HAL
open science

Optimal Real-Time Scheduling of Control Tasks with State Feedback Resource Allocation

Mohamed El Mongi Ben Gaid, Arben Cela, Yskandar Hamam

To cite this version:

Mohamed El Mongi Ben Gaid, Arben Cela, Yskandar Hamam. Optimal Real-Time Scheduling of Control Tasks with State Feedback Resource Allocation. IEEE Transactions on Control Systems Technology, 2009, 17 (2), pp.309-326. 10.1109/TCST.2008.924566 . inria-00364947

HAL Id: inria-00364947

https://inria.hal.science/inria-00364947

Submitted on 27 Feb 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Optimal Real-Time Scheduling of Control Tasks with State Feedback Resource Allocation

Mohamed El Mongi Ben Gaid, Arben Çela and Yskandar Hamam

***Abstract*— This paper proposes a new approach for the optimal integrated control and real-time scheduling of control tasks. First, the problem of the optimal integrated control and non-preemptive off-line scheduling of control tasks in the sense of the $\mathcal{H}_2$ performance criterion is addressed. It is shown that this problem may be decomposed into two sub-problems. The first sub-problem aims at finding the optimal non-preemptive off-line schedule, and may be solved using the branch and bound method. The second sub-problem uses the lifting technique to determine the optimal control gains, based on the solution of the first sub-problem. Second, an efficient on-line scheduling algorithm is proposed. This algorithm, called Reactive Pointer Placement (RPP) scheduling algorithm, uses the plant state information to dispatch the computational resources in a way that improves control performance. Control performance improvements as well as stability guarantees are formally proven. Finally, simulations as well as experimental results are presented in order to illustrate the effectiveness of the proposed approach.**

***Index Terms*— Control and scheduling co-design, embedded systems, optimal control, real-time scheduling**

I. Introduction

In computer systems, the scheduler is the entity that is responsible of the allocation of the computational resources. Consequently, using efficiently these resources amounts to designing appropriate scheduling algorithms. The need for an efficient use of the computational resources comes from cost pressure, which requires the realization of system functionalities with minimum resources. Traditionally, control design and real-time scheduling design have been decoupled in the design processes of embedded control applications. This separation of concerns allowed each community to focus on specific problems, and led to the spectacular development that we are familiar with today [1]. However, this separation of concerns relies on the fact that these two domains use very simplified models of their interface with each other. In fact, as pointed out in [2], control designers assume that the implementation is able to provide an equidistant sampling and actuation, and to ensure a fixed input output latency as well as an infinite quantization precision. Real-time designers on the other hand see the control loop as a periodic task with a hard deadline.

In both control and real-time scheduling theories, the notion of "instantaneous computational needs" of a control application is not defined. Computation and communication resources are consequently allocated according to the "worst case needs" of these applications. This corresponds to the use of periodic sampling, which on the one hand simplifies the control design problem, but on the other hand leads to an unnecessary usage of some computational resources. In fact, a control task that is close to the equilibrium needs less computing resources than another one which is severely disturbed [3]–[5]. Similarly, the real-time scheduling design of control tasks is based on their worst-case execution time. As illustrated in [6], 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. Our opinion is that 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 [1], [6]–[8].

In this paper, a new approach for the co-design of control and real-time scheduling is proposed. In the first part, we motivate the use of the $\mathcal{H}_2$ performance index for the problem of the optimal integrated control and non-preemptive off-line scheduling of control tasks. A new approach for solving this problem is proposed. This approach decomposes the problem into two sub-problems. The first sub-problem aims at finding the optimal non-preemptive off-line schedule, and may be solved using the branch and bound method. The second sub-problem uses the lifting technique to determine the optimal control gains, based on the solution of the first sub-problem. We illustrate how existing knowledge from the considered control systems models may be used to considerably reduce the time needed to find the optimal solution, fully taking advantage of the use of the branch and bound method. In the second part, a plant state feedback scheduling algorithm, called reactive pointer placement (RPP) scheduling algorithm is proposed. Its objective is to improve the control performance by quickly reacting to unexpected disturbances. Performance improvements as well as stability guarantees using the RPP algorithm are formally proven and then illustrated on a comprehensive implementation model, which was simulated using the tool TrueTime [9]. Finally, the RPP algorithm is implemented on an embedded processor in order to achieve the concurrent real-time speed regulation of two DC motors. This paper extends and generalizes the preliminary results exposed in [10].

This work was supported and conducted at the Embedded Systems Department, ESIEE Paris, University of Paris East

M-M. Ben Gaid is with the Technology, Computer Science, and Applied Mathematics Division, Institut Français du Pétrole, Rueil-Malmaison, FRANCE (e-mail:mongi.ben-gaid@ifp.fr)

A. Çela is with the Embedded Systems Department, ESIEE Paris, University of Paris East, Noisy-Le-Grand, FRANCE (e-mail: a.cela@esiee.fr)

Y. Hamam is with A2SI Laboratory, ESIEE Paris, University of Paris East, Noisy-Le-Grand, FRANCE and with LISIV Laboratory, UVSQ, Velizy, FRANCE and with F'SATIE, Tshwane University of Technology, Pretoria, SOUTH AFRICA (e-mail: HamamA@tut.ac.za)

This paper is organized as follows. After a review of the related work in Section II, the formulation and solving of the optimal control and non-preemptive off-line scheduling according to the $\mathcal{H}_2$ performance criterion is addressed in Section III. In Section IV, the RPP scheduling algorithm is introduced, described and evaluated using simulation. Finally, in Section V, the experimental evaluation of the proposed approach is presented.

II. Related work

The related work may be classified into four categories:

1) Adaptive sampling : The analysis of asynchronously sampled systems was undertaken at the end of the fifties. The researches, which were performed at the sixties and at the beginning of the seventies, focused on SISO systems. To the best of our knowledge, the first adaptive sampling method was proposed by Dorf *et al.* [11]. 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 [12]–[14]. The evaluation of these approaches and their comparison to the periodic sampling was performed in [15]. 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 [11] and [14] are closely related, Hsia proposed a generic approach allowing to derive adaptive sampling laws [16].

2) Optimal control and monoprocessor scheduling : The problem of the optimal selection of control tasks periods subject to schedulability constraints was addressed in [17]. Assuming that the discrete-time control laws are designed in the continuous-time domain and then discretized, the notion of Performance Index (PI) was introduced. The Performance Index quantifies the performance of the discrete 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 [18]). 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 [19], 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 [20]. The problem of the optimal control and scheduling in the sense of LQG was introduced and developed in [21]. The proposed method allows pruning of the search tree in order to reduce the combinatoric explosion.

3) Scheduling of control tasks in environments with variable computing workload : It is well-known that worst-case analysis techniques [22] 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 [6], [23] and the elastic task model [24].

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” which 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 [7].

In the elastic task model [24], a periodic task set containing N tasks may be seen as a sequence of 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 [25], 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 [17] 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. [26] generalized this approach to take into account the degradations that may occur to the control systems 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 approaches are mainly based on the assumption that control performance is a convex function of the sampling period. In reality, as illustrated in [3], [4], the quality of control is also dependent on the dynamical state of the controlled system (in equilibrium, in transient state). In this paper, instead of using feedback from execution times measures, the plant state information is used to dispatch the available computing resources.

4) Scheduling of control tasks based on plant state information : The idea of the dynamical allocation of processor resources as a function of the plant state was proposed in [27], assuming that the performance index of a task controlling a system at a given state is proportional to its period and to the controlled system’s error. The problem of the optimal integrated control and scheduling was investigated in [5]. It

was demonstrated that the optimal scheduling policy in the sense of a LQ performance index is dependent on the plant state. A significant improvement in control performance may be obtained by rapidly reacting to the disturbances [4]. This rapid reaction can be performed by plant state based on-line scheduling algorithms. The problem of the optimal on-line sampling period assignment was studied in [28]. A sub-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 opposition to this approach, the on-line scheduling heuristic proposed in this paper, and called Reactive Pointer Placement (RPP) scheduling, changes the effective sampling period[1] in reaction to the disturbances, and not according to a periodic triggering.

III. Optimal off-line scheduling

A. Problem formulation

Consider a collection of N continuous-time linear time-invariant (LTI) systems $(S^{(j)})_{1\leq j\leq N}$. Assume that each system $S^{(j)}$ is controlled by a task $\tau^{(j)}$, which is characterized by its worst-case execution time $d^{(j)}$. Since we are interested in deriving an optimal off-line schedule, the scheduling decisions (i.e. releasing and starting of control tasks) must be made periodically (i.e. at instants that are multiple of an elementary *tick period*), rather than at arbitrary time instants, as illustrated in ([29], Chapter "Clock-Driven Scheduling"). Let T_p be this elementary tick 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 [30], [31].

Usually, control tasks share the processor with other sporadic or aperiodic tasks, which do not perform the computations of the control laws, and are called *non-control tasks*. Examples of such tasks include signal processing, monitoring or communication tasks. The total computational load that these tasks induce may be described by their processor utilization rate, noted $\mathcal{R}$. If these non-control tasks do not have any real-time constraints, then they may be scheduled in the background, after the completion of the control tasks jobs (i.e. in the slices of the time slots where control tasks do not execute), as shown in ([29], Chapter "Clock-Driven Scheduling"). Otherwise, their schedulability and real-time constraints have to be checked using state of the art methods. This issue is tackled in the previously cited reference. 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 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 (i.e. whose utilization rate $\mathcal{R}=0.44$) is given in Fig. 1.

