Control-System Stability Under Consecutive Deadline Misses Constraints

Martina Maggio
Saarland University, Department of Computer Science, Saarbrücken, Germany
Lund University, Department of Automatic Control, Sweden
Robert Bosch GmbH, Renningen, Germany
maggio@cs.uni-saarland.de

Arne Hamann
Robert Bosch GmbH, Renningen, Germany
arne.hamann@de.bosch.com

Eckart Mayer-John
Robert Bosch GmbH, Renningen, Germany
eckart.mayer@de.bosch.com

Dirk Ziegenbein
Robert Bosch GmbH, Renningen, Germany
dirk.ziegenbein@de.bosch.com

Abstract

This paper deals with the real-time implementation of feedback controllers. In particular, it provides an analysis of the stability property of closed-loop systems that include a controller that can sporadically miss deadlines. In this context, the weakly hard m-K computational model has been widely adopted and researchers used it to design and verify controllers that are robust to deadline misses. Rather than using the m-K model, we focus on another weakly-hard model, the number of consecutive deadline misses, showing a neat mathematical connection between real-time systems and control theory. We formalise this connection using the joint spectral radius and we discuss how to prove stability guarantees on the combination of a controller (that is unaware of deadline misses) and its system-level implementation. We apply the proposed verification procedure to a synthetic example and to an industrial case study.

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

Keywords and phrases Real-Time Control, Deadline Misses, Weakly Hard Models

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2020.21

Funding This work was supported by: the ELLIIT Strategic Research Area, the project *ARAMiS II* of the German Federal Ministry for Education and Research with the funding ID 01IS16025. The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871259 (ADMORPH). The responsibility for the content remains with the authors.

Acknowledgements This research was developed while Martina Maggio was on sabbatical at Robert Bosch GmbH.

1 Introduction

The contribution of this paper is a verification procedure to prove the robustness of controller implementations to deadline misses. For this task, literature contributions focus on the computation model where the control task can *miss* at most m deadlines in a window of K


© Martina Maggio, Arne Hamann, Eckart Mayer-John, and Dirk Ziegenbein;
licensed under Creative Commons License CC-BY

32nd Euromicro Conference on Real-Time Systems (ECRTS 2020).
Editor: Marcus Völp; Article No. 21; pp. 21:1–21:24

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

activations (i.e., the m-K model). This was one of four proposed models [3] to analyse systems with deadline misses. In this paper we show that there is a natural analytical connection between another of these four models, the number of maximum consecutive deadline misses, and control-theoretical tools that can be used to prove properties of closed-loop systems, such as stability.

Historical Perspective. During the past couple of decades the real-time systems community made an effort in formalising requirements, models, and algorithms to handle systems that can (sporadically) miss deadlines. In this quest, Hamdaoui and Ramanathan analysed tasks that behave according to the (m, k) model [18]. With this model, tasks can be modelled as sequences of jobs. In every set of k consecutive jobs, at least m jobs must meet their deadline. This model was analysed, finding schedulability conditions and scheduling schemes, e.g., [35]. Building on the ideas from this research, the *weakly hard* model of computation was formalised [3]. A weakly hard real-time system is a system in which the distribution of deadline misses and hits during a window of time is precisely bounded.

Bernat, Burns and Liamosí [3] give four possible definitions for a weakly hard task τ.

1. $\tau \vdash \binom{n}{m}$, with $1 \leq n < m$: According to this definition, for each set of m consecutive deadlines, τ meets at least n of them. With a slight difference in notation, this model corresponds to the (m, k) model by Hamdaoui and Ramanathan [18].
2. $\tau \vdash \left\langle {n \atop m} \right\rangle$, with $1 \leq n < m$: Here, the system guarantees that for each set of m consecutive deadlines, τ meets consecutively n of them.
3. $\tau \vdash \overline{\binom{n}{m}}$, with $1 \leq n < m$: This is the dual definition with respect to the first one. In this case, the system guarantees that for each set of m consecutive deadlines, τ misses at most n of them.
4. $\tau \vdash \overline{\langle n \rangle}$, $n \geq 1$: According to this definition the maximum number of consecutive deadline misses that τ can experience is n.[1]

The third of these models gained traction in the research community, and the term weakly-hard task started to indicate a task that can experience a bounded number of misses in a window of jobs. In particular, with a slightly confusing terminology, this third model is also often called the m-K model.[2] This specifies that a task can experience at most m misses in a window of K consecutive jobs [1, 9, 10, 12, 14, 19, 20, 33, 34, 41–43].

The m-K Model for Control Tasks. In the attempt to achieve computing and control co-design, both schedulability results like [43] and analysis using model checking [12] have been investigated. The research community contributed with criteria to determine stability [5], determine convergence rates [14], and to design controllers in the presence of deadline misses [29]. Furthermore, the performance cost of deadline misses was investigated [34, 44], together with the role of the strategy used to handle the misses [33, 41], i.e., killing the task or allowing its continuation with different policies for the following iteration. The general consensus is that sequences of misses and hits (i.e., the m-K model) are to be considered and analysed to determine physical properties of the system.

---

[1] The original definition was $\tau \vdash \overline{\left\langle {n \atop m} \right\rangle}$, with $1 \leq n < m$. However, according to [3, Theorem 4] there is no need to specify the window size, i.e., the task τ can be equivalently defined using any window, hence we use $\tau \vdash \overline{\langle n \rangle}$.

[2] Notice the difference between the (m, k) model (that stated that there were m hits for every sequence of k consecutive jobs, with $1 \leq m < k$) and the m-K model (that imposes that there are at most m misses in every window of K jobs, with $1 \leq m < K$).

Contribution. In this paper we use the $\tau \vdash \overline{\langle n \rangle}$ model and cast the problem of verifying the stability of closed-loop systems into a neat mathematical framework. The solution ensures the stability of the combination of: (1) the physics of the plant under control and, (2) the execution of the controller (that may miss deadlines). We believe that the $\tau \vdash \overline{\langle n \rangle}$ model is as relevant as the m-K model from the industrial standpoint.

In fact, in industrial products, deadline misses are often caused by transient overload periods or faults. In many industrial applications, a system load of over 80% is targeted for cost reasons. Such a high system load can usually not be achieved with purely formal methods that are based on worst-case considerations, especially on multi-core platforms. For this reason, many industrial systems are designed for average runtimes plus a safety margin, in conjunction with rate monotonic scheduling. Our experience is that, following this practical approach in contrast to analytical ones, deadline violations may occur, e.g., due to a transient interrupt load. During these transient periods the miss ratio often is quite high, making the number of consecutive misses a very relevant indicator of the system performance. We argue that methods that evaluate and prove the robustness of controllers to deadline violations in this setup are of high industrial interest.

Outline. In the remainder of this paper we will recap the necessary control background and then provide our contribution. In particular, Section 2 explains how a plant is modelled and how a *state feedback controller* is applied to regulate the plant's behaviour. Section 3 describes the strategies that are typically used to handle deadline misses and provides some insights on what is the best choice from the system perspective. Section 4 shows how to guarantee properties of closed-loop systems (like stability) in the presence of deadline misses with the $\tau \vdash \overline{\langle n \rangle}$ model. Section 5 shows some experimental results validating our claims. Finally, Section 6 presents an overview of related work and Section 7 concludes the paper.

2 Control Background

In this section we recap the basic concepts of control theory that are used in the rest of the paper. We analyse linear time-invariant models and controllers implemented as periodic tasks with implicit deadlines.

Plant Model. The starting point for control design is always understanding the object that the controller should act upon. The control engineer obtains a model $\mathcal{P}_c$ of the plant to control. In most cases, this model is linear and time-invariant, and represents with ordinary differential equations the dynamics of the system in the following form.

$$\mathcal{P}_c : \begin{cases} \dot{x}(t) = A_c\, x(t) + B_c\, u(t) \\ y(t) = C_c\, x(t) + D_c\, u(t) \end{cases} \tag{1}$$

Here, the system state $x(t) = [x_1(t), \ldots, x_p(t)]^{\mathrm{T}}$ evolves depending on the current state and the input signal $u(t) = [u_1(t), \ldots, u_r(t)]^{\mathrm{T}}$, where the superscript $^{\mathrm{T}}$ indicates the transposition operator. We denote with p the number of state variables (i.e., the length of vector x) and with r the number of input variables (i.e., the length of vector u). The matrices A_c, B_c, C_c, and D_c encode the dynamics of the system. In the following, we will make two assumptions: D_c is a zero matrix of appropriate size, and C_c is the unit matrix of appropriate size.[3] The

[3] A_c is a $p \times p$ matrix, B_c is a $p \times r$ matrix, C_c is a $p \times p$ matrix, and D_c is a $p \times r$ matrix.

first assumption means that the system is *strictly proper* and holds for almost all the physical models used in control. The second assumption means that the state is measurable. This does not always hold for real systems, but state observers can be built whenever this is not true [26], to estimate the state $x(t)$.[4] This means that, without losing generality, we can represent $\mathcal{P}_c$ as

$$\dot{x}(t) = A_c\, x(t) + B_c\, u(t), \tag{2}$$

and describe the system dynamics using only A_c and B_c.

