Parameter Selection for Real-Time Controllers in Resource-Constrained Systems

Yifan Wu, *Student Member, IEEE*, Giorgio Buttazzo, *Senior Member, IEEE*, Enrico Bini, *Member, IEEE*, and Anton Cervin, *Member, IEEE*

***Abstract*—In resource-constrained systems, the interference generated by the concurrent execution of multiple controller tasks leads to extra delay and jitter, which degrade control performance and may even jeopardize the stability of the controlled system. This work presents a general methodology that integrates control issues and real-time schedulability analysis to improve the control performance in embedded systems with time and resource constraints. The performance increase is achieved by properly selecting task periods and deadlines under feasibility constraints.**

***Index Terms*—Control performance, embedded systems, real-time control, real-time systems.**

I. Introduction

In MODERN control systems, the control algorithm is normally implemented as a periodic task performing activities such as sensory sampling, control signal calculation, state updating, and actuation. The period of such a task is typically derived in accordance with the traditional discrete-time control theory, which analyzes the system behavior and guarantees stability based on the periodicity assumption.

To ensure the timely behavior on a particular execution platform, multiple periodic controllers are usually scheduled by a real-time scheduling policy, such as Rate Monotonic (RM) or Earliest Deadline First (EDF) [1]. When multiple tasks run concurrently on a computing platform with scarce resources, however, possible overload conditions could cause tasks to run at frequencies lower than initially assumed in the control design phase, possibly jeopardizing the system performance and stability. Such a situation can easily be prevented offline by performing schedulability analysis under worst-case conditions. Nevertheless, even when schedulability is guaranteed under the specified timing requirements, the reciprocal interference caused by the concurrent access to shared resources may introduce variable delays and jitter in task executions. Such an extra platform-induced interference, if not properly considered, could lead to significant performance degradation, or even instability and system failure [2], [3]. To avoid such problems, it is essential to design the whole system considering both the control aspects and the execution platform aspects at the same time, rather than at subsequent design stages. Such an integrated approach is referred to as *real-time/control co-design* and a lot of research has been focused on this area during the last decade.

To distribute the limited computing resources to different controller tasks, Seto *et al.* [4] proposed to formulate the real-time control co-design problem as an optimization problem, where the control performance index, expressed as a function of the sampling period, is constrained by the feasibility condition of the task set. By solving the optimization problem, the sampling period for each controller is computed to maximize the overall system performance. This methodology, further extended by many researches, is referred to as the *period selection problem*. Bini and Di Natale [5] applied Seto's methodology to a set of controller tasks scheduled by Fixed Priorities.

To cope with the problem of delay and jitter in real-time control applications, different techniques have been developed. Nilsson [3] analyzed the performance and stability of real-time control systems with varying delays, and derived an optimal stochastic controller to compensate for jitter. Cervin *et al.* [6] introduced the concept of *jitter margin*, defined as the upper bound of the input-output jitter of a control task that guarantees the stability of the controlled system. Martí *et al.* [7] presented an online method to compensate the control performance degradation caused by jitter. Another approach for reducing delay and jitter is to use nonpreemptive or limited-preemptive scheduling policies [8], [9]. For example, Wu and Bertogna [10] discussed the benefits of using EDF with limited preemptions to reduce input-output delay and jitter without impairing the schedulability of the task set.

Another widely adopted method to reduce delay and jitter is to limit the execution interval of each task by setting a proper relative deadline. Like *period selection*, this method can be referred to as *deadline selection*. Different algorithms for computing the minimum deadline have been proposed in the literature. Some methods [11], [12] allow minimizing the relative deadline of a single task at a time, following a given order. In this way, however, the first task in the sequence experiences the most significant deadline reduction, leaving little slack for the remaining tasks. A more uniform deadline reduction can be achieved by scaling all deadlines by the same factor [12], but the improvement achieved in terms of delay and jitter is not significant and, in some cases, the schedule could even remain unchanged. Other methods [13], [14] use binary search to reduce

Manuscript received December 14, 2009; revised March 20, 2010, May 18, 2010, and May 26, 2010; accepted June 02, 2010. Date of publication July 19, 2010; date of current version November 05, 2010. Paper no. TII-09-12-0352.

Y. Wu, G. Buttazzo, and E. Bini are with Scuola Superiore Sant'Anna, 56127 Pisa, Italy (e-mail: y.wu,g.buttazzo,e.bini@sssup.it).

A. Cervin is with the Department of Automatic Control, Lund University, Lund S-221 00, Sweden (e-mail: anton@control.lth.se).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2010.2053378

task relative deadlines as much as possible according to given reduction factors, while keeping the task set schedulable. These methods, however, are mainly focused on schedulability aspects and barely considered control issues; moreover, it is not clear how reduction factors can be assigned to tasks.

Different delay/jitter reduction methods have been discussed and compared in [15], where it is shown that the effectiveness of a particular method depends on the characteristic of the controlled system, although the deadline reduction approach is the simplest and most effective for most control systems.

Ryu and Hong [16] used a heuristic method to select periods and deadlines with respect to performance specification and schedulability constraints. The control performance was specified in terms of steady state error, overshoot, settling time, and rise time, which were expressed as functions of the sampling period and input-output latency. At each step of the heuristic method, the periods and deadlines were derived using the Period Calibration Method solving a nonlinear optimization problem. The optimization goal, however, was to minimize the utilization of the task set.

Kim [17] suggested to express the control cost as a function of both periods and delays, where periods were found assuming that the delays were given. Then, the new delays were computed by simulating the schedule of all the tasks up to the hyperperiod, and iteratively the periods were updated assuming the new delay values. However, this method considered only fixed priorities and was extremely time consuming.

Palopoli *et al.* [18] proposed to use resource reservation to serve control tasks as soft real-time threads. It was revealed that control tasks may tolerate a certain amount of deadline misses owing to their inherent robustness, therefore relaxing the hard timing constraints allows higher activation rates, which may lead to improved performance. However, no optimization was performed to select reservation parameters and only experimental results were presented.

Chantem *et al.* [19] proposed a heuristic search algorithm to find feasible period-deadline pairs, based on the assumption that task deadlines are piecewise first-order differentiable functions of their respective periods. However, this work mainly focused on schedulability issues.

Bini and Cervin [20] approximated the delays as a function of task periods and incorporated the delay consideration into the performance optimization, while the resource constraint remains to be the feasibility region with respect to task periods. This method only applies to fixed priority systems, because in dynamic priority systems delays are functions of both periods and deadlines.

Wu *et al.* [21] proposed a general framework for real-time control design in embedded environment considering the sampling period, delay, and jitter effect in the control performance evaluation, and manifested the possibility to form an optimization problem with the help of EDF deadline space. However, the proposed linkage between timing attributes and task parameters was not clear, and a formal approach for parameter selection was missing.

This work presents an integrated approach to enhance the control performance of a system through proper selection of task periods and deadlines, under EDF scheduling. A general framework is proposed to extend Seto's method to optimize performance with respect to not only sampling periods but also other timing attributes. In particular, task deadlines are chosen to balance the scheduling-induced performance loss of each controller task exploiting the feasibility region in the space of EDF deadlines [22]. Detailed simulations are also provided to demonstrate the usage of the proposed methodology and verify its effectiveness over other methods.

The rest of this paper is organized as follows. Section II presents the system model and the terminology. Section III defines the problem to be solved. Section IV describes the general method to formalize the optimization problem. Section V explains the characterization of the resource constraints. Section VI presents the experimental results and compares the proposed method with other approaches. Finally, Section VII states our conclusions.

II. System Model

This work considers a set τ of n periodic real-time tasks that are executed on a uniprocessor system under the Earliest Deadline First (EDF) scheduling policy. The task set τ is logically divided into two subsets: one subset τ_{ctrl}, consisting of n_{ctrl} controller tasks, and another subset $\tau_{n\text{ctrl}}$, consisting of $n_{n\text{ctrl}}$ regular tasks that are not related to control. Each task τ_i is characterized by the following *scheduling parameters*:

C_i	the worse-case execution time (WCET);
C_i^b	the best-case execution time (BCET);
$D_i^{\min}$	the minimum allowed relative deadline;
$D_i^{\max}$	the maximum allowed relative deadline;
$T_i^{\min}$	the minimum allowed period;
$T_i^{\max}$	the maximum allowed period;
D_i	the actual relative deadline, whose value has to be selected within the range $[D_i^{\min}, D_i^{\max}]$;
T_i	the actual period, whose value has to be selected within the range $[T_i^{\min}, T_i^{\max}]$. For controller tasks, T_i is also referred to as the *sampling period*.