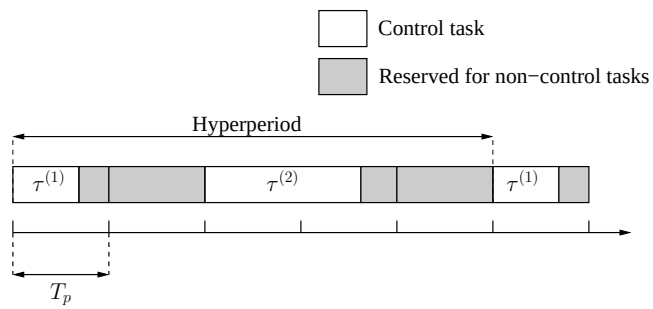

Fig. 1. An example of off-line scheduling of control and non-control tasks

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

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

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

$$e^{(j)}(k)=\sum_{l=k}^{k+\left\lceil \frac{d^{(j)}}{T_p}\right\rceil -1}\gamma^{(j)}(l). \tag{2}$$

The variable $e^{(j)}(k)\in\{0,1\}$ is the *task execution indicator* corresponding to the jobs of task $\tau^{(j)}$ and verifies

$$e^{(j)}(k)=1 \Longleftrightarrow \quad \text{the processor is allocated to task } \tau^{(j)} \text{ during a sub-interval of } [(k-1)T_p, kT_p). \tag{3}$$

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

$$\sum_{j=1}^{N} e^{(j)}(k)\leq 1. \tag{4}$$

In order to guarantee the computational needs of non-control tasks, described by their processor utilization rate $\mathcal{R}$, the

[1]the "effective sampling period" depends on the execution rate of the associated real-time control task

scheduling decisions of control tasks must satisfy

$$\mathcal{R} + \frac{1}{TT_p}\sum_{k=0}^{T-1}\sum_{j=1}^{N} d^{(j)}\gamma^{(j)}(k) \leq 1. \tag{5}$$

Each system $S^{(j)}$ is characterized by its sampled-data model, derived at the sampling period T_p according to the approach of [32], and described by

$$\begin{aligned} x^{(j)}(k+1) &= A^{(j)}x^{(j)}(k) + B_1^{(j)}w^{(j)}(k) + B_2^{(j)}u^{(j)}(k) \\ z^{(j)}(k) &= C_1^{(j)}x^{(j)}(k) + D_{11}^{(j)}w^{(j)}(k) + D_{12}^{(j)}u^{(j)}(k) \end{aligned}$$

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

In the following, we will assume that for all $j \in \{1, \ldots, N\}$:

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

Let $\mathcal{S}$ be the global system composed of systems $(S^{(j)})_{1\leq j\leq N}$. Assumption 1 is a necessary condition for the existence of a hyperperiod $T \times T_p$ and an off-line schedule (with hyperperiod $T \times T_p$) ensuring the reachability of the global system $\mathcal{S}$ [33]. When the scheduling of system $\mathcal{S}$ is performed according to a given off-line schedule ensuring its reachability, Assumption 2 guarantees the existence of a stabilizing controller (and also an optimal controller) and is usually made in optimal control problems.

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

$$x(k+1) = Ax(k) + B_1 w(k) + B_2 u(k) \tag{6a}$$

$$z(k) = C_1 x(k) + D_{11} w(k) + D_{12} u(k). \tag{6b}$$

Let $n = \sum_{j=1}^{N} n_j$, $m = \sum_{j=1}^{N} m_j$, $q = \sum_{j=1}^{N} q_j$ and Q the matrix defined by

$$Q = \begin{bmatrix} {C_1}^T \\ {D_{12}}^T \end{bmatrix} \begin{bmatrix} C_1 & D_{12} \end{bmatrix}. \tag{7}$$

In the considered modeling, when a control task finishes its execution, then it immediately updates the plant, which means that

$$u^{(j)}(k) \text{ is updated during interval } [(k-1)T_p, kT_p) \text{ if and only if } \gamma^{(j)}(k) = 1. \tag{8}$$

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

$$\gamma^{(j)}(k) = 0 \Longrightarrow u^{(j)}(k) = u^{(j)}(k-1). \tag{9}$$

Remark 1: Introducing scheduling variables allows modeling the computational delays in an abstract way. The computational delay of a given task is thus approximated in terms of numbers of elementary time slots.

In order to formulate the joint problem of optimal control and scheduling, in addition to the modeling of the tasks and the representation of the system's dynamics, it is necessary to choose an adequate criterion of performance. The previous studies that 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 dependent on the chosen initial states of the systems [5], [34]. 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 $\mathcal{H}_2$ performance criterion provides a solution to meet these objectives. In fact, using this performance index, the obtained off-line schedules will be independent of any initial conditions and disturbances. Moreover, the results may be easily transposed to a linear quadratic Gaussian (LQG) context, as illustrated in ([35], pp. 365-366). In fact, LQG optimal control problems are particular cases of $\mathcal{H}_2$ optimal control problems.

B. Decomposability of the optimal integrated control and off-line scheduling problem

An off-line schedule is called *admissible* if it satisfies constraints (2), (4) and (5). In theory, for a fixed T, the number of feasible off-line schedules with hyperperiod $T \times T_p$ is finite. Consequently, finding an optimal off-line schedule with hyperperiod $T \times T_p$ that minimizes the $\mathcal{H}_2$ norm of the global system amounts to the exploration of the set of feasible off-line schedules (with hyperperiod $T \times T_p$) and to the computation of the $\mathcal{H}_2$ norm of each feasible off-line schedule, in order to find an optimal one. However, in practice, *explicit search* methods like exhaustive search will rapidly suffer from the curse of dimensionality when used for solving problems with relatively large values of T or N. That's why we propose the use of the branch and bound method, which is an *implicit search* method, allowing a more intelligent exploration of the set of admissible off-line schedules.

When the scheduling of the control tasks is performed according to an admissible fixed off-line schedule with hyperperiod $T \times T_p$, the scheduling functions $\gamma^{(j)}(k)$ and $e^{(j)}(k)$ that are associated to this off-line schedule will be both periodic with period T (i.e. $\gamma^{(j)}(k) = \gamma^{(j)}(k+T)$ and $e^{(j)}(k) = e^{(j)}(k+T)$). It may be easily shown [10] that the model of the global system $\mathcal{S}$ may be described by a T-periodic discrete-time state representation. This remains true when the tasks are scheduled using the optimal off-line schedule that was previously determined. Consequently, using the well known results of the optimal periodic control theory, we know that the optimal control of each system $S^{(j)}$ is a state-feedback control law. Since $\mathcal{H}_2$ optimal control problems are solved over an infinite horizon, the optimal controller of the global system will be T-periodic.

These observations show that the optimal integrated control and off-line scheduling problem may be decomposed into two sub-problems, which may be solved successively: the optimal

scheduling sub-problem (which has to be solved first) and the optimal control sub-problem (whose solution requires the knowledge of the used scheduling).

C. Formal definition of the $\mathcal{H}_2$ norm

Assume that the control tasks are scheduled according to an admissible off-line schedule with hyperperiod $T \times T_p$ and that ensures the reachability of system $\mathcal{S}$. As previously mentioned, the off-line schedule as well as the optimal controller are T-periodic. For those reasons, system $\mathcal{S}$ will be a T-periodic discrete-time system. Consequently, in order to determine the $\mathcal{H}_2$ norm of the global system $\mathcal{S}$, we adopt a definition of this norm which is based on the impulse response of linear discrete-time periodic systems [36]. This definition generalizes the well-known definition of the $\mathcal{H}_2$ norm of discrete-time LTI systems. Let $(e_i)_{1\leq i\leq q}$ be the canonical basis vectors in $\mathbb{R}^q$ and δ_k the Dirac impulse applied at instant k. Using these notations, $\delta_k e_i$ is a Dirac impulse applied to the ith disturbance input at instant k. Let g_{ik} be the resulting controlled output of the global system $\mathcal{S}$ (in closed-loop) assuming zero initial conditions and that the control input u to the global system $\mathcal{S}$ satisfies the constraints (9). More formally, the response g_{ik} to the Dirac impulse $\delta_k e_i$ assuming that the global system is controlled by the control input u that satisfies the constraints (9), is completely defined by the following equations:

$$\begin{array}{rcl} x(0) & = & 0 \\ w(k) & = & e_i \\ \text{and for all } l \in \mathbb{N} & & \\ w(l) & = & 0 \text{ if } l \neq k \\ \gamma^{(j)}(k) = 0 & \Longrightarrow & u^{(j)}(k) = u^{(j)}(k-1) \text{ for all} \\ & & j \in \{1, \ldots, N\} \\ x(l+1) & = & Ax(l) + B_1 w(l) + B_2 u(l) \\ z(l) & = & C_1 x(l) + D_{11} w(l) + D_{12} u(l) \\ g_{ik}(l) & = & z(l). \end{array} \tag{10}$$

The $\mathcal{H}_2$ norm of the periodic system $\mathcal{S}$ is defined as

$$\|\mathcal{S}\|_2 = \sqrt{\frac{1}{T}\sum_{k=0}^{T-1}\sum_{i=1}^{q}\|g_{ik}\|_{l_2}^2}, \tag{11}$$

where the l_2 norm of g_{ik} is defined by

$$\|g_{ik}\|_{l_2} = \sqrt{\sum_{l=0}^{+\infty} g_{ik}(l)}. \tag{12}$$

Using this definition to compute the $\mathcal{H}_2$ norm involves the computation of $\|g_{ik}\|_{l_2}$, which requires the observation of the system's response over an infinite time horizon. In this work, a very close approximation of $\|g_{ik}\|_{l_2}$ is obtained through a finite horizon H from the instant where the impulse is applied. For that reason, it is necessary to choose H greater than the response time of the global system $\mathcal{S}$. 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 proportional to the square root of the sum of the squares of the l_2 norms corresponding to these responses. As a result, the "finite-horizon computed $\mathcal{H}_2$ norm" $\|\mathcal{S}\|_2(H)$ converges asymptotically to the true $\mathcal{H}_2$ norm $\|\mathcal{S}\|_2(\infty)$ computed over an infinite horizon, when $H \longrightarrow +\infty$. Consequently, for a desired precision, specified by the maximal absolute error ε_{H_2} between the true $\mathcal{H}_2$ norm $\|\mathcal{S}\|_2(\infty)$ computed over an infinite horizon and the "finite-horizon computed $\mathcal{H}_2$ norm" $\|\mathcal{S}\|_2(H)$, there will exist a horizon H_{min}, such that, for all $H \geq H_{min}$, $|\|\mathcal{S}\|_2(\infty) - \|\mathcal{S}\|_2(H)| \leq \varepsilon_{H_2}$. Based on this remark, and on the visualization tools which were implemented, the horizon H_{min} may be determined iteratively.

