Stability of Linear Systems Under Extended Weakly-Hard Constraints

Nils Vreman, Paolo Pazzaglia, Victor Magron, Jie Wang, and Martina Maggio, *Senior Member, IEEE*

***Abstract*—Control systems can show robustness to many events, like disturbances and model inaccuracies. It is natural to speculate that they are also robust to sporadic deadline misses when implemented as digital tasks on an embedded platform. This letter proposes a comprehensive stability analysis for control systems subject to deadline misses, leveraging a new formulation to describe the patterns experienced by the control task under different handling strategies. Such analysis brings the assessment of control systems robustness to computational problems one step closer to the controller implementation.**

***Index Terms*—Fault tolerant systems, linear systems, networked control systems.**

I. Introduction

ROBUSTNESS is an essential concern in the design of control systems; they must be able to reliably handle nonlinear effects, unmodeled dynamics and noise, as well as delays in signal transmissions and dropped packets. A lesser known problem concerns the assessment of robustness to *computational issues* when controllers are implemented as periodic tasks in cheap embedded platforms. Such tasks are expected to execute with real-time guarantees, i.e., their execution must be completed before a well-defined *deadline*, when the control output must be sent to the actuator. However, it is common in practice [1] that tasks do not always complete within their deadline, causing what is called a *deadline miss*. This may be caused by delays in computation and memory accesses, transient overloads, bugs and other issues.

Manuscript received March 3, 2022; revised May 3, 2022; accepted May 23, 2022. Date of publication June 2, 2022; date of current version June 10, 2022. This work was supported by the European Union's Horizon 2020 Research and Innovation Programme (ADMORPH Project) under Grant 871259. 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. Recommended by Senior Editor F. Dabbene. *(Corresponding author: Martina Maggio.)*

Nils Vreman is with the Department of Automatic Control, Lund University, 223 63 Lund, Sweden.

Paolo Pazzaglia is with the Department of Computer Science, Saarland University, 66123 Saarbrücken, Germany.

Victor Magron is with the Laboratory for Analysis and Architecture of Systems, CNRS, 31031 Toulouse, France.

Jie Wang is with the Academy of Mathematics and Systems Science, CAS, Beijing 100190, China.

Martina Maggio is with the Department of Computer Science, Saarland University, 66123 Saarbrücken, Germany, and also with the Department of Automatic Control, Lund University, 223 63 Lund, Sweden (e-mail: martina@control.lth.se).

Digital Object Identifier 10.1109/LCSYS.2022.3179960

A popular model to describe real-time systems allowing deadline misses is the *weakly-hard* model [2]. Weakly-hard tasks feature constraints defining a maximum number of deadlines that can be missed (alternatively, a minimum number to be satisfied) in a given number of consecutive periods. This model is also the focus of this letter. To analyse the effects on the controlled plant, it is necessary to specify also *what happens when the miss is experienced*, both in terms of changes to the control signal and of actions taken to deal with the failed computation [19]. An instance that experiences a deadline miss can be allowed to continue executing until completion (and possibly used later), while in other applications it is stopped and discarded instead.

There is however a mismatch between the guarantees that can be obtained for real-time tasks and platforms [4], [28], and the analysis available for *control* tasks under the weakly-hard model. Fewer works deal with *stability* analysis of weakly-hard real-time control tasks, often targeting specific use-cases. For instance, the analysis in [16] is limited to constraints specifying a maximum number of *consecutive* deadline misses. The results in [14], [15], obtained for networked linear control systems having packet dropouts bounded using the weakly-hard model, can not be generalised for *late completions* or *sets* of weakly-hard constraints. The authors of [12], [13] studied safety guarantees of weakly-hard controllers, considering a miss as a discarded computation with a known periodic pattern. In [8], [9], an over-approximation-based approach is proposed to check the safety of nonlinear weakly-hard systems, where misses are treated as discarded computations and the actuator holds its previous value. Convergence rates (providing sufficient stability guarantees) are analysed in [6]. A Lyapunov-based stability analysis of nonlinear weakly-hard systems is studied in [7], with deadline misses treated as packet dropouts. However, the state-of-the-art listed above lack generalisability to more expressive real-time implementations, such as different deadline miss models or handling strategies.

This letter aims at filling the gap, by providing a stability analysis that can be applied to a class of generic weakly-hard models and deadline miss handling strategies. First, we formally extend the weakly-hard model to explicitly consider the strategy used to handle the miss events. By leveraging an automaton representation of the sequences allowed by (a set of) extended weakly-hard constraints, we use Kronecker lifting and the joint spectral radius to properly express its stability

conditions. Using the concept of constraint dominance, we prove analytic bounds on the stability of a weakly-hard system with respect to *less dominant* constraints. Finally, we analyse the stability of the resulting closed-loop systems using `SparseJSR` [26], which exploits the sparsity pattern that naturally arises in the Kronecker lifted representation. The proposed analysis calls for modularity and separation of concern, and can be a useful tool to decouple the constraint specification and the control verification.