It is assumed that $C_i, C_i^b, D_i^{\min}, D_i^{\max}, T_i^{\min}$ and $T_i^{\max}$ are known, whereas T_i and D_i are the *design parameters* to be selected.

To derive more general results, relative deadlines are allowed to be less than, equal to, or greater than periods. In particular, it has been shown that, for some controllers, shortening the period T_i below the relative deadline may improve the performance [20]. In addition, U_i denotes the task utilization, also called *bandwidth*, given by $U_i = (C_i)/(T_i)$. Accordingly, its value can range within $[U_i^{\min}, U_i^{\max}]$, where $U_i^{\min} = (C_i^b)/(T_i^{\max})$, and $U_i^{\max} = (C_i)/(T_i^{\min})$. Similarly, $U, U_{\text{ctrl}}, U_{n\text{ctrl}}$ denote the total utilization of the whole task set ($U = \sum_{i=1}^{n} U_i$), the utilization of all the controller tasks ($U_{\text{ctrl}} = \sum_{\tau_i \in \tau_{\text{ctrl}}} U_i$), and the utilization of all the regular tasks ($U_{n\text{ctrl}} = \sum_{\tau_i \in \tau_{n\text{ctrl}}} U_i$), respectively.

Each periodic task τ_i generates an infinite sequence of jobs $\tau_{i,k}, k \in \mathbb{N}$. Each job $\tau_{i,k}$ has an arrival time $a_{i,k}$ and an absolute deadline $d_{i,k} = a_{i,k} + D_i$. The start time $s_{i,k}$ of job $\tau_{i,k}$ is the

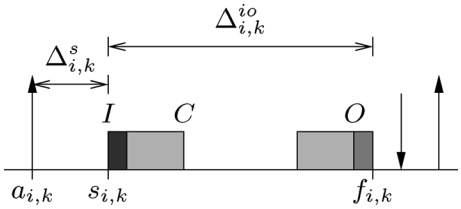

Fig. 1. Timing attributes of job $\tau_{i,k}$.

first time it is scheduled for execution, while the finishing time $f_{i,k}$ is the time at which its last instruction is terminated.

As illustrated in Fig. 1, each controller task $\tau_i \in \tau_{\text{ctrl}}$ consists of three stages of operations, denoted as *Input* (I), *Calculation* (C), and *Output* (O), corresponding to sampling, control signal calculation, and actuation, respectively. Without loss of generality, it is assumed that the *Input* occurs at the beginning of each job, i.e., at $s_{i,k}$, whereas the *Output* occurs at the end, i.e., at $f_{i,k}$.

The timing of the input and output operations may have a large impact on the control performance [23]. Hence, the following additional timing attributes for controller tasks are defined.

- The **sampling delay** $\Delta^s_{i,k}$ of a job $\tau_{i,k}$ is the time between the arrival time and the start time: $\Delta^s_{i,k} = s_{i,k} - a_{i,k}$.
- The **input-output delay** (*IO delay*) $\Delta^{\text{io}}_{i,k}$ of a job $\tau_{i,k}$ is the time between sampling and actuation, and is equivalent to the time between the start time and the finishing time: $\Delta^{\text{io}}_{i,k} = f_{i,k} - s_{i,k}$.
- The **sampling jitter** j^s_i of a task τ_i is the maximum difference between $\Delta^s_{i,k}$ of all the jobs: $j^s_i = \max_k \Delta^s_{i,k} - \min_k \Delta^s_{i,k}$.
- The **input-output jitter** (*IO jitter*) j^{io}_i of a task τ_i is the maximum difference between $\Delta^{\text{io}}_{i,k}$ of all the jobs: $j^{\text{io}}_i = \max_k \Delta^{\text{io}}_{i,k} - \min_k \Delta^{\text{io}}_{i,k}$.

The importance of these parameters will be discussed in the sections below.

III. Problem Statement

A. The Performance Loss Index

The primary goal of a control system is to meet stability and performance requirements, such as transient response and steady-state accuracy [24]. Beyond such requirements, controller design attempts to minimize the system error, defined as the difference between the desired response and the actual response. The smaller the difference, the better the performance. Hence, performance criteria are mainly based on measures of the system error. Traditional criteria (reported in control textbooks, e.g., [25]), such as Integral of the Absolute Error (IAE), Integral of Time-Weighted Absolute Error (ITAE), Integral of Square Error (ISE), or Integral of Time-Weighted Square Error (ITSE), provide quantitative measures of a control system response and are used to evaluate (and design) controllers.

More sophisticated performance criteria, mainly used in optimal control problems, account both for the system error and for the energy that is spent to accomplish the control objective. The higher the energy demanded by the controller, the higher the penalty paid in the performance criterion. The system error and control energy can be multiplied by a weight to balance their relative importance.

The performance index used in this work is the same as the one used in Linear Quadratic Gaussian (LQG) controller design (e.g., [26]). The performance of a controller task is given by a quadratic cost function

$$J = \mathrm{E} \lim_{t_p \to \infty} \frac{1}{t_p} \int_0^{t_p} (x' Q_1 x + u' Q_2 u) dt \tag{1}$$

where x and u denote the state vector and the control signal vector, x' and u' denote the corresponding transpose vectors, t_p is the maximum time to be considered in the performance evaluation, Q_1, Q_2 are weighting matrices, and E denotes the expectation operator. The performance J can be interpreted as the weighted sum of state errors and control energy. Higher values of J indicate larger deviation from the desired states or larger energy spent for control, which means worse control performance. For this reason, in the remainder of this paper, J is referred to as the *performance loss index*. For discrete-time control, both the state x and the control signal u depend on the sampling period T. Hence, the performance loss index J can also be written as a function of the period T as follows:

$$J = J(T).$$

In most realistic cases, for a reasonable range of sampling intervals, the performance loss (1) is an increasing function of the sampling period. Cervin *et al.* [27] argued that the performance loss index can often be approximated by a linear function of the sampling period

$$J \approx \alpha + \beta T$$

or by a quadratic function of the sampling period

$$J \approx \alpha + \beta T + \gamma T^2.$$

Delay and jitter in the controller task execution can have a large impact on the control performance, especially if the sampling frequency is too low compared to the speed of the closed-loop system. It would hence be desirable to include the delay and jitter in the performance loss index. The relationship between these timing attributes and the resulting control performance is, however, very complex. The solution proposed in this work is to include the relative deadline D in the cost function

$$J = J(T, D). \tag{2}$$

As will be shown in Section IV, the relative deadline D upper limits the amount of delay and jitter the controller can experience. Knowing T and D, it is hence possible to predict the worst-case performance degradation introduced by the scheduling. In general, $J(T, D)$ is a nonlinear function. It is realistic to assume that it is an increasing function in both T and D, since the control performance typically degrades as the sampling period, delay, or jitter increases, as later shown in Fig. 8.

B. The Optimization Problem

The *period selection* problem has received considerable attention in the real-time literature. It can be expressed as an optimization problem to find the best periods for the controller tasks that minimize the performance loss while guaranteeing

the system schedulability. Such an optimization problem under EDF can be expressed as follows:

$$\min_{\{T_i\}} J = \sum_{\tau_i \in \tau_{\text{ctrl}}} J_i(T_i)$$
$$\text{s.t.} \quad \sum_{\tau_i \in \tau_{\text{ctrl}}} \frac{C_i}{T_i} + \sum_{\tau_i \in \tau_{n\text{ctrl}}} \frac{C_i}{T_i} \leq 1$$

where the objective function is the sum of all the controller tasks' performance indices, which are assumed to be function of the sampling period. The equation of the constraint imposes the schedulability constraint for the given scheduling policy (EDF).