D. Solving of the optimal scheduling sub-problem

In this section, the translation of the optimal scheduling sub-problem in the sense of $\mathcal{H}_2$ into the mixed integer quadratic formulation is described. This translation requires the transformation of the involved constraints into linear equalities and/or inequalities. Constraints (9) may be translated into an equivalent conjunction of linear inequalities and equalities if extra variables are introduced, as illustrated in [37]. Remarking that (9) is equivalent to

$$u^{(j)}(k) - u^{(j)}(k-1) = \gamma^{(j)}(k)u^{(j)}(k) - \gamma^{(j)}(k)u^{(j)}(k-1), \tag{13}$$

and introducing the extra variables

$$v^{(j)}(k) = \gamma^{(j)}(k)u^{(j)}(k), \tag{14}$$

then (9) may be rewritten in the equivalent form

$$\begin{array}{l} v^{(j)}(k) \leq U^{(j)}\gamma^{(j)}(k) \\ v^{(j)}(k) \geq L^{(j)}\gamma^{(j)}(k) \\ v^{(j)}(k) \leq u^{(j)}(k) - L^{(j)}(1-\gamma^{(j)}(k)) \\ v^{(j)}(k) \geq u^{(j)}(k) - U^{(j)}(1-\gamma^{(j)}(k)), \end{array} \tag{15}$$

where $U^{(j)}$ and $L^{(j)}$ are respectively the upper and the lower bounds of the control commands $u^{(j)}$ of system $S^{(j)}$. In practice, these bounds correspond to the saturation thresholds of the actuators. If saturation does not occur, then $U^{(j)}$ (respectively $L^{(j)}$) may be set big enough (respectively small enough) with respect to the values that the control signals may take during the evolution of the system. Note that the product $o^{(j)}(k) = \gamma^{(j)}(k)u^{(j)}(k-1)$ may also be translated using the same procedure.

The constraints involved in this problem may be classified into two groups. The first group is related to the scheduling constraints (2), (4) and (5). Let

$$\bar{\Gamma}^{(j)} = \begin{bmatrix} \gamma^{(j)}(0) \\ \vdots \\ \gamma^{(j)}(H-1) \end{bmatrix} \text{ and } \bar{\Gamma} = \begin{bmatrix} \bar{\Gamma}^{(1)} \\ \vdots \\ \bar{\Gamma}^{(N)} \end{bmatrix}$$

then the constraints belonging to this group may be described by

$$\mathcal{A}_s\bar{\Gamma} \leq \mathcal{B}_s \tag{16}$$

where $\mathcal{A}_s$ and $\mathcal{B}_s$ are respectively a Boolean matrix and a Boolean vector of appropriate dimensions.

The second group is related to the computation of the impulsive responses g_{ik}, for $0 \leq k \leq T-1$ and $1 \leq i \leq$

q, over the horizon H. Let u^{ik}, x^{ik}, z^{ik}, v^{ik} and o^{ik} be respectively the values of the control, the state, the controlled output and the auxiliary variables corresponding to a Dirac impulse applied at instant k to the ith disturbance input of the global system. Let S^{ik} be the set of the involved constraints, for a given response g_{ik}. S^{ik} includes the state model (6), control updates constraints (9), the Dirac impulse verifying $w_i^{ik}(k) = 1$ and $w_r^{ik}(l) = 0$ for $r \neq i$ and $l \neq k$, in addition to the constraints that must be added to the problem to ensure the causality of the response (i.e. $u^{ik}(l) = 0$ for $l < k$).

Let $\bar{U}^{ik} = \begin{bmatrix} u^{ik}(0) \\ \vdots \\ u^{ik}(H-1) \end{bmatrix}$, $\bar{X}^{ik} = \begin{bmatrix} x^{ik}(0) \\ \vdots \\ x^{ik}(H-1) \end{bmatrix}$, $\bar{Z}^{ik} = \begin{bmatrix} z^{ik}(0) \\ \vdots \\ z^{ik}(H-1) \end{bmatrix}$, $\bar{V}^{ik} = \begin{bmatrix} v^{ik}(0) \\ \vdots \\ v^{ik}(H-1) \end{bmatrix}$, $\bar{O}^{ik} = \begin{bmatrix} o^{ik}(0) \\ \vdots \\ o^{ik}(H-1) \end{bmatrix}$ and $\mathcal{V}^{ik} = \begin{bmatrix} \bar{U}^{ik} \\ \bar{X}^{ik} \\ \bar{Z}^{ik} \\ \bar{V}^{ik} \\ \bar{O}^{ik} \end{bmatrix}$, then the set of constraints S^{ik} may be described by

$$\mathcal{A}^{ik} \begin{bmatrix} \bar{\Gamma} \\ \mathcal{V}^{ik} \end{bmatrix} \leq \mathcal{B}^{ik}, \tag{17}$$

where $\mathcal{A}^{ik}$ and $\mathcal{B}^{ik}$ are matrices of appropriate dimensions. Consequently, the optimal scheduling sub-problem in the sense of the $\mathcal{H}_2$ norm may be written in the form

$$\begin{cases} \min\limits_{\bar{\Gamma},(\mathcal{V}^{ik})_{1\leq i\leq q, 0\leq k\leq T-1}} \sum\limits_{k=0}^{T-1} \sum\limits_{i=1}^{q} (\mathcal{V}^{ik})^T \mathcal{H} \mathcal{V}^{ik} \\ \mathcal{A}_s \bar{\Gamma} \leq \mathcal{B}_s \\ \mathcal{A}^{ik} \begin{bmatrix} \bar{\Gamma} \\ \mathcal{V}^{ik} \end{bmatrix} \leq \mathcal{B}^{ik}, \text{ for } 1 \leq i \leq q, 0 \leq k \leq T-1, \end{cases} \tag{18}$$

where $\mathcal{H}$ is a matrix of appropriate dimensions, whose elements are functions of C_1, D_{11}, D_{12} and T. Problem (18) is a mixed integer quadratic problem (MIQP). The resolution of this problem leads to an optimal off-line schedule (computed over the horizon H) $\bar{\Gamma}^*$. This resolution may be performed using the branch and bound algorithm.

Remark 2: In operations research literature, branch and bound is considered among the most efficient methods for solving MIQP problems. As its name indicates it, this method is based on two complementary mechanisms: *branching* and *bounding*.

- 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.
- 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 subproblem **A**. Subproblem **A** is then *pruned*.

An intrinsic advantage of this method is that it allows finding sub-optimal solutions with a guaranteed distance to the true optimal solutions. The optimality distance is the difference between the cost of the best obtained solution and the lower bound on the optimal cost. The fact that the branch and bound algorithm allows finding sub-optimal solutions with a guaranteed distance to the true optimal ones comes from the fact that at each stage, this algorithm is able to compute a lower bound on the cost of the true optimal solutions (using continuous relaxation, i.e., replacing integer variables in $\{0, 1\}$ by continuous ones in $[0, 1]$ and solving an efficient quadratic program). For a description of the application of the branch and bound method for the solving of MIQP problems, the reader may consult reference [38] (Chapter 4, pp. 49-52) and references therein.

E. Solving of the optimal control sub-problem

Given an admissible 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 $(s(0), \ldots, s(\mathcal{T}-1))$, 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 $\bar{\Gamma}^*$, which is a solution of the optimization problem (18), it is then possible to derive the optimal control gains, according to the $\mathcal{H}_2$ performance criterion. In opposition to problem (18), 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 has only access to the plant state, the optimal $\mathcal{H}_2$ controller becomes identical to the optimal LQR controller (for a complete proof, see [39], p. 143). For that reason, in the following, we will consider that $D_{11} = 0$. The expression of $z^{(j)}(k)$ reduces to

$$z^{(j)}(k) = C_1^{(j)}(k) x^{(j)}(k) + D_{12}^{(j)}(k) u^{(j)}(k).$$

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

$\exists k \in \{1, \ldots, T\}, \exists i \in \mathbb{N}$ such that $k_q^{(j)} = k + iT$ and $\gamma^{(j)^*}(k) = 1$.

Based on this definition, the optimal control sub-problem may be stated as follows.

$$\begin{cases} \min\limits_{u^{(j)}} \sum\limits_{k=0}^{+\infty} {z^{(j)}}^T(k) z^{(j)}(k) \\ \text{subject to} \\ x^{(j)}(k+1) = A^{(j)} x^{(j)}(k) + B_2^{(j)} u^{(j)}(k) \\ u^{(j)}(k) \text{ may be updated if and only if } \exists q \in \mathbb{N} \text{ such that} \\ k_q^{(j)} = k \\ u^{(j)}(k) = u^{(j)}(k-1) \text{ if and only if } \forall q \in \mathbb{N},\ k_q^{(j)} \neq k. \end{cases} \tag{19}$$

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:

$$\vec{x}^{(j)}(q+1) = \vec{A}^{(j)}(q)\vec{x}^{(j)}(q) + \vec{B}_2^{(j)}(q)\vec{u}^{(j)}(q) \tag{20}$$

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 (20) describes the evolution of the state of system $S^{(j)}$, at the time slots where its control inputs are updated. Let

$$\Phi^{(j)}(k, k_q^{(j)}) = \begin{bmatrix} {A^{(j)}}^{k-k_q^{(j)}} & \sum_{i=0}^{k-k_q^{(j)}-1} {A^{(j)}}^i B_2^{(j)} \\ 0_{m_j,n_j} & I_{m_j} \end{bmatrix}.$$

Remarking that for k such that $k_q^{(j)} \leq k < k_{q+1}^{(j)}$,

$$\begin{bmatrix} x^{(j)}(k) \\ u^{(j)}(k) \end{bmatrix} = \Phi^{(j)}(k, k_q^{(j)}) \begin{bmatrix} \vec{x}^{(j)}(q) \\ \vec{u}^{(j)}(q) \end{bmatrix},$$

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

$$\begin{aligned} \vec{Q}^{(j)}(q) &= \sum_{k=k_q^{(j)}}^{k_{q+1}^{(j)}-1} \Phi^{(j)}(k, k_q^{(j)})^T Q^{(j)} \Phi^{(j)}(k, k_q^{(j)}) \\ &= \begin{bmatrix} \vec{Q}_1^{(j)}(q) & \vec{Q}_{12}^{(j)}(q) \\ {\vec{Q}_{12}^{(j)}}^T(q) & \vec{Q}_2^{(j)}(q) \end{bmatrix} \end{aligned}$$