II. Background and Notation

We consider a controllable and fully observable *discrete-time* sampled linear time invariant system, expressed as

$$P: \quad \begin{cases} x_{t+1} = A_p\, x_t + B_p\, u_t \\ y_t = C_p\, x_t + D_p\, u_t, \end{cases} \tag{1}$$

where $x_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}^r$ and $y_t \in \mathbb{R}^q$ are the plant state, the control signal and the plant output, sampled at time $t \cdot T$, T is the sampling period, and $t \in \mathbb{N}$. The plant is controlled by a stabilising, LTI, one-step delay discrete-time controller

$$C: \quad \begin{cases} z_{t+1} = A_c\, z_t + B_c\, (r_t - y_t) \\ u_{t+1} = C_c\, z_t + D_c\, (r_t - y_t), \end{cases} \tag{2}$$

where $z_t \in \mathbb{R}^s$ is the controller's internal state and $r_t \in \mathbb{R}^q$ is the setpoint. Without loss of generality, we consider $r_t = 0$.

A. Real-Time Tasks That May Miss Deadlines

The controller in (2) is implemented as a real-time task τ, and designed to be executed periodically with period T in a real-time embedded platform. Under nominal conditions the task releases an instance (called *job*) in each period, that should be completed before the release of the next instance. We denote the sequence of activation instants for τ with $(a_i)_{i\in\mathbb{N}}$, such that, in nominal conditions, $a_{i+1} = a_i + T$, the sequence of completion instants $(f_i)_{i\in\mathbb{N}}$, and the sequence of job deadlines with $(d_i)_{i\in\mathbb{N}}$, such that $d_i = a_i + T$ (also called *implicit* deadline). This requirement can be either satisfied or not, leading respectively to deadline hits and misses.

Definition 1 (Deadline Hit and Miss): The i-th job of a periodic task τ with period T hits its deadline when $f_i \leq d_i$ and misses its deadline when $f_i > d_i$.

We refer to both deadline hits and misses using the term *outcome* of a job. Intuitively, each job's outcome is dependent on the characteristics of the remaining tasks executing in the real-time system and the chosen scheduling algorithm. Given a taskset and a (worst-case) schedule, it is possible to bound the worst-case behavior of the job outcomes [2], [28]. This bound is generally denoted using the *weakly-hard model* [2]. Following such model, a task τ may satisfy any combination of these weakly-hard constraints, defined as follows. (i) $\tau \vdash \overline{\binom{m}{k}}$: in any window of k consecutive jobs, at most m deadlines are missed; (ii) $\tau \vdash \binom{h}{k}$: in any window of k consecutive jobs, at least h deadlines are hit; (iii) $\tau \vdash \overline{\langle\!\binom{m}{k}\!\rangle}$: in any window of k consecutive jobs, at most m *consecutive* deadlines are missed; and (iv) $\tau \vdash \langle\!\binom{h}{k}\!\rangle$: in any window of k consecutive jobs, at least h *consecutive* deadlines are hit.

In all such cases, $m, h \in \mathbb{N}$, $k \in \mathbb{N} \setminus \{0\}$, $m \leq k$, and $h \leq k$. A generic weakly-hard constraint is hereafter denoted with the symbol λ, while a set of L constraints will be referred to as $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_L\}$.

We define a *string* $\omega = \langle \alpha_1, \alpha_2, \ldots, \alpha_N \rangle$ as a sequence of N consecutive outcomes, where each outcome α_i is a character in the alphabet $\Sigma = \{\mathtt{M}, \mathtt{H}\}$. We use $\omega \vdash \lambda$ to denote that ω satisfies the constraint λ. Stating that $\tau \vdash \lambda$ means that all the possible sequences of outcomes that τ can experience satisfy the corresponding constraint λ. The set of such sequences naturally results from the definition of λ, and is formally defined as the *satisfaction set* as follows [2].

Definition 2 [Satisfaction Set $\mathcal{S}_N(\lambda)$]: We denote with $\mathcal{S}_N(\lambda)$ the set of strings of length $N \geq 1$ that satisfy a constraint λ. Formally, $\mathcal{S}_N(\lambda) = \{\omega \in \Sigma^N \mid \omega \vdash \lambda\}$.

Taking the limit to infinity, the set $\mathcal{S}(\lambda)$ contains all the strings of infinite length that satisfy λ. The notion of *domination* between constraints [2] then follows.

Definition 3 (Constraint Domination): Constraint λ_i *dominates* λ_j (formally, $\lambda_i \preceq \lambda_j$) if $\mathcal{S}(\lambda_i) \subseteq \mathcal{S}(\lambda_j)$.

B. Control Tasks That May Miss Deadlines

When a control task τ is implemented on an embedded platform with limited computational power, alongside other applications, it is not uncommon for it to experience deadline misses, even in case of simple control designs (PID, LQG, etc) [1], [18]. Computational overruns may be caused by, e.g., bursts of interrupts, cache misses, variable execution times of ancillary functions, or other complex interactions. If such events are rare or temporary, choosing a longer period for the controller to avoid them may result in worse performance and stability margins for nominal conditions [19].

Characterising the stability and performance of such controllers requires knowing what happens when a control deadline is missed [16], [19], [24]. In particular, we need a *deadline miss handling strategy* to decide the fate of the job that missed the deadline (and possibly the next ones), and an *actuator mode* to deal with the loss of a new control signal, for example by holding the previous value constant or zeroing it [22]. A few handling strategies for periodic controllers have been proposed in literature, the most interesting being *Kill* and *Skip-Next* [3], [16], [19].

Definition 4 (Kill Strategy): Under the Kill strategy, a job that misses its deadline is terminated immediately. Formally, for the i-th job of τ either $f_i \leq d_i$ or $f_i = \infty$.