From this starting point, control systems are usually designed and realised in one of these two ways:

1. The plant model $\mathcal{P}_c$ is used to synthesise a controller (model) $\mathcal{C}_c$ in continuous-time. Closed-loop system properties, like stability, are proven on the feedback interconnection of $\mathcal{C}_c$ and $\mathcal{P}_c$. However, when the controller is implemented, digital hardware is used. This means that the controller model $\mathcal{C}_c$ has to be discretised, obtaining $\mathcal{C}_d$. $\mathcal{C}_d$ describes the behaviour of $\mathcal{C}_c$ at given sampling instants.
2. The model of the plant in continuous time $\mathcal{P}_c$ is discretised, obtaining a discrete-time plant model $\mathcal{P}_d$. $\mathcal{P}_d$ describes the behaviour of $\mathcal{P}_c$ at given sampling instants. A controller $\mathcal{C}_d$ is designed directly in the discrete-time framework, using the discrete-time plant model $\mathcal{P}_d$. Closed-loop properties are proven on the feedback interconnection of $\mathcal{C}_d$ and $\mathcal{P}_d$.

In both cases, when an object (being it the plant or the controller) is discretised, a sampling period π is chosen. With either design methods, we can obtain a discrete-time model of the plant $\mathcal{P}_d$ and of the controller $\mathcal{C}_d$. In control theory, usually it is possible to prove properties of the interconnection between these two models. In particular, we use $\mathcal{C}_d$ rather than $\mathcal{C}_c$ to prove properties using the controller that is closer to the real implementation. However, on top of what is done in classical control theory, we want to take into account deadline misses.

We discretise $\mathcal{P}_c$ from Equation (2). From the representation in terms of ordinary differential equations, we obtain the system of difference equations $\mathcal{P}_d$ as

$$\mathcal{P}_d : x_{[k+1]} = A_d\, x_{[k]} + B_d\, u_{[k]}. \tag{3}$$

Here, k counts the sampling instants (i.e., there is a distance of π [s] between the k-th and the $k+1$-th instant). The matrices A_d and B_d are the counterparts of A_c and B_c for the continuous-time system. They describe the evolution of the system in discrete-time, have the same dimensions of the corresponding continuous-time matrices, and their elements depend on the choice of the sampling period π.

Controller Model. Once a model of the plant is available, control design can be carried out with many different methods. In this paper we tackle periodic controllers expressed as state feedback controllers, i.e., controllers that execute periodically with period π and whose discrete-time form is

$$u_{[k]} = K_k\, x_{[k]}. \tag{4}$$

The control design problem is the problem of finding the matrix K_k that stabilises the system and obtains some desired properties. The state feedback formulation is more general than it may seem at a first glance. State feedback controllers are not purely proportional controllers,

[4] As a remark, if a state observer is present its dynamics should be taken into account in the analysis. This extension only requires to augment the system state with the rows and columns corresponding to the execution of the observer, but the analysis method remains the same.

although their update is proportional to the state vector. It is possible to augment the state vector of the system – for example introducing an error term and its integral – to achieve controllers that are not simply proportional but contain integral action.[5] Additionally, it is possible to use pole placement [26], or to compute optimal controllers using the Linear Quadratic Regulator [25] formulation.

In an industrial setting,[6] many controllers are still designed assuming zero latency and instantaneous computation [46], i.e., assuming that it takes zero time to retrieve the sensor measurement from the plant, compute the control signal, and apply it. When the dynamics of the plant are slow and the controller is able to sample and measure signals at a reasonable speed, this assumption does not significantly affect the behaviour of the system. However, in most cases, basic properties like stability can be violated because of the computational delays that are introduced in the loop. The controller job that is activated at time t_a completes its execution at time t_c, where t_c is in the controller period, i.e., $t_c \in (t_a, t_a + \pi]$, introducing a computational delay $t_c - t_a$.

Due to this computational delay, in industry, it became common practice to design control systems following the Logical Execution Time (LET) paradigm and to synchronise input and output exactly to the period boundary. In this case, the control signal is computed within a control period and applied at the beginning of the next period. This enhances the predictability of the system, allows the processor to execute other tasks without affecting the control properties, and ensures a consistent behaviour.

In control terms, this means that the controller actuates its control signal computation with a one-step delay. Assuming that the cycle of sampling, computing, and actuating can be always terminated within a control period, this allows the designer to synthesise an optimal controller regardless of the time-varying components of the computational delay such as activation jitter, unpredictable interrupts, uncertain computation times [28]. The equation for the state feedback controller then becomes

$$\mathcal{C}_d : u_{[k]} = K\, x_{[k-1]}, \tag{5}$$

where K is the designed controller. With very few exceptions, the vast literature on control design assumes that the deadlines to compute control signals are always met. Recently, Linsenmayer and Allgöwer started to connect the theory of m-K real-time systems (i.e., the $\tau \vdash \overline{\binom{m}{K}}$ model) with control design [29], showing that it is in some cases possible to design a state feedback controller that is robust (i.e., guarantees stability) to deadline misses. In this paper we will connect the amount of possible consecutive deadline misses (i.e., the $\tau \vdash \overline{\langle n \rangle}$ model) to the analysis of stability as a control design property.

[5] The most widely adopted controllers in industry are the Proportional and Integral (PI) or the Proportional, Integral and Derivative (PID) controllers. These controllers can be expressed in state-feedback form (as seen later in Section 5 for a specific example), by augmenting the system state $x(t)$ with the difference between the desired state values and the obtained ones, i.e., the error, and its integral, or sum, over time. There is a small difference between the controller expressed in state-feedback form and the controller expressed as a state-space system. In the first case, when the controller misses its deadline, the update function for the state is still executed (as it is now part of the system equation). It is however possible to generalise the findings in this paper to handle controllers in state-space form.

[6] In fact, a survey published in 2001 by Honeywell [11] states that 97% of the existing industrial controllers are PI controllers and use no delay compensation. This does not mean that the control community has not developed solutions to properly address delays in the control design. It simply means that in many industrial settings the design is still simple and limited to considering the computation instantaneous.

Feedback Interconnection. Assume there are no deadline misses. In this case, we can plug the value of $u_{[k]}$ obtained from Equation (5) into the plant Equation (3), obtaining

$$x_{[k+1]} = A_d\, x_{[k]} + B_d\, K\, x_{[k-1]}. \tag{6}$$

To analyse the closed-loop system, we define a new state variable $\tilde{x}_{[k]} = [x_{[k]}^{\mathrm{T}}, x_{[k-1]}^{\mathrm{T}}]^{\mathrm{T}}$ (the superscript $^{\mathrm{T}}$ indicates the result of the transposition operator). We recall that p denotes the order of the system (i.e., the number of state variables in vector $x_{[k]}$). Using the new state variable $\tilde{x}_{[k]}$, Equation (6) can be rewritten as

$$\tilde{x}_{[k+1]} = \begin{bmatrix} x_{[k+1]} \\ x_{[k]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & B_d\, K \\ I_p & 0_{p\times p} \end{bmatrix}}_{A} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \end{bmatrix} = A\, \tilde{x}_{[k]}, \tag{7}$$

where I_p and $0_{p\times p}$ are respectively the identity matrix and the zero matrix of size of the number of state variables p.

Stability. A discrete-time linear time-invariant system is asymptotically stable if and only if all the eigenvalues of its state matrix are strictly inside the unit disk. For the system shown in Equation (7), this means that the eigenvalues of A should have magnitude strictly less than one.

Another way of formulating the stability requirement uses the concept of *spectral radius* $\rho(A)$. The spectral radius is defined as the maximum magnitude of the eigenvalues of A. If we denote with $\{\lambda_1, ..., \lambda_n\}$ the set of eigenvalues of A, this means

$$\rho(A) = \max\{|\lambda_1|, \ldots, |\lambda_n|\}. \tag{8}$$

Requiring that all the eigenvalues have magnitude strictly less than one is equivalent to stating that the spectral radius of the A matrix should be less than 1.

This only proves the stability of the system in absence of deadline misses. However, we are aware that sporadic misses can occur, either due to faults [16] or to the chosen period π not satisfying the requirement of worst-case response time for the controller task τ being less than the controller period [12, 33, 34].

3 Deadline Miss

In order to properly analyse the closed-loop system properties when deadlines can be missed, it is necessary to define a model of how the system reacts to deadline misses. There are two aspects of this reaction: (i) what is the chosen control signal when a miss occurs [29], and (ii) how is the operating system treating the job that missed the deadline [33]. In the remainder of this section, suppose that in the k-th iteration the controller task τ did not complete its execution before the deadline, i.e., it does not complete its computation before time $(k+1)\,\pi$ [s]. We denote time $(k+1)\,\pi$ [s] with t_m.

Control Signal. At time t_m, a control signal should be applied to the plant. Two alternatives have been identified for how to select the next control signal [29]: zero and hold.

1. *Zero:* The control signal $u_{[k+1]}$ is set to zero.
2. *Hold:* The control signal $u_{[k+1]}$ is unchanged, i.e., it is the previous value of the control signal $u_{[k]}$.

The choice of these two alternatives often depends on the control goal that should be achieved.

When a controller is designed for setpoint tracking (i.e., to ensure that the value of some physical quantity follows a desired profile – e.g., to have a robot follow a desired trajectory), the control signal is usually zero in case the measured physical quantity is equal to its setpoint. In this case, setting the control signal to zero means assuming that the model of the plant is correct and the computation does not need correction. When a controller is designed for disturbance rejection (i.e., to ensure that the effect of some physical disturbance is not visible in the measurements – e.g., to keep the altitude of a helicopter constant despite wind) then the control signal is usually a reflection of the effort needed to counteract the disturbance. In this case, holding the previous value of the control signal means making the assumption that the system is experiencing the same disturbance.

