to the inverse of steady state performance index of each separate subsystem controlled through a bus having an infinite bandwidth ($\mu_1 = 2.9$, $\mu_2 = 0.13$ and $\mu_3 = 0.016$).

Remark 4.1. The use of a resource-limited shared communication medium for the transmission of the control commands to the distributed actuators introduces a coupling between the three subsystems, which requires the weighting of the relative importance of each subsystem using coefficients μ_1, μ_2 and μ_3. This contrasts with the optimal control problem without communication constraints, where these constants have no impact on the optimal control of the three independent subsystems.

The length of the optimal control and communication sequences is $N = 100$. An optimal solution with an error bound of 1×10^{-5} was required. The global system is started from the initial state $x(0) = [1\ 0\ 1\ 0\ 1\ 0\ 0\ 0]^T$. Its responses are depicted in Figures 4.2 and 4.3. The optimal schedule is depicted in Figure 4.4. In this schedule, $\delta_i = 1$ means that the bus is dedicated to the transmission of control signal u_i.

The first network slots are mainly dedicated to subsystem $S^{(1)}$ until it is stabilized. It may be observed that subsystem $S^{(2)}$, which is open-loop stable and whose response time is bigger than $S^{(1)}$, needs only three time slots to be stabilized. After the stabilization of subsystems $S^{(1)}$ and $S^{(2)}$, network resources are entirely dedicated to the stabilization of subsystem $S^{(3)}$. When subsystem $S^{(3)}$ is close to the equilibrium state (from $t = 0.13$ s), then its control

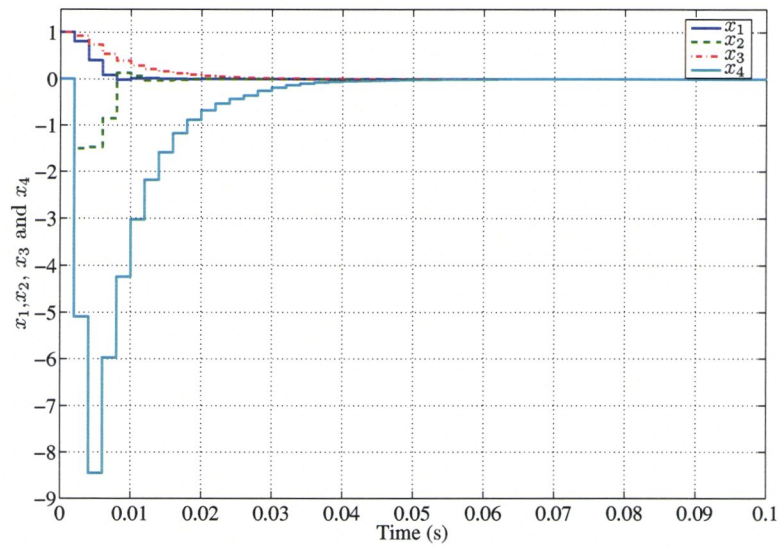


Figure 4.2: Global system response - states x_1, x_2 (subsystem $S^{(1)}$) and x_3, x_4 (subsystem $S^{(2)}$) from $t = 0\ s$ to $t = 0.1\ s$

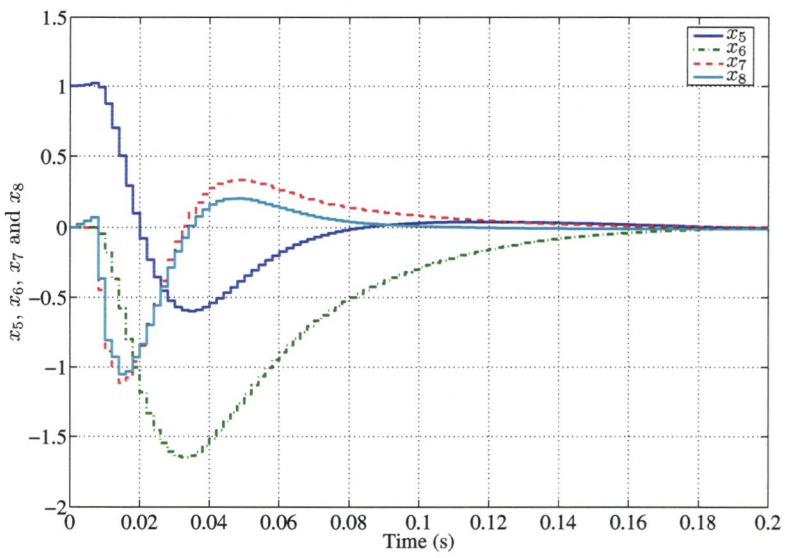


Figure 4.3: Global system response - states x_5, x_6, x_7 and x_8 (subsystem $S^{(3)}$)

signals changes are minor. Consequently, the scheduling has no significant impact on control, because control signals of the three subsystems are relatively constant, which explains the shape of the scheduling diagram, after $t = 0.13\ s$. However, this optimal schedule is dependent on the initial conditions. If the initial condition $x(0)$ is modified, then the optimal control and schedule would be different. It is clear that such an open-loop schedule, which is generated off-line, cannot be applied at run time as static schedule. In fact, assume that

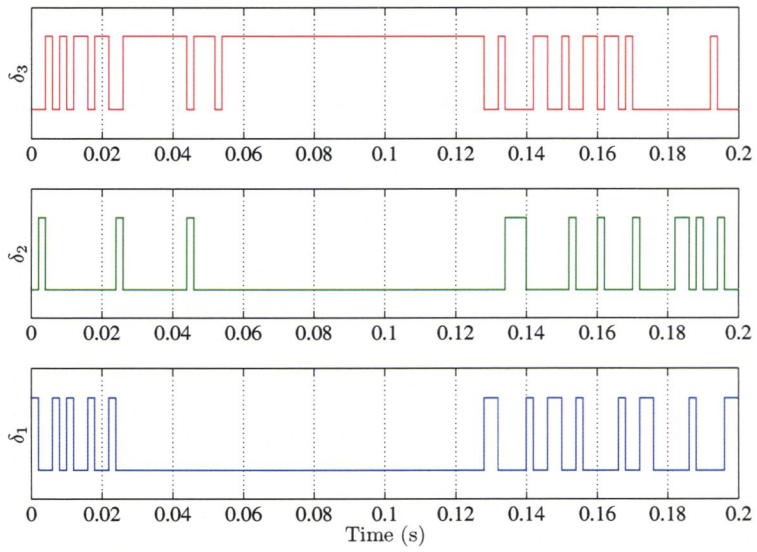


Figure 4.4: Optimal scheduling of the controller-to-actuators link

subsystem $S^{(1)}$ is disturbed at $t = 0.024\ s$. Observing the schedule, no slots are allocated to the transmission of the messages of subsystem $S^{(1)}$ between $t = 0.024\ s$ and $t = 0.128\ s$, which would induce performance degradations.

These observations show that in the same way as the optimal control, the optimal scheduling depends on the current state of the system. Consequently, fixed schedules may not be optimal if the system is started from another initial state or if it is disturbed at runtime.

4.6 Conclusion

In this chapter, we have first studied the problem of the optimal control, for a given fixed finite communication sequence. Then, we have formulated and solved the problem of the joint optimization of control and scheduling, for a given initial state. The numerical examples illustrating this method have shown that the obtained optimal schedule is extremely dependent on the chosen initial state $x(0)$.

These observations show that in the same way as the optimal control, the optimal scheduling depends on the current state of the system. However, from a computer science point of view, off-line scheduling has many advantages, essentially because it consumes few computing resources and does not induce execution overheads. In order to obtain off-line schedules that are optimal from a certain point of view, it is necessary to use performance criteria that depend on the intrinsic characteristics of the system, not on a particular initial state. This will be the objective of the next chapter.

Nevertheless, this dependency between the optimal schedule and the plant state may be seen as promising way for improving the quality of control by means of plant state-based scheduling algorithms. The design of such algorithms will be studied in Chapter 6.

5

Optimal integrated control and off-line scheduling of resource-constrained systems

5.1 Introduction

In order to formulate the joint problem of the optimal control and off-line scheduling, in addition to the modeling of the resource limitations and the representation of the system's dynamics, it is necessary to choose an adequate criterion of performance. The previous studies (illustrated in Chapter 4), which were carried out on the joint problem of the optimal control and scheduling, starting from a given initial condition, have shown that the optimal schedule is extremely dependant on the chosen initial state of the controlled dynamical system. This dependence may be exploited by the on-line scheduling algorithms in order to improve the control performance. But when only a fixed schedule is desired, it is necessary to use performance criteria that depend on the intrinsic characteristics of the system, not on a particular evolution or initial state. The use of the well-known $\mathcal{H}_2$ norm provides a solution to meet these objectives. In fact, using this performance index, the obtained off-line schedules will be independent from any initial condition. Moreover, the results may be easily transposed to an LQG context [Doyle et al., 1989]. Intuitively, a $\mathcal{H}_2$ optimal off-line schedule may be seen as a "mean square schedule", obtained when all the system' components are uniformly disturbed by a zero mean unit intensity Gaussian white noise. Off-line schedules are generally periodic. For that reason, we will focus on *T-periodic resource constrained systems*. A T-periodic resource constrained system is a resource-constrained system whose scheduling is performed according to a T-periodic communication sequence and whose control is ensured by a T-periodic linear discrete-time controller.

In this chapter, we first define the $\mathcal{H}_2$ norm of T-periodic resource-constrained systems. Based on this definition, we propose a method for the joint control and off-line scheduling in the sense of the $\mathcal{H}_2$ criterion. We show that this problem may be decomposed into two sub-problems, which may be solved separately. The first sub-problem aims at determining the optimal off-line scheduling in the sense of the $\mathcal{H}_2$ criterion and may be solved using the branch and bound method. The second sub-problem aims at determining the optimal control

gains and may be solved using the optimal periodic control theory. The proposed approach is illustrated in a numerical example and methods for the improvement of its computational complexity are proposed and discussed.

5.2 $\mathcal{H}_2$ norm of resource-constrained systems

In this section, we describe the different steps allowing the definition and the computation of the $\mathcal{H}_2$ norm of a T-periodic resource-constrained system. Taking into account the communication constraints, a T-periodic resource-constrained system may be viewed as a sampled-data model of a continuous-time LTI system controlled by a T-periodic linear discrete-time controller. In order to compute its $\mathcal{H}_2$ norm, a sampled-data model of system (3.2) is first established. The particularity of this sampled-data model is that its $\mathcal{H}_2$ norm (computed in the discrete-time domain) is identical to the $\mathcal{H}_2$ norm of the continuous-time LTI system (3.2) (computed in the continuous-time domain), when they are both controlled by a discrete-time LTI controller. Based on this sampled-data model, and on the periodicity of the scheduling and control, the $\mathcal{H}_2$ norm of a T-periodic resource-constrained system is defined. Before presenting the definition of the $\mathcal{H}_2$ norm of a T-periodic resource-constrained system, the basic definitions of the $\mathcal{H}_2$ norm of a continuous-time LTI, discrete-time LTI and sampled-data systems are presented.

5.2.1 Standard extended model definition

Consider the continuous-time LTI plant defined in equations (3.2) (Chapter 3) and the performance criterion (4.1) that was assigned to it in Section 4.2 of Chapter 4. The performances of the controlled plant may be expressed as the power of continuous-time signal $z_c(t)$. In fact, let $\breve{Q}_c$ and $\breve{R}_c$ be the matrices obtained by the Choleski decomposition of Q_c and R_c (defined in equation (4.1) in Chapter 4). Then the following relation is obtained

$$\breve{Q}_c^T \breve{Q}_c = Q_c$$

and

$$\breve{R}_c^T \breve{R}_c = R_c.$$

Let C_{1_c} and D_{12_c} be the matrices defined by

$$C_{1_c} = \begin{bmatrix} \breve{Q}_c \\ 0_{m,n} \end{bmatrix}$$

and

$$D_{12_c} = \begin{bmatrix} 0_{n,m} \\ \breve{R}_c \end{bmatrix},$$

then signal z_c may be written as

$$z_c(t) = C_{1_c} x_c(t) + D_{12_c} u_c(t).$$

Consequently

$$\int_0^{T_f} \left(x_c^T(t) Q_c x_c(t) + u_c^T(t) R_c u_c(t) \right) dt = \int_0^{T_f} z_c^T(t) z_c(t) dt.$$

An extended model that takes into account both the state and output equations (3.2) as well as the quality output z_c is given by

$$\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t) + W_c w_c(t) \quad (5.1a)$$
$$y_c(t) = C_c x_c(t) \quad (5.1b)$$
$$z_c(t) = C_{1_c} x_c(t) + D_{12_c} u_c(t). \quad (5.1c)$$

Representation (5.1) may be seen as a linear operator G_c:

$$\begin{aligned} G_c : \mathcal{L}_2^r(\mathbb{R}) \times \mathcal{L}_2^m(\mathbb{R}) &\longrightarrow \mathcal{L}_2^{n+m}(\mathbb{R}) \times \mathcal{L}_2^p(\mathbb{R}) \\ (w_c, u_c) &\longmapsto (z_c, y_c)\,. \end{aligned}$$

First, assume that G_c is controlled by a continuous-time LTI controller K_c. The controller may also be seen as a linear operator K_c defined by

$$\begin{aligned} K_c : \mathcal{L}_2^p(\mathbb{R}) &\longrightarrow \mathcal{L}_2^m(\mathbb{R}) \\ y_c &\longmapsto u_c. \end{aligned}$$

In $\mathcal{H}_2$ optimal control problems, the control system is described following the standard representation of Figure 5.1. In this representation:

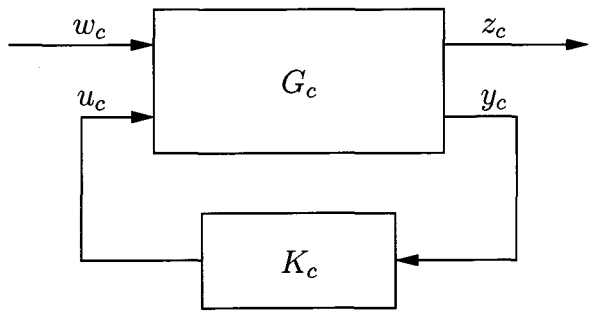


Figure 5.1: Standard representation

- w_c represents the exogenous inputs,
- u_c represents the control inputs,
- z_c represents the controlled outputs,
- y_c represents the measured outputs.

Naturally, as G_c is linear and possesses two types of inputs (w_c, u_c) and two types of outputs (z_c, y_c), it may be partitioned following:

$$z_c = G_{11_c} w_c + G_{12_c} u_c \quad (5.2a)$$
$$y_c = G_{21_c} w_c + G_{22_c} u_c. \quad (5.2b)$$

If we denote by $\hat{H}(s)$ the transfer function (in the Laplace domain) corresponding to the linear operator H, then the expressions of transfer functions of G_{11_c}, G_{12_c}, G_{21_c} and G_{22_c} are given by

$$\begin{aligned}
\hat{G}_{11_c}(s) &= C_c(sI - A_c)^{-1}B_c \\
\hat{G}_{12_c}(s) &= C_c(sI - A_c)^{-1}W_c \\
\hat{G}_{21_c}(s) &= C_{1_c}(sI - A_c)^{-1}B_c + D_{12_c} \\
\hat{G}_{22_c}(s) &= C_{1_c}(sI - A_c)^{-1}W_c.
\end{aligned}$$

5.2.2 $\mathcal{H}_2$ norm of a continuous-time LTI system

Let $T_{z_c w_c}$ be the transfer operator between the exogenous inputs w_c and the controlled outputs z_c of the closed loop system. w_c and z_c are linked by the relation

$$z_c = T_{z_c w_c} w_c.$$

The transfer function $\hat{T}_{z_c w_c}(s)$ of $T_{z_c w_c}$ is given by

$$\hat{T}_{z_c w_c}(s) = \hat{G}_{11_c}(s) + \hat{G}_{12_c}(s)\left(Is - \hat{K}_c(s)\hat{G}_{22_c}(s)\right)^{-1}\hat{K}_c(s)\hat{G}_{21_c}(s).$$

Let $(e_i)_{1\leq i\leq r}$ be the canonical basis vectors in $\mathbb{R}^r$ and δ^c the continuous-time Dirac impulse. A Dirac impulse applied to the i^{th} exogenous input of system G_c is obtained by applying an input $w_c(t) = \delta^c(t)e_i$. The produced output, assuming zero initial conditions, is $z_c(t) = T_{z_c w_c}\delta^c e_i(t)$.

Definition 5.1. The $\mathcal{H}_2$ norm of a continuous-time LTI system G_c is defined over all stabilizing continuous-time LTI controllers K_c by

$$\|G_c\|_{\mathcal{H}_2} = \left(\sum_{i=1}^{r} \|T_{z_c w_c}\delta^c e_i\|_{\mathcal{L}_2}^2\right)^{1/2}. \tag{5.3}$$

The $\mathcal{H}_2$ norm of system G_c is the quadratic mean of the $\mathcal{L}_2$ norms of the responses to r Dirac impulses, each impulse affecting only one exogenous input i, $1 \leq i \leq r$.

5.2.3 $\mathcal{H}_2$ norm of a discrete-time LTI system

In a similar way to the continuous-time case, consider the discrete-time LTI plant G controlled by a discrete-time LTI controller K. Let δ^d be the discrete-time Dirac impulse. A Dirac impulse applied to the i^{th} exogenous input of system G is obtained by applying the input $w(k) = \delta^d e_i(k)$. The produced output, assuming zero initial conditions, is $z(k) = T_{zw}\delta^d e_i(k)$.

Definition 5.2. The $\mathcal{H}_2$ norm of a discrete-time LTI system G is defined over all stabilizing discrete-time LTI controllers K by

$$\|G\|_{\mathcal{H}_2} = \left(\sum_{i=1}^{r} \left\|T_{z_c w_c}\delta^d e_i\right\|_{\mathcal{L}_2}^2\right)^{1/2}. \tag{5.4}$$

5.2.4 $\mathcal{H}_2$ norm of a sampled-data system

Figure 5.2 represents a continuous-time LTI plant G_c, which is controlled by a discrete-time LTI controller K, the discrete controller being preceded by a sampler **SPL** and followed by a zero order holder **ZOH**. In this setting, the previous definition of the $\mathcal{H}_2$ norm of a continuous-time LTI system does not make sense because the transfer operator $T_{z_c w_c}$ is no longer invariant but periodically time-varying with period T_s (which is imposed by the sampler **SPL**). In fact, the response of this sampled-data system to a Dirac impulse δ^c applied at instant $t = 0$ will not be necessarily identical to the response to a Dirac impulse δ^c_τ applied at instant $0 < \tau < T_s$ (i.e. defined by $\delta^c_\tau(t) = \delta^c(t - \tau)$). This problem was pointed out and tackled in [Chen and Francis, 1991, Khargonekar and Sivashankar, 1991, Bamieh and Boyd Pearson, 1992].

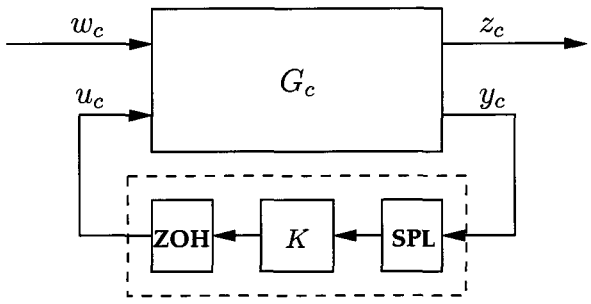


Figure 5.2: Standard representation of a sampled-data system

Definition 5.3. The generalized $\mathcal{H}_2$ norm of a sampled-data system is defined over all the stabilizing discrete-time LTI controllers K by

$$\|G_c\|_{\mathcal{H}_2} = \left(\frac{1}{T_s} \int_0^{T_s} \left(\sum_{i=1}^{r} \|T_{z_c w_c} \delta^c_\tau e_i\|^2_{\mathcal{L}_2} \right) d\tau \right)^{1/2}. \tag{5.5}$$

This generalization is based on the observation that the periodicity implies that $T_{z_c w_c}$ is completely determined by its responses to Dirac impulses $\delta^c_\tau(t) = \delta^c(t - \tau)$, applied at instants τ verifying $0 \leq \tau \leq T_s$. The $\mathcal{H}_2$ norm of sampled-data systems may be seen as the quadratic mean (in the continuous-time domain) of the previously defined $\mathcal{H}_2$ of continuous-time LTI systems corresponding to Dirac impulses applied from instant $\tau = 0$ to instant $\tau = T_s$.

5.2.5 Computation of the $\mathcal{H}_2$ norm of a sampled-data system

The computation of the $\mathcal{H}_2$ norm of a sampled-data system may be reduced to the computation of the $\mathcal{H}_2$ norm of a discrete-time LTI system. In the following, we present the method of [Khargonekar and Sivashankar, 1991], allowing computing the $\mathcal{H}_2$ norm of a sampled-data system from the $\mathcal{H}_2$ norm of an equivalent discrete-time LTI system. We will describe thereafter the different steps allowing building the state-space model of this equivalent discrete-time system.

Let S_1 and S_2 the $n \times s_1$ and $r \times s_2$ matrices such that

$$S_1 S_1^T = \frac{1}{T_s} \int_0^{T_s} e^{A_c \tau} W_c W_c^T e^{A_c^T \tau} d\tau$$

and

$$S_2 S_2^T = \frac{1}{T_s} \int_0^{T_s} \int_0^t C_{1_c} e^{A_c \tau} W_c W_c^T e^{A_c^T \tau} C_{1_c}^T d\tau dt.$$

Let $\mathcal{W}$ the $s_3 \times (n+m)$ matrix verifying

$$\mathcal{W}^T \mathcal{W} = \int_0^{T_s} \left[\begin{array}{cc} C_{1_c} e^{A_c t} & C_{1_c} \left(\int_0^t e^{A_c \tau} d\tau \right) B_c + D_{12_c} \end{array} \right]^T \left[\begin{array}{cc} C_{1_c} e^{A_c t} & C_{1_c} \left(\int_0^t e^{A_c \tau} d\tau \right) B_c + D_{12_c} \end{array} \right] dt.$$

Consider the partition of $\mathcal{W}$ such that

$$\mathcal{W} = \left[\begin{array}{cc} S_3 & S_4 \end{array} \right]$$

where S_3 and S_4 are the matrices of size $s_3 \times n$ and $s_3 \times m$. The equivalent discrete-time system may be described by the following state-space model

$$\begin{aligned} x(k+1) &= Ax(k) + B_1 w(k) + B_2 u(k) && \text{(5.6a)} \\ y(k) &= Cx(k) && \text{(5.6b)} \\ z(k) &= C_1 x(k) + D_{11} w(k) + D_{12} u(k) && \text{(5.6c)} \end{aligned}$$

where

$$\begin{aligned} A &= e^{A_c T_s} \\ B_1 &= \left[\begin{array}{cc} S_1 & 0_{n,s_2} \end{array} \right] \\ B_2 &= \int_0^{T_s} e^{A_c \tau} d\tau B_c \\ C_1 &= \left[\begin{array}{c} S_3 \\ 0_{r,n} \end{array} \right] \\ D_{11} &= \left[\begin{array}{cc} 0_{s_3,s_1} & 0_{s_3,s_2} \\ 0_{r,s_1} & S_2 \end{array} \right] \\ D_{12} &= \left[\begin{array}{c} S_4 \\ 0_{r,m} \end{array} \right]. \end{aligned}$$

5.2.6 $\mathcal{H}_2$ norm of periodically scheduled resource-constrained systems

Taking into account the communication constraints, and assuming that the full state vector is available to the controller at each sampling period and that the scheduling of the controller-to-actuators link is performed by a maximal T-periodic communication sequence, the open-loop model of a T-periodic resource-constrained system may be described by the following extended model:

$$\begin{aligned} x(k+1) &= Ax(k) + B_1 w(k) + B_2 u(k) && \text{(5.7a)} \\ z(k) &= C_1 x(k) + D_{12} u(k) + D_{22} w(k) && \text{(5.7b)} \\ \delta_i(k) = 0 &\Rightarrow u_i(k) = u_i(k-1). && \text{(5.7c)} \end{aligned}$$

Matrices A, B_1, B_2, C_1, D_{12} and D_{22} are computed using the discretization approach of the previous paragraph. We will assume, in this chapter, that there exists a maximal T-periodic

communication sequence ensuring the reachability of system (5.7). System (5.7) being T-periodic, it becomes then natural to control it using a T-periodic controller. Consequently, the global system obtained by the interconnection of the plant and the controller will be also T-periodic. In order to evaluate its $\mathcal{H}_2$ norm, the definition $\mathcal{H}_2$ norm of discrete-time periodic system proposed by [Xie et al., 2001, Zhou et al., 2002] was adopted. This definition generalizes the well known definition of the $\mathcal{H}_2$ norm for the discrete LTI systems, previously introduced.

Let δ_k^d be the discrete-time Dirac impulse applied at instant k, $0 \leq k \leq T-1$. Based on these notations, $\delta_k^d e_i$ represents the Dirac impulse applied to the i^{th} exogenous input at instant k, $1 \leq i \leq s_1 + s_2$. To simplify the notation, let $\bar{r} = s_1 + s_2$. Let z^{ik} the resulting controlled output assuming zero initial conditions.

Definition 5.4. The $\mathcal{H}_2$ norm of discrete-time linear T-periodic system G is defined over all the stabilizing discrete-time linear T-periodic controllers by

$$\|G\|_2 = \sqrt{\frac{1}{T}\sum_{k=0}^{T-1}\sum_{i=1}^{\bar{r}}\|z^{ik}\|_{l_2}^2}. \tag{5.8}$$

This definition will be used in the following in order to compute the $\mathcal{H}_2$ norm of the T-periodic resource-constrained system (5.7). This involves the computation of $\left\|z^{ik}\right\|_{l_2}$, which requires the observation of the system's response over an infinite time horizon. In this work, a very close approximation of $\left\|z^{ik}\right\|_{l_2}$ is obtained through a finite horizon $\mathcal{H}$ from the instant when the impulse is applied. For that reason, it is necessary to choose $\mathcal{H}$ greater than the response time of the system. In the practical implementation of the algorithm, it is possible to visualize the responses to the different Dirac impulses, and to evaluate how much these responses, which correspond to exponentially stable trajectories, are close to zero. The $\mathcal{H}_2$ norm of the system is the square root of the sum of the squares of the l_2 norms corresponding to these responses. Consequently, the the "finite-horizon computed $\mathcal{H}_2$ norm" $\|G\|(\mathcal{H})$ converges asymptotically to the true $\mathcal{H}_2$ norm $\|G\|(\infty)$ computed over an infinite horizon, when $\mathcal{H} \longrightarrow +\infty$. Consequently, for a desired precision, specified by the maximal absolute error $\varepsilon_{\mathcal{H}_2}$ between the true $\mathcal{H}_2$ norm $\|G\|(\infty)$ computed over an infinite horizon and the "finite-horizon computed $\mathcal{H}_2$ norm" $\|G\|(\mathcal{H})$, there will exist a horizon $\mathcal{H}_{min}$, such that, for all $\mathcal{H} \geq \mathcal{H}_{min}$, $|\|G\|(\infty) - \|G\|(\mathcal{H})| \leq \varepsilon_{\mathcal{H}_2}$. Based on this remark, and on the visualization tools which were implemented, the horizon $\mathcal{H}_{min}$ may be determined iteratively.

5.3 $\mathcal{H}_2$ optimal integrated control and off-line scheduling problem

5.3.1 Solving of the optimal scheduling subproblem

In this paragraph, the optimal scheduling subproblem in the sense of the $\mathcal{H}_2$ performance index is considered. The formulation of this problem requires first the definition of its constraints, which may be classified into two groups.

- the first group includes the constraints that intervenes in the computation of all the impulsive responses z^{ik}, $0 \leq k \leq T-1$ and $1 \leq i \leq \bar{r}$,
- the second group contains the constraints that are specific to a given impulsive response z^{ik}.

The first group includes the resource constraints (3.6)

$$\sum_{i=1}^{m} \delta_i(k) = b$$

as well as the communication sequence periodicity constraints

$$\delta(k) = \delta(k+T), \text{for } 0 \leq k \leq \mathcal{H} - T - 1. \tag{5.9}$$

In consequence, the constraints belonging to the first category may be written as

$$\mathcal{A}_s \check{\Delta} \leq \mathcal{B}_s \tag{5.10}$$

where $\check{\Delta}$ is the vector

$$\check{\Delta} = \begin{bmatrix} \delta(0) \\ \vdots \\ \delta(H-1) \end{bmatrix}.$$

The second group is related to the computation of the impulsive responses z^{ik}, for $0 \leq k \leq T-1$ and $1 \leq i \leq \bar{r}$, over the horizon $\mathcal{H}$, knowing the communication constraints previously defined. Let u^{ik}, x^{ik}, z^{ik}, ξ^{ik} and o^{ik} be respectively the values of the control input, the state, the controlled output and the auxiliary variables (introduced in Section 4.5.1 of Chapter 4) corresponding to a Dirac impulse applied at instant k to the i^{th} exogenous input of system (5.7). Let S^{ik} be the set of the involved constraints, for a given response z^{ik} (excepting the constraints belonging to the first group). S^{ik} includes:

- the constraints defined by the model (5.7) and whose translation into mixed integer quadratic programs was addressed in the previous chapter,
- the constraints defining the Dirac impulses, which must satisfy

$$w_i^{ik}(k) = 1 \tag{5.11}$$

and

$$w_j^{ik}(l) = 0 \text{ for } j \neq i \text{ or } l \neq k, \tag{5.12}$$

- the constraints that must be added to the problem to ensure the causality of the response

$$u^{ik}(l) = 0 \text{ for } l < k. \tag{5.13}$$

Let

$$\check{U}^{ik} = \begin{bmatrix} u^{ik}(0) \\ \vdots \\ u^{ik}(H-1) \end{bmatrix}$$

$$\check{X}^{ik} = \begin{bmatrix} x^{ik}(0) \\ \vdots \\ x^{ik}(H-1) \end{bmatrix}$$

$$\check{Z}^{ik} = \begin{bmatrix} z^{ik}(0) \\ \vdots \\ z^{ik}(H-1) \end{bmatrix}$$

$$\check{\Xi}^{ik} = \begin{bmatrix} \xi^{ik}(0) \\ \vdots \\ \xi^{ik}(H-1) \end{bmatrix}$$

$$\check{O}^{ik} = \begin{bmatrix} o^{ik}(0) \\ \vdots \\ o^{ik}(H-1) \end{bmatrix}$$

and

$$\mathcal{V}^{ik} = \begin{bmatrix} \check{U}^{ik} \\ \check{X}^{ik} \\ \check{Z}^{ik} \\ \check{\Xi}^{ik} \\ \check{O}^{ik} \end{bmatrix}$$

the set of constraints S^{ik} may be described by

$$\mathcal{A}^{ik} \begin{bmatrix} \check{\Delta} \\ \mathcal{V}^{ik} \end{bmatrix} \leq \mathcal{B}^{ik} \tag{5.14}$$

Consequently, the subproblem of the optimal scheduling in the sense of the $\mathcal{H}_2$ performance index may be written in the form

$$\begin{cases} \min\limits_{\check{\Delta},(\mathcal{V}^{ik})_{1\leq i\leq \bar{r},0\leq k\leq T-1}} \sum\limits_{k=0}^{T-1} \sum\limits_{i=1}^{\bar{r}} z^{ik^T} z^{ik} \\ \mathcal{A}_s \check{\Delta} \leq \mathcal{B}_s \\ \mathcal{A}^{ik} \begin{bmatrix} \check{\Delta} \\ \mathcal{V}^{ik} \end{bmatrix} \leq \mathcal{B}^{ik}, \text{ for } 1 \leq i \leq \bar{r}, 0 \leq k \leq T-1 \end{cases}.$$

Problem (5.15) is a mixed integer quadratic program and may be solved using the branch and bound method, which was described in the previous chapter. Its resolution provides the optimal T-periodic off-line communication sequence δ^{T-1^*}.