Definition 5 (Skip-Next Strategy): Under the Skip-Next strategy, a job that misses its deadline is allowed to continue during the following period. Formally, if the i-th job of τ misses its deadline d_i, a new deadline $d_i^+ = d_i + T$ is set for the job, and $a_{i+1} = d_i^+$.

C. Stability Analysis Techniques Based on JSR

In [16], the authors identify a set of subsequences of hit and missed deadlines, which can be arbitrarily combined to obtain all possible sequences in $\mathcal{S}(\overline{\langle\!\binom{m}{k}\!\rangle})$. The stability analysis of the resulting arbitrary switching system is then obtained by leveraging the *Joint Spectral Radius* (JSR) [21].

Given $\ell \in \mathbb{N} \setminus \{0\}$ and a set of matrices $\mathcal{A} = \{A_1, \ldots, A_\ell\} \subseteq \mathbb{R}^{n\times n}$, under the hypothesis of arbitrary switching over any

sequence $s = \langle a_1, a_2, \dots \rangle$ of indices of matrices in $\mathcal{A}$, the JSR of $\mathcal{A}$ is defined by:

$$\rho(\mathcal{A}) = \lim_{k\to\infty} \max_{s\in\{1,\dots,\ell\}^k} \|A_{a_k} \cdots A_{a_2} A_{a_1}\|^{\frac{1}{k}}. \tag{3}$$

The number $\rho(\mathcal{A})$ characterizes the maximal asymptotic growth rate of matrix products from $\mathcal{A}$ (thus $\rho(\mathcal{A}) < 1$ means that the system is asymptotically stable), and is independent of the norm $\|\cdot\|$ used in (3). Existing practical tools such as the JSR MATLAB toolbox [23] include multiple algorithms to compute both upper and lower bounds on $\rho(\mathcal{A})$.

When the switching sequences between the dynamics of $\mathcal{A}$ are not arbitrary, but constrained by a graph $\mathcal{G}$, the so called *constrained joint spectral radius* (CJSR) [5] can be applied. Introducing $S_k(\mathcal{G})$ as the set of all possible switching sequences s of length k that satisfy the constraints of a graph $\mathcal{G}$, the CJSR of $\mathcal{A}$ is defined by

$$\rho(\mathcal{A}, \mathcal{G}) = \lim_{k\to\infty} \max_{s\in S_k(\mathcal{G})} \|A_{a_k} \cdots A_{a_2} A_{a_1}\|^{\frac{1}{k}}. \tag{4}$$

In general, computing or approximating the CJSR is harder than using the JSR. In [20], the authors propose a multinorm-based method to approximate with arbitrary accuracy the CJSR. Other works [10], [29] propose the creation of an arbitrary switching system such that its JSR is equal to the CJSR of the original system, based on a Kronecker lifting method. This will be also our approach, as detailed later.

In [17], the authors propose an efficient approach to compute upper bounds of the JSR based on positive polynomials which can be decomposed as *sums of squares* (SOS). Finding the coefficients of a polynomial being SOS simplifies to solving an SDP [11]. To reduce time and space complexity, a *sparse* variant has been proposed in [26] exploiting the sparsity of the input matrices, based on the *term sparsity* SOS (TSSOS) framework [27]. By contrast, the procedure in [17] will be denoted hereafter as *dense*. While providing a more conservative result, the sparse upper bound can be obtained significantly faster if the matrices from $\mathcal{A}$ are sparse [26], e.g., the matrices we analyse in Section IV.

III. Extended Weakly-Hard Task Model

To provide a comprehensive analysis framework, we need to examine what occurs in each time interval $(\pi_i)_{i\in\mathbb{N}}$, with $\pi_i = [a_0 + i \cdot T, a_0 + (i+1) \cdot T)$. In this context, an extension of the weakly-hard model is required to account for the given deadline miss handling strategy, denoted with the symbol $\mathcal{H}$.

Definition 6 (Extended Weakly-Hard Model $\tau \vdash \lambda^{\mathcal{H}}$): A task τ may satisfy any combination of the four *extended weakly-hard constraints* (EWHC) $\lambda^{\mathcal{H}}$:

(i) $\tau \vdash \overline{\binom{m}{k}}^{\mathcal{H}}$: in any window of k consecutive jobs, at most m intervals lack a job completion;
(ii) $\tau \vdash \binom{h}{k}^{\mathcal{H}}$: in any window of k consecutive jobs, at least h intervals have a job completion;
(iii) $\tau \vdash \overline{\langle\!\binom{m}{k}\!\rangle}^{\mathcal{H}}$: in any window of k consecutive jobs, at most m *consecutive* intervals lack a job completion;
(iv) $\tau \vdash \langle\!\binom{h}{k}\!\rangle^{\mathcal{H}}$: in any window of k consecutive jobs, at least h *consecutive* intervals have a job completion

with $m, h \in \mathbb{N}$, $k \in \mathbb{N} \setminus \{0\}$, $m \leq k$, and $h \leq k$, while using strategy $\mathcal{H}$ to handle potential deadline misses.

The definition above differs from the original weakly-hard model of [2], since (i) it explicitly introduces the handling strategy $\mathcal{H}$; and (ii) it focuses on the presence of a new control command at the end of each time interval π_i, instead of checking the deadline miss events, which guarantees its applicability also for strategies different than Kill.