System-Level Action. The second decision to make is the choice of what to do with the job that missed the deadline. In this case three alternatives have been proposed [33]: kill, skip-next, and queue(1).

1. *Kill:* At time t_m the job that missed the deadline is killed and a new job is activated.
2. *Skip-next:* At time t_m the job that missed its deadline is allowed to continue with the same scheduling parameters (e.g., priority or budget) and carries on in the next period. The job that should have been activated at the deadline missed is not activated, and the next activation is set to $t_m + \pi$.
3. *Queue(1):* At time t_m the job that missed its deadline is continued. A new job is activated with deadline $t_m + \pi$. The two jobs share the scheduling parameters during the period interval $[t_m, t_m + \pi]$. At time $t_m + \pi$, the most recent update of the control value is applied. If both jobs finish their computation, the control variable is set to the value produced by the most recently activated job (i.e., the job that started at time t_m and was placed in the queue until the old job that missed its previous deadline finished). If only the first job finishes the computation the control variable is set to the value of the job that finished and the following one is continued in the subsequent period.

In Section 4 we will analyse the system in all possible configurations. However, we point out that, from an implementation perspective, killing the control job may not be feasible in many industrial settings. In fact, the system has reached an inconclusive intermediate state. The internal state of the controller could have been updated and the system should implement a clean rollback of these changes. Implementing a clean rollback procedure is risky. Furthermore, if the lengthy computation (and subsequent deadline miss) is due to the received input values, it is likely that the next iteration will start from state values that are fairly close to the previous ones, with higher than normal risk of missing a deadline.

We also notice that enqueuing the task could be beneficial from the control perspective, because a computation with most recent measurements of the state variables could be applied. However, the scheduling parameters for τ have most likely been tuned for one single control job to be executed in a period. For example, if the control task is executed using reservation-based scheduling, its budget is selected to match one execution. When using fixed-priority scheduling, the controller priority has been selected. Executing a second control task may create ripple effects and have a disruptive effect on lower priority tasks.

Finally, if the deadline is missed, this means that the system is likely experiencing a transient overload state, which would make *skip-next* the best option to relieve some pressure from the system.

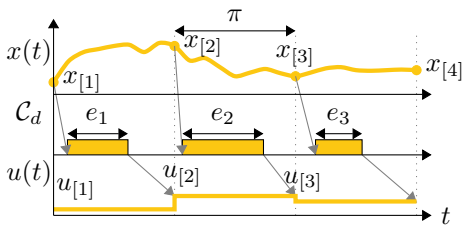

Figure 1 System evolution in case no deadline is missed. The state feedback controller $\mathcal{C}_d$ computes the control signal u based on measurements of the state x.

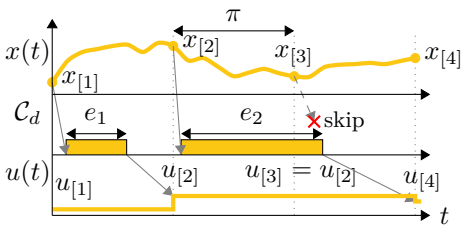

Figure 2 System evolution in case of a deadline miss with the hold policy and skip-next strategy. The controller misses the deadline and completes in the subsequent period.

4 System Analysis

In this section we present our analysis of the closed-loop system with deadline misses. We first discuss the fundamentals of what happens from the physical perspective when a deadline is missed and then discuss the combinations identified in Section 3 for how the system handles the miss.

Fundamentals. Here we present the general methodology that we apply to verify the stability of closed-loop systems with different strategies. We cast the problem into a switching-systems stability problem and show how real-world implementations behave.

Within one control period, there are two possible realisations. The controller job that was activated at time $k\,\pi$ can either hit or miss its deadline. Figure 1 shows the case in which no deadline is missed, while Figure 2 shows the behaviour of the system when a deadline miss occur with the *hold and skip-next* strategy. In the figures, we use e_i to indicate the execution time of the i-th job of the controller. The figure just provides a visual representation of a lengthy execution, but misses can occur due to other sources of interference, e.g., higher priority task being executed with a fixed-priority scheduling algorithm, interrupts being raised and served during the execution of the control task, or access to locked shared resources being requested. In Figure 2, the control signal $u_{[2]}$ is held as $u_{[3]}$. The next controller execution instance is skipped and the result of the completion of e_2 is applied as $u_{[4]}$.

The procedure that we follow to analyse the closed loop system is the following:

1. We express the dynamics of the closed-loop system in the cases of hit and of miss. Following a procedure similar to the one we used in Equation (7), we determine the state matrices for the closed-loop systems in case of deadline hit and deadline miss, respectively A_H and A_M. We then know that the system with (unconstrained) deadline misses can be expressed as a switching system [27] that arbitrarily switches between these two matrices. If the original system in Equation (3) was unstable, there is no hope that the switching system that arbitrarily switches between A_H and A_M is stable (as either an old or no control action is applied when a miss occurs). However, we still have not introduced any weakly hard constraint.
2. We determine the set of possible cases for the evolution of the system when $\tau \vdash \overline{\langle n \rangle}$ guarantees are provided, i.e., the possible realisations of the system behaviour. We denote with Σ the set of possible matrices that these realise. For $\tau \vdash \overline{\langle n \rangle}$ guarantees, the set of possible realisations is $\{H, MH, \ldots, M^n H\}$. The set contains either a single

hit, or a certain number of misses (up to n) followed by a hit.[7] This means that $\Sigma = \{A_H, A_H A_M, A_H A_M^2, ... A_H A_M^n\}$. This can be written in a compact form as $\Sigma = \{A_H A_M^i \mid i \in \mathbb{Z}^{\geq}, i \leq n\}$ where $\mathbb{Z}^{\geq}$ indicates the set of integers including zero. Notice that matrices are multiplied from the right to the left (denoting the standard evolution of the system from a mathematical standpoint). This step introduces the weakly hard constraint for which we investigate the system stability.

3. We compute a generalisation of the spectral radius concept, called *joint spectral radius* $\rho(\Sigma)$ [21, 36], that allows us to assess the stability of the closed-loop system that switches between the realisations (i.e., the valid scenarios including a number of misses between 0 and n followed by a hit) included in Σ. More precisely, the closed-loop system that can switch between the realisations included in Σ is asymptotically stable *if and only if* $\rho(\Sigma) < 1$ [21, Theorem 1.2].

In order to generalise the spectral radius to a set of matrices, we introduce some notation. The following paragraphs are using the notation and sequential treatise proposed in [21] to introduce the concept of the joint spectral radius. We recap only what is needed for the purpose of understanding our analysis.

Joint Spectral Radius [21, 36]. The first step for our definition is to determine what happens when some steps of evolution of the switching system occur. We then denote with $\rho_\mu(\Sigma)$ the spectral radius of the matrices that we find after μ-steps. Precisely,

$$\rho_\mu(\Sigma) = \sup\{\rho(A)^{1/\mu} : A \in \Sigma^\mu\}. \tag{9}$$

In this definition we quantify the average growth over μ time steps, as the supremum of the spectral radius (elevated to the power $1/\mu$) of all the matrices that can be evolutions of the system after μ matrix multiplications (i.e., after μ evolution steps, where an evolution step is either a hit or a set of constrained misses followed by a hit). Equation (9) denotes the supremum of all the possible combinations of products of μ matrices that are included in Σ.

Using $\rho_\mu(\Sigma)$ we can define the joint spectral radius of a bounded set of matrices Σ as

$$\rho(\Sigma) = \limsup_{\mu \to \infty} \rho_\mu(\Sigma). \tag{10}$$

We are then looking at the evolution of the system for an infinite amount of time, i.e., pushing μ to the limit.

Determining that the switching system is asymptotically stable is equivalent to assessing that the joint spectral radius of the set of matrices Σ is less than 1. This condition is both sufficient and necessary [21, Theorem 1.2]. This means that if the joint spectral radius is higher than 1, there is at least a sequence of switches of hits and misses that destabilises the closed-loop system.

Joint Spectral Radius Computation. On the practical side, the problem of computing if the joint spectral radius is less than 1 is undecidable [8]. In many cases it is possible to approximate the joint spectral radius with satisfactory precision [6, 7, 17, 32] and obtain upper

[7] The notation used to define Σ is slightly simplified here, as the matrix A_H may be different depending on how many deadlines have been missed (for example, with the skip-next strategy the controller uses an old measurement of the state to compute the control signal). We will be more precise in the following when we show how to apply the procedure to the different cases. Furthermore, notice that the matrices in Σ represent the evolution across a different number of time steps: A_H advances the time in the system of π, while $A_H A_M$ advances the system time of $2\,\pi$. This is not a concern for the system analysis.

and lower bounds for $\rho(\Sigma)$. Clearly, the closest the two bounds are, the more precise is the estimation of the true value of the joint spectral radius. We can safely say that our controller design is sufficiently robust to deadline misses if the upper bound on the joint spectral radius $\rho(\Sigma)$ is less than 1.

Joint Spectral Radius with at most n Consecutive Misses. If the joint spectral radius of the set Σ is less than 1, the stability of all the combinations of realisations (of hits and misses, that include at most n consecutive misses) is proven, regardless of the window size.