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 (19) is equivalent to the following periodic optimal control problem

$$\begin{cases} \min\limits_{\vec{u}^{(j)}} \sum\limits_{q=0}^{\infty} \begin{bmatrix} \vec{x}^{(j)}(q) \\ \vec{u}^{(j)}(q) \end{bmatrix}^T \vec{Q}^{(j)}(q) \begin{bmatrix} \vec{x}^{(j)}(q) \\ \vec{u}^{(j)}(q) \end{bmatrix} \\ \text{subject to} \\ \vec{x}^{(j)}(q+1) = \vec{A}^{(j)}(q)\vec{x}^{(j)}(q) + \vec{B}_2^{(j)}(q)\vec{u}^{(j)}(q). \end{cases} \tag{21}$$

Problem (21) is a periodic optimal control problem over an infinite horizon. Let $\vec{S}^{(j)}(q)$ be the solution of the Riccati equation associated to the optimal control problem (21), which is described by

$$\begin{aligned} \vec{S}^{(j)}(q) = {} & {\vec{A}^{(j)}}^T(q)\vec{S}^{(j)}(q+1)\vec{A}^{(j)}(q) + \vec{Q}_1^{(j)}(q) \\ & - \left({\vec{A}^{(j)}}^T(q)\vec{S}^{(j)}(q+1)\vec{B}^{(j)}(q) + \vec{Q}_{12}^{(j)}(q)\right) \\ & \times \left({\vec{B}^{(j)}}^T(q)\vec{S}^{(j)}(q+1)\vec{B}^{(j)}(q) + \vec{Q}_2^{(j)}(q)\right)^{-1} \\ & \times \left({\vec{B}^{(j)}}^T(q)\vec{S}^{(j)}(q+1)\vec{A}^{(j)}(q) + {\vec{Q}_{12}^{(j)}}^T(q)\right). \end{aligned} \tag{22}$$

Problem (21) admits a unique solution $\vec{u}^{(j)^*}(q)$ defined by

$$\vec{u}^{(j)^*}(q) = -\vec{K}^{(j)}(q)\vec{x}^{(j)}(q)$$

with

$$\begin{aligned} \vec{K}^{(j)}(q) = {} & \left(\vec{Q}_2^{(j)}(q) + {\vec{B}^{(j)}}^T(q)\vec{S}^{(j)}(q+1)\vec{B}^{(j)}(q)\right)^{-1} \\ & \times \left({\vec{B}^{(j)}}^T(q)\vec{S}^{(j)}(q+1)\vec{A}^{(j)}(q) + {\vec{Q}_{12}^{(j)}}^T(q)\right). \end{aligned} \tag{23}$$

Using the lifting approach described in [34], it may be proved that matrices $\vec{S}^{(j)}(q)$ and $\vec{K}^{(j)}(q)$ are both $\mathcal{T}^{(j)}$-periodic. The optimal cost corresponding to an evolution of system $S^{(j)}$ from instant $k_\iota^{(j)}$ to $+\infty$ is given by

$$\sum_{k=k_\iota^{(j)}}^{+\infty} {z^{(j)}}^T(k) z^{(j)}(k) = {\vec{x}^{(j)}}^T(\iota)\vec{S}^{(j)}(\iota)\vec{x}^{(j)}(\iota). \tag{24}$$

Remark 3: Note that optimal control gains $\vec{K}^{(j)}(q)$, 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 4: In, equation (24), 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 (24) 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 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 $\tilde{S}^{(j)}(k)$ is defined in equation (25) on the top of the next page. and

$$\check{S}^{(j)}(q) = \bar{\mathcal{Y}}^{(j)^T} \vec{S}^{(j)}(q) \bar{\mathcal{Y}}^{(j)}.$$

F. A numerical example

We 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 and corresponds to a linearized model of a quick inverted pendulum. This benchmark was obtained through the discretization of three continuous time systems having different response times and stability properties, in order to allow the evaluation of the influence of these characteristics on the obtained optimal off-line schedules. These three systems are defined by:

$$\begin{aligned} x^{(1)}(k+1) &= \begin{bmatrix} 0.996 & 0.026 \\ -0.250 & 0.998 \end{bmatrix} x^{(1)}(k) + \begin{bmatrix} 0.013 \\ 0.999 \end{bmatrix} w_1^{(1)}(k) \\ &\quad + \begin{bmatrix} 0.013 \\ 0.999 \end{bmatrix} u^{(1)}(k) \\ z^{(1)}(k) &= \begin{bmatrix} \begin{bmatrix} 89.4 & 0 \\ 0 & 3.16 \end{bmatrix} x^{(1)}(k) \\ 10^{-3} \begin{bmatrix} 0 & 15.9 & 14.7 \\ 0 & 14.7 & 69.1 \end{bmatrix} w^{(1)}(k) \\ u^{(1)}(k) \end{bmatrix} \end{aligned}$$

$$\begin{aligned} x^{(2)}(k+1) &= \begin{bmatrix} 0.993 & 0.345 \\ -0.025 & 1.000 \end{bmatrix} x^{(2)}(k) + \begin{bmatrix} 0.124 \\ 0.791 \end{bmatrix} w_1^{(2)}(k) \\ &\quad + \begin{bmatrix} 0.138 \\ 0.800 \end{bmatrix} u^{(2)}(k) \\ z^{(2)}(k) &= \begin{bmatrix} \begin{bmatrix} 2.236 & 0 \\ 0 & 1 \end{bmatrix} x^{(2)}(k) \\ 10^{-3} \begin{bmatrix} 0 & 4.16 & 3.81 \\ 0 & 3.81 & 17.4 \end{bmatrix} w^{(2)}(k) \\ 3.16\, u^{(2)}(k) \end{bmatrix} \end{aligned}$$

$$\begin{aligned} x^{(3)}(k+1) &= \begin{bmatrix} 1.11 & 0 & 0.002 & 0.55 \\ 0 & 1 & 0.212 & 0 \\ 0 & 0 & 0.761 & 0 \\ 0.45 & 0 & 0.009 & 1.11 \end{bmatrix} x^{(3)}(k) \\ &\quad + 10^{-4} \begin{bmatrix} 23 \\ 9 \\ 75 \\ 81 \end{bmatrix} w_1^{(3)}(k) + 10^{-4} \begin{bmatrix} 28 \\ 11 \\ 88 \\ 106 \end{bmatrix} u^{(3)}(k) \\ z^{(3)}(k) &= \begin{bmatrix} \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 31.6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x^{(3)}(k) \\ 10^{-4} \begin{bmatrix} 0 & 3.08 & 3.80 & 0.71 & 0.92 \\ 0 & 3.80 & 4.98 & 1.03 & 1.27 \\ 0 & 0.71 & 1.03 & 1.11 & 1.17 \\ 0 & 0.92 & 1.27 & 1.17 & 1.24 \end{bmatrix} w^{(3)}(k) \\ u^{(3)}(k) \end{bmatrix} \end{aligned}$$

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 has to be executed immediately after task $\tau^{(3)}$.

We assume that the execution platform is an Allen-Bradley CompactLogix 5320 programmable logic controller (PLC) [40] 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 can be found in [40]. 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 I. We also assume that a set of sporadic and aperiodic tasks may need to be executed on the same processor. Based on this requirement as well as on the WCET of the tasks, the elementary time slot duration T_p was chosen equal to $1\ ms$.

TABLE I

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 $\tau^{(2)}$ require only one time slot. The consecutive execution of tasks $\tau^{(3)}$ and τ^c requires two time slots. Sporadic and aperiodic tasks require at most $40\ \%$ of the CPU power.

The optimal solutions, corresponding to different choices of T are illustrated in table II. 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 resolved using the solver CPLEX (Release 9.1.0) from ILOG. In this particular implementation of the optimization algorithm, H must be a multiple of T. It is sufficient to choose H bigger than 27 to guarantee a maximal absolute error $\varepsilon_{\mathcal{H}_2} = 10^{-4}$.

The optimization results are given in table II. In this table, 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 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 $T = 5$. 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 that gives the best $\mathcal{H}_2$ norm ($\mathcal{H}_2 = 9.7463$) is a multiple

$$\tilde{S}^{(j)}(k) = \begin{cases} \left[\tilde{\Psi}^{(j)^{k_q^{(j)}-k}}\right]^T \check{S}^{(j)}(q)\tilde{\Psi}^{(j)^{k_q^{(j)}-k}} + \sum_{l=k}^{k_q^{(j)}-1} \left[\tilde{\Psi}^{(j)^{l-k}}\right]^T Q^{(j)}\tilde{\Psi}^{(j)^{l-k}} & \text{if } k_{q-1}^{(j)} < k < k_q^{(j)} \\ \vec{S}^{(j)}(q) & \text{if } k = k_q^{(j)} \end{cases} \tag{25}$$

TABLE II
OPTIMAL $\mathcal{H}_2$ NORM AS A FUNCTION OF T

T	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

of 5. The resource allocation depends on the dynamics of the systems and on their sensitivity to the Dirac impulse disturbance of the $\mathcal{H}_2$ performance evaluation. It may be proved (using the formal definition of the $\mathcal{H}_2$ norm of system $\mathcal{S}$) that a circular permutation of an optimal schedule remains optimal.

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 [41] (for example, choosing the sampling period T_s of a first-order system such that $\frac{T_r}{T_s} \approx 4 \ldots 10$, where T_r is the rising time of the system). The advantage of the branch and bound method is that it is able to use these feasible solutions to considerably reduce the search space by pruning the regions that are proved to be worst than these given initial solutions. 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.

Remark 5: When the number of systems (and corresponding control tasks) increases, the potential for improving the global performance is more important but the complexity of the decision increases also, due to the "curse of dimensionality". Finding the true optimal off-line schedule becomes then difficult. The advantage of the proposed approach is its ability to provide sub-optimal solutions with an upper-bounded distance from the optimality, along the execution of the branch and bound algorithm. For problems with a large number of tasks, the algorithm has to be run during a predetermined amount of time, starting from a feasible off-line schedule that the designer might determine *ad-hoc* using state of the art methods, in order to find solutions that are better than the given initial schedule, which was derived by the designer.

Remark 6: 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 $\tau_2^{(1)}$.

IV. 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, which 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

$$p := (p+1) \mod \mathcal{T}. \tag{26}$$

Knowing the pointer position p, the control task to execute is $s(p)$. It is convenient to define the map t, which associates to the pointer position its absolute position (expressed as multiples of T_p in the schedule). The map t linking the different pointer positions to their absolute positions in the example of Fig. 2 is defined in table III.