We now require an expressive alphabet $\Sigma(\mathcal{H})$ to characterize the behaviour of task τ in each possible time interval. For both Kill and Skip-Next strategies, each interval π_i contains at most one activated and one completed job. This restricts the possible behaviours to three cases:

(i) a time interval in which the same job is both released and completed is denoted by `H` (*hit*);
(ii) a time interval in which no job is completed is denoted by `M` (*miss*);
(iii) a time interval in which no job is released, but a job (released in a previous interval) is completed, is denoted by `R` (*recovery*).

By checking all unique combinations of job activations and completions in each interval, we obtain the alphabets for Kill and Skip-Next as $\Sigma(\text{Kill}) = \{\texttt{M}, \texttt{H}\}$ and $\Sigma(\text{Skip-Next}) = \{\texttt{M}, \texttt{H}, \texttt{R}\}$, respectively. The recovery character `R` is used in the Skip-Next alphabet to identify the late *completion* of a job. As a consequence, `R` is treated equivalently to `H` when verifying the extended weakly hard constraints (EWHC).

The algebra presented in Section II-A is extended to the new alphabet. We assign a character of the alphabet $\Sigma(\mathcal{H})$ to each interval π_i. A string $\omega = \langle \alpha_1, \alpha_2, \dots, \alpha_N \rangle$ is used to represent a sequence of N outcomes for task τ, with $\alpha_i \in \Sigma(\mathcal{H})$ representing the outcome associated to the interval π_i. To enforce only feasible sequences, we introduce an order constraint for the `R` character with the following Rule.

Rule 1: For any string $\omega \in \Sigma(\text{Skip-Next})^N$, `R` may only directly follow `M`, or be the initial element of the string.

The extended weakly-hard model also inherits all the properties of the original weakly-hard model. In particular, the satisfaction set of $\lambda^{\mathcal{H}}$ can be defined for $N \geq 1$ as $\mathcal{S}_N(\lambda^{\mathcal{H}}) = \{\omega \in \Sigma(\mathcal{H})^N \mid \omega \vdash \lambda^{\mathcal{H}}\}$, and the constraint domination still holds as $\lambda_i^{\mathcal{H}} \preceq \lambda_j^{\mathcal{H}}$ if $\mathcal{S}(\lambda_i^{\mathcal{H}}) \subseteq \mathcal{S}(\lambda_j^{\mathcal{H}})$.

A. Automaton Representation of EWHC

Any EWHC, as presented in Definition 6, can be systematically represented using an *automaton*. In this letter we build upon the `WeaklyHard.jl` automaton model presented in [25]. Here, a (minimal) automaton $\mathcal{G}_{\lambda^{\mathcal{H}}} = (V_{\lambda^{\mathcal{H}}}, E_{\lambda^{\mathcal{H}}})$ associated to $\lambda^{\mathcal{H}}$ consists of a set of vertices $(V_{\lambda^{\mathcal{H}}})$ and a set of directed labeled edges $(E_{\lambda^{\mathcal{H}}})$. Each vertex $v_i \in V_{\lambda^{\mathcal{H}}}$ corresponds to a string of outcomes of the extended weakly-hard task executions. Trivially, there exists no vertices for strings that do not satisfy the EWHC. A directed labeled edge $e_{i,j} = (v_i, v_j, \alpha) \in E_{\lambda^{\mathcal{H}}}$ (also denoted *transition*) connects two vertices iff the outcome $\alpha \in \Sigma(\mathcal{H})$ – the edge's label – appended to the tail vertex's string representation (v_i) would result in the string equivalent to the one of the head vertex (v_j). Thus, a random walk in the automaton corresponds to

a random string satisfying the EWHC. In particular, all the walks in the automaton corresponds to *all* strings in $\mathcal{S}(\lambda^{\mathcal{H}})$.

Since the `WeaklyHard.jl` automaton model only uses the binary alphabet $\Sigma = \{\mathtt{M}, \mathtt{H}\}$, we require the additional character `R` to handle the Skip-Next strategy properly. Recall that both a hit (`H`) and a recovery (`R`) are considered job completions. Thus, for the Skip-Next strategy, we post-process the automaton by enforcing that Rule 1 is honoured and that the corresponding transitions are correct, i.e., switching the labels on some edges from `H` to `R`. We emphasise that despite the extended automaton model appear similar for the Kill and Skip-Next strategies, the differing transitions of the two automata significantly affect the corresponding closed-loop systems, as will be clear in Section IV.

The `WeaklyHard.jl` automaton model also allows for the case where the task τ is subject to a set of multiple constraints. Since the stability analysis presented in this letter is invariant to the type (and amount) of the constraints acting on the control task τ, we henceforth say that τ is subject to a set of EWHC $\Lambda^{\mathcal{H}}$ (unless stated otherwise).

Extracting all transitions in $E_{\Lambda^{\mathcal{H}}}$ corresponding to a character $\alpha \in \Sigma(\mathcal{H})$ yields what is generally known as a *directed adjacency matrix* [30], denoted here as a *transition matrix*.