For example, let us assume that we are analysing a system with the real-time guarantee that we cannot experience more than two consecutive misses. The realisations that we analyse are $\{A_H, A_H A_M, A_H A_M A_M\}$ and the joint spectral radius unfolds and checks all the possible (infinitely long) sequences of combinations of these realisations.

For a length of two, this means that we check: (1) $A_H A_H$ as the product of the first term twice, (2) $A_H A_H A_M$ as the product of the first two terms picking the first as final (in terms of time evolution of the system), (3) $A_H A_H A_M A_M$ as the product of the first and last terms picking the first as initial, (4) $A_H A_M A_H$ as the product of the first two terms picking the second as final, (5) $A_H A_M A_H A_M$ as the product of the second term twice, (6) $A_H A_M A_H A_M A_M$ as the product of the last two terms, picking the second as final, (7) $A_H A_M A_M A_H$ as the product of the last and first term, (8) $A_H A_M A_M A_H A_M$ as the product of the last and second term, (9) $A_H A_M A_M A_H A_M A_M$ as the last term twice. This procedure is repeated for more products, up to *infinitely long* sequences. From the computation side, the results are an upper and a lower bound on the value of the (true) joint spectral radius.

The analysis is sound on the control side, as stability is guaranteed if and only if the joint spectral radius is less than 1. On the theoretical side, this demonstrates that the $\tau \vdash \overline{\langle n \rangle}$ model can elegantly provide a necessary condition for the stability of the system. The only if part means that there is at least a sequence of hits and misses (where at most we experience n consecutive misses) that causes the system to be unstable if the (true value of the) joint spectral radius is larger than 1.

Considering that we only compute an upper and lower bound on the joint spectral radius, what we can conclude is: if the lower bound that we obtain is above 1, we are entirely certain that such a sequence exists, while if the lower bound is below 1 and the upper bound is above 1 we have no mathematical certainty that the system is unstable. Nonetheless, on the practical side, the bounds obtained with modern approximation techniques [6, 7, 17, 32] are usually very close to one another, implying that they are a very good estimate of the true value of the joint spectral radius.

It is important to note that even if the control system is able to stabilise the system in the presence of n consecutive misses, this does not mean that executing the controller with a period of $n\,\pi$, rather than its original period π, is a sensible choice. In fact, the performance (measured for example using the integral of the squared error) of the controller that is executing with a larger period would be dramatically worse than the performance of the controller with the shorter period that can experience misses. Being able to tolerate misses is very different than performing well when these misses occur.

Relation with m-K model. We here briefly discuss the relation between the guarantees that we obtain with the $\tau \vdash \overline{\langle n \rangle}$ model and the m-K model, $\tau \vdash \overline{\binom{m}{K}}$.

The $\tau \vdash \overline{\langle n \rangle}$ model includes all the realisations that are contained in the $\tau \vdash \overline{\binom{n}{K}}$ regardless of the value of K. However, it can also include additional realisations (that could generate instability) that are not included in the $\tau \vdash \overline{\binom{n}{K}}$ model, if $K > n + 1$. The $\tau \vdash \overline{\langle n \rangle}$ model

over-approximates the set of possible realisations that one can obtain with an m-K task (assuming that $n = m$). This means that there is a chance that an m-K control task stabilises the system when the corresponding task with $n = m$ consecutive deadline misses would not.

The difference between n and K determines the extent of the potential over-approximation. With a smaller difference, the set of possible realisations converges to the set of realisations that are included in the $\tau \vdash \overline{\langle n \rangle}$ model. More precisely, the set of possible realisations considered with the $\tau \vdash \overline{\binom{n}{n+1}}$ model is the same as the set obtained with the $\tau \vdash \overline{\langle n \rangle}$ model. However, increasing K, reduces the set of possibilities that are considered, shrinking the size of the set of valid realisations. Figure 3 shows the relationship between three sets when $K_2 > K_1 > n + 1$.

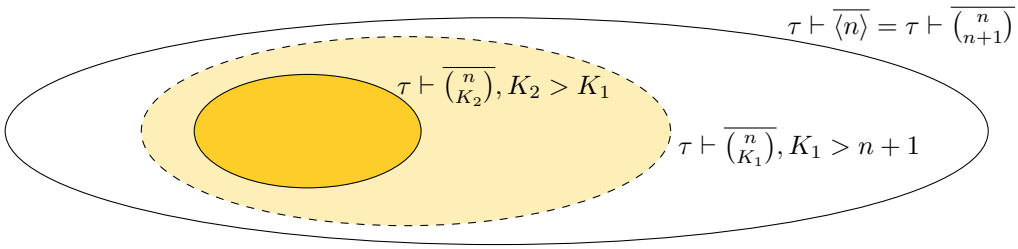

Figure 3 Sets of potential realisations with different models.

If the system with $\tau \vdash \overline{\langle n \rangle}$ is found stable, then control task τ that is given m-K guarantees also stabilises the plant if $m \leq n$. This means that as a first approximation, regardless of the value of K, when dealing with the m-K model, one can check the stability for a maximum of m consecutive deadline misses and if the condition is satisfied, then the closed-loop is stable regardless of the value of K.

As a final remark, from the industrial point of view, the analysis of the case when $K \gg n$ is not particularly interesting, because stability and quality of control guarantees are provided by the design of robust controllers [26] (i.e., a small perturbation in scheduling is anyway covered by the redundancy and design of control systems). On the contrary, there is a clear industrial interest in analysing the $K \approx n + 1$ case, due to transient heavy load perturbations.

Application. We now show how to apply the theory to practical case studies. We implemented our analysis methods in MATLAB®. The input values for our stability verification procedures are: n (the number of contiguous deadline that the system can miss), A_d and B_d (the matrices that determine the dynamics of the system), and K (the controller that is designed and should be validated). We used the JSR Matlab toolbox [22, 45] to compute bounds on the joint spectral radius. We constructed the set Σ based on the expressions derived for the given deadline-handling methods, i.e., for the combination of the control signal management policy (*zero* or *hold*) and the system-level action (*kill*, *skip-next*, or *queue(1)*).

Zero&Kill. The zero and kill strategy is the simplest to analyse. For this strategy we can look at Equations (3) and (5). The system behaves according to

$$\tilde{x}_{[k+1]} = \begin{bmatrix} x_{[k+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & B_d \\ K & 0_{r \times r} \end{bmatrix}}_{A_H} \begin{bmatrix} x_{[k]} \\ u_{[k]} \end{bmatrix} = A_H \, \tilde{x}_{[k]} \tag{11}$$

in case of a hit. Notice that this is in principle exactly the same for each strategy, as when the controller hits the deadline the behaviour is the same. Kill implies that in case of a deadline

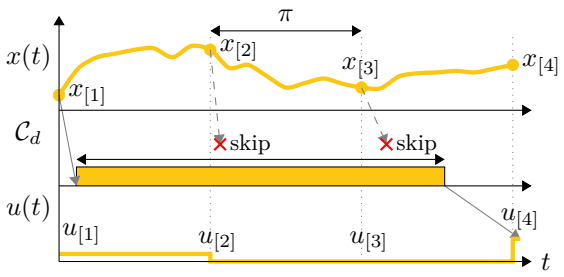

Figure 4 System evolution in case of two consecutive deadline misses with the skip-next strategy and the zero policy.

miss there is an abort of what the control task has been computing up to its deadline, which means there is no need to take into account its (partial) behaviour. In case of a deadline miss in the k-th iteration, the control signal $u_{[k+1]}$ is set to zero. This means that the system evolves according to

$$\tilde{x}_{[k+1]} = \begin{bmatrix} x_{[k+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & B_d \\ 0_{r\times r} & 0_{r\times r} \end{bmatrix}}_{A_M} \begin{bmatrix} x_{[k]} \\ u_{[k]} \end{bmatrix} = A_M\, \tilde{x}_{[k]} \tag{12}$$

in case of a miss. With n maximum consecutive misses, we can then compute all the matrices in $\Sigma = \{A_H A_M^i \mid i \in \mathbb{Z}^{\geq}, i \leq n\}$[8] and then compute the upper bound on $\rho(\Sigma)$.

We would like to remark that while this strategy is simple to analyse, for practical applications it is hard to guarantee – using kill – that the control task τ will miss at most n consecutive deadlines (as the state of the running task is always re-set at every period start).

Zero&Skip-Next. The difference between Zero&Kill and Zero&Skip-Next lays in the freshness of the measurements that are used for the computation of the control signal when the task τ hits its deadline. In fact, if the task was killed and a new job was activated then the state measurement would occur at the beginning of the new activation, while if the task was allowed to continue, it would use old measurements. Figure 4 shows how the system evolves in case of consecutive deadline misses. Suppose that the control task τ completes its execution in the third period. Then this is not equivalent to experiencing two misses and a hit, because the completion uses old state measurements. We need to then express the state matrix evolution when there is a *recovery hit*, rather than a regular hit (in the figure $u_{[4]}$ is set using $x_{[1]}$).

To properly analyse this system, our state has to include the previous values that can be used for the control signal computation. With $\tau \vdash \overline{\langle n \rangle}$ guarantees, we can then define our augmented state vector as $\tilde{x}_{[k]} = [x_{[k]}^{\mathrm{T}}, x_{[k-1]}^{\mathrm{T}}, ..., x_{[k-n]}^{\mathrm{T}}, u_{[k]}^{\mathrm{T}}]^{\mathrm{T}}$, i.e., the state vector of the closed-loop system is composed of $n+1$ elements of the state vector and 1 element for the control signal.

