Stability and Performance Analysis of Control Systems Subject to Bursts of Deadline Misses

Nils Vreman
Lund University, Department of Automatic Control, Sweden

Anton Cervin
Lund University, Department of Automatic Control, Sweden

Martina Maggio
Saarland University, Department of Computer Science, Germany
Lund University, Department of Automatic Control, Sweden

Abstract

Control systems are by design robust to various disturbances, ranging from noise to unmodelled dynamics. Recent work on the weakly hard model—applied to controllers—has shown that control tasks can also be inherently robust to deadline misses. However, existing exact analyses are limited to the stability of the closed-loop system. In this paper we show that stability is important but cannot be the only factor to determine whether the behaviour of a system is acceptable also under deadline misses. We focus on systems that experience bursts of deadline misses and on their recovery to normal operation. We apply the resulting comprehensive analysis (that includes both stability and performance) to a Furuta pendulum, comparing simulated data and data obtained with the real plant. We further evaluate our analysis using a benchmark set composed of 133 systems, which is considered representative of industrial control plants. Our results show the handling of the control signal is an extremely important factor in the performance degradation that the controller experiences—a clear indication that only a stability test does not give enough indication about the robustness to deadline misses.

2012 ACM Subject Classification Computer systems organization → Embedded and cyber-physical systems; Computer systems organization → Real-time systems; Computer systems organization → Dependable and fault-tolerant systems and networks

Keywords and phrases Fault-Tolerant Control Systems, Weakly Hard Task Model

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2021.15

Acknowledgements The authors are members of the ELLIIT Strategic Research Area at Lund University. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Number 871259 (ADMORPH project). This publication reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

1 Introduction

Feedback control systems have been used as prime examples of hard real-time systems ever since the term was coined. However, in the past twenty years, it has become increasingly clear that the hard real-time task model is overly strict for most control systems. Requiring that *all* deadlines of a periodic control task must be met can lead to very conservative designs with low utilisation, low sampling rates, and—in the end—worse than necessary control performance. Following this line of reasoning, researchers started looking into task models in which tasks can sporadically miss some deadlines, and defined concepts like the "skip factor" [45], i.e., the number of correctly executed jobs that must occur between two failed instances. Task models with failed jobs eventually led to the definition of the weakly hard task model [11], that specify constraints on the sequence of jobs that complete their

33rd Euromicro Conference on Real-Time Systems (ECRTS 2021).
Editor: Björn B. Brandenburg; Article No. 15; pp. 15:1–15:23
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

execution correctly and the ones that miss their deadlines. Adopting the weakly hard model allows a control task to opportunistically execute more frequently, which in general improves reference tracking and disturbance rejection [41, 46, 55].

A recent industrial survey has shown that practitioners are used to work with systems that experience deadline misses [5, Questions 14 and 15]. In a significant percentage of cases, these systems are subject to blackout events that can persist for more than ten consecutive task periods. Examples of such events are mode switches in mixed-criticality systems, resets due to hardware faults, security attacks, specific types of cache misses, and connectivity issues in networked control systems. Handling all of these situations by design could require extreme resource over-provisioning.

In this paper we focus precisely on these sporadic system events, which may cause a control task to stall for one or several cycles. To determine the effect of deadline misses on the control system, it is of utmost importance to analyse the physics of the plant and the effect of control signals not being delivered to it. For these systems, stability guarantees have been given on the maximum number of tolerable consecutive deadline misses [48]. These guarantees only consider *stability* of the closed-loop system as the property to be preserved. In this paper, we demonstrate that while stability may be preserved, the control system *performance* may be severely affected by the burst of misses. Performance and stability have been considered simultaneously in the literature. For example, in [30] a controller is developed that guarantees stability, accepting some level of performance degradation for a given plant. However, we believe that a lot is left open to investigate, especially with respect to general guarantees. In particular, in this paper we aim to understand the effect that the deadline handling strategies jointly have on performance and stability, providing a holistic evaluation. Furthermore, we evaluate our results on both simulated platforms and real control plants. More precisely, we offer the following contributions:

- We propose a new type of weakly hard task model, which specifies a *consecutive* deadline miss interval followed by a minimum *consecutive* deadline hit (recovery) interval. This model is crucial to properly assess the performance effect of a burst of deadline misses, as the ones reported by practitioners [5].
- We provide an analysis methodology for stability and performance of control tasks executing under this task model using a variety of implementation choices to handle deadline misses (Kill vs. Skip-Next, Zero vs. Hold). In particular, we separately consider the two cases in which a miss pattern is repeated (which fits an increased workload situation—for example due to a different mode of execution), and in which it is not possible to specify constraints on the repetition of the miss pattern.
- We compare experimental results obtained with a real process—a Furuta pendulum that is stabilised in the upright position—with simulation results based on a linear model of the same process, using the same controller. This shows that simulated data is representative enough to draw conclusions on the controller performance, despite unmodelled nonlinear dynamics and noise.
- We present the result of a large scale evaluation campaign of commonly used controllers on a benchmark of 133 industrial plants. From this evaluation we conclude that the choice of actuation strategy (i.e., what to do with the control signal when a miss occurs) affects control performance significantly more than the choice of deadline handling strategy (i.e., what to do with the control task when a miss occurs).

The rest of this paper is outlined as follows. In Section 2 we give a brief overview of related work. In Section 3 we present relevant control theory and introduce the stability and performance concepts. Section 4 describes the weakly hard task models and the strategies that are commonly used to handle deadline misses. Section 5 presents our extension to the weakly hard task model, and the corresponding stability and performance analysis. Section 6 presents our experimental results, and Section 7 concludes the paper.

2 Related Work

The work presented in this paper is closely related to two broad research areas, namely, the analysis of (i) weakly hard systems and (ii) fault-tolerant control systems.

Weakly Hard Systems: Deadline misses can be seen as sporadic events caused by unforeseen delays in the system. Such delays could for instance be induced by overload activations [36, 64] or cache misses [6, 22]. The idea behind weakly hard analysis is that deadline misses are permitted under predefined constraints. Such systems have been analysed extensively from a real-time scheduling perspective [10, 15, 21, 37]. The weakly hard models have gained traction in the research community as a tool to understand and analyse systems with sporadic faults [4, 12, 13, 26, 29, 35, 38, 55, 59–61]. In a recent paper, Gujarati et al. [33] analysed and compared different methods for estimating the overall reliability of control systems using the weakly hard task model. Furthermore, the authors of [50] proposed a toolchain for analysing the strongest, satisfied weakly hard constraints as a function of the worst-case execution time.

Fault-Tolerant Control Systems: Real-time systems are sensitive to faults. Due to their safety-critical nature, it is arguably more important to guarantee fault-tolerance with respect to other classes of systems. Some of these faults can be described using the weakly hard model. Due to the nature of control systems, special analysis techniques can combine fault models and the physical characteristics of systems.

Fault-tolerance has been investigated in many of its aspects, e.g., fault-aware scheduling algorithms [16, 23] and the analysis of systems with unreliable components [43]. Furthermore, restart-based design [1, 2] has been used as a technique to guarantee resilience. The fault models are frequently assumed to target overload-prone systems, or systems with components subject to sporadic failures. Bursts of faults have been observed to affect real systems [20, 63]. Gujarati et al. [32] proposed an analysis method for networked control systems that uses active replication and quantifies the resilience of the control system to stochastic errors. Maggio et al. [48] developed a tool for determining the stability of a control system where the control task behaves according to the weakly hard model. From the control perspective, there has been extensive research into both analysis and mitigation of real-time faults in feedback systems [30, 31, 57]. Very often, this research produced tools to analyse the effect of computational delays [19] and of choosing specific scheduling policies or parameters [18, 52], possibly including deadline misses. In a few instances, researchers looked at how to improve the performance of control systems in conjunction with scheduling information [14]. One such effort analyses modifications to the code of classic and simple control systems to handle overruns that reset the period of execution of the control task [53]. Abdi et al. [3] proposed a control design method for safe system-level restart, mitigating unknown faults during runtime execution, while keeping the system inside a safe operating space. Pazzaglia et al. [54] used the scenario theory to derive a control design method accounting for potential deadline misses, and discussed the effect of different deadline handling strategies. Linsenmayer et al. [47] worked on the stabilisation of weakly-hard linear control systems for networked control systems, with some extension for nonlinear systems [39]. In the considered setup, faults compromise network transmissions, but do not interfere with the controller computation

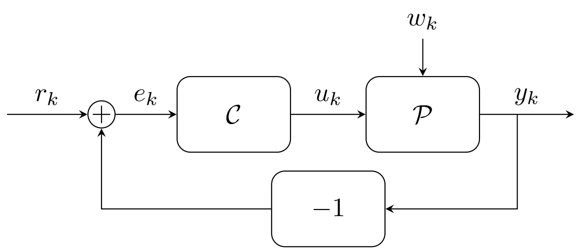

Figure 1 Control loop: The reference value r_k is compared with the output y_k of the plant $\mathcal{P}$. The control error $e_k = r_k - y_k$ is used by the controller $\mathcal{C}$ to compute the value of the control signal u_k. The plant is disturbed by the stochastic process w_k.