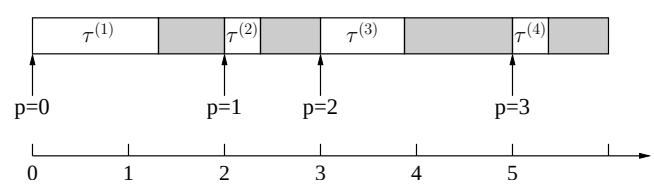

Fig. 2. Absolute positions of the pointer

TABLE III
THE MAP t

Pointer position p	Absolute position $t(p)$
0	0
1	2
2	3
3	5

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 improvements may be possible by executing more frequently the control tasks of the systems that 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 crucial for the stability and performance guarantees.

The idea behind the proposed scheduling strategy is to use the free available computing resources (i.e. where 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 optimal pointer placement scheduling strategy, which represents the cornerstone of the used adaptive scheduling strategy. Finally, the method for reduction of the computational complexity of the feedback scheduler is described.

A. 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 can 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 which 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 7: 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. In fact, few real-time operating systems support the run-time change of priorities.

B. Adaptive scheduling of control tasks

Choosing the pointer replacement as an on-line decision allows exploiting more efficiently the available computational resources in order to improve the control performance. These resources are thus allocated according to the "instantaneous needs" of the controlled systems. In the proposed adaptive scheduling strategy, instead of systematically using (26), 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

$$\tilde{x}(k) = \begin{bmatrix} \tilde{x}^{(1)}(k) \\ \vdots \\ \tilde{x}^{(N)}(k) \end{bmatrix}.$$

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

$$\begin{aligned} J^{(j)}(k,p) &= \sum_{i=0}^{\infty} {z^{(j)}}^T(k+i) z^{(j)}(k+i) \\ &= \tilde{x}^{(j)}(k)^T \tilde{S}^{(j)}(t(p)) \tilde{x}^{(j)}(k). \end{aligned} \tag{27}$$

If the pointer is placed at position p at instant k, then the cost function corresponding to an evolution of the global system $\mathcal{S}$ over an infinite horizon starting from the state $\tilde{x}(k)$ at instant k and using the static scheduling algorithm is

$$J(k,p) = J^{ss}(\tilde{x}(k), k, +\infty, p) = \sum_{j=1}^{N} J^{(j)}(k,p). \tag{28}$$

This strategy (called optimal pointer placement scheduling) was employed in [5] 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 [4]).

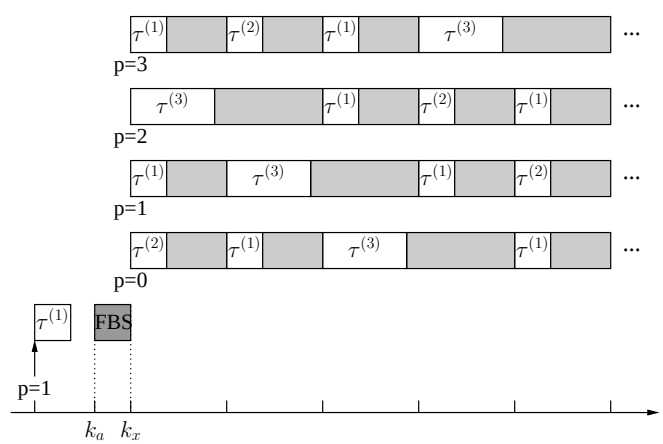

Fig. 3. Optimal pointer placement scheduling of tasks $\tau^{(1)}$, $\tau^{(2)}$ and $\tau^{(3)}$. FBS is the abbreviation of feedback scheduler

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 task

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 instant k_a (the instant where the 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 task 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. Fig. 3 illustrates the method (in the case of the adaptive scheduling of the numerical example of section III-F).

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

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

The expression of $\hat{S}(t(p))$ may be easily deduced from the plant model, the expression of $\tilde{S}(t(p))$ and the control gains.

C. Reduction of the feedback scheduler overhead

In the following, a method of 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 can be formalized by defining a practical equilibrium region $R^{(i)}$ 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 can be formally defined by

$$R^{(i)} = \left\{\tilde{x}^{(i)}(k) \text{ such that } \left\|\tilde{x}^{(i)}(k)\right\|_\infty \leq \varepsilon_x^{(i)}\right\}.$$

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 has 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 can take advantage from knowing that a given system is practically stable. For a given system $S^{(j)}$, let

$$\bar{J}_{min}^{(j)}(p) = \min_{\tilde{x}^{(j)}(k) \in R^{(j)}} \left(J^{(j)}(k,p)\right)$$

and

$$\bar{J}_{max}^{(j)}(p) = \max_{\tilde{x}^{(j)}(k) \in R^{(j)}} \left(J^{(j)}(k,p)\right),$$

then we have the following proposition.

Proposition 1: Let p_1 and p_2 be two pointer positions and I a subset of $\{1, \ldots, N\}$. If

$$\forall i \in I, \quad \left\|\tilde{x}^{(i)}(k)\right\|_\infty \leq \varepsilon_x^{(i)}$$

and

$$\sum_{j \in \{1,\ldots,N\}-I} J^{(j)}(k,p_2) + \sum_{j \in I} \bar{J}_{max}^{(j)}(p_2) < \sum_{j \in \{1,\ldots,N\}-I} J^{(j)}(k,p_1) + \sum_{j \in I} \bar{J}_{min}^{(j)}(p_1)$$

then

$$J(k,p_2) < J(k,p_1)$$

Proof: This result directly follows from the fact that

$$\begin{aligned} J(k,p_1) &= \sum_{j \in \{1,\ldots,N\}-I} J^{(j)}(k,p_1) + \sum_{j \in I} J^{(j)}(k,p_1) \\ &\geq \sum_{j \in \{1,\ldots,N\}-I} J^{(j)}(k,p_1) + \sum_{j \in I} \bar{J}_{min}^{(j)}(p_1) \end{aligned}$$

and

$$\begin{aligned} J(k,p_2) &= \sum_{j \in \{1,\ldots,N\}-I} J^{(j)}(k,p_2) + \sum_{j \in I} J^{(j)}(k,p_2) \\ &\leq \sum_{j \in \{1,\ldots,N\}-I} J^{(j)}(k,p_2) + \sum_{j \in I} \bar{J}_{max}^{(j)}(p_2). \end{aligned}$$

■

Constants $\bar{J}_{min}^{(j)}(p)$ and $\bar{J}_{max}^{(j)}(p)$ may be easily pre-computed off-line using a QP solver.

Example 2: In order to illustrate the gains in terms of computational complexity, we consider the example of section III-F. 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 situation where input operations are performed by independent hardware devices and that their computational overhead may be neglected, a possible pseudocode of 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 $\bar{I}_p = \{1, \ldots, N\} - I_p$. 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 $\bar{I}_p = \{s(\pi), \pi \in \mathcal{P}_p\}$. When this choice

is performed, I_p contains $s(p)$. Note that I_p may also be chosen such that $I_p = \emptyset$ (when more computational resources are available to the feedback scheduler). 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. The computational resources that are needed are time-varying and are in the worst-case proportional to the cardinality of the sets $\bar{I}_p$ (which is the complementary of the set I_p in $\{1, \ldots, N\}$) and $\mathcal{P}_p$. In fact, these resources are essentially needed for the on-line computation of the functions $J^{(j)}(k, \pi) - J^{(j)}(k, p)$, for $j \in \bar{I}_p$, as illustrated in the RPP algorithm listing below.

Read $x(k)$;
$p := p + 1 \mod \mathcal{T}$;
if $\forall i \in I_p, \ \left\|\tilde{x}^{(i)}(k)\right\|_\infty \leq \varepsilon_x^{(i)}$ **or** $I_p = \emptyset$ **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 1: Pseudocode of the feedback scheduling algorithm, called Reactive Pointer Placement (RPP) scheduling algorithm

Remark 8: Using the proposed complexity reduction methodology, it is possible to bound the computational requirements of the feedback scheduler even when the number of control tasks increases. In general, the situations where all the independent systems are severely disturbed at the same time are less frequent than the situations where some systems are disturbed and some others are in normal operation. The RPP algorithms, allocates the available resources according to the needs of the controlled systems. Therefore, intuitively, when the number of tasks increases, the potential of improving the control performance may be more important because the quantity of unneeded resources that may be allocated to the disturbed systems will increase.

D. 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 1: Let $\tilde{x}(0)$ be a given initial extended state of the global system $\mathcal{S}$ (composed of systems $(S^{(j)})_{1 \leq j \leq N}$) 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 $\mathcal{S}$ when scheduled using RPP, $p(l)$ the pointer position at the $(l+1)^{th}$ execution of RPP and k_l the index of the time slot that corresponds to the end of this $(l+1)^{th}$ execution. Assume without loss of generality that $k_0 = 0$. Let $J^{rpp-ss}(l)$ be the cost function corresponding to an evolution starting from the initial state $\tilde{x}(0)$ where RPP is applied from instant $k_0 = 0$ to instant k_l and then followed by the application of the static scheduling algorithm (which is applied from instant k_{l+1} to $+\infty$). By construction of the RPP scheduling strategy (as described in listing Algorithm 1), and using Proposition 1, the pointer position at the $(l+1)^{th}$ execution is set to position $p(l)$ verifying equations (29) on the top of the next page. In (29), $J^{ss}(\tilde{x}^{rpp}(k_l), k_l, +\infty, p(l))$ (respectively $J^{ss}(\tilde{x}^{rpp}(k_l), k_l, +\infty, (p(l-1)+1) \mod \mathcal{T})$) represents the predicted cost corresponding to an evolution over an infinite horizon of the global system from state $\tilde{x}^{rpp}(k_l)$ where the pointer at instant k_l is placed at position $p(l)$ (respectively $(p(l-1)+1) \mod \mathcal{T}$). In fact, as previously mentioned, the scheduling decisions that are preformed by the RPP algorithm are based on a prediction of the evolution of the system, for selected pointer positions, and assuming that the static scheduling algorithm is used during these predicted evolutions. Note that $(p(l-1)+1) \mod \mathcal{T})$ represents the pointer position obtained by incrementing the pointer according to relation (26) (i.e. open-loop static scheduling).