[8] As defined in the introductory part of Section 4 (see point 2 in *Fundamentals*), $\mathbb{Z}^{\geq}$ indicates the set of integers including zero.

Then we can write the closed-loop system in case of a deadline hit, i.e., the equivalent of Equation (7), as

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \dots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & 0_{p\times(n\cdot p)} & B_d \\ & I_{n\cdot p} & 0_{(n\cdot p)\times(p+r)} \\ K & 0_{r\times(n\cdot p)} & 0_{r\times r} \end{bmatrix}}_{A_H} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \dots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix}. \tag{13}$$

Here, we added some padding to the matrix to identify that state variables are transferred from one time instant to the next; i.e., to add the trivial equations $x_{[i]} = x_{[i]}, \forall i \mid x-n+1 \leq i \leq k$. In fact, when the deadline is hit (not after a miss) there is no use of the previous values of the state.

When a miss occurs, the control signal is set to zero. This means that we can use the A_H matrix defined in Equation (13) and substitute the value of K with a zero matrix of appropriate size, i.e., $0_{r\times p}$ to obtain the A_M matrix,

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \dots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & 0_{p\times(n\cdot p)} & B_d \\ & I_{n\cdot p} & 0_{(n\cdot p)\times(p+r)} \\ & 0_{r\times(n+1)\cdot p+r} & \end{bmatrix}}_{A_M} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \dots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix}. \tag{14}$$

Now we should remark that a hit and a *recovery* hit have two different matrix realisation, i.e., a hit that follows a certain number of misses (up to n) has a different matrix with respect to A_H. In fact, we have to take into account the use of the old state measurement to produce the control signal of the recovery hit. We denote with A_{R_i} the matrix that represents the evolution of the closed-loop system when a recovery happens after i deadlines were missed. This matrix can be constructed modifying the last row of A_H, and switching the position of K to use the correct state vector (i.e., the one that corresponds to the measurements obtained n steps before).

For $i = 1$, i.e., with one deadline miss, we can write

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \dots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & 0_{p\times(n\cdot p)} & B_d \\ & I_{n\cdot p} & 0_{(n\cdot p)\times(p+r)} \\ 0_{r\times p} & K \quad 0_{r\times(n-1)\cdot p} & 0_{r\times r} \end{bmatrix}}_{A_{R_1}} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \dots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix}. \tag{15}$$

Consistently with our treatise, $A_{R_0} = A_H$. We can then compute the set Σ as $\Sigma = \{A_{R_i} A_M^i \mid i \in \mathbb{Z}^{\geq}, i \leq n\}$ and then compute the upper bound on $\rho(\Sigma)$ using the computed set of matrices[9].

[9] The drawback of constructing the set Σ as shown above is that the size of the matrices in the set grows linearly with the number of deadline misses. It is possible to construct a compact representation that uses as state vector $\tilde{x}_{[k]} = [x_{[k]}^{\mathrm{T}}, u_{[k]}^{\mathrm{T}}]^{\mathrm{T}}$ but writes the evolution of the system directly as the relation between $\tilde{x}_{[k]}$ and $\tilde{x}_{[k+n+1]}$. This second way of expressing the system dynamics has the disadvantage of hiding misses and hits and only showing the evolution at each hit (still keeping track of what happened at the instants in which the misses occurred). We implemented both versions in our code and checked that the obtained results are the same except for the computational speedup. This remark applies to all the strategies using *skip-next*.

Zero&Queue(1). The behaviour of the combination of the zero policy and the queue(1) strategy vary depending on the possibility of the queued job to complete before the deadline or not. We are going to make the additional hypothesis that the worst-case response time for a job is less than $n\,\pi$, where n is the maximum number of consecutive deadline misses. We can start from the set Σ used for the Zero&Skip-Next combination and add to the set all the matrices $A_H A_M^i$, that take into account the possibility that the queued job completed before its deadline. We should also include in the set Σ the matrices A_{R_i} alone, as it could happen that a queued job doesn't terminate in the period it was started in. We then obtain $\Sigma = \{A_H A_M^i, A_{R_i}, A_{R_i} A_M^i \mid i \in \mathbb{Z}^{\geq}, i \leq n\}$, and we can use the set to compute the upper bound on the joint spectral radius $\rho(\Sigma)$.

Hold&Kill. The hold and kill strategy aborts the task but applies the previously computed control signal to the plant. An easy way to analyse this case is to include in the state of the system also the control signal, such that we can determine the switch between two different matrices without having the matrices grow. We denote with $\tilde{x}_{[k]} = [x_{[k]}^{\mathrm{T}}, u_{[k]}^{\mathrm{T}}]^{\mathrm{T}}$. We recall that r is used to represent the number of input variables.

We can write the evolution of the system when a deadline is hit as

$$\begin{bmatrix} x_{[k+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & B_d \\ K & 0_{r\times r} \end{bmatrix}}_{A_H} \begin{bmatrix} x_{[k]} \\ u_{[k]} \end{bmatrix}, \tag{16}$$

meaning that the computation (the third row of the A_H matrix) is completed and the new control variable is updated with the information from the plant. When a deadline is missed, we compute the system evolution as

$$\begin{bmatrix} x_{[k+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & B_d \\ 0_{r\times r} & I_r \end{bmatrix}}_{A_M} \begin{bmatrix} x_{[k]} \\ u_{[k]} \end{bmatrix}, \tag{17}$$

encoding the hold as the identity matrix that multiplies the old control value for the equation that determines the evolution of $u_{[k]}$. As done for the zero&kill alternative, if we assume there can be a maximum of n consecutive deadline misses, we can then compute all the matrices in $\Sigma = \{A_H A_M^i \mid i \in \mathbb{Z}^{\geq}, i \leq n\}$ and then compute the upper bound on $\rho(\Sigma)$.

Hold&Skip-Next. In order to analyse the combination of hold and skip-next we need to augment the state vector as we did for the zero&skip-next handling strategy. Also in this case, we use our newly defined state vector $\tilde{x}_{[k]} = [x_{[k]}^{\mathrm{T}}, x_{[k-1]}^{\mathrm{T}}, \ldots, x_{[k-n]}^{\mathrm{T}}, u_{[k]}^{\mathrm{T}}]^{\mathrm{T}}$. We obtain the following expression for the closed-loop system when we hit a deadline,

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \cdots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & 0_{p\times(n\cdot p)} & B_d \\ & I_{n\cdot p} & 0_{(n\cdot p)\times(p+r)} \\ K & 0_{r\times(n\cdot p)} & 0_{r\times r} \end{bmatrix}}_{A_H} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \cdots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix}. \tag{18}$$

When we miss a deadline we use the old control value, introducing an identity matrix in the last column and last row of the closed-loop state matrix to indicate that the previous control

signal is saved,

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \dots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & 0_{p\times(n\cdot p)} & B_d \\ & I_{n\cdot p} & 0_{(n\cdot p)\times(p+r)} \\ 0_{r\times(n+1)\cdot p} & & I_r \end{bmatrix}}_{A_M} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \dots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix}. \tag{19}$$

Finally, we should define the behaviour of system in the case of a recovery hit, as done for the zero&skip-next strategy, but including the dynamic evolution of the control signal that has been added to $\tilde{x}$. For one deadline miss we obtain A_{R_1} as

$$\begin{bmatrix} x_{[k+1]} \\ x_{[k]} \\ \dots \\ x_{[k-n+1]} \\ u_{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} A_d & 0_{p\times(n\cdot p)} & B_d \\ & I_{n\cdot p} & 0_{(n\cdot p)\times(p+r)} \\ 0_{r\times p} & K \quad 0_{r\times(n-1)\cdot p} & 0_{r\times r} \end{bmatrix}}_{A_{R_1}} \begin{bmatrix} x_{[k]} \\ x_{[k-1]} \\ \dots \\ x_{[k-n]} \\ u_{[k]} \end{bmatrix}, \tag{20}$$

and the following matrices are obtained by moving the position of the term K in the state evolution matrix to reflect how old is the sensed data that is being used for the computation of the control signal.

Again, $A_{R_0} = A_H$, and we can define the set Σ as $\Sigma = \{A_{R_i} A_M^i \mid i \in \mathbb{Z}^{\geq}, i \leq n\}$. With this, we can compute the upper bound on $\rho(\Sigma)$.

Hold&Queue(1). To analyse the hold&queue(1) strategy, we follow the same principles used for the zero&queue(1) strategy. We start from the hold&skip-next matrices and determine $\Sigma = \{A_H A_M^i, A_{R_i}, A_{R_i} A_M^i \mid i \in \mathbb{Z}^{\geq}, i \leq n\}$.

5 Experimental Validation

In this section we present a few examples of how the analysis presented in Section 4 can be applied to determine the robustness to deadline misses of control system implementations. In particular, we first present some results obtained with an unstable second-order system, which could be used to approximate unstable plants such as a segway that has to be stabilised about the top position. Then, we verify the stability of a permanent magnet synchronous motor for an automotive electric steering application.