(assuming that the computation is triggered). The work also focused on stability, with no control performance evaluation.

To the best of our knowledge, no previous work has devised a combined stability and performance analysis to understand how faults (even when they can be tolerated) affect the plant that should be controlled when different deadline handling strategies are used.

3 System Behaviour in Nominal Conditions

In this section, we introduce the relevant control background needed for the remainder of the paper, and we detail how the controller and the system behave under normal operation.

3.1 Plant Model

We first describe the model we use for the object we are trying to control. In control terms—mostly due to historical reasons—this object is called a *plant*. Examples range from a pendulum that we would like to stabilise in the upward position, to a chemical dilution process, to the distribution of workload in a datacenter.

Plants are usually modelled as continuous- or discrete-time dynamical systems. All real-world plants are nonlinear, but for control design purposes they are often linearised around their operating points. Around such a point, the resulting model becomes a Linear Time-Invariant (LTI) system. In this paper, we restrict our analysis to discrete-time LTI systems, because we investigate controllers implemented with fixed-rate sampling and actuation in digital electronics. To design and analyse these systems, we use the discrete-time counterpart of the continuous-time physical model, which can be obtained with standard techniques [9].

We consider a plant $\mathcal{P}$ described in state-space form:

$$\mathcal{P} : \begin{cases} x_{k+1} = A_p\, x_k + B_p\, u_k + G_p\, w_k \\ y_k = C_p\, x_k + D_p\, u_k \end{cases} \tag{1}$$

In (1), k counts the discrete instants that represent the plant's sampling points. We assume periodic sampling; the time between two consecutive samples k and $k+1$ is fixed and equal to sampling period t_s. In the equation, x_k is a column vector with d_p elements. These elements represent the state variables that account for, e.g., the storage of mass, momentum, and energy. Similarly, u_k is a column vector with i_p elements. These values represent the inputs that affect the dynamics of the plant. We also consider w_k, a column vector with i_p elements. The term w_k represents an unknown load disturbance, modelled as a stationary stochastic process with known properties. Finally, y_k is a column vector with o_p

elements, that represents the measurements that are taken from our plant. The matrices A_p (size $d_p \times d_p$), B_p (size $d_p \times i_p$), C_p (size $o_p \times d_p$), D_p (size $o_p \times i_p$), and G_p (size $d_p \times i_p$) characterise the dynamics of the plant.

3.2 Controller Model

The plant $\mathcal{P}$ is controlled by a periodically executing controller $\mathcal{C}$ with implicit deadlines, i.e., the deadline of each task instance (job) coincides with the next task activation. We consider the class of all linear controllers with a one-step delay between sampling and actuation.[1] In other words, we consider all the controllers that can be written as linear systems, according to the following state-space equation:

$$\mathcal{C} : \begin{cases} z_{k+1} = A_c\, z_k + B_c\, e_k \\ u_{k+1} = C_c\, z_k + D_c\, e_k \end{cases} \tag{2}$$

Here, z_k is a column vector with d_c elements that represents the state of the controller. The input of the controller is e_k, a vector of $i_c = o_p$ elements. Each element in the vector is the error between the corresponding plant output and its reference value ($e_k = r_k - y_k$, where r_k represents the reference values for the plant outputs). Finally, u_k is a vector of $o_c = i_p$ elements, that encodes the output of the controller, which is connected to the plant input vector. The matrices A_c (size $d_c \times d_c$), B_c (size $d_c \times i_c$), C_c (size $o_c \times d_c$), and D_c (size $o_c \times i_c$) characterise the dynamics of the controller. For every task activation, the controller first applies the value of u_k that was computed by the previous job and then reads the inputs r_k and y_k. It then calculates the values of z_{k+1} and u_{k+1} that will be used in the next iteration.

The analysis methodology presented in the remainder of this paper is valid for *all* linear controllers. The class of linear controllers includes some of the most frequently used controllers in industry, in particular proportional and integral (PI), proportional, integral, and derivative (PID), lead–lag compensators, and linear-quadratic-Gaussian (LQG) controllers. Although the performance analysis is presented for the time-invariant case, the formulas are valid also for systems with time-varying matrices. Hence, it is possible to analyse plants and controllers that transition between different local linear models.

3.3 Closed-Loop System Dynamics

We now analyse the closed-loop system shown in Figure 1. Combining the dynamical models from (1) and (2), we obtain matrices that represent the closed-loop system. We denote the state vector of the closed-loop system with $\tilde{x}_k = \left[x_k^T,\, z_k^T,\, u_k^T\right]^T$, where ^{T} is the transpose operator. In this way, we obtain a system that has the vectors r_k and w_k as input, and is described by

$$\mathcal{S} : \begin{cases} \tilde{x}_{k+1} = A\,\tilde{x}_k + B_r\, r_k + B_w\, w_k \\ \quad y_k = C\,\tilde{x}_k, \end{cases} \tag{3}$$

[1] One-step delay controllers are controllers in which a control signal is computed in the k-th interval and actuated at the beginning of the $k+1$-th period. In the real-time systems jargon, one-step delay controllers are often referred to as controllers that follow the Logical Execution Time (LET) paradigm [25, 44]. From the real-time perspective, implementing the controller following the LET paradigm improves the timing predictability. From the control perspective, one-step delay controllers reduce activation jitter and allows the engineer to neglect time-varying computational delays.

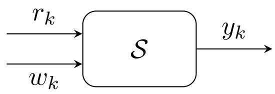

Figure 2 Closed-loop system rewritten as a new linear system $\mathcal{S}$. The resulting system has two inputs, r_k and w_k and one output. The feedback loop shown in Figure 1 is hidden inside $\mathcal{S}$.

where the closed-loop state matrix A is

$$A = \begin{bmatrix} A_p & 0_{d_p \times d_c} & B_p \\ -B_c C_p & A_c & -B_c D_p \\ -D_c C_p & C_c & -D_c D_p \end{bmatrix}, \tag{4}$$

the input matrices B_r and B_w are

$$B_r = \begin{bmatrix} 0_{d_p \times i_c} \\ B_c \\ D_c \end{bmatrix}, \quad B_w = \begin{bmatrix} G_p \\ 0_{d_c \times i_p} \\ 0_{i_p \times i_p} \end{bmatrix}, \tag{5}$$

and the output matrix C is

$$C = \begin{bmatrix} C_p & 0_{d_p \times d_c} & D_p \end{bmatrix}. \tag{6}$$

Figure 2 shows the graphical representation of the closed-loop system $\mathcal{S}$, with input and output signals.

Stability

To assess the stability of the closed-loop system under normal operation, it is sufficient to check the eigenvalues of the state matrix. According to the Schur stability criterion [9], if the eigenvalues of A lie within the unit disc, then the system is asymptotically stable. Formally, a closed-loop system is Schur stable if and only if

$$\max_i |\lambda_i(A)| < 1, \tag{7}$$

where $\lambda_i(A)$ is a function that returns the i-th eigenvalue of A.

If the system dynamics change at runtime (e.g., in the case of a lost sample, unexpected delay, or computational problem), Schur stability is no longer a sufficient stability criterion. Instead, *switching stability analysis* can be employed to check the stability of a system with alternating dynamics [40]. There has been a lot of research on the switching stability analysis, with multiple tools developed in order to simplify the analysis. Two main methods are employed: (i) the search for a common Lyapunov function, e.g., as done in [46], (ii) the computation of the Joint Spectral Radius (JSR), e.g., as done in [48, 62].

Performance

Alongside stability, it is important to look at the *performance* of the closed-loop system. Performance can be defined in different ways, often depending on the application [8]. Whichever way is chosen, a common way to quantify performance is to define a cost function and evaluate the cost function during the execution of the controller. In our work, we use a quadratic cost function

$$J_k = \mathbb{E}\left(e_k^T Q_e e_k + u_k^T Q_u u_k\right). \tag{8}$$

The cost function penalises deviations from the reference value as well as usage of the control signal. $\mathbb{E}$ denotes expected value, and the positive semidefinite weighting matrices Q_e (size $o_p \times o_p$) and Q_u (size $o_c \times o_c$) weigh the different terms against each other. A small cost value means that the controller successfully makes the error approach zero, using a small control signal.

If the stochastic properties of the external signals r_k and w_k are known, it is possible to calculate the value of the cost function analytically. For simplicity and without loss of generality, we will henceforth assume that $r_k = 0$ (i.e., we want to regulate the output to zero) and that w_k is a zero-mean Gaussian white noise process with variance $R = \mathbb{E}\left(w_k w_k^T\right)$. More elaborate disturbance models can be realised by adding extra states in the plant model.