To take the impact of delay and jitter on control performance into account, the relative deadlines are included in the performance loss indices and the optimization problem is generalized to

$$\min_{\{T_i, D_i\}} J = \mathcal{F}_{\tau_i \in \tau_{\text{ctrl}}}(J_i(T_i, D_i))$$
$$\text{s.t.} \quad \{T_i, D_i\} \in \mathcal{S}, \quad \forall\, \tau_i \in \tau \tag{3}$$

where $\mathcal{F} : \mathbb{R}^{n_{\text{ctrl}}} \rightarrow \mathbb{R}$ is a system-wide function used to combine the individual performance indices of control tasks into a global system performance index, and $\mathcal{S}$ is the space of feasible parameter values that guarantee schedulability. The choice of function $\mathcal{F}$ depends on the user's interest and can be, for instance, a linear combination of all the individual performance loss indices, or the maximum among the performance loss indices.

IV. Linking Task Parameters to Control Performance

This section explains how to derive the performance loss index given in (2) in a simulative or experimental fashion, describes the relation between control performance and scheduling parameters, and formalizes the optimization problem expressed by (3).

A. Characterization of the Delay and Jitter

Assuming that the task set is schedulable, each job will finish no later than its absolute deadline. This puts a limit on the amount of delay and jitter that a controller task with period T_i and relative deadline D_i can experience.

Consider the worst-case scenario depicted in Fig. 2. In this scenario, task τ_i releases 3 consecutive jobs, where job $\tau_{i,0}$ finishes with best-case execution time C_i^b, job $\tau_{i,1}$ starts at its release time and finishes at its deadline, and finally, job $\tau_{i,2}$ starts $D_i - C_i$ before its deadline to ensure that it will not cause an overrun. By analyzing the worse-case scenario, the following bounds on the delays can be derived:

$$\begin{aligned}
\max \Delta^{\text{io}}_{i,k} &= \Delta^{\text{io}}_{i,1} \leq D_i \\
\min \Delta^{\text{io}}_{i,k} &= \Delta^{\text{io}}_{i,0} \geq C_i^b \\
\max \Delta^{s}_{i,k} &= \Delta^{s}_{i,2} \leq D_i - C_i \\
\min \Delta^{s}_{i,k} &= \Delta^{s}_{i,1} \geq 0.
\end{aligned} \tag{4}$$

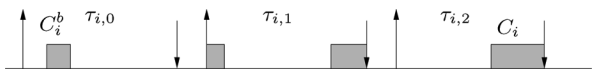

Fig. 2. Worst-case scenario for delay and jitter.

Also, the following relations on the jitter hold:

$$\begin{aligned}
j^{\text{io}}_i &\leq D_i - C_i^b \\
j^{s}_i &\leq D_i - C_i.
\end{aligned} \tag{5}$$

Notice that the reported worst-case scenario must not necessarily take place in an actual schedule, since the interference on task τ_i depends on the scheduling parameters of other tasks as well. Therefore, the relations derived above represent only the lower or upper bounds of the actual delay and jitter.

The analysis above shows that a shorter relative deadline implies both shorter delays and less jitter, which should imply better control performance. How to find the actual performance loss index is treated next.

B. Performance Loss Index Derivation

In most cases, it is impossible to evaluate the exact value of the performance loss index (1) for a controller executing in a real-time system. An execution of the real-time system will generate an infinite sequence of sampling and input-output delays $\{\Delta^s_{i,0}, \Delta^{\text{io}}_{i,0}, \Delta^s_{i,1}, \Delta^{\text{io}}_{i,1}, \Delta^s_{i,2}, \Delta^{\text{io}}_{i,2}, \ldots\}$, for each controller task τ_i. The delays are in general random and depend on the execution-time characteristics of the control algorithm and the preemption pattern created by the scheduling algorithm, which in turn depend on the execution of the other tasks in the system.

Using the bounds on the delay and jitter derived in the previous subsection, various approaches can be used to evaluate the performance loss index approximately.

- Taking a stochastic approach, one can assume that $\{\Delta^s_{i,j}, \Delta^{\text{io}}_{i,j}\}_{j=0}^{\infty}$ describe a sequence of independent two-dimensional uniform random variables with bounds given by (4). The performance index can then be evaluated numerically using a tool such as Jitterbug [28]. A limitation of Jitterbug, however, is that the maximum delay variation allowed is bounded by the sampling period. Hence, it is not possible to evaluate the case when $D_i > T_i$.
- Taking a worst-case approach, one may try to evaluate the largest theoretically possible performance degradation given the delay bounds (4). For the case of pure input-output jitter, the jitter margin [6] can be used. Unfortunately, however, no performance degradation theorem for mixed sampling jitter and input-output jitter exists today.
- A third option, which is advocated in this paper, is to perform a quantitative analysis with respect to delay and jitter to determine which factor has the larger influence on the performance degradation. Such an analysis can be carried out using Jitterbug, simulation (using tools like TrueTime [29] or RTSim [18], [30]), or by experiments on the real system.

The last option is elaborated upon in the rest of this section.

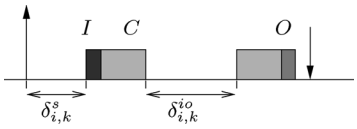

Fig. 3. Inserting artificial delays.

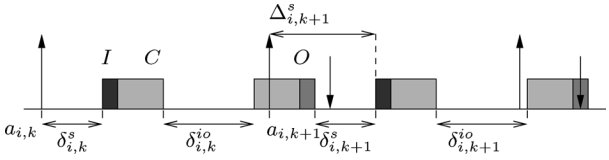

Fig. 4. Problem when deadlines are larger than periods.

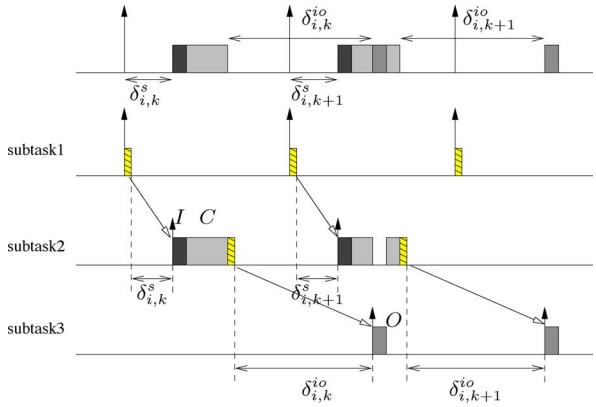

Fig. 5. Sequence of subtasks to generate delays larger than periods.

C. Quantitative Performance Degradation Analysis

As mentioned above, an approximative performance loss index for a controller task can be derived in a simulative or experimental fashion. When the system model is not available or it is not accurate, the control performance can be directly monitored using a real-time kernel, like S.Ha.R.K [31], that allows enforcing desired and precise delays in task executions. In other words, instead of deriving the control performance by running a real task set under different workload conditions, a single task is executed at a time, as a function of configurable timing attributes, simulating the interference by injecting artificial delays in the task execution.

The most intuitive solution to generate a sampling delay is to defer the start time of the job of the controller task by inserting a delay primitive before the input procedure. Similarly, the input-output delay can be introduced by inserting a delay primitive before the output procedure, as shown in Pseudocode 1, where $\delta^s_{i,k}$ and $\delta^{\rm io}_{i,k}$ represent the injected artificial sampling delay and the IO delay for each job $\tau_{i,k}$, respectively. Fig. 3 illustrates this intuitive method. Notice that, assuming *Input* and *Output* operations consume negligible computation times, the actual input-output delay is $\Delta^{\rm io}_{i,k} = \delta^{\rm io}_{i,k} + C_i$, while the actual sampling delay is always equal to the artificial one, that is $\Delta^s_{i,k} = \delta^s_{i,k}$.

