Deadline-Miss-Adaptive Controller Implementation for Real-Time Control Systems

Nils Vreman, Claudio Mandrioli, Anton Cervin
Lund University, Sweden

2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium (RTAS) | 978-1-6654-9998-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/RTAS54340.2022.00010

***Abstract*—The policy used to implement a control algorithm in a real-time system can significantly affect the quality of control. In this paper, we present a method to adapt the controller implementation, with the objective to improve the system's performance under real-time faults. Our method compensates for missing state updates by adapting the controller parameters according to the number of consecutively missed deadlines. It extends the state-of-the-art by considering dynamic controllers, which have had limited coverage in previous literature. The adaptation mechanism can be precomputed offline, solely based on knowledge about the controller and not on the controlled plant. The approach is indifferent to the control design, as well as to the scheduling policy, and can be automatically realised by the operating system, thus improving the robustness of the control system to intermittent and unexpected real-time faults. We develop a stochastic performance analysis method and apply it to both a real plant and numerous simulated plants to evaluate our adaptive controller. Complementary to the stochastic analysis, we also do worst-case stability analysis of the resulting system. The results confirm the conjuncture that the adaptive controller improves both the performance and robustness in the presence of deadline misses.**

***Index Terms*—Fault-Tolerant Control Systems, Deadline Miss, Adaptive Control**

I. Introduction

Computer-controlled systems are prime instances of real-time systems [1], [48]. Due to the tight interconnection between the environment, hardware, and software, designing such systems has been considered challenging. Part of this challenge resides in the design of the real-time software, specifically considering both the normal operation [2], [41] (e.g., correct output computation) and system malfunctions [12], [52] (e.g., temporary overloads). For this reason, the research community has undertaken a significant effort to merge design choices on the algorithmic side and on the real-time implementation side.

In computer-controlled systems, algorithms are developed using control theory [6]. The theory offers strong and practically relevant formal guarantees, but also makes strict assumptions on the real-time execution of the implemented algorithm. These assumptions are naturally translated into periodic tasks with hard deadlines. However, meeting every single deadline in a periodic control task is not necessary [51], [52]. Instead, the timing requirements are the result of design choices and engineering trade-offs between resource utilisation and performance [15], [41].

Co-design approaches have been proposed to maximise the control performance while minimising the real-time resource utilisation [45], [53]. However, the tight integration between control algorithms and the real-time implementation has limitations, due to the complexity of the resulting systems. This complexity generally translates into (i) complex design methodologies, (ii) conservative results, and (iii) strong assumptions on the system properties.

Differently from existing approaches, in this paper we avoid the additional complexity by *automatically* adapting the real-time implementation of the controller. The approach is inspired by the concept of autotuning, originating in the control literature [5], [26]. Autotuning was developed to simplify the PID control design process by automatically optimising the controller's design parameters. This idea is here translated to the real-time implementation, where instead the predesigned control algorithm is automatically adapted for the real-time architecture to minimise the design effort.

To enable an automatic adaptation of the control algorithm for a wide range of systems, we make as few assumptions about the implementation platform as possible. The system requirements that we pose are nonintrusive and seen in industrial applications [1]. The real-time operating system is required to be able to: schedule periodic tasks with implicit deadlines, abort (kill) instances of tasks that miss their deadlines, roll back the state of aborted tasks [59], [71], and handle controller inputs and outputs at task release times [21], [34]. While previous co-design approaches require, e.g., probabilistic or weakly hard descriptions of deadline misses [33], [50], our adaptation approach works for any deadline miss model. From the point of view of the controller, we assume a linear time-invariant control law, but we require no prior information about the control design, nor about the system to be controlled. Similarly to [49], our adaptive control implementation is applicable to general linear discrete-time controllers and not only static ones.

To evaluate the performance of our adaptive implementation, we propose a stochastic analysis of the control system. The analysis assumes a probabilistic model of the deadline misses; however, it is agnostic to how the model is obtained. We

This work was supported by the ELLIIT Strategic Research Area. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871259 (ADMORPH project). This (publication/report) reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

978-1-6654-9998-9/$31.00 ©2022 IEEE
DOI 10.1109/RTAS54340.2022.00010

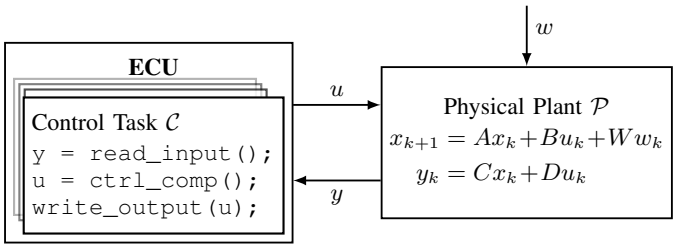

Fig. 1. Typical structure of a computer-controlled system. *Left:* the Electronic Control Unit (ECU) implementing a real-time system, including a task executing the controller. *Right:* the physical plant controlled by the actuation variable u, affected by the disturbance w, and producing the measurement y.

utilise this analysis to evaluate our approach on both a *real* system and a benchmark set of 268 *simulated* control systems from the process industrial domain. We use the former to evaluate the practical relevance of the proposed approach and the latter to evaluate its general applicability. We complement our performance analysis with a worst-case stability analysis. In all of our tests, the adaptive implementation improves both the performance and the worst-case stability of the system.

The paper provides the following two main contributions:

- It proposes a novel, modular and intuitive control law implementation that adapts the control action upon the occurrence of deadline misses in the periodic controller task. The adaptive implementation has a small overhead and is applicable to all linear dynamic controllers.
- It proposes a probabilistic analysis of the resulting control system subject to deadline misses. The analysis is based on a comparison with the ideal system without deadline misses and is used to evaluate the performance of both a real system and numerous simulated control systems. The results show that the adaptive implementation significantly improves the system performance and robustness, compared to the nominal implementation.

The remainder of this paper is outlined as follows. In Section II we present the relevant control and real-time system background. Section III presents and discusses the previous literature on the topic, identifying the limitations of the state-of-the-art that we are addressing. In Section IV we first propose our adaptive implementation of the control law, and then we integrate the proposed controller with a probabilistic analysis method to enable its evaluation. Section V presents an empirical evaluation of the adaptive control law based on the proposed analysis for both a physical system and numerous simulated systems. Finally, Section VI concludes the paper.

II. System Model

This section introduces the necessary background and models needed for the remainder of the paper. We discuss the real-time implementation of a general linear controller and how it can affect the performance of the system. We start by describing the behaviour of the system under *ideal* conditions. Based on this, we state how *deadline misses* in the real-time implementation affect the system's behaviour.

A. Control Systems under Ideal Operations

The objective of a control system is to regulate a physical process, usually called a *plant*, so that it behaves as desired. Figure 1 shows the structure of a control system, where an Electronic Control Unit (ECU) implements a real-time system. Among the different tasks executed in the system, there is a task responsible for the control computations, denoted as the control task. Every job released by this task performs the following actions: (i) Read measurements from the sensors. (ii) Use the sensor information to update its state and compute a control action. (iii) Write the control action to the actuators. The executed algorithm is generally designed using control theory, where the effective application of the algorithm relies on assumptions about the real-time execution.

The dynamic behaviour of a plant is commonly described using a state-space model [6]. Such models are constituted of two sets of equations: one describing the dynamics of the plant and another one describing the relation between the plant state and the available measurements. Describing physical phenomena, those equations are for the most part continuous and nonlinear. However, for control design and implementation purposes, they are commonly transformed into a discrete-time linear time-invariant (LTI) state-space model:

$$\mathcal{P}: \begin{cases} x_{k+1} = Ax_k + Bu_k + Ww_k \\ y_k = Cx_k + Du_k \end{cases} \tag{1}$$

Here, the variable k counts the number of discrete time steps that have passed since the system started executing. Furthermore, the variable $x_k \in \mathbb{R}^{n_x}$ represents the *plant state*, $u_k \in \mathbb{R}^{n_u}$ corresponds to the *control signal* computed in order to affect the plant, $w_k \in \mathbb{R}^{n_w}$ models disturbances and reference signals, and $y_k \in \mathbb{R}^{n_y}$ is the *measurement signal* available to the controller. For what concerns the matrices, $A \in \mathbb{R}^{n_x \times n_x}$ captures the relation between the current state and the next state, while $B \in \mathbb{R}^{n_x \times n_u}$ and $W \in \mathbb{R}^{n_x \times n_w}$ respectively capture how the control signal and the exogenous signals affect the state at the next time step. Furthermore, $C \in \mathbb{R}^{n_y \times n_x}$ and $D \in \mathbb{R}^{n_y \times n_u}$ respectively describe how the current state and control signal relate to the measurements.

To control the behaviour of the plant, a *controller* $\mathcal{C}$ is synthesised to follow some desired properties, such as: (i) stability, (ii) speed of convergence, (iii) control effort, and (iv) disturbance rejection. The stability requirement enforces that none of the signals diverge and is a necessary condition for all controllers. Moreover, a controller that fulfils all requirements will make the output converge to the reference value within a specified time, while minimising the control effort and the effect of possible disturbances.

Controllers are commonly implemented as fixed-rate, periodically executing tasks, following the Logical Execution Time (LET) paradigm [21], [28], [34]. Adopting this paradigm, the sensors are read at the beginning of the task period, and the control action is written to the actuators at the end of the period. This minimises the effect of fluctuations in the

execution pattern of the control algorithm (called jitter) at the cost of introducing a one-step delay in the actuation.