We now detail how to evaluate (8). Let P_k denote the covariance of the closed-loop state vector at time k,

$$P_k = \mathbb{E}\left(\tilde{x}_k \tilde{x}_k^T\right). \tag{9}$$

The state covariance evolves according to

$$P_{k+1} = A\, P_k\, A^T + B_w\, R\, B_w^T. \tag{10}$$

Given P_k, we can calculate the cost for time step k as

$$J_k = \mathbb{E}\left(\tilde{x}_k^T\, Q\, \tilde{x}_k\right) = \operatorname{tr}\left(P_k\, Q\right), \tag{11}$$

where tr computes the trace of the matrix, and

$$Q = \begin{bmatrix} C_p^T\, Q_e\, C_p & 0_{d_p \times d_c} & 0_{d_p \times i_p} \\ 0_{d_c \times d_p} & 0_{d_c \times d_c} & 0_{d_c \times i_p} \\ 0_{i_p \times d_p} & 0_{i_p \times d_c} & Q_u \end{bmatrix} \tag{12}$$

is the total cost matrix. The stationary cost of the system is defined as J_∞. This is the cost that the system converges to when operating under normal conditions:

$$J_\infty = \lim_{k \to \infty} J_k. \tag{13}$$

This means that there exists an instant $\bar{k}$ for which J_k reaches a value arbitrarily close to the steady-state value J_∞, or $\forall \varepsilon,\ \exists \bar{k}$ s.t. $\forall k > \bar{k},\ |(J_k - J_\infty)/J_\infty| < \varepsilon$.

4 System Behaviour with Deadline Misses

The analysis above holds when the control task meets all its deadlines. However, the presence of deadline misses changes the behaviour of the system. The stability of controllers with a number of consecutive deadline misses has been investigated in [48]. The results of this investigation attested that, due to their inherent robustness, many control systems can withstand at least a small number of consecutive misses.

To analyse the system, we need to clarify three aspects about the miss behaviour:

(i) What happens to the control signal.
(ii) What happens to the control task.
(iii) The computational model used for the analysis (how many deadlines can we miss, and in what pattern).

For the first item, the actuator can either output a *zero* ($u_k = 0_{o_c \times 1}$), or *hold* the previous value ($u_k = u_{k-1}$). The choice depends on both the plant dynamics and on the controller, as no strategy in general dominates the other one [58]. For controllers with integral action, it makes sense to hold the previous control value, under the presumption that the system is still disturbed and that a non-zero control signal is needed to keep the plant close to its operating point. On the other hand, the zero strategy may be preferred for plants with unstable or integrator dynamics, where outputting a zero control action may be the safer option.

Considering the second item, at least three different strategies can be employed to deal with a control task that misses its deadline [18]: (i) *Kill*, (ii) *Skip-Next*, (iii) and *Queue*(λ) ($\lambda \in \{1, 2, 3, \dots\}$). When the Kill strategy is used, the job that missed its deadline is terminated, its changes are rolled back, and the next job is released. Following the Skip-Next strategy, the job that missed its deadline continues its execution. No new control task jobs are released until the currently running one completes its execution. Queue(λ) behaves similarly to Skip-Next in allowing the current job to complete execution, but also allows the activation of new jobs (the queue of active jobs holds up to the most recent λ instances of the control task). In this paper we only analyse Kill and Skip-Next. In fact, the results presented in [18, 48] suggest that Queue(λ) is not a feasible strategy to handle misses. The presence of two or more active jobs in the same period creates a chain effect that is hard to recover from and that deteriorates stability and performance.

The last item refers to models of computation. The weakly hard task model [11, 34] is usually considered expressive enough to analyse the behaviour of tasks that miss their deadlines. The authors of [11] propose four definitions for a weakly hard real-time task τ:

▶ **Definition 1** (Weakly Hard Task Models [11]). *A task τ may satisfy any of these four weakly hard constraints:*

(i) $\tau \vdash \binom{n}{\ell}$*: there are at least n hits for every ℓ jobs,*
(ii) $\tau \vdash \binom{\overline{m}}{\ell}$*: there are at most m misses for every ℓ jobs,*
(iii) $\tau \vdash \left\langle {n \atop \ell} \right\rangle$*: there are at least n consecutive hits for every ℓ jobs,*
(iv) $\tau \vdash \left\langle {\overline{m} \atop \ell} \right\rangle$*: there are at most m consecutive misses for every ℓ jobs.*

There has been a lot of research on the second model, often also called m-K model [4, 12, 13, 26, 29, 35, 38, 45, 54, 55, 57, 59–61] (with m being the maximum number of misses in a window of K activations). Recently there has also been an analysis of the stability of control systems when the control task behaves according to the fourth model [48].

If the misses are due to faults or security attacks, usually the control task experiences an interval of consecutive misses. When the fault is resolved, the control task starts hitting its deadlines again. From the performance standpoint, a consecutive number of misses degrades the control quality. We are interested in what degradation is acceptable and how much time should occur between two potential failures. Specifically, we look at how many deadline hits should follow a given number of consecutive misses for the system to *recover*. None of the four models above allow us to formulate this requirement (as they specify either consecutive hits or misses but not both), which leads us to introduce a different weakly hard model of computation, together with its analysis, in Section 5.

5 Burst Interval Analysis

In this section, we analyse the stability and performance of a real-time control system that experiences bursts of deadline misses. Section 5.1 introduces the fault model, Section 5.2

derives the control system behaviour subject to different real-time policies and delves into both the stability and performance analysis.

5.1 Fault Model

Faults can happen during the normal execution of tasks on a platform. Informally, as a result of a fault, tasks miss their deadlines. When the fault is resolved, then the original situation is recovered (possibly after a transient initial phase).

Specifically, given a system $\mathcal{S}$, we define a *burst interval* $\mathcal{M}$ as an interval of controller activations in which the control task executing $\mathcal{C}$ consecutively misses m deadlines, regardless of the strategy used to handle the misses. We assume that the burst interval $\mathcal{M}$ is followed by a *recovery interval* $\mathcal{R}$, defined as an interval in which the control task consecutively hits n deadlines.

During the burst interval, the deadline misses of the control task are handled using a *deadline handling strategy* $\mathcal{D}$ (Kill, K, or Skip-Next, S). The control signal u_k is selected in accordance with the *actuation strategy* $\mathcal{A}$ (Zero, Z, or Hold, H). We denote the combination of $\mathcal{D}$ and $\mathcal{A}$ with $\mathcal{H} = (\mathcal{D}, \mathcal{A})$. For example $\mathcal{H}$ could be SZ to indicate that the Skip-Next deadline handling strategy is paired with the Zero actuation strategy. The system *recovers* once it operates close to steady-state.

From an industrial viewpoint, the proposed fault model is highly relevant. The common approach is to treat faults as pseudo-independent events adhering to predefined constraints on their incidence rate [42, 49, 51]. However, during the operation of a control system, faults can be caused by events like network connection problems (e.g., cutting the connection between the sensor and the controller), security attacks, contention on resources. Studies in the automotive sector, for example, indicate that deadline misses can occur in bursts [56, 64]. In these cases, the controller does not execute properly for a given amount of time (e.g., until the connection is restored, the attack is terminated, or the resource contention is reduced). The analysis methods we propose allow us to address such situations and to provide tighter bounds on the closed-loop stability and performance than under the previously proposed weakly hard models. Moreover, following a burst interval, we are interested in analysing the length of the recovery interval $\mathcal{R}$ that is needed to return to normal operation under each implementation strategy $\mathcal{H}$. Hence, we here extend the weakly hard models of computation with a fifth alternative and then devote the remainder of the paper to its analysis.

▶ **Definition 2** (Weakly Hard Fault Model With Burst Of Misses). *A real-time task τ may satisfy the weakly hard task model*

(v) $\tau \vdash \left\{ {m \atop \ell} \right\}$: there are at most m consecutive misses, followed by $\ell - m$ consecutive hits for every ℓ jobs.

This means that a real-time task τ behaves according to the model $\tau \vdash \left\{ {m \atop \ell} \right\}$, if, whenever τ experiences a burst interval $\mathcal{M}$ consisting of m consecutive deadline misses, it is always followed by a recovery interval $\mathcal{R}$ consisting of $n = \ell - m$ consecutive deadline hits.

5.2 Closed-Loop System Dynamics

In this section we derive the system dynamics for a closed-loop control system under the assumption that we enter a burst interval of length m after time instant k, and after m deadline misses we start completing the control job in time.

Normal Operation: Under *normal operating conditions* the system is not experiencing any deadline misses. In other words, the system evolves according to the closed-loop system dynamics (3).

Kill&Zero: If a control task deadline miss occurs at time instant k, the plant states x_k still evolve as normal. However, the controller terminates its execution prematurely by killing the job, thus not updating its states ($z_{k+1} = z_k$). The controller output is determined by the actuation strategy and is here zero ($u_{k+1} = 0$). Now, consider a burst interval of length m after time instant k. Recalling that $\tilde{x}_k = \left[{x_k}^T {z_k}^T {u_k}^T\right]^T$, we can write the evolution of the closed-loop system for the sequence of m deadline misses followed by a single deadline hit as the product of a matrix representing the behaviour of the system for a hit and a matrix representing the behaviour in case of miss elevated to the power of m to indicate m steps of the system evolution.