Pseudocode 1 Controller_Task τ_i

```
1: Delay(δ^s_{i,k})
2: sampled-data ← Input()
3: control-signal ← Calculation(sampled-data)
4: Delay(δ^io_{i,k})
5: Output(control-signal)
```

A problem with this implementation is that, when deadlines are larger than periods, delays can be larger than expected, as depicted in Fig. 4. In fact, when the kth job of task τ_i completes after the beginning of the next period, the actual sampling delay results to be higher than the specified $\delta^s_{i,k+1}$, and in particular equal to

$$\Delta^s_{i,k+1} = \delta^s_{i,k+1} + f_{i,k} - a_{i,k+1}.$$

To solve this problem, the controller task is split into three subtasks: a periodic subtask and two aperiodic subtasks, as illustrated in Fig. 5. At the end of each job of the periodic subtask (*subtask1*), a system-level event is posted to activate the first aperiodic subtask (*subtask2*) after a given amount of time, equal to the specified sampling delay $\delta^s_{i,k}$. Such an aperiodic subtask performs *Input* and *Calculation* and then it posts another system-level event to activate the second aperiodic subtask (*subtask3*) after the specified input-output delay $\delta^{\rm io}_{i,k}$. The second aperiodic subtask performs the *Output* and finishes the control job. The two aperiodic subtasks are scheduled with a lower priority with respect to the periodic task to ensure the proper activation sequence.

The timeline at the top of the figure shows the equivalent execution of the controller task with the proper enforced delays. It can be easily seen that, except for a negligible overhead due to the subtask activation, the specified sampling delay $\delta^s_{i,k}$ and input-output delay $\delta^{\rm io}_{i,k}$ are not affected by the task finishing time. It is worth mentioning that the second aperiodic subtask is assigned a priority higher than that of the first aperiodic subtask, because the *Output* is less time consuming and should not be preempted by the execution of the first aperiodic subtask. Also notice that this approach allows generating tasks with arbitrary jitter as well, obtained by introducing random activation delays in the subtasks.

The pseudocode of the controller subtasks is listed in Pseudocodes 2, 3, and 4, where Post_Kernel_Event(t, e) is a function that posts a system-level event e at time t, and $t_{\rm cur}$ is the current system time.

Pseudocode 2 Subtask1 of τ_i

```
1: Post_Kernel_Event(
     t_cur + δ^s_{i,k},
     event_activate_subtask2
   )
```

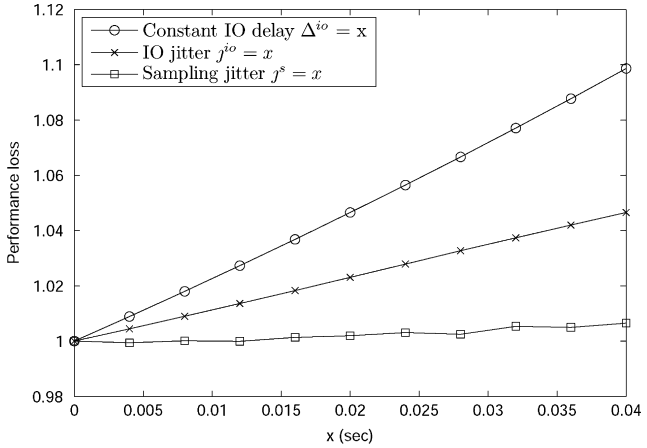

Fig. 6. Comparison of the influence of delay and jitter for a double integrator with $T = 0.02$ s.

Pseudocode 3 Subtask2 of τ_i

1: *sampled-data* ← Input()
2: *control-signal* ← Calculate(*sampled-data*)
3: Post_Kernel_Event(

$t_{\text{cur}} + \delta^{\text{io}}_{i,k}$,

event_activate_subtask3

)

Pseudocode 4 Subtask3 of τ_i

1: Output(*control-signal*)

D. Example of Analysis Results

As an example of the quantitative performance analysis, the LQG control of a double integrator process with the sampling interval $T = 0.02$ s is studied. Fig. 6 illustrates the performance loss as a function of the sampling jitter, input-output delay, and input-output jitter, when each parameter (reported as x on the x axis) is varied alone, while keeping the others equal to zero.

The values of x can be as large as twice the sampling period. Notice that the constant sampling delay is not considered in the comparison, since a task τ_i where all jobs have a constant sampling delay Δ^s_i is equivalent to a task with a release offset of Δ^s_i and sampling delay equal to 0 for all jobs. It is seen that, in this case, the IO delay is the most significant timing attribute influencing the control performance. Hence, the worst case respecting the bounds (4) occurs when $\Delta^{\text{io}}_i = D_i$ and $j^s_i = j^{\text{io}} = 0$.

V. Resource Constraints Characterization

Since the performance loss index is assumed to decrease as the period or the deadline of the controllers decrease, the solution of the design problem is to find the smallest values for T_i and D_i that guarantee schedulability.

To determine the feasible task parameters under EDF, the processor demand criterion proposed by Baruah *et al.* [32] is used. According to this test, a task set is schedulable by EDF if and only if

$$\begin{cases} \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \\ \forall t \in \text{dlSet} \ \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t - D_i + T_i}{T_i} \right\rfloor\right\} C_i \leq t \end{cases} \tag{6}$$

where dlSet is the set of time instances in which the feasibility test has to be performed.

Unfortunately, this test does not provide a description of the feasible parameters that is well suited for maximizing the performance. In fact, since periods and deadlines appear within the floor operator, the shape of the boundary necessary to apply constrained optimization techniques (such as the Lagrange multipliers) is not easy to derive. Performing a global optimization would require the execution of algorithms, such as Simulated Annealing, which turned out to be extremely time consuming and not scalable with the size of the problem, while the number of the constraints in the proposed method is quadratic with the number of tasks, as shown later in this section.

To overcome such a problem, the following two-step approach is adopted.

1) First, consider $D_i = T_i$ for all the tasks and find the periods that minimize the performance loss index, using the Liu and Layland necessary and sufficient test for EDF

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \tag{7}$$

which is linear and it can be used in the optimization process [4].

2) Then, fix the task periods as derived in the previous step, relax the assumption $D_i = T_i$, and perform the optimization in the space of the feasible deadlines [22].

As shown by Bini and Buttazzo [22], the difference between the exact region of feasible deadlines and its convex approximation is not large when periods are far from harmonic relations. Since this is the typical condition resulting from step 1, the consequent performance loss is also small. A better performance could indeed be achieved by selecting periods with a higher degree of harmonicity, but this is not trivial and is still an open issue.

Due to the convexity of the constraint of (7), if the performance loss index can be approximated by a convex function (such as linear, exponential or logarithmic), then the first step can be solved by standard convex optimization techniques [33]. For the cases in which the performance loss index cannot be tightly approximated by a convex function, global optimization techniques, such as Simulated Annealing, must be used.

The second step can be accomplished by exploiting the geometric properties of the space of feasible deadlines. Bini and Buttazzo [22] proved that given the computation times $\mathbf{C} = (C_1, \ldots, C_n)$ and the periods $\mathbf{T} = (T_1, \ldots, T_n)$, the region of the feasible deadline can be expressed as follows:

$$\mathbb{S} = \bigcap_{\mathbf{k} \in \mathbb{N}^n} \bigcup_{i: k_i \neq 0} \{\mathbf{D} \in \mathbb{R}^n : D_i \geq \mathbf{k} \cdot \mathbf{C} - (k_i - 1) T_i\}. \tag{8}$$

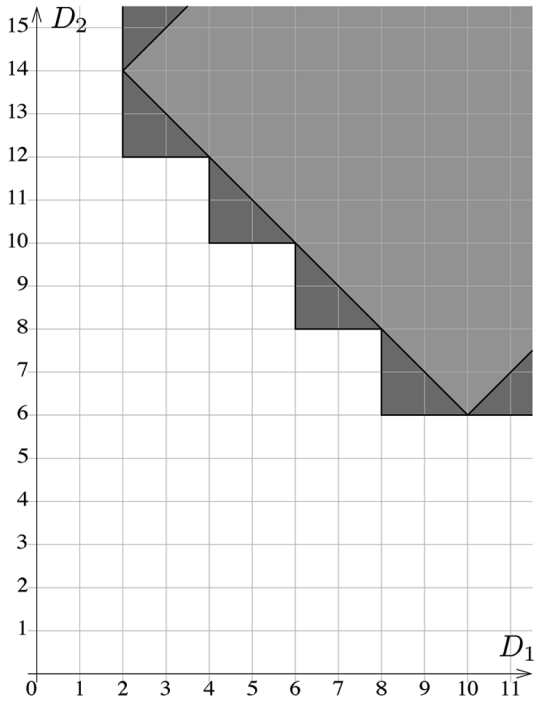

Fig. 7. Exact space of feasible deadlines (union of the shadowed regions) and its convex approximation (light gray region).