While they are sometimes specified as transfer functions [6], controllers are often *implemented* as discrete-time state-space systems. More specifically, we assume that the controller is an LTI system that takes the measurement y_k as input and produces the control signal u_{k+1} as output:[1]

$$\mathcal{C}: \begin{cases} z_{k+1} = Fz_k + Gy_k \\ u_{k+1} = Hz_k + Ky_k. \end{cases} \tag{2}$$

Here, $z_k \in \mathbb{R}^{n_z}$ represents the internal state of the controller. The second equation, specifying the control action at step $k+1$, captures the one-step delay introduced by the LET paradigm. The matrices $F \in \mathbb{R}^{n_z \times n_z}$, $G \in \mathbb{R}^{n_z \times n_y}$, $H \in \mathbb{R}^{n_u \times n_z}$, and $K \in \mathbb{R}^{n_u \times n_y}$ govern the behaviour of the controller.

In conjunction with Equation (2), we define two types of controllers: *static* and *dynamic*.

Static Controller: We denote a *static controller* as any controller $\mathcal{C}$ that is *stateless* (i.e., it has no internal state z; $n_z = 0$).

Dynamic Controller: We denote a *dynamic controller* as any controller $\mathcal{C}$ that is *stateful* (i.e., it has an internal state z; $n_z \geq 1$).

From the definitions above, we note that a static controller can be written as a fixed gain matrix times the input (i.e., $\mathcal{C}: u_{k+1} = Ky_k$), while a dynamic controller is equivalent to (2) with non-empty matrices F, G, H, and K. Examples of static controllers include proportional (P) controllers, state feedback controllers, and linear–quadratic regulators (LQR), while dynamic controllers include proportional–integral–derivative (PID) controllers, lead–lag compensators, and linear–quadratic–Gaussian (LQG) regulators [6].

B. Control Systems Subject to Deadline Misses

We assume that the controller $\mathcal{C}$ is implemented as a periodic task with period T and implicit deadlines. Intuitively, each execution period of the control task corresponds to one time step k. At the start of period k, the task releases a *job* that should be completed before the deadline at time $(k+1)T$.

Faults in the real-time system can affect the timely execution of the control task [61]. We denote the outcome of a job's execution as either a deadline *hit* or *miss*, corresponding to whether the job completed its execution before its deadline or not. The source of a deadline miss could be a temporary CPU overload [8], cache misses [46], [68], or unexpected preemption from hardware interrupts or higher priority tasks [60]. However, the analysis and adaptation methods presented in this paper are independent of the origin of the deadline miss.

To study what happens when the controller misses a deadline, we have to define how the system behaves in such circumstances. In particular, three aspects need to be considered: (i) how the controller state is updated, (ii) how the actuator handles the lack of a new control signal [55], and (iii) how the operating system handles a job that misses its deadline [14], [50]. The first item refers to what happens to the internal state z when the controller is unable to finish its execution ahead of its dedicated deadline. Henceforth, we assume that when the controller misses its deadline, the controller state is *not* updated (implying $z_{k+1} = z_k$). This is motivated by the possibility to roll back z_k to a previous state [1], [59], [71] and by the impossibility to guarantee that the state update was finished if the job was only partially completed.

Regarding the second item, mainly two actuator models have previously been considered in the literature: *Zero* and *Hold* [55]. Under the Zero model, if the job released at time kT misses its deadline at time $(k+1)T$, the actuator outputs $u_{k+1} = 0$. This strategy is uncommon in practice, because it performs well only in very specific cases [65]. Instead, the more common actuator model is to hold the control signal in the case of a missed deadline: $u_{k+1} = u_k$. Although our proposed adaptation is in itself independent of the actuator model, the Hold model has been adopted in the analysis and examples of this paper.

For what concerns the handling of the job that missed the deadline, there exist at least three different employable strategies: (a) *Kill* (b) *Skip-Next*, and (c) *Queue*. When using the Kill strategy, the job that missed its deadline gets terminated, the controller state is rolled back, and the next job is released. The Skip-Next strategy does not terminate the job that missed its deadline. Instead, it lets the job continue its execution, not releasing subsequent jobs until the active one has finished executing. Queue behaves similarly to the Skip-Next strategy: it does not kill the current job, but it does release the subsequent jobs to the job queue. Both the Skip-Next and Queue strategies allow jobs to work with outdated input data, since they do not terminate jobs that miss their deadline. Thus, they introduce a lag in the actuation of the control law, which in most cases reduces the control performance with respect to what could be achieved with the Kill strategy.

For the remainder of this paper we will use the Kill strategy, since it (a) introduces explicit breakpoints in which to adapt the controller, (b) supplies the controller with the latest sensor measurement after an overrun, (c) helps free computational resources in overrun situations, and (d) is a common choice in both industry and literature [1], [9], [29]. Furthermore, we note that in [65] it has been observed that the actuator model is of greater relevance than the handling of the deadline overrun. We also note that an effective implementation of the kill strategy that includes state roll-back requires the implementation of checkpointing mechanisms [59], [71], which might not be implemented by default in a real-time system.

We conclude this section by mentioning that various models have been proposed to describe tasks that may experience deadline misses, e.g., soft [43] or weakly hard models [9], [27]. For the adaptive controller in this work, we only assume that the number of consecutive deadline misses q is bounded by a finite quantity $q_{max} < \infty$. This assumption is a practical necessity, since accepting an infinite run of deadline misses would completely disconnect the plant from the controller [42].

[1]Note that we adopt a positive feedback convention.

Solely for performance analysis, in Section IV-B we adopt a stochastic model of the sequences of deadline hits and misses. Such a model can be computed from a probabilistic task set model using existing techniques [16], [17], [44]. However, we remark that the proposed adaptive controller implementation is independent of the stochastic deadline miss model.

C. Control System Stability under Deadline Misses

Guaranteeing the stability of control systems is essential in control engineering. A linear control system without exogenous inputs is *stable* if and only if the system's state always converges to zero irrespective of its initial value. Assuming ideal conditions, i.e., no deadline misses, stability can be verified by checking whether all the closed-loop system's eigenvalues lie inside the unit circle. However, in the presence of deadline misses, the classical stability criteria are no longer sufficient, since the dynamics of the control system is time-varying and changes according to the specific pattern of deadline misses. For such cases, switched system stability analysis (also known as switching stability) is a viable extension of classical stability [36]. In this paper, we analyse the switching stability using both a time-averaged (Markov Jump Linear Analysis [22]) and a worst-case (Joint Spectral Radius [54]) approach.

Modelling the deadline misses as a stationary random process with known statistical properties allows us to calculate an analytical time-averaged performance index of the closed-loop system. If the performance index is finite, then the system is guaranteed to be stable in the mean-square sense (meaning that the state will not diverge with probability 1). We develop our approach to compute the time-averaged performance in Section IV-B.

If the statistical properties of the deadline misses are unknown or uncertain, switching stability can still be analysed under worst-case conditions. The Joint Spectral Radius (JSR) generalises the spectral radius of a matrix (i.e., the largest absolute eigenvalue) to a set of matrices; thus, the JSR characterises the largest asymptotic growth (or contraction) rate of the states. If the JSR of the set of closed-loop matrices representing $i = \{0, 1, \ldots, q_{max}\}$ deadline misses (followed by a hit) is below 1 then the system is switching stable. Conversely, if it is above 1, there exists at least one sequence of deadline misses and hits that makes the system unstable. There exist both toolboxes and methods for calculating the JSR for control systems subject to deadline misses [42], [64], [67].

III. Problem Description

In this section, we discuss the problem of implementing a control system, as defined in Section II, that is robust to deadline misses. The problem has been investigated in the existing literature, and different methods have been proposed to solve it. We offer a discussion on the scientific literature to identify the strengths and limitations of previous work and motivate the research problem studied in this paper.

A. Related Work

The robustness of control algorithms to timing faults has been discussed both from the analysis and synthesis perspectives. From the *analysis* perspective, the literature proposes different methods to analyse the stability and performance of control systems in the presence of real-time faults. The performance of different overrun handling strategies was discussed in the context of control systems in [14]. Similar studies, in the context of networked control systems, were presented by [56] and [66]. In [23], the authors developed a network of hybrid automata to analyse the consequences of timing variations in control software. The analysis of control systems under consecutive deadline misses was addressed in [42] and [70], using respectively the joint spectral radius and Lyapunov theory. Both [31] and [30] also leveraged Lyapunov theory to analyse nonlinear systems subject to network problems. Shifting the focus from stability to performance, [65] analysed control systems subject to bursts of deadline misses.

From the *synthesis* perspective, many works have studied overruns from a control and scheduling co-design perspective [3]. In [57], the authors designed state feedback controllers with time-varying gain, guaranteeing control performance under arbitrary period changes of the control task. A similar approach was taken in [52], where instead a weakly hard task model with known execution pattern was assumed. The authors of [35] designed stabilising state feedback controllers with different controller gains depending on the system's time delay. In [50], the authors used a probabilistic characterisation of the task model to develop state feedback controllers that are robust to deadline overruns. Motivated by industrial practices, [49] proposed the re-initialisation of the control task's period after the occurrence of overruns as well as the design of an adaptive control law. In the field of networked control [25], [63], different works have proposed compensation schemes for jitter and dropped data packets [32], [47], [72]. Within the same field, [33] proposed a control compensator design scheme for dropped packets under the weakly hard model that also guaranteed stability. Similarly, [39] used design techniques from optimal control theory to co-design controllers. [11] proposed to change the control task period according to the task execution time to improve schedulability and control performance. Finally, several works discussed the trade-offs between schedulability and control performance [12], [18], [20], [45].

This paper distinguishes itself from previous synthesis literature as it assumes that a linear controller has already been developed. We propose an adaptive implementation of the control algorithm that does not require modifications to the system's design phase.

B. Research Problem Motivation

Despite the considerable amount of prior research, we argue that existing design approaches to handling real-time faults in controllers suffer from at least one of the following limitations:

(i) Assumptions about static controller – i.e., the controller cannot have any internal dynamics [33], [35], [38], [42], [50], [52], [57], [73].