The resulting closed-loop system in state-space form is

$$\begin{bmatrix} x_{k+m+1} \\ z_{k+m+1} \\ u_{k+m+1} \end{bmatrix} = \underbrace{A \begin{bmatrix} A_p & 0_{d_p \times d_c} & B_p \\ 0_{d_c \times d_p} & I & 0_{d_c \times i_p} \\ 0_{i_p \times d_p} & 0_{i_p \times d_c} & 0_{i_p \times i_p} \end{bmatrix}^m}_{A_{KZ}(m)} \begin{bmatrix} x_k \\ z_k \\ u_k \end{bmatrix}, \tag{14}$$

where $A_{KZ}(m)$ represents the system matrix for m misses under the Kill&Zero strategy, followed by a single hit (the matrix A that is multiplied to the left of the equation). The matrix A is the same specified in (4), and represents the first hit that follows the m misses, hence, we determine how $\tilde{x}_k$ influences $\tilde{x}_{k+m+1}$ (m misses and one hit).

Kill&Hold: Changing the actuation strategy to Hold, slightly alters the system matrix we derived for the Kill&Zero case. The plant states x_k evolve as normal and the control states z_k are still not updated ($z_{k+1} = z_k$). However, due to the change in actuation strategy, the last actuated value is instead held ($u_{k+1} = u_k$). The resulting closed-loop state-space form can be seen in (15), where $A_{KH}(m)$ is used to represent the system matrix for m misses under the Kill&Hold strategy and matrix A is specified in (4).

$$\begin{bmatrix} x_{k+m+1} \\ z_{k+m+1} \\ u_{k+m+1} \end{bmatrix} = \underbrace{A \begin{bmatrix} A_p & 0_{d_p \times d_c} & B_p \\ 0_{d_c \times d_p} & I & 0_{d_c \times i_p} \\ 0_{i_p \times d_p} & 0_{i_p \times d_c} & I \end{bmatrix}^m}_{A_{KH}(m)} \begin{bmatrix} x_k \\ z_k \\ u_k \end{bmatrix} \tag{15}$$

Skip-Next&Zero: When the control task misses a deadline under the Skip-Next strategy, the job missing the deadline is allowed to continue its execution until completion. However, no subsequent job of the control task is released until the current job has finished executing. If the currently active job terminates during period k, the next control job is released at the start of the $k+1$-th period. We can then write the evolution of the system where the control job experiences m misses before completing its execution, meaning that there is a subsequent hit that uses old information for the error measurements. While the controller executed only once to completion, the plant evolved for $m+1$ steps. The resulting closed-loop state-space form can be seen in (16), where $A_{SZ}(m)$ is used to represent the system matrix under the Skip-Next&Zero strategy for m misses and one completion using old measurements.

$$\begin{bmatrix} x_{k+m+1} \\ z_{k+m+1} \\ u_{k+m+1} \end{bmatrix} = \underbrace{\begin{bmatrix} A_p^{m+1} & 0_{d_p \times d_c} & A_p^m B_p \\ -B_c C_p & A_c & -B_c D_p \\ -D_c C_p & C_c & -D_c D_p \end{bmatrix}}_{A_{SZ}(m)} \begin{bmatrix} x_k \\ z_k \\ u_k \end{bmatrix} \tag{16}$$

Skip-Next&Hold: Similar to Skip-Next&Zero, one job finishes execution after m consecutive misses. However, the actuation strategy holds the previous control value during the entire burst interval. Therefore, the plant evolution is affected by a cumulative sum over the prior control values. The resulting closed-loop state-space form can be seen in (17), where $A_{SH}(m)$ is used to represent the system matrix for m misses under the Skip-Next&Hold strategy.

$$\begin{bmatrix} x_{k+m+1} \\ z_{k+m+1} \\ u_{k+m+1} \end{bmatrix} = \underbrace{\begin{bmatrix} A_p^{m+1} & 0_{d_p \times d_c} & \sum_{i=0}^{m} A_p^i B_p \\ -B_c C_p & A_c & -B_c D_p \\ -D_c C_p & C_c & -D_c D_p \end{bmatrix}}_{A_{SH}(m)} \begin{bmatrix} x_k \\ z_k \\ u_k \end{bmatrix} \tag{17}$$

Equations (14)–(17) are inspired by the analysis in [48], but we have we introduced two generalisations. The first one is that our controller is specified as a general state-space system; therefore our method is able to address *all* linear controllers. The second generalisation is that we could include estimates of the plant states in the controller. We can thus properly handle the presence of an observer.[2] Furthermore, we simplify the calculations by reducing the number of states $\tilde{x}_k$ of the closed-loop matrices.

Stability

We now describe how the system matrices above can be used to analyse stability. Recall that a closed-loop control system is stable if and only if the (fixed) system matrix A is Schur stable. This criterion is also valid for cyclic patterns, where A represents the product of all closed-loop state matrices experienced in a full burst–recovery cycle. Hence, we can search for the shortest recovery interval length n such that

$$\max_i \left|\lambda_i \left(A^{n-1} A_{\mathcal{H}}(m)\right)\right| < 1, \quad \mathcal{H} \in \{KZ, KH, SZ, SH\}. \tag{18}$$

Recall that $A_{\mathcal{H}}(m)$ already includes one hit, thus the left multiplication with A^{n-1}. This is a sufficient condition and not necessary, meaning that a miss occurring during the recovery interval does not immediately imply that the closed-loop system is destabilised. We summarise the analysis in the following definition.

▶ **Definition 3** (Static-Cyclic Stability Analysis). *We denote the stability analysis from* (18) *with the term* static-cyclic stability analysis*. The system under analysis cycles through a sequence of m misses followed by a sequence of n hits, indefinitely.*

The static-cyclic analysis assumes a repeating burst–recovery cycle with no interruptions. This works well for instance in case the misses are due to a permanent overload condition caused by a mode switch (for example from low to high criticality mode in mixed-critical systems). However, the setting is not very general. To foster generality, we complement the stability evaluation with a less restrictive stability analysis, based on the proposed task model in Definition 2.

▶ **Definition 4** (Miss-Constrained Stability Analysis). *To guarantee* miss-constrained stability*, a system has to be stable under arbitrary switching between all the possible m realisations*

[2] In [48] the controller state is specified as part of the plant (e.g., when the proportional and integral controller is introduced). This implies that the state is computed although the controller did not execute. Our formulation fixes this by separating the plant execution and the controller states.

(i.e., closed-loop matrices) that comply with all task models $\tau \vdash \left\{ {m_{\subset} \atop \ell} \right\}, m_{\subset} \in \{1, \ldots, m\}$ and also include the case in which the system does not miss deadlines.

In other words, a system is miss-constrained stable if and only if it is stable under arbitrary switching of the closed-loop matrices in the set

$$\left\{ A^{\ell-1} A_{\mathcal{H}}\,(1)\,,\, A^{\ell-2} A_{\mathcal{H}}\,(2)\,,\, \ldots,\, A^{\ell-m} A_{\mathcal{H}}\,(m)\,,\, A \right\}. \tag{19}$$

Switching stability is unfortunately quite involved.[3] However, many excellent tools have been developed to simplify this analysis (e.g., `MJSR` [48] or the `JSR toolbox` [62] for MATLAB).

Performance

We now show how the cost function in Equation (11) can be used as a time-varying performance metric. Before a burst interval, we assume that the system is in the neighbourhood of its steady-state covariance P_∞ and performance J_∞.

When a burst interval of m missed deadlines occurs, the system will be disrupted and its covariance matrix will evolve according to

$$P_{k+m+1} = A_{\mathcal{H}}\,(m)\, P_k \left(A_{\mathcal{H}}\,(m)\right)^T + A^{j_n} R_w \left(A^{j_n}\right)^T, \tag{20}$$

where

$$\begin{aligned} R_w &= \begin{bmatrix} \sum_{i=0}^{j_m} A_p^i\, G_p\, R\, G_p^T\, (A_p^i)^T & 0_{d_p \times d_c + i_p} \\ 0_{d_c + i_p \times d_p} & 0_{d_c + i_p \times d_c + i_p} \end{bmatrix}, \\ j_m &= \begin{cases} m-1 & \text{if } \mathcal{D} = K \text{ (Kill)}, \\ m & \text{if } \mathcal{D} = S \text{ (Skip-Next)}, \end{cases} \\ j_n &= \begin{cases} 1 & \text{if } \mathcal{D} = K \text{ (Kill)}, \\ 0 & \text{if } \mathcal{D} = S \text{ (Skip-Next)}. \end{cases} \end{aligned} \tag{21}$$

A_p and G_p are matrices from the plant evolution in (1), R is the noise intensity from (10), and A is the closed-loop matrix from (4). The cost will simultaneously change following (11). In the recovery interval, the covariance is again governed by the normal closed-loop evolution described in (10). The system is said to have recovered once the cost is arbitrarily close to the steady-state cost. We evaluate this as

$$\left| \frac{J_\infty - J_k}{J_\infty} \right| < \varepsilon, \tag{22}$$

where $\varepsilon > 0$ is the *recovery threshold.*

▶ **Definition 5** (Performance Recovery Interval). *We define the recovery length interval $n^*_{\mathcal{H}}$ as the smallest n such that* (22) *is satisfied for all $k \geq n$ when using $\mathcal{H}$ to handle deadline misses.*

▶ **Definition 6** (Maximum Normalised Cost). *We denote the maximum normalised cost of the system by*

$$J_{M,\mathcal{H}} = \max_k \frac{J_{k,\mathcal{H}}}{J_\infty}, \tag{23}$$

[3] We have devoted some research effort into the investigation of a suitable stability analysis for control tasks subject to a set of weakly-hard constraints (of the type presented in Defintion 1). A summary of our findings can be found at `https://arxiv.org/abs/2101.11312`.