To clarify the geometry of the space of feasible deadlines we propose an example with two periodic tasks, whose parameters are $\mathbf{C} = (2, 6)$ and $\mathbf{T} = (4, 12)$. According to (8), the resulting space of feasible deadlines is illustrated in Fig. 7 as the union of the shadowed regions.

Since the performance always improves as deadlines become smaller (i.e., $(\partial J_i)/(\partial D_i) \leq 0$), then all the corners of the region of the feasible deadlines are a local optima. An optimization routine should then test the performance value at all these local optima and select the best performing solution. In the example shown in Fig. 7, local optima are in the set $\mathbb{S} = \{(8, 6), (6, 8), (4, 10), (2, 12)\}$.

Unfortunately, the cardinality of the set of local optima does not increase polynomially with the number of tasks, hence this method can be time consuming for large task sets. An alternative solution is to use a convex subregion of the exact space. In [22], it is proved that if the following set of linear constraints are satisfied:

$$\begin{cases} D_i - D_j \leq T_i & \forall i, j \\ D_j \left(1 - \sum_{i=1}^{n} U_i\right) + \sum_{i=1}^{n} U_i D_i \geq \sum_{i=1}^{n} C_i & \forall j \end{cases}$$

then the resulting deadline assignment is feasible. Notice that the number of the linear constraints is n^2. Moreover, if in the first step of the optimization procedure the periods are assigned such that the total utilization $\sum_i U_i$ reaches 1 (i.e., the computing resource is fully exploited), the convex constraint becomes

$$\begin{cases} D_i - D_j \leq T_i & \forall i, j \\ \sum_{i=1}^{n} U_i D_i \geq \sum_{i=1}^{n} C_i \end{cases} \tag{9}$$

whose region is delimited by $n \cdot (n-1)+1$ linear constraints. In Fig. 7 the convex subregion is depicted in light gray. Although (9) provides only a sufficient test, the convexity of the region allows implementing a very efficient algorithm for finding a deadline assignment.

VI. Experimental Results

This section illustrates how the proposed methodology can be used for selecting periods and deadlines in a system consisting of both controller tasks and regular tasks. The overall performance of the system is evaluated by simulating the runtime of the whole system scheduled by EDF on a uniprocessor using TrueTime [29] in Matlab.

A. The Control Systems

Two types of plant have been considered with highly different dynamics to control. The first type, denoted as **Plant A**, is a double integrator with the following state-space model:

$$\frac{dx}{dt} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u + \begin{bmatrix} 1 \\ 0 \end{bmatrix} v$$
$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} x + \sqrt{0.1} e.$$

The cost function used for both LQG design [26] and control performance evaluation is

$$J = \mathrm{E} \lim_{t_p \to \infty} \frac{1}{t_p} \int_0^{t_p} \left(x' \begin{bmatrix} 0 & 0 \\ 0 & 10 \end{bmatrix} x + u^2 \right) dt.$$

The second type, denoted as **Plant B**, has the following state-space model:

$$\frac{dx}{dt} = \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 35 \\ -61 \end{bmatrix} v$$
$$y = \begin{bmatrix} 2 & 1 \end{bmatrix} x + e$$

with its corresponding quadratic cost function

$$J = \mathrm{E} \lim_{t_p \to \infty} \frac{1}{t_p} \int_0^{t_p} \left(x' \begin{bmatrix} 700 & 20\sqrt{35} \\ 20\sqrt{35} & 20 \end{bmatrix} x + u^2 \right) dt.$$

This plant is a modification of the one investigated in [3], where the LQG design results in a controller that is extremely sensitive to delay and jitter.

For all the plant models, x is the system state vector, u is the control signal, v is a continuous-time zero-mean white noise process with unit intensity, and e is a discrete-time zero-mean white noise process with unit variance. In the cost function, $[0, t_p]$ is the time span to be considered. Although t_p should be ∞ in LQG design, when evaluating control performance, it is reasonable to use a suitable large value, which in this case was set to 50 s, also equal to the simulation time of the experiments.

The control performance loss index with respect to sampling period and relative deadline was derived for both types of plant. To obtain such an index, a performance derivation procedure using the method in Section IV-B was set up in TrueTime. The adjustable ranges of sampling periods were set to [4, 20] ms for **Plant A** and [30, 70] ms for **Plant B**, respectively. For both types of plant, the values of different timing attributes can be as

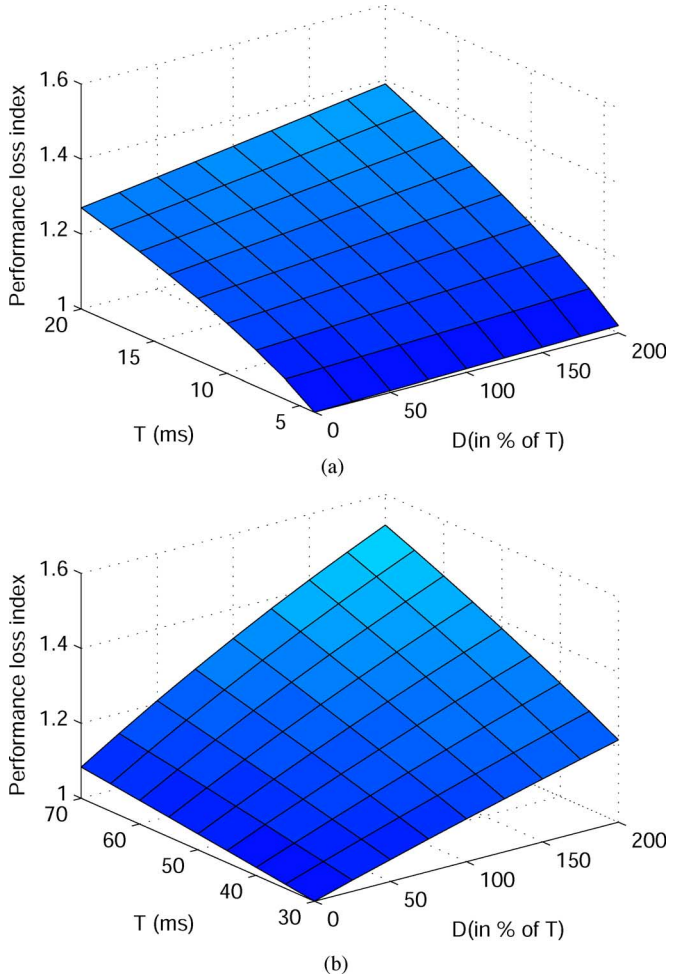

Fig. 8. Derived performance loss indices. (a) Performance loss index of **Plant A**. (b) Performance loss index of **Plant B**.

large as twice the sampling period. The performance loss indices derived for the two plants are plotted separately in Fig. 8.

To facilitate the comparison of the performance between different plants, each performance loss index has been normalized so that the minimum performance value is 1. Fig. 8(a) shows that **Plant A** is only sensitive to the sampling period, and quite tolerant to the relative deadline, especially for small sampling periods. On the contrary, Fig. 8(b) shows that **Plant B** is much more sensitive to the relative deadline than to the sampling period. The figures also show that the derived performance loss index of both plants can be approximated by a linear function. For example, the performance loss index of **Plant A** as a function of the period, in the case of zero delay, can be approximated by the linear function $J = 16.4244 \cdot T + 0.9564$, where the resulting mean squared error of the approximation is $1.3314 \cdot 10^{-4}$.