(ii) Complex co-design methodology, leading to conservative results – i.e., a detailed system model and a large extra design effort are needed [33], [35], [38], [45], [49], [57].
(iii) Increased runtime overhead – i.e., the approach reduces the likelihood of completing the control execution within the same total time budget [11], [13], [18].

The first limitation substantially narrows the applicability of the proposed design techniques, since the vast majority of real-world controllers include a dynamical part (e.g., an integrator to remove stationary errors, or a filter to estimate states or remove measurement noise). It is important to study dynamic controllers as they are more severely affected by computational faults than static ones. Intuitively, a static controller computes the control action u solely based on the latest measurement, and is therefore always up to date with respect to the plant state. In contrast, a dynamic controller computes u also with respect to its internal state, which is computed on the base of older measurements. When deadline misses occur, the controller state diverges from its desired value and degrades the control performance. We support this intuition with a motivating example, considering a static controller and a comparable dynamic controller that includes a Kalman filter [7]. The two controllers are designed to give similar performance under ideal conditions. The example shows that, while the static controller inherently provides robustness to deadline misses, the dynamic controller is significantly more sensitive.

Motivating Example: Consider the discrete-time LTI plant

$$\mathcal{P}: \begin{cases} x_{k+1} = \begin{bmatrix} 0 & 1 \\ -0.8 & 1.8 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k \\ y_k = x_k. \end{cases} \tag{3}$$

The control system's objective is to bring the state x to zero. The plant $\mathcal{P}$ consists of a stable pole and an integrator, which is compatible with several real-world applications, e.g., a cart on a rail or a joint of a robotic arm. To regulate the plant, we first consider the static controller $\mathcal{C}_1$:

$$\mathcal{C}_1: \; u_{k+1} = \begin{bmatrix} 0.256 & -0.372 \end{bmatrix} y_k.$$

The system is sampled and actuated using a sample time of $T = 0.1$ s. We consider a scenario where the ECU executing the control law is experiencing sporadic CPU overloads, resulting in 25% of the control jobs missing their corresponding deadlines. The initial state of the plant is $x_0 = [1, 1]^T$, and we want the controller to move it to the final position $x_f = [0, 0]^T$. In Figure 2, we show only the second component of the state vector x: the other component follows a very similar trajectory and is thus not plotted (for clarity). The behaviour of the plant controlled by $\mathcal{C}_1$ without deadline overruns is the solid blue line. The dashed blue line instead corresponds to the state evolution in the presence of deadline misses. The control task overruns are marked by red crosses on the horizontal axis. Despite the missed job completions, the plant state recovers gracefully in very few steps, similarly to the ideal behaviour of the controller (in the absence of overruns).

Practically, in all industrial applications, the control law $\mathcal{C}_1$ would be preceded by a noise filter or a state estimator, which

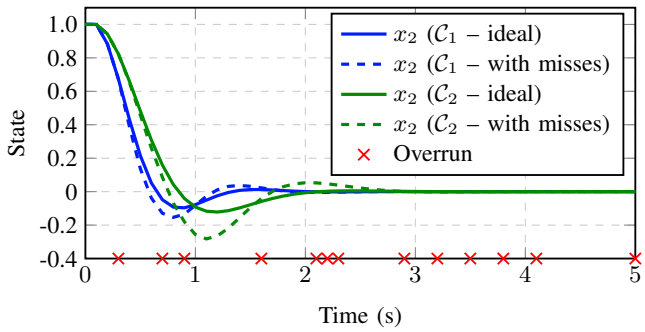

Fig. 2. The state x_2 corresponding to $\mathcal{P}$ (sampled with $T = 0.1$s) being controlled by the static controller $\mathcal{C}_1$ (dashed blue) or the LQG controller $\mathcal{C}_2$ (dashed green) during an interval of sporadic overloads ($\times$). The corresponding ideal behaviours (system not subject to overruns) for the static (solid blue) and LQG (solid green) controllers are also plotted. The state x_1 follows a very similar trajectory and is not plotted for readability.

introduces dynamic behaviour in the controller. Thus, we now consider the LQG controller $\mathcal{C}_2$, which contains a Kalman filter designed to suppress noise:

$$\mathcal{C}_2: \begin{cases} z_{k+1} = \begin{bmatrix} -0.151 & 0.810 \\ -0.711 & 1.206 \end{bmatrix} z_k + \begin{bmatrix} 0.151 & 0.190 \\ 0.166 & 0.221 \end{bmatrix} y_k \\ u_{k+1} = \begin{bmatrix} 0.226 & -0.242 \end{bmatrix} z_k + \begin{bmatrix} -0.023 & -0.034 \end{bmatrix} y_k. \end{cases}$$

The dashed green line in Figure 2 shows the second component of the plant state x, controlled by the dynamic LQG controller $\mathcal{C}_2$ and subject to the *same* deadline misses as controller $\mathcal{C}_1$. Comparing it to the ideal behaviour (solid green line), we observe a significant performance degradation compared to the static controller $\mathcal{C}_1$. This showcases that dynamic controllers suffer more under sporadic overloads than static controllers.

As for the second limitation listed above, introducing additional complexity for the gain of better performance is not always a desired design solution. The more convoluted the control design, the higher the design cost. This is a consequence of the increased development time needed to design the controller. When the complexity of the controller increases, the cost of the system upkeep increases due to the expert knowledge required. The most popular controllers in industry are thus simple ones that still perform adequately (e.g., the PI controller [5], [19], [62]).

Considering the third limitation, introducing additional overhead for fault-prone systems risks even more deadline misses. Depending on the plant or task model, there are cases where increasing the overhead might be acceptable, e.g., if deadline misses appear sporadically in bursts. However, for many task models, increasing the execution time after a deadline miss could have severe consequences, causing domino effects where subsequent jobs also miss their deadlines.

One major strength of many previously proposed fault-tolerant control design methods is that they can guarantee closed-loop stability under their respective fault and system models [35], [38], [40], [57]. A majority of the studies investigate static controllers. However, in most real-world applications, the controllers are dynamic, e.g., an integrator,

estimator, or filter is included in the loop. The a priori guarantees provided for the static controller are hence lost. Additionally, if the model of the controlled plant is unknown, no a priori stability guarantees could be achieved. However, the resulting closed-loop system could be analysed a posteriori using any existing general method (see Section II-C).

Research Objective: Tackling all the shortcomings mentioned above in a holistic manner is a complex task that does not necessarily have a general solution. In this paper, we set out to derive a control adaptation strategy for *dynamic* controllers to handle deadline misses whilst providing a *simple* structure, *minimal* overhead, and that does not reduce the system performance under ideal conditions. To analyse the performance and robustness of the adaptive control strategy, we also seek to derive a stochastic mean-square analysis of the resulting closed-loop system to be performed a posteriori.

IV. Real-Time Controller Adaptation

In this section, we derive an adaptive implementation scheme for real-time controllers to address the problems discussed in Section III-B. We explain the intuition behind the proposed adaptation; then, starting from an arbitrary linear control algorithm, we show how to derive the adapted controller. The adaptive controller is complemented with a probabilistic analysis method to evaluate its effectiveness for a given control system and a given deadline miss model.

A. Adaptive Controller Synthesis

To simplify and generalise the adaptation approach, we do not assume any information about the control design, apart from the controller itself. In practice, this implies that we consider neither the specific control design technique, the control system requirements, nor the plant's dynamics. The controller is assumed to be given in state-space form, specified by (2). This assumption is made without loss of generality, since any realisable linear digital controller can be expressed in this form [6].

The controller behaves *ideally* when every control period includes one job completing the execution of Equation (2), i.e., every job hits its deadline. By iterating the equation, the desired controller state and control action at any future time step can be computed. We formally define this behaviour as follows:

Ideal Controller: We denote the *ideal controller*, $\mathcal{C}\,(q)$, as the discrete-time LTI controller $\mathcal{C}$ evolved over an interval of $q+1$ consecutive deadline hits. The controller state and output at the end of the interval are

$$\mathcal{C}\,(q):\begin{cases} z_{k+q+1} = F^{q+1} z_k + \sum_{i=0}^{q} F^i G y_{k+q-i} \\ u_{k+q+1} = H\,F^q\,z_k + \\ \quad + H\,\sum_{i=1}^{q}\left[F^{i-1}\,G\,y_{k+q-i}\right] + K\,y_{k+q}. \end{cases} \tag{4}$$

With $q = 0$ we obtain the ideal controller behaviour over a single time step.

From the definition above, we see how the quantities of interest, z and u, are computed using the values of y in the interval $[k, k+q]$. These values correspond to the periodic measurements coming from the sensors. In the presence of deadline misses, the control task discards the measurements and does not update the controller's state. Thus, both the state and output of the controller deviate from their desired behaviours. Applying Equation (2) to a scenario of q consecutive deadline misses, we obtain the following:

Nominal Controller: We denote the *nominal controller*, $\mathcal{C}^n\,(q)$, as the discrete-time LTI controller $\mathcal{C}$ evolved over an interval of q consecutive deadline misses followed by one deadline hit. After the hit, the state and output are given by

$$\mathcal{C}^n\,(q):\begin{cases} z_{k+q+1} = F z_k + G y_{k+q} \\ u_{k+q+1} = H z_k + K y_{k+q}. \end{cases} \tag{5}$$

In the absence of deadline misses, we have $\mathcal{C}^n\,(0) = \mathcal{C}$.

By comparing (4) and (5) we see that, when deadline misses occur, the controller state z_{k+q+1} and control action u_{k+q+1} diverge from the ideal values, i.e., the ones that would be obtained in the presence of only deadline hits. To compensate for this error (and consequently minimise it), we propose to dynamically alter the controller's computations in real time according to the number of deadline misses.