Unstable Second-Order System. We analyse the following continuous-time linear time-invariant open-loop system,

$$\dot{x}(t) = \underbrace{\begin{bmatrix} 10 & 0 \\ -2 & -1 \end{bmatrix}}_{A_c} x(t) + \underbrace{\begin{bmatrix} 5 & 1 \\ 4 & 10 \end{bmatrix}}_{B_c} u(t), \tag{21}$$

where both the state and the input vector are composed of two variables. The expression above is the equivalent of Equation (2). Since the A_c matrix is a lower triangular matrix, one can immediately see that the poles of the system are 10 and -1. Since one pole is in the right half plane, the system has one unstable mode and there is a need for control to stabilise the system.

An optimal linear quadratic regulator [26] is designed for this system, assuming that the controller execution is instantaneous and there is no one-step delay actuation, obtaining

$$K = \begin{bmatrix} -4.7393 & 0.2430 \\ 0.2277 & -0.8620 \end{bmatrix}. \tag{22}$$

First, we check the stability of the closed loop system when the controller is executed with one step delay. We select a sampling period of $10\,ms$ and discretise the system obtaining

$$x_{[k+1]} = \underbrace{\begin{bmatrix} 1.1053 & 0.0000 \\ -0.0209 & 0.9900 \end{bmatrix}}_{A_d} x_{[k]} + \underbrace{\begin{bmatrix} 0.0526 & 0.0105 \\ 0.0393 & 0.0994 \end{bmatrix}}_{B_d} u_{[k]}. \tag{23}$$

This corresponds to Equation (3). The matrix A_d is also lower triangular, which means that the poles of the open-loop system are the numbers indicated in the main diagonal. Since one of them is outside the unit circle, the discretised version of the continuous-time system is (unsurprisingly) also unstable and needs control. The poles of the closed-loop system corresponding to the execution of the LET controller every $10\,ms$ are $\{0.8911, 0.8141, 0.3013, 0.0888\}$ and they are all inside the unit circle, meaning that the system is stabilised by the LET controller K from Equation (22), and our control design is a good choice.

Our research question is how many deadlines can we miss when we execute the controller with all the possible deadline miss handling strategies. Table 1 summarises the results we obtain for the analysis.

With the zero strategy, irrespective of the choice of how to handle the job that misses the deadline (kill, skip-next, or queue), the system is robust to missing one deadline (the upper bound on the joint spectral radius is less than 1 for one deadline miss). A subsequent miss is not tolerated, and the closed-loop system becomes provably unstable (the lower bound on the joint spectral radius is above 1). We now look at what happens when the control signal is kept constant in case of misses, i.e., when we hold. Hold&kill ensures that we could miss five deadlines in a row without the emergence of unstable behaviour. However, if a sixth deadline is missed, the system could become unstable. In this case, in fact, the upper-bound on the joint spectral radius exceeds 1. The lower bound, however, is below 1. This means that there is no complete certainty that the system is unstable, but there is a high risk. The true value (for which we have certainty) lies in between the the two bounds. If we continue our analysis, the situation where the true values is around 1 and uncertain persists up to 7 deadline misses. When we introduce the possibility of missing 8 consecutive deadlines, both the lower-bound and the upper-bound are above 1, meaning that the system is provably unstable for some sequences. Notice that this does not mean that the instability is found when we repeat the sequence with 8 consecutive misses followed by a hit. It could happen that the unstable sequence is a combination of a number of deadline misses up to 8 followed by a different number, followed by a number of deadline hits, and so forth. In fact, in our investigation we have encountered cases in which the closed-loop system was stable in case of the repetition of the sequence with n misses followed by a hit, but unstable with a number of consecutive misses up to n. For any practical application, terminating the investigation when the upper-bound crosses 1 ensures a safety margin and guarantees the correct system operation.

When hold is paired with skip-next the system tolerates 2 misses. When queue(1) is uses, on the contrary, the system does not tolerate even a single miss. As a final remark, notice that while the number of tolerated deadline misses is higher for hold&kill, when a task is killed it is difficult to guarantee – with real-time analysis – that the subsequent job will

Table 1 Stability Results for the Unstable Second-Order System.

Misses		Stability	Lower Bound	Upper Bound
Zero&Kill	1	✓	0.961037	0.961975
	2	✗	1.071911	1.071915
Zero&Skip-Next	1	✓	0.914298	0.920769
	2	✗	1.059819	1.059822
Zero&Queue(1)	1	✓	0.961037	0.964287
	2	✗	1.071911	1.071915
Hold&Kill	1	✓	0.891089	0.891090
	2	✓	0.891089	0.891090
	3	✓	0.891089	0.891098
	4	✓	0.891089	0.891251
	5	✓	0.891089	0.935272
	6	✗	0.891089	1.004593
	7	✗	0.961344	1.083038
	8	✗	1.065537	1.172249
Hold&Skip-Next	1	✓	0.891089	0.891090
	2	✓	0.914556	0.944458
	3	✗	1.076507	1.091171
Hold&Queue(1)	1	✗	1.347066	1.370827

not miss its deadline (especially due to locality effects). On the contrary, in general, when skip-next is applied, it is easier to guarantee the termination of in the consecutive periods. This is true unless the deadline misses are caused by a bug in the control task itself, in which case the control task may never terminate.

Electric Steering Application. Here we verify the stability of a permanent magnet synchronous motor for an automotive electric steering application in the presence of deadline misses. We first present a standard model for the motor and a proportional and integral (PI) controller for setpoint tracking, written in the state-feedback form.

When modeling the motor, our system state is $x(t) = [i_d(t), i_q(t)]^{\mathrm{T}}$ where $i_d(t)$ and $i_q(t)$ represent respectively the currents in the d and q coordinates over time. The aim of our control design is to track arbitrary reference values for the currents. The control signal is $u(t) = [u_d(t), u_q(t)]^{\mathrm{T}}$, where $u_d(t)$ and $u_q(t)$ represent respectively the voltages applied to the motor in the d and q coordinate system (subject to an affine transformation). The model of the motor can be written as

$$\dot{x}(t) = \begin{bmatrix} -R/L_d & L_q\,\omega_{el}/L_d \\ -L_d\,\omega_{el}/L_q & -R/L_q \end{bmatrix} x(t) + \begin{bmatrix} 1/L_d & 0 \\ 0 & 1/L_q \end{bmatrix} u(t). \tag{24}$$

Here, L_d [ht] and L_q [ht] denote respectively the inductance in the d and q direction, R [Ohm] is the winding resistance, and ω_{el} [rad/s] is the frequency of the rotor-induced voltage (assumed as constant). We used the parameters of our motor[10] and discretised the plant

[10] The constants used for the calculations are: $R = 0.025$ [Ohm], $\omega_{el} = 6283.2$ [rad/s], $L_d = 0.0001$ [ht], $L_q = 0.00012$ [ht].

using Tustin's method[11], and a sampling period of 10μs, obtaining

$$x_{[k+1]} = \underbrace{\begin{bmatrix} 0.996 & 0.075 \\ -0.052 & 0.996 \end{bmatrix}}_{A_{d,\text{base}}} x_{[k]} + \underbrace{\begin{bmatrix} 0.100 & 0.003 \\ -0.003 & 0.083 \end{bmatrix}}_{B_{d,\text{base}}} u_{[k]}. \tag{25}$$

Notice that the eigenvalues of $A_{d,\text{base}}$ are $0.9957 \pm 0.0626i$ and their absolute value is 0.9977, meaning that the open-loop system is stable (even without control). Control is here added to constrain the behaviour of the system and make sure it tracks current setpoints without errors.

To achieve zero steady-state error, we would like to design our controller in the PI form, using a term that is proportional to the error between the actual current vector and the setpoint vector and a term that is proportional to the integral of the error. We then need to augment the state vector $x_{[k]}$ and keep track of the error at the current time step and at the previous time step. We also need to add to the input vector the setpoints for the currents in the D and Q directions. This allows us to write our PI controller in the state-feedback form.

After this transformation, we denote the new system input as $v_{[k]} = [u_{[k]}^{\mathrm{T}}, w_{[k]}^{\mathrm{T}}]^{\mathrm{T}}$ where $w_{[k]}$ denotes a vector that contains the desired values for the currents i_d and i_q at time k. We also define the new system state as $s_{[k]} = [x_{[k]}^{\mathrm{T}}, e_{[k]}^{\mathrm{T}}, e_{[k-1]}^{\mathrm{T}}]^{\mathrm{T}}$, where $e_{[k]}$ is the error at time k, i.e., $e_{[k]} = w_{[k]} - x_{[k]}$. We therefore model the system as

$$s_{[k+1]} = \underbrace{\begin{bmatrix} A_{d,\text{base}} & 0_{2\times2} & 0_{2\times2} \\ -I_2 & 0_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & I_2 & 0_{2\times2} \end{bmatrix}}_{A_d} s_{[k]} + \underbrace{\begin{bmatrix} B_{d,\text{base}} & 0_{2\times2} \\ 0_{2\times2} & I_2 \\ 0_{2\times2} & 0_{2\times2} \end{bmatrix}}_{B_d} v_{[k]}. \tag{26}$$

We design our PI controller as

$$K = \begin{bmatrix} & \overbrace{\begin{bmatrix} 5 & 0 \\ 1 & 7 \end{bmatrix}}^{K_1} & \overbrace{\begin{bmatrix} -4 & 0 \\ -3 & 7 \end{bmatrix}}^{K_2} \\ 0_{2\times2} & 0_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & 0_{2\times2} & 0_{2\times2} \end{bmatrix}. \tag{27}$$

We can test that in absence of deadline misses, when the controller is implemented with LET (i.e., with one-step delay), the system preserves stability. We now move on to investigate the stability property of the system when some deadlines are missed. Table 2 presents a summary of the results we obtained.