B. Experimental Setup

To evaluate the performance of the proposed approach, a synthetic task set has been considered for creating different workload situations, since a specific benchmark would not explore the whole design space. The considered task set τ consists of

TABLE I
SUMMARY OF THE CONTROLLER TASKS

Task	WCET (ms)	Period (ms)	Utilization
τ_1	4	$[4, 20]$	$[0.2, 1]$
τ_2	4	$[4, 20]$	$[0.2, 1]$
τ_3	4	$[4, 20]$	$[0.2, 1]$
τ_4	4	$[30, 70]$	$[0.057, 0.133]$
τ_5	4	$[30, 70]$	$[0.057, 0.133]$
U_{ctrl}			$[0.714, 3.266]$

$n = 9$ hard real-time tasks scheduled by EDF on a uniprocessor. The task set is split into a subset τ_{ctrl} of $n_{\text{ctrl}} = 5$ controller tasks and a subset $\tau_{n\text{ctrl}}$ of $n_{n\text{ctrl}} = 4$ regular tasks. The 5 controller tasks in τ_{ctrl} are labeled as $\tau_1, \tau_2, \tau_3, \tau_4$ and τ_5, where τ_1, τ_2 and τ_3 control a **Plant A** type each, whereas τ_4 and τ_5 control a **Plant B** type each. The derived performance loss indices of both types of plant are saved as 2-D lookup tables, which allow the optimization procedure to interpolate the cost value.

The WCET of each controller task is equal to 4 ms and task periods vary in the range reported in Table I, as for the evaluation of the performance loss indices presented in Section VI-A. The resulting maximum and minimum utilization of each task is also reported in the table.

Notice that the maximum utilization of all controller tasks U_{ctrl} ranges from 0.714 to 3.266, meaning that the controller tasks cannot be scheduled by EDF at their maximum sampling rates.

To investigate situations under different system loads, the utilization of all the controller tasks U_{ctrl} was fixed to 0.75 throughout the simulation, and the total utilization of the whole task set U was varied from 0.8 to 1, with a step of 0.05. The tasks within $\tau_{n\text{ctrl}}$ were generated using the UUNIFAST algorithm [34], with computation time C_i uniformly distributed in $[1, 10]$ ms and utilization U_i chosen according to a 4-D uniform distribution to reach $U_{n\text{ctrl}} = U - U_{\text{ctrl}}$. For each value of U, the performance loss index reported on the graphs was computed as the average on 100 repetitions with randomly generated subsets of $\tau_{n\text{ctrl}}$.

To select the scheduling parameters that optimize the overall control performance, the function $\mathcal{F}$ in (3) has been chosen as follows to form up a global performance loss index:

$$J = \sum_{i=1}^{n_{\text{ctrl}}} w_i \cdot J_i(T_i, D_i)$$

where $w = [w_1, w_2, w_3, w_4, w_5]$ is a weight vector. For this case, all weights have been set to 1, meaning that all plants have the same importance.

As long as the utilization of all controller tasks U_{ctrl} is decided, period selection can be performed without consideration of any regular tasks, using the resource constraint of $\sum_{\tau_i \in \tau_{\text{ctrl}}} (C_i)/(T_i) \leq U_{\text{ctrl}}$, as the first step described in Section V. By solving the optimization problem with deadlines equal to periods, the results shown in Table II were obtained.

Once periods have been derived, deadline selection can then be performed in the deadline space. In the next section, the proposed approach is compared with respect to other approaches under different load conditions.

TABLE II
RESULTS OF PERIOD SELECTION

Task	Period (ms)	Utilization
τ_1	0.0198	0.2022
τ_2	0.0198	0.2022
τ_3	0.0198	0.2022
τ_4	0.0558	0.0717
τ_5	0.0558	0.0717
U_{ctrl}		0.75

C. Comparison With Other Methods

In this section, the following methods are considered for comparison.

- *Standard*: According to this method, control performance is optimized only by period selection (see Section VI-B) and relative deadlines are set equal to periods.
- *D-convex*: This is the two-step approach proposed in this paper, where periods are first selected and then relative deadlines of controller tasks are determined using the deadline convex space, as proposed in Section V.
- *Binary Search*: According to this method, periods are first selected, as described in Section V, and then deadlines are adjusted using the algorithm proposed by Hoang and Buttazzo [14], where each task deadline can be adjusted within the range $[D_i^{\min}, D_i^{\max}]$, according to a scaling factor δ_i, specified for each task, where 1 denotes maximum deadline reduction (still guaranteeing schedulability), and 0 means no deadline reduction with respect to $D_i^{\max}$. The deadline adjustment of the entire task set is achieved by binary search.

In all the following experiments, only the deadlines of the controller tasks are adjusted, while those of the regular tasks are set equal to their periods.

In a first experiment, *D-convex* is compared with *Standard* and *Binary Search* where all the controller tasks have the same scaling factor, leading to a uniform reduction of deadlines. Fig. 9 shows the average value of the ratio of the selected deadline D_i and the period T_i. Note that a ratio larger than 1 means that deadline is extended beyond the period. The ratios of τ_1, τ_2 and τ_3 are the same due to the same performance loss index and the same weight, and thus reported in the same figure [Fig. 9(a)]. The same applies to tasks τ_4 and τ_5, which are reported in Fig. 9(b).

In both subfigures, the ratios under *Standard* stay at 1, whereas the ratios under *Binary Search* have the same value due to the uniform deadline reduction. However, as shown in Fig. 9(a), applying the *D-convex* method, the ratio of τ_1, τ_2 and τ_3 becomes greater than 1, meaning that their deadlines are extended beyond their periods to achieve a greater reduction of τ_4 and τ_5's deadlines. Indeed, Fig. 9(b) shows that, using *D-convex*, τ_4 and τ_5's deadlines can be reduced more than under *Binary Search* with the same scaling factor.

The resulting control performance loss for the three considered methods is illustrated in Fig. 10. As shown in the figure, under *Standard* and *Binary Search*, for high workload conditions the performance loss of the whole system is significantly degraded. This means that, in a highly loaded system, the interference introduced on the execution of τ_4 and τ_5 leads to a worse behavior of their controlled plants (**Plant B** type). However, under *D-convex*, the performance loss is kept at an acceptable level, even if the system is highly loaded. This is possible because the *D-convex* method allows a more aggressive reduction of τ_4 and τ_5's deadlines, limiting their delay and jitter to maintain the performance.

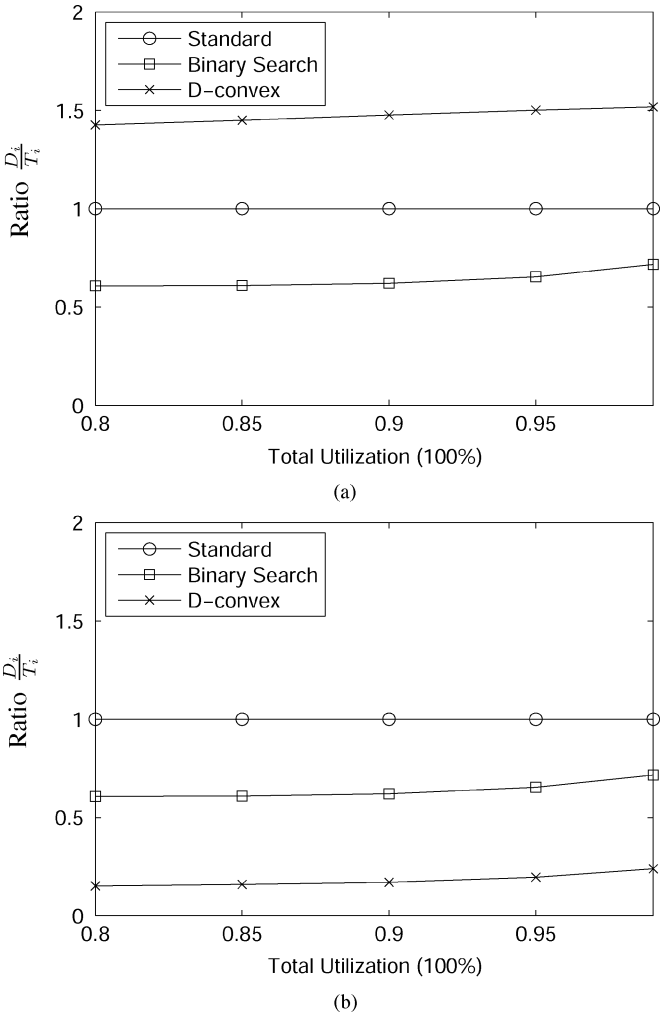

Fig. 9. Ratios of selected deadline and period $(D_i)/(T_i)$. (a) Ratio $(D_i)/(T_i)$ of τ_1, τ_2 and τ_3. (b) Ratio $(D_i)/(T_i)$ of τ_4 and τ_5.