5.3.2 Solving of the optimal control subproblem

When the controller has full access to the state, and when the disturbances cannot be measured, the optimal $\mathcal{H}_2$ controller becomes identical to the optimal LQR controller [Chen and Francis, 1995]. Knowing the optimal off-line communication sequence δ^{T-1^*}, finding the optimal control boils down to the solving of the optimal control problem over an infinite horizon for a fixed communication sequence. The general solution to this problem was described in Section 4.4 of Chapter 4. The solving of this problem leads to the optimal sequence of control gains $\tilde{K}^{T-1} = \left(\tilde{K}(0), \ldots, \tilde{K}(T-1)\right)$.

5.3.3 A numerical example

Consider the collection of 2 open-loop unstable continuous-time LTI subsystems defined by

$$A_c^{(1)} = \begin{bmatrix} 0 & 26.6 \\ -250 & 2 \end{bmatrix} \qquad W_c^{(1)} = \begin{bmatrix} 1.5811 \\ 37.9473 \end{bmatrix} \quad B_c^{(1)} = \begin{bmatrix} 0 \\ 1000 \end{bmatrix} \qquad S^{(1)}$$

$$A^{(2)} = \begin{bmatrix} 0 & 0 & 0 & 532.3 \\ 0 & 0 & 243.3 & 0 \\ 0 & 0 & -272.67 & 0 \\ 434.86 & 0 & 10.184 & 0 \end{bmatrix} \quad W_c^{(2)} = \begin{bmatrix} 0 \\ 0 \\ 0.0632 \\ 0.4743 \end{bmatrix} \quad B_c^{(2)} = \begin{bmatrix} 0 \\ 0 \\ 10 \\ 10.184 \end{bmatrix} \quad S^{(2)}$$

The global system S composed by $S^{(1)}$ and $S^{(2)}$ may be described by the state matrix A_c, the exogenous input matrix W_c and the control input matrix B_c defined by

$$A_c = \text{Diag}\left(A_c^{(1)}, A_c^{(2)}\right),$$

$$W_c = \text{Diag}\left(W_c^{(1)}, W_c^{(2)}\right)$$

and

$$B_c = \text{Diag}\left(B_c^{(1)}, B_c^{(2)}\right).$$

The global system S is controlled by a discrete-time controller executed at the sampling period $T_s = 1$ ms. The control commands are sent to the actuators through a bus with that can carry at most one control command every 1 ms ($b = b_w = 1$). Design criteria of the optimal continuous time controller for each subsystem are defined by matrices $Q_c^{(1)} = \text{Diag}(8000, 10)$, $R_c^{(1)} = 1$, $Q_c^{(2)} = \text{Diag}(100, 1000, 1, 1)$ and $R_c^{(2)} = 1$. Desired closed loop specifications for the global system are described by the closed loop weighting matrices

$$Q_c = \text{Diag}\left(Q_c^{(1)}, Q_c^{(2)}\right)$$

and

$$R_c = \text{Diag}\left(R_c^{(1)}, R_c^{(2)}\right).$$

Using the approach of Section 5.2.5, the equivalent sampled-data model at the sampling period $T_s = 1$ ms was derived. This equivalent model was used in the $\mathcal{H}_2$ optimization.

The optimal solutions, corresponding to different choices of the period T are illustrated in Table 5.1. The relative optimality gap of the used branch and bound algorithm is equal to 10^{-5}, which means that the best-obtained solution will be considered as an *optimal* solution if the difference between its cost and the lower bound of the *true optimal* cost is less than 0.01%. The computations were performed on a PC equipped with a 3.6 GHz Intel Pentium processor and 1 GB of RAM. The optimization problem was solved using the solver CPLEX (Release 9.1.0) from ILOG. In this particular implementation of the optimization algorithm, $\mathcal{H}$ must be a multiple of T. It is sufficient to choose $\mathcal{H}$ greater than 27 to guarantee a maximal absolute error $\varepsilon_{\mathcal{H}_2} = 10^{-4}$.

In Table 5.1, the column *CPU Time* indicates the time needed by the optimization algorithm to finish. The algorithm finishes when it proves that the best obtained solution is *close enough* to the lower bound of the true optimal solution (i.e. the relative difference between

Table 5.1: Optimal $\mathcal{H}_2$ norm as a function of the period T - exact discretization

Period T	$\mathcal{H}$	$\mathcal{H}_2$ norm	Optimal Schedule	CPU Time (s)
2	28	10.3271	12...	20
3	30	9.0312	112...	144
4	28	9.7701	1112...	206
5	30	9.4861	11212...	343
6	30	9.0312	112112...	541
7	28	9.3481	1121121...	696
8	32	9.3176	11211212...	1746
9	27	9.0312	112112112...	1102
10	30	9.2538	1121121112...	3191

the best obtained solution and the true optimal one is less than the specified relative optimality gap). Theses results indicate that the minimal optimal schedule is of length $T = 3$. The length of the optimal schedules which gives the best $\mathcal{H}_2$ norm ($\mathcal{H}_2 = 9.0312$) is a multiple of 3. The resource allocation depends on the dynamics of the subsystems and on their sensitivity to the Dirac impulse disturbance of the $\mathcal{H}_2$ performance evaluation. It is clear from the used definition of the $\mathcal{H}_2$ norm that a circular permutation of an optimal schedule remains optimal.

When the required the CPU time in the last column of Table 5.1 is analyzed, it may be remarked that the contribution of the time needed to solve the relaxed root problem is very important. For example, the continuous relaxation of the root problem in the case $T = 2$ takes 13 s, whereas the total CPU time is 20 s. For $T = 3$, it takes 139 s whereas the total CPU time is 144 s. Since our objective is to find primarily an optimal off-line schedule, it is worth questioning whether this extreme precision in the discretization is necessary for our objective. In fact, the exact discretization causes the loss of the sparsity of the original problem. This loss of sparsity increases the time needed to perform the continuous relaxation of the obtained subproblems, which constitutes the major part of the bounding phase of the branch and bound algorithm. For example, the exact discretization of the continuous-time cost function (4.1) leads to the matrices C_1 and D_{12} such that

$$
\begin{aligned}
Q &= \begin{bmatrix} C_1^T \\ D_{12}^T \end{bmatrix} \begin{bmatrix} C_1 & D_{12} \end{bmatrix} \\
&= \begin{bmatrix}
7.9825 & 0.1050 & 0 & 0 & 0 & 0 & 0.0346 & 0 \\
0.1050 & 0.0119 & 0 & 0 & 0 & 0 & 0.0057 & 0 \\
0 & 0 & 0.1081 & 0 & 0.0001 & 0.0290 & 0 & 0.0001 \\
0 & 0 & 0 & 1.0000 & 0.1113 & 0 & 0 & 0.0004 \\
0 & 0 & 0.0001 & 0.1113 & 0.0169 & 0.0000 & 0 & 0.0001 \\
0 & 0 & 0.0290 & 0 & 0.0000 & 0.0110 & 0 & 0.0000 \\
0.0346 & 0.0057 & 0 & 0 & 0 & 0 & 0.0046 & 0 \\
0 & 0 & 0.0001 & 0.0004 & 0.0001 & 0.0000 & 0 & 0.001
\end{bmatrix}
\end{aligned}
$$

whereas the original continuous-time cost matrix is the diagonal matrix

$$\text{Diag}(Q_c, R_c) = \text{Diag}(8000, 10, 100, 1000, 1, 1, 1, 1).$$

We may remark that these matrices contain new elements with very low magnitude. The original matrix W_c is 6×2 whereas the discretized matrix B_1 is 6×6. In the following, the optimization procedure is performed considering the approximation $\hat{Q}$ of Q such that

$$\hat{Q} = T_s \ \text{Diag}(Q_c, R_c).$$

The optimization results are illustrated in Table 5.2. The obtained optimal schedules are identical to the results of the optimization using the exact discretization. The column *CPU Time* indicates the required CPU time using the default parameters of the solver. The computation time is essentially spent in order to prove that the obtained solution is optimal. In comparison to the exact discretization case, the relaxation of the root problem takes 0.25 s for $T = 2$ and 0.47 s for $T = 3$ when the approximate discretization is employed. The improvements in computation time that are due to the use of the approximate discretization are very significant and vary from 70% to 92%. In general, further improvements may be obtained if a feasible initial solution is used to prune the tree. This will be illustrated in Chapter 8. In this particular example, since the optimal solution is found quickly by the algorithm, the use of these initial solutions does not contribute to a significant improvement in computation time.

Table 5.2: Optimal $\mathcal{H}_2$ norm as a function of the period T – approximate discretization

Period T	$\mathcal{H}$	$\mathcal{H}_2$ norm	Optimal Schedule	CPU Time (s)
2	28	11.2703	12...	6
3	30	9.7650	112...	11
4	28	10.2593	1112...	18
5	30	10.3372	11212...	38
6	30	9.7650	112112...	65
7	28	9.9447	1121121...	86
8	32	10.1259	11211212...	182
9	27	9.7650	112112112...	299
10	30	9.8908	1121121112...	400

5.4 Conclusion

In this chapter, we have motivated the use of the $\mathcal{H}_2$ performance index as criterion allowing the optimal integrated control and off-line scheduling of resource-constrained systems. The $\mathcal{H}_2$ norm of a periodically off-line scheduled resource-constrained system was defined. Based on this definition, a new method for solving this problem was proposed. This method relies on the decomposition of the optimal control and off-line scheduling problem into two independent subproblems. The first subproblem aims at the finding of the optimal cyclic

schedule and is solved using the branch and bound method. The second sub-problem uses the result of the first sub-problem to determine the optimal control gains, applying the lifting technique. This method was evaluated through a numerical example and techniques allowing improving its efficiency were proposed.

6

Optimal integrated control and on-line scheduling of resource-constrained systems

6.1 Introduction

As illustrated in the considered numerical results of the optimal integrated control and scheduling problem, in Chapter 4, the optimal scheduling is closely dependent on the dynamical state of the controlled systems. This dependence confirms the intuition that the most disturbed plants have always more important "needs" in terms of communication resources than the plants that are at the equilibrium. By using on-line scheduling algorithms, it is possible to exploit this dependency to achieve a better control performance, thanks to a more efficient use of the available resources.

In this chapter, we consider a resource-constrained system $\mathcal{S}$ where the full state vector $x(k)$ is available to the controller at each sampling period. We first propose the use of the model predictive control approach as an algorithmic solution allowing computing on-line, at the same time, the optimal values of the control signals and the communication scheduling of resource-constrained systems. In opposition to [Palopoli et al., 2002a], control signals that could not be updated are held constant (and not put to zero), a quadratic cost function (and not linear function) is used to evaluate the control performance and the ability of the adaptive scheduling to improve the performance of sampled-data systems (instead of discrete-time systems) is demonstrated. However, the on-line solving of the optimization algorithm, which is required by the MPC approach, is very costly. For that reason, an on-line scheduling algorithm, called OPP is proposed. While being based on a pre-computed optimal off-line schedule, OPP makes it possible to allocate on-line the communication resources, based on the state of the controlled dynamical systems. It is shown that under mild conditions, OPP ensures the asymptotic stability of the controlled systems and enables in all the situations the improvement of the control performance compared to the basic static scheduling. Furthermore, under these conditions, the determination of OPP control and scheduling amounts to comparing a limited number of quadratic functions of the state. Finally, OPP is applied to a typical example of a distributed control system: the active suspension of car.

6.2 Model predictive control of resource-constrained systems

6.2.1 Problem formulation

Open-loop optimization problems, like those described in Chapter 4, constitute the cornerstone of a successful control method: the model predictive control (MPC). MPC has strong theoretical foundations, and many interesting properties that make it suitable to address constrained control problems. However, its main drawback is that it requires very expensive computational resources, which make it only applicable to slow systems, like chemical processes. Model predictive control is the standard approach to control MLD systems [Bemporad and Morari, 1999]. Its application to this particular problem was motivated by:

- The need to optimize simultaneously control actions and network scheduling, in order to achieve a better quality of control than the static network allocation schemes.
- The need for a control law that changes on-line the "actuation period" in order to improve the quality of control. This requires that these variations are taken into account by the control law [Martí et al., 2001].

Using MPC, an optimal control problem is solved on-line at each sampling period T_s. It aims at finding the optimal control values sequence $\hat{u}^{N-1^*} = (\hat{u}^*(0), ..., \hat{u}^*(N-1))$ and the optimal communication sequence $\hat{\delta}^{N-1^*} = (\hat{\delta}^*(0), ..., \hat{\delta}^*(N-1))$, which are solutions of the following optimization problem:

$$\begin{cases} \min\limits_{\hat{u}^{N-1}, \hat{\delta}^{N-1}} \sum\limits_{h=0}^{N-1} \begin{bmatrix} \hat{x}(h) \\ \hat{u}(h) \end{bmatrix}^T Q \begin{bmatrix} \hat{x}(h) \\ \hat{u}(h) \end{bmatrix} + \hat{x}^T(N) Q_0 \hat{x}(N) \\ \text{subject to} \\ \hat{x}(0) = x(k) \\ \hat{x}(h+1) = A\hat{x}(h) + B\hat{u}(h)\,, h \in \{0, \ldots, N-1\} \\ \sum\limits_{i=1}^{m} \hat{\delta}_i(h) = b\,, h \in \{0, \ldots, N-1\} \\ \hat{\delta}_i(0) = 0 \Longrightarrow \hat{u}_i(0) = u_i(k-1) \\ \hat{\delta}_i(h) = 0 \Longrightarrow \hat{u}_i(h) = \hat{u}_i(h-1)\,, h \in \{1, \ldots, N-1\}. \end{cases} \tag{6.1}$$

The solution of this problem is based on the prediction of the future evolution of the system over a horizon of N sampling periods. This predicted evolution is calculated according to the model of the plant, knowing the current state $x(k)$ of the system. The variables $\hat{x}(h)$, $h \in \{0, \ldots, N\}$ represent the predicted values of system states $x(k+h)$. The sequences $(\hat{u}(0), \ldots, \hat{u}(N-1))$ (virtual control sequence) and $(\hat{\delta}(0), \ldots, \hat{\delta}(N-1))$ (virtual communication sequence) are called virtual sequences, because they are based on the predicted evolution of the system. The resolution of this problem aims at finding the optimal virtual control sequence $(\hat{u}^*(0), \ldots, \hat{u}^*(N-1))$ and the optimal virtual communication sequence $(\hat{\delta}^*(0), \ldots, \hat{\delta}^*(N-1))$ that minimize a quadratic cost function over a finite horizon of N sampling periods. Assuming that the optimal virtual sequences exist, the actual control commands are obtained by setting

$$v(k) = M_{\hat{\delta}^*}(0)\hat{u}^*(0) \tag{6.2}$$

and

$$\delta(k) = \hat{\delta}^*(0) \tag{6.3}$$

and disregarding the remaining elements $(\hat{u}^*(1), \ldots, \hat{u}^*(N-1))$ and $(\hat{\delta}^*(1), \ldots, \hat{\delta}^*(N-1))$. At the next sampling period (step $k+1$), the whole optimization procedure is repeated, based on $x(k+1)$.

A important issue concerns the stability of the proposed model predictive controller. If the following constraint is added to Problem (6.1):

$$\hat{x}(N) = 0, \tag{6.4}$$

the following result is obtained.

Theorem 6.1. *If at $k = 0$, a feasible solution exists for the Problem (6.1) augmented with the additional constraint (6.4), then $\forall Q = Q^T > 0$, the MPC law (6.1)(6.4) stabilizes the system $\mathcal{S}$ such that*
$\lim_{k \to +\infty} x(k) = x_e = 0$
$\lim_{k \to +\infty} u(k) = u_e = 0.$

Proof. The proof may be easily performed following the same ideas of the proof of the sufficient stability conditions for the model predictive control of MLD systems stated in [Bemporad and Morari, 1999]. □

6.2.2 Optimality

The optimality of the model predictive controlled may be proved if an infinite horizon cost function $J(\tilde{x}, v, \delta, 0, +\infty) = J(x, u, 0, +\infty)$ is used and if the prediction horizon N is chosen infinite. At each time step k, and for any extended state $\tilde{x}(k)$, the model predictive controller over an infinite horizon computes the optimal solutions $v^*(k)$ and $\delta^*(k)$ that minimizes the cost function $J(\tilde{x}, v, \delta, k, +\infty)$, subject to the communication constraints. Its optimality directly results from the Bellman optimality principle, which states: *"An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with respect to the state resulting from the first decision."*

In many practical situations, it is sufficient to choose the prediction horizon N bigger enough than the response time of the system to get a performance that is close to the optimality. This is possible when the virtual sequences of the optimal control commands, which are computed at each sampling period, converge exponentially to zero as the horizon increases. The obtained finite horizon solution will then approximate the optimal infinite horizon solution.

6.2.3 A numerical example

In order to illustrate the proposed approach and to study the interdependency of control and scheduling, especially the relationship between the state space vector of the plant and the optimal network allocation, consider the continuous-time LTI system described by the state matrix

$$A_c = \begin{bmatrix} A_c^{(1)} & 0 \\ 0 & A_c^{(2)} \end{bmatrix}$$

and the input matrix

$$B_c = \begin{bmatrix} B_c^{(1)} & 0 \\ 0 & B_c^{(2)} \end{bmatrix}$$

with

$$A_c^{(1)} = A_c^{(2)} = \begin{bmatrix} 0 & 130 \\ -800 & 10 \end{bmatrix}$$

and

$$B_c^{(1)} = B_c^{(2)} = \begin{bmatrix} 0 \\ 230 \end{bmatrix}.$$

The global system consists of two identical and independent subsystems, $S^{(1)}$ and $S^{(2)}$, which are open-loop unstable. The two control inputs u_1 (corresponding to subsystem $S^{(1)}$) and u_2 (corresponding to subsystem $S^{(2)}$) are sent to the corresponding actuators through a shared communication bus. The bus bandwidth is such that only a control command may be transmitted each sampling period $T_s = 2$ ms, thus $b = 1$. The specifications of the optimal continuous-time controller are described by matrices $Q_c = S_c = \text{Diag}(100, 10, 100, 10)$ and $R_c = \text{Diag}(1, 1)$.

In order to take into account the bandwidth limitations, a simple strategy consists on sending alternately control commands u_1 and u_2 over the bus. This strategy consists in applying an off-line schedule, defined by the periodic communication sequence

$$\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right).$$

Subsystems $S^{(1)}$ and $S^{(2)}$ being identical, an optimal off-line schedule should fairly share the available resources between them. In order to apply this strategy, optimal sampled-data controllers (at the sampling period of 4 ms) were devised.

This static scheduling strategy is compared to an adaptive scheduling strategy that is based on the model predictive control algorithm. The MPC relies on a plant discrete-time model at the period of 2 ms. The prediction horizon N is equal to 14 (and is greater than the global system response time). A sub-optimal solution with a relative error bound of 1×10^{-5} was required for the branch and bound solver, which is used by the MPC.

Figure 6.1 compares the evolution of the state variables of subsystem $S^{(1)}$ (x_1 and x_2) and those of subsystem $S^{(2)}$ (x_3 and x_4), corresponding respectively to the application of the static strategy (SS), and to the use of the model predictive controller (MPC). The accumulated cost functions corresponding to these responses are depicted in Figure 6.2. The initial state is $[1 \;\; 0 \;\; -0.2 \;\; 0]^T$. At instant $t = 20$ ms, subsystem $S^{(1)}$ is severely disturbed.

These results show that significant improvements in control performance are achieved by the adaptive scheduling scheme, compared to the static fair network allocation. In order to understand the reasons behind these improvements, it is necessary to analyze the operation of the model predictive controller. For that, consider the controller-to-actuators schedule corresponding the use of the MPC algorithm (in Figure 6.3). Figure 6.4 describes the static off-line schedule. At $t = 0$ s, subsystem $S^{(1)}$ has the greatest deviation from the equilibrium position. In order to optimize the cost function, the MPC allocates the two first slots to the transmission of the control signal u_1, which contrasts with the static strategy, where resources are pre-allocated independently on the dynamical state of the controlled plants. Next, when the two subsystems reach approximately the same "distance" from the equilibrium, the network bandwidth is allocated fairly. At $t = 20$ ms, when subsystem $S^{(1)}$ is disturbed (subsystem $S^{(2)}$ being at the equilibrium), the model predictive controller quickly reacts by allocating 11 consecutive slots to the transmission of the control command u_1. This contrasts with the operation of the static strategy, where half of the bandwidth is allocated to

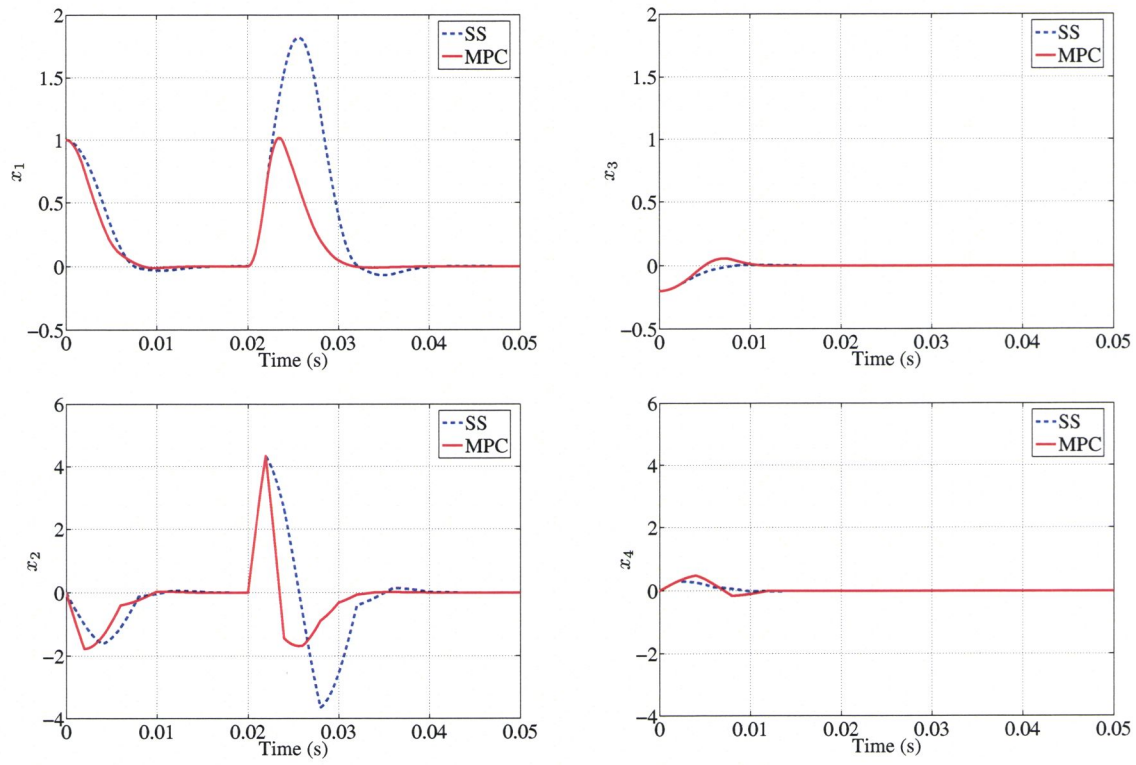


Figure 6.1: Subsystems $S^{(1)}$ and $S^{(2)}$ responses

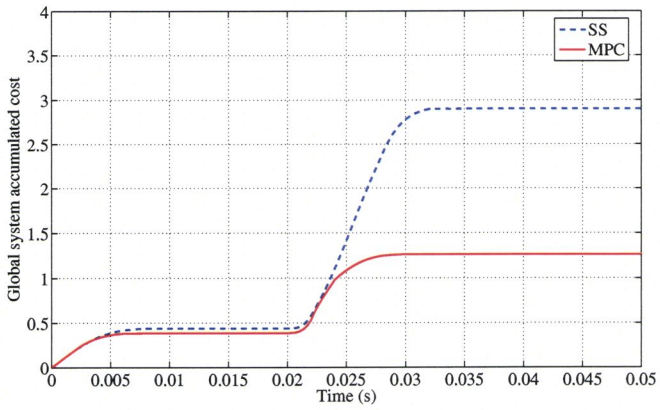


Figure 6.2: Accumulated continuous-time cost functions

subsystem $S^{(2)}$, which is at the equilibrium. This aptitude of the model predictive controller to change the "actuation period" comes from its capacity to compensate the control commands for these changes. This dynamic allocation of the resources (as well as the automatic compensation of the control commands) make it possible to reject disturbances in a much

more powerful way, contributing to the significant improvement of the quality of control (as illustrated to the Figure 6.2).

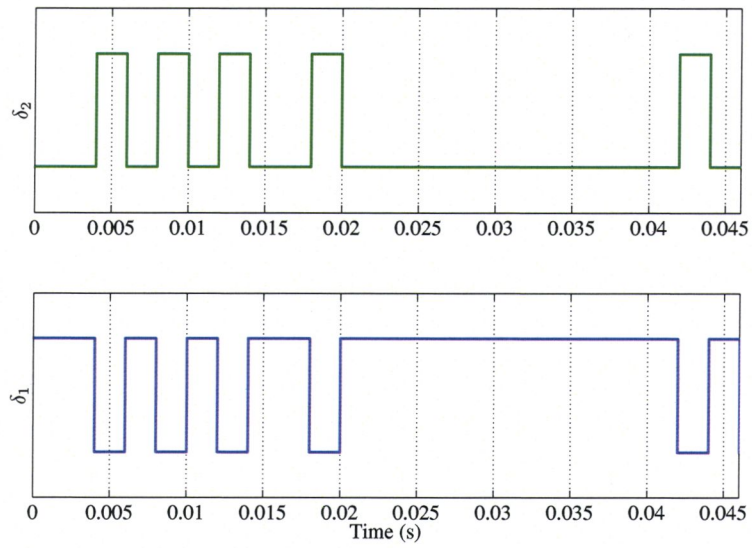


Figure 6.3: Adaptive schedule (MPC)

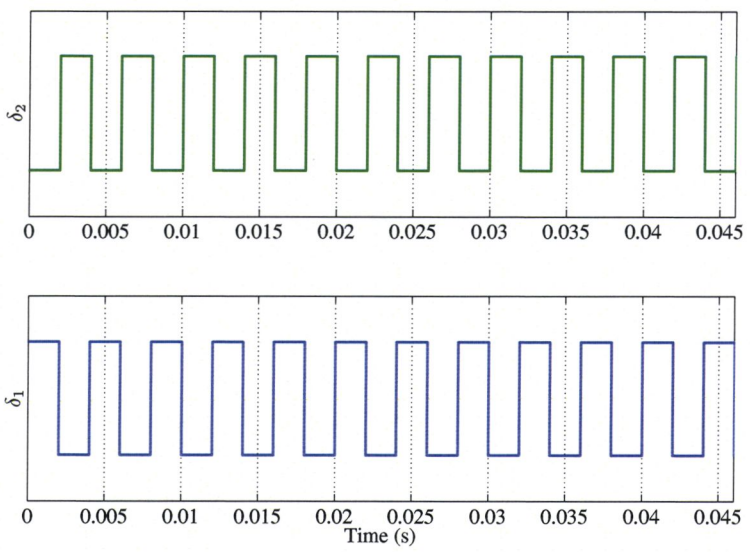


Figure 6.4: Static schedule (SS)

These results illustrate the abilities of the model predictive controller to play the role of an adaptive controller-scheduler, as well as its aptitudes to improve the quality of control compared to any off-line scheduling strategy. These improvements manifest themselves by better disturbance rejection capabilities, which are due to the ability of the adaptive scheduling

scheme to react earlier and quicker to the disturbances, compared to any off-line scheduling strategy. These results are is accordance with those of [Martí et al., 2002], where the empirical study of the relationship between resource allocation and control performance was undertaken.

6.3 Optimal pointer placement scheduling

6.3.1 Problem formulation

The motivation behind the optimal pointer placement (OPP) scheduling algorithm presented in this section is to be a compromise between the advantages of the on-line scheduling (control performance) and those of the off-line scheduling (a very limited usage of computing resources).

Consider a static off-line control and scheduling strategy, which is based on:

- a T-periodic controller defined by a sequence of state-feedback control gains $\tilde{K}^{T-1} = (\tilde{K}(0), \ldots, \tilde{K}(T-1))$,
- a static off-line schedule defined by a T-periodic communication sequence $\gamma^{T-1} = (\gamma(0), \ldots, \gamma(T-1))$.

Note that the optimal off-line selection of $\tilde{K}^{T-1}$ and γ^{T-1} was described in the previous chapter.

At runtime, the execution of the periodic off-line controller and scheduler may be described using the notion of pointer. The pointer may be seen as a variable that contains the index of the control gain to use and the scheduling to apply. The pointer is incremented at each sampling period. If it reaches the end of the sequence, its position is reset. More formally, if the pointer is started at position p ($0 \leq p < T$) its expression $\mathcal{I}_p(k)$ at the k^{th} sampling period is

$$\mathcal{I}_p(k) = (k+p) \mod T \tag{6.5}$$

According to the off-line strategy, the control commands that are transmitted as well as the scheduling decisions are chosen such that

$$v(k) = \tilde{K}(\mathcal{I}_p(k))\tilde{x}(k) \tag{6.6}$$

and

$$\delta(k) = \gamma(\mathcal{I}_p(k)). \tag{6.7}$$

***Example* 6.1.** Consider the periodic communication sequence γ^5 defined by

$$\gamma^5 = \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right)$$

and the associated control gains sequence

$$\tilde{K}^5 = (\tilde{K}(0), \tilde{K}(1), \tilde{K}(2), \tilde{K}(3), \tilde{K}(4), \tilde{K}(5))$$

Figure 6.5 graphically illustrated the notion of pointer. The communication sequence γ^5 is depicted as well as the pointer, which is located at position $p = 3$.