Adding $J^{rpp}(\tilde{x}(0), 0, k_{l-1})$ to both left and right terms of inequality (29b), for $l > 0$, and remarking that

$$\begin{aligned} J^{rpp-ss}(l) &= J^{rpp}(\tilde{x}(0), 0, k_{l-1}) \\ &\quad + J^{ss}(\tilde{x}^{rpp}(k_l), k_l, +\infty, p(l)), \end{aligned}$$

and

$$\begin{aligned} &J^{rpp-ss}(l-1) = J^{rpp}(\tilde{x}(0), 0, k_{l-1}) \\ &+ J^{ss}(\tilde{x}^{rpp}(k_l), k_l, +\infty, (p(l-1)+1) \mod \mathcal{T}), \end{aligned}$$

we get

$$J^{rpp-ss}(l) \leq J^{rpp-ss}(l-1), \text{ for } l > 0. \tag{30}$$

Recalling that $J^{ss}(\tilde{x}^{rpp}(k_0), k_0, +\infty, p(0)) = J^{rpp-ss}(0)$, remarking that $\lim_{l \to +\infty} J^{rpp-ss}(l) = J^{rpp}(\tilde{x}(0), 0, +\infty)$ and using inequalities (29a) and (30), we get

$$J^{rpp}(\tilde{x}(0), 0, +\infty) \leq J^{ss}(\tilde{x}(0), 0, +\infty, p_0). \tag{31}$$

■

Theorem 1 demonstrates that the RPP strategy guarantees the performance improvements with respect to the static scheduling algorithm. The stability of the RPP scheduling algorithm directly follows from Theorem 1 and is given the following corollary:

Corollary 1: If Q is positive definite and if the asymptotic stability of the global system $\mathcal{S}$ (composed of systems $(S^{(j)})_{1 \leq j \leq N}$) is guaranteed 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 $\mathcal{S}$ is asymptotically stable. Knowing the asymptotic stability of the system scheduled using the static scheduling algorithm and using relation (31), the corollary is proved. ■

Remark 9: Using the RPP strategy, the effective frequency with which the control tasks are executed is not necessarily

$$J^{ss}(\tilde{x}^{rpp}(k_0), k_0, +\infty, p(0)) \leq J^{ss}(\tilde{x}(0), 0, +\infty, p_0) \quad \text{if} \quad l = 0, \tag{29a}$$

$$J^{ss}(\tilde{x}^{rpp}(k_l), k_l, +\infty, p(l)) \leq J^{ss}(\tilde{x}^{rpp}(k_l), k_l, +\infty, (p(l-1)+1) \mod \mathcal{T}) \quad \text{if} \quad l > 0, \tag{29b}$$

constant. In particular, some tasks may be executed more frequently during some periods of time. Based on Theorem 1, we proved that this irregular execution, which is performed according to a well defined methodology (on-line scheduling that minimizes an infinite horizon cost) provides a better (and in the worst-case situation a similar) control performance than the basic static schedule. The performance improvements may be seen as a direct application of the Bellman optimality principle. In fact, the RPP algorithm has more degrees of freedom than the static scheduling algorithm. Its worst-case performance is that of the static scheduling strategy. The control coefficients (gains) that will be used by each control task are taken into account when the infinite horizon cost is computed. For that reason, there is no need to re-compute the control coefficients of the tasks that will be executed with an irregular rate. This considerably reduces the complexity of the on-line scheduling algorithm.

E. 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 of 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

$$\begin{aligned} \zeta_\pi &= (\zeta_\pi(0), \zeta_\pi(1), \ldots, \zeta_\pi(\mathcal{T}-1)) \\ &= (s(\pi(0)), s(\pi(1)), \ldots, s(\pi(\mathcal{T}-1))) \,. \end{aligned}$$

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:

$$\begin{cases} \mathcal{Z}_\pi^{(j)}(p) = 1 & \text{if } \exists k \in \{1, \ldots, n_s\} \text{ such that } \zeta_{\pi_k}(p) = j \\ \mathcal{Z}_\pi^{(j)}(p) = 0 & \text{otherwise.} \end{cases} \tag{32}$$

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 $(\pi_1^*, \ldots, \pi_{n_s}^*)$ of the following optimization problem

$$\begin{cases} (\pi_1^*, \ldots, \pi_{n_s}^*) = \min\limits_{(\pi_1, \ldots, \pi_{n_s}) \in \Pi^{n_s}} \sum\limits_{j=1}^{N} \max\limits_{k_1, k_2, k_1 \leq k_2} \{k_2 - k_1, \\ \text{such that: } \bar{\mathcal{Z}}_\pi^{(j)}(k_1) = 1,\ \bar{\mathcal{Z}}_\pi^{(j)}(k_2) = 1 \text{ and for all} \\ k_1 < k < k_2,\ \bar{\mathcal{Z}}_\pi^{(j)}(k) = 0\}. \end{cases} \tag{33}$$

This optimization problems aims at minimizing the sum over j of the "maximal distances" between two successive output reading of system $S^{(j)}$.

F. A numerical example

In order to evaluate the proposed approach, the real-time implementation of the example of Section III-F is considered. Based on the assembler language of the CompactLogix 5320 PLC, a handwritten assembler code of the feedback scheduler was written. Tasks execution was simulated using the toolbox TRUETIME [9], [30], 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 systems dynamics.

Based on Table I, 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 the feedback scheduler must be less or equal to $717.06\ \mu s$ or $980.08\ \mu s$, depending on the non-control tasks slots lengths. To solve this problem, an implementation of the feedback scheduler (as described in Section IV-C) 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 $\pi^* = (\pi^*(0), \pi^*(1), \pi^*(2), \pi^*(3)) = (1, 0, 3, 2)$. 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. These parameters are simply a consequence of the available computational resources to execute the feedback scheduler. In order to improve the responsiveness of the feedback scheduler, $\pi^*(p)$ was computed using the optimization heuristic (33). Based on these choices, the worst-case execution times of the feedback scheduler (for

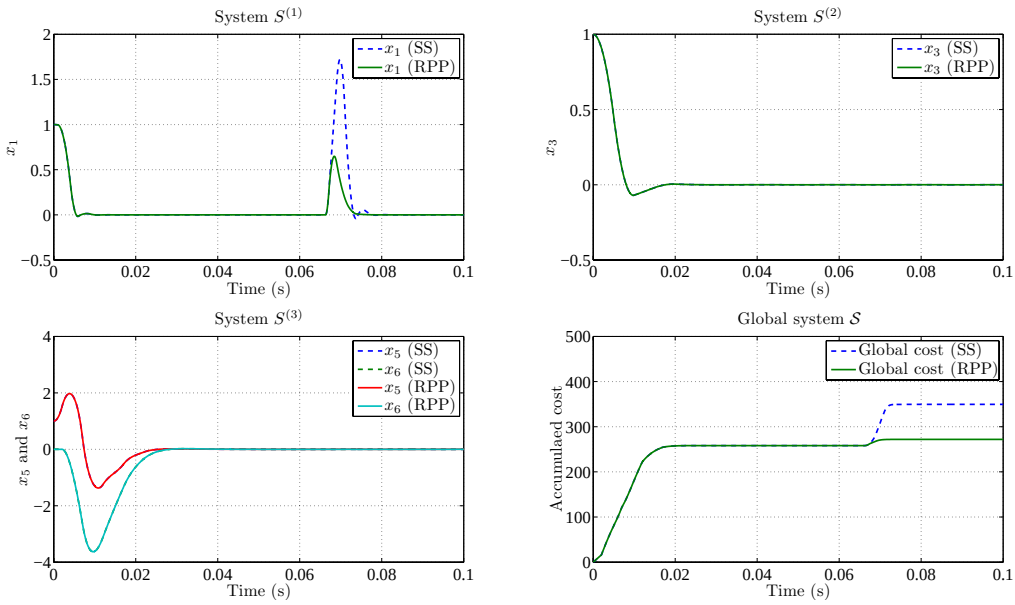

Fig. 4. Global system responses and accumulated cost using the static scheduling (SS) and the RPP scheduling algorithms

TABLE IV
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

a given pointer position) are given in Table IV. 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 $3393.8\ \mu s$.

The global system responses corresponding to states x_1, x_3, x_5 and x_6 of the global system (i.e. states $x_1^{(1)}$, $x_1^{(2)}$, $x_1^{(3)}$ and $x_2^{(3)}$) as well as the associated accumulated global cost are depicted in Figure 4. 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 5). These changes allowed to better react to this disturbance and to improve the quality of control (as illustrated in Figure 4). It may be seen that when these disturbances occur, the execution time of the feedback scheduler increases (as illustrated in Figure 5). This increase is due to the additional test which is needed to guarantee that the "reactive pointer change" will improve the performances while maintaining the stability.

Finally, the RPP scheduling algorithm is tested, in the case when all the systems are in the practical equilibrium region, except one, which is perpetually disturbed. In the simulations depicted in Fig. 6, 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 (Fig. 6, left). 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 (Fig. 6, right).

Remark 10: In all the situations, the RPP scheduling algorithm is able to maintain the stability of all the plants and to provide a performance that is better (and in the worst-case similar) to that obtained by the static scheduling algorithm, even when all the plants are disturbed at the same time. In this example, the restriction concerning the fact that RPP is able to react to only one disturbance is only due to the limitations of the computational resources that are allocated to its execution. If more resources are available to the execution of the feedback scheduler, then the parameter I_p may be set to $\emptyset$. In this situation, the reactive pointer change may be performed even when all the plants are disturbed at the same time. This situation will be illustrated in the experimental study of the next section.