The second experiment was aimed at comparing *D-convex* with *Binary Search*, for different scaling factors of the controller tasks. Because of their higher sensitivity to delay and jitter, the scaling factors of τ_4 and τ_5 were set to 1, whereas those of τ_1, τ_2 and τ_3 were set equal to a common value δ, which was varied in the experiment. For each controller task, $D_i^{\min} = C_i$ and $D_i^{\max}$ is twice the period selected by the procedure in Section V. The scaling factors δ_i of all the regular tasks were set to 0, so forcing their relative deadlines equal to periods.

The results of this experiment are reported in Fig. 11. Notice that $\delta = 1$ implies that all controller tasks have the same scaling factor, hence equivalent to uniform reduction of deadlines. The figure shows that the performance obtained using *D-convex* is better than using a scaling factor $\delta = 0.8$ and $\delta = 1.0$. However, the *Binary Search* method with $\delta = 0.2$ or $\delta = 0.5$ gives similar performance as *D-convex* when the system utilization U is below 0.9, and may lead to better performance when $U > 0.9$. The results show that selecting the proper scaling factors is not trivial, and a method for determining the best scaling factors is

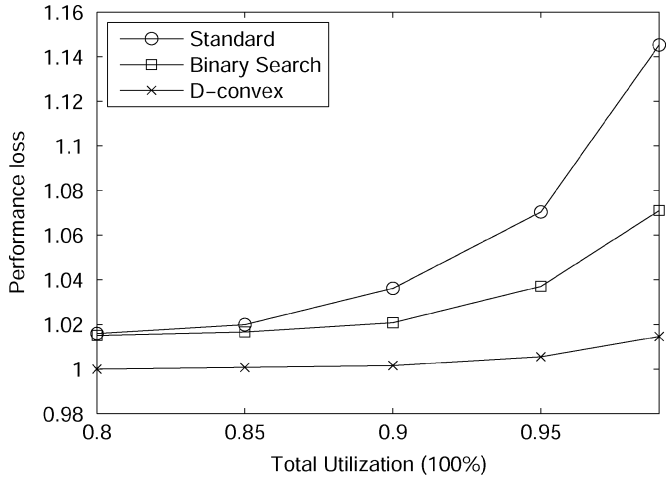

Fig. 10. Control performance under different strategies.

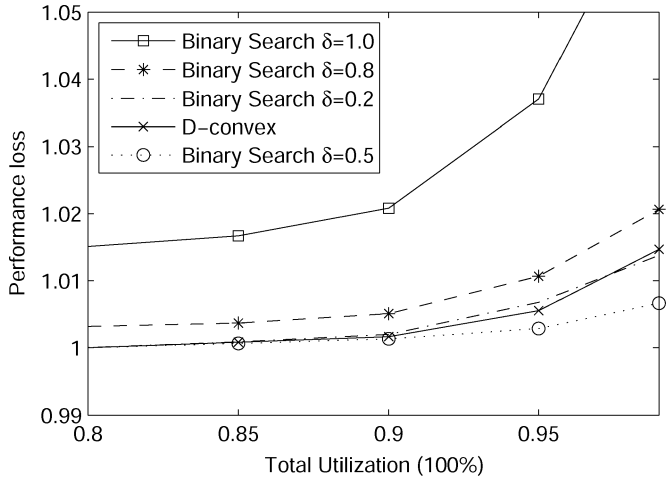

Fig. 11. Comparison with *Binary Search* method using different scaling factors.

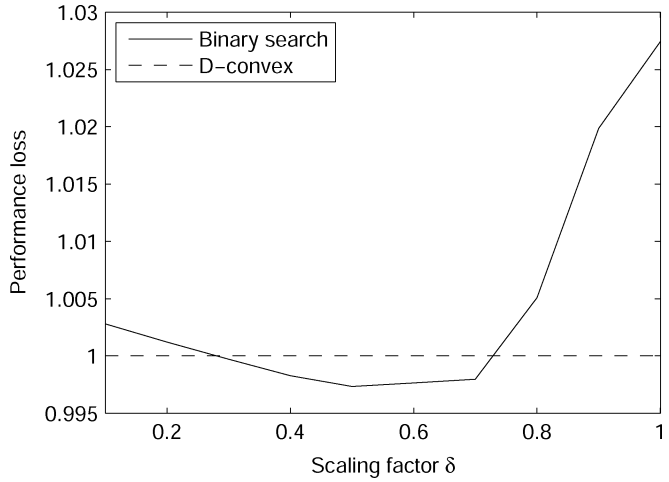

Fig. 12. Comparison with *Binary Search* method using different scaling factors $(\mathrm{U} = 0.95)$.

still missing. To further investigate how the choice of different scaling factors affects the performance, an additional experiment was carried out to test J as a function of δ, for $U = 0.95$. The evaluated performance is normalized with respect to the performance obtained using *D-convex*, as reported in Fig. 12. The figure shows that, the *Binary Search* method performs better than *D-convex* when the scaling factor δ is within [0.3, 0.74], approximately, which indicates that an improper selection of scaling factors may produce a non negligible performance loss. On the other hand, even when *Binary Search* gives better performance, the difference from the proposed *D-convex* method is not significant.

VII. Conclusion

This paper addressed the problem of task parameter selection for real-time controller tasks in resource-constrained systems. In particular, a general method has been proposed to derive the control performance loss index in either a simulative or experimental way, with respect to various timing attributes, and arbitrary deadlines, which are allowed to be less than, equal to or greater than the periods. Task periods and deadlines were then selected by optimization upon the convex approximation of EDF deadline space, considering the delay and jitter effects on control performance.

Extensive simulations have been performed to compare the proposed methodology with other methods. The results have shown that the proposed method managed to keep the performance loss at an acceptable level even in highly loaded systems which might lead to significant performance loss using other methods.

References

[1] C. L. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[2] H. J. Kushner and L. Tobias, "On the stability of randomly sampled systems," *IEEE Trans. Automat. Control*, vol. 14, no. 4, pp. 319–324, Aug. 1969.

[3] J. Nilsson, B. Bernhardsson, and B. Wittenmark, "Stochastic analysis and control of real-time systems with random time delays," *Automatica*, vol. 34, no. 1, pp. 57–64, Jan. 1998.

[4] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, "On task schedulability in real-time control systems," in *Proc. 17th IEEE Real-Time Syst. Symp. (RTSS 1996)*, Washington, DC, Dec. 1996, pp. 13–21.

[5] E. Bini and M. Di Natale, "Optimal task rate selection in fixed priority systems," in *Proc. 26th IEEE Real-Time Syst. Symp. (RTSS 2005)*, Miami, FL, Dec. 2005, pp. 399–409.

[6] A. Cervin, B. Lincoln, J. Eker, K.-E. Årzén, and G. Buttazzo, "The jitter margin and its application in the design of real-time control systems," in *Proc. 10th Int. Conf. Real-Time Embedded Comput. Syst. Appl.*, Göteborg, Sweden, Aug. 2004.

[7] P. Martí, J. Fuertes, G. Fohler, and K. Ramamritham, "Jitter compensation for real-time control systems," in *Proc. 22nd IEEE Real-Time Syst. Symp. (RTSS 2001)*, Dec. 2001, pp. 39–48.

[8] S. Baruah, "The limited-preemption uniprocessor scheduling of sporadic task systems," in *Proc. 17th Euromicro Conf. Real-Time Syst. (ECRTS 2005)*, 2005, pp. 137–144.