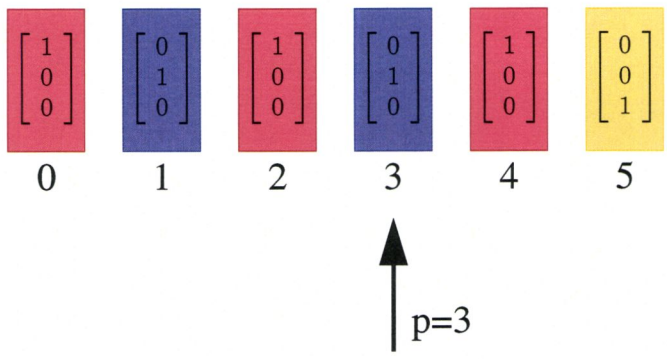


Figure 6.5: Illustration of the notion of pointer

The idea behind the OPP scheduling heuristic is that instead of finding an optimal solution to Problem (6.1), the search is restricted to the finding of a sub-optimal solution, based on an optimal off-line schedule, over a horizon N (which is assumed to be a multiple of T), according to the following problem:

$$\begin{cases} \min\limits_{p} \sum\limits_{h=0}^{N-1} \begin{bmatrix} \hat{x}(h) \\ \hat{u}(h) \end{bmatrix}^T Q \begin{bmatrix} \hat{x}(h) \\ \hat{u}(h) \end{bmatrix} + \hat{x}^T(N) Q_0 \hat{x}(N) \\ \text{subject to} \\ \hat{x}(0) = x(k) \\ \hat{u}(-1) = u(k-1) \\ \text{and for all } h \in \{0, \ldots, N-1\} \\ \hat{v}(h) = \tilde{K}(\mathcal{I}_p(h)) \begin{bmatrix} \hat{x}(h) \\ \hat{u}(h-1) \end{bmatrix} \\ \hat{u}(h) = D_\gamma(\mathcal{I}_p(h))\hat{v}(h) + E_\gamma(\mathcal{I}_p(h))\hat{u}(h-1) \\ \hat{x}(h+1) = A\hat{x}(h) + B\hat{u}(h). \end{cases} \tag{6.8}$$

The cost function is computed according to a prediction of the future evolution of the system (described by $\hat{x}(h)$). This evolution is calculated assuming that sequences $\tilde{K}^{T-1}$ and γ^{T-1} are started from position p. For example, if the pointer is located at position $p = 3$, as in Figure 6.5, the evolution of the system is predicted based on the communication sequence

$$(\gamma(3), \gamma(4), \gamma(5), \gamma(0), \gamma(1), \gamma(2), \gamma(3), \ldots)$$

and the control gains sequence

$$\left(\tilde{K}(3), \tilde{K}(4), \tilde{K}(5), \tilde{K}(0), \tilde{K}(1), \tilde{K}(2), \tilde{K}(3), \ldots\right).$$

The solution of Problem (6.8) is the pointer's position p^* that minimizes the cost function $\hat{J}(\tilde{x}(k), p)$, subject to the constraints expressed above, where

$$\hat{J}(\tilde{x}(k), p) = \sum_{h=0}^{N-1} \begin{bmatrix} \hat{x}(h) \\ \hat{u}(h) \end{bmatrix}^T Q \begin{bmatrix} \hat{x}(h) \\ \hat{u}(h) \end{bmatrix} + \hat{x}^T(N) Q_0 \hat{x}(N).$$

The control command $v(k) = \tilde{K}(p^*)\tilde{x}(k)$ is sent according to the scheduling vector $\delta(k) = \gamma(p^*)$. At the next sampling period, this procedure is reiterated in a receding horizon manner.

6.3.2 A numerical example

In order to illustrate the effectiveness of the OPP scheduling algorithm, the example of Section 4.5.3 of Chapter 4 is reconsidered. The control performance corresponding to the use of a static scheduling (SS) algorithm, OPP and MPC is compared. Static scheduling and OPP algorithms use the communication sequence

$$\gamma^5 = \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right)$$

which together with the control gains sequence $\tilde{K}^5 = (\tilde{K}_1, \tilde{K}_2, \tilde{K}_1, \tilde{K}_2, \tilde{K}_1, \tilde{K}_3)$ guarantees the asymptotic stability of the global system, where

$$\tilde{K}_1 = [-1.14 \ \ -1.04 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$$

$$\tilde{K}_2 = [0 \ 0 \ 4.12 \ 0.36 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$$

$$\tilde{K}_3 = [0 \ 0 \ 0 \ 0 \ 17.80 \ \ -6.28 \ \ -33.28 \ 48.91 \ 0 \ 0 \ 0] .$$

The period T of the schedule is equal to 6 and the horizon N of the OPP and MPC algorithms is equal to 60. A sub-optimal solution with a relative error of 1×10^{-5} was required for the MPC algorithm. Global system responses corresponding to state variables x_1, x_3, x_5 and x_6 (the positions) are depicted in Figure 6.6. The accumulated continuous-time cost functions corresponding to these responses are illustrated in Figure 6.7.

The global system is started from the initial state $[1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0]^T$. The three subsystems $S^{(1)}$, $S^{(2)}$ and $S^{(3)}$ converge progressively to the steady state. At $t = 0.14$ s, the subsystem $S^{(1)}$ is disturbed. This deviation is quickly corrected. The best response is achieved by the MPC. However, this algorithm cannot be implemented in practice, because the required computation time is too long (a few seconds at each step in this example, on a PC equipped with an Intel Celeron processor cadenced at 2.2 GHz). For $N = 60$, the number of admissible maximal communication sequences is $3^{60} = 4.23 \times 10^{28}$. The fact that the used branch and bound algorithm finishes in few seconds shows its efficiency to address this particular problem. The OPP algorithm significantly improves the control performance with respect to the static scheduling algorithm, requiring fewer computing resources than the MPC. In fact, using the OPP scheduling algorithm, the maximum number of possible communication sequences is equal to $T = 6$.

The schedule obtained by the OPP and MPC algorithms are respectively depicted in Figures 6.8 and 6.9. At $t = 0$ s, the OPP scheduling algorithm chooses to reserve the first slot to subsystem $S^{(3)}$. This choice contributes to the improvement of its performance and that of the global system (shown in Figure 6.6). MPC and OPP have the ability to change on-line the "actuation period" of each subsystem. MPC may compensate for these changes. Consequently, the scheduling pattern is irregular. Network slots are allocated in order to improve the control performance. When a subsystem is closer to the equilibrium, then its control commands remains constant and they may be sent less frequently over the network. That's why the MPC algorithm does not allocate network slots to subsystems in the equilibrium, when other subsystems are "far" form the steady state. At $t = 0.14$ s, when the subsystem $S^{(1)}$ is disturbed, OPP and MPC algorithms allocates the network slots mainly for the transmission of the control command of subsystem $S^{(1)}$, allowing to react earlier and quicker to the

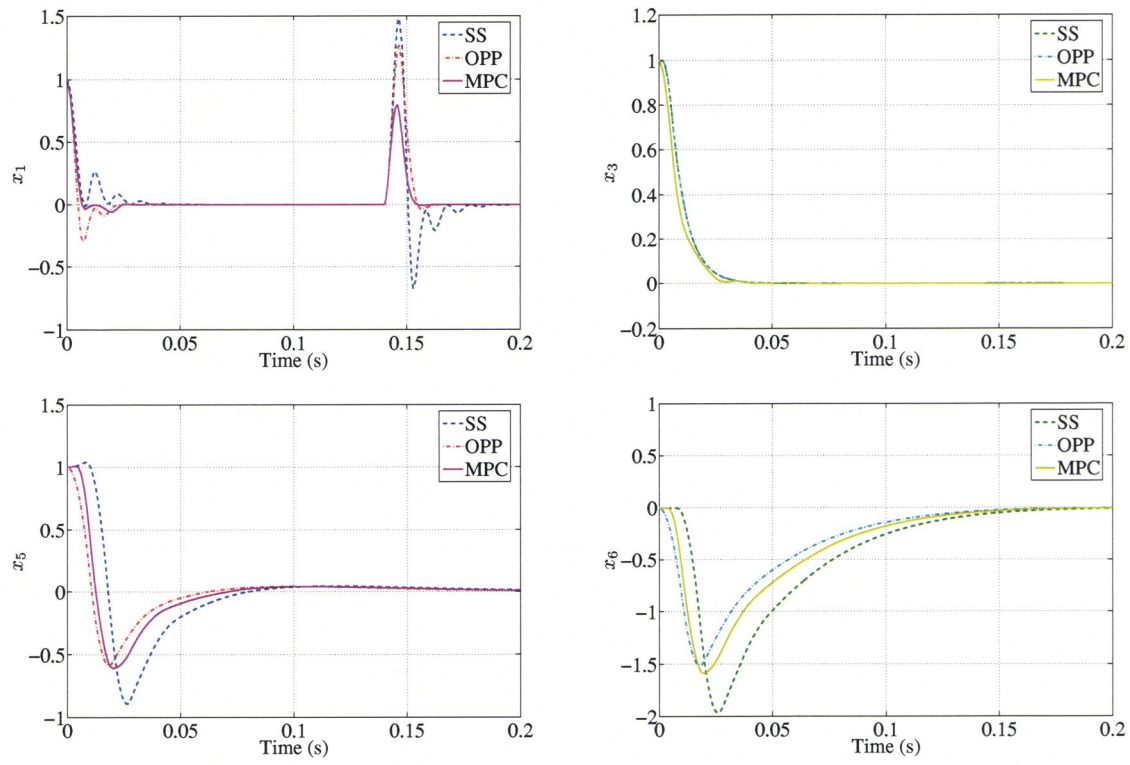


Figure 6.6: Global system responses - states x_1 and x_3

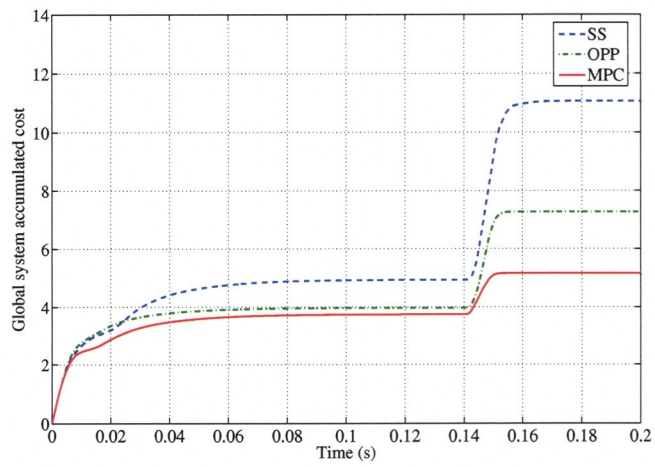


Figure 6.7: Accumulated continuous-time cost functions

disturbance. This contrasts with the static scheduling algorithm, where the allocation of the network resources is independent from the dynamical state of the subsystems.

Finally, the three subsystems are disturbed with a band limited white noise characterized by a noise power of 0.1 and a correlation time of 1×10^{-5}. Simulation results are depicted in

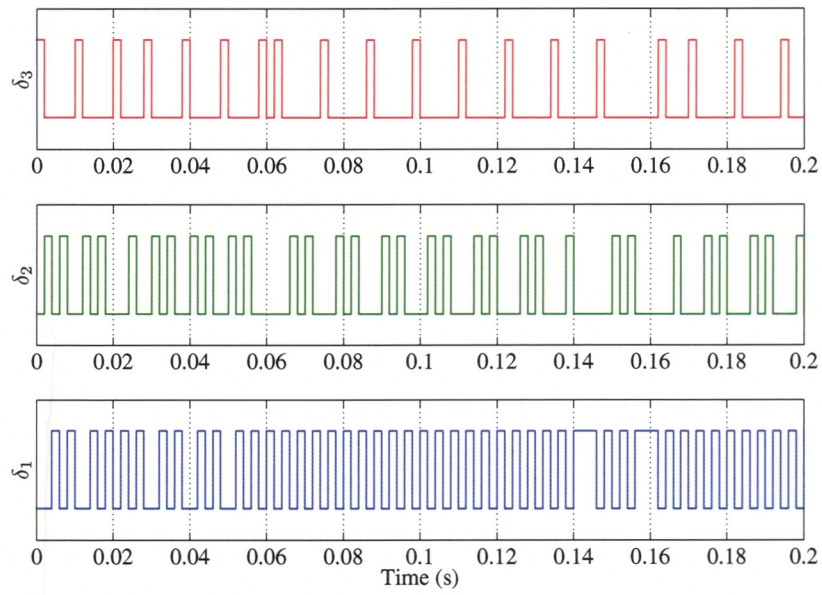


Figure 6.8: OPP schedule

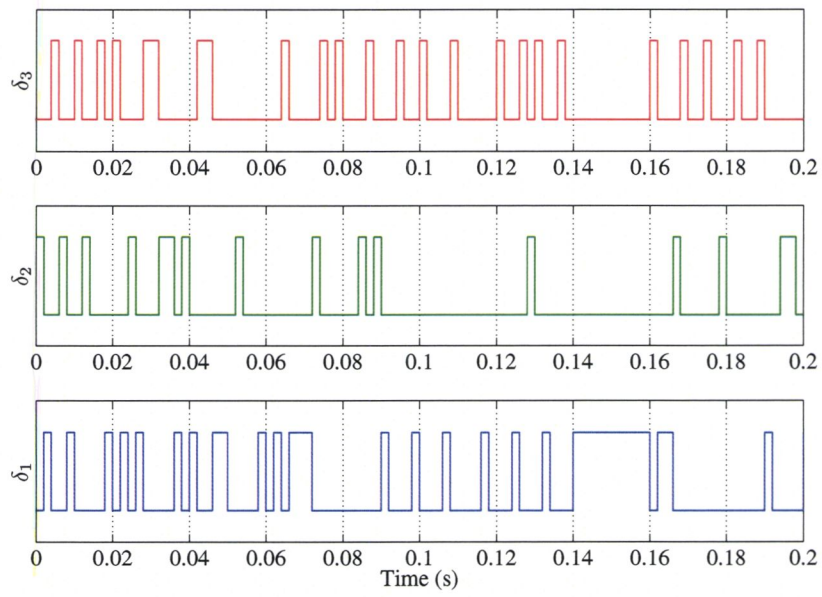


Figure 6.9: MPC schedule

Figure 6.10. Performance improvements using the OPP and the MPC algorithms are similar to those observed in the previous simulations.

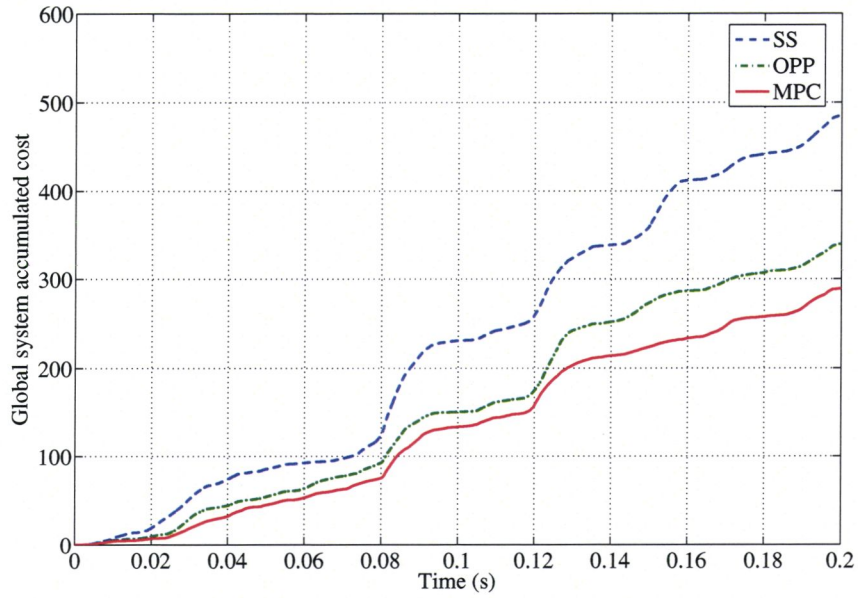


Figure 6.10: Accumulated continuous-time cost functions resulting from a band limited white noise disturbance

6.3.3 Optimal pointer placement over infinite horizon

If the horizon N is infinite, the OPP scheduling algorithm presents interesting properties : its implementation becomes simpler and a formal proof of its stability may be given. Moreover, it may also be proven that the quality of control obtained using OPP is always better or similar (in the worst-case) compared to the quality of control obtained using its basic static scheduling algorithm.

The simplification of OPP comes from the relation between the expression of the cost function over an infinite horizon (corresponding the use of a T-periodic communication sequence) and the solution of the discrete algebraic periodic Riccati equation over an infinite horizon described in Section 4.4 of Chapter 4. This relation is formalized by

$$\hat{J}(\tilde{x}(k), p) = \tilde{x}^T(k)\tilde{S}(p)\tilde{x}(k).$$

Based on this relation, the implementation of OPP becomes simpler and may be described by the following algorithm:

$$\text{Find } p^* = \underset{p}{\operatorname{argmin}}\, \tilde{x}^T(k)\tilde{S}(p)\tilde{x}(k)$$
$$v(k) = \tilde{K}(p^*)\tilde{x}(k) \text{ and } \delta(k) = \gamma(p^*).$$

For $N = +\infty$, the stability of the OPP scheduling algorithm is stated in the following theorem.

Theorem 6.2. *If the asymptotic stability of the resource-constrained system $\mathcal{S}$ is guarantied by the off-line control and scheduling using the control gains sequence $\tilde{K}^{T-1}$ and the network scheduling sequence γ^{T-1}, then it is also ensured by the OPP scheduling algorithm.*

Proof. The proof is based on the comparison of the trajectory of the system scheduled by the static scheduling algorithm (denoted by $\tilde{x}^{ss}$) and that of the system scheduled by the OPP algorithm (denoted by $\tilde{x}^{opp}$), starting from the same arbitrary initial condition $\tilde{x}^{ss}(0) = \tilde{x}^{opp}(0) = \tilde{x}(0)$.
Let $J^{ss}(\tilde{x}(i), i, f, p)$ be the cost function corresponding to an evolution from instant $k = i$ to instant $k = f$ starting from the state $\tilde{x}(i)$ where the static scheduling algorithm is applied and where the pointer at instant i is placed at position p:

$$J^{ss}(\tilde{x}(i), i, f, p) = \sum_{k=i}^{f} \begin{bmatrix} x^{ss}(k) \\ u^{ss}(k) \end{bmatrix}^T Q \begin{bmatrix} x^{ss}(k) \\ u^{ss}(k) \end{bmatrix}.$$

Let $J^{opp}(\tilde{x}(i), i, f)$ be the cost function corresponding to an evolution from instant $k = i$ to instant $k = f$ starting from the state $\tilde{x}(i)$ where the OPP algorithm is applied:

$$J^{opp}(\tilde{x}(i), i, f) = \sum_{k=i}^{f} \begin{bmatrix} x^{opp}(k) \\ u^{opp}(k) \end{bmatrix}^T Q \begin{bmatrix} x^{opp}(k) \\ u^{opp}(k) \end{bmatrix}.$$

In order to prove the stability of the system scheduled using the OPP algorithm, we must prove that for all $p_0 \in \{0, \cdots, T-1\}$

$$J^{opp}(\tilde{x}(0), 0, +\infty) \leq J^{ss}(\tilde{x}(0), 0, +\infty, p_0). \tag{6.9}$$

In fact, Q is definite positive. As a consequence, $J^{ss}(\tilde{x}(0), 0, +\infty, p_0)$ (resp. $J^{opp}(\tilde{x}(0), 0, +\infty)$) is finite if and only if the system scheduled using the static scheduling algorithm (resp. the OPP algorithm) is asymptotically stable.
Let $J^{opp-ss}(\tilde{x}(0), l)$ be the cost function corresponding to an evolution starting from the initial state $\tilde{x}(0)$ where OPP is applied form $k = 0$ to $k = l$ and then followed by the static scheduling algorithm (which is applied from $k = l+1$ to $k = +\infty$). Then it is easy to see that

$$J^{opp}(\tilde{x}(0), 0, +\infty) = \lim_{l \to +\infty} J^{opp-ss}(\tilde{x}(0), l). \tag{6.10}$$

Consequently, to proove (6.9), it is sufficient to verify that for all $p_0 \in \{0, \cdots, T-1\}$ and for all $l \in [0, +\infty)$

$$J^{opp-ss}(\tilde{x}(0), l) \leq J^{ss}(\tilde{x}(0), 0, +\infty, p_0). \tag{6.11}$$

This proof may be done by recurrence on l. Let $p_0 \in \{0, \cdots, T-1\}$ an arbitrary start position of the static scheduling algorithm.
At the stage $l = 0$, the OPP scheduling algorithm will choose the pointer position such that

$$p^*(0) = \underset{p}{\operatorname{argmin}}\, J^{ss}(\tilde{x}(0), 0, +\infty, p). \tag{6.12}$$

Knowing that

$$J^{ss}(\tilde{x}(0), 0, +\infty, p^*(0)) = J^{opp-ss}(\tilde{x}(0), 0)$$

implies that

$$J^{opp-ss}(\tilde{x}(0), 0) \leq J^{ss}(\tilde{x}(0), 0, +\infty, p_0).$$

The property is then valid at stage 0.
Assume that (6.11) is valid at stage $l-1$ with $l > 0$. We have to prove that (6.11) is also valid

at stage l.
At stage l, according to the OPP strategy, the pointer position $p^*(l)$ will be chosen such that

$$p^*(l) = \underset{p}{\operatorname{argmin}}\, J^{ss}(\tilde{x}^{opp}(l), l, +\infty, p). \tag{6.13}$$

Consequently

$$J^{ss}(\tilde{x}^{opp}(l), l, +\infty, p^*(l)) \leq J^{ss}(\tilde{x}^{opp}(l), l, +\infty, I_{p^*(l-1)}(1)). \tag{6.14}$$

where $p^*(l-1)$ is the optimal pointer position at instant $l-1$. Adding $J^{opp}(\tilde{x}(0), 0, l-1)$ to both left and right terms of the previous inequality, we get

$$J^{opp-ss}(\tilde{x}(0), l) \leq J^{opp-ss}(\tilde{x}(0), l-1). \tag{6.15}$$

Using the recurrence assumption

$$J^{opp-ss}(\tilde{x}(0), l-1) \leq J^{ss}(\tilde{x}(0), 0, +\infty, p_0), \tag{6.16}$$

and the inequality (6.15), the theorem is proved. □

6.4 Optimal pointer placement scheduling: application to a car suspension system

In this section, the OPP scheduling algorithm is applied to a distributed active suspension controller. The considered controller is based on a full-vehicle model and is implemented on a central processor. The controller sends the control commands to four hydraulic actuators located on the vehicle's corners through a bus subject to bandwidth limitations. In the following, the considered active suspension model is described, the control design methodology is illustrated and finally the OPP scheduling strategy is evaluated and compared to a fair static scheduling strategy.

6.4.1 The suspension control system

The simulated model (Figure 6.11) was adopted from [Chalasani, 1986]. It consists of a seven degree-of-freedom system. In this model, the car body, or sprung mass, is free to heave, roll and pitch. In order to obtain a linear model, roll and pitch angles are assumed to be small. The suspension system connects the sprung mass to the four unsprung masses (front-left, front-right, rear-left and rear-right wheels), which are free to bounce vertically with respect to the sprung mass. The suspension system consists of a spring, a shock absorber and a hydraulic actuator at each corner. The shock absorbers are modeled as linear viscous dampers, and the tires are modeled as linear springs in parallel to linear dampers.

In order to describe this system, fifteen variables need to be considered:

x_{c_1} : heave velocity of the center of gravity of the sprung mass
x_{c_2} : pitch angular velocity of the sprung mass
x_{c_3} : roll angular velocity of the sprung mass
x_{c_4} : front-left suspension deflection
x_{c_5} : rear-left suspension deflection

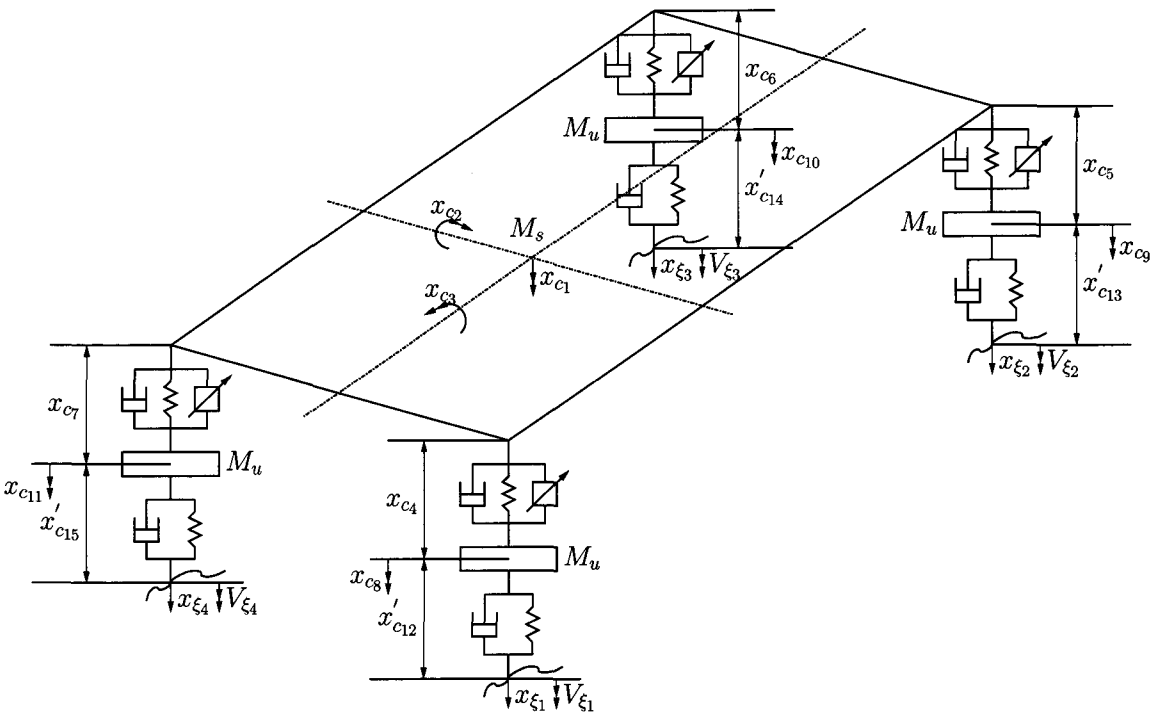


Figure 6.11: Full vehicle model

x_{c6} : rear-right suspension deflection
x_{c7} : front-right suspension deflection
x_{c8} : front-left unsprung mass velocity
x_{c9} : rear-left unsprung mass velocity
x_{c10} : rear-right unsprung mass velocity
x_{c11} : front-right unsprung mass velocity
x'_{c12} : front-left tire deflection
x'_{c13} : rear-left tire deflection
x'_{c14} : rear-right tire deflection
x'_{c15} : front-right tire deflection

Road disturbances acting on the four vehicle wheels consist of height displacement inputs $(x_{\xi_1}, x_{\xi_2}, x_{\xi_3}, x_{\xi_4})$ and height velocity inputs $(V_{\xi_1}, V_{\xi_2}, V_{\xi_3}, V_{\xi_4})$ defined with respect to an inertial reference frame.

The suspension model has seven degrees of freedom. Consequently, only fourteen state variables are needed to describe it. The extra variable may be eliminated if the wheel deflections are expressed as a function of three state variables $x_{c_{12}}$, $x_{c_{13}}$ and $x_{c_{14}}$ and of the road disturbances x_{ξ_1}, x_{ξ_4}, x_{ξ_3}, x_{ξ_4} as illustrated in [Chalasani, 1986].

Applying a force-balance analysis to the model in Figure 6.11, the state equation may be derived from the equations of motion and is given by

$$\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t) + W_c w_c(t) \tag{6.17}$$

where

$x_c(t)$ is the state vector (14 variables),

$u_c(t)$ is the control vector: $u_c(t) = [u_{c_1}(t)\ u_{c_2}(t)\ u_{c_3}(t)\ u_{c_4}(t)]^T$, u_{c_1}, u_{c_2}, u_{c_3} and u_{c_4} represent the control forces applied respectively by the front-left, rear-left, rear-right and front-right hydraulic actuators,

$w_c(t)$ is the vector of road disturbances (displacements and velocities) :
$w_c(t) = [x_{\xi_1}(t)\ x_{\xi_2}(t)\ x_{\xi_3}(t)\ x_{\xi_4}(t)\ V_{\xi_1}(t)\ V_{\xi_2}(t)\ V_{\xi_3}(t)\ V_{\xi_4}(t)]^T$.

6.4.2 Active suspension control law

The control design for a vehicle's active suspension aims to maximize the driving comfort (as measured by sprung mass accelerations) and the safety (as measured by tire load variations) under packaging constraints (as measured by suspension deflections). However, comfort and safety are two conflicting criteria [Rettig and von Stryk, 2001]. We adopt the control design methodology of [Chalasani, 1986], who divides the control design problem for a vehicle's active suspension into two sub-problems:

- The design of the ride controller, whose role is to improve ride comfort by isolating the sprung mass from road disturbances. The ride controller has also to maintain a sufficient contact force between the tires and the road to insure convenient road holding.
- The design of the attitude controller, responsible of maintaining load-leveling, performing convenient load distribution and controlling roll and pitch angles during vehicle maneuvers (roll during cornering and pitch during breaking and acceleration).

In this section, the ride controller part of the suspension controller is considered. The used ride controller is a linear quadratic regulator which aims at minimizing the following cost function:

$$J_c = \int_0^{+\infty} \left\{ \sum_{i=0}^{12} \mu_i y_i^2(t) + \sum_{j=0}^{4} a_j u_{c_j}^2(t) \right\} dt, \tag{6.18}$$

where μ_i and a_j are weighting factors and

y_1 : vertical acceleration of the sprung mass
y_2 : pitch angular acceleration of the sprung mass
y_3 : roll angular acceleration of the sprung mass
y_4 : sum of the suspension deflections at the four corners
y_5 : difference between the suspension deflections at the right- and left- hand sides
y_6 : difference between the suspension deflections at the front and rear of the vehicle
y_7 : difference between the suspension deflections at the diagonally opposite corners
y_8 : sum of the velocities of the unsprung masses
y_9 : difference between the velocities of the unsprung masses on the left- and right-hand sides
y_{10} : difference between the velocities of the unsprung masses at the front and rear of the vehicle
y_{11} : difference between the velocities of the unsprung masses at the diagonally opposite corners
y_{12} : wrap torque acting on the sprung mass.

6.4.3 Simulation setup and results

The communication network connecting the controller to the actuators is subject to communication constraints: only a control command can be sent to an actuator every 10 ms. A

simple approach to tackle this problem is to send the control commands alternately according to the periodic communication sequence

$$\gamma^3 = \left(\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right).$$

Using this communication sequence, the discretized model of the controlled car suspension system becomes periodic. Applying the methodology described in Chapter 4, Section 4.4, the optimal periodic control gains $\left(\tilde{K}(0), \tilde{K}(1), \tilde{K}(2), \tilde{K}(3)\right)$ may be derived.

The suspension system is evaluated by subjecting the left side of the vehicle to a "chuck hole" discrete road disturbance [Chalasani, 1986] (Figure 6.12). The vehicle speed is equal to 40 km/h. First, the performance of the designed active suspension controller is evaluated

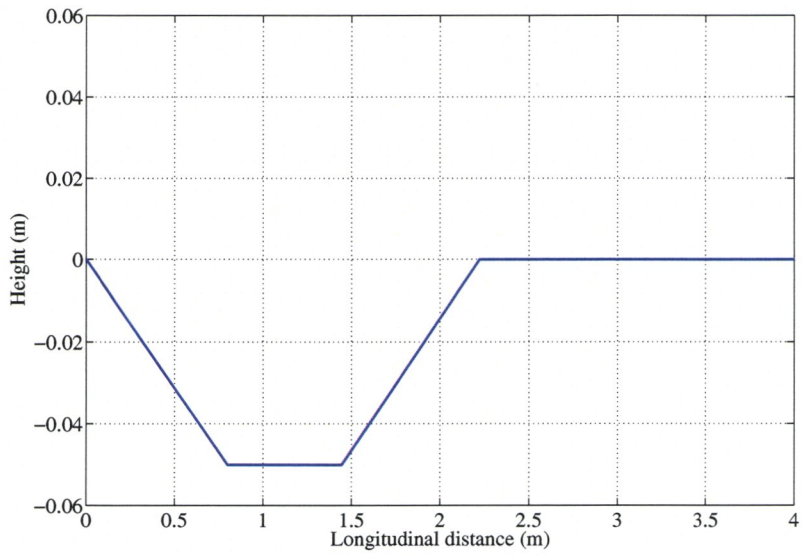


Figure 6.12: "Chuck hole" road disturbance

and compared to the passive suspension. Then, the performance of the OPP algorithm is compared to that obtained by the application of the static scheduling (SS) algorithm. OPP and the static scheduling algorithm are both based on the communication sequence and control gains described above. Heave, roll and pitch velocity responses are illustrated in Figures 6.13 and 6.14.