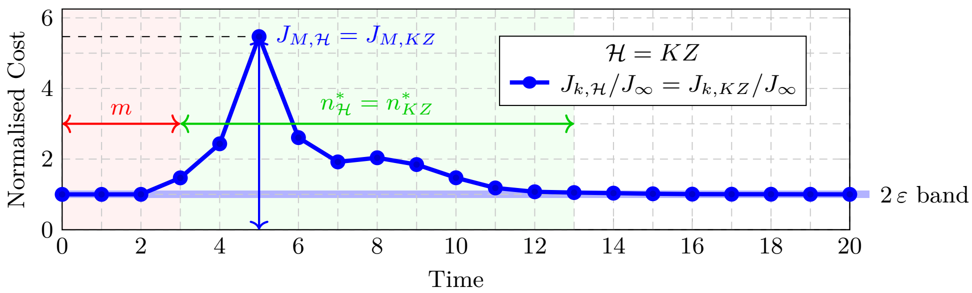

Figure 3 Illustration of normalised cost (J_k/J_∞), performance recovery interval $n^*_{\mathcal{H}}$ and maximum normalised cost $J_{M,\mathcal{H}}$ on a data trace. The example uses $\mathcal{H}$ = Kill&Zero and $\varepsilon = 0.1$.

where $J_{k,\mathcal{H}}$ is the cost computed according to (11) *when using $\mathcal{H}$ to handle the deadline misses.*

Figure 3 gives a graphical representation of $n^*_{\mathcal{H}}$ and $J_{M,\mathcal{H}}$ for an execution trace in which the controller experiences 3 misses and uses Kill&Zero as strategy $\mathcal{H}$.

Compared to the stability analysis, the performance analysis also takes into account state deviations and uncertainty due to disturbances. In Section 5.2 we used the system dynamics to analyse the stability of the system. The disturbance term w_k was neglected as it does not influence the system stability. However, its presence (as the presence of any disturbance) changes the dynamic behaviour of the system. For the performance metric, the state covariance matrix P_k evolves according to both the noise intensity and the system dynamics (20). The result is that the performance analysis provides us with a conservative (but more realistic) recovery interval, that takes system uncertainties into consideration.

To find the length of the recovery interval, we evolve the state covariance during a burst interval, using a specific strategy $\mathcal{H}$ according to (20). Thereafter, the state covariance is evolved under normal operation, according to (10), until (22) is satisfied, allowing us to find the performance recovery interval $n^*_{\mathcal{H}}$.

6 Experimental Results

In this section, we apply the analysis presented in Section 5 to a set of case studies, analysing stability and performance. We first present detailed results with a Furuta pendulum, both in simulation and with real hardware, using the same controller. The simulated results are compared to the real physical plant. This shows that the performance analysis does capture the important trends for real control systems. We then present some aggregate results obtained with a set of 133 different plants from a control benchmark. One noteworthy aspect is that the Furuta pendulum model is linearised for the control design and the pendulum stabilised around an unstable equilibrium—the top position—while the control benchmark includes (by design) stable systems. The difference between simulation results and real experiments for stable linear systems should in principle be smaller than for unstable nonlinear systems, making our pendulum the ideal stress test for the similarity of simulated and real data.

6.1 Furuta Pendulum

We here analyse the behaviour of a Furuta pendulum [27], a rotational inverted pendulum in which a rotating arm is connected to a pendulum. The rotation of the arm induces a swing

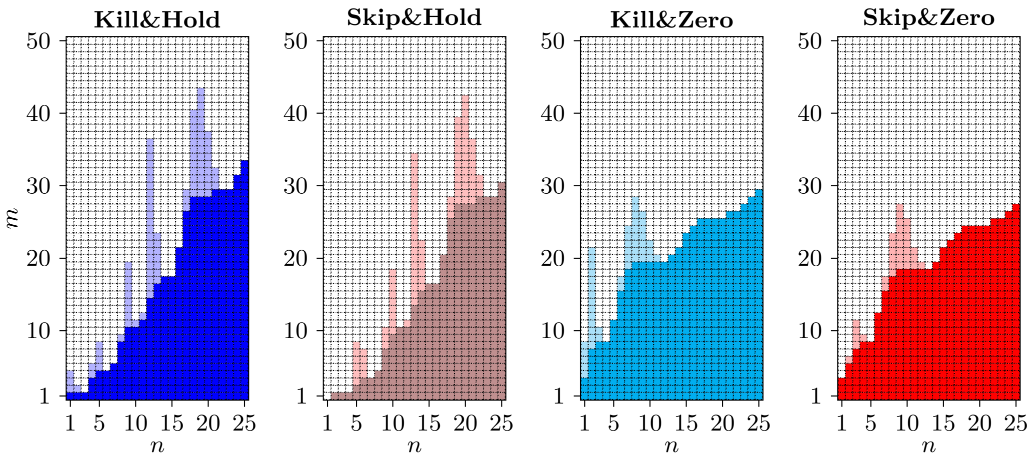

Figure 4 Miss-constrained stability (dark coloured area) and static-cyclic stability (light coloured area) when different strategies $\mathcal{H}$ are used in the example and the weakly hard model in Definition 2 is considered. Each square represents a window of size $\ell = m + n$. The dark area satisfies both the miss-constrained and static-cyclic stability whilst the light area only provides static-cyclic stability. The white squares denote potentially unstable combinations of m and n.

movement on the pendulum. The pendulum has two equilibria: a stable position in which the pendulum is downright, and an unstable position in which the pendulum is upright. Our objective is to keep the pendulum in the up position, by moving the rotating arm.

The Furuta pendulum is a highly nonlinear process. In order to design a control strategy to keep the pendulum in the top position, it is necessary to linearise the dynamics of the system around the desired equilibrium point. We consider this as a stress test to check the divergence between simulation results and real hardware results, because of the instability of the equilibrium and the nonlinearity of the dynamics. In fact, the controller necessarily acts with information that is valid only around the upright position, and there is only a range of states in which the linearised model closely describes the behaviour of the physical plant.

We design a linear-quadratic regulator (LQR) to control the plant. Every $t_s = 10\,\text{ms}$ the plant is sampled and the control signal is actuated. Based on state-of-the-art models [17] and on our control design, the plant model $\mathcal{P}$ is

$$\mathcal{P} \;:\; \begin{cases} x_{k+1} = \begin{bmatrix} 1.002 & 0.0100 & 0 & 0 \\ 0.3133 & 1.002 & 0 & 0 \\ -2.943 \cdot 10^{-5} & -9.808 \cdot 10^{-8} & 1 & 0.01 \\ -0.0059 & -2.943 \cdot 10^{-5} & 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} -0.0036 \\ -0.7127 \\ 0.0096 \\ 1.9120 \end{bmatrix} u_k + I w_k, \\ y_k = I x_k, \end{cases} \tag{24}$$

the controller $\mathcal{C}$ takes the form

$$\mathcal{C} \;:\; u_{k+1} = \begin{bmatrix} 8.8349 & 1.5804 & 0.2205 & 0.3049 \end{bmatrix} x_k \tag{25}$$

and is designed and analysed using the following parameters (see Section 3.3):

$$Q_e = \mathrm{diag}(100, 1, 10, 10), \quad Q_u = 100, \quad R = \mathrm{diag}(0, 0, 10, 1). \tag{26}$$

We first apply the stability analyses presented in Section 5.2 to our model. Figure 4 shows the results. Each square in the figure represents a combination of (at most) m deadline