Ideally, we would like an adaptive controller $\mathcal{C}^a\,(q)$ that mimics the ideal controller $\mathcal{C}\,(q)$ for any q. However, this is infeasible due to how the controller's dynamics evolves during an interval of deadline misses. More specifically, Equation (4) shows that $\mathcal{C}\,(q)$ depends on the values of y_{k+i} for $i \in \{0, 1, \ldots, q-1\}$. When the control task is subjected to faults, the corresponding jobs j_{k+i} miss their respective deadline. Thus, the unfinished jobs are terminated prematurely, a new job is released, and the corresponding measurement values y_{k+i} are lost. The nominal controller, after a series of misses, has access only to the sample y_{k+q}. This fundamentally limits how well the controller's ideal behaviour can be reconstructed.

Hence, we propose the use of an interpolation scheme to minimise the effect of the lost measurement values. We approximate the missing measurement values using a linear interpolation between y_{k-1} and y_{k+q} according to

$$\hat{y}_{k+q-i} = \frac{i y_{k-1} + (q+1-i) y_{k+q}}{q+1}. \tag{6}$$

If we substitute the values of y that are missing in the ideal controller, Equation (4), with the corresponding interpolated ones $\hat{y}$ from Equation (6), we obtain an adaptive controller as follows:

Adaptive Controller: We denote the *adaptive controller*, $\mathcal{C}^a\,(q)$, as an adaptation of the controller $\mathcal{C}$ evolved over an interval of q consecutive deadline misses followed by one deadline hit. After the hit, the state and output are

$$\mathcal{C}^a\,(q):\begin{cases} z_{k+q+1} = F_z(q) z_k + F_y(q) y_{k-1} + G_y(q) y_{k+q} \\ u_{k+q+1} = H_z(q) z_k + H_y(q) y_{k-1} + K_y(q) y_{k+q}, \end{cases} \tag{7}$$

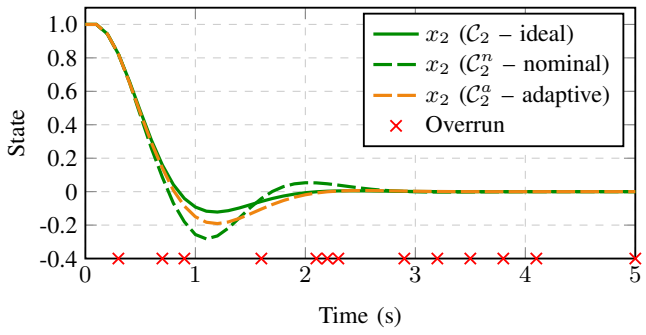

Fig. 3. The state x_2 corresponding to $\mathcal{P}$ being controlled by the LQG controller $\mathcal{C}_2$ (dashed green) or adaptive LQG (dashed orange) controller during an interval of sporadic overloads (×). The corresponding ideal behaviour (system not subject to overruns, solid green) is also plotted.

where

$$
\begin{aligned}
F_z(q) &= F^{q+1} \\
F_y(q) &= \textstyle\sum_{i=0}^{q} \frac{i}{q+1} F^i G \\
G_y(q) &= \textstyle\sum_{i=0}^{q} \frac{q+1-i}{q+1} F^i G \\
H_z(q) &= H F^q \\
H_y(q) &= H \textstyle\sum_{i=1}^{q} \frac{i}{q+1} F^{i-1} G \\
K_y(q) &= K + H \textstyle\sum_{i=1}^{q} \frac{q+1-i}{q+1} F^{i-1} G.
\end{aligned}
$$

Note that for $q = 0$ we recover the original controller.

All matrices in $\mathcal{C}^a(q)$ ($q \in \{0, \ldots, q_{max}\}$) can be precomputed and stored in memory. Compared to the nominal controller, the two matrices F_y and H_y of size $n_z \times n_y$ and $n_u \times n_y$ need to be added to the controller, and one full set of controller matrices needs to be stored for each value of q. The memory requirement thus grows linearly with q_{max}. The adaptive controller also needs to store the old measurement value y_{k-1}, which makes it marginally more complex.

We now briefly discuss how the limitations described in Section III-B are addressed. This work proposes an adaptation of the F, G, H, and K matrices of the controller, thus adapting also the *dynamic* part of the controller. The adaptive controller's matrices can be precomputed directly from the specified controller using Equation (7), and then stored in memory for each $q \leq q_{max}$. The control algorithm can be developed independently of this adaptation, hence no extra design effort is needed. For $q = 0$ the adaptive and the ideal controllers are equivalent, i.e., $\mathcal{C}^a(0) = \mathcal{C}(0)$, thus guaranteeing that the controller's performance is not degraded under ideal conditions. Concerning the limitation of increased runtime overhead, each time a job is released, the counter q of immediately preceding deadline misses is evaluated and the corresponding controller matrices are selected. Hence, the execution time of the adaptive controller will be independent of q and only marginally longer than for the nominal controller.

In Section III-B, we presented a motivating example of why it is insufficient to analyse static controllers subject to overruns. We conclude this section with a continuation of this example showing the benefits of our scheme.

Application to Motivating Example: Consider the plant $\mathcal{P}$, from Equation (3), controlled by the LQG controller $\mathcal{C}_2$. We apply our proposed scheme to $\mathcal{C}_2$, obtaining the corresponding adaptive controller $\mathcal{C}_2^a(q)$. In Figure 3, we compare the plant state obtained by controlling the plant using the two controllers, subject to the same deadline misses as in Figure 2. The dashed green line and the dashed orange line correspond, respectively, to the nominal LQG (the same as in Figure 2) and adaptive LQG. The solid green line corresponds to the ideal controller's behaviour, i.e., in the absence of deadline misses. We note a significant improvement in the performance of the adaptive LQG compared to the nominal LQG: the oscillations disappear and the state converges faster.

The proposed adaptive controller is inspired by the ideal controller $\mathcal{C}(q)$, but there is no a priori guarantee that it will perform better than the nominal controller $\mathcal{C}^n(q)$. However, for a given plant and deadline miss model, the closed-loop system can be analysed and compared for different controllers. Since the actual sequence of hits and misses is generally unpredictable, in the next section, we propose a probabilistic method to compare the performance of the adaptive controller with the nominal controller.

B. Stochastic Performance Analysis

To analyse the performance of the closed-loop system, we need to model the plant, the external disturbance w, and the sequences of deadline misses. For what concerns the plant, we consider the standard state-space model presented in Equation (1). The plant disturbance w and the sequence of deadline hits and misses are modelled as stationary random processes with known statistical properties. This allows us to analyse the time-averaged performance of the closed-loop system subject to deadline misses. The analysis offered here is inspired by [47], which in turn relies on classical results for jump linear systems [10].

The control performance is measured in terms of the weighted stationary variance of the plant output y,

$$J = \mathbb{E}\left[y_k^{\mathrm{T}} Q y_k\right], \tag{8}$$

where $Q \in \mathbb{R}^{n_y \times n_y}$ is a positive semidefinite weighting matrix[2]. A smaller value of the cost J is better, as it means that the controller can suppress the disturbance w more effectively. Additionally, the closed loop is guaranteed to be *stable* in the mean-square sense as long as J is finite [22], i.e., $J < \infty$.

The state of the complete system consists of the plant and the controller in feedback interconnection, together with the buffered previous measurement value and the calculated next control signal. Accordingly, we define the closed-loop state vector as

$$\xi_k = \begin{bmatrix} x_k \\ z_k \\ y_{k-1} \\ u_k \end{bmatrix}.$$

[2]Without loss of generality, we assume that zero is the desired plant output. A non-zero reference value can be modelled as an exogenous signal that is stored in the plant and offset in the measurement signal.

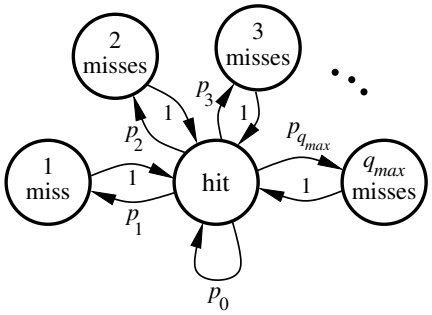

Fig. 4. Markov model for the random sequence of hits and misses.

As seen in Figure 1, the only external input of the closed-loop system is the disturbance w. Hence, the state of the closed loop evolves according to the time-varying linear system

$$\xi_{k+1} = \Phi_k \xi_k + \Gamma w_k. \tag{9}$$

Here,

$$\Gamma = \begin{bmatrix} W \\ 0_{(n_z+n_y+n_u)\times n_w} \end{bmatrix},$$

while Φ_k can be either $\Phi_{hit}(q)$ or Φ_{miss}, depending on whether the current deadline was hit or missed, and defined as

$$\Phi_{hit}(q) = \begin{bmatrix} A & 0 & 0 & B \\ G_y(q)\,C & F_z(q) & F_y(q) & G_y(q)\,D \\ C & 0 & 0 & D \\ K_y(q)\,C & H_z(q) & H_y(q) & K_y(q)\,D \end{bmatrix},$$

where q denotes the number of consecutive deadline misses since the last hit, and

$$\Phi_{miss} = \begin{bmatrix} A & 0 & 0 & B \\ 0 & I_{n_z} & 0 & 0 \\ 0 & 0 & I_{n_y} & 0 \\ 0 & 0 & 0 & I_{n_u} \end{bmatrix}.$$

Assuming that w is a zero-mean white noise process with known covariance matrix $R = \mathbb{E}\big[w_k \cdot w_k^{\mathrm{T}}\big] \in \mathbb{R}^{n_w \times n_w}$, we can calculate how the state covariance, $\Pi_k = \mathbb{E}\big[\xi_k \cdot \xi_k^{\mathrm{T}}\big]$, evolves over time. Evaluating the covariance of both sides in Equation (9) we obtain (see, e.g., [7])

$$\Pi_{k+1} = \Phi_k \Pi_k \Phi_k^{\mathrm{T}} + \Gamma R \Gamma^{\mathrm{T}}. \tag{10}$$

We model the task execution as a random process, assuming that the pattern of hits and misses in the real-time system is described by the homogeneous Markov model shown in Figure 4. In this model, after each hit, the system will experience a miss interval of length $q \in \{0, \ldots, q_{max}\}$ with independent probability p_q. Naturally, $\sum_{i=0}^{q_{max}} p_i = 1$.