From these simulation results, it may be seen that the active suspension induces an important improvement of the ride performance compared to the passive suspension (smaller and better damped velocities and thus accelerations). The responses using the OPP algorithm show a slight improvement with respect to the static scheduling (SS) algorithm. The improvements in ride comfort shown by the active suspension are obtained with suspension and tire deflection levels which are close to those obtained with the passive suspension (the rear-right suspension and tire deflections are depicted in Figure 6.15).

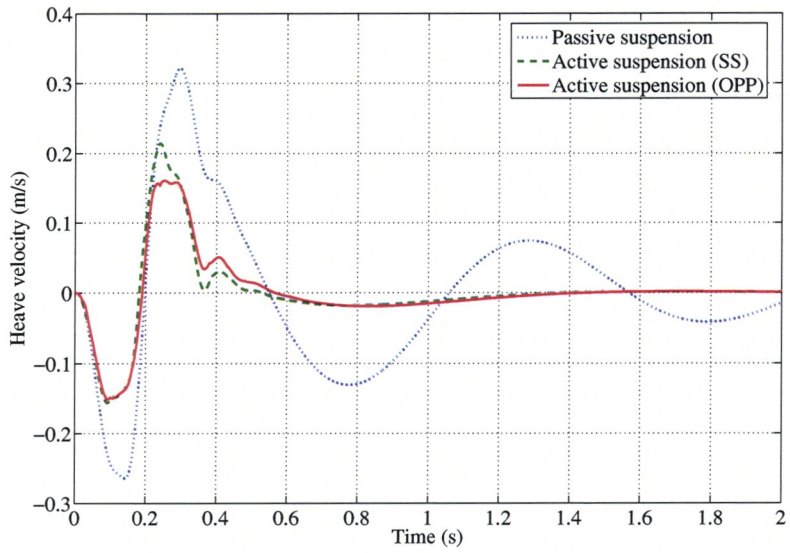


Figure 6.13: Heave velocity of the passive and active suspension (controlled with the static scheduling and the OPP algorithms)

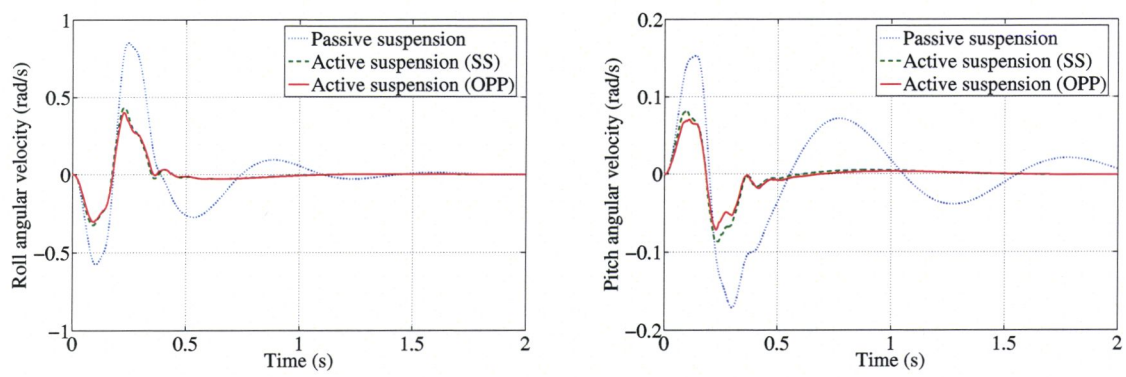


Figure 6.14: Roll and pitch velocities of the passive and active suspension (controlled with the static scheduling and the OPP algorithms)

Finally, the quadratic cost functions corresponding to the ideal continuous time LQR controller, the static scheduling algorithm and the OPP scheduling algorithm are compared in Figure 6.16. The steady state cost function values corresponding to the static scheduling, OPP and to the ideal implementation are respectively equal to 4459, 4122 and 3290. Consequently, the improvements in terms of quality of control that were achieved by the OPP algorithm are equal to 28.8%. These improvements are significant, but not as "spectacular" as those observed in the previous example because the different components of the suspension system are tightly coupled: a disturbance affecting a single wheel influences all the state variables of the system.

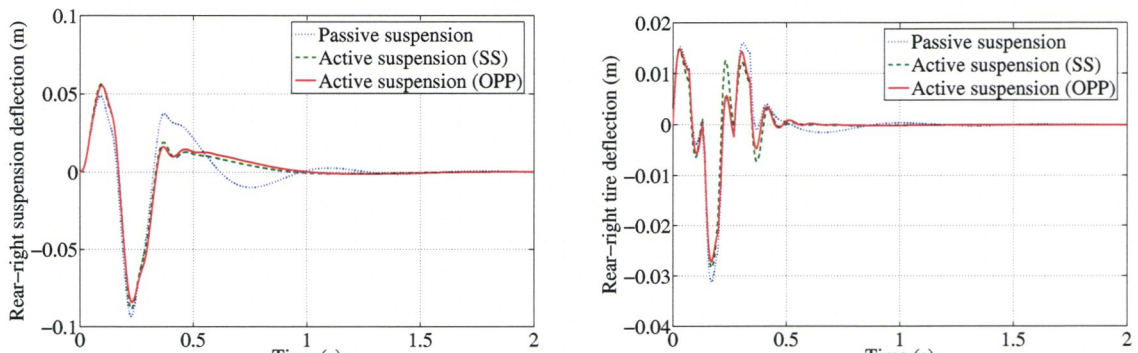


Figure 6.15: Suspension and tire deflections of the passive and active suspension (controlled with the static scheduling and the OPP algorithms)

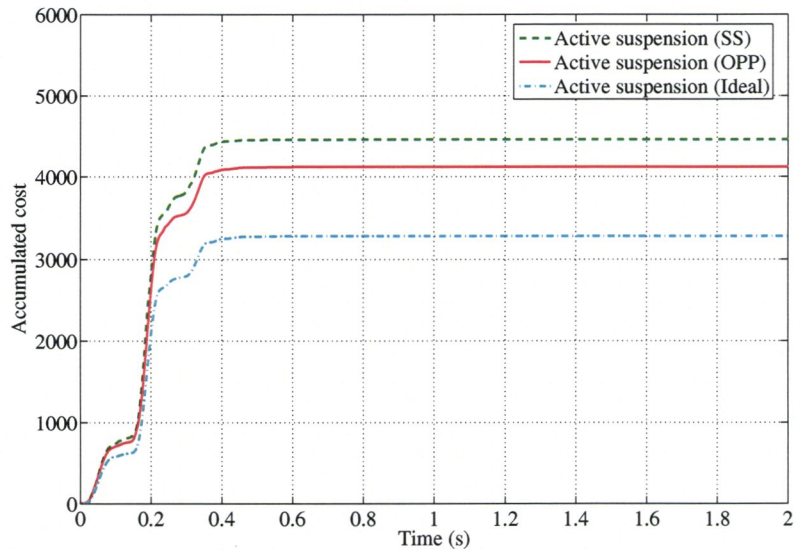


Figure 6.16: Quadratic cost functions corresponding to the active suspension (controlled with the static scheduling, with the OPP algorithm and using an ideal implementation)

6.4.4 Real-time implementation aspects of the OPP over infinite horizon algorithm

The OPP over infinite horizon algorithm requires the on-line evaluation of T quadratic functions of the extended state vector $\tilde{x}(k)$. The computational complexity is linear with respect to the period of the sequence, quadratic with respect to the extended state (like a classical state-feedback control law) and independent of the size of the horizon. In the case of the suspension example, the additional computational requirements of OPP are approximately 4.3 times those required by the state feedback operation $v(k) = \tilde{K}(p)\tilde{x}(k)$. Using OPP follows the idea of trading additional computations for a more efficient use of network resources [Yook et al., 2002]. Further reducing the computational requirements is a both a difficult and inter-

esting research issue. The application of multi-parametric programming techniques to the optimal control of hybrid systems have been recently considered [Borrelli et al., 2005]. In [Johansen and Grancharova, 2002], an approximate solution to the model predictive control of linear systems with input and state constraints was proposed. This method is based on the partitioning the state space onto hypercubes which may be further partitioned in order to meet on the cost function approximation error bounds and constraints violations bounds. By imposing an orthogonal search tree on the partition, the on-line computational requirements are significantly reduced with respect to the true optimal explicit MPC law. The search method is logarithmic with respect to the number of regions, but this number may augment exponentially with respect to the state vector size. The memory requirements, which are needed to the storage of this partitioning information and of the state-feedback control law parameters, are also tightly dependent on this number of regions. This problem is tractable for low dimensional systems but it may be problematic for problems like that treated in this paper. The difference between this approach and that of [Johansen and Grancharova, 2002] resides in the fact that we explore the *set of pointer positions*, which is in general, and particularly in this application, less complex than the *state space*. This considerably reduces the computational complexity.

6.4.5 Implementation aspects of the distributed suspension model, using state of the art methods

The study of the practical implementation of the considered distributed active suspension model, based on state of the art methods, was undertaken in [Ben Gaid et al., 2004, Kocik et al., 2005, Ben Gaid et al., 2006a]. The Controller Area Network bus was deployed to ensure the information exchange between the distributed components. In [Ben Gaid et al., 2004, Ben Gaid et al., 2006a], the tool TRUETIME [Andersson et al., 2005, Cervin et al., 2003] was used to simulate these implementations. The impact of messages priorities on control performance was studied in [Ben Gaid et al., 2004]. It was shown that the assignment of some messages priorities, which may be chosen arbitrarily from a real-time scheduling point of view, may have a considerable impact on the robustness of system. In [Kocik et al., 2005], the control performance resulting from different implementation choices of the suspension system were studied. It was shown that synchronous implementations achieve the best control performance. In [Ben Gaid et al., 2006a], the impact of the traffic that is generated by the other unrelated network nodes on the control performance of the suspension system was studied, as a function of the available bandwidth resources.

6.5 Conclusion

In this chapter, we have presented an algorithm, based on the model predictive control approach, allowing assigning on-line the optimal values of the control inputs and the scheduling inputs of resource-constrained systems. Focusing on its practical implementation aspects, such as its computational requirements, we have proposed a more efficient heuristic, called OPP. Using a pre-computed off-line schedule, OPP is able to assign on-line the values of the control inputs and the scheduling inputs based on the plant state information. The computational requirements of this heuristic are considerably reduced compared to those of the model predictive control approach. We have proved that if some mild conditions are

satisfied, then OPP guaranties the stability of the system and performance improvements compared to its basic off-line schedule. We have shown that the use of the periodic optimal control theory in the design of the control gains that will used by OPP leads to a significant simplification of its implementations, which boils down to the comparison of T quadratic cost functions, T being the period of the basic off-line schedule. The OPP scheduling algorithm was finally applied to a distributed control system: the active suspension controller of car.

7

Trading quantization precision for sampling rates

7.1 Introduction

In this chapter, we refine the previously considered model of resource-constrained systems to take into account quantization aspects and mainly focus on performance considerations in presence of information limitations. The communication constrains are modeled at the bit level, in bits per second. Using this modeling, we have to determine the control inputs that have to be updated as well as their quantization precision. In general, increasing the sampling frequency improves the disturbance rejection abilities whereas increasing the quantization precision improves the steady state precision. However, when the bandwidth is limited, increasing the sampling frequency necessitates the reduction of the quantization precision. In the opposite, augmenting the quantization precision requires the lowering of the sampling frequency. Motivated by these observations, an approach for the dynamical on-line assignment of sampling frequencies and control inputs quantization is proposed. This approach, which is based on the model predictive control (MPC) philosophy, enables to choose the sampling frequency and the quantization levels of control signals from a predefined set, in order to optimize the control performance. Naturally, handling dynamically the quantization precision requires some communication resources and some extra computational resources. Consequently, we have to jointly handle the computational complexity, the protocol bandwidth consumption and performance improvements. In order to limit the inherent complexity of the proposed protocol, we suppose that the quantization choices of the input control signals belong to a reduced finite set, which may be chosen by the designer in order to ensure the stability and to comply with the computational requirements. At each sampling period, quantization possibilities may be chosen from this set. This contrasts with the approach of [Goodwin et al., 2004], where the quantization precision of control signals is fixed. The proposed approach aims to capture the intuitive notion that high sampling rates improve the disturbance rejection and the transient behavior whereas the fine quantization improves the static precision near the origin [Elia and Mitter, 2001]. The proposed method

allows to dynamically choosing the pertinent control information to send, knowing the plant state and subject to the communication constraints.

7.2 Problem Formulation

Consider the continuous-time LTI system described by the state equation

$$\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t) \tag{7.1}$$

where $x_c(t) \in \mathbb{R}^n$ and $u_c(t) \in \mathbb{R}^m$. The system is controlled by a discrete-time controller, running at the sampling period T_s. Let $x(k) = x_c(kT_s)$ and $u(k) = u_c(kT_s)$, then the discrete-time representation of system (7.1) is given by

$$x(k+1) = Ax(k) + Bu(k). \tag{7.2}$$

We assume that the pair (A, B) is reachable and that the full state vector $x(k)$ is available to the controller at each sampling period.

The controller is connected to the actuators of the plant through a limited bandwidth communication channel. At each sampling period T_s, at most R bits can be sent to the actuators through the communication channel. The considered communication scheme is described in Figure 7.1. The control inputs, which are computed by the controller, need to be

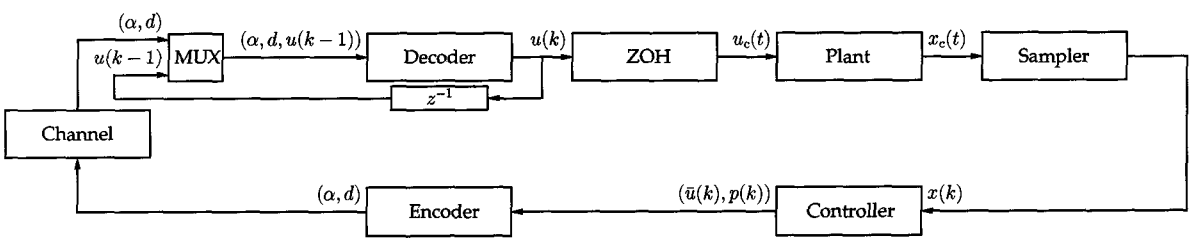


Figure 7.1: Information pattern

properly encoded before their transmission over the network. This function is performed by the encoder, which converts these control inputs into a sequence of binary symbols. The transmitted information is then decoded by the decoder and applied to the inputs of the plant through a zero order holder (ZOH). Finally, we assume that the inputs of the plant are subject to saturation constraints at the actuators, which are defined by

$$-U_i \leq u_i(k) \leq U_i, \text{ where } U_i > 0 \text{ and } i = 1, \ldots, m. \tag{7.3}$$

In the following, this information pattern (depicted in figure 7.1) is formally described.

7.2.1 Quantization aspects

The quantization is the process of approximating a continuous range of values into a relatively small finite set of discrete values, called *reconstruction levels* (or *quantization levels*). In this chapter, the quantization is performed using *mid-tread uniform quantizers*, which are characterized by an odd number of reconstruction levels (which include the value of zero). Let u a bounded continuous scalar signal verifying

$$|u| \leq U, U \in \mathbb{R}^+. \tag{7.4}$$

The set of reconstruction levels of the mid-tread uniform quantizer which may be encoded using M bits, given the lower and upper bounds $-U$ and $+U$, is defined by the set-valued function

$$\mathbb{U}(M,U) = \left\{-U + lL(M,U),\ l = 0,\dots,2^M - 2\right\} \tag{7.5}$$

where l is the *quantization index* and $L(M,U) = \frac{U}{2^{M-1}-1}$ is the *quantization step size.*

Remark 7.1. In the definition of the set valued function $\mathbb{U}(M,U)$, we have intentionally chosen to obtain an odd number of reconstruction levels $0,\dots,2^M-2$ instead of having an even number of reconstruction levels $0,\dots,2^M-1$. This choice was motivated by the need of integrating the zero reconstruction level, which have a particular signification, from the control point of view (i.e. open loop control).

Given $M \in \mathbb{N}^*$ and $U \in \mathbb{R}^+$, the quantizer $\mathcal{Q}_{(M,U)}$ is defined by

$$\mathcal{Q}_{(M,U)}(u) = \begin{cases} +U & \text{if } u > U \\ -U & \text{if } u \leq -U \\ -U + \left\lfloor \frac{u+U}{L(M,U)} + \frac{1}{2} \right\rfloor L(M,U) & \text{if } -U < u \leq U \end{cases}$$

The quantizer $\mathcal{Q}_{(M,U)}$ associates to a real scalar u its nearest neighbor in $\mathbb{U}(M,U)$.

For a given set $\mathbb{U}(M,U)$, it is convenient to define the map $\mathcal{L}_{(M,U)}$ that associates to each element v of $\mathbb{U}(M,U)$ its quantization index l. $\mathcal{L}_{(M,U)}$ is defined by

$$\begin{array}{rccl} \mathcal{L}_{(M,U)} : \mathbb{U}(M,U) & \longrightarrow & \left\{0,1,\dots,2^M-2\right\} \\ v & \longmapsto & l \end{array}$$

such that $v = -U + lL(M,U)$.

The number of reconstruction levels of the components of a given control input may vary over the time according to an optimization problem (which will be developed in the next section). Consequently, it is necessary for the decoder to identify how to reconstruct the different components of the control input of the plant from the received binary symbols. To this end, the *precision vector* $p(k) \in \mathbb{N}^m$ is introduced. $p_i(k)$ describes the number of bits that are required to encode all the reconstruction levels of the quantized control signal $u_i(k)$ (using the quantizer $\mathcal{Q}_{(p_i(k),U_i)}$). A compact representation of the precision vector has to be sent to the decoder, together with the control information. To minimize the necessary decoding information (which is described by $p(k)$), and to be able to produce compact representations of $p(k)$, the set $\mathcal{P}$ of possible values of $p(k)$ should contain a limited number of elements.

Let R_I and R_D be the number of bits which are used to encode respectively the control information and the decoding information. The communication constraints impose

$$R_I + R_D = R. \tag{7.6}$$

Then $\mathcal{P}$ must verify

$$\mathcal{P} \subset \left\{ p \in \mathbb{N}^m \text{ such that } \sum_{i=1}^{m} p_i = R_I \right\} \tag{7.7}$$

and

$$|\mathcal{P}| \leq 2^{R_D} \tag{7.8}$$

where $|\mathcal{P}|$ denotes the cardinality of $\mathcal{P}$. A compact representation of the elements of $\mathcal{P}$ may be given by the *key mapping function*

$$\begin{array}{rcl} \mathcal{K} : \mathcal{P} & \longrightarrow & \{0, 1, \ldots, |\mathcal{P}| - 1\} \\ p & \longmapsto & \kappa. \end{array}$$

The key mapping function associates to each element p of $\mathcal{P}$ a unique key κ, which is the compact representation of p. This map is bijective and known by both the encoder and decoder side.

7.2.2 Information pattern

The information that is transmitted over the communication channel is formatted as a binary stream. This binary stream is modeled by a vector of Booleans. For that reason, it is practical to define the binary conversion map $\mathcal{B}_i$, described by

$$\begin{array}{rcl} \mathcal{B}_i : \{0, \ldots, 2^i - 1\} & \longrightarrow & \{0, 1\}^i \\ q & \longmapsto & b = \begin{pmatrix} b_1 \\ \vdots \\ b_i \end{pmatrix} \end{array}$$

such that

$$q = 2^0 b_1 + 2^1 b_2 + \ldots + 2^{i-1} b_i. \tag{7.9}$$

The encoder

The encoder is a map $\mathcal{E}$ defined by:

$$\begin{array}{rcl} \mathcal{E} : \mathbb{R}^m \times \mathcal{P} & \longrightarrow & \{0, 1\}^{R_I} \times \{0, 1\}^{R_D} \\ (u, p) & \longmapsto & (\alpha, d) \end{array}$$

such that

$$\alpha = \begin{pmatrix} \mathcal{B}_{p_1} \mathcal{L}_{(p_1, U_1)} \mathcal{Q}_{(p_1, U_1)}(u_1) \\ \vdots \\ \mathcal{B}_{p_m} \mathcal{L}_{(p_m, U_m)} \mathcal{Q}_{(p_m, U_m)}(u_m) \end{pmatrix} \tag{7.10}$$

and

$$d = \mathcal{B}_{R_D} \mathcal{K}(p). \tag{7.11}$$

Knowing the chosen precision vector as well as the output of the controller, the encoder constructs the binary stream that will be transmitted over the communication channel.

The decoder

The decoder is a map $\mathcal{D}$ defined by

$$\begin{array}{rcl} \mathcal{D} : \{0, 1\}^{R_I} \times \{0, 1\}^{R_D} \times \mathbb{R}^m & \longrightarrow & \mathbb{R}^m \\ (\alpha, d, u^-) & \longmapsto & u = \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} \end{array}$$

such that

$$\begin{cases} u_i = \mathcal{L}^{-1}_{(p_i,U_i)}\left(\sum_{j=1}^{p_i} \alpha_{j+s_i} 2^{j-1}\right) & \text{if } p_i \neq 0 \\ u_i = u_i^- & \text{if } p_i = 0 \end{cases} \tag{7.12}$$

where

$$s_1 = 0 \text{ and for } i \geq 2,\ s_i = \sum_{j=1}^{i-1} p_j \tag{7.13}$$

and

$$p = \mathcal{K}^{-1}(\mathcal{B}^{-1}_{R_D}(d)). \tag{7.14}$$

The decoder performs the reconstruction of the plant control inputs from the received binary stream. It first determines the decoding key by applying the inverse mapping $\mathcal{K}^{-1}\mathcal{B}^{-1}_{R_D}$ to d. Then, the reconstruction of the quantized inputs may be performed, using the relations (7.13) and (7.12) . It is assumed that saturation levels U_i, $i = 1 \ldots m$ are know by the decoder.

7.2.3 Performance index definition

The performance of the controlled system (7.1) is evaluated using a quadratic cost function, which may be seen as the design specification of its ideal controller.

$$J_c(x_c, u_c, 0, T_f) = \int_0^{T_f} \left(x_c^T(t) Q_c x_c(t) + u_c^T(t) R_c u_c(t)\right) dt + x_c^T(T_f) S_c x_c(T_f) \tag{7.15}$$

where $T_f = NT_s$ is the final time, expressed in multiples of the sampling period T_s, N is the final time in discrete-time, and Q_c, R_c and S_c are positive definite matrices. These matrices define the design specifications of the ideal controller. The sampled-data representation of the cost function $J_c(x_c, u_c, 0, T_f)$ at the sampling period T_s is

$$J(x, u, 0, N) = \sum_{k=0}^{N-1} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}^T & Q_2 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + x^T(N) Q_0 x(N). \tag{7.16}$$

The expressions of Q_1, Q_2, Q_{12} and Q_0 are given in ([Åström and Wittenmark, 1997], pp. 411–412). Let

$$Q = \begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}^T & Q_2 \end{bmatrix}.$$

In the following, we assume that Q, Q_2 and Q_0 are positive definite.

7.3 Integrated control and communication

7.3.1 An introductory example

Consider the continuous-time LTI system defined by the state matrix

$$A_c = \begin{bmatrix} 0 & 110 & 0 & 0 \\ -900 & 10 & 0 & 0 \\ 0 & 0 & 0 & 110 \\ 0 & 0 & -900 & 10 \end{bmatrix}$$

and the input matrix

$$B_c = \begin{bmatrix} 0 & 0 \\ 210 & 0 \\ 0 & 0 \\ 0 & 210 \end{bmatrix}.$$

This system is composed of two independent and identical open-loop unstable sub-systems. The design criteria of the ideal controller for the closed-loop system are defined by the matrices $Q_c = \mathrm{Diag}(30, 10, 30, 10)$ and $R_c = \mathrm{Diag}(1, 1)$. The communication channel linking the controller to the two distant actuators has a bandwidth of 5 kbps, which means that every 2 ms, at most 10 bits of information can be sent to the actuators. To tackle the problem of the control over this limited bandwidth communication channel, two simple solutions may be proposed:

1. sending alternately 10 bits of information to a single actuator such that each actuator receives 10 bits every 4 ms,

2. sending 5 bits of information to the two actuators at the same time every 2 ms.

For the first solution, the controller was derived at the sampling period of 4 ms whereas for the second one, the controller was synthesized at the sampling period of 2 ms.

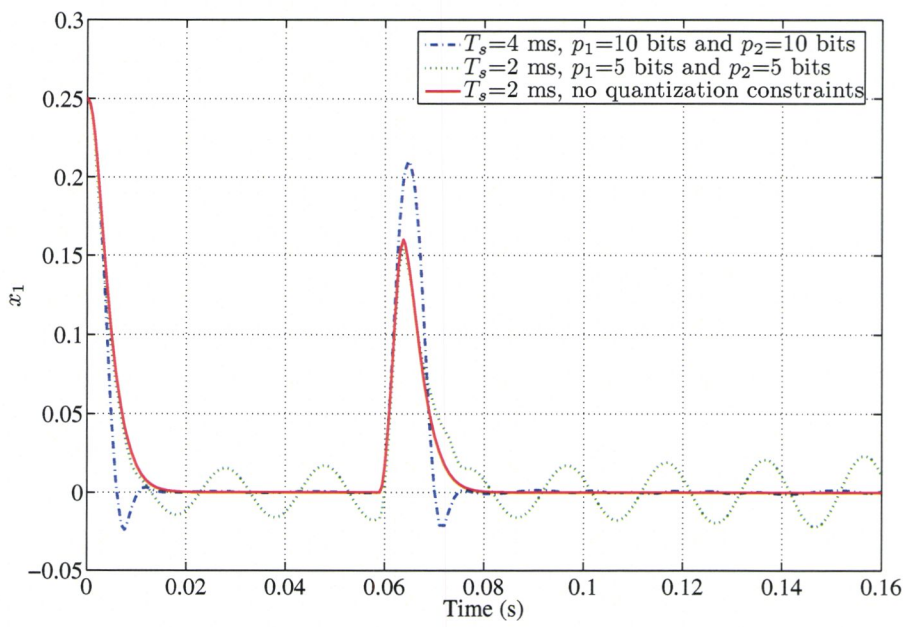


Figure 7.2: System response – state x_1

In the simulation results (depicted in Figures 7.2 and 7.3), the control performance of the first ($T_s = 4$ ms, $p_1 = 10$ bits and $p_2 = 10$ bits) and the second solutions ($T_s = 2$ ms, $p_1 = 5$ bits and $p_2 = 5$ bits) is evaluated and compared to the performance of the optimal sampled-data controller at 2 ms (with no quantization constraints). In these simulations, the global system is started from the initial condition $[0.25 \;\; 0 \;\; -0.1 \;\; 0]^T$ and disturbed at

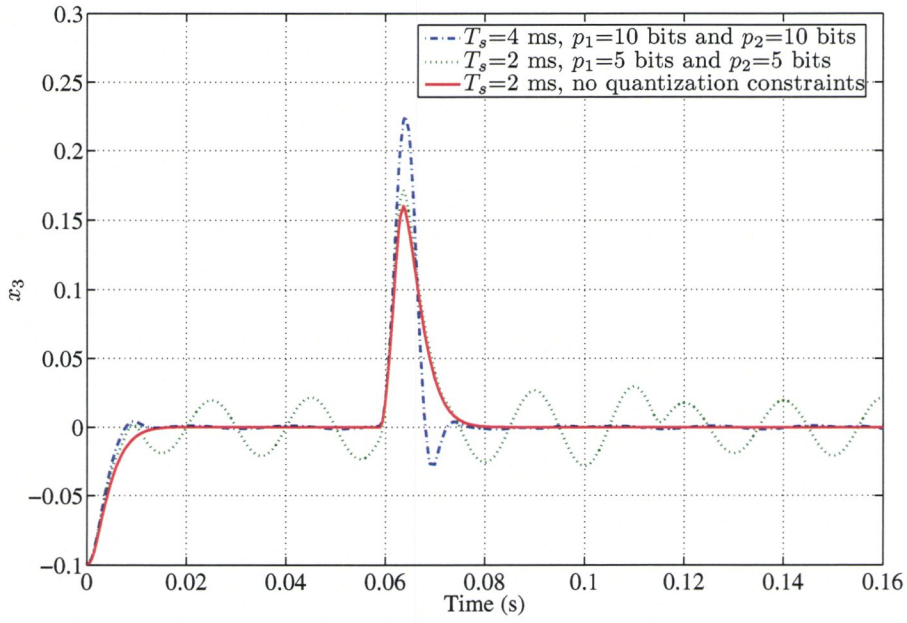


Figure 7.3: System response – state x_3

$t = 59$ ms. It may be observed that the first solution behaves badly in the transient state and has poor disturbance rejection abilities. In the opposite, the second solution behaves well in the transient state, but when the system is close to the steady state an oscillatory behavior appears. These limit cycle oscillations are due to insufficient quantization precision of the control inputs. The period of these oscillations (which is equal to 20 ms), corresponds to the natural frequency of the system (its eigenvalues are $5 \pm j314.6$). However, the first solution performs much better than the second one at the steady state. These observations are the motivation of the approach which will be presented in the next section, and which aims at dynamically assigning the number of quantization levels of the control signals in order to improve the control performance.

7.3.2 On-line control and communication algorithm

Let $s^{N-1} = (s(0), \ldots, s(N-1)) \in \mathcal{P}^N$. The sequence s^{N-1} is called the *quantization sequence*, and may be seen as the generalization of the notion of communication sequence first introduced in [Brockett, 1995]. In opposition to the notion of communication sequence introduced in [Brockett, 1995] and [Hristu and Morgansen, 1999] and used in the previous chapters, quantization sequences considered in this chapter take into account quantization aspects. To each quantization sequence s^{N-1}, a control gains sequence $K_s^{N-1} = (K_s(0), \ldots, K_s(N-1))$ may be associated.

Remark 7.2. The notation s^{N-1} was used to denote a quantization sequence, which is a *static* notion. The notation $p(k)$ denotes the precision vector, which represent the quantization precision of a given control input $u(k)$, at any given discrete-time instant k, and is rather a *runtime* notion.

Assume that a set $\mathcal{C}$ of quantization sequences is defined. Then, the problem of the integrated control and communication may be solved on-line using the model predictive control philosophy. The model predictive control is an elegant solution to tackle the hybrid aspect of the considered model, where both the control inputs and the quantization decisions need to be determined at each sampling period. Model predictive control was successfully applied to the control of hybrid systems [Bemporad and Morari, 1999] and to the problems of integrated control and medium access allocation, as illustrated in Chapter 6.