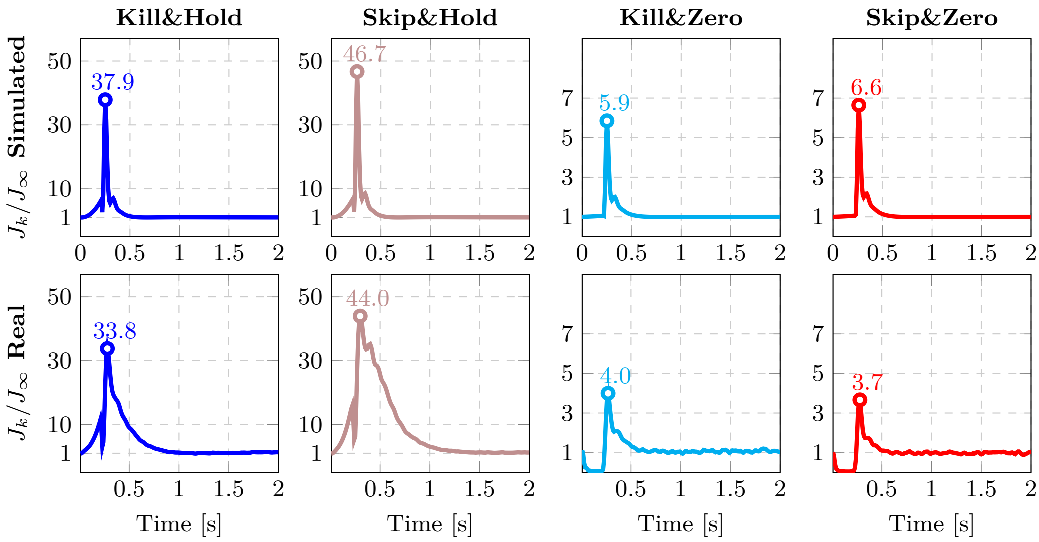

Figure 5 Normalised performance cost J_k/J_∞ obtained with the Furuta pendulum. The upper part of the figure shows simulated data, while the lower part of the figure shows the corresponding values obtained averaging the results of 500 experiments with the real process and hardware. Each experiment corresponds to a 500 jobs of the controller (20 misses and 480 hits).

misses (on the vertical axis) and (at least) n deadline hits (on the horizontal axis). If a square is coloured with a dark colour, the corresponding combination of misses and hits is both static-cyclic and miss-constrained stable, found using the `JSR Toolbox` [62]. The light squares in the figure show combinations for which the system only satisfies the static-cyclic stability condition. The white squares mark configurations for which stability cannot be guaranteed.

We remark on the presence of peaks in the static-cyclic stability region of $\mathcal{H} = KH$ at $n = \{1, 5, 9, 13, 19\}$. Similar peaks are also found for the other strategies, but for different values of n. These peaks indicate that the system would be stable if that particular burst and recovery interval length would be repeated indefinitely. However, this assumption is not robust to variations in the burst or recovery interval lengths as can be seen from the miss-constrained stability region being more conservative with its guarantees. Instead, the peaks in the static-cyclic region can be explained by stable modes occurring due to the natural frequencies of the open-loop (for the Zero actuation mode) and closed-loop (for the Hold actuation mode) systems. It is also interesting to note that Kill seems to consistently yield a larger stability region than Skip-Next, while neither Zero nor Hold dominate each other in terms of stability guarantees. An example of the latter fact was given already in [58].

For the performance analysis, we considered a one-shot burst fault of a specific length m, followed by a long period of normal execution. Assuming that the pendulum starts close to the upright equilibrium, with stationary cost J_∞, we calculate how the covariance P_k and performance cost J_k evolve during and after the burst interval using Equations (20)–(21).[4] These calculations assume an ideal, linear model of the pendulum. The simulation results for different strategies and bursts of length $m = 20$ are shown in the upper half of Figure 5. For Hold, it is seen that the cost grows exponentially during the initial fault interval (the first

[4] The analysis is implemented using `JitterTime` [19], `https://www.control.lth.se/jittertime`.

$20\,t_s = 0.2\,\mathrm{s}$). This is true also for Zero, although the growth rate is too small to be visible. The reason for the poor performance of Hold is that any non-zero held control signal will actively push the pendulum away from its unstable upright equilibrium even further than either disturbances or noise would already do without a proper control action.

The large spike in cost comes when the controller is reactivated at time 0.2 s. Here, the Hold strategy again shows much worse performance than Zero, with the peak cost being almost an order of magnitude worse. The difference between Kill and Skip-Next is relatively small, with the latter strategy consistently performing slightly worse than the former. This is due to the small extra delay caused by using old data in the Skip-Next strategy.

We conducted experiments on a Furuta pendulum, using the same controller for the real plant rather than its model.[5] Initially, we performed 500 experiments with 500 jobs each and no deadline misses, to determine the nominal variance of the system—i.e., the stationary variance used to find the static cost J_∞. For each strategy $\mathcal{H}$ we then ran 500 identically set up experiments. In each experiment, the control task operated according to the task model from Definition 2, experiencing a burst of length $m = 20$ misses, followed by by a recovery interval with $n = 480$ deadline hits.

Due to system model uncertainties (e.g., friction) being significant, the rotation angle around the arm axis displayed a considerable variance. We removed the state from the covariance calculations, since the arm angle majorly impacted the variance despite its inconsequential significance on the system dynamics (the pendulum can be stabilised with the arm being around any position, provided that the pendulum itself is kept in the upright position). Including the rotation angle would not change the shape of the performance degradation seen in Figure 5. However, it would make the results obtained with different strategies $\mathcal{H}$ not comparable (in some of them, the rotation angle could have varied less across the 500 experiments). The covariance matrix P_k was derived by calculating the variance of the closed-loop state vector $\tilde{x}_k$ according to Equation (9), in each time step k.

The resulting performance cost can be seen in the lower half of Figure 5, where the cost J_k was calculated according to Equation (11) and normalised using the stationary cost J_∞. Comparing the simulated (upper) and real (lower) performance costs in Figure 5, we notice the similarities between the simulated analysis and the analysis performed on the physical plant. Particularly, the strategies involving Hold actuation show similar behaviours. For these strategies, the simulated and real values are very close for the transient burst interval, the secondary cost peak (seen around time 0.4 s), and the maximum normalised cost $J_{M,\mathcal{H}}$. However, the real cost is recovering slower than in the simulations—an effect that arises due to the nonlinear effects present in the real process, but unmodelled in the simulated environment. Instead, comparing the Zero actuation strategies, the performance cost of the physical experiments during the burst interval seem to improve compared to the simulations. This is again likely due to the unmodelled dynamics (e.g., friction) appearing in the physical experiment but not in the simulations. The stiction component of the friction reduces the variance of the states when the actuation signal becomes zero. With longer burst intervals, a similar behaviour as for the Hold actuation strategies would appear. Despite this difference, both the recovery interval, the secondary cost peak (around 0.4 s), and the

[5] A video, showing experiments with the real system and bursts of deadline misses can be viewed at `https://youtu.be/OPOK_7lvKVU`. The video shows a comparison of all the strategies for bursts of ($m = 20, n = 480$). Furthermore, we have included additional experiments with ($m = 50, n = 450$) and ($m = 75, n = 425$) for the Skip&Hold strategy. The results of the additional experiments with higher values of m are not described in the paper, as stability could not be guaranteed (and in fact the pendulum is not at all times kept in the upright position).

maximum normalised costs $J_{M,\mathcal{H}}$ are comparable.

We conclude that the results of the experiments performed on the physical process support the validity of the performance analysis presented in Section 5.2.

6.2 Control Benchmark

In Section 6.1 we extensively discussed the results obtained with a single plant (the Furuta pendulum), with the aim of showing that simulating the performance cost yields interesting and relevant results. As the main novelty of this paper lays in the introduction of the performance analysis as an additional tool to evaluate the behaviour of control systems that can miss deadlines, we here focus on performance.

We use a set of representative process industrial plants [7], developed to benchmark PID design algorithms in the control literature. The set includes 9 different batches of stable plants, each presenting different features that can be encountered in process industrial plants, for a total of 133 plants.[6] For each batch, all systems have the same structure, but different parameters. For example, the fourth batch is a stable system with a set of repeated eigenvalues, and a single parameter specifying the system order, which can take six possible values (3, 4, 5, 6, 7, or 8). Almost all the plants have a single independent parameter. The only exception is Batch 7, for which we can specify two different configuration parameters, the first one having 4 possible values and the second one having 9 potential alternatives, with a total of 36 possible configurations.

The analysis methodology presented in this paper is valid for *all* linear control systems. In Section 6.1, we introduced an LQR controller to analyse the Furuta pendulum. To demonstrate the generality of the analysis, here, we focus on the most common controller class: proportional and integral (PI) controllers. These controllers constitute the vast majority of all the control loops in the process industry.[7] We also performed the analysis for proportional, integral, and derivative (PID) controllers obtaining similar results. Introducing our tuning for PID controllers requires additional clarifications and details, which we omit due to space limitations.

For each plant we derived a PI controller according to the methodology presented in [28]. In order to showcase the applicability of our analysis to different linear systems, controllers, and noise models, we analyse the resulting closed-loop systems for $m \in [1, 20]$, under the assumption that the systems are affected by brown noise (in comparison to the white noise applied to the Furuta Pendulum). The brown noise model integrates the white noise and is thus applicable to systems where the noise is more dominant at lower frequencies (e.g., oscillations from nearby machinery). Figure 6 shows the results for $m = 10$.