In each interval, the system will experience q deadline misses followed by one deadline hit. Iterating (10) over said interval, the covariance will then develop as

$$\begin{aligned} \Pi_{k+q+1} = \Phi_{hit}(q)\Big(& (\Phi_{miss})^q \Pi_k (\Phi_{miss}^{\mathrm{T}})^q \\ & + \textstyle\sum_{i=0}^{q-1} (\Phi_{miss})^i \Gamma R \Gamma^{\mathrm{T}} (\Phi_{miss}^{\mathrm{T}})^i \Big) \Phi_{hit}^{\mathrm{T}}(q) \\ & + \Gamma R \Gamma^{\mathrm{T}}. \end{aligned}$$

The time-varying closed-loop system together with the Markov model define a *discrete-time Markov jump linear system* for which well-established results exist (e.g., [10], [37], [47]). Using this theory, it is possible to calculate the *stationary* (time-averaged) state covariance, denoted $\overline{\Pi}$. With this, the performance (8) can finally be obtained as

$$J = \operatorname{trace}\left(\overline{\Pi}\,\overline{Q}\right),$$

where

$$\overline{Q} = \begin{bmatrix} C^{\mathrm{T}} Q C & 0 \\ 0 & 0_{n_z+n_y+n_u} \end{bmatrix}.$$

To compare the performance of different implementations, we first define the *ideal performance* as the cost J obtained when there are no deadline misses (i.e., $p_0 = 1$ and $p_i = 0$, $i \geq 1$). We then obtain the *relative performance degradation* of an arbitrary controller $\mathcal{C}^\dagger$ by calculating the weighted mean-square difference between the actual and ideal systems' outputs, $y_k^\dagger$ and y_k respectively, and normalising it with respect to the ideal performance J:

$$\frac{\Delta J^\dagger}{J} = \frac{\mathbb{E}\big[(y_k^\dagger - y_k)^{\mathrm{T}} Q (y_k^\dagger - y_k)\big]}{\mathbb{E}\big[y_k^{\mathrm{T}} Q y_k\big]}. \tag{11}$$

This can be found by analysing both systems in parallel when driven by the same noise sequence w_k:

$$\begin{bmatrix} \xi_{k+1}^\dagger \\ \xi_{k+1} \end{bmatrix} = \begin{bmatrix} \Phi_k^\dagger & 0 \\ 0 & \Phi_k \end{bmatrix} \begin{bmatrix} \xi_k^\dagger \\ \xi_k \end{bmatrix} + \begin{bmatrix} \Gamma \\ \Gamma \end{bmatrix} w_k$$

After finding the stationary state covariance $\overline{\Pi}_e$ of this extended system (using the same technique as referred to above), we can retrieve the absolute performance difference as

$$\Delta J^\dagger = \operatorname{trace}\left(\overline{\Pi}_e \begin{bmatrix} \overline{Q} & -\overline{Q} \\ -\overline{Q} & \overline{Q} \end{bmatrix}\right).$$

V. Experimental Evaluation

In this section, we compare our adaptive controller with the nominal controller implementation for different case studies. We demonstrate the practical usefulness of the proposed controller by examining its impact on real hardware, namely, a ball and beam plant. We compare the performance of the adaptive control system with the nominal one, according to the analysis presented in Section IV-B. Finally, we complement the results with a worst-case switching stability analysis of the nominal and adaptive controlled systems.

In addition to the evaluation on the physical system, we present aggregate results obtained from a set of control benchmarks, representative of the process industry. We use this set of plants to evaluate the general applicability of our approach. To make the evaluation comprehensive, we chose an unstable plant (the ball and beam) for the physical experiments and a set of mainly stable plants for the aggregate results. Furthermore, we remark that all the considered controllers are dynamic. As discussed in Section III-A, to the best of our knowledge, only one previous work considers dynamic controllers [49]. In that work, however, a different overrun handling method is used, and a proper comparison is therefore not possible.

TABLE I

ANALYTICAL STUDY OF THE RELATIVE PERFORMANCE DEGRADATION OF THE BALL AND BEAM PLANT $\mathcal{P}$ USING EITHER THE NOMINAL $\mathcal{C}^n$ OR ADAPTIVE CONTROLLER $\mathcal{C}^a$.

ρ	10%	20%	30%	40%	50%	60%	70%
$\Delta J^n/J$	2.5%	9.2%	20.8%	39.9%	75.3%	156%	452%
$\Delta J^a/J$	0.1%	0.1%	0.3%	0.6%	1.1%	2.5%	6.7%

TABLE II

EMPIRICAL STUDY OF THE RELATIVE PERFORMANCE DEGRADATION OF THE REAL BALL AND BEAM PLANT USING EITHER THE NOMINAL $\mathcal{C}^n$ OR ADAPTIVE CONTROLLER $\mathcal{C}^a$.

ρ	10%	20%	30%	40%	50%	60%	70%
$\Delta J^n/J$	6.3%	27.1%	22.5%	50.5%	73.1%	260%	∞
$\Delta J^a/J$	6.1%	7.8%	3.2%	4.6%	3.8%	4.9%	11.7%

A. Real World Evaluation – Ball and Beam

System Description and Models: The ball and beam [69] is a common example in the automatic control literature and education, where a ball is free to roll over a beam that in turn is tilted by a servo motor. The control objective is to make the ball position follow a reference trajectory across the beam by adjusting the voltage sent to the motor. Both the beam angle and the ball position can be measured. Assuming the sampling period $T = 0.01$ s, a discrete-time plant model $\mathcal{P}$ was derived as

$$\mathcal{P}: \begin{cases} x_{k+1} = \begin{bmatrix} 1 & 0.015 & 0.0003 & 0 \\ 0 & 1 & 0.045 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 2.9\cdot 10^{-5} \\ 0.0058 \\ 0.256 \\ 0 \end{bmatrix} u_k + w_k \\ y_k = \begin{bmatrix} 0.5 & 0 & 0 & -1 \\ 0 & 0 & 0.25 & 0 \end{bmatrix} x_k, \end{cases}$$

where the four components of x_k represent the ball position, ball velocity, beam velocity, and ball reference, respectively. The external signal vector w_k is assumed to be white noise with variance $R = \mathrm{diag}(1, 1, 1, 1)$. Under this state-space model, the objective is to regulate both outputs y_k to zero, with the performance weighting matrix $Q = \mathrm{diag}(1, 1)$.

To control $\mathcal{P}$ we design a cascaded P–PID controller. Cascaded controllers are frequently applied to systems with multiple measurements where one measured quantity affects another, but not vice versa. Thus, the plant measurements can be controlled in sequential order (hence the naming *cascaded*) using a controller designed for each measurement signal. In our case study, this is implemented by a proportional (P) controller designed for controlling the beam's angle and a proportional–integral–derivative (PID) controller for the ball's position. The controller is run as a periodically executing task with period $T = 0.01$ s on a single core CPU where overrun deadlines are killed and the corresponding sensor data is discarded. In between actuator calls, the control signal is assumed to be held constant. The state-space representation of our controller is

$$\mathcal{C}: \begin{cases} z_{k+1} = \begin{bmatrix} 1 & 0 \\ 0 & 0.9685 \end{bmatrix} z_k + \begin{bmatrix} 0.025 & 0 \\ -0.2608 & 0 \end{bmatrix} y_k, \\ u_{k+1} = \begin{bmatrix} -0.108 & -0.2608 \end{bmatrix} z_k + \begin{bmatrix} -2.43 & -3 \end{bmatrix} y_k. \end{cases}$$

Experiments Design: We apply the performance analysis presented in Section IV-B to the plant model $\mathcal{P}$ controlled using either the ideal ($\mathcal{C}$), nominal ($\mathcal{C}^n$), or adaptive ($\mathcal{C}^a$) implementations from Section IV-A. We include the effect of deadline misses only on the nominal and adaptive control systems. The probability distribution p_q can be chosen arbitrarily according to the desired task model. For simplicity, we assume here that the deadline misses are Bernoulli distributed [56], i.e., the probabilities of missing deadlines in each period are independently and identically distributed with probability ρ. This results in the probability $p_q = (1-\rho)\rho^q$ of q consecutive deadline misses followed by a hit. We assume that no more than $q_{max} = 20$ consecutive deadlines can be missed. The latter assumption might seem restrictive, but if the probability of missing a deadline is 30%, the probability of missing 20 consecutive deadlines is less than $4 \cdot 10^{-11}$.

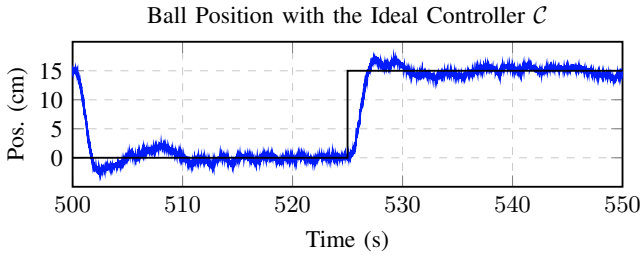

Fig. 5. Snippet of the test performed on the real ball and beam plant using the ideal controller, i.e., without deadline misses. The plot shows one period of the square wave used as reference, the black line. The blue line shows the ball's position.