Using the MPC approach, an optimization problem is solved at each sampling period, in order to determine both the control inputs of the plant $u(k)$ and their quantization precision $p(k)$. This problem is formulated as in the set of equations (7.17) below.

$$\begin{cases} \min\limits_{s^{N-1}\in\mathcal{C}} \sum\limits_{h=0}^{N-1} \begin{bmatrix} \hat{x}(h) \\ \hat{u}(h) \end{bmatrix}^T Q \begin{bmatrix} \hat{x}(h) \\ \hat{u}(h) \end{bmatrix} + \hat{x}^T(N) Q_0 \hat{x}(N) \\ \text{subject to} \\ \hat{x}(0) = x(k) \\ \hat{u}(-1) = u(k-1) \\ \hat{p}(h) = s(h) \\ \text{and for all } h \in \{0,\dots,N-1\}\text{, for all } i \in \{1,\dots,m\} \\ \hat{v}(h) = K_s(h)\hat{x}(h) \\ \hat{u}_i(h) = \mathcal{Q}_{(\hat{p}_i(h),U_i)}(\hat{v}(h)) \text{ if } \hat{p}_i(h) \neq 0 \\ \hat{u}_i(h) = \hat{u}_i(h-1) \text{ if } \hat{p}_i(h) = 0 \\ \hat{x}(h+1) = A\hat{x}(h) + B\hat{u}(h) \end{cases} \tag{7.17}$$

In these equations, for any sequence f^{N-1}, the notation $\hat{f}(h)$ denotes the predicted value of the variable $f(k+h)$. Using this approach, the control performance $J(\hat{x},\hat{u},0,N)$ over a horizon of N is predicted, for the $|\mathcal{C}|$ quantization sequences of $\mathcal{C}$. $\hat{x}(h)$ and $\hat{u}(h)$ constitute the predicted values of $x(k+h)$ and $u(k+h)$, for a given control sequence K_s^{N-1} and quantization sequence s^{N-1}. $\hat{p}(h)$ represents the quantization precision of the predicted inputs $\hat{u}(h)$. If $\hat{p}_i(h) = 0$, then the predicted control input $\hat{u}(h)$ cannot be updated. Consequently, its previous value $\hat{u}(h-1)$ will be maintained. Control inputs $u_i(h)$ which have to be updated (i.e. whose precision vectors satisfy $\hat{p}_i(h) \neq 0$) are computed by the application of the quantizer $\mathcal{Q}$ to the i^{th} element of the result of the state feedback operation $\hat{v}(h) = K_s(h)\hat{x}(h)$. The solution of this optimization problem is the quantization sequence $s^{N-1^*} = (s^*(0),\dots,s^*(N-1))$ that minimizes the cost function $J(\hat{x},\hat{u},0,N)$ subject to the communication constraints. According to the receding horizon philosophy, the precision vector and control inputs at instant k are respectively given by

$$p(k) = \hat{p}^*(0) = s^*(0) \tag{7.18}$$

and

$$\begin{cases} u_i(k) = \hat{u}_i^*(0) = \mathcal{Q}_{(p_i(k),U_i)}(\hat{v}^*(0)) & \text{if } p_i(k) \neq 0 \\ u_i(k) = u_i(k-1) & \text{if } p_i(k) = 0 \end{cases} \tag{7.19}$$

where

$$\hat{v}^*(0) = K_{s^*}(0)x(k). \tag{7.20}$$

The choice of set $\mathcal{C}$ plays in important role in ensuring the stability and performance improvements. In the following, a heuristic method for choosing the elements of the set $\mathcal{C}$ is proposed.

7.3.3 Determination of the set $\mathcal{C}$

In theory, if all the quantization possibilities which satisfy the communication constraints are considered, the set $\mathcal{C}$ remains finite. However, the number of its elements may be gigantic. The sequences of $\mathcal{C}$ have to be chosen in order to perform a tradeoff between performance improvements and real-time computation constraints.

In the following, the general ideas concerning the choice of the set $\mathcal{C}$ are described. The main idea is that the use of $\mathcal{C}$ maintains the stability and improves the disturbance rejection using the on-line MPC algorithm (described by equations (7.17), (7.18), (7.20) and (7.19)). To this end, $\mathcal{C}$ may be chosen as the union of two sets:

The set of basic sequences $\mathcal{C}_B$

This set is constructed based on all the possible circular permutations of a basic sequence, called the *nominal sequence*. Let the nominal sequence be

$$s_{Nom}^{N-1} = (s_{Nom}(0), s_{Nom}(1), \ldots, s_{Nom}(N-1)) .$$

Then the elements of $\mathcal{S}_B$ are the N sequences:

$$\begin{array}{c} (s_{Nom}(0), s_{Nom}(1), \ldots, s_{Nom}(N-2), s_{Nom}(N-1)) \\ (s_{Nom}(1), \ldots, s_{Nom}(N-2), s_{Nom}(N-1), s_{Nom}(0)) \\ \vdots \\ (s_{Nom}(N-1), s_{Nom}(0), s_{Nom}(1), \ldots, s_{Nom}(N-2)) . \end{array}$$

The nominal sequence s_{Nom}^{N-1} has to be chosen such that the following static control algorithm:

$$\begin{array}{l} p(k) = s_{Nom}((k+k_0) \mod N) \\ v(k) = K_{s_{Nom}}((k+k_0) \mod N)x(k) \\ u_i(k) = \mathcal{Q}_{(p_i(k),U_i)}(v(k)) \text{ if } p_i(k) \neq 0 \\ u_i(k) = u_i(k-1) \text{ if } p_i(k) = 0 \end{array} \tag{7.21}$$

ensures the stability of the plant for all fixed $k_0 \in \{0, \ldots, N-1\}$.

Intuitively, the elements of this set should be sequences whose precision vectors are "extremal", in the sense that the precision of some inputs is maximal, whereas the precision of the remaining control inputs is zero. Using this choice, the optimal selection of quantization and control gains sequences may be done by the methods which ignore quantization, such as [Rehbinder and Sanfridson, 2004, Lincoln and Bernhardsson, 2002, Zhang and Hristu-Varsakelis, 2005] or the $\mathcal{H}_2$ optimization described in Chapter 5.

The set of reaction sequences $\mathcal{C}_R$

This set has to be chosen by the designer in order to improve the responsiveness to disturbances. Intuitively, the elements of these sequences should be more "equilibrated" (i.e. containing precision vectors whose elements share fairly the available quantization precision). In the same way as the construction of the set $\mathcal{C}_B$, the set $\mathcal{C}_R$ contains the basic reaction sequences $\bar{\mathcal{C}}_R$ as well as all their circular permutations. An example illustrating the choice of the reaction sequences is given in the next sub-section.

7.3.4 A numerical example

In order to evaluate the proposed approach, the example of Section 7.3.1 is reconsidered. The parameters of the MPC algorithm are

$$R_I = 8 \text{ and } R_D = 2,$$

$$\mathcal{P} = \left\{ p_1 = \begin{bmatrix} 8 \\ 0 \end{bmatrix}, p_2 = \begin{bmatrix} 0 \\ 8 \end{bmatrix}, p_3 = \begin{bmatrix} 4 \\ 4 \end{bmatrix} \right\},$$

$$\mathcal{C}_B = \left\{ s^{20}_{Nom} = s^{20}_1, s^{20}_2 \right\} \text{ and } \bar{\mathcal{C}}_R = \left\{ s^{20}_3, s^{20}_4, s^{20}_5 \right\},$$

where

$$s^{20}_1 = (p_1, p_2, p_1, p_2, p_1, p_2, \ldots),$$
$$s^{20}_2 = (p_2, p_1, p_2, p_1, p_2, p_1, \ldots),$$
$$s^{20}_3 = (p_3, p_3, p_1, p_2, p_1, p_2, \ldots),$$
$$s^{20}_4 = (p_3, p_3, p_2, p_1, p_2, p_1, \ldots),$$

and

$$s^{20}_5 = (p_3, p_3, p_3, p_3, p_3, p_3, \ldots).$$

Finally, the control gains sequences are

$$K^{20}_{s_1} = K^{20}_{s_2} = (K_A, K_A, K_A, K_A, \ldots),$$

$$K^{20}_{s_3} = K^{20}_{s_4} = (K_B, K_B, K_A, K_A, K_A, K_A, \ldots),$$

and

$$K^{20}_{s_5} = (K_B, K_B, K_B, K_B, K_B, K_B, \ldots),$$

where

$$K_A = \begin{bmatrix} -0.6483 & 1.4188 \\ -0.6483 & 1.4188 \end{bmatrix}$$

and

$$K_B = \begin{bmatrix} 2.8819 & 2.5938 \\ 2.8819 & 2.5938 \end{bmatrix}.$$

Control gains K_A and K_B were derived using the method developed in Chapter 5. Simulation results are depicted in Figures 7.4 and 7.5. It may be observed that using the MPC approach, the tradeoff between precision and rapidity is performed. The oscillations near the practical stability region are reduced and the response to the unpredictable disturbances improved. These improvements are due to a more intelligent choice of the number of quantization levels of the control signals, which are allocated according to the state value of the system.

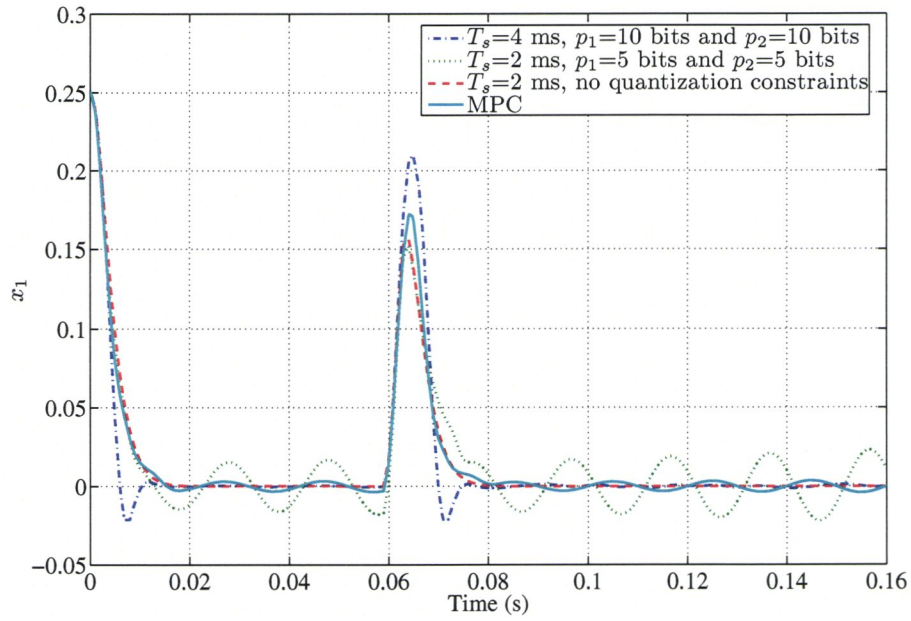


Figure 7.4: System response – state x_1

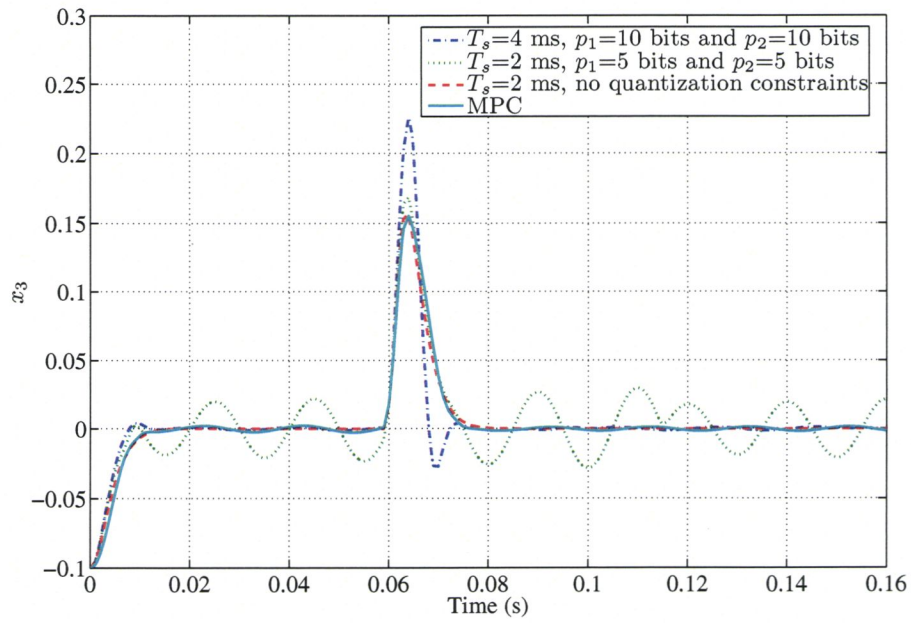


Figure 7.5: System response – state x_3

7.4 Conclusion

In this chapter, another point of view of the problem of the control over limited bandwidth communication channels was studied. A finely grained model was adopted, ensuring the re-

spect of the bandwidth constraints and allowing the characterization of the influence of the sampling frequency and quantization precision on the control performance. An efficient approach for the improvement of disturbance rejection capabilities and steady state precision was then proposed. This approach dynamically assigns the quantization precision of the control signals in order to improve the control performance, taking into account the communication and computation requirements of the introduced dynamic protocol. The model predictive control technique that was deployed in this chapter uses a few number of quantization sequences (the sequences of the set $\mathcal{C}$), in its "prediction phase", in opposition to the model predictive controller that was deployed in Chapter 6, and that computed the predicted cost corresponding to all the possible communication sequences. Thus, the computational complexity of the MPC algorithm that was deployed in this chapter is comparable to that of the OPP algorithm over a finite horizon as addressed in Chapter 6.

8

Optimal real-time scheduling of control tasks based on state-feedback resource allocation

8.1 Introduction

In both control and real-time scheduling theories, the notion of "instantaneous computational needs" of a control application is not defined. The computational resources are consequently allocated according to the "worst case needs" of these applications. This corresponds to the use of periodic sampling, which on one hand simplifies the design problem, but on the other hand leads to an unnecessary usage of some computational resources. In fact, as previously illustrated, a control task that is close to the equilibrium needs less computational resources that another control task that is severely disturbed. Similarly, the real-time design of control tasks is based on their worst-case execution time. As illustrated in [Cervin et al., 2002], a resource saving may be obtained if these hard real-time constraints are "conveniently" relaxed, since control systems may be situated in between hard and soft real-time systems. A significant economy in computing resources may be performed if more elaborate models are used by the two communities, which amounts to co-designing the control and the scheduling [Årzén et al., 2000,Palopoli, 2002,Martí, 2002,Cervin, 2003,Årzén and Cervin, 2005,Simon et al., 2005,Henriksson, 2006,Xia and Sun, 2006].

The idea of the dynamical allocation of processor resources as a function of the plant state was proposed in [Martí et al., 2004], assuming that the performance index of a task controlling a dynamic system at a given state is proportional to its period and to the controlled system's error. However, this assumption is not theoretically justified. The problem of the optimal on-line sampling period assignment was studied in [Henriksson and Cervin, 2005]. An optimal periodic sampling period assignment heuristic was proposed. The feedback scheduler is triggered periodically and computes the optimal sampling frequencies of the control tasks based on a finite horizon predicted cost. However, the stability and computational complexity issues of the feedback scheduler were not addressed.

In this chapter, the optimal integrated control and scheduling approach, which was applied in the previous chapters to the scheduling of communication resources, is generalized

to the problem of the real-time scheduling of control tasks. We propose a model of the single processor execution of control and non-control tasks. The co-design of the control and the real-time scheduling is performed in a two-step approach. First, an optimal off-line schedule is derived, using the previously described approach of the optimal $\mathcal{H}_2$ scheduling, taking into account the particular constraints that are specific to the single processor execution of tasks. Second, a plant state feedback scheduling approach is applied on-line. Its objective is to improve the control performance by quickly reacting to unexpected disturbances. Performance improvements as well as stability guarantees using this approach are proven. The proposed method is evaluated on a comprehensive implementation model, which was simulated using the tool TRUETIME [Andersson et al., 2005].

8.2 Optimal off-line scheduling

8.2.1 Problem formulation

Consider a collection of $\mathcal{N}$ independent continuous-time LTI systems $(S^{(j)})_{1\leq j\leq \mathcal{N}}$. Assume that each system $S^{(j)}$ is controlled by a task $\tau^{(j)}$, which is characterized by its worst-case execution time $c^{(j)}$. Since we are interested in deriving an optimal off-line schedule, the scheduling decisions must be made periodically, rather than at arbitrary time instants [Liu, 2000]. Let T_p be this elementary time period. The time line is then partitioned into intervals of length T_p called *time slots*. Control tasks may be started only at the beginning of the time slots. A control task may need more than one time slot to execute. In the proposed model, we assume that control tasks are executed *non-preemptively*. This assumption has two essential benefits. First, it simplifies the problem formulation and solving. Second, non-preemptive scheduling ensures a minimal and constant input/output latency for control tasks. In fact, if the input and output operations are performed respectively at the beginning and at the end of a control task, preemption causes variations in the input/output latency, which may degrade the control performance, as illustrated in [Cervin et al., 2003, Martí, 2002].

Usually, control tasks share the processor with other sporadic or aperiodic tasks, which do not perform the computations of the control laws, and called *non-control tasks*. Examples of such tasks include signal processing, monitoring or communication tasks. The computational load that these tasks induce may be described by their processor utilization rate, noted $\mathcal{U}^{nc}$. If these non-control tasks do not have any real-time constraints, then they may be executed in the portions of the time slots where control tasks do not execute. Otherwise, their schedulability and real-time constraints have to be taken into account at design time. Using off-line scheduling, the scheduling pattern (starting time instants of control tasks or non-control tasks reserved slots) is repeated identically every *major cycle*, or *hyperperiod*. In the following, we will denote by $T \times T_p$ the hyperperiod of the static schedule. The hyperperiod is equal to T when expressed in terms of numbers of elementary time slots T_p.

***Example* 8.1.** An example of an off-line schedule of two control tasks $\tau^{(1)}$ and $\tau^{(2)}$ and a set of sporadic tasks using at most 44% of the CPU is given in Figure 8.1.

Task scheduling may be described by associating Boolean scheduling functions $\gamma^{(j)}$ to control tasks $\tau^{(j)}$ such that

$$\gamma^{(j)}(k) = \begin{cases} 1 & \text{if the execution of task } \tau^{(j)} \text{ finishes in the interval } [(k-1)T_p, kT_p) \\ 0 & \text{otherwise.} \end{cases} \tag{8.1}$$

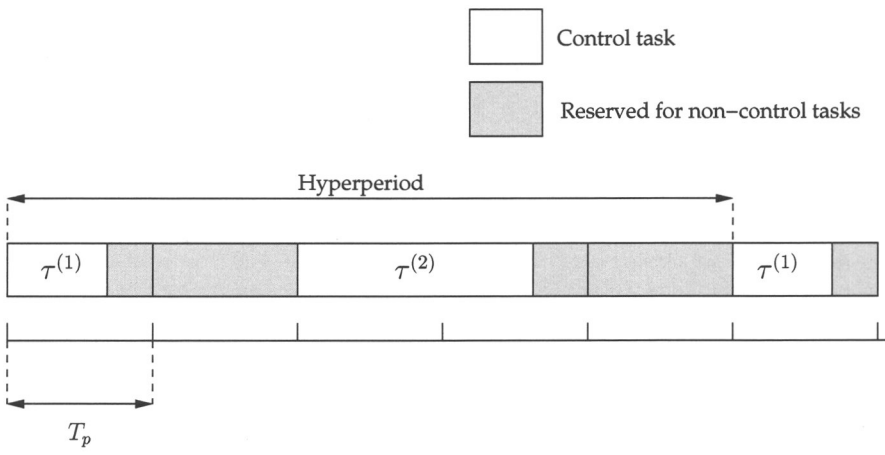


Figure 8.1: An example of an off-line scheduling of control and non-control tasks

$\gamma^{(j)}(k)$ is called the *execution end indicator* of the jobs of task $\tau^{(j)}$. Due to the use of a non-preemptive scheduling, the $\left\lceil \frac{c^{(j)}}{T_p} \right\rceil$ slots containing or preceding the end of a job of task $\tau^{(j)}$ are allocated to its execution. Using this observation, the processor time slots utilization by a given control task $\tau^{(j)}$ may be described by

$$e^{(j)}(k) = \sum_{l=k}^{k+\left\lceil \frac{c^{(j)}}{T_p} \right\rceil - 1} \gamma^{(j)}(l). \tag{8.2}$$

$e^{(j)}(k)$ is the *task execution indicator* corresponding to the jobs of task $\tau^{(j)}$ and verifies

$$e^{(j)}(k) = 1 \iff \begin{array}{l} \text{the processor is allocated to task } \tau^{(j)} \\ \text{during a sub-interval of } [(k-1)T_p, kT_p). \end{array} \tag{8.3}$$

During interval $[(k-1)T_p, kT_p)$, the processor can execute only one control task. This constraint may be modeled by the following inequality

$$\sum_{j=1}^{\mathcal{N}} e^{(j)}(k) \leq 1. \tag{8.4}$$

In order to guarantee the computational needs of non-control tasks, described by their processor utilization rate $\mathcal{U}^{nc}$, the scheduling decisions of control tasks must satisfy

$$\mathcal{U}^{nc} + \frac{1}{TT_p} \sum_{k=1}^{T} \sum_{j=1}^{\mathcal{N}} c^{(j)} \gamma^{(j)}(k) \leq 1. \tag{8.5}$$

Each system $S^{(j)}$ is characterized by its sampled-data model, derived at the sampling period T_p, and described by

$$x^{(j)}(k+1) = A^{(j)}x^{(j)}(k) + B_1^{(j)}w^{(j)}(k) + B_2^{(j)}u^{(j)}(k) \tag{8.6a}$$

$$z^{(j)}(k) = C_1^{(j)}x^{(j)}(k) + D_{11}^{(j)}w^{(j)}(k) + D_{12}^{(j)}u^{(j)}(k). \tag{8.6b}$$

where $x^{(j)}(k) \in \mathbb{R}^{n_j}$ is the state vector, $w^{(j)}(k) \in \mathbb{R}^{r_j}$ is the disturbance input, $u^{(j)}(k) \in \mathbb{R}^{m_j}$ is the control input and $z^{(j)}(k) \in \mathbb{R}^{q_j}$ is the controlled output. We assume that:

1. The pair $(A^{(j)}, B_2^{(j)})$ is controllable,
2. $Q^{(j)} = \begin{bmatrix} C_1^{(j)T} \\ D_{12}^{(j)T} \end{bmatrix} \begin{bmatrix} C_1^{(j)} & D_{12}^{(j)} \end{bmatrix} = \begin{bmatrix} Q_{xx}^{(j)} & Q_{xu}^{(j)} \\ Q_{xu}^{(j)T} & Q_{uu}^{(j)} \end{bmatrix} > 0.$

Using straightforward algebraic manipulations, the global system S constituted by systems $(S^{(j)})_{1 \leq j \leq \mathcal{N}}$ may be described using an extended state model representing a global system S described by

$$x(k+1) = Ax(k) + B_1 w(k) + B_2 u(k) \tag{8.7a}$$
$$z(k) = C_1 x(k) + D_{11} w(k) + D_{12} u(k). \tag{8.7b}$$

Let $n = \sum_{j=1}^{\mathcal{N}} n_j$, $m = \sum_{j=1}^{\mathcal{N}} m_j$ and Q the matrix defined by

$$Q = \begin{bmatrix} C_1{}^T \\ D_{12}{}^T \end{bmatrix} \begin{bmatrix} C_1 & D_{12} \end{bmatrix}. \tag{8.8}$$

In the considered modeling, when a control task finishes its execution, then it immediately updates the plant, which means that

$$u^{(j)}(k) \text{ is updated during interval } [(k-1)T_p, kT_p) \iff \gamma^{(j)}(k) = 1. \tag{8.9}$$

The digital-to-analog converters, use zero-order-holders to maintain the last received control commands constant until new control values are updated. Consequently, if a control command is not updated during the time slot $[(k-1)T_p, kT_p)$, then it is held constant. This assertion may be modeled by

$$\gamma^{(j)}(k) = 0 \Longrightarrow u^{(j)}(k) = u^{(j)}(k-1). \tag{8.10}$$

Based on this problem formulation, the optimal integrated control and off-line scheduling problem may be stated and solved, in a similar way to Chapter 5.

8.2.2 Solving of the optimal scheduling sub-problem

The scheduling functions $\gamma^{(j)}(k)$ and $e^{(j)}(k)$, describing the optimal off-line schedule, are both periodic with period T (i.e. $\gamma^{(j)}(k) = \gamma^{(j)}(k+T)$ and $e^{(j)}(k) = e^{(j)}(k+T)$). Note that T expresses the number of processor slot times T_p that are allocated to the execution of the different tasks, during the hyperperiod. In order to determine the $\mathcal{H}_2$ norm of the system, we adopt a definition of this norm that is based on the impulsive response of a linear discrete-time periodic system, as in Chapter 5. Following the same methodology as in Chapter 5, it is easy to show that the optimal off-line scheduling problem is also a MIQP. The constraints may be also classified into two groups. The first group, which includes the constraints that intervenes in the computation of all the impulsive responses $z^{ik}, 1 \leq i \leq \sum_{j=1}^{\mathcal{N}} r_j, 0 \leq k \leq T-1$ over the horizon $\mathcal{H}$, contains the constraints (8.2), (8.4), (8.5) as well

as the constraints ensuring the periodicity of the scheduling (i.e $\gamma^{(j)}(k) = \gamma^{(j)}(k+T)$ for all $k \in \{1,\dots,\mathcal{H}\}$ and $j \in \{1,\dots\mathcal{N}\}$). The second group contains the constraints that are related to the computation of a single impulsive response z^{ik}. Assuming that there exists a T-periodic off-line scheduling ensuring the reachability of system (8.7), the resolution of this problem leads to the optimal off-line schedule $\check{\Gamma}^*$ defined by

$$\check{\Gamma}^* = \begin{bmatrix} \check{\Gamma}^{(1)^*} \\ \vdots \\ \check{\Gamma}^{(\mathcal{N})^*} \end{bmatrix}, \text{where } \check{\Gamma}^{(j)^*} = \begin{bmatrix} \check{\Gamma}^{(j)^*}(1) \\ \vdots \\ \check{\Gamma}^{(j)^*}(\mathcal{H}) \end{bmatrix} \text{ for } j = 1,\dots,\mathcal{N}.$$

8.2.3 Solving of the optimal control sub-problem

Given an off-line schedule, the ordered execution of the control tasks during the major cycle $[0, T \times T_p)$ may be described by the sequence $(s(0),\dots,s(\mathcal{T}-1))$, where $\mathcal{T}$ is the number of control task executions during the hyperperiod $[0, T \times T_p)$. For example, the sequence $(s(0), s(1), s(2), s(3)) = (1,2,2,3)$ indicates that during the hyperperiod, the processor begins by executing task $\tau^{(1)}$, followed by two consecutive executions of task $\tau^{(2)}$, which are followed by the execution of task $\tau^{(3)}$. The total number of control task executions during the hyperperiod is $\mathcal{T} = 4$. Knowing the optimal off-line schedule $\check{\Gamma}^*$, which is a solution of the optimal $\mathcal{H}_2$ scheduling subproblem, it is then possible to derive the optimal control gains, according to the $\mathcal{H}_2$ performance criterion. In opposition to the optimal $\mathcal{H}_2$ scheduling subproblem, the determination of the optimal control gains may be performed on each system separately.

In the practical implementation of control tasks, it is impossible to directly measure the disturbances that act on the system. It is only possible to detect their effects on the state. When the controller have only access to the plant state, the optimal $\mathcal{H}_2$ controller becomes identical to the optimal LQR controller [Chen and Francis, 1995]. For that reason, in the following, we will consider that $D_{11} = 0$. The expression of $z^{(j)}(k)$ reduces to

$$z^{(j)}(k) = C_1^{(j)} x^{(j)}(k) + D_{12}^{(j)} u^{(j)}(k).$$

Let $k_q^{(j)}$ be the index of the time slot corresponding to the $(q+1)^{th}$ update of the control commands of task $\tau^{(j)}$. More formally, $k_q^{(j)}$ are the discrete instants verifying

$$\exists k \in \{1,\dots,T\}, \exists i \in \mathbb{N} \text{ such that } k_q^{(j)} = k + iT \text{ and } \check{\Gamma}^{(j)^*}(k) = 1.$$

Remark 8.1. Note that instants $k_q^{(j)}$ are irregularly spaced along the time line (i.e. $k_{q+1}^{(j)} - k_q^{(j)}$ is not necessarily equal to $k_q^{(j)} - k_{q-1}^{(j)}$). This comes from the fact that in the optimal off-line schedule, due to the resource contention, the actuation time instants of a given control task are not regularly spaced (as illustrated, for example, in Figure 8.1).

Based on this definition, the optimal control subproblem may be stated as follows.

$$\begin{cases} \min\limits_{u^{(j)}} \sum\limits_{k=0}^{+\infty} {z^{(j)}}^T(k) z^{(j)}(k) \\ \text{subject to} \\ x^{(j)}(k+1) = A^{(j)} x^{(j)}(k) + B_2^{(j)} u^{(j)}(k) \\ u^{(j)}(k) \text{ may be updated if and only if } \exists q \in \mathbb{N} \text{ such that } k_q^{(j)} = k \\ u^{(j)}(k) = u^{(j)}(k-1) \text{ if and only if } \forall q \in \mathbb{N},\ k_q^{(j)} \neq k. \end{cases} \tag{8.11}$$

Since zero order holders are used to maintain the last received control inputs constant, an equivalent down-sampled discrete-time representation of system $S^{(j)}$ may be deduced. Let

$$\vec{x}^{(j)}(q) = x^{(j)}(k_q^{(j)}),$$
$$\vec{u}^{(j)}(q) = u^{(j)}(k_q^{(j)}),$$

then the following relation is obtained:

$$\vec{x}^{(j)}(q+1) = \vec{A}^{(j)}(q)\vec{x}^{(j)}(q) + \vec{B}_2^{(j)}(q)\vec{u}^{(j)}(q) \tag{8.12}$$

with

$$\begin{aligned} \vec{A}^{(j)}(q) &= A^{(j)^{k_{q+1}^{(j)} - k_q^{(j)}}}, \\ \vec{B}_2^{(j)}(q) &= \sum_{i=0}^{k_{q+1}^{(j)} - k_q^{(j)} - 1} A^{(j)^i} B_2^{(j)}. \end{aligned}$$

Equation (8.12) describes the evolution of the state of system $S^{(j)}$, at the time slots where its control inputs are updated, and that may not be regularly spaced. We will assume in the following that system (8.12) is reachable. Remarking that for k such that $k_q^{(j)} \leq k < k_{q+1}^{(j)}$,

$$\begin{bmatrix} x^{(j)}(k) \\ u^{(j)}(k) \end{bmatrix} = \begin{bmatrix} A^{(j)^{k-k_q^{(j)}}} & \sum_{i=0}^{k-k_q^{(j)}-1} A^{(j)^i} B_2^{(j)} \\ 0_{m_j,n_j} & I_{m_j} \end{bmatrix} \begin{bmatrix} \vec{x}^{(j)}(q) \\ \vec{u}^{(j)}(q) \end{bmatrix},$$

then

$$\sum_{k=k_q^{(j)}}^{k_{q+1}^{(j)}-1} z^{(j)^T}(k) z^{(j)}(k) = \begin{bmatrix} \vec{x}^{(j)}(q) \\ \vec{u}^{(j)}(q) \end{bmatrix}^T \vec{Q}^{(j)}(q) \begin{bmatrix} \vec{x}^{(j)}(q) \\ \vec{u}^{(j)}(q) \end{bmatrix},$$

where

$$\vec{Q}^{(j)}(q) = \sum_{k=k_q^{(j)}}^{k_{q+1}^{(j)}-1} \begin{bmatrix} A^{(j)^{k-k_q^{(j)}}} & \sum_{i=0}^{k-k_q^{(j)}-1} A^{(j)^i} B_2^{(j)} \\ 0_{m_j,n_j} & I_{m_j} \end{bmatrix}^T Q^{(j)} \begin{bmatrix} A^{(j)^{k-k_q^{(j)}}} & \sum_{i=0}^{k-k_q^{(j)}-1} A^{(j)^i} B_2^{(j)} \\ 0_{m_j,n_j} & I_{m_j} \end{bmatrix}.$$

Let $\mathcal{T}^{(j)}$ be the number of executions of task $\tau^{(j)}$ during the hyperperiod. Since the optimal schedule is periodic, then matrices $\vec{A}^{(j)}(q)$, $\vec{B}^{(j)}(q)$ and $\vec{Q}^{(j)}(q)$ are $\mathcal{T}^{(j)}$-periodic. Based on these notations, it is easy to see that problem (8.11) is equivalent to the following periodic optimal control problem:

$$\begin{cases} \min_{\vec{u}^{(j)}} \sum_{q=0}^{\infty} \begin{bmatrix} \vec{x}^{(j)}(q) \\ \vec{u}^{(j)}(q) \end{bmatrix}^T \vec{Q}^{(j)}(q) \begin{bmatrix} \vec{x}^{(j)}(q) \\ \vec{u}^{(j)}(q) \end{bmatrix} \\ \text{subject to} \\ \vec{x}^{(j)}(q+1) = \vec{A}^{(j)}(q)\vec{x}^{(j)}(q) + \vec{B}_2^{(j)}(q)\vec{u}^{(j)}(q) \end{cases} \tag{8.13}$$

Since matrix $A^{(j)}$ is invertible (because it was obtained by discretization), then for k such that $k_q^{(j)} \leq k < k_{q+1}^{(j)}$, matrix

$$\begin{bmatrix} A^{(j)^{k-k_q^{(j)}}} & \sum_{i=0}^{k-k_q^{(j)}-1} A^{(j)^i} B_2^{(j)} \\ 0_{m_j,n_j} & I_{m_j} \end{bmatrix}$$

has full rank. Consequently, matrix $\vec{Q}^{(j)}(q)$ is definite positive. Problem (8.13) is a periodic optimal control problem over an infinite horizon, whose solution was presented and discussed in Section 4.4 of Chapter 4. Problem (8.13) admits a unique solution $\vec{u}^{(j)^*}(q)$ defined by

$$\vec{u}^{(j)^*}(q) = -\vec{K}^{(j)}(q)\vec{x}^{(j)}(q)$$

Let $\vec{S}^{(j)}(q)$ be the solution of the Riccati equation associated to the optimal control problem (8.13). Matrices $\vec{S}^{(j)}(q)$ and $\vec{K}^{(j)}(q)$ are both $\mathcal{T}^{(j)}$-periodic. Their expressions may be deduced following the lifting approach of Section 4.4 of Chapter 4. The optimal cost corresponding to an evolution from instant ι to $+\infty$ is given by

$$\sum_{q=\iota}^{+\infty} \vec{z}^{(j)^T}(q)\vec{z}^{(j)}(q) = \vec{x}^{(j)^T}(\iota)\vec{S}^{(j)}(\iota)\vec{x}^{(j)}(\iota). \tag{8.14}$$

Remark 8.2. Note that optimal control gains $\vec{K}^{(j)}(l)$, which are $\mathcal{T}^{(j)}$-periodic, are independent of ι, which represent the initial time of the optimal control problem. This considerably simplifies the implementation of the controller, and justifies their use in the on-line scheduling approach, which will be described in the next section.