The first result that the figure shows is that the plant dynamics plays an important role in how the system reacts to misses. For example, the plants in Batch 4 and Batch 8 need around 20 hits to recover from a burst of 10 misses. On the contrary, the plants in Batch 6 and Batch 7 need a higher number of hits to recover from the same burst interval. The second result that is apparent from the figure is that the Hold actuation strategy recovers much better (performance-wise) than Zero. The reason why Hold outperforms Zero can be explained by the brown noise. The control signal will actively counteract the integrated noise dynamics, meaning that zeroing the control signal removes the compensation against the integrated

[6] In our analysis, we present results with 134 plants. In fact, the test set was used in [28] to assess a control design method, and an additional plant was added to the set during this assessment. We included this additional plant in our analysis.

[7] A 2001 survey by Honeywell [24] states that 97% of the existing industrial controllers are PI controllers.

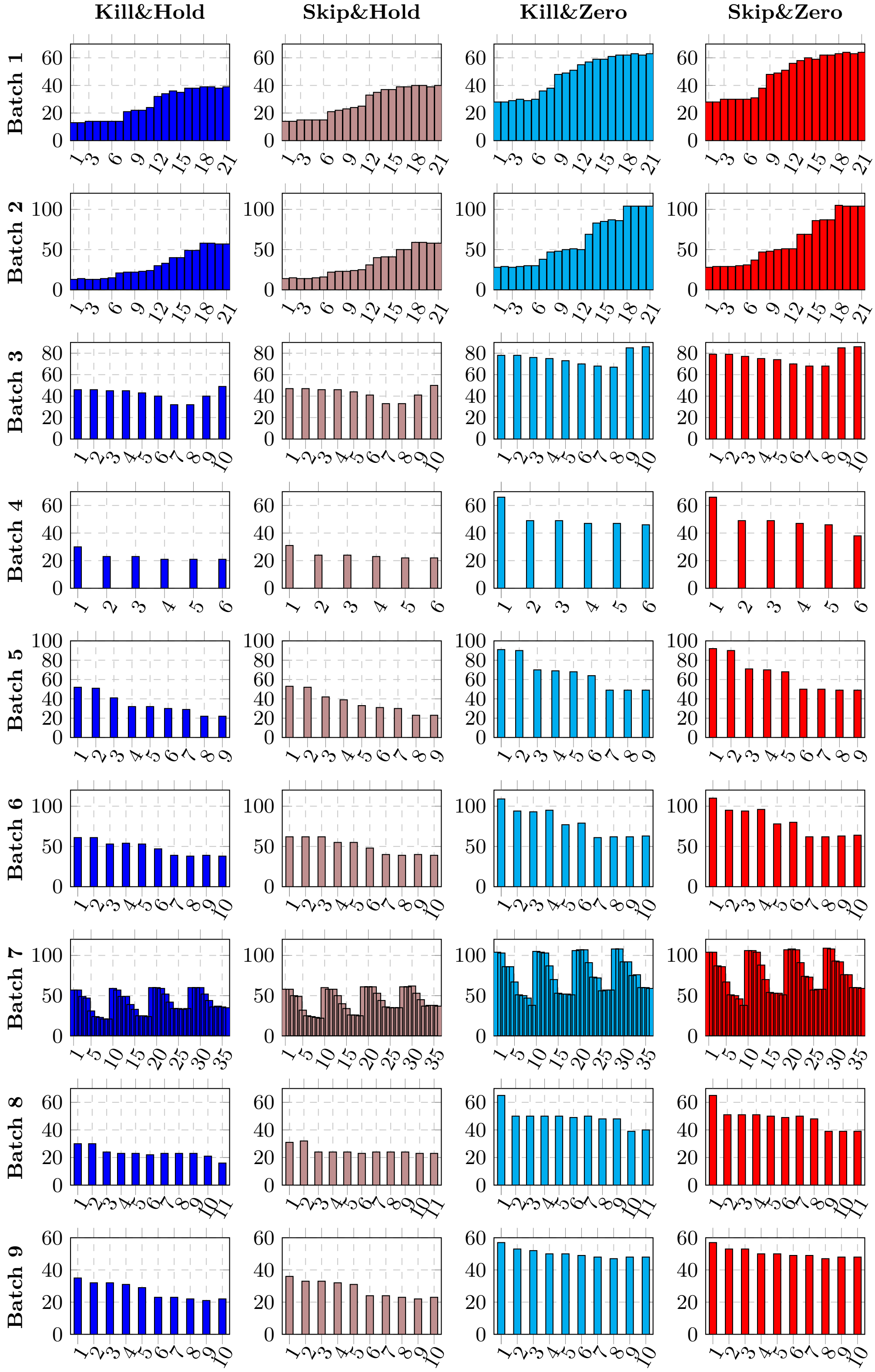

Figure 6 Performance Recovery Interval $n^*_{\mathcal{H}}$ needed to recover from a burst of 10 deadline misses for different strategies and all the plants in the 9 batches for PI controllers designed according to [28].

noise. Finally, comparing the deadline handling strategies, Kill performs marginally better than Skip-Next. Under Kill, the controller uses fresh data at the beginning of the recovery interval, while Skip-Next uses old data. However, we assumed ideal rollback (i.e., zero additional computation time for the rollback and clean state) for the Kill strategy. In real systems, rollback is difficult to realise and the advantage provided by Kill over Skip-Next may therefore become unimportant. These findings are consistent throughout all the plants in the experimental set, regardless of the burst interval length m.

The plant dynamics and noise affect the behaviour and performance of the strategies. Comparing the results of Section 6.1 with the aggregate results, it becomes apparent that the actuation strategy (Zero or Hold) affects control performance significantly more than the deadline handling strategy. For the Furuta pendulum (an unstable, nonlinear plant influenced by white noise) Zero performed the best, but for the process industrial systems (stable, linear plants influenced by brown noise) Hold outperformed Zero. These results were apparent even with no consideration taken to the deadline handling strategies. Thus, we conclude that the plant and noise model should be the ruling factor when choosing the actuation strategy, while the deadline handling strategy is mainly limited by the constraints imposed by the real-time implementation.

7 Conclusions

In this paper we analysed control systems and their behaviour in the presence of bursts of deadline misses. We provided a comprehensive set of tools to determine how robust a given control system is to faults that hinder the computation to complete in time, with different handling strategies. Our analysis tackles both stability and performance. In fact, we have shown that analysing the stability of the system is not enough to properly quantify the robustness to deadline misses, as the performance loss could be significant even for stable systems. We introduced two performance metrics, linked to the recovery of a system from a burst of deadline misses.

A limitation of the presented performance analysis is that it only applies to linear control systems. However, the approach can easily be extended to analyse *time-varying* linear systems and can also be used for local analysis of a nonlinear system that should follow a given reference trajectory. In fact, to illustrate the applicability to real (e.g., nonlinear) systems, we applied the analysis to a Furuta pendulum and compared the results of simulations obtained with a model of the process to the real execution data. The results support our claim that the proposed performance analysis is a valid approximation of the real-world system performance.

We performed additional tests on a large batch of industrial plants, using modern control design techniques. From our experimental campaign, we conclude that the choice of actuation strategy affects the control performance significantly more than the choice of deadline handling strategy.

References

1. F. Abdi, C. Chen, M. Hasan, S. Liu, S. Mohan, and M. Caccamo. Preserving physical safety under cyber attacks. *IEEE Internet of Things Journal*, 6(4), 2019.
2. F. Abdi, R. Mancuso, R. Tabish, and M. Caccamo. Restart-based fault-tolerance: System design and schedulability analysis. In *23rd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2017.

3 F. Abdi, R. Tabish, M. Rungger, M. Zamani, and M. Caccamo. Application and system-level software fault tolerance through full system restarts. In *8th International Conference on Cyber-Physical Systems (ICCPS)*, 2017.

4 L. Ahrendts, S. Quinton, T. Boroske, and R. Ernst. Verifying weakly-hard real-time properties of traffic streams in switched networks. In *30th Euromicro Conference on Real-Time Systems (ECRTS)*, volume 106 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

5 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 (RTSS)*, 2020.

6 S. Altmeyer and R. I. Davis. On the correctness, optimality and precision of static probabilistic timing analysis. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, 2014.

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

8 K. J. Åström and T. Hägglund. *Advanced PID Control.* The Instrumentation, Systems and Automation Society, 2006.

9 K. J. Åström and B. Wittenmark. *Computer-Controlled Systems: Theory and Design.* Prentice Hall, 3rd edition, 1997.

10 G. Bernat and A. Burns. Combining $\binom{n}{m}$-hard deadlines and dual priority scheduling. In *18th IEEE Real-Time Systems Symposium (RTSS)*, pages 46–57, 1997.

11 G. Bernat, A. Burns, and A. Liamosi. Weakly hard real-time systems. *IEEE Transactions on Computers*, 50:308–321, 2001.

12 T. Bund and F. Slomka. Controller/platform co-design of networked control systems based on density functions. In *4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, pages 11–14. ACM, 2014.

13 T. Bund and F. Slomka. Worst-case performance validation of safety-critical control systems with dropped samples. In *23rd International Conference on Real Time and Networks Systems (RTNS)*, pages 319–326. ACM, 2015.