Definition 7 (Transition Matrix): Given an automaton $\mathcal{G}_{\Lambda^{\mathcal{H}}}$, the *transition matrix* $F_\alpha(\mathcal{G}_{\Lambda^{\mathcal{H}}}) \in \mathbb{R}^{n_V \times n_V}$, with $n_V = \left|V_{\Lambda^{\mathcal{H}}}\right|$ and $\alpha \in \Sigma(\mathcal{H})$, is computed as $F_\alpha(\mathcal{G}_{\Lambda^{\mathcal{H}}}) = \{f_{i,j}(\alpha)\}$ with

$$f_{i,j}(\alpha) = \begin{cases} 1, & \text{if } \exists\, e = (v_j, v_i, \alpha) \in E_{\Lambda^{\mathcal{H}}} \\ 0, & \text{otherwise.} \end{cases}$$

Since *at most one* successor exists from each vertex with a transition labeled with $\alpha \in \Sigma(\mathcal{H})$, matrix F_α will have a column sum of either 1 or 0. We now introduce a vector $q_t \in \mathbb{R}^{n_V}$ called *G-state*, with $n_V = |V_{\Lambda^{\mathcal{H}}}|$, representing the state of the given automaton $\mathcal{G}_{\Lambda^{\mathcal{H}}}$ at interval π_t.

Definition 8 (G-State q_t): Given an automaton $\mathcal{G}_{\Lambda^{\mathcal{H}}}$ and a string $\omega \in \Sigma(\mathcal{H})^N$, $\omega = \langle \alpha_1, \alpha_2, \ldots, \alpha_N \rangle$, for $k = |v|$, $v \in V_{\Lambda^{\mathcal{H}}}$, we define $q_t \in \mathbb{R}^{n_V}$, where the i-th element $q_{t,i}$ is:

$$q_{t,i} = \begin{cases} 1, & \text{if } \langle \alpha_{t-k}, \ldots, \alpha_{t-1} \rangle \equiv v_i \in V_{\Lambda^{\mathcal{H}}} \\ 0, & \text{otherwise.} \end{cases}$$

The G-state q_t is the vector representation of the vertex *left* at step t: here, $q_t = 0$ means that the transition at step $t-1$ was infeasible for the automaton. Given an arbitrary string $\omega = \langle \alpha_1, \ldots, \alpha_t, \ldots \rangle$, the G-state dynamics is defined as $q_{t+1} = F_\alpha(\mathcal{G}_{\Lambda^{\mathcal{H}}}) \cdot q_t$, and the following property holds [30].

Lemma 1 (Infeasible Sequence): If $\omega \notin \mathcal{S}_N(\Lambda^{\mathcal{H}})$, then $F_\omega(\mathcal{G}_{\Lambda^{\mathcal{H}}}) = F_{\alpha_N}(\mathcal{G}_{\Lambda^{\mathcal{H}}}) \cdots F_{\alpha_2}(\mathcal{G}_{\Lambda^{\mathcal{H}}}) \cdot F_{\alpha_1}(\mathcal{G}_{\Lambda^{\mathcal{H}}}) = 0$

Thus, if $q_t = 0$ for an arbitrary t, then $q_{t'} = 0$ for $t' \geq t$.

IV. Stability Analysis

Using the alphabet $\Sigma(\mathcal{H})$ and the chosen actuator mode (i.e., zeroing, or holding the previous value), we compute the closed-loop behavior of the controlled system. We identify one matrix for each dynamics corresponding to an interval π_t associated by $\alpha \in \Sigma(\mathcal{H})$, building the set $\mathcal{A}^{\mathcal{H}}$.

Kill: Defining $\tilde{x}_t^{\mathrm{K}} = [x_t^T \; z_t^T \; u_t^T]^T$ as the closed-loop state vector, we compute the discrete time closed-loop system dynamics $A_{\mathtt{H}}^{\mathrm{K}}$, corresponding to the character `H`:

$$\tilde{x}_{t+1}^{\mathrm{K}} = A_{\mathtt{H}}^{\mathrm{K}} \tilde{x}_t^{\mathrm{K}}, \quad A_{\mathtt{H}}^{\mathrm{K}} = \begin{bmatrix} A_p & 0 & B_p \\ -B_c C_p & A_c & -B_c D_p \\ -D_c C_p & C_c & -D_c D_p \end{bmatrix}.$$

For the case of `M`, the controller execution terminates prematurely and its states are not updated ($z_{t+1} = z_t$). Therefore, depending on the actuation mode, the controller output is either zeroed ($u_{t+1} = 0$) or held ($u_{t+1} = u_t$). The resulting closed-loop system in state-space form is denoted with $A_{\mathtt{M}}^{\mathrm{K}}$:

$$\tilde{x}_{t+1}^{\mathrm{K}} = A_{\mathtt{M}}^{\mathrm{K}} \tilde{x}_t^{\mathrm{K}}, \quad A_{\mathtt{M}}^{\mathrm{K}} = \begin{bmatrix} A_p & 0 & B_p \\ 0 & I & 0 \\ 0 & 0 & \Delta \end{bmatrix}.$$

Here, $\Delta = I$ (identity matrix) if the control signal is held and $\Delta = 0$ if zeroed. The set of dynamic matrices under the Kill strategy is then $\mathcal{A}^{\mathrm{Kill}} = \{A_{\mathtt{H}}^{\mathrm{K}}, A_{\mathtt{M}}^{\mathrm{K}}\}$.