Remark 8.3. In, equation (8.14), the cost corresponding to an evolution of system $S^{(j)}$ from instant $k_\iota^{(j)}$ to $+\infty$ is expressed as a quadratic function of the state $\vec{x}^{(j)}(\iota) = x^{(j)}(k_\iota^{(j)})$. However, equation (8.14) may only be used in instants where the control inputs of system $S^{(j)}$ are updated (i.e. belonging to the set $\left\{k_q^{(j)}, q \in \mathbb{N}\right\}$). In the following, we prove how the cost corresponding to an evolution of system $S^{(j)}$ from instant an arbitrary time instant k to $+\infty$ may be written as quadratic function of an extended state $\tilde{x}^{(j)}(k)$ such that

$$\tilde{x}^{(j)}(k) = \begin{bmatrix} x^{(j)}(k) \\ u^{(j)}(k-1) \end{bmatrix}.$$

Let $\mathcal{Y}^{(j)} = \begin{bmatrix} 0_{m_j,n_j} & I_{m_j} \end{bmatrix}$, $\bar{\mathcal{Y}}^{(j)} = \begin{bmatrix} I_{n_j} & 0_{n_j,m_j} \end{bmatrix}$ and $\tilde{\Psi}^{(j)}$ the matrix defined by

$$\tilde{\Psi}^{(j)} = \begin{bmatrix} \begin{bmatrix} A^{(j)} & 0_{n_j,m_j} \end{bmatrix} + B^{(j)}\mathcal{Y}^{(j)} \\ \mathcal{Y}^{(j)} \end{bmatrix}.$$

Let k such that $k_{q-1}^{(j)} < k < k_q^{(j)}$. Since for l such that $k \leq l < k_q^{(j)}$, $u^{(j)}(l) = u^{(j)}(k) = u^{(j)}(k_{q-1}^{(j)})$, then for $k \leq l \leq k_q^{(j)}$

$$\tilde{x}^{(j)}(l) = \tilde{\Psi}^{(j)^{l-k}}\tilde{x}^{(j)}(k).$$

Consequently, the cost corresponding to an evolution of system $S^{(j)}$ from an arbitrary time instant k to $+\infty$ is

$$\sum_{l=k}^{+\infty} z^{(j)^T}(l)z^{(j)}(l) = \tilde{x}^{(j)^T}(k)\tilde{S}^{(j)}(k)\tilde{x}^{(j)}(k),$$

where

$$\tilde{S}^{(j)}(k) = \begin{cases} \left[\tilde{\Psi}^{(j)^{k_q^{(j)}-k}}\right]^T \check{S}^{(j)}(q)\tilde{\Psi}^{(j)^{k_q^{(j)}-k}} + \sum_{l=k}^{k_q^{(j)}-1} \left[\tilde{\Psi}^{(j)^{l-k}}\right]^T Q^{(j)}\tilde{\Psi}^{(j)^{l-k}} & \text{if } k_{q-1}^{(j)} < k < k_q^{(j)} \\ \vec{S}^{(j)}(q) & \text{if } k = k_q^{(j)} \end{cases} \tag{8.15}$$

and

$$\check{S}^{(j)}(q) = \bar{\mathcal{Y}}^{(j)^T} \vec{S}^{(j)}(q) \bar{\mathcal{Y}}^{(j)}.$$

8.2.4 A numerical example

Consider the collection of 3 sampled-data LTI systems $S^{(1)}$, $S^{(2)}$ and $S^{(3)}$. $S^{(1)}$ and $S^{(2)}$ are two second order systems, the first is open-loop unstable whereas the second is open-loop stable. System $S^{(3)}$ is a fourth order open-loop unstable system. These three systems are defined by

$$\begin{aligned}
x^{(1)}(k+1) &= \begin{bmatrix} 0.9967 & 0.0266 \\ -0.2500 & 0.9987 \end{bmatrix} x^{(1)}(k) + \begin{bmatrix} 0.0132 \\ 0.9998 \end{bmatrix} w_1^{(1)}(k) + \begin{bmatrix} 0.0133 \\ 0.9999 \end{bmatrix} u^{(1)}(k) \\
z^{(1)}(k) &= \begin{bmatrix} \begin{bmatrix} 89.4427 & 0 \\ 0 & 3.1623 \end{bmatrix} x^{(1)}(k) \\ 10^{-3} \begin{bmatrix} 0 & 15.9630 & 14.7321 \\ 0 & 14.7321 & 69.1671 \end{bmatrix} w^{(1)}(k) \\ u^{(1)}(k) \end{bmatrix} \\
x^{(2)}(k+1) &= \begin{bmatrix} 0.9937 & 0.3458 \\ -0.0250 & 1.0007 \end{bmatrix} x^{(2)}(k) + \begin{bmatrix} 0.1243 \\ 0.7911 \end{bmatrix} w_1^{(2)}(k) + \begin{bmatrix} 0.1384 \\ 0.8008 \end{bmatrix} u^{(2)}(k) \\
z^{(2)}(k) &= \begin{bmatrix} \begin{bmatrix} 2.2361 & 0 \\ 0 & 1 \end{bmatrix} x^{(2)}(k) \\ 10^{-3} \begin{bmatrix} 0 & 4.1674 & 3.8133 \\ 0 & 3.8133 & 17.4947 \end{bmatrix} w^{(2)}(k) \\ 3.1623\, u^{(2)}(k) \end{bmatrix} \\
x^{(3)}(k+1) &= \begin{bmatrix} 1.1180 & 0 & 0.0025 & 0.5531 \\ 0 & 1 & 0.2129 & 0 \\ 0 & 0 & 0.7613 & 0 \\ 0.4518 & 0 & 0.0093 & 1.1180 \end{bmatrix} x^{(3)}(k) + 10^{-4} \begin{bmatrix} 23 \\ 9 \\ 75 \\ 81 \end{bmatrix} w_1^{(3)}(k) \\
&\quad + 10^{-4} \begin{bmatrix} 28 \\ 11 \\ 88 \\ 106 \end{bmatrix} u^{(3)}(k) \\
z^{(3)}(k) &= \begin{bmatrix} \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 31.62 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x^{(3)}(k) \\ 10^{-3} \begin{bmatrix} 0 & 0.3088 & 0.3809 & 0.0712 & 0.0922 \\ 0 & 0.3809 & 0.4984 & 0.1031 & 0.1278 \\ 0 & 0.0712 & 0.1031 & 0.1118 & 0.1176 \\ 0 & 0.0922 & 0.1278 & 0.1176 & 0.1249 \end{bmatrix} w^{(3)}(k) \\ u^{(3)}(k) \end{bmatrix}
\end{aligned}$$

Each system $S^{(j)}$ is controlled by an independent control task $\tau^{(j)}$, $j \in \{1, 2, 3\}$. Task $\tau^{(3)}$ is followed by a communication task τ^c. There is a data dependency between tasks $\tau^{(3)}$ and τ^c. Consequently, task τ^c have to be executed immediately after task $\tau^{(3)}$.

We assume that the execution platform is an Allen-Bradley CompactLogix 5320 programmable logic controller [Allen-Bradley, 2001] from Rockwell Automation. This execution platform is characterized by the execution times of its basic assembler instructions. The complete list of the execution times of all the instructions may be found in [Allen-Bradley, 2001]. Based on this assembler language, a handwritten assembler code of the control tasks was derived. The estimated worst-case execution times (WCET) of the different tasks are given in Table 8.1. We also assume that a set of aperiodic tasks may need to be executed on the same processor, requiring an utilization rate $\mathcal{U}^{nc}$ up to 40%. Based on this requirement as well as WCET of the tasks, the elementary time slot duration T_p was chosen equal to 1 ms.

Table 8.1: Worst-case execution times of the control tasks

Task	WCET (μs)
$\tau^{(1)}$	282.94
$\tau^{(2)}$	282.94
$\tau^{(3)} + \tau^{c}$	779.82 + 240.19

The execution of tasks $\tau^{(1)}$ and $\tau^{(2)}$ requires only one time slot. The consecutive execution of tasks $\tau^{(3)}$ and τ^{c} requires two time slots.

The optimal solutions, corresponding to different choices of T are illustrated in Table 8.2. The relative optimality gap of the used branch and bound algorithm is equal to 10^{-5}, which means that the best-obtained solution will be considered as an *optimal* solution if the difference between its cost and the lower bound of the *true optimal* cost is less than 0.01%. The computations were performed on a PC equipped with a 3.6 GHz Intel Pentium IV processor and 1 GB of RAM. The optimization problem was solved using the solver CPLEX (Release 9.1.0) from ILOG. In this particular implementation of the optimization algorithm, the horizon $\mathcal{H}$ of the computation of the impulsive responses must be a multiple of T. It is sufficient to choose $\mathcal{H}$ greater than 27.

Table 8.2: Optimal $\mathcal{H}_2$ norm as a function of T

T	$\mathcal{H}$	$\mathcal{H}_2$ norm	Optimal Schedule	CPU Time Default (s)	CPU Time with initial solution (s)
4	28	13.2472	321...	86	13
5	30	9.7463	2131...	58	40
6	30	11.7966	11213...	243	185
7	28	13.0122	13213...	533	482
8	32	12.1146	312131...	1482	1151
9	27	10.6545	1312312...	1796	1244
10	30	9.7463	21312131...	4288	2062

In Table 8.2, the two columns *CPU Time* indicate the time needed by the optimization

algorithm to finish. The algorithm finishes when it proves that the best obtained solution is *close enough* to the lower bound of the true optimal solution (i.e. the relative difference between the best obtained solution and the true optimal one is less than the specified relative optimality gap). The optimization results indicate that the minimal optimal schedule is of length $T = 5$, and is described by the sequence $(s(0), s(1), s(2), s(3)) = (2, 1, 3, 1)$. In this schedule, task $\tau^{(2)}$ is first executed, followed by the execution of task $\tau^{(1)}$, which is followed by the execution of task $\tau^{(3)}$, which is followed by the execution of task $\tau^{(1)}$. The length of the optimal schedules which gives the best $\mathcal{H}_2$ norm ($\mathcal{H}_2 = 9.7463$) is a multiple of 5. The column *CPU Time Default* indicates the required CPU time using the default parameters of the solver. The column *CPU time with initial solution* indicates the required CPU time when the cost of a feasible initial solution is taken into account by the algorithm. In fact, it is always possible for the designer to derive a feasible schedule *ad-hoc*, using rules of thumb like those described in [Åström and Wittenmark, 1997]. The advantage of the branch and bound method is that it is able to use this feasible solution to considerably reduce the search space by pruning the regions that are proved to be worst than this given initial solution. This reduces the number of regions to be explored by the algorithm. Finally, note that the required time to find the optimal solution is lower than the needed time to prove that it is optimal. For example, for $T = 6$, the optimal solution was found after 69 seconds but 185 seconds were necessary to prove that it is optimal. For large-scale problems, this algorithm may be run during a determined amount of time to find solutions that are better than the given initial schedule, which was derived *ad-hoc* by the designer.

Remark 8.4. In the obtained off-line schedule, task $\tau^{(1)}$ was executed twice in the hyperperiod. Consequently, the optimal control gains for its two instances may be different. For that reason, a subscript will be added to distinguish two instances of the same task that may use different control gains. The two instances of task $\tau^{(1)}$ will be noted $\tau_1^{(1)}$ and $\tau_2^{(1)}$.

8.3 On-line scheduling of control tasks

Assume that an off-line schedule, specifying how the different control and non-control tasks should be executed, was designed. This off-line schedule, which may be stored in a table, describes when each control task should be started and possibly the starting time instants of the time slots that are pre-allocated to non-control tasks.

At runtime, the execution of the periodic off-line schedule may be described using the notion of *pointer*. The pointer may be seen as a variable p that contains the index of the control task to execute. The pointer is incremented after each control task execution. If it reaches the end of the sequence, its position is reset. After each task execution, the position of the pointer is updated according to:

$$p := (p + 1) \mod \mathcal{T} \tag{8.16}$$

Knowing the pointer position p, the control task to execute is $s(p)$. It is convenient to define the map φ, which associates to the pointer position its absolute position (expressed as multiples of T_p in the schedule). The map φ linking the different pointer positions to their absolute positions in the example of Figure 8.2 is defined in Table 8.3.

Since the non-control tasks are sporadic or aperiodic, they do not necessarily use all the pre-allocated time slots for their execution. Consequently, it appears useful and perspicacious to employ this unused CPU power for improving the quality of control. Such improve-

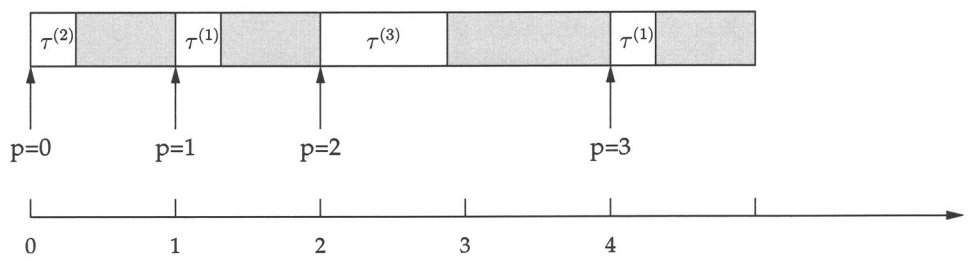


Figure 8.2: Absolute positions of the pointer

Table 8.3: The map φ

Pointer position p	Absolute position $\varphi(p)$
0	0
1	1
2	2
3	4

ments may be possible by executing more frequently the control tasks of the systems who have the greatest need for additional computing resources. This requires specific scheduling algorithms, because the control tasks are used to control dynamical systems, where the timing of the input and output operations is capital for the stability and performance guarantees.

The idea behind the proposed scheduling strategy is to use the free available computing resources (i.e. when there are no sporadic nor aperiodic tasks to execute for example) in order to execute a feedback scheduler, which is responsible of determining the best scheduling of the control tasks in the future. In the following, the issues that are related to the execution of the feedback scheduler are discussed as well as the optimal pointer placement scheduling strategy, which represents the cornerstone of the used adaptive scheduling strategy. Finally, the method of reduction of the computational complexity of the feedback scheduler is described.

8.3.1 Execution of the feedback scheduler

As previously mentioned, the feedback scheduler is executed in the non-control tasks reserved time frames, when they are "free". Depending on the possibilities of the execution platform (whether it supports preemption or not), the feedback scheduler may be executed as:

- A preemptive task, with a given priority and deadline, and which will be discarded by the scheduler if it misses its deadline (in this case the schedule execution is not modified).
- A non-preemptive task that is executed on predefined sub-slots of the non-control tasks slots.

In both cases, the possible real-time constraints of the sporadic or aperiodic tasks should be taken into account, when the feedback scheduler task is added.

Remark 8.5. In real-time scheduling theory, static scheduling is mostly used in safety critical systems, in order to ensure a strong temporal determinism. The first motivation behind the use of a basic static schedule in the on-line scheduling heuristic is the need for predictability. In fact, the adaptive feedback scheduler, which will described thereafter, needs to compute a predicted cost function, in order to determine the scheduling decision. Using the basic sequence, the computation of the predicted cost is considerably reduced, because it boils down to the computation of a reduced number of quadratic functions. The use of the basic static schedule simplifies also the computations of the feedback gains. Furthermore, the implementation of the feedback scheduler as an extension of the basic sequencer may be easier than handling dynamic priorities in priority-based schedulers.

8.3.2 Adaptive scheduling of control tasks

In the proposed adaptive scheduling strategy, instead of systematically using (8.16), the position of the pointer is placed according to the knowledge of the controlled plants states, in order to ensure the improvement of the control performance as measured by a quadratic cost function. This strategy relies on computing $\mathcal{T}$ predicted cost functions corresponding to an evolution of the global system for $\mathcal{T}$ different positions of the pointer. Let

$$\tilde{x}(k) = \begin{bmatrix} \tilde{x}^{(1)}(k) \\ \vdots \\ \tilde{x}^{(\mathcal{N})}(k) \end{bmatrix}.$$

If the pointer is placed at position p at instant k, then the cost function corresponding to an evolution of system $S^{(j)}$ over an infinite horizon starting from the state $\tilde{x}^{(j)}(k)$ at instant k and using the static scheduling algorithm is

$$J^{(j)}(k,p) = \sum_{i=0}^{\infty} z^{(j)^T}(k+i) z^{(j)}(k+i) = \tilde{x}^{(j)}(k)^T \tilde{S}^{(j)}(\varphi(p)) \tilde{x}^{(j)}(k). \tag{8.17}$$

If the pointer is placed at position p at instant k, then the cost function corresponding to an evolution of the global system S over an infinite horizon starting from the state $\tilde{x}(k)$ at instant k and using the static scheduling algorithm is

$$J(k,p) = J^{ss}(\tilde{x}(k), k, +\infty, p) = \sum_{j=1}^{\mathcal{N}} J^{(j)}(k,p). \tag{8.18}$$

This strategy (called optimal pointer placement scheduling) was employed in Chapter 6 in the context of networked control systems in order to reduce the considerable computational complexity, which is required to find the true optimal control and scheduling decisions (this latter problem was investigated in Chapter 4).

In the following, we describe how this concept may be deployed for monoprocessor scheduling. As mentioned previously, the feedback scheduler is triggered in a non-control tasks reserved time frame, if this time frame is free (i.e. there are no sporadic nor aperiodic tasks to execute for example). It first acquires the state of all the controlled plants at

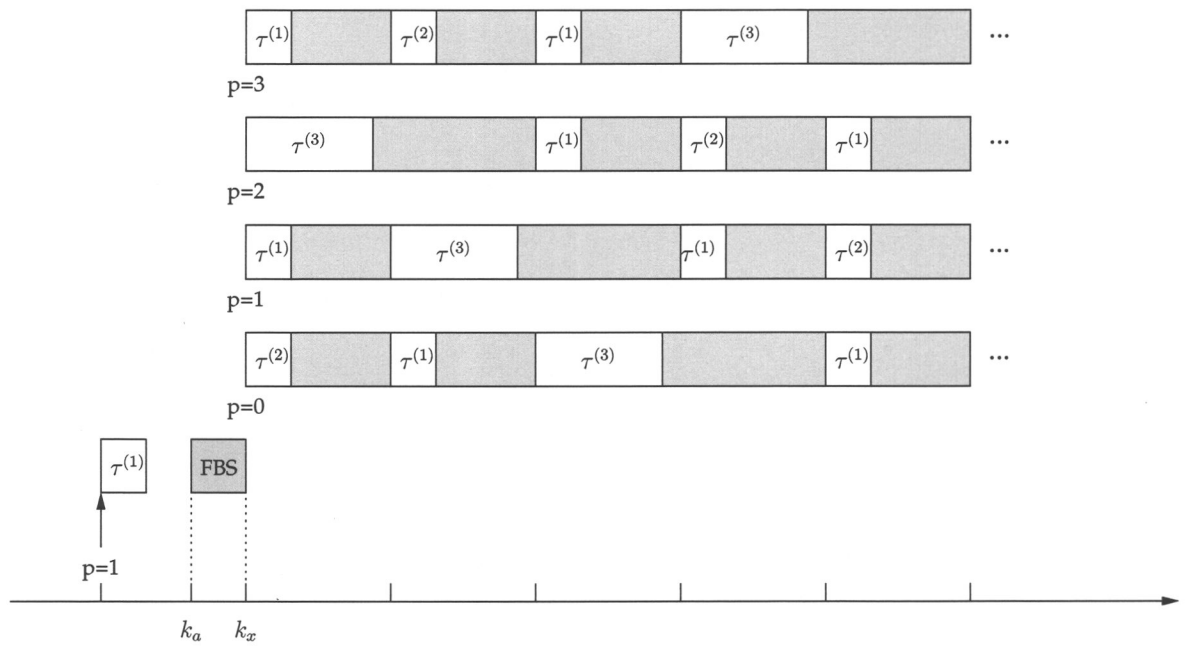


Figure 8.3: Optimal pointer placement scheduling of tasks $\tau^{(1)}$, $\tau^{(2)}$ and $\tau^{(3)}$. FBS is abbreviation of feedback scheduler

instant k_a (the instant where feedback scheduler begins its execution), and then computes $\mathcal{T}$ predicted cost functions corresponding to an evolution over an infinite horizon starting at instant k_x (the instant where the next control tasks will begin its execution) for the $\mathcal{T}$ possible pointer positions. The task that will be executed at instant k_x is the one that corresponds to the pointer position that gives the minimal predicted cost. Figure 8.3 illustrates the method (in the case of the adaptive scheduling of the numerical example of Section 8.2.4).

Note that the feedback scheduler uses the knowledge of the state at instant k_a to compute the predicted cost functions $J(k_x, p)$, for $p \in \{0, \ldots, \mathcal{T}-1\}$, corresponding to an evolution starting at k_x. If the plant model is used to predict the state at instant k_x knowing the state at instant k_a, then it is easy to establish that

$$J(k_x, p) = \tilde{x}^T(k_x)\tilde{S}(\varphi(p))\tilde{x}(k_x) = \tilde{x}^T(k_a)\hat{S}(\varphi(p))\tilde{x}(k_a).$$

The expression of $\hat{S}(\varphi(p))$ may be easily deduced from the plant model, the expression of $\tilde{S}(\varphi(p))$ and the control gains.

8.3.3 Reduction of the feedback scheduler overhead

In the following, a method for the reduction of the computational complexity of the feedback scheduler is proposed. This method relies on intuitive observations, which may be related to the concept of practical stability. The method is motivated by the fact that when a system $S^{(j)}$ is at the equilibrium (i.e. $x^{(j)} = 0$), its computational resources may be given to other tasks, which control perturbed systems. If system $S^{(j)}$ is close to the equilibrium, it may be less frequently updated than other systems that experience severe disturbances. This proximity from the equilibrium may be formalized by defining a practical equilibrium region $\mathcal{R}^{(i)}$ for each system $S^{(i)}$. To define how much a system $S^{(i)}$ is close to the equilibrium, positive

constants $\varepsilon_x^{(i)}$ are introduced. $\varepsilon_x^{(i)}$ have to be chosen small enough to consider that when $\left\|\tilde{x}^{(i)}(k)\right\|_\infty \leq \varepsilon_x^{(i)}$, then system $S^{(i)}$ is considered practically at the equilibrium. Each practical equilibrium region may be formally defined by

$$\mathcal{R}^{(i)} = \left\{ \tilde{x}^{(i)}(k) / \left\|\tilde{x}^{(i)}(k)\right\|_\infty \leq \varepsilon_x^{(i)} \right\}.$$

Remark 8.6. Computing the infinity norm of the state vector is less expensive, in terms of computations, than computing its 1-norm or 2-norm. This observation motivates its use in the proposed approach. However, from a strictly conceptual point of view, any other norm may also be used for the practical equilibrium detection.

Systems in the practical equilibrium region are systems whose affected computational resources may be released for the profit of other tasks. In order to ensure stability and performance improvements, this dynamic resource allocation have to be done using a well-defined methodology, which will be developed in the following.

The basic ideas behind the proposed method for the reduction of the computational complexity of the feedback scheduler relies on changing the adaptive scheduling paradigm from
find the best pointer position
to
find a pointer position that is better than the cyclic schedule
The answer to the latter question requires less computational resources than the answer to the first one, and may take advantage from knowing that a given system is practically stable. For a given system $S^{(j)}$, let

$$\bar{J}_{min}^{(j)}(p) = \min_{\tilde{x}^{(j)}(k) \in \mathcal{R}^{(j)}} \left(J^{(j)}(k,p) \right)$$

and

$$\bar{J}_{max}^{(j)}(p) = \max_{\tilde{x}^{(j)}(k) \in \mathcal{R}^{(j)}} \left(J^{(j)}(k,p) \right),$$

then we have the following proposition.

Proposition 8.1. *Let p_1 and p_2 be two pointer positions and I a subset of $\{1,\ldots,\mathcal{N}\}$. If*

$$\forall i, \in I, \ \left\|\tilde{x}^{(i)}(k)\right\|_\infty \leq \varepsilon_x^{(i)}$$

and

$$\sum_{j \in \{1,\ldots,\mathcal{N}\}-I} J^{(j)}(k,p_2) + \sum_{j \in I} \bar{J}_{max}^{(j)}(p_2) < \sum_{j \in \{1,\ldots,\mathcal{N}\}-I} J^{(j)}(k,p_1) + \sum_{j \in I} \bar{J}_{min}^{(j)}(p_1)$$

then

$$J(k,p_2) < J(k,p_1)$$

Proof. This result directly follows from the fact that

$$J(k,p_1) = \sum_{j \in \{1,\ldots,\mathcal{N}\}-I} J^{(j)}(k,p_1) + \sum_{j \in I} J^{(j)}(k,p_1) \geq \sum_{j \in \{1,\ldots,\mathcal{N}\}-I} J^{(j)}(k,p_1) + \sum_{j \in I} \bar{J}_{min}^{(j)}(p_1)$$

and

$$J(k,p_2) = \sum_{j \in \{1,\ldots,\mathcal{N}\}-I} J^{(j)}(k,p_2) + \sum_{j \in I} J^{(j)}(k,p_2) \leq \sum_{j \in \{1,\ldots,\mathcal{N}\}-I} J^{(j)}(k,p_2) + \sum_{j \in I} \bar{J}_{max}^{(j)}(p_2).$$

□

Constants $\bar{J}^{(j)}_{min}(p)$ and $\bar{J}^{(j)}_{max}(p)$ may be easily pre-computed off-line using a QP solver.

***Example* 8.2.** In order to illustrate the gains in terms of computational complexity, we reconsider the example of Section 8.2.4. The computation of the true pointer position requires $\mathcal{T}(n+m-1)(n+m+1) = 480$ additions and $\mathcal{T}(n+m)(n+m+1) = 528$ multiplications. If systems $S^{(2)}$ and $S^{(3)}$ are practically stable, searching for a pointer position that may be better than the next pointer position requires at the worst case $\mathcal{T}((n_1+m_1-1)(n_1+m_1+1) = 32$ additions and $\mathcal{T}((n_1+m_1)(n_1+m_1+1) = 48$ multiplications, thus a reduction of respectively 95% and 90% of the computational complexity.

In the case that input operations are performed by independent hardware devices and that their computational overhead may be neglected, a possible pseudocode for the feedback scheduler is given in the listing below. This feedback scheduling algorithm is called reactive pointer placement (RPP) scheduling algorithm. In this listing, for a given $p \in \{0,\ldots,\mathcal{T}-1\}$, I_p is a set of plant indices and $\bar{I}_p = \{1,\ldots,\mathcal{N}\} - I_p$. For a given $p \in \{0,\ldots,\mathcal{T}-1\}$, $\mathcal{P}_p$ is a set of pointer positions that does not contain p. Typically, I_p is chosen such that $\bar{I}_p = \{s(\pi), \pi \in \mathcal{P}_p\}$. Consequently, I_p contains $s(p)$. The choice of the elements of $\mathcal{P}$ and I_p depends on the available computing resources that may be dedicated to the feedback-scheduler. More resources are available; more potential pointer positions may be tested.

Read $x(k)$;
$p := p+1 \mod \mathcal{T}$;
if $\forall i, \in I_p,\ \left\|\tilde{x}^{(i)}(k)\right\|_\infty \leq \varepsilon_x^{(i)}$ **then**
  if $\exists \pi \in \mathcal{P}_p / \sum_{j\in\bar{I}_p} J^{(j)}(k,\pi) + \sum_{j\in I_p} \bar{J}^{(j)}_{max}(\pi) < \sum_{j\in\bar{I}_p} J^{(j)}(k,p) + \sum_{j\in I_p} \bar{J}^{(j)}_{min}(p)$ **then**
    $p := \pi$;
  endif
endif
execute task $s(p)$;

Algorithm 8.1: Pseudocode of the feedback scheduling algorithm, called Reactive Pointer Placement (RPP) scheduling algorithm

The algorithm first checks whether the plants whose indices are in I_p are in the practical stability region. If yes, then system $S^{s(p)}$ is also in the practical stability region. Consequently, the resources of control task $\tau^{s(p)}$ may be released for the profit of other control tasks controlling disturbed plants. The algorithm searches whether the execution of another control task $\tau^{s(\pi)}, \pi \in \mathcal{P}_p$ instead of $\tau^{s(p)}$ improves the global system performance. It may be remarked that the first "**if** test" consumes less computational resource than the second one.