The zero&kill strategy presents a special property. Using this strategy, the closed-loop system is always going to be stable, regardless of the number of deadlines that are missed. In fact, using the joint spectral radius, it is possible to prove that the system that switches between A_H and A_M is always stable, when the matrices are the ones identified in Section 4 for zero&kill. This special property comes from the fact that the open-loop system is stable and control is applied only using fresh measurements (assuming that the kill procedure is able to rollback the system to a clean state). The fact that A_M and A_H are stable individually is not enough to guarantee stability, but all of their combinations prove to be contractions, making the switching system stable as well. Notice that this is not generalisable, but only

[11] While Tustin's method introduces frequency distortion, the method is what is currently applied in our industrial case study. Similar results can be obtained with the exact matrix exponential, changing the discretisation command parameters in the Matlab code.

Table 2 Stability Results for the Electric Steering Application.

Misses		Stability	Lower Bound	Upper Bound
Zero&Kill	∞	✓	0.997713	0.997713
Zero&Skip-Next	1	✓	0.892575	0.892575
	2	✓	0.892575	0.892575
	3	✓	0.892575	0.892575
	4	✓	0.892575	0.892575
	5	✓	0.892575	0.892575
	6	✓	0.892575	0.892575
	7	✓	0.892575	0.892575
	8	✓	0.900922	0.900922
	9	✓	0.912902	0.912902
	10	✓	0.938565	0.938565
	11	✓	0.940823	0.940823
	12	✓	0.942610	0.942610
	13	✓	0.951092	0.951092
	14	✓	0.962846	0.962846
	15	✓	0.973776	0.973776
	16	✓	0.983954	0.983954
	17	✓	0.993436	0.993436
	18	✗	1.002273	1.002273
Zero&Queue(1)	1	✓	0.925966	0.925966
	2	✗	1.001620	1.001620
Hold&Kill	1	✓	0.892575	0.892575
	2	✓	0.938332	0.938332
	3	✗	1.073542	1.073542
Hold&Skip-Next	1	✓	0.968574	0.968574
	2	✗	1.107390	1.107390
Hold&Queue(1)	1	✗	1.423968	1.423968

due to the properties of the particular system we are controlling – in fact, this is not true for the the second-order system example above. For this specific system, this tells us that implementing a clean rollback is very beneficial, and goes a long way to ensure fault tolerance. However, it is not always possible to implement a clean rollback with the given hardware and software setup.

When zero is paired with skip-next, old measurements of the state are used when a recovery hit happens. This means that the system can be unstable even though the behaviour is similar to the zero&kill strategy one. In fact, differently from zero&kill, the system is not able to tolerate an infinite number of misses. We then ask how many many consecutive deadline misses the system experience without violating the stability property. As reported in Table 2, both the lower-bound and the upper-bound on the joint spectral radius are less than 1 for a system that experiences up to 17 consecutive misses. However, with 18 consecutive misses the closed-loop system becomes provably unstable (lower-bound above 1).

This result means that if the controller can miss 18 deadlines in a row, then the delay introduced between the sensing and the actuation is harmful for the system and can cause instability. In this case, there is at least one sequence of deadline misses (which satisfies the

condition that there cannot be more than 18 consecutive misses) that leads to instability. In this case, it is certain that the closed-loop switching system is unstable. Notice that the harmful sequence does not necessarily have to be a repetition of 18 misses and 1 hit, but can be a combination of different terms (for example it could happen that due to the time constant of the system missing 16 deadlines, hitting 2 of them, and then missing 18 would cause instability). For this particular case, however, it is simple enough to check that the spectral radius of the closed-loop matrix with 18 misses and 1 hit is above 1 and this immediately means that the sequence of 18 misses and 1 hit destabilises the system (although it might not be the only one).

If we pair zero with queue(1), the results differ dramatically. The number of matrices in the set Σ used to compute the joint spectral radius increases, and there is a combination of events (specific number of misses, recovery, and recovery with immediate information) that can lead to instability even when just 2 consecutive deadlines are missed. This immediately tells us that using the queue(1) strategy is a bad idea for this system and should be avoided.

When the control signal is held, the kill strategy guarantees stability for 2 deadline misses, while a third potential miss makes the system provably unstable. The skip-next strategy, paired with hold, can tolerate one miss, but a second one makes the system unstable. The queue(1) strategy cannot even tolerate a single miss.

From this experimental campaign, we can conclude that for this system kill is working better than any other system-level strategy, and zero works better than hold in terms of guaranteeing the system stability. We can also conclude that while queuing a task when a previous one is executing could improve the control performance, the risk of harming the system is much higher. We advocate that performing the analysis presented in this paper can give important runtime information to determine how robust control systems are to temporary faults and problems.

6 Related Work

Studying the non-ideal behaviour that emerges from the implementation and execution of control systems is an important problem for practical applications of control. Control tasks for example may have variable periods and may require to be executed with different rates depending on the operating conditions [4]. Also, late information can affect the system's performance [24, 30, 31, 37], especially for networked systems. These timing effects are usually characterised as independent events with stochastic distributions [13], or using worst case bounds [2, 15]. Control researchers proposed deadline-aware design methods to guarantee stability [5, 29] and improve the control performance [38]. Sinopoli et al. [40] proposed an optimal control design for networked system leveraging the probability of packet losses. Lincoln and Cervin [28] proposed a design tool for optimal controllers in the presence of probabilistic delays, that can be exploited for LET design setting the delay to one period. In many circumstances, the control designer can usually trade off computing time and accuracy [39]. In some cases, an inaccurate and faster solution that can be executed at a faster rate is preferable to an accurate and precise solution that can only be executed at a slower rate [23]. However, it is extremely hard in practice to guarantee that the delays will *never* exceed the control period.

These types of systems have been studied both from the schedulability perspective [43] and using model checking [12]. The performance cost of deadline misses was investigated [34, 44], together with the role of the strategy used to handle the misses [33, 41, 42]. All these papers used the m-K model, starting from the assumption that windows of hits and misses have to

be analysed in order to determine the behaviour of the system. We build on the previous literature to determine how the implementation is going to react to missed deadlines, both in terms of selection of the control signal [29] and in terms of management of the job that misses its deadline [33]. There are some important differences between [33] and the contribution of this paper: [33] presents a control design technique to guarantee probabilistic robustness to deadline misses, on the contrary we assume that the controller is already designed and executes without any change and we demonstrate an analysis method to guarantee exactly that the controller tolerates (in terms of stability) a maximum number of consecutive misses.

In this work we showed that it may not be necessary to study windows of hits and misses. We argue that before looking at the m-K model representation, which is harder to analyse, one should check the stability of the $\overline{\langle m \rangle}$ model. If the $\overline{\langle m \rangle}$ is stable, there is no need to include complex window-based analysis – it happens quite often that the information needed to study the stability of the closed-loop systems is already contained in the number of consecutive deadline misses.

Ghosh et. al. [16] studied how to design control systems in the presence of faults that cause them to miss at most n deadlines, providing a synthesis method to achieve fault-tolerance – without specifying how the system is going to react to the misses (e.g., kill or skip-next). Here we take a different perspective and want to validate the behaviour of a control system (in terms of stability) when deadlines are missed, including the deadline management strategy from a system and implementation perspective.

7 Conclusion

In this paper we revisited the weakly hard real-time model for control tasks. We formalised the problem of determining the stability of the closed-loop system in the presence of a given number of consecutive misses that the controller task can experience. With the number of consecutive deadline misses, we derived stability criteria for systems where the deadline miss is handled in different ways, both from the perspective of the control signal applied (either zeroing or holding the previous value of the control signal) and with respect to the management of the task that misses the deadline (that can be either killed or allowed to continue in the subsequent control period). We solved this problem using a mathematical tool called joint spectral radius, for the computation of which open-source toolboxes are available. We applied our analysis to two different examples: an unstable system and an industrial application for electric steering. In both cases, we showed the limitations of the controller implementations that miss a given number of deadlines.

In the future, we plan to look at applying the joint spectral radius analysis to systems specified using the m-K model. This is particularly challenging, because there is no direct mapping between the potential sequences of hits and misses and a set of matrices. Furthermore, we want to investigate the performance of controllers that experience deadline misses.

References

1 Leonie Ahrendts, Sophie Quinton, Thomas Boroske, and Rolf Ernst. Verifying Weakly-Hard Real-Time Properties of Traffic Streams in Switched Networks. In *30th Euromicro Conference on Real-Time Systems (ECRTS 2018)*, volume 106 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:22. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ECRTS.2018.15.

2 Philip Axer, Maurice Sebastian, and Rolf Ernst. Probabilistic response time bound for CAN messages with arbitrary deadlines. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, pages 1114–1117. IEEE, 2012.

3 G. Bernat, A. Burns, and A. Liamosí. Weakly hard real-time systems. *IEEE Transactions on Computers*, 50(4):308–321, April 2001. doi:10.1109/12.919277.

4 Alessandro Biondi, Marco Di Natale, Giorgio C. Buttazzo, and Paolo Pazzaglia. Selecting the transition speeds of engine control tasks to optimize the performance. *ACM Transaction on Cyber-Physical Systems*, 2(1):1:1–1:26, January 2018. doi:10.1145/3127022.