V. EXPERIMENTAL STUDY

In order to evaluate the extent of the considered theoretical assumptions, to study experimentally the robustness of the proposed on-line scheduling algorithm and to compare its performance to state of the art methods (i.e. fixed-priority preemptive scheduling), the proposed methodology is applied to a real world application. The considered application is the concurrent speed control of two DC motors using an embedded processor. The control objective is to impose desired angular velocities to each DC motor. The considered motors present

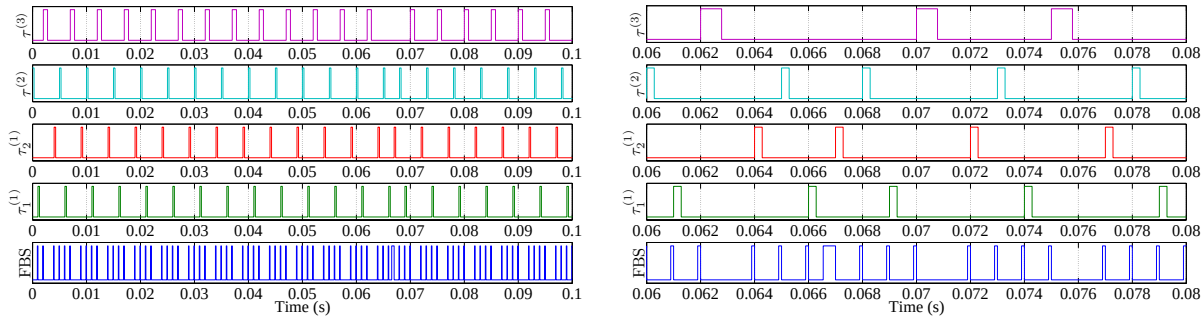

Fig. 5. RPP schedule (left) and zoom on the RPP schedule between instants 60 ms and 80 ms (right)

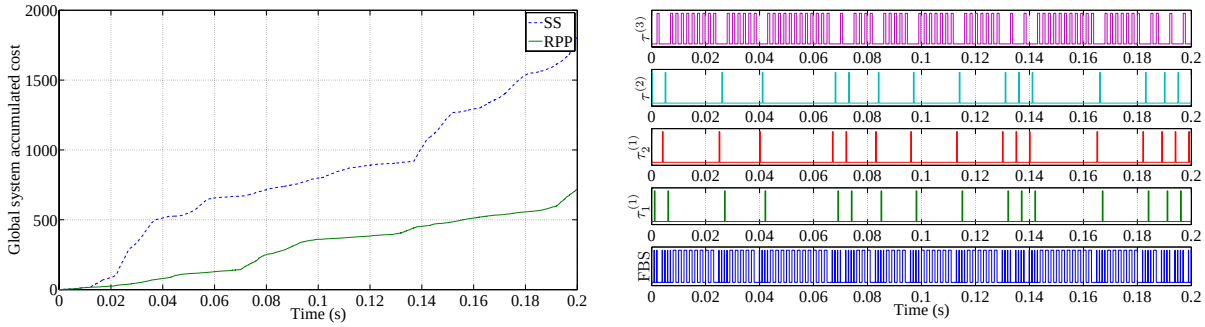

Fig. 6. Accumulated cost functions (left) and tasks schedules (right) when systems $S^{(1)}$ and $S^{(2)}$ are in the equilibrium regions whereas system $S^{(3)}$ is continuously disturbed

some non-linearities at zero crossing velocities (dry friction). However, they may be approximated by first-order LTI models. The control tasks are executed on an aJile aJ-PC104 board, equipped with a JEM2 processor, clocked at 40 MHz. A PC with a Matlab/RTW (Real-Time Workshop) board generates the desired set points (R_1 and R_2) and measures the voltages that are the image of the motors speed (i.e. proportional to the motors angular velocities). The experimental setup is described in Figure 7. The model of each DC motor (viewed between

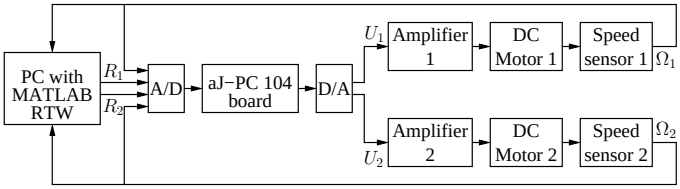

Fig. 7. Experimental setup

the input of the amplifier and the output of the speed sensor) may be respectively approximated by the first-order models

$$\Omega_1(s) = \frac{0.94}{1+6.2s}U_1(s),$$

and

$$\Omega_2(s) = \frac{1.17}{1+2.9s}U_2(s),$$

where Ω_i and U_i are respectively the image of the angular velocity (output of the speed sensor) and the control voltage (input to the amplifier) of the ith DC motor, $i \in \{1,2\}$. Using the LQR method, two proportional integral (PI) controllers were derived in order to impose similar closed loop characteristics for the two motors.

Each DC motor is controlled by a control task $\tau^{(i)}$, $i \in \{1,2\}$. The processor is shared with other periodic and sporadic tasks. During the experiments, a set of periodic tasks, with respective periods 4, 40 and 200 ms, are executed in parallel (with the highest priorities, assigned according to the rate monotonic rule). Their CPU utilization load is equal to 60 %. The measured execution times of the different segments (i.e. parts) of the motors control tasks and the feedback scheduler are given in table V. The segments *Inputs reading*

TABLE V

MEASURED EXECUTION TIMES OF THE SEGMENTS OF A MOTOR CONTROL TASK AND OF THE RPP ALGORITHM

Motor Control Task		RPP algorithm	
Segment	Execution Time (μs)	Segment	Execution Time (μs)
Inputs reading	363	Inputs reading	371
Control computation	58	Scheduling computation	230
Output application	322		
Total	743	Total	601

and *Output application* represent the computations that are necessary for the acquisition of the controller inputs and the application of the control outputs. The segment *Control computation* represents the computation of the state feedback

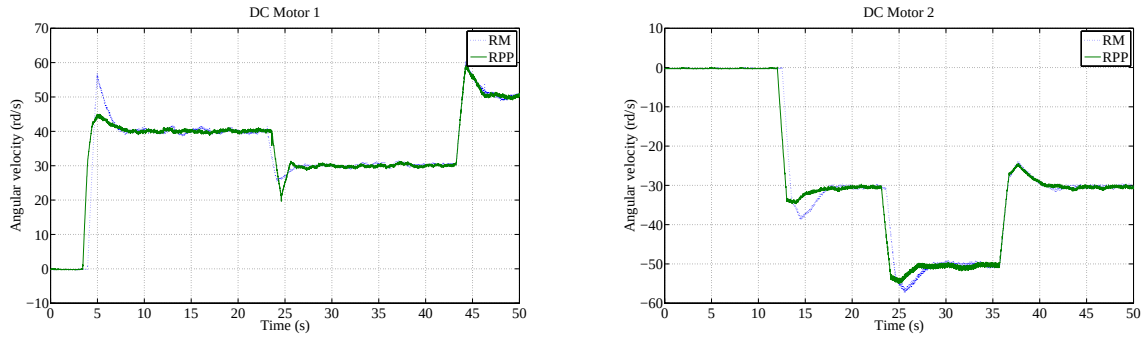

Fig. 8. Speed responses corresponding to the RPP and RM scheduling algorithms

control, whereas the segment *Scheduling computation* represents the code of the RPP algorithm, as illustrated in the listing Algorithm 1 (evaluation and comparison of the quadratic cost functions and pointer placement decision).

The objective of the experiments is to compare the control performance when the two control tasks $\tau^{(1)}$ and $\tau^{(2)}$ are scheduled using:

- the rate monotonic scheduling algorithm (RM), where the preemption is authorized, and where each DC motor is controlled by an independent control task running at the period of $1\ s$.
- the RPP scheduling algorithm with the parameters T_p=0.5 s, a basic optimal off-line schedule 121212... (obtained using the $\mathcal{H}_2$ optimization), $I_1 = \emptyset$, $I_2 = \emptyset$, $\mathcal{P}_0 = \{1\}$ and $\mathcal{P}_1 = \{0\}$.

Based on these parameters, the controllers that are used in both cases are identical (both derived at the sampling period of $1\ s$).

The experimentation scenario is the following. Initially, the two DC motors are stopped (i.e. their angular velocity is zero). The higher priority tasks use $60\ \%$ of the CPU. At instant $t = 3\ s$, the set point of the first motor is set to $40\ rad/s$. At instant $t = 12\ s$, the set point of the second motor is set to $-30\ rad/s$. At instant $t = 23\ s$, the set points of the two motors are respectively changed to 30 and $-50\ rad/s$. At instant $t = 30\ s$, a set of sporadic tasks, using more than $40\ \%$ of the CPU, begin their execution. The priority of these tasks is lower than the priority of the motors control tasks but higher than the priority of the feedback scheduler. At instant $t = 35\ s$, the set point of the second motor is set to $-30\ rad/s$. Finally, at instant $t = 43\ s$, the set point of the first motor is set to $50\ rad/s$.

The experimental results (i.e. the measured angular velocities of DC motors 1 and 2 using the RM and the RPP scheduling algorithms) are depicted in Fig. 8. They show that a significant improvement in the control performance is achieved by the RPP scheduling algorithm with respect to the rate monotonic scheduling algorithm. These improvements consist on a considerable reduction of the overshoot and an improvement of the response time, manifesting themselves after set point changes. These improvements are due to a more efficient use of the available computing resources by the RPP algorithm. The steady state behavior using the two algorithms is similar. At instant $t = 23\ s$, the set points are applied at the same time. The RPP algorithm chooses to allow the resources to the control task of motor 2, which experiences the largest deviations, in order to optimize the global cost. From instant $t = 30\ s$, the admitted sporadic tasks induce a situation of processor overload. The feedback scheduler cannot be executed. In fact, as previously indicated, the feedback scheduler has the lowest priority in the system. Consequently, in this situation of overload, motors control tasks are executed according to the optimal static $\mathcal{H}_2$ schedule. For that reason, the obtained control performance (obtained from instant $t = 30\ s$) is identical to that obtained with the rate monotonic algorithm (i.e. very similar responses are observed).

VI. Conclusions

In this paper, a new approach for control tasks scheduling is 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 (which is closely approximated over a sufficiently large finite horizon) 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 new method for solving this problem is proposed. It is based on the decomposition of the optimal control and off-line scheduling problem into two independent sub-problems. The first sub-problem aims at finding 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. 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 by simulation and experimentation. This algorithm allows to perform a trade-off between control performance and real-time implementation constraints.

We are of the opinion that our approach is a first step towards the problems of the state-feedback scheduling of

control tasks. Considering LTI systems, and separating control and observation problems, which are widely used assumptions in control practice, are reasonable choices in this first step. We also believe that the ideas that we proposed may be applied to more general models, including non-linear systems, because they are based on principles (for example, model predictive control) that may be generalized to non-linear systems.

Future work will focus on the generalization of the proposed approach to systems with partial state measurements.