[9] G. Yao, G. Buttazzo, and M. Bertogna, "Bounding the maximum length of non-preemptive regions under fixed priority scheduling," in *Proc. 15th IEEE Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA 2009)*, Beijing, China, Aug. 2009.

[10] Y. Wu and M. Bertogna, "Improving task responsiveness with limited preemptions," in *Proc. 14th IEEE Int. Conf. Emerging Technol. Factory Autom. (ETFA 2009)*, Mallorca, Spain, Sep. 2009.

[11] H. Hoang, G. Buttazzo, M. Jonsson, and S. Karlsson, "Computing the minimum edf feasible deadline in periodic systems," in *Proc. 12th IEEE Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Sydney, Australia, Aug. 2006, pp. 125–134.

[12] P. Balbastre, I. Ripoll, and A. Crespo, "Optimal deadline assignment for periodic real-time tasks in dynamic priority systems," in *Proc. 18th Euromicro Conf. Real-Time Syst.*, Dresden, Germany, Jul. 2006, pp. 65–74.
[13] S. Baruah, G. Buttazzo, S. Gorinsky, and G. Lipari, "Scheduling periodic task systems to minimize output jitter," in *Proc. 6th Int. Conf. Real-Time Comput. Syst. Appl., RTCSA'99*, 1999, pp. 62–69.
[14] H. Hoang and G. Buttazzo, "Reducing delay and jitter in software control systems," in *Proc. 15th Int. Conf. Real-Time and Network Syst. (RTNS2007)*, Nancy, France, Mar. 2007.
[15] G. Buttazzo and A. Cervin, "Comparative assessment and evaluation of jitter control methods," in *Proc. 15th Int. Conf. Real-Time Network Syst. (RTNS2007)*, Mar. 29–30, 2007, pp. 137–144.
[16] M. Ryu and S. Hong, "Toward automatic synthesis of schedulable real-time controllers," *Integr. Comput.-Aided Eng.*, vol. 5, no. 3, pp. 261–277, 1998.
[17] B. K. Kim, "Task scheduling with feedback latency for real-time control systems," in *Proc. 5th Int. Conf. Real-Time Comput. Syst. Appl.*, Hiroshima, Japan, Oct. 1998, pp. 37–41.
[18] L. Palopoli, L. Abeni, F. Conticelli, M. Di Natale, and G. Buttazzo, "Real-time control system analysis: An integrated approach," in *Proc. 21st IEEE Real-Time Syst. Symp. (RTSS 2000)*, Orlando, FL, U.S.A., Dec. 2000.
[19] T. Chantem, X. Wang, M. D. Lemmon, and X. S. Hu, "Period and deadline selection for schedulability in real-time systems," in *Proc. 20th Euromicro Conf. Real-Time Systems (ECRTS 2008)*, Jul. 2008, pp. 168–177.
[20] E. Bini and A. Cervin, "Delay-aware period assignment in control systems," in *Proc. 29th IEEE Real-Time Syst. Symp. (RTSS 2008)*, Barcelona, Spain, Dec. 2008, pp. 291–300.
[21] Y. Wu, E. Bini, and G. Buttazzo, "A framework for designing embedded real-time controllers," in *Proc. 14th IEEE Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA 2008)*, Aug. 2008, pp. 303–311.
[22] E. Bini and G. Buttazzo, "The space of edf deadlines: The exact region and a convex approximation," *Real-Time Systems*, vol. 41, no. 1, pp. 27–51, 2009.
[23] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén, "How does control timing affect performance?," *IEEE Control Syst. Mag.*, vol. 23, no. 3, pp. 16–30, Jun. 2003.
[24] G. Buttazzo, P. Martí, and M. Velasco, "Quality-of-control management in overloaded real-time systems," *IEEE Trans. Comput.*, vol. 56, no. 2, pp. 253–266, Feb. 2007.
[25] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 10th ed. Englewood Cliffs, NJ: Prentice-Hall, 2004.
[26] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1997.
[27] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén, "Feedback-feedforward scheduling of control tasks," *Real-Time Systems*, vol. 23, no. 1–2, pp. 25–53, Jul. 2002.
[28] B. Lincoln and A. Cervin, "Jitterbug: A tool for analysis of real-time control performance," in *Proc. 41st IEEE Conf. Decision and Control*, Las Vegas, NV, Dec. 2002.
[29] D. Henriksson, A. Cervin, M. Andersson, and K.-E. Årzén, "TrueTime: Simulation of networked computer control systems," in *Proc. 2nd IFAC Conf. Analy. Design of Hybrid Systems*, Alghero, Italy, Jun. 2006.
[30] A. Casile, G. C. Buttazzo, G. Lamastra, and G. Lipari, "Simulation and tracing of hybrid task sets on distributed systems," in *Proc. RTCSA*, 1998, pp. 249–256.
[31] P. Gai, L. Abeni, M. Giorgi, and G. Buttazzo, "A new kernel approach for modular real-time systems development," in *Proc. 13th Euromicro Conf. Real-Time Syst.*, Delft, The Nederlands, Jun. 2001, pp. 199–206.
[32] S. K. Baruah, A. K. Mok, and L. E. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *Proc. 11th IEEE Real-Time Syst. Symp. (RTSS 1990)*, Lake Buena Vista, FL, Dec. 1990, pp. 182–190.
[33] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, USA: Cambridge Univ. Press, 2004.
[34] E. Bini and G. C. Buttazzo, "Biasing effects in schedulability measures," in *Proc. 16th Euromicro Conf. Real-Time Syst. (ECRTS 2004)*, Catania, Italy, Jul. 2004, pp. 196–203.

Yifan Wu (S'08) received the M.Eng. degree in control science and engineering in 2006 from Zhejiang University, Zhejiang, China, and the Ph.D. degree in computer engineering from Scuola Superiore Sant'Anna of Pisa, Pisa, Italy, in 2010.

In 2009, he was a visiting student at Lund University, collaborating with Prof. K.-E. Årzén and Associate Prof. A. Cervin. His research interests include real-time control systems, scheduling algorithms, and dataflow programming model.

Giorgio Buttazzo (SM'05) graduated in electronic engineering from the University of Pisa, Pisa, Italy, in 1985, received the M.S. degree in computer science from the University of Pennsylvania, Philadelphia, in 1987, and the Ph.D. degree in computer engineering from the Scuola Superiore Sant'Anna of Pisa, Pisa, in 1991.

He is a Full Professor of Computer Engineering at the Scuola Superiore Sant'Anna of Pisa. From 1987 to 1988, he worked on active perception and real-time control at the G.R.A.S.P. Laboratory, University of Pennsylvania. He has authored six books on real-time systems and over 200 papers in the field of real-time systems, scheduling algorithms, overload management, robotics, and neural networks.

Prof. Buttazzo has been Program Chair and General Chair of the major international conferences on real-time systems. He is Editor in Chief of *Real-Time Systems* (Springer) and Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. He is Vice-Chair of the IEEE Technical Committee on Real-Time Systems and member of the Euromicro Executive Board on Real-Time Systems.

Enrico Bini (M'09) received the Laurea degree in computer engineering from the University of Pisa, Pisa, Italy, in 2000 and the Ph.D. degree in computer engineering from Scuola Superiore Sant'Anna, Pisa, in 2004. He also completed graduate studies in mathematics in January 2010.

He is an Assistant Professor at the Scuola Superiore Sant'Anna, Pisa, Italy. In 2003, he was a visiting student at the University of North Carolina, Chapel Hill, collaborating with Prof. S. Baruah. He has published more than 50 papers on scheduling algorithms, real-time operating systems, sensitivity analysis in embedded systems design and optimization techniques.

Anton Cervin (M'10) received the M.Sc. degree in computer science and engineering in 1998 and the Ph.D. degree in automatic control in 2003 from Lund University, Lund, Sweden.

He is currently an Associate Professor at the Department of Automatic Control, Lund University. From 2006 to 2009, he also held a Junior Researcher position at the Swedish Research Council. His research interests include embedded and networked control systems, event-based control, real-time scheduling theory, and computer tools for analysis and simulation of controller timing.