5 R. Blind and F. Allgöwer. Towards networked control systems with guaranteed stability: Using weakly hard real-time constraints to model the loss process. In *54th IEEE Conference on Decision and Control (CDC)*, pages 7510–7515, December 2015. doi:10.1109/CDC.2015.7403405.

6 V. Blondel and Y. Nesterov. Computationally efficient approximations of the joint spectral radius. *SIAM Journal on Matrix Analysis and Applications*, 27(1):256–272, 2005. doi:10.1137/040607009.

7 Vincent Blondel, Yurii Nesterov, and Jacques Theys. On the accuracy of the ellipsoid norm approximation of the joint spectral radius. *Linear Algebra and its Applications*, 394:91–107, 2005. doi:10.1016/j.laa.2004.06.024.

8 Vincent Blondel and John N. Tsitsiklis. The boundedness of all products of a pair of matrices is undecidable. *Systems & Control Letters*, 41(2):135–140, 2000. doi:10.1016/S0167-6911(00)00049-9.

9 Tobias Bund and Frank Slomka. Controller/platform co-design of networked control systems based on density functions. In *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, CyPhy '14, pages 11–14. ACM, 2014. doi:10.1145/2593458.2593467.

10 Tobias Bund and Frank Slomka. Worst-case performance validation of safety-critical control systems with dropped samples. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, RTNS '15, pages 319–326. ACM, 2015. doi:10.1145/2834848.2834860.

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

12 G. Frehse, A. Hamann, S. Quinton, and M. Woehrle. Formal analysis of timing effects on closed-loop properties of control software. In *2014 IEEE Real-Time Systems Symposium*, pages 53–62, December 2014. doi:10.1109/RTSS.2014.28.

13 Maximilian Gaukler, Andreas Michalka, Peter Ulbrich, and Tobias Klaus. A new perspective on quality evaluation for control systems with stochastic timing. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (Part of CPS Week)*, HSCC, pages 91–100. ACM, 2018. doi:10.1145/3178126.3178134.

14 Maximilian Gaukler, Tim Rheinfels, Peter Ulbrich, and Günter Roppenecker. Convergence rate abstractions for weakly-hard real-time control. *arXiv preprint arXiv:1912.09871*, 2019.

15 Maximilian Gaukler and Peter Ulbrich. Worst-case analysis of digital control loops with uncertain input/output timing. In Goran Frehse and Matthias Althoff, editors, *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 61 of *EPiC Series in Computing*, pages 183–200. EasyChair, 2019. doi:10.29007/c4zl.

16 Saurav Kumar Ghosh, Soumyajit Dey, Dip Goswami, Daniel Mueller-Gritschneder, and Samarjit Chakraborty. Design and validation of fault-tolerant embedded controllers. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1283–1288, 2018. doi:10.23919/DATE.2018.8342212.

17 N. Guglielmi, F. Wirth, and M. Zennaro. Complex polytope extremality results for families of matrices. *SIAM Journal on Matrix Analysis and Applications*, 27(3):721–743, 2005. doi:10.1137/040606818.

18 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, December 1995. doi:10.1109/12.477249.

19 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), 2017*, pages 584–589, March 2017. doi:10.23919/DATE.2017.7927054.

20 Zain A. H. Hammadeh, Sophie Quinton, Marco Panunzio, Rafik Henia, Laurent Rioux, and Rolf Ernst. Budgeting Under-Specified Tasks for Weakly-Hard Real-Time Systems. In *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*, volume 76 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:22. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ECRTS.2017.17.

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

22 Raphael Jungers. JSR Toolbox. https://www.mathworks.com/matlabcentral/fileexchange/33202-the-jsr-toolbox, accessed October 23, 2019.

23 E. C. Kerrigan, G. A. Constantinides, A. Suardi, A. Picciau, and B. Khusainov. Computer architectures to close the loop in real-time optimization. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 4597–4611, December 2015. doi:10.1109/CDC.2015.7402937.

24 Antzela Kosta, Nikolaos Pappas, Anthony Ephremides, and Vangelis Angelakis. Age and value of information: Non-linear age case. In *Information Theory (ISIT), 2017 IEEE International Symposium on*, pages 326–330. IEEE, 2017.

25 Huibert Kwakernaak. *Linear Optimal Control Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1972.

26 W.S. Levine. *The Control Handbook*. Electrical Engineering Handbook. Taylor & Francis, 1996.

27 D. Liberzon. *Switching in Systems and Control*. Systems & Control: Foundations & Applications. Birkhäuser Boston, 2003.

28 B. Lincoln and A. Cervin. JITTERBUG: a tool for analysis of real-time control performance. In *41st IEEE Conference on Decision and Control*, volume 2, pages 1319–1324, December 2002. doi:10.1109/CDC.2002.1184698.

29 S. Linsenmayer and F. Allgower. Stabilization of networked control systems with weakly hard real-time dropout description. In *IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4765–4770, December 2017. doi:10.1109/CDC.2017.8264364.

30 L. Palopoli, L. Abeni, G. Buttazzo, F. Conticelli, and M. Di Natale. Real-time control system analysis: an integrated approach. In *Real-Time Systems Symposium, 2000. Proceedings. The 21st IEEE*, pages 131–140, 2000. doi:10.1109/REAL.2000.896003.

31 P. Park, S. Coleri Ergen, C. Fischione, C. Lu, and K. H. Johansson. Wireless network design for control systems: A survey. *IEEE Communications Surveys Tutorials*, 20(2):978–1013, 2018. doi:10.1109/COMST.2017.2780114.

32 Pablo A. Parrilo and Ali Jadbabaie. Approximation of the joint spectral radius using sum of squares. *Linear Algebra and its Applications*, 428(10):2385–2402, 2008. doi:10.1016/j.laa.2007.12.027.

33 Paolo Pazzaglia, Claudio Mandrioli, Martina Maggio, and Anton Cervin. DMAC: Deadline-Miss-Aware Control. In *31st Euromicro Conference on Real-Time Systems (ECRTS 2019)*, volume 133 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:24, 2019. doi:10.4230/LIPIcs.ECRTS.2019.1.

34 Paolo Pazzaglia, Luigi Pannocchi, Alessandro Biondi, and Marco Di Natale. Beyond the Weakly Hard Model: Measuring the Performance Cost of Deadline Misses. In *30th Euromicro Conference on Real-Time Systems (ECRTS 2018)*, volume 106 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:22, 2018. doi:10.4230/LIPIcs.ECRTS.2018.10.

35 Gang Quan and Xiaobo Hu. Enhanced fixed-priority scheduling with (m,k)-firm guarantee. In *Proceedings of the 21st IEEE Conference on Real-time Systems Symposium*, RTSS, pages 79–88. IEEE Computer Society, 2000.

36 Gian–Carlo Rota and W. Gilbert Strang. A note on the joint spectral radius. *Indagationes Mathematicae*, 63:379–381, 1960. doi:10.1016/S1385-7258(60)50046-1.

37 Abusayeed Saifullah, Chengjie Wu, Paras Babu Tiwari, You Xu, Yong Fu, Chenyang Lu, and Yixin Chen. Near optimal rate selection for wireless control systems. *ACM Transactions on Embedded Computing Systems*, 13(4s):128:1–128:25, April 2014. doi:10.1145/2584652.

38 Luca Schenato, Bruno Sinopoli, Massimo Franceschetti, Kameshwar Poolla, and S Shankar Sastry. Foundations of control and estimation over lossy networks. *Proceedings of the IEEE*, 95(1):163–187, 2007.

39 Amir Shahzad, Eric C. Kerrigan, and George A. Constantinides. A stable and efficient method for solving a convex quadratic program with application to optimal control. *SIAM Journal on Optimization*, 22(4):1369–1393, 2012. doi:10.1137/11082960X.

40 Bruno Sinopoli, Luca Schenato, Massimo Franceschetti, Kameshwar Poolla, and Shankar Sastry. An lqg optimal linear controller for control systems with packet losses. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 458–463. IEEE, 2005.

41 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, March 2018. doi:10.1109/TCNS.2016.2583064.

42 Damoon Soudbakhsh, Linh T. X. Phan, Oleg Sokolsky, Insup Lee, and Anuradha Annaswamy. Co-design of control and platform with dropped signals. In *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems*, ICCPS '13, pages 129–140. ACM, 2013. doi:10.1145/2502524.2502542.

43 Youcheng Sun and Marco Di Natale. Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks. *ACM Transactions on Embedded Computing Systems*, 16(5s):171:1–171:19, September 2017. doi:10.1145/3126497.

44 E. P. van Horssen, A. R. B. Behrouzian, D. Goswami, D. Antunes, T. Basten, and W. P. M. H. Heemels. Performance analysis and controller improvement for linear systems with (m, k)-firm data losses. In *2016 European Control Conference (ECC)*, pages 2571–2577, June 2016. doi:10.1109/ECC.2016.7810677.

45 Guillaume Vankeerberghen, Julien Hendrickx, and Raphaël 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, pages 151–156. ACM, 2014. doi:10.1145/2562059.2562124.

46 Dirk Ziegenbein and Arne Hamann. Timing-aware control software design for automotive systems. In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, pages 56:1–56:6, 2015. doi:10.1145/2744769.2747947.