Skip-Next: For the Skip-Next strategy, we introduce two additional states $\hat{x}_t$ and $\hat{u}_t$ storing the old values of x_t and u_t while the controller awaits an update. The resulting state vector then becomes $\tilde{x}_t^{\mathrm{S}} = [x_t^T \; z_t^T \; u_t^T \; \hat{x}_t^T \; \hat{u}_t^T]^T$. When π_t is associated to `H`, the two additional states mirror the behaviour of the states of which they are storing data. The resulting closed-loop system is described using $A_{\mathtt{H}}^{\mathrm{S}}$:

$$\tilde{x}_{t+1}^{\mathrm{S}} = A_{\mathtt{H}}^{\mathrm{S}} \tilde{x}_t^{\mathrm{S}}, \quad A_{\mathtt{H}}^{\mathrm{S}} = \begin{bmatrix} A_p & 0 & B_p & 0 & 0 \\ -B_c C_p & A_c & -B_c D_p & 0 & 0 \\ -D_c C_p & C_c & -D_c D_p & 0 & 0 \\ A_p & 0 & B_p & 0 & 0 \\ -D_c C_p & C_c & -D_c D_p & 0 & 0 \end{bmatrix}.$$

For the case of `M` in π_t, $\hat{x}_t$ and $\hat{u}_t$ maintain their previous values. The resulting closed-loop is described by $A_{\mathtt{M}}^{\mathrm{S}}$:

$$\tilde{x}_{t+1}^{\mathrm{S}} = A_{\mathtt{M}}^{\mathrm{S}} \tilde{x}_t^{\mathrm{S}}, \quad A_{\mathtt{M}}^{\mathrm{S}} = \begin{bmatrix} A_p & 0 & B_p & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & \Delta & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix}.$$

Finally, for the case of `R`, the new control command is calculated using the values stored in $\hat{x}_t$ and $\hat{u}_t$. The resulting closed-loop system is described by $A_{\mathtt{R}}^{\mathrm{S}}$:

$$\tilde{x}_{t+1}^{\mathrm{S}} = A_{\mathtt{R}}^{\mathrm{S}} \tilde{x}_t^{\mathrm{S}}, \quad A_{\mathtt{R}}^{\mathrm{S}} = \begin{bmatrix} A_p & 0 & B_p & 0 & 0 \\ 0 & A_c & 0 & -B_c C_p & -B_c D_p \\ 0 & C_c & 0 & -D_c C_p & -D_c D_p \\ A_p & 0 & B_p & 0 & 0 \\ 0 & C_c & 0 & -D_c C_p & -D_c D_p \end{bmatrix}.$$

The resulting set of matrices under Skip-Next strategy is then defined as $\mathcal{A}^{\text{Skip-Next}} = \{A_{\mathtt{H}}^{\mathrm{S}}, A_{\mathtt{M}}^{\mathrm{S}}, A_{\mathtt{R}}^{\mathrm{S}}\}$.

A. Kronecker Lifted Switching System

Combining the set of system dynamics $\mathcal{A}^{\mathcal{H}}$ with the associated automaton $\mathcal{G}_{\Lambda^{\mathcal{H}}}$, we seek to obtain an equivalent system model based on Kronecker lifting, characterized by a set of matrices denoted by $\mathcal{L}_{\Lambda^{\mathcal{H}}}$ and behaving as an *arbitrary switching system*, such that $\rho(\mathcal{L}_{\Lambda^{\mathcal{H}}}) = \rho(\mathcal{A}^{\mathcal{H}}, \mathcal{G}_{\Lambda^{\mathcal{H}}})$. In this way, powerful algorithms applicable to arbitrary switching system [23], [26] can be used to find tight stability

bounds. We build upon the Kronecker lifting approach of [29]. Leveraging the vector q_t of Definition 8, we introduce the *lifted discrete-time state* $\xi_t \in \mathbb{R}^{n \cdot n_V}$, defined as $\xi_t = q_t \otimes x_t$, where $n_V = \left|V_{\Lambda^{\mathcal{H}}}\right|$ and $\otimes$ is the Kronecker product. By construction, ξ_t is a vector composed of n_V blocks of size n, where at most one block is equal to x_t and all other blocks are equal to the 0 vector. Then, we build a set of lifted matrices $P_\alpha(\mathcal{G}_{\Lambda^{\mathcal{H}}}) \in \mathbb{R}^{n \cdot n_V \times n \cdot n_V}$, which incorporates both the system dynamics and the possible transitions given a certain outcome $\alpha \in \Sigma(\mathcal{H})$:

$$P_\alpha(\mathcal{G}_{\Lambda^{\mathcal{H}}}) = F_\alpha(\mathcal{G}_{\Lambda^{\mathcal{H}}}) \otimes A_\alpha^{\mathcal{H}}, \quad \alpha \in \Sigma(\mathcal{H}). \tag{5}$$

The lifted dynamics of the closed loop system then become $\xi_{t+1} = P_\alpha(\mathcal{G}_{\Lambda^{\mathcal{H}}}) \cdot \xi_t$. Formally, we obtain a system composed of a set of switching dynamic matrices, $\mathcal{L}_{\Lambda^{\mathcal{H}}}$.