We measure the relative performance of the nominal and adaptive controllers according to the quantity $\Delta J^\dagger/J$ in Equation (11). Since the mean-square deviation from the ideal controller is used to evaluate the relative performance, the *optimal* achievable cost is 0. For the real system, we do not feed the system with white noise, but we expose the system to a repeatable exogenous signal in the form of periodic reference changes. Furthermore, we evaluate the relative performance degradation empirically from the measured signals using Equation (11), with $\mathbb{E}[\cdot]$ being interpreted as the mean value of the real signals.

To complement the performance analysis, we perform a JSR stability analysis on the model to determine the maximum number of consecutive deadline misses that are tolerated while still guaranteeing closed-loop stability.

Analytical Evaluation: The performance results obtained with the analytical study of $\mathcal{P}$ for different values of ρ are summarised in Table I. From the table, we can see that the adaptive controller drastically improves the relative performance

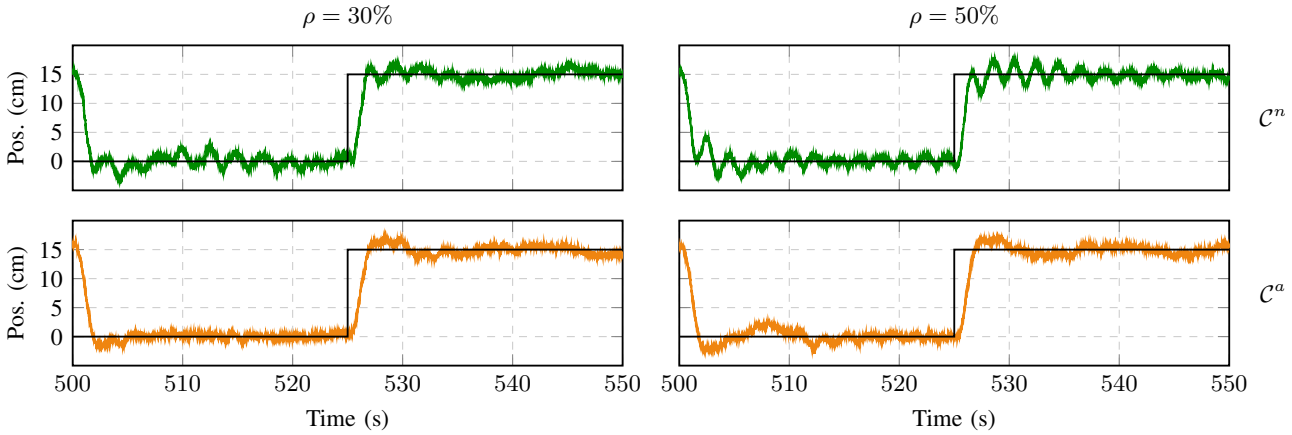

Fig. 6. Snippets of the tests performed on the real ball and beam plant for $\rho = 30\%$ (two left plots) and $\rho = 50\%$ (two right plots). The plots show one period of the square wave used as reference, the black line. The coloured lines show the ball position, in green for the nominal controller (upper plots) and orange for the adaptive controller (lower plots).

(in comparison to the nominal controller) across all deadline miss probabilities. Already for small probabilities, the nominal controller significantly degrades the relative performance compared to the ideal controller; e.g., for $\rho = 30\%$ the relative performance is degraded by 20.8%. This can be compared to the adaptive controller, where the relative performance reduction stays below 5% until the miss probability reaches 70%.

Analysing the switching stability, we calculated the JSR for the set of closed-loop matrices corresponding to $i = \{0, 1, \ldots, q\}$ consecutive deadline misses followed by one hit ($q \leq q_{max}$). The nominal control system is guaranteed to be switching stable (i.e., the JSR is below 1) for a maximum of $q = 2$ consecutive deadline misses, while the adaptive control system is guaranteed stable up to $q = 8$. We conclude that the adaptive controller improves also worst-case robustness against deadline misses for the ball and beam. However, we emphasise that these results do *not* imply that the system will go unstable if more deadline misses occurs; only that the system is *guaranteed* switching stable if no more than q consecutive deadline misses are ever experienced.

Empirical Evaluation: We conducted experiments on the physical ball and beam plant to evaluate the performance of the controller on a real system.[3] Each experiment is run for 10 minutes, where the control objective is for the ball to follow a square-wave reference across the beam. The square wave has a period of 50 s and alternates between position 0 and 15 cm. Differently from the analytical evaluation, it is impossible to obtain the same exogenous signal w_k in the different experiments. While the reference changes can be exactly repeated, the real stochastic disturbances (in the form of electrical noise, mechanical glitches, etc.) are not repeatable. This means that the empirical cost relative to the ideal case, as measured by Equation (11), is not expected to be zero even in the complete absence of deadline misses.

Figure 5 displays a snippet of the ball's position (blue line) under said ideal conditions. The ball quite successfully follows the reference (black line). Here, the fluctuations around the reference are caused by measurement noise and irregularities in the beam surface, where the latter can cause the ball to get lodged in an undesired position and thus result in oscillations.

After measuring the performance of the ideal controller, each controller ($\mathcal{C}^n$ and $\mathcal{C}^a$) was applied to the system, using probabilities $\rho \in \{10\%, 20\%, \ldots, 70\%\}$ of missing each deadline (with $q_{max} = 20$). The results of the experiments are reported in Table II, where the relative performance degradation $\Delta J^\dagger / J$ is computed for both the nominal and adaptive controllers. To give an intuition for how the physical system behaves, in Figure 6 we provide a snippet of a time plot portraying the ball's position controlled by either the nominal (upper plots) or adaptive (lower plots) controller. We distinguish the differences between the nominal and adaptive controllers for a probability $\rho = 30\%$ (left plots) of missing a deadline. The nominal controller shows oscillations around the reference value. When the probability of missing a deadline is increased to $\rho = 50\%$ (right plots), the nominal controller's oscillations grow more evident, while the adaptive controller appears unaffected (compared to the ideal controller in Figure 5).

From Table II, we observe that the adaptive controller has a lower performance degradation across all deadline miss probabilities $\rho \geq 20\%$ compared to the nominal controller. The performance of the adaptive controller seems virtually unaffected for $\rho \leq 60\%$, where the baseline relative degradation of approximately 4% to 8% is due to the natural disturbances in the system. The nominal controller on the other hand experiences significant performance degradation at higher miss probabilities, and for $\rho = 70\%$ the system becomes unstable – we report this as an infinite cost.

In summary, both the analytical and empirical studies show that the adaptive controller $\mathcal{C}^a$ consistently outperforms the

[3] A video, showing experiments with the real ball and beam system can be viewed at `https://youtu.be/6y_C7NIzXto`. The video provides a real-world comparison between the nominal and adaptive controllers for $\rho = \{30\%, 50\%, 70\%\}$.

nominal controller $\mathcal{C}^n$ for the ball and beam. Furthermore, the adaptive controller can tolerate a large likelihood of random deadline misses (at least 60%) without any noticeable performance degradation.

B. Benchmark Evaluation – Process Industry

System Description and Models: To evaluate the general applicability of the proposed adaptive controller, we perform an extensive evaluation campaign on a benchmark set of plants. The set was developed specifically to evaluate various PID designs [4] in the process industry. It consists of 134 unique plants separated into 9 categories, where each category has its own specific properties frequently recognised in the process industry. Since the benchmark was developed specifically with process industrial plants in mind, the majority of the plants are stable, i.e., all their eigenvalues lie inside the unit circle. However, there are also plants with integrating dynamics included in the benchmark, i.e., an eigenvalue in 1; these plants are generally not considered stable. For each plant, two controllers – a PI and a PID controller – are optimised using known methods [24]; hence, 268 unique control systems are analysed in total.

Experiments Design: Similarly to the ball and beam, we analyse the relative performance of the nominal and adaptive controllers in accordance with the analysis described in Section IV-B. We again consider the probability of missing a deadline to follow a Bernoulli distribution with probabilities $\rho \in \{10\%, 30\%, 50\%, 70\%\}$ and a maximum of $q_{max} = 20$ consecutive deadline misses. We feed the systems with a stochastic disturbance and analytically evaluate the ability of the controllers to reject it. Differently from the ball and beam, we analyse the systems when subject to brown noise, i.e., integrated white noise [58]. The brown noise model is generally considered appropriate for process industrial plants since it is dominant for low frequencies (e.g., load disturbances and disturbances from nearby heavy machinery). We assume that the *same* disturbance process enters the ideal, nominal, and adaptive control systems; this guarantees an unbiased comparison between the different controllers. For each of the 268 control systems we calculate the relative performance $\Delta J^\dagger/J$ for both the nominal and the adaptive controller.

Similarly to the ball and beam, we complement our performance analysis with a JSR worst-case stability analysis.

Experiments Results: In Figure 7 we display histograms reporting the relative performance degradation of all the 268 control systems. The horizontal axis displays the relative performance $\Delta J^\dagger/J$ in logarithmic scale. The vertical axis counts the number of control systems with a given relative performance. The four plots correspond to the different deadline miss probabilities considered. In each plot, we represent the nominal controllers with green bars and the adaptive controllers with orange bars. Unstable closed-loop systems have an infinite cost and are thus marked in the rightmost part of the plot, beyond the red dashed threshold.