14 G. Buttazzo, M. Velasco, and P. Marti. Quality-of-control management in overloaded real-time systems. *IEEE Transactions on Computers*, 56(2):253–266, 2007.

15 M. Caccamo and G. Buttazzo. Exploiting skips in periodic tasks for enhancing aperiodic responsiveness. In *18th IEEE Real-Time Systems Symposium (RTSS)*, pages 330–339, 1997.

16 M. Caccamo, G. Buttazzo, and L. Sha. Capacity sharing for overrun control. In *21st IEEE Real-Time Systems Symposium (RTSS)*, pages 295–304, 2000.

17 B. S. Cazzolato and Z. Prime. On the dynamics of the Furuta pendulum. *Journal of Control Science and Engineering*, 2011.

18 A. Cervin. Analysis of overrun strategies in periodic control tasks. *IFAC Proceedings Volumes*, 38(1):219–224, 2005.

19 A. Cervin, P. Pazzaglia, M. Barzegaran, and R. Mahfouzi. Using JitterTime to analyze transient performance in adaptive and reconfigurable control systems. In *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1025–1032, 2019.

20 A. Chen, Ha. Xiao, A. Haeberlen, and L. T. X. Phan. Fault tolerance and the five-second rule. In *Workshop on Hot Topics in Operating Systems (HotOS)*, 2015.

21 H. Choi, H. Kim, and Q. Zhu. Job-class-level fixed priority scheduling of weakly-hard real-time systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 241–253, 2019.

22 R. I. Davis, L. Santinelli, S. Altmeyer, C. Maiza, and L. Cucu-Grosjean. Analysis of probabilistic cache related pre-emption delays. In *25th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 168–179, 2013.

23 D. de Niz, L. Wrage, A. Rowe, and R. Rajkumar. Utility-based resource overbooking for cyber-physical systems. In *19th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 217–226, 2013.

24 L. Desborough. Increasing customer value of industrial control performance monitoring-honeywell's experience. *Preprints of CPC*, pages 153–186, 2001.

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

26 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 (RTSS)*, pages 53–62, 2014.

27 K. Furuta, M. Yamakita, and S Kobayashi. Swing-up control of inverted pendulum using pseudo-state feedback. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 206(4):263–269, 1992.

28 O. Garpinger and T. Hägglund. Software-based optimal PID design with robustness and noise sensitivity constraints. *Journal of Process Control*, 33:90–101, 2015.

29 M. Gaukler, T. Rheinfels, P. Ulbrich, and G. Roppenecker. Convergence rate abstractions for weakly-hard real-time control. *arXiv preprint arXiv:1912.09871*, 2019.

30 S. Kumar Ghosh, S. Dey, D. Goswami, D. Mueller-Gritschneder, and S. Chakraborty. Design and validation of fault-tolerant embedded controllers. In *Design, Automation & Test in Europe Conference Exhibition (DATE)*. IEEE, 2018.

31 D. Goswami, D. Mueller-Gritschneder, T. Basten, U. Schlichtmann, and S. Chakraborty. Fault-tolerant embedded control systems for unreliable hardware. In *International Symposium on Integrated Circuits (ISIC)*. IEEE, 2014.

32 A. Gujarati, M. Nasri, and B. B. Brandenburg. Quantifying the resiliency of fail-operational real-time networked control systems. In *30th Euromicro Conference on Real-Time Systems (ECRTS)*, volume 106 of *Leibniz International Proceedings in Informatics (LIPIcs)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

33 A. Gujarati, M. Nasri, R. Majumdar, and B. B. Brandenburg. From iteration to system failure: Characterizing the fitness of periodic weakly-hard systems. In *31st Euromicro Conference on Real-Time Systems (ECRTS)*, volume 133 of *Leibniz International Proceedings in Informatics (LIPIcs)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

34 M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m,k)-firm deadlines. *IEEE Transactions on Computers*, 44(12):1443–1451, 1995.

35 Z. A. H. Hammadeh, R. Ernst, S. Quinton, R. Henia, and L. Rioux. Bounding deadline misses in weakly-hard real-time systems with task dependencies. In *Design, Automation & Test in Europe Conference Exhibition (DATE)*, pages 584–589, 2017.

36 Z. A. H. Hammadeh, S. Quinton, and R. Ernst. Extending typical worst-case analysis using response-time dependencies to bound deadline misses. In *14th International Conference on Embedded Software (EMSOFT)*. ACM, 2014.

37 Z. A. H. Hammadeh, S. Quinton, and R. Ernst. Weakly-hard real-time guarantees for earliest deadline first scheduling of independent tasks. *ACM Transactions of Embedded Computing Systems*, 18(6), 2019.

38 Z. A. H. Hammadeh, S. Quinton, M. Panunzio, R. Henia, L. Rioux, and R. Ernst. Budgeting under-specified tasks for weakly-hard real-time systems. In *29th Euromicro Conference on Real-Time Systems (ECRTS)*, volume 76 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.

39 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)*, pages 1680–1685, 2019.

40 R. Jungers. *The Joint Spectral Radius: Theory and Applications*. Lecture Notes in Control and Information Sciences. Springer Berlin Heidelberg, 2009.

41 M. Kauer, D. Soudbakhsh, D. Goswami, S. Chakraborty, and A. M. Annaswamy. Fault-tolerant control synthesis and verification of distributed embedded systems. In *Design, Automation & Test in Europe Conference Exhibition (DATE)*, 2014.

42 F. Khosravi, M. Glaß, and J. Teich. Automatic reliability analysis in the presence of probabilistic common cause failures. *IEEE Transactions on Reliability*, 66(2), 2017.

43 F. Khosravi, M. Müller, M. Glaß, and J. Teich. Uncertainty-aware reliability analysis and optimization. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 97—102, 2015.

44 C. Kirsch and A. Sokolova. The logical execution time paradigm. In *Advances in Real-Time Systems*, pages 103–120. Springer Berlin Heidelberg, 2012.

45 G. Koren and D. Shasha. Skip-Over: algorithms and complexity for overloaded systems that allow skips. In *16th IEEE Real-Time Systems Symposium (RTSS)*, pages 110–117, 1995.

46 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.

47 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.

48 M. Maggio, A. Hamann, E. Mayer-John, and D. Ziegenbein. Control-system stability under consecutive deadline misses constraints. In *32nd Euromicro Conference on Real-Time Systems (ECRTS)*, Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

49 D.C. Montgomery. *Introduction to Statistical Quality Control*. Wiley, 2009.

50 S. Natarajan, M. Nasri, D. Broman, B. B. Brandenburg, and G. Nelissen. From code to weakly hard constraints: A pragmatic end-to-end toolchain for timed C. In *40th IEEE Real-Time Systems Symposium (RTSS)*, pages 167–180, 2019.

51 P. P. O'Connor and A. Kleyner. *Practical Reliability Engineering*. Wiley Publishing, 5th edition, 2012.

52 L. Palopoli, L. Abeni, G. Buttazzo, F. Conticelli, and M. Di Natale. Real-time control system analysis: an integrated approach. In *21st IEEE Real-Time Systems Symposium (RTSS)*, pages 131–140, 2000.

53 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.

54 P. Pazzaglia, C. Mandrioli, M. Maggio, and A. Cervin. DMAC: Deadline-Miss-Aware Control. In *31st Euromicro Conference on Real-Time Systems (ECRTS)*, volume 133 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

55 P. Pazzaglia, L. Pannocchi, A. Biondi, and M. Di Natale. Beyond the weakly hard model: Measuring the performance cost of deadline misses. In *30th Euromicro Conference on Real-Time Systems (ECRTS)*, volume 106 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:22, 2018.

56 S. Quinton, T. T. Bone, J. Hennig, M. Neukirchner, M. Negrean, and R. Ernst. Typical worst case response-time analysis and its use in automotive network design. In *51st Annual Design Automation Conference (DAC)*, pages 1–6, New York, NY, USA, 2014. ACM.

57 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.

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

59 D. Soudbakhsh, L. T. X. Phan, A. M. Annaswamy, and O. Sokolsky. Co-design of arbitrated network control systems with overrun strategies. *IEEE Transactions on Control of Network Systems*, 5(1):128–141, 2018.

60 D. Soudbakhsh, L. T. X. Phan, O. Sokolsky, I. Lee, and A. Annaswamy. Co-design of control and platform with dropped signals. In *4th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pages 129–140. ACM, 2013.

61 Y. Sun and M. Di Natale. Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks. *ACM Transactions on Embedded Computing Systems*, 16(5s), 2017.

62 G. Vankeerberghen, J. Hendrickx, and R. M. Jungers. JSR: A toolbox to compute the joint spectral radius. In *17th International Conference on Hybrid Systems: Computation and Control (HSCC)*, pages 151—156. ACM, 2014.

63 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)*, volume 80 of *OpenAccess Series in Informatics*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

64 W. Xu, Z. A. H. Hammadeh, A. Kröller, R. Ernst, and S. Quinton. Improved deadline miss models for real-time systems using typical worst-case analysis. In *27th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 247–256, 2015.