Definition 9 (Lifted Switching Set $\mathcal{L}_{\Lambda^{\mathcal{H}}}$): Given a set of dynamic matrices $\mathcal{A}^{\mathcal{H}}$ and an automaton $\mathcal{G}_{\Lambda^{\mathcal{H}}}$, the switching set $\mathcal{L}_{\Lambda^{\mathcal{H}}}$ is defined as: $\mathcal{L}_{\Lambda^{\mathcal{H}}} = \{P_\alpha(\mathcal{G}_{\Lambda^{\mathcal{H}}}) \,|\, \alpha \in \Sigma(\mathcal{H})\}$.

Leveraging the mixed-product property of $\otimes$ and introducing a proper submultiplicative norm, it is possible to prove that $\rho(\mathcal{L}_{\Lambda^{\mathcal{H}}}) = \rho\left(\mathcal{A}^{\mathcal{H}}, \mathcal{G}_{\Lambda^{\mathcal{H}}}\right)$. For more details and a formal proof we refer the interested reader to [29].

B. Extended Weakly Hard and JSR Properties

We now provide a general relation between *all* EWHCs in terms of the joint spectral radii.

Theorem 1 (JSR Dominance): Given $\lambda_1^{\mathcal{H}}$ and $\lambda_2^{\mathcal{H}}$ as arbitrary EWHCs, if $\lambda_2^{\mathcal{H}} \preceq \lambda_1^{\mathcal{H}}$ then $\rho(\mathcal{L}_{\lambda_2^{\mathcal{H}}}) \leq \rho(\mathcal{L}_{\lambda_1^{\mathcal{H}}})$.

Proof: From Equation (3), for a generic EWHC $\lambda^{\mathcal{H}}$,

$$\rho\left(\mathcal{L}_{\lambda^{\mathcal{H}}}\right) = \lim_{\ell \to \infty} \rho_\ell\left(\mathcal{L}_{\lambda^{\mathcal{H}}}\right), \;\; \rho_\ell\left(\mathcal{L}_{\lambda^{\mathcal{H}}}\right) = \max_{a \in \mathcal{S}_\ell(\lambda^{\mathcal{H}})} \|A_a\|^{1/\ell}.$$

Definition 3 gave us that $\lambda_2^{\mathcal{H}} \preceq \lambda_1^{\mathcal{H}}$ iff $\mathcal{S}(\lambda_2^{\mathcal{H}}) \subseteq \mathcal{S}(\lambda_1^{\mathcal{H}})$. Thus, if for a string b it holds that $b \in \mathcal{S}_\ell(\lambda_2^{\mathcal{H}})$, then it also holds that $b \in \mathcal{S}_\ell(\lambda_1^{\mathcal{H}})$. The set of all possible A_b is thus included in the set of all possible A_a, $a \in \mathcal{S}_\ell(\lambda_1^{\mathcal{H}})$, thus:

$$\max_{b \in \mathcal{S}_\ell(\lambda_2^{\mathcal{H}})} \|A_b\|^{1/\ell} \leq \max_{a \in \mathcal{S}_\ell(\lambda_1^{\mathcal{H}})} \|A_a\|^{1/\ell}, \quad \forall \ell \in \mathbb{N} \setminus 0.$$

The theorem follows immediately when $\ell \to \infty$. ∎

Theorem 1 is the first result that provides an analytic, correlation between the control theoretical analysis and real-time implementation. Primarily, it implies that the constraint dominance from Definition 3 also carries on to the JSR, giving us a notion of *JSR dominance*. The results of Theorem 1 are strategy-independent, further reducing the coupling between the control analysis and real-time implementation, and are also independent of the controlled system's dynamics.

Two Corollaries of Theorem 1 are derived for the commonly used models $\langle\overline{m}\rangle^{\mathcal{H}}$ and $\overline{\binom{m}{k}}^{\mathcal{H}}$, highlighting some practical relations between such constraints.

Corollary 1 ($\overline{\binom{m}{k}}^{\mathcal{H}}$ dominance): Given $\lambda_1^{\mathcal{H}} = \overline{\binom{m}{k_1}}^{\mathcal{H}}$ and $\lambda_2^{\mathcal{H}} = \overline{\binom{m}{k_2}}^{\mathcal{H}}$, if $k_1 \leq k_2$ then $\rho(\mathcal{L}_{\lambda_2^{\mathcal{H}}}) \leq \rho(\mathcal{L}_{\lambda_1^{\mathcal{H}}})$.

Corollary 2 ($\langle\overline{m}\rangle^{\mathcal{H}}$ dominance): Given $\lambda_1^{\mathcal{H}} = \langle\overline{m}\rangle^{\mathcal{H}}$ and $\lambda_2^{\mathcal{H}} = \overline{\binom{m}{k}}^{\mathcal{H}}$, then $\rho(\mathcal{L}_{\lambda_2^{\mathcal{H}}}) \leq \rho(\mathcal{L}_{\lambda_1^{\mathcal{H}}})$.

The conclusions drawn from Theorem 1 are theoretical, but its practical applicability lies in the algorithm used to find ρ^{LB} and ρ^{UB}, i.e., lower and upper bounds for the JSR value. Using these bounds we can determine the stability of the corresponding switching systems, as follows:

$$\rho^{LB}\left(\mathcal{L}_{\lambda_2^{\mathcal{H}}}\right) \leq \rho\left(\mathcal{L}_{\lambda_2^{\mathcal{H}}}\right) \leq \rho\left(\mathcal{L}_{\lambda_1^{\mathcal{H}}}\right) \leq \rho^{UB}\left(\mathcal{L}_{\lambda_1^{\mathcal{H}}}\right).$$

Regardless of the algorithm used to find the bounds, if $\lambda_2^{\mathcal{H}} \preceq \lambda_1^{\mathcal{H}}$ and $\rho^{UB}(\mathcal{L}_{\lambda_1^{\mathcal{H}}}) < 1$, the system under $\lambda_2^{\mathcal{H}}$ is switching stable. A similar relation holds for the lower bound.