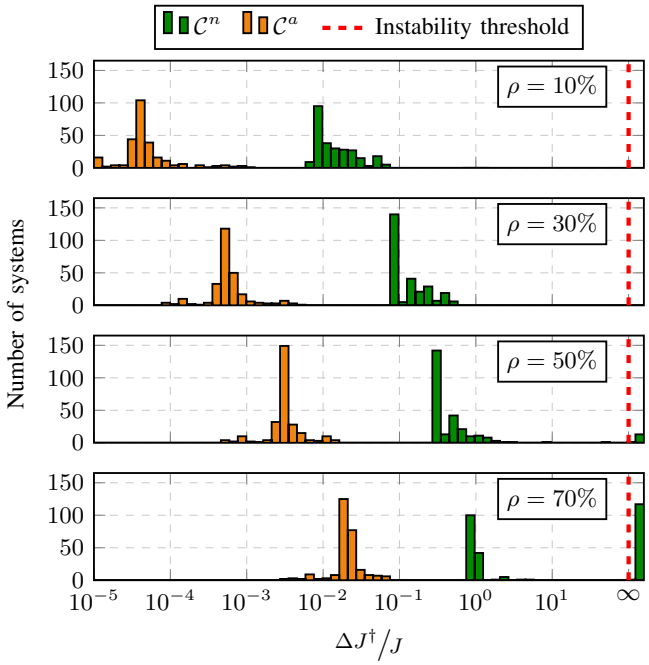

Fig. 7. Histograms comparing the relative performance degradation of the nominal and adaptive controllers for the benchmark plants. The plots correspond to different deadline miss probabilities ρ. The orange bars report the performance obtained with the adaptive controller $\mathcal{C}^a$, while the green bars report the performance obtained with the nominal controller $\mathcal{C}^n$. The systems with a performance worse than the stability threshold (red dashed line) resulted in unstable dynamics.

From Figure 7 we see that the adaptive controller performs better than the nominal one for *all* the 268 control systems, regardless of the probability of missing a deadline. Despite the control systems' dynamics varying significantly (e.g., lag dominated, lead dominated, oscillatory, high system order, integrating), the worst adaptive control system still performs better than the best nominal control system for all ρ. The improvement is particularly distinguishable for lower probabilities, e.g., $\rho = 10\%$, where the mean relative cost over all the control systems is improved by two orders of magnitude.

Second, when the probability of missing a deadline grows, the relative performance degradation increases accordingly. For $\rho = 50\%$ and $\rho = 70\%$ some of the systems using the nominal controller become unstable, i.e., $\Delta J^\dagger/J = \infty$. In the case of $\rho = 70\%$, more than 40% of the nominal control systems are unstable. On the other hand, *all* the adaptive control systems are stable and have a relative cost degradation below 10%. This suggests that $\mathcal{C}^a$ improves both performance and robustness compared to the nominal controller.

To verify that the adaptive controller improves the robustness to deadline misses compared to the nominal controller, we complement the evaluation with a JSR analysis. The histogram in Figure 8 shows, for each value of q, the number of control systems that are guaranteed switching stable for a maximum number of consecutive deadline misses q, when they are controlled with either the nominal ($\mathcal{C}^n$) or the adaptive ($\mathcal{C}^a$) controller. Intuitively, the more control systems that can

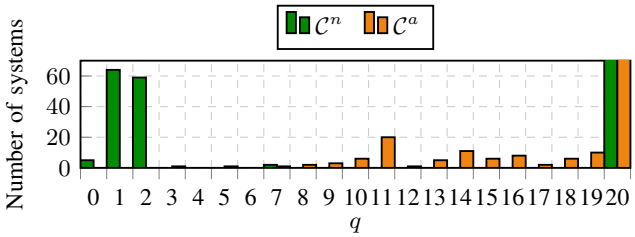

Fig. 8. Histogram reporting the number of benchmark systems (out of 268) that are guaranteed switching stable for up to q consecutive deadline misses, according to the JSR analysis. For each value of q, the green bar (left) reports how many $\mathcal{C}^n$ controlled systems can tolerate up to q consecutive deadline misses and the orange bar (right) reports the corresponding number of $\mathcal{C}^a$ controlled systems. For readability the y axis is cut at 65: a total of 138 plants can tolerate 20 or more misses with the nominal controller, and a total of 185 plants can tolerate 20 or more misses with the adaptive controller.

guarantee switching stability for a higher value of q, the better. The vertical axis of the histogram is cut at 65 for legibility: this affects only the columns for $q = 20$ where the nominal controller can guarantee stability for 138 plants while the adaptive controller can guarantee stability for 185 plants.

For the nominal controller, we see that the maximum number of consecutive deadline misses tolerated by the system varies greatly between the different control systems. In the whole benchmark, 138 systems were stable for (at least) 20 consecutive deadline misses, but 123 systems were guaranteed stable only for one or two misses. Furthermore, 5 of the nominal control systems were unstable unless *all* of the control task's deadlines were hit.

For the adaptive controller, on average, a much larger number of consecutive deadline misses can be tolerated. Out of all the control systems, 185 were stable for (at least) $q = 20$, and the large majority of the remaining control systems are guaranteed to tolerate between $q = 10$ and $q = 19$ consecutive deadline misses. Additionally, we see that *all* adaptively controlled systems can tolerate at least 3 deadline misses.

We note that both for the nominal and adaptive controllers, a significant number of control systems are stable for 20 deadline misses. This presumably follows from the (mainly) stable nature of the plants in the benchmark, an attribute that generally makes the system more robust.

The results of the evaluation campaign confirm the hypothesis that the proposed adaptive controller improves the control system's performance in the presence of deadline misses. While we observed some cases in which the nominal controller goes unstable and the adaptive controller is stable, we never observed the opposite. Additionally, the adaptive controller does not compromise the performance under ideal conditions, and it preserves the major part of the ideal controller's performance when deadline misses are present.

VI. Conclusion

In this paper, we have proposed a novel adaptive implementation scheme for real-time embedded controllers, aiming to increase their robustness to arbitrary patterns of deadline misses. The approach adapts a general, predesigned linear controller according to the number of consecutive deadline overruns. No additional assumptions are made on the predesigned controller or the plant model, and only minimal requirements are put on the control task (Logical Execution Time) and the real-time operating system (late tasks being killed). The adaptation scheme can be implemented without any additional complexity or design overhead, hence strengthening the industrial applicability. Additionally, we extend the state-of-the-art by considering dynamic controllers.

To analyse our controller implementation, we developed a probabilistic approach. With said approach, it is possible to leverage a plant model and a probabilistic model of deadline misses to evaluate the effectiveness of the adaptive controller. We complement this average-case analysis with a worst-case stability analysis based on JSR. We used both approaches to evaluate the controller on both a physical plant and a considerable number of simulated plants from the literature on process control. The results show that our adaptive controller implementation consistently improves system performance and robustness to deadline misses.

References

[1] B. Akesson, M. Nasri, G. Nelissen, S. Altmeyer, and R. I. Davis. An empirical survey-based study into industry practice in real-time systems. In *41st IEEE Real-Time Systems Symposium*, 2020.

[2] A. Aminifar, S. Samii, P. Eles, and Z. Peng. Control-quality driven task mapping for distributed embedded control systems. In *17th International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 133–142, 2011.

[3] K.-E. Årzén, A. Cervin, J. Eker, and L. Sha. An introduction to control and scheduling co-design. In *39th IEEE Conference on Decision and Control*, pages 4865–4870, 2000.

[4] K. Åström and T. Hägglund. Revisiting the Ziegler—Nichols step response method for PID control. *Journal of Process Control*, 14(6):635–650, 2004.

[5] K. J. Åström and T. Hägglund. Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, 20(5):645–651, 1984.

[6] K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[7] K. J. Åström and B. Wittenmark. *Computer controlled systems: Theory and design (1 ed.)*. Prentice-Hall, 1984.

[8] S. K. Baruah and J. R. Haritsa. Scheduling for overload in real-time systems. *IEEE Transactions on Computers*, 46(9):1034–1039, 1997.

[9] G. Bernat, A. Burns, and A. Liamosi. Weakly hard real-time systems. *IEEE Transactions on Computers*, 50(4), 2001.

[10] W. P. Blair Jr. and D. D. Sworder. Feedback control of a class of linear discrete systems with jump parameters and quadratic cost criteria. *International Journal of Control*, 21(5):833–841, 1975.

[11] M. Caccamo, G. Buttazzo, and Lui Sha. Elastic feedback control. In *12th Euromicro Conference on Real-Time Systems*, pages 121–128, 2000.

[12] M. Caccamo, G. Buttazzo, and L. Sha. Handling execution overruns in hard real-time control systems. *IEEE Transactions on Computers*, 51(7):835–849, July 2002.

[13] E. F. Camacho, T. Alamo, and D. M. de la Peña. Fault-tolerant model predictive control. In *15th IEEE Conference on Emerging Technologies in Factory Automation*, pages 1–8, 2010.

[14] A. Cervin. Analysis of overrun strategies in periodic control tasks. In *16th IFAC World Congress*. Elsevier, 2005.

[15] 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 *10th International Conference on Real-Time and Embedded Computing Systems and Applications*, 2004.

[16] K.-H. Chen and J.-J. Chen. Probabilistic schedulability tests for uniprocessor fixed-priority scheduling under soft errors. In *12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 1–8, 2017.

[17] K.-H. Chen, G. Von Der Brüggen, and J.-J. Chen. Analysis of deadline miss rates for uniprocessor fixed-priority scheduling. In *24th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 168–178, 2018.

[18] A. Crespo, I. Ripoll, and P. Albertos. Reducing delays in RT control: The control action interval. In *14th IFAC World Congress 1999, Beijing, China*, 1999.

[19] L. Desborough and R. Miller. Increasing customer value of industrial control performance monitoring—Honeywell's experience. *AIChE Symposium Series*, 98, 01 2002.

[20] J. Eker and A. Cervin. A Matlab toolbox for real-time and control systems co-design. In *6th International Conference on Real-Time Computing Systems and Applications*, Dec 1999.

[21] R. Ernst, S. Kuntz, S. Quinton, and M. Simons. The logical execution time paradigm: New perspectives for multicore systems. *Dagstuhl Reports*, 8(2):122–149, 2018.

[22] Y. Fang and K. Loparo. Stochastic stability of jump linear systems. *IEEE Transactions on Automatic Control*, 47(7):1204–1208, 2002.