8.3.4 Stability and performance improvements

Let $J^{rpp}(\tilde{x}(i), i, f)$ be the cost function corresponding to an evolution from instant $k = i$ to instant $k = f$ starting from the extended state $\tilde{x}(i)$ where the RPP scheduling algorithm is applied. The performance improvements of the RPP scheduling algorithm are stated in the following theorem:

Theorem 8.1. *Let $\tilde{x}(0)$ be a given initial extended state of the global system S (composed of systems $(S^{(j)})_{1\leq j\leq\mathcal{N}}$) and p_0 an initial pointer position of the static scheduling algorithm, then*

$$J^{rpp}(\tilde{x}(0), 0, \infty) \leq J^{ss}(\tilde{x}(0), 0, \infty, p_0).$$

Proof. Let $\tilde{x}^{rpp}$ be the extended state trajectory of system S when scheduled using the RPP algorithm, $p(l)$ the pointer position at the l^{th} execution of the RPP algorithm and k_l the instant corresponding to the end of this l^{th} execution. According to the RPP strategy, if the pointer position at the l^{th} execution is set to $p(l)$ instead of $(p(l-1)+1) \mod \mathcal{T}$ then necessarily

$$J^{ss}(\tilde{x}(k_l), k_l, +\infty, p(l)) \leq J^{ss}(\tilde{x}(k_l), k_l, +\infty, (p(l-1)+1) \mod \mathcal{T}).$$

Consequently, at each instant k_l, the relation

$$J^{rpp}(\tilde{x}(0), 0, k_{l-1}) + J^{ss}(\tilde{x}^{rpp}(k_l), k_l, +\infty, p(l)) \leq J^{ss}(\tilde{x}(0), 0, +\infty, p_0)$$

holds. When $l \to +\infty$, the last relation reduces to

$$J^{rpp}(\tilde{x}(0), 0, +\infty) \leq J^{ss}(\tilde{x}(0), 0, +\infty, p_0) \tag{8.19}$$

□

Theorem 8.1 demonstrates that the control performance of the RPP strategy is better or similar to that obtained using the static scheduling algorithm. The stability of the RPP scheduling algorithm directly follows from Theorem 8.1 and is given the following corollary:

Corollary 8.1. *If Q is positive definite and if the asymptotic stability of the global system S (composed of systems $(S^{(j)})_{1\leq j\leq \mathcal{N}}$) is guarantied by the static scheduling algorithm, then it is also ensured by the RPP scheduling algorithm.*

Proof. When Q is positive definite, then $J^{ss}(\tilde{x}(0), 0, +\infty, p_0)$ (respectively $J^{rpp}(\tilde{x}(0), 0, +\infty)$) is finite if and only if system S is asymptotically stable. Knowing the asymptotic stability of the system scheduled using the static scheduling algorithm and using relation (8.19), the corollary is proved. □

8.3.5 Reduction of input readings overhead

In some situations, the input operations are performed directly by the processor. The control application may also be networked: the control inputs from the plant and outputs to the plant are transmitted over the network. Consequently, reading the inputs from the plant at each execution of the feedback scheduler may result in a computational or bandwidth overhead. In order to reduce this overhead, the feedback scheduler must read at most n_s inputs, such that $n_s < n$. The most efficient way is to read these inputs according to the optimal off-line sampling periods of the controlled systems. In the following a heuristic approach for selecting the inputs that are read by the feedback scheduler is presented. Since the feedback scheduler is executed in the non-control slots which follow the execution of the control tasks, we may associate to each pointer position of set n_s control inputs that will be read. Let Π be the set of all the permutations of the $\mathcal{T}$-tuple $(0, 1, \ldots, \mathcal{T}-1)$. Let $\pi \in \Pi$ a given permutation and ζ_π the sequence defined by

$$\zeta_\pi = (\zeta_\pi(0), \zeta_\pi(1), \ldots, \zeta_\pi(\mathcal{T}-1)) = (s(\pi(0)), s(\pi(1)), \ldots, s(\pi(\mathcal{T}-1)))\,. \tag{8.20}$$

Assume that a n_s-tuple of permutations $\pi = (\pi_1, \ldots, \pi_{n_s})$ is defined. If the pointer is at position p, then the inputs $\{\zeta_{\pi_1}(p), \ldots, \zeta_{\pi_{n_s}}(p)\}$ are read. The *binary detection indicators* associated to each system $S^{(j)}$ are defined by:

$$\begin{cases} \mathcal{Z}_\pi^{(j)}(p) = 1 & \text{if } \exists k \in \{1, \ldots, n_s\} \text{ such that } \zeta_{\pi_k}(p) = j \\ \mathcal{Z}_\pi^{(j)}(p) = 0 & \text{otherwise.} \end{cases} \tag{8.21}$$

The binary detection indicators indicate whether or not the outputs of system $S^{(j)}$ are read, for a given position p of the sequence pointer. Let $\bar{\mathcal{Z}}_\pi^{(j)}(k) = \mathcal{Z}_\pi^{(j)}(k \mod \mathcal{T})$. The inputs that are read at position p are given by the the detection sequence

$$\zeta = (\{\zeta_{\pi_1^*}(0), \ldots, \zeta_{\pi_{n_s}^*}(0)\}, \ldots, \{\zeta_{\pi_1^*}(\mathcal{T}-1), \ldots, \zeta_{\pi_{n_s}^*}(\mathcal{T}-1)\})$$

that is determined by the solution $\left(\pi_1^*, \ldots, \pi_{n_s}^*\right)$ of the following optimization problem

$$(POS)\begin{cases} \left(\pi_1^*, \ldots, \pi_{n_s}^*\right) = \min\limits_{(\pi_1,\ldots,\pi_{n_s})\in\Pi^{n_s}} \sum\limits_{j=1}^{\mathcal{N}} \max\limits_{k_1,k_2,k_1\leq k_2} \{k_2 - k_1, \text{ such that: } \bar{\mathcal{Z}}_\pi^{(j)}(k_1) = 1, \\ \bar{\mathcal{Z}}_\pi^{(j)}(k_2) = 1 \text{ and for all } k_1 < k < k_2, \bar{\mathcal{Z}}_\pi^{(j)}(k) = 0\}. \end{cases}$$

This optimization problems aims at minimizing the sum over j of the "maximal distances" between two successive output reading of system $S^{(j)}$.

8.3.6 A numerical example

In order to evaluate the proposed approach, the real-time implementation of the example of Section 8.2.4 is considered. Based on the assembler language, a handwritten assemble code of the feedback scheduler was written. Tasks execution was simulated using the toolbox TRUETIME [Andersson et al., 2005, Cervin et al., 2003], which allows the co-simulation of distributed real-time control systems, taking into account the effects of the execution of the control tasks and the data transmission on the controlled system dynamics.

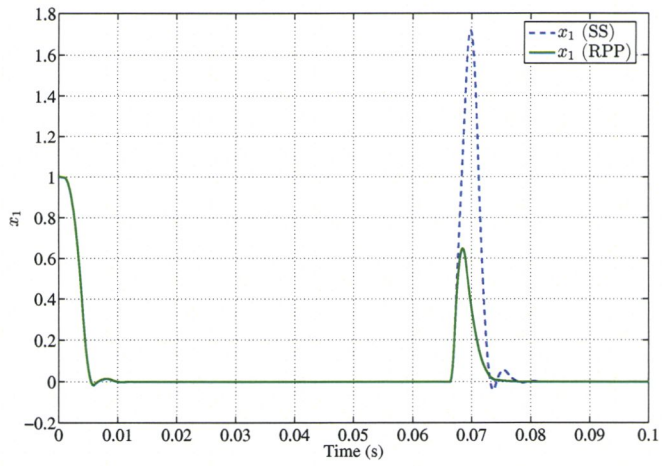
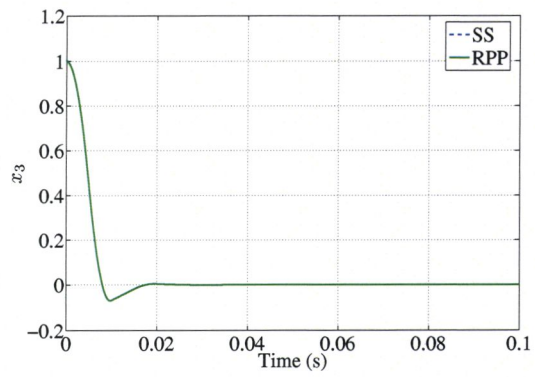


Figure 8.4: System responses – state x_1 ($x_1^{(1)}$) using the static scheduling (SS) and the RPP scheduling algorithms

Based on Table 8.1, the processor utilization of the control tasks, when scheduled using a basic sequencer, is equal to 32.57%. This means that aperiodic tasks as well as the feedback scheduler may have at most a utilization rate of 67.43%. Note that this implementation assumes that input operations are performed by independent hardware devices. An important issue concerns the execution overhead of the feedback scheduler. In order to be able to implement the proposed feedback scheduling algorithm, the worst case execution time of



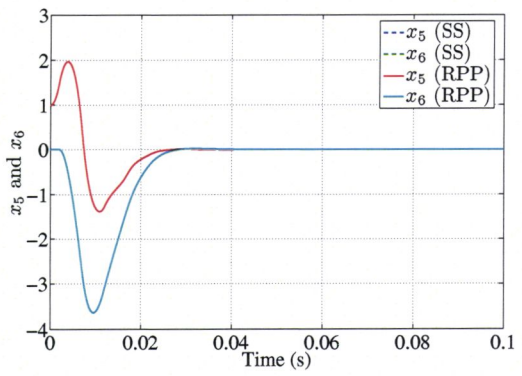


Figure 8.5: System responses – state x_3 ($x_1^{(2)}$) (left) and states x_5 ($x_1^{(3)}$) and x_6 ($x_2^{(3)}$) (right) using the static scheduling (SS) and the RPP scheduling algorithms

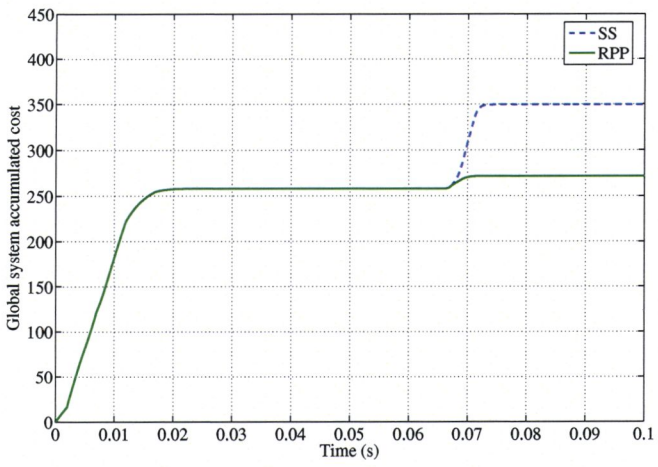


Figure 8.6: Accumulated continuous-time cost functions

the feedback scheduler must be less or equal to 717.06 μs or 980.08 μs, depending on the non-control tasks slots lengths (see Figure 8.2). To solve this problem, an implementation of the feedback scheduler (as described in Section 8.3.3) was performed. When the pointer is at position p, the feedback scheduler tries to answer the question:
Is position $\pi^*(p)$ better than position p?
where π^* is the sequence

$$\pi^* = (\pi^*(0), \pi^*(1), \pi^*(2), \pi^*(3)) = (1, 0, 3, 2).$$

The parameters $\varepsilon_x^{(1)} = \varepsilon_x^{(2)} = \varepsilon_x^{(3)} = 0.001$, $I_0 = \{2, 3\}$, $I_1 = \{1, 3\}$, $I_2 = \{2, 3\}$, $I_3 = \{1, 2\}$ and $\mathcal{P}_p = \{\pi^*(p)\}$, for $p \in \{0, \ldots, 3\}$ were chosen. In order to improve the responsiveness of the feedback scheduler, $\pi^*(p)$ was computed using the optimization problem (POS). Based on these choices, the worst-case execution times of the feedback scheduler (for a given pointer position) are given in Table 8.4. Note that in this example, testing over all the pointer positions based on the global system model (which corresponds to the use of the OPP algorithm) requires an execution time of 3393.8 μs.

Table 8.4: Worst-case execution time of the feedback scheduler (for each pointer position)

Pointer position	WCET (μs)
0	455.27
1	455.27
2	968.60
3	455.27

The global system responses corresponding to states x_1, x_3, x_5 and x_6 are depicted in Figures 8.4 and 8.5. The global system is started from the initial state $[1\ 0\ 1\ 0\ 1\ 0\ 0\ 0]^T$. The three systems $S^{(1)}$, $S^{(2)}$ and $S^{(3)}$ reach the practical stability region respectively at $t = 0.019$ s, $t = 0.026$ s and $t = 0.057$ s. At $t = 0.0663$ s, system $S^{(1)}$ is severely disturbed. The conditions of the "reactive pointer change" are fulfilled. The RPP algorithm reacts then at instant $t = 0.0665$ s to execute task $\tau_2^{(1)}$ instead of task $\tau^{(3)}$ (as illustrated in Figure 8.7). These changes allowed to react better to this disturbance and to improve the quality of control (as illustrated in Figure 8.6). It may be seen that when these disturbances occur, the execution time of the feedback scheduler increases (as illustrated in Figure 8.8). This increase is due to the additional test which is needed to guarantee that the "reactive pointer change" will improve the performances while maintains the stability.

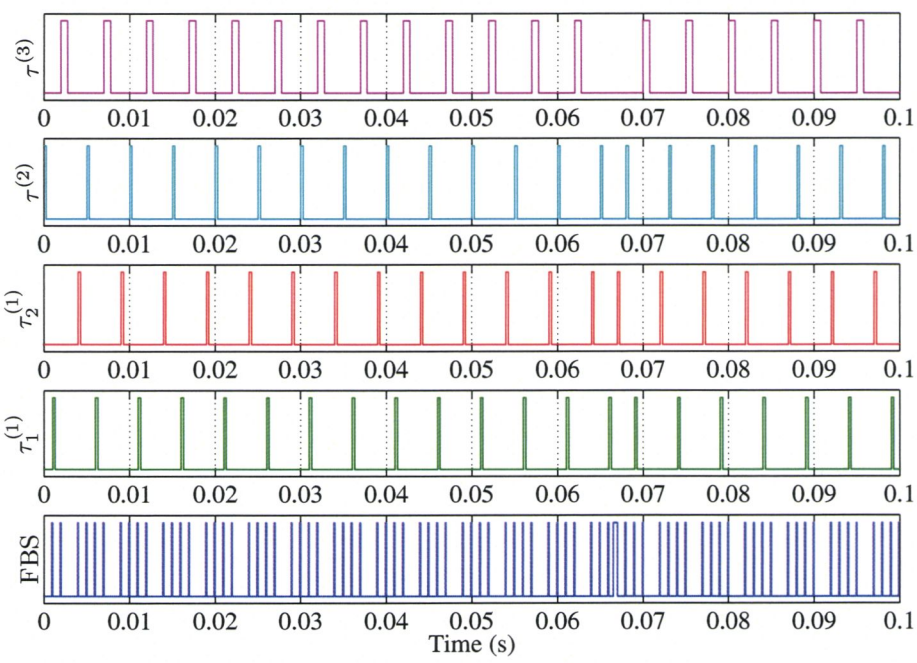


Figure 8.7: RPP schedule

Finally, the RPP scheduling algorithm is tested, in the case when all the systems are in the

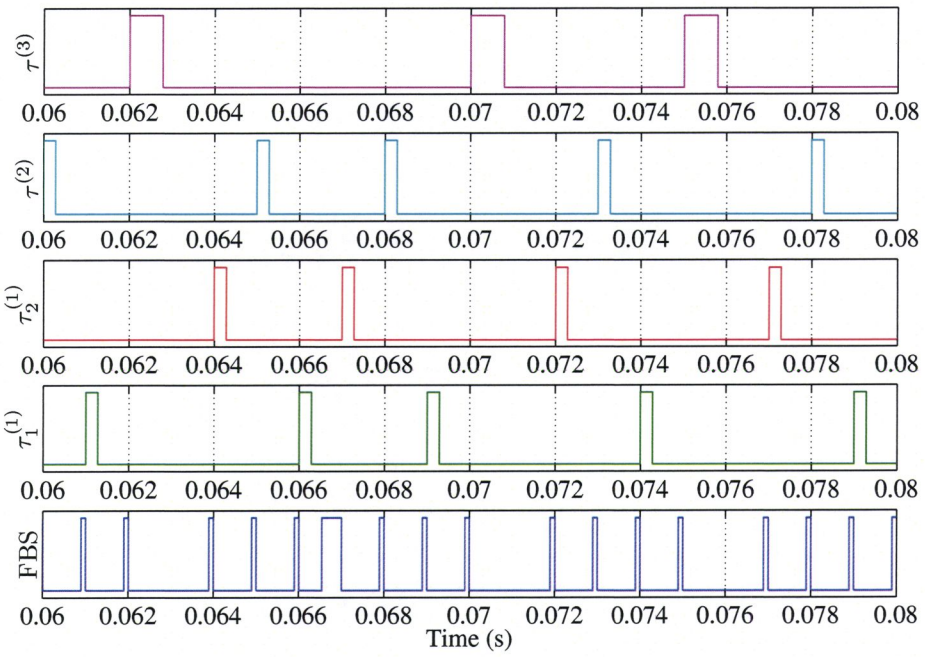


Figure 8.8: Zoom on the RPP schedule between instants 60 ms and 80 ms

practical equilibrium region, except one, which is perpetually disturbed. In the simulations depicted in Figure 8.9 (left), systems $S^{(1)}$ and $S^{(2)}$ remain in the equilibrium region whereas system $S^{(3)}$ is continuously disturbed with a band limited white noise characterized by a noise power of 0.1 and a correlation time of 1×10^{-4}. Simulation results indicate significant improvements in control performance. These improvements are due to the fact that the unused computing resources are allocated by the feedback scheduler to the control of system $S^{(3)}$, as illustrated in Figure 8.9 (right).

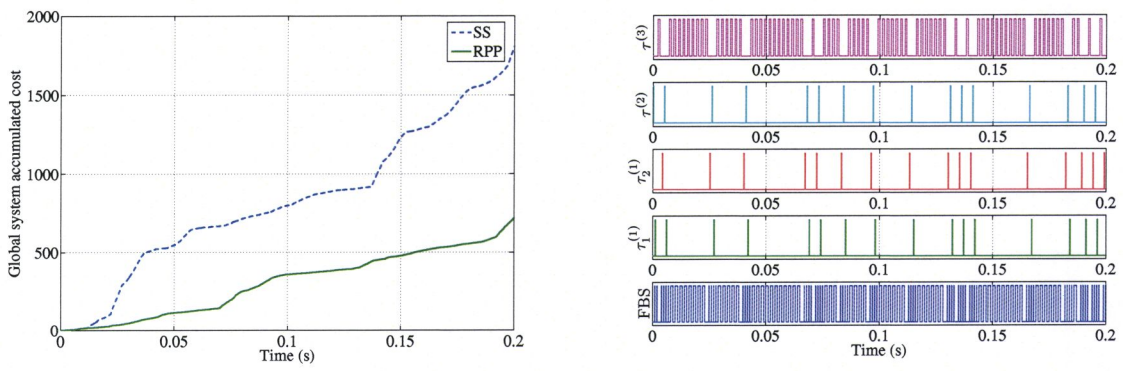


Figure 8.9: Accumulated cost functions (left) and tasks schedules (right) when systems $S^{(1)}$ and $S^{(2)}$ are in the equilibrium regions whereas system $S^{(3)}$ is continuously disturbed

8.4 Conclusion

In this chapter, a new approach for control tasks scheduling was proposed. This approach aims to improve control performance through a more efficient use of the available computational resources. First, an optimal integrated control and non-preemptive off-line scheduling problem is formulated. This problem is based on the $\mathcal{H}_2$ performance criterion to statically allocate the computing resources according to the intrinsic characteristics of the controlled systems. Using this approach, the "sampling periods" of control tasks are optimally chosen.

A plant state based feedback scheduling mechanism is then proposed, enabling to enhance the control performance with respect to the optimal off-line scheduling algorithm. This algorithm combines a more frequent reading of systems inputs with a state feedback based resource allocation to improve the control performance. The performance improvements resulting from the use of this algorithm as well as stability guaranties are proved mathematically and illustrated in a simulation examples. This algorithm allows to trade-off between control performance and real-time implementation constraints.

9 Conclusion

9.1 Summary

In this thesis, a new approach for the optimal control and scheduling of distributed embedded control system was proposed, focusing on two main problems: the scheduling of the messages from the controller to the distributed actuators and the scheduling of the control tasks. In this approach, the control performance, expressed as a quadratic cost function, was used to evaluate the optimality of the resource allocation. The proposed approach aims at improving the performance of the controlled dynamical systems that are challenged to operate with limited communication or computation resources, assuming that the available resources are sufficient to ensure their stability. To this end, novel on-line and off-line scheduling algorithms were introduced; allowing improving the control performance, by allocating the *available shared resources* according to the *needs* of the controlled dynamical systems. These algorithms were designed in order to ensure a tradeoff between their computational complexity and their optimality. Their effectiveness was illustrated through various simulation examples. Although the main motivation beyond this work was the problem of the control and resource allocation in distributed embedded control systems, the results of this thesis may be also of interest in all applications where actuations are costly to perform. Examples of such applications may be found in economical systems. The proposed approach was developed according to the following main steps.

We first considered the problem of communication resources limitations. We adopted a model, where control and communication resource allocation aspects are strongly dependent. By interpreting this model as a hybrid model, having two types of inputs: control inputs and scheduling inputs, we formalized the problem of the joint optimization of control and scheduling, for a quadratic performance criterion, as a mixed-integer quadratic program. This latter problem being NP-hard, the branch and bound method was used for its resolution. The study of the properties of the optimal schedule, through some selected numerical examples, has shown that it is strongly dependent on the system state. This dependence offers prospects for the improvement of the performances in terms of quality of con-

trol, by the use of on-line scheduling algorithms, which are based on the knowledge of the system state. However, this dependence shows that it is necessary to find other performance metrics for the synthesis of optimal off-line schedules.

We motivated the use of the $\mathcal{H}_2$ norm as a design criterion for obtaining optimal off-line schedules, only depending on the intrinsic characteristics of the system. The use of an optimal off-line schedule, described by a periodic communication sequence, has the advantage of simplifying the implementation and allows the use of periodic control gains, which represents a factorization of the control problem. We proposed a method for the joint control and off-line scheduling in the sense of the $\mathcal{H}_2$ criterion. We have shown that this problem may be decomposed into two sub-problems, which may be solved separately. The first sub-problem aims at determining the optimal off-line scheduling in the sense of the $\mathcal{H}_2$ criterion and may be solved using the branch and bound method. The second sub-problem aims at determining the optimal control gains and may be solved using the optimal periodic control theory. Techniques allowing improving the efficiency of this method were proposed.

We proposed an approach that allows determining on-line, at the same time, the optimal values of both control and scheduling, in the sense of a quadratic cost function. The implementation of this method, which is based on the resolution of the joint optimization of control and scheduling problem previously described, is however expensive on-line, due to the combinatorial explosion problem. For that reason, an on-line scheduling algorithm, called OPP was proposed. While being based on a pre-computed optimal off-line schedule, OPP makes it possible to allocate on-line the communication resources, based on the state of the controlled systems. It has been shown that under mild conditions, OPP ensures the asymptotic stability of the controlled systems and in all the situations a better or similar control performance compared to the basic static scheduling. The OPP scheduling algorithm was applied to the integrated control and scheduling of a distributed car suspension system.

We refined our considered model of a distributed embedded control system with communication constraints, to take into account the quantization aspects. Using this refined model, the exchange of information was modeled in terms of *bits*, in opposition to the previously considered model, where it was modeled in terms of *messages*. In general, increasing the sampling frequency improves the disturbance rejection abilities whereas increasing the quantization precision improves the steady state precision. However, when the bandwidth is limited, increasing the sampling frequency necessitates the reduction of the quantization precision and reciprocally. Based on these observations, we proposed the use of the model predictive control approach as means for automatically assigning the quantization precision and update rate of the control signals, in order to improve the control performance.

Finally, based on fine-grained model, taking into account both the execution of the control tasks and the scheduling algorithm, we generalized the previously described approaches to the problem of control tasks scheduling. We proposed a model of the single processor execution of control and non-control tasks. The co-design of the control and the real-time scheduling was performed in a two-step approach. In the first step, an optimal off-line schedule is derived, using the previously introduced approach of the optimal $\mathcal{H}_2$ scheduling, taking into account the particular constraints that are specific to the single processor execution of tasks. In the second step, an on-line plant state feedback scheduling approach is deployed. Its objective is to improve the control performance by quickly reacting to unexpected disturbances. Performance improvements as well as stability guarantees using this approach were proven. The proposed method is evaluated on a comprehensive implementation model, which was simulated using the tool TRUETIME.

9.2 Future work

The following research directions represent possible extensions to the work presented in the thesis:

Integrated control and scheduling under variable communication resources: In this thesis, the problem of the integrated control and scheduling under constant fixed communication resources, characterized by the fixed parameter b_w, was addressed. An interesting extension is the generalization of this problem to the networks with variable resources, which may be modeled by the time varying parameter b_w. Using this extension, two basic situations may be considered: the predictable variation and the unpredictable variation of b_w. Considering an unpredictable time varying parameter b_w would capture the problems of packet losses, delays and bandwidth limitations. The insights that were gained from studying the integrated control and scheduling problems in a deterministic setting may be helpful to study the more general situations of delays and information losses, because in both cases, the scheduling decision have to maximize the relevancy of the exchanged control information.

On-line sensors-to-controller scheduling: In the situations where the sensors of the control application are physically distributed, the problem of the existence of an on-line scheduling strategy of the sensors-to-controller link that is better than predefined off-line scheduling strategies is an open problem. To this problem, the possible on-line scheduling strategies may be classified into two categories: centralized decision-making strategies and distributed decision-making strategies. The deployment of strategies belonging to the first category would require that selected information from the distributed sensors is sent to a predefined node in the network that will have to decide, based on this information, which sensor node will be allowed to send over the network. The evaluation of the introduced communication overhead has to taken into account in the evaluation of these protocols. The strategies belonging to the second category would only use the local information to perform the scheduling decision, and would not consequently introduce an extra communication overhead. The most important issue concerns the evaluation of their optimality with regard to the whole distributed control application, and to compare it to centralized on-line scheduling schemes and to static off-line scheduling schemes. In both cases, the most important issue is to find an adequate performance criterion, usable on-line, which may quantify the impact to the measures scheduling decisions on the control performance.

Improving the efficiency of the $\mathcal{H}_2$ optimal scheduling: The improvement of the efficiency of the solving of the $\mathcal{H}_2$ optimal scheduling problem is an interesting issue. In this work, the branch and bound method was employed. The used branch and bound algorithm exploited the continuous relaxation method for finding the lower bounds of a given subproblem. The continuous relaxation is a general method for finding the lower bounds of mixed integer problems. The finding of methods for computing the lower bounds that are more specific to the considered $\mathcal{H}_2$ optimal scheduling problem, allowing a more efficient computation of tighter lower bounds, may considerably improve the efficiency of the solving of this problem.

Dynamic encoding/decoding scheme: In chapter 7, a fixed number of R_D bits were employed in order to encode the quantization precision of each component of the control command vector $u(k)$. All the R_D bits were used to encode a key specifying to the receiver how to decode the R_I bits of information. An important improvement may be obtained if dynamic encoding schemes are used, allowing using fewer bits for the decoding information and more bits for the control information. This may be related to the natural languages, where the most used words are the shorter. The design of appropriate dynamic encoding decoding schemes is an interesting issue, that may help improving, in the average, the quantization precision of control signals thanks to the reduction of the decoding information.

Extensions of the state feedback scheduler: The plant state based feedback scheduling approach of Chapter 8 was studied in the simple case where the control tasks are simply constant state feedback control laws, which may be considered among the less computationally expensive control laws. The measurement of the full state was assumed. A possible extension of this method consists on considering the case where only the outputs of the plant are measured, instead of the full state and where state observers are used. Another important extension consists on studying more computationally complex control laws, such as model predictive, hybrid or non-linear controllers. The issue that arises in this situation is the efficient computation of the predicted cost. The definition of appropriate approximation of the predicted cost, called *abstractions*, may be of interest. It is clear that the plant state based feedback scheduler will be more effective as the computational complexity of the scheduled tasks is greater than its own computational complexity.

Combining plant state and execution time feedback scheduling mechanisms: In this thesis, it was shown that the control performance is not a static measure that is only dependant on the sampling period of a given task, but that it is also dependent on the state of the controlled dynamic system. Intuitively, it is clear that increasing the sampling frequency of a system that is in the equilibrium does not improve its control performance, since the system does not *need* any additional computational resources. In practice, the control tasks may be implemented on non-deterministic platforms that are characterized by varying execution times. In many previous works, as previously mentioned, execution time measurement based feedback scheduling approaches were proposed in order to use more efficiently the available computational resources and to improve the control performance. However, these approaches were based on the assumption that the control performance is a convex function of the sampling period. These feedback scheduling approaches may be significantly improved if the execution time based feedback scheduling, which quantifies *the available resources* is combined with a plant state based feedback scheduling, which quantifies *the needed resources*. The architecture of such double loop feedback scheduler is an interesting issue.