REFERENCES

[1] K.-E. Årzén and A. Cervin, "Control and embedded computing: Survey of research directions," in *Proc. of the 16th IFAC World Congress on Automatic Control*, Prague, Czech Republic, July 2005.

[2] A. Cervin, "Integrated control and real-time scheduling," Ph.D. dissertation, Lund Institute of Technology, Sweden, April 2003.

[3] P. Martí, J. Fuertes, G. Fohler, and K. Ramamritham, "Improving quality-of-control using flexible timing constraint : Metric and scheduling issues," in *Proc. of the 23rd IEEE Real-Time Systems Symposium*, Austin, Texas, USA, December 2002.

[4] M.-M. Ben Gaid and A. Çela, "Model predictive control of systems with communication constraints," in *Proc. of the 16th IFAC World Congress on Automatic Control*, Prague, Czech Republic, July 2005.

[5] M.-M. Ben Gaid, A. Çela, and Y. Hamam, "Optimal integrated control and scheduling of systems with communication constraints," in *Proc. of the Joint 44th IEEE Conf. on Decision and Control and European Control Conf.*, Seville, Spain, December 2005.

[6] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén, "Feedback-feedforward scheduling of control tasks," *Real-Time Systems*, vol. 23, no. 1, pp. 25–53, July 2002.

[7] D. Simon, D. Robert, and O. Sename, "Robust control / scheduling co-design: application to robot control," in *Proc. of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Francisco, USA, March 2005.

[8] F. Xia and Y. Sun, "Control-scheduling codesign: A perspective on integrating control and computing," To appear in Journal of Dynamics of Continuous, Discrete and Impulsive Systems, Series B, 2006.

[9] M. Andersson, D. Henriksson, and A. Cervin, TRUETIME *1.3—Reference Manual*, Department of Automatic Control, Lund Institute of Technology, Sweden, June 2005.

[10] M.-M. Ben Gaid, A. Çela, Y. Hamam, and C. Ionete, "Optimal scheduling of control tasks with state feedback resource allocation," in *Proc. of the 2006 American Control Conf.*, Minneapolis, Minnesota, USA, June 2006.

[11] R. Dorf, M. Farren, and C. Phillips, "Adaptive sampling frequency for sampled-data control systems," *IRE Trans. on Automatic Control*, vol. 7, no. 1, pp. 38–47, 1962.

[12] S. C. Gupta, "Adaptive gain and adaptive sampfing sampled-data systems," in *Proc. of the IEEE Winter General Meeting*, New York, USA, January 1963.

[13] R. Tomovic and G. Bekey, "Adaptive sampling based on amplitude sensitivity," *IEEE Trans. on Automatic Control*, vol. 11, no. 2, pp. 282–284, April 1966.

[14] J. R. Mitchell and W. L. McDaniel Jr., "Sensitivity of discrete systems to variation of sampling interval," *IEEE Trans. on Automatic Control*, vol. 14, no. 2, pp. 200–201, April 1969.

[15] M. J. Smith, "An evaluation of adaptive sampling," *IEEE Trans. on Automatic Control*, vol. 16, no. 3, pp. 282–284, June 1971.

[16] T. C. Hsia, "Analytic design of adaptive sampling control law in sampled-data systems," *IEEE Trans. on Automatic Control*, vol. 19, no. 1, pp. 39–42, February 1974.

[17] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, "On task schedulability in real-time control systems," in *Proc. of the 17th IEEE Real-Time Systems Symposium*, New York, USA, December 1996.

[18] J. Eker, "Flexible embedded control systems. design and implementation," Ph.D. dissertation, Department of Automatic Control, Lund Institute of Technology, Sweden, December 1999.

[19] H. Rehbinder and M. Sanfridson, "Integration of off-line scheduling and optimal control," in *Proc. of the 12th Euromicro Conf. on Real-Time Systems*, Stockholm, Sweden, June 2000.

[20] L. Palopoli, C. Pinello, A. Bicchi, and A. Sangiovanni-Vincentelli, "Maximizing the stability radius of a set of systems under real-time scheduling constraints," *IEEE Trans. on Automatic Control*, vol. 50, no. 11, pp. 1790–1795, November 2005.

[21] B. Lincoln and B. Bernhardsson, "LQR optimization of linear system switching," *IEEE Trans. on Automatic Control*, vol. 47, no. 10, pp. 1701–1705, October 2002.

[22] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real-time scheduling theory: A historical perspective," *Real-Time Systems*, vol. 28, no. 2–3, pp. 101–155, November 2004.

[23] C. Lu, J. Stankovic, G. Tao, and S. Son, "Feedback control real-time scheduling: Framework, modeling and algorithms," *Special issue of Real-Time Systems Journal on Control-Theoretic Approaches to Real-Time Computing*, vol. 23, no. 1/2, pp. 85–126, Jul/Sept 2002.

[24] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic scheduling for flexible workload management," *IEEE Trans. on Computers*, vol. 51, no. 3, pp. 289–302, March 2002.

[25] X. Liu, L. Sha, M. Caccamo, and G. Buttazzo, "Online control optimization using load driven scheduling," in *Proc. of the 39th IEEE Conf. on Decision and Control*, Sydney, Australia, December 2000.

[26] G. Buttazzo, M. Velasco, P. Martí, and G. Fohler, "Managing quality-of-control performance under overload conditions," in *Proc. of the 16th Euromicro Conf. on Real-Time Systems*, Catania, Italy, July 2004.

[27] P. Martí, C. Lin, S. Brandt, M. Velasco, and J. Fuertes, "Optimal state feedback based resource allocation for resource-constrained control tasks," in *Proc. of the 25th IEEE Real-Time Systems Symposium*, Lisbon, Portugal, December 2004.

[28] D. Henriksson and A. Cervin, "Optimal on-line sampling period assignment for real-time control tasks based on plant state information," in *Proc. of the Joint 44th IEEE Conf. on Decision and Control and European Control Conf.*, Seville, Spain, December 2005.

[29] J. W. S. Liu, *Real-Time Systems*. Prentice Hall, 2000.

[30] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén, "How does control timing affect performance?" *IEEE Control Systems Magazine*, vol. 23, no. 3, pp. 16–30, June 2003.

[31] P. Martí, "Analysis and design of real-time control systems with varying control timing constraints," Ph.D. dissertation, Technical University of Catalonia, 2002.

[32] P. Khargonekar and N. Sivashankar, "$\mathcal{H}_2$ optimal control for sampled-data systems," *Systems and Control Letters*, vol. 17, no. 6, pp. 424–436, 1991.

[33] C. Ionete and A. Çela, "Structural properties and stabilization of NCS with medium access constraints," in *Proc. of the 45th IEEE Conf. on Decision and Control*, San Diego, California, USA, December 2006.

[34] M.-M. Ben Gaid, A. Çela, and Y. Hamam, "Optimal integrated control and scheduling of networked control systems with communication constraints: Application to a car suspension system," *IEEE Trans. on Control Systems Technology*, vol. 14, no. 4, pp. 776–787, July 2006.

[35] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. Wiley, 1996.

[36] L. Xie, H. Zhou, and C. Zhang, "H_2 optimal deconvolution of periodic IIR channels: an LMI approach," in *Proc. of the 6th International Symposium on Signal Processing and its Applications*, Kuala-Lumpur, Malaysia, August 2001.

[37] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[38] M.-M. Ben Gaid, "Optimal scheduling and control for distributed real-time systems," Ph.D. dissertation, Université d'Evry Val d'Essonne, France, November 2006.

[39] T. Chen and B. A. Francis, *Optimal Sampled-Data Control Systems*. Springer-Verlag, 1995.

[40] Allen-Bradley, "Compact logix system (catalog numbers 1769-l20 and 1769-l30) : user manual," Publication 1769-UM007C-EN-P, June 2001.

[41] K. J. Åström and B. Wittenmark, *Computer-controlled systems: theory and design*. Prentice Hall, 1997.

Mohamed El Mongi Ben Gaid was born in Tunis, Tunisia, in 1978. He received the Dipl.Ing. degree in electrical engineering from the National School of Engineers of Tunis, Tunisia, in 2002, the M.S. degree in distributed computing from the University of Paris XI, Orsay, France, in 2003, and the Ph.D. degree in control and computer engineering from the University of Evry Val d'Essonne, Evry, France, in 2006. His PhD research was conducted at the Embedded Systems department of ESIEE Paris, under the supervision of Profs. A. Çela and Y. Hamam.

He is currently a R&D engineer in real-time computing at the Technology, Computer Science and Applied Mathematics Division of the Institut Français du Pétrole (IFP), the French national research center in energy, transportations and environment. Prior to this, he spent one year as a postdoctoral researcher at the Networked Control Systems project-team of INRIA Rhône-Alpes. His research interests include real-time control systems, process monitoring with application to engine test beds and multi-domain simulation.

Arben S. Çela received the engineering diploma from the Polytechnic University of Tirana, Albania in 1984. From 1984 to 1986 he was with the Department of Power Generation of Faculty of Engineering of Polytechnic University of Tirana, Albania. In 1992, he obtained the Ph.D. from the University of Paris XI-Orsay. From 1988 he is with the Embedded Systems Department of the Paris-East University, Noisy Le Grand France, where he is actually head of the department. His research interests include optimisation, automatic control, dynamic system theory, hybrid systems, networked control, and relation between network and communication.

Yskandar Hamam graduated as a Bachelor of Electrical Engineering from the American University of Beirut (AUB) in 1966. He then obtained his M.Sc. in 1970 and Ph.D. in 1972 from the University of Manchester Institute of Science and Technology. He also obtained his "Diplôme d'Habilitation à Diriger des Recherches" (equivalent to D.Sc.) from the "Université des Sciences et Technologies de Lille" in 1998. He conducted research activities and lectured in England, Brazil, Lebanon, Belgium and France. He was the head of the Control department and dean of faculty at ESIEE, France. He was an active member in modeling and simulation societies and was the president of EUROSIM. He is presently Emeritus Professor at ESIEE and is a member of the A2SI laboratory of ESIEE and a permanent member of the LISV laboratory of the "Université de Versailles Saint Quentin en Yvelines". He occupies presently the position of Scientific Director of French South African Technical Institute of Electronics at the Tshwane University of Technology in South Africa.