[23] G. Frehse, A. Hamann, S. Quinton, and M. Woehrle. Formal analysis of timing effects on closed-loop properties of control software. In *35th IEEE Real-Time Systems Symposium*, pages 53–62, 2014.

[24] O. Garpinger and T. Hägglund. A software tool for robust PID design. *IFAC Proceedings Volumes*, 41(2), 2008.

[25] R. A. Gupta and M. Chow. Networked control system: Overview and research trends. *IEEE Transactions on Industrial Electronics*, 57(7):2527–2535, 2010.

[26] T. Hägglund and K. J. Åström. A method and an apparatus in tuning a PID-regulator, 1983. Patent application WO/1983/000753.

[27] Z. A. H. Hammadeh, S. Quinton, R. Henia, L. Rioux, and R. Ernst. Bounding deadline misses in weakly-hard real-time systems with task dependencies. In *Design Automation and Test in Europe (DATE)*, Lausanne, Switzerland, Mar. 2017.

[28] T. Henzinger, B. Horowitz, and C. Kirsch. Giotto: A time-triggered language for embedded programming. *Proceedings of the IEEE*, 91(1):84–99, 2003.

[29] M. Hertneck, S. Linsenmayer, and F. Allgöwer. Nonlinear dynamic periodic event-triggered control with robustness to packet loss based on non-monotonic Lyapunov functions. In *58th IEEE Conference on Decision and Control (CDC)*, 2019.

[30] M. Hertneck, S. Linsenmayer, and F. Allgöwer. Stability analysis for nonlinear weakly hard real-time control systems. *IFAC-PapersOnLine*, 53(2):2594–2599, 2020. 21st IFAC World Congress.

[31] M. Hertneck, S. Linsenmayer, and F. Allgöwer. Efficient stability analysis approaches for nonlinear weakly-hard real-time control systems. *Automatica*, 133:109868, 2021.

[32] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, 2007.

[33] M. Kauer, D. Soudbakhsh, D. Goswami, S. Chakraborty, and A. M. Annaswamy. Fault-tolerant control synthesis and verification of distributed embedded systems. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, 2014.

[34] C. Kirsch and A. Sokolova. The logical execution time paradigm. In *Advances in Real-Time Systems*, pages 103–120, 2012.

[35] P. Kumar, D. Goswami, S. Chakraborty, A. Annaswamy, K. Lampka, and L. Thiele. A hybrid approach to cyber-physical systems verification. In *49th Annual Design Automation Conference (DAC)*, page 688–696, New York, NY, USA, 2012.

[36] D. Liberzon. *Switching in Systems and Control*. Systems & control. Birkhauser, 2003.

[37] B. Lincoln and A. Cervin. Jitterbug: A tool for analysis of real-time control performance. In *Proceedings of the 41st IEEE Conference on Decision and Control*, pages 1319–1324 vol.2, 2002.

[38] S. Linsenmayer and F. Allgower. Stabilization of networked control systems with weakly hard real-time dropout description. In *56th IEEE Conference on Decision and Control (CDC)*, pages 4765–4770, 2017.

[39] S. Linsenmayer, B. W. Carabelli, S. Wildhagen, K. Rothermel, and F. Allgöwer. Controller and triggering mechanism co-design for control over time-slotted networks. *IEEE Transactions on Control of Network Systems*, 8:222–232, 2021.

[40] S. Linsenmayer, M. Hertneck, and F. Allgower. Linear weakly hard real-time control systems: Time- and event-triggered stabilization. *IEEE Transactions on Automatic Control*, 2020.

[41] C. Lozoya, P. Martí, M. Velasco, J. M. Fuertes, and E. X. Martin. Resource and performance trade-offs in real-time embedded control systems. *Real-Time Systems*, 49(3):267–307, 2013.

[42] M. Maggio, A. Hamann, E. Mayer-John, and D. Ziegenbein. Control-system stability under consecutive deadline misses constraints. In M. Völp, editor, *32nd Euromicro Conference on Real-Time Systems (ECRTS 2020)*, volume 165 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:24. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

[43] A. Marchand and M. Chetto. Dynamic scheduling of periodic skippable tasks in an overloaded real-time system. In *2008 IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, pages 456–464, 2008.

[44] F. Marković, A. V. Papadopoulos, and T. Nolte. On the convolution efficiency for probabilistic analysis of real-time systems. In B. B. Brandenburg, editor, *33rd Euromicro Conference on Real-Time Systems (ECRTS)*, volume 196 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:22, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[45] P. Marti, J. M. Fuertes, G. Fohler, and K. Ramamritham. Jitter compensation for real-time control systems. In *22nd IEEE Real-Time Systems Symposium (RTSS 2001)*, pages 39–48, 2001.

[46] M. Milligan and H. Cragon. The use of cache memory in real-time systems. *Control Engineering Practice*, 4(10):1435–1442, 1996.

[47] J. Nilsson, B. Bernhardsson, and B. Wittenmark. Stochastic analysis and control of real-time systems with random time delays. *Automatica*, 34(1):57–64, 1998.

[48] R. Oshana. Overview of embedded systems and real-time systems. In R. Oshana, editor, *DSP Software Development Techniques for Embedded and Real-Time Systems*, Embedded Technology, pages 19–34. Newnes, Burlington, 2006.

[49] P. Pazzaglia, A. Hamann, D. Ziegenbein, and M. Maggio. Adaptive design of real-time control systems subject to sporadic overruns. In *Design, Automation & Test in Europe Conference Exhibition (DATE)*, 2021.

[50] P. Pazzaglia, C. Mandrioli, M. Maggio, and A. Cervin. DMAC: Deadline-miss-aware control.

[51] K. Ramamritham. Where do time constraints come from? Where do they go? *International Journal of Database Management*, 7(2):4–11, 1996.

[52] P. Ramanathan. Graceful degradation in real-time control applications using (m,k)-firm guarantee. In *27th IEEE International Symposium on Fault Tolerant Computing*, pages 132–141, 1997.

[53] H. Rehbinder and M. Sanfridson. Integration of off-line scheduling and optimal control. In *12th Euromicro Conference on Real-Time Systems*, pages 137–143, 2000.

[54] G. Rota and W. Gilbert Strang. A note on the joint spectral radius. *Indagationes Mathematicae (Proceedings)*, 63, 1960.

[55] L. Schenato. To zero or to hold control inputs with lossy links? *IEEE Transactions on Automatic Control*, 54(5):1093–1099, 2009.

[56] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry. Foundations of control and estimation over lossy networks. *Proceedings of the IEEE*, 95(1):163–187, 2007.

[57] M. Schinkel and W.-H. Chen. Control of sampled data systems with variable sampling rate. *International Journal of Systems Science*, 37, July 2006.

[58] P. C. Schmidt. Handbook of stochastic methods for physics, chemistry and the natural sciences. *Berichte der Bunsengesellschaft für physikalische Chemie*, 89(6):721–721, 1985.

[59] Seong Woo Kwak, Byung Jae Choi, and Byung Kook Kim. An optimal checkpointing-strategy for real-time control systems under transient faults. *IEEE Transactions on Reliability*, 50(3):293–301, 2001.

[60] J. Stankovic, M. Spuri, M. D. Natale, and G. Buttazzo. Implications of classical scheduling results for real-time systems. *Computer*, 28(6):16–25, June 1995.

[61] G. Steinbauer. A survey about faults of robots used in RoboCup. In X. Chen, P. Stone, L. E. Sucar, and T. van der Zant, editors, *RoboCup 2012: Robot Soccer World Cup XVI*, pages 344–355, Berlin, Heidelberg, 2013. Springer.

[62] L. Sun, D. Li, and K. Y. Lee. Optimal disturbance rejection for PI controller with constraints on relative delay margin. *ISA Transactions*, 63:103–111, 2016.

[63] M. Törngren. Fundamentals of implementing real-time control applications in distributed computer systems. *Real-Time Systems*, 14(3):219–250, 1998.

[64] G. Vankeerberghen, J. Hendrickx, and R. M. Jungers. JSR: A toolbox to compute the joint spectral radius. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (HSCC)*, page 151–156. Association for Computing Machinery, 2014.

[65] N. Vreman, A. Cervin, and M. Maggio. Stability and performance analysis of control systems subject to bursts of deadline misses. In *33rd Euromicro Conference on Real-Time Systems (ECRTS)*, volume 196. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[66] N. Vreman and C. Mandrioli. Evaluation of burst failure robustness of control systems in the fog. In A. Cervin and Y. Yang, editors, *2nd Workshop on Fog Computing and the IoT (Fog-IoT 2020)*, volume 80 of *OpenAccess Series in Informatics (OASIcs)*, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[67] N. Vreman, P. Pazzaglia, J. Wang, V. Magron, and M. Maggio. Stability of control systems under extended weakly-hard constraints, 2021. arXiv preprint 2101.11312. https://doi.org/10.48550/arXiv.2101.1131.

[68] W. Wang, P. Mishra, and A. Gordon-Ross. Dynamic cache reconfiguration for soft real-time systems. *ACM Trans. Embed. Comput. Syst.*, 11(2), July 2012.

[69] P. E. Wellstead, V. Chrimes, P. R. Fletcher, and A. J. R. R. Moody. The ball and beam control experiment. *The International Journal of Electrical Engineering & Education*, 15(1), 1978.

[70] J. Xiong and J. Lam. Stabilization of linear systems over networks with bounded packet loss. *Automatica*, 43(1):80–87, 2007.

[71] Ying Zhang and Krishnendu Chakrabarty. Fault recovery based on checkpointing for hard real-time embedded systems. In *18th IEEE Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 320–327, 2003.

[72] W. Zhang, M. S. Branicky, and S. M. Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, 21(1):84–99, 2001.

[73] W. Zhang and L. Yu. Stabilization of sampled-data control systems with control inputs missing. *IEEE Transactions on Automatic Control*, 55(2):447–452, 2010.