Appendices

A

Expressions of matrices $\mathcal{A}$, $\mathcal{B}$, $\mathcal{F}$ and $\mathcal{G}$

Matrices $\mathcal{A}$ and $\mathcal{B}$ include both the equality and the inequality constraints of the problem, and may be written in the form

$$\mathcal{A} = \begin{bmatrix} \mathcal{A}_{eq} \\ -\mathcal{A}_{eq} \\ \mathcal{A}_{in} \end{bmatrix}$$

$$\mathcal{B} = \begin{bmatrix} \mathcal{B}_{eq} \\ -\mathcal{B}_{eq} \\ \mathcal{B}_{in} \end{bmatrix}.$$

It easy to see that the relation

$$\mathcal{A}_{eq}\mathcal{V} = \mathcal{B}_{eq}$$

is equivalent to

$$(\mathcal{A}_{eq}\mathcal{V} \leq \mathcal{B}_{eq}) \wedge (-\mathcal{A}_{eq}\mathcal{V} \leq -\mathcal{B}_{eq}).$$

Matrices $\mathcal{A}_{eq}$ and $\mathcal{B}_{eq}$ describe equalities (3.4a), (4.14) and impose to the scheduling sequence of the controller-to-actuators link to be maximal. Matrix $\mathcal{A}_{eq}$ is defined by

$$\mathcal{A}_{eq} = \begin{bmatrix} SC & 0_{N,n(N+1)} & 0_{N,mN} & 0_{N,mN} & 0_{N,mN} \\ 0_{n(N+1),mN} & ST & & 0_{n(N+1),mN} & 0_{n(N+1),mN} \\ 0_{mN,mN} & 0_{mN,n(N+1)} & UU & -I_{mN} & I_{mN} \end{bmatrix}$$

where

- SC is a $N \times mN$ matrix described by

$$SC = \begin{bmatrix} 1_{1,m} & 0_{1,m} & \cdots & 0_{1,m} \\ 0_{1,m} & 1_{1,m} & \cdots & 0_{1,m} \\ & & \ddots & \\ 0_{1,m} & 0_{1,m} & \cdots & 1_{1,m} \end{bmatrix}.$$

- ST is a $n(N+1) \times (n(N+1)+mN)$ matrix that is used to represent the constraint (3.4a) :

$$ST = \begin{bmatrix} I_n & 0_{n,n} & 0_{n,n} & \cdots & 0_{n,n} & 0_{n,m} & 0_{n,m} & \cdots & 0_{n,m} \\ -A & I_n & 0_{n,n} & \cdots & 0_{n,n} & -B & 0_{n,m} & \cdots & 0_{n,m} \\ 0_{n,n} & -A & I_n & \cdots & 0_{n,n} & 0_{n,m} & -B & \cdots & 0_{n,m} \\ & & \ddots & \ddots & & & & \ddots & \\ 0_{n,n} & 0_{n,n} & & -A & I_n & 0_{n,m} & 0_{n,m} & \cdots & -B \end{bmatrix}.$$

- UU is a $mN \times mN$ matrix described by

$$UU = \begin{bmatrix} I_m & 0_{m,m} & 0_{m,m} & \cdots & 0_{m,m} \\ -I_m & I_m & 0_{m,m} & \cdots & 0_{m,m} \\ 0_{m,m} & -I_m & I_m & \cdots & 0_{m,m} \\ & & \ddots & \ddots & \\ 0_{m,m} & 0_{m,m} & & -I_m & I_m \end{bmatrix}.$$

Matrix $\mathcal{B}_{eq}$ is defined by

$$\mathcal{B}_{eq} = \begin{bmatrix} b \times 1_{N,1} \\ x(0) \\ 0_{nN,1} \\ 0_{mN,1} \end{bmatrix}.$$

Let U and L the vectors defined by

$$U = \begin{bmatrix} U_1 \\ \vdots \\ U_m \end{bmatrix}$$

$$L = \begin{bmatrix} L_1 \\ \vdots \\ L_m \end{bmatrix}.$$

Matrices $\mathcal{A}_{in}$ and $\mathcal{B}_{in}$ describe the constraints defining the variables $\xi(k)$ and $o(k)$ ((4.15) and (4.17)). Matrix $\mathcal{A}_{in}$ is defined by

$$\mathcal{A}_{in} = \begin{bmatrix} -\mathrm{Diag}(U) & 0_{mN,n(N+1)} & 0_{mN,mN} & I_{mN} & 0_{mN,mN} \\ \mathrm{Diag}(L) & 0_{mN,n(N+1)} & 0_{mN,mN} & -I_{mN} & 0_{mN,mN} \\ -\mathrm{Diag}(L) & 0_{mN,n(N+1)} & -I_{mN} & I_{mN} & 0_{mN,mN} \\ \mathrm{Diag}(U) & 0_{mN,n(N+1)} & I_{mN} & -I_{mN} & 0_{mN,mN} \\ -\mathrm{Diag}(U) & 0_{mN,n(N+1)} & 0_{mN,mN} & 0_{mN,mN} & I_{mN} \\ \mathrm{Diag}(L) & 0_{mN,n(N+1)} & 0_{mN,mN} & 0_{mN,mN} & -I_{mN} \\ -\mathrm{Diag}(L) & 0_{mN,n(N+1)} & -UM & 0_{mN,mN} & I_{mN} \\ \mathrm{Diag}(U) & 0_{mN,n(N+1)} & UM & 0_{mN,mN} & -I_{mN} \end{bmatrix}$$

where

$$UM = -(UU - I_{mN}).$$

Matrix $\mathcal{B}_{in}$ is defined by

$$\mathcal{B}_{in} = \begin{bmatrix} 0_{m,1} \\ 0_{m,1} \\ -L \\ U \\ 0_{m,1} \\ 0_{m,1} \\ -L \\ U \end{bmatrix}.$$

Finally, matrices $\mathcal{F}$ and $\mathcal{G}$ are respectively defined by

$$\mathcal{F} = \begin{bmatrix} 0_{mN,mN} & 0_{mN,n(N+1)+mN} & 0_{mN,2mN} \\ 0_{n(N+1)+mN,mN} & \check{\mathcal{Q}} & 0_{n(N+1)+mN,2mN} \\ 0_{2mN,mN} & 0_{2mN,n(N+1)+mN} & 0_{2mN,2mN} \end{bmatrix}$$

and

$$\mathcal{G} = 0_{(4m+(n+1))N,1}$$

where

$$\check{\mathcal{Q}} = \begin{bmatrix} Q_1 & 0_{n,n} & \cdots & 0_{n,n} & 0_{n,n} & Q_{12} & 0_{n,m} & \cdots & 0_{n,m} \\ 0_{n,n} & Q_1 & \cdots & 0_{n,n} & 0_{n,n} & 0_{n,m} & Q_{12} & \cdots & 0_{n,m} \\ & & \ddots & & & & & \ddots & \\ 0_{n,n} & 0_{n,n} & \cdots & Q_1 & 0_{n,n} & 0_{n,m} & 0_{n,m} & \cdots & Q_{12} \\ 0_{n,n} & 0_{n,n} & \cdots & 0_{n,n} & Q_0 & 0_{n,m} & 0_{n,m} & \cdots & 0_{n,m} \\ Q_{12}^T & 0_{m,n} & \cdots & 0_{m,n} & 0_{m,n} & Q_2 & 0_{m,m} & \cdots & 0_{m,m} \\ 0_{m,n} & Q_{12}^T & \cdots & 0_{m,n} & 0_{m,n} & 0_{m,m} & Q_2 & \cdots & 0_{m,m} \\ & & \ddots & & & & & \ddots & \\ 0_{m,n} & 0_{m,n} & \cdots & Q_{12}^T & 0_{m,n} & 0_{m,m} & 0_{m,m} & \cdots & Q_2 \end{bmatrix}.$$

Bibliography

[Allen-Bradley, 2001] Allen-Bradley (2001). Compact logix system (catalog numbers 1769-l20 and 1769-l30) : user manual. Publication 1769-UM007C-EN-P.

[Andersson et al., 2005] Andersson, M., Henriksson, D., and Cervin, A. (2005). TRUETIME *1.3—Reference Manual*. Department of Automatic Control, Lund Institute of Technology, Sweden.

[Årzén and Cervin, 2005] Årzén, K.-E. and Cervin, A. (2005). Control and embedded computing: Survey of research directions. In *Proceedings of the 16th IFAC World Congress on Automatic Control*, Prague, Czech Republic.

[Årzén et al., 2000] Årzén, K.-E., Cervin, A., Eker, J., and Sha, L. (2000). An introduction to control and real-time scheduling co-design. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia.

[Åström and Wittenmark, 1997] Åström, K. J. and Wittenmark, B. (1997). *Computer-controlled systems: theory and design*. Prentice Hall.

[Attiya, 2004] Attiya, G. (2004). *Assignment of tasks on parallel and distributed computer systems*. PhD thesis, University of Marne-la-Vallée.

[Bajic and Bouard, 2002] Bajic, E. and Bouard, B. (2002). Réseau Profibus. In *Techniques de l'Ingénieur – Traité Informatique Industrielle*, volume S 8 160. Editions Techniques de l'Ingénieur.

[Bamieh and Boyd Pearson, 1992] Bamieh, B. and Boyd Pearson, J. (1992). The $\mathcal{H}_2$ problem for sampled-data systems. *Systems and Control Letters*, 19(1):1–12.

[Belanger et al., 2004] Belanger, G., Speyer, J., Ananyev, S., Chichka, D., and Carpenter, R. (2004). Decentralized control of satellite clusters under limited communication. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Providence, Rhode Island, USA.

[Bemporad and Morari, 1999] Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427.

[Ben Gaid and Çela, 2005] Ben Gaid, M.-M. and Çela, A. (2005). Model predictive control of systems with communication constraints. In *Proceedings of the 16th IFAC World Congress on Automatic Control*, Prague, Czech Republic.

[Ben Gaid and Çela, 2006] Ben Gaid, M.-M. and Çela, A. (2006). Trading quantization precision for sampling rates in networked systems with limited communication. In *Proceedings of the 45th IEEE Conference on Decision an Control*, San Diego, CA, USA.

[Ben Gaid et al., 2006a] Ben Gaid, M.-M., Çela, A., Diallo, S., Kocik, R., Hamouche, R., and Reama, A. (2006a). Performance evaluation of the distributed implementation of a car suspension system. In *In Proceedings of the IFAC Workshop on Programmable Devices and Embedded Systems*, Brno, Czech Republic.

[Ben Gaid et al., 2005] Ben Gaid, M.-M., Çela, A., and Hamam, Y. (2005). Optimal integrated control and scheduling of systems with communication constraints. In *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference*, Seville, Spain.

[Ben Gaid et al., 2006b] Ben Gaid, M.-M., Çela, A., and Hamam, Y. (2006b). Optimal integrated control and scheduling of networked control systems with communication constraints: Application to a car suspension system. *IEEE Transactions on Control Systems Technology*, 14(4):776–787.

[Ben Gaid et al., 2006c] Ben Gaid, M.-M., Çela, A., Hamam, Y., and Ionete, C. (2006c). Optimal scheduling of control tasks with state feedback resource allocation. In *Proceedings of the 2006 American Control Conference*, Minneapolis, Minnesota, USA.

[Ben Gaid et al., 2004] Ben Gaid, M.-M., Çela, A., and Kocik, R. (2004). Distributed control of a car suspension system. In *Proceedings of the 5th EUROSIM Congress on Modeling and Simulation*, Paris, France.

[Berlin and Gabriel, 1997] Berlin, A. and Gabriel, K. (1997). Distributed MEMS: New challenges for computation. *IEEE Computational Science and Engineering Magazine*, 4(1):12–16.

[Bittanti et al., 1991] Bittanti, S., Laub, A. J., and Willems, J., editors (1991). *The Riccati Equation*, chapter The periodic Riccati Equation. Springer-Verlag.

[Borrelli et al., 2005] Borrelli, F., Baotić, M., Bemporad, A., and Morari, M. (2005). Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41(10):1709–1721.

[Bouard, 2005] Bouard, B. (2005). Ethernet en tant que réseau de terrain : standard Profinet. In *Techniques de l'Ingénieur – Traité Informatique Industrielle*, volume S 8 162. Editions Techniques de l'Ingénieur.

[Branicky et al., 2002] Branicky, M., Phillips, S., and Zhang, W. (2002). Scheduling and feedback co-design for networked control systems. In *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, USA.

[Brockett, 1995] Brockett, R. (1995). Stabilization of motor networks. In *Proceedings of the 34th IEEE Conference on Decision and Control*, New Orleans, Los Angeles, USA.

[Brockett, 1997] Brockett, R. (1997). Minimum attention control. In *Proceedings of the 36th IEEE Conference on Decision and Control*, San Diego, California, USA.

[Brockett and Liberzon, 2000] Brockett, R. W. and Liberzon, D. (2000). Quantized feedback stabilization of linear systems. *IEEE Transactions on Automatic Control*, 45(7):1279–1289.

[Burns and Wellings, 2001] Burns, A. and Wellings, A. (2001). *Real-Time Systems and Programming Languages*. Addison-Wesley.

[Buttazzo et al., 2004] Buttazzo, G., Velasco, M., Martí, P., and Fohler, G. (2004). Managing quality-of-control performance under overload conditions. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, Catania, Italy.

[Buttazzo, 1997] Buttazzo, G. C. (1997). *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Prentice Hall.

[Buttazzo et al., 2002] Buttazzo, G. C., Lipari, G., Caccamo, M., and Abeni, L. (2002). Elastic scheduling for flexible workload management. *IEEE Transactions on Computers*, 51(3):289–302.

[Cervin, 2003] Cervin, A. (2003). *Integrated Control and Real-Time Scheduling*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Sweden.

[Cervin et al., 2002] Cervin, A., Eker, J., Bernhardsson, B., and Årzén, K.-E. (2002). Feedback-feedforward scheduling of control tasks. *Real-Time Systems*, 23(1):25–53.

[Cervin et al., 2003] Cervin, A., Henriksson, D., Lincoln, B., Eker, J., and Årzén, K.-E. (2003). How does control timing affect performance? *IEEE Control Systems Magazine*, 23(3):16–30.

[Chalasani, 1986] Chalasani, R. M. (1986). Ride performance potential of active suspension systems - Part II: Comprehensive analysis based on a full-car model. In *Proceedings of the Symposium on Simulation and Control of Ground Vehicles and Transportation Systems, ASME AMD*, Anaheim, CA, USA.

[Chen and Francis, 1991] Chen, T. and Francis, B. A. (1991). $\mathcal{H}_2$-optimal sampled-data control. *IEEE Transactions on Automatic Control*, 36(4):387–397.

[Chen and Francis, 1995] Chen, T. and Francis, B. A. (1995). *Optimal Sampled-Data Control Systems*. Springer-Verlag.

[ControlNet International, 1999] ControlNet International (1999). ControlNet specification, release 2.0, including errata 2.

[Dakin, 1965] Dakin, R. J. (1965). A tree search algorithm for mixed integer programming problems. *Computer Journal*, 8(3):250–255.

[Decotigny, 2003] Decotigny, D. (2003). *Une infrastructure de simulation modulaire pour l'évaluation de performances de systèmes temps-réel*. PhD thesis, Université de Rennes 1.

[Delchamps, 1990] Delchamps, D. F. (1990). Stabilizing a linear system with quantized state feedback. *IEEE Transactions on Automatic Control*, 35(8):916–924.

[Dhall, 1978] Dhall, S. K. ad Liu, C. L. (1978). On a real-time scheduling problem. *Operations Research*, 26(1):127–140.

[Di Natale, 2000] Di Natale, M. (2000). Scheduling the CAN bus with earliest deadline techniques. In *Proceedings of the 21st IEEE Real-Time Systems Symposium*, Orlando, Floria, USA.

[Dorf et al., 1962] Dorf, R., Farren, M., and Phillips, C. (1962). Adaptive sampling frequency for sampled-data control systems. *IRE Transactions on Automatic Control*, 7(1):38–47.

[Doyle et al., 1989] Doyle, J. C., Glover, K., Khargonekar, P. P., and Francis, B. (1989). State-space solutions to the standard H_2 and H_∞ control problems. *IEEE Transactions on Automatic Control*, 34(8):831–847.

[Eker, 1999] Eker, J. (1999). *Flexible Embedded Control Systems. Design and Implementation*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Sweden.

[Elia and Mitter, 2001] Elia, N. and Mitter, S. (2001). Stabilization of linear systems with limited information. *IEEE Transactions on Automatic Control*, 46(9):1384–1400.

[Fagnani and Zampieri, 2004] Fagnani, F. and Zampieri, S. (2004). Quantized stabilization of linear systems: complexity versus performance. *IEEE Transactions on Automatic Control*, 49(9):1534–1548.

[Fletcher and Leyffer, 1998] Fletcher, R. and Leyffer, S. (1998). Numerical experience with lower bounds for MIQP branch-and-bound. *SIAM Journal on Optimization*, 8(2):604–616.

[Flippo and Rinnoy Kan, 1990] Flippo, O. E. and Rinnoy Kan, A. H. G. (1990). A note on benders decomposition in mixed-integer quadratic programming. *Operations Research Letters*, 9(2):81–83.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman.

[Goodwin et al., 2004] Goodwin, G. C., Haimovich, H., Quevedo, D. E., and Welsh, J. S. (2004). A moving horizon approach to networked control systems design. *IEEE Transactions on Automatic Control*, 49(9):1562–1572.

[Grandpierre et al., 1999] Grandpierre, T., Lavarenne, C., and Sorel, Y. (1999). Optimized rapid prototyping for real-time embedded heterogeneous multiprocessors. In *Proceedings of 7th International Workshop on Hardware/Software Co-Design*, Rome, Italy.

[Gupta, 1963a] Gupta, S. C. (1963a). Adaptive gain and adaptive sampfing sampled-data systems. In *Proceedings of the IEEE Winter General Meeting*, New York, USA.

[Gupta, 1963b] Gupta, S. C. (1963b). Increasing the sampling efficiency for a control system. *IEEE Transactions on Automatic Control*, 8(3):263–264.

[Henriksson, 2006] Henriksson, D. (2006). *Resource-Constrained Embedded Control and Computing Systems*. PhD thesis, Department of Automatic Control, Lund University, Sweden.

[Henriksson and Cervin, 2005] Henriksson, D. and Cervin, A. (2005). Optimal on-line sampling period assignment for real-time control tasks based on plant state information. In *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference*, Seville, Spain.

[Hespanha and Xu, 2004a] Hespanha, J. P. and Xu, Y. (2004a). Communication logics for networked control systems. In *Proceedings of the 2004 American Control Conference*, Boston, Massachusetts, USA.

[Hespanha and Xu, 2004b] Hespanha, J. P. and Xu, Y. (2004b). Optimal communication logics for networked control systems. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas.

[Hristu, 1999] Hristu, D. (1999). *Optimal control with limited communication*. PhD thesis, Division of Engineering and Applied Sciences, Harvard University.

[Hristu and Morgansen, 1999] Hristu, D. and Morgansen, K. (1999). Limited communication control. *System and Control Letters*, 37(4):193–205.

[Hsia, 1972] Hsia, T. C. (1972). Comparisons of adaptive sampling control laws. *IEEE Transactions on Automatic Control*, 17(6):830–831.

[Hsia, 1974] Hsia, T. C. (1974). Analytic design of adaptive sampling control law in sampled-data systems. *IEEE Transactions on Automatic Control*, 19(1):39–42.

[Ionete and Çela, 2006] Ionete, C. and Çela, A. (2006). Structural properties and stabilization of NCS with medium access constraints. In *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, California, USA.

[ISO, 1993] ISO (1993). Road vehicles: Interchange of digital information – Controller Area Network (CAN) for high-speed communication. ISO Standard-11898.

[Johansen and Grancharova, 2002] Johansen, T. A. and Grancharova, A. (2002). Approximate explicit model predictive control implemented via orthogonal search tree partitioning. In *Proceedings of the 15th IFAC World Congress*, Barcelona, Spain.

[Joseph and Pandya, 1986] Joseph, M. and Pandya, P. (1986). Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395.

[Kalman et al., 1963] Kalman, R. E., Ho, B., and Narendra, K. (1963). Controllability of linear dynamical systems. *Contributions to Differential Equations*, 1:188–213.

[Khargonekar and Sivashankar, 1991] Khargonekar, P. and Sivashankar, N. (1991). $\mathcal{H}_2$ optimal control for sampled-data systems. *Systems and Control Letters*, 17(6):424–436.

[Kocik et al., 2005] Kocik, R., Ben Gaid, M.-M., and Hamouche, R. (2005). Software implementation simulation to improve control laws design. In *In Proceedings of the First European Congress: Sensors & Actuators for Advanced Automotive Applications*, Paris, France.

[Kocik and Sorel, 1998] Kocik, R. and Sorel, Y. (1998). A methodology to design and prototype optimized embedded robotic systems. In *Proceedings of 2nd IMACS International Multiconference, CESA'98*, Hammamet, Tunisia.

[Kopetz and Ochsenreiter, 1987] Kopetz, H. and Ochsenreiter, W. (1987). Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, 36(8):933–940.

[Krishna and Shin, 1997] Krishna, C. M. and Shin, K. G. (1997). *Real-Time Systems*. McGraw-Hill.

[Kuschel et al., 2006] Kuschel, M., Kremer, P., Hirche, S., and Buss, M. (2006). Lossy data reduction methods for haptic telepresence systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Orlando, Floria, USA.

[Kwok and Ahmad, 1999] Kwok, Y.-K. and Ahmad, I. (1999). Benchmarking and comparison of the task graph scheduling algorithms. *Journal of Parallel and Distributed Computing*, 59(3):381–422.

[Land and Doig, 1960] Land, A. H. and Doig, A. G. (1960). An automatic method for solving discrete programming problems. *Econometrica*, 28(3):497–520.

[Lazimy, 1985] Lazimy, R. (1985). Improved algorithm for mixed-integer quadratic programs and a computational study. *Mathematical Programming*, 12:100–113.

[Lian et al., 2001] Lian, F.-L., Moyne, J., and Tilbury, D. (2001). Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet. *IEEE Control Systems Magazine*, 21(1):66–83.

[Lincoln and Bernhardsson, 2002] Lincoln, B. and Bernhardsson, B. (2002). LQR optimization of linear system switching. *IEEE Transactions on Automatic Control*, 47(10):1701–1705.

[Liu and Layland, 1973] Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61.

[Liu, 2000] Liu, J. W. S. (2000). *Real-Time Systems*. Prentice Hall.

[Liu et al., 2000] Liu, X., Sha, L., Caccamo, M., and Buttazzo, G. (2000). Online control optimization using load driven scheduling. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia.

[Lu et al., 2002] Lu, C., Stankovic, J., Tao, G., and Son, S. (2002). Feedback control real-time scheduling: Framework, modeling and algorithms. *Special issue of Real-Time Systems Journal on Control-Theoretic Approaches to Real-Time Computing*, 23(1/2):85–126.

[Lu et al., 2003] Lu, L., Xie, L., and Fu, M. (2003). Optimal control of networked systems with limited communication: a combined heuristic and convex optimization approach. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, Hawaii, USA.

[MAP/TOP Users Group, 1988] MAP/TOP Users Group (1988). Manufacturing automation protocol specification - version 3.0.

[Martí, 2002] Martí, P. (2002). *Analysis and Design of Real-Time Control Systems with Varying Control Timing Constraints*. PhD thesis, Technical University of Catalonia.

[Martí et al., 2002] Martí, P., Fuertes, J., Fohler, G., and Ramamritham, K. (2002). Improving quality-of-control using flexible timing constraint : Metric and scheduling issues. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium*, Austin, Texas, USA.

[Martí et al., 2001] Martí, P., Fuertes, J., Ramamritham, K., and Fohler, G. (2001). Jitter compensation for real-time control systems. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, London, England.

[Martí et al., 2004] Martí, P., Lin, C., Brandt, S., Velasco, M., and Fuertes, J. (2004). Optimal state feedback based resource allocation for resource-constrained control tasks. In *Proceedings of the 25th IEEE Real-Time Systems Symposium*, Lisbon, Portugal.

[Mitchell and McDaniel Jr., 1969] Mitchell, J. R. and McDaniel Jr., W. L. (1969). Sensitivity of discrete systems to variation of sampling interval. *IEEE Transactions on Automatic Control*, 14(2):200–201.

[Montestruque and Antsaklis, 2003] Montestruque, L. and Antsaklis, P. (2003). On the model-based control of networked systems. *Automatica*, 39(10):1837–1843.

[Montestruque and Antsaklis, 2004] Montestruque, L. and Antsaklis, P. (2004). On the model-based control of networked systems. *IEEE Transactions on Automatic Control*, 49(9):1562–1572.

[Murray et al., 2003] Murray, R., Åström, K., Boyd, S., Brockett, R., and Stein, G. (2003). Future directions in control in an information-rich world. *IEEE Control Systems Magazine*, 23(2):20–33.

[Nair and Evans, 2000] Nair, G. N. and Evans, R. J. (2000). Stabilization with data-rate-limited feedback: tightest attainable bounds. *Systems and Control Letters*, 41(1):49–56.

[Nešic and Teel, 2004] Nešic, D. and Teel, A. R. (2004). Input-output stability properties of networked control systems. *IEEE Transaction on Automatic Control*, 49(10):1650–1667.

[Nilsson, 1998] Nilsson, J. (1998). *Real-Time Control Systems with Delays*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Sweden.

[Palopoli, 2002] Palopoli, L. (2002). *Design of Embedded Control Systems under real-time scheduling constraints*. PhD thesis, ReTiS Lab - Scuola Supoeriore S. Anna, Pisa Italy.

[Palopoli et al., 2002a] Palopoli, L., Bicchi, A., and Sangiovanni Vincentelli, A. (2002a). Numerically efficient control of systems with communication constraints. In *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, USA.

[Palopoli et al., 2005] Palopoli, L., Pinello, C., Bicchi, A., and Sangiovanni-Vincentelli, A. (2005). Maximizing the stability radius of a set of systems under real-time scheduling constraints. *IEEE Transactions on Automatic Control*, 50(11):1790–1795.

[Palopoli et al., 2002b] Palopoli, L., Pinello, C., Sangiovanni-Vincentelli, A. L., El-Ghaoui, L., and Bicchi, A. (2002b). Synthesis of robust control systems under resource constraints. In Greenstreet, M. and Tomlin, C., editors, *Hybrid Systems: Computation and Control*, volume LNCS 2289 of *Lecture Notes in Computer Science*, pages 337–350. Springer-Verlag, Heidelberg, Germany.

[Quazi and Konrad, 1982] Quazi, A. and Konrad, W. (1982). Underwater acoustic communications. *IEEE Communications Magazine*, 20(2):24–30.

[Rachid and Collet, 2000] Rachid, A. and Collet, F. (2000). Bus CAN. In *Techniques de l'Ingénieur – Traité Informatique Industrielle*, volume S 8 140. Editions Techniques de l'Ingénieur.

[Rehbinder and Sanfridson, 2000] Rehbinder, H. and Sanfridson, M. (2000). Integration of off-line scheduling and optimal control. In *Proceedings of the 12th Euromicro Conference on Real-Time Systems*, Stockholm, Sweden.

[Rehbinder and Sanfridson, 2004] Rehbinder, H. and Sanfridson, M. (2004). Scheduling of a limited communication channel for optimal control. *Automatica*, 40(3):491–500.

[Rettig and von Stryk, 2001] Rettig, U. and von Stryk, O. (2001). Numerical optimal control strategies for semi-active suspension with electrorheological fluid dampers. In *Proceedings of the workshop: Fast solution of discretized optimization problems*. International Series on Numerical Mathematics (Birkhäuser).

[Robert et al., 2005] Robert, D., Sename, O., and Simon, D. (2005). Sampling period dependent rst controller used in control/scheduling co-design. In *Proceedings of the 16th IFAC World Congress*, Prague, Czech republic.

[Rugh, 1996] Rugh, W. J. (1996). *Linear System Theory*. Prentice Hall.

[Seto et al., 1996] Seto, D., Lehoczky, J. P., Sha, L., and Shin, K. G. (1996). On task schedulability in real-time control systems. In *Proceedings of the 17th IEEE Real-Time Systems Symposium*, New York, USA.

[Sha et al., 2004] Sha, L., Abdelzaher, T., Årzén, K.-E., Cervin, A., Baker, T., Burns, A., Buttazzo, G., Caccamo, M., Lehoczky, J., and Mok, A. K. (2004). Real-time scheduling theory: A historical perspective. *Real-Time Systems*, 28(2–3):101–155.

[Simon et al., 2005] Simon, D., Robert, D., and Sename, O. (2005). Robust control / scheduling co-design: application to robot control. In *Proceedings of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Francisco, USA.

[Sinopoli et al., 2003] Sinopoli, B., Sharp, C., Schenato, L., Schaffert, S., and Sastry, S. S. (2003). Distributed control applications within sensor networks. *Proceedings of the IEEE*, 91(8):1235–1246.

[Smith, 1971] Smith, M. J. (1971). An evaluation of adaptive sampling. *IEEE Transactions on Automatic Control*, 16(3):282–284.

[Sorel, 2004] Sorel, Y. (2004). Syndex: System-level cad software for optimizing distributed real-time embedded systems. *Journal ERCIM News*, 59:68–69.

[Sozer et al., 2000] Sozer, E., Stojanovic, M., and Proakis, J. (2000). Underwater acoustic networks. *IEEE Journal of Oceanic Engineering*, 25(1):72–83.

[Speranzon et al., 2004] Speranzon, A., Silva, J., de Sousa, J., and Johansson, K. H. (2004). On collaborative optimization and communication for a team of autonomous underwater vehicles. In *Proceedings of Reglermöte*, Göteborg , Sweden.

[Stojanovic, 1996] Stojanovic, M. (1996). Recent advances in high-speed underwater acoustic communications. *IEEE Journal of Oceanic Engineering*, 21(2):125–136.

[Tatikonda and Mitter, 2004] Tatikonda, S. and Mitter, S. (2004). Control under communication constraints. *IEEE Transactions on Automatic Control*, 49(7):1056–1068.

[Thomesse, 1998] Thomesse, J. P. (1998). A review of the fieldbusses. *Annual Reviews in Control*, 22:35–45.

[Tindell and Burns, 1994] Tindell, K. and Burns, A. (1994). Guaranteeing message latencies on control area network (CAN). In *Proceedings of the 1st International CAN Conference*, Mainz, Germany.

[Tindell et al., 1995] Tindell, K., Burns, A., and Wellings, A. J. (1995). Calculating controller area network (CAN) message response times. *Control Engineering Practice*, 3(8):1163–1169.

[Tomovic and Bekey, 1966a] Tomovic, R. and Bekey, G. (1966a). Adaptive sampling based on amplitude sensitivity. *IEEE Transactions on Automatic Control*, 11(2):282–284.

[Tomovic and Bekey, 1966b] Tomovic, R. and Bekey, G. (1966b). Sensitivity of discrete systems to variation of sampling interval. *IEEE Transactions on Automatic Control*, 11(2):284–287.

[Tovar and Vasques, 1999] Tovar, E. and Vasques, F. (1999). Cycle time properties of the profibus timed-token protocol. *Computer Communications*, 22:1206–1216.

[TTTech, 2003] TTTech (2003). Time-triggered protocol TTP/C - High-level specification document - Protocol version 1.1.

[Walsh et al., 2002] Walsh, G., Beldiman, O., and Bushnell, L. (2002). Error encoding algorithms for networked control systems. *Automatica*, 38(2):261–267.

[Walsh and Ye, 2001] Walsh, G. and Ye, H. (2001). Scheduling of networked control systems. *IEEE Control Systems Magazine*, 21(1):57–65.

[Walsh et al., 2001] Walsh, G., Ye, H., and Bushnell, L. (2001). Asymptotic behavior of nonlinear networked control systems. *IEEE Transactions on Automatic Control*, 46(7):1093–1097.

[Wong and Brockett, 1997] Wong, W. S. and Brockett, R. W. (1997). Systems with finite communication bandwidth constraints – Part I : State estimation problems. *IEEE Transactions on Automatic Control*, 42(9):1294–1299.

[Wong and Brockett, 1999] Wong, W. S. and Brockett, R. W. (1999). Systems with finite communication bandwidth constraints – Part II : Stabilization with limited information feedback. *IEEE Transactions on Automatic Control*, 44(5):1049–1053.

[Xia and Sun, 2006] Xia, F. and Sun, Y. (2006). Control-scheduling codesign: A perspective on integrating control and computing. To appear in Journal of Dynamics of Continuous, Discrete and Impulsive Systems, Series B.

[Xie et al., 2001] Xie, L., Zhou, H., and Zhang, C. (2001). H_2 optimal deconvolution of periodic IIR channels: an LMI approach. In *Proceedings of the 6th International Symposium on Signal Processing and its Applications*, Kuala-Lumpur, Malaysia.

[Yang and Gerasoulis, 1993] Yang, T. and Gerasoulis, A. (1993). List scheduling with and without communication delays. *Parallel Computing*, 19(12):1321–1344.

[Ye, 1993] Ye, Y. (1993). A fully polynomial-time approximation algorithm for computing a stationary point of the general linear complementarity problem. *Mathematics of Operations Research*, 18(2):334–345.

[Yook et al., 2002] Yook, J., Tilbury, D., and Soparkar, N. (2002). Trading computation for bandwidth: Reducing communication in distributed control systems using state estimators. *IEEE Transactions on Control Systems Technology*, 10(4):503–518.

[Zhang and Hristu-Varsakelis, 2005] Zhang, L. and Hristu-Varsakelis, D. (2005). Stabilization of networked control systems: Communication and controller co-design. In *Proceedings of the 16th IFAC World Congress*, Prague, Czech Republic.

[Zhou et al., 2002] Zhou, H., Xie, L., and Zhang, C. (2002). A direct approach to H_2 optimal deconvolution of periodic digital channels. *IEEE Transactions on Signal Processing*, 50(7):1685–1698.

[Zimmermann, 1980] Zimmermann, H. (1980). OSI reference model - The ISO model of architecture for open system interconnection. *IEEE Transactions on Communications*, 28(4):425–432.