Theorem 1 can be further extended by relating the joint spectral radius of a single constraint to sets of constraints.

Theorem 2: Given an arbitrary EWHC $\lambda^{\mathcal{H}}$, it holds that $\rho(\mathcal{L}_{\Lambda^{\mathcal{H}}}) \leq \rho(\mathcal{L}_{\lambda^{\mathcal{H}}}), \;\; \forall \Lambda^{\mathcal{H}} \ni \lambda^{\mathcal{H}}$.

Proof: For an arbitrary EWHC set $\Lambda^{\mathcal{H}} = \{\lambda_1^{\mathcal{H}}, \ldots, \lambda_N^{\mathcal{H}}\}$, its satisfaction set is $\mathcal{S}_\ell\left(\Lambda^{\mathcal{H}}\right) = \bigcap_{i \in \{1,\ldots,N\}} \mathcal{S}_\ell\left(\lambda_i^{\mathcal{H}}\right)$. Thus, for any $\lambda_i^{\mathcal{H}} \in \Lambda^{\mathcal{H}}$ it holds that $\mathcal{S}_\ell\left(\Lambda^{\mathcal{H}}\right) \subseteq \mathcal{S}_\ell\left(\lambda^{\mathcal{H}}\right)$. If a string b is in $\mathcal{S}_\ell\left(\Lambda^{\mathcal{H}}\right)$ it also belongs to $\mathcal{S}_\ell\left(\lambda^{\mathcal{H}}\right)$. The set of all possible A_b is thus included in the set of all possible A_a, $a \in \mathcal{S}_\ell\left(\lambda^{\mathcal{H}}\right)$. As a consequence it holds that

$$\max_{b \in \mathcal{S}_\ell(\Lambda^{\mathcal{H}})} \|A_b\|^{1/\ell} \leq \max_{a \in \mathcal{S}_\ell(\lambda^{\mathcal{H}})} \|A_a\|^{1/\ell}, \quad \forall \ell \in \mathbb{N}^{>}.$$

The theorem follows immediately when $\ell \to \infty$. ∎

As in Theorem 1, the more we restrict the execution pattern of the control task with sets of constraints, the lower its JSR will be. Theorem 2 delivers the practical insight that enforcing tighter EWHC to a stable system will *never* destabilise it, as formally stated in the following corollary.

Corollary 3: Given an arbitrary EWHC $\lambda^{\mathcal{H}}$, if $\rho(\mathcal{L}_{\lambda^{\mathcal{H}}}) < 1$ then $\rho(\mathcal{L}_{\Lambda^{\mathcal{H}}}) < 1, \;\; \forall \Lambda^{\mathcal{H}} \ni \lambda^{\mathcal{H}}$.

V. Evaluation

We apply the lifted dynamics model presented in Section IV to a representative plant for the process industry, controlled using a PI-controller, sampled with $T = 0.5$ s:

$$\begin{cases} x_{t+1} = \begin{bmatrix} 0.606 & 0.304 & 0.076 \\ 0 & 0.606 & 0.304 \\ 0 & 0 & 0.606 \end{bmatrix} x_t + \begin{bmatrix} 0.014 \\ 0.091 \\ 0.394 \end{bmatrix} u_t \\ y_t = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x_t \end{cases}$$
$$\begin{cases} z_{t+1} = z_t + 0.359 y_t \\ u_{t+1} = 0.454 z_t + 0.633 y_t. \end{cases}$$

We analyse the stability of the control systems subject to different $\overline{\binom{m}{k}}^{\mathcal{H}}$ constraints. We consider all combinations of strategy (Kill or Skip-Next) and actuator mode (zeroing or holding). For each combination, we generate the lifted set $\mathcal{L}_{\lambda^{\mathcal{H}}}$. Its JSR $\rho(\mathcal{L}_{\lambda^{\mathcal{H}}})$ is then approximated using three different algorithms. First, a lower and upper bound of $\rho(\mathcal{L}_{\lambda^{\mathcal{H}}})$ is computed using the `JSR toolbox` [23]. Then, an upper bound of the JSR is obtained via SOS relaxations, using both the *dense* and *sparse* algorithm from `SparseJSR` [26].

Table I displays our results, acquired on an Intel Core i5-8265U@1.60GHz CPU with 8GB RAM. Lower and upper bounds are denoted "LB" and "UB". All upper bounds obtained with `JSR toolbox` was found greater than the ones obtained with SOS, thus omitted from the Table. The symbol "–" means that the SDP solver runs out of memory. The SDP solver in `SparseJSR` uses a second-order method. Thus, a different solver (utilizing a first-order method) could reduce memory usage at the cost of potential accuracy loss.

TABLE I
RESULTS OBTAINED WITH OUR CONTROLLED EXAMPLE

$\overline{\binom{m}{k}}\mathcal{H}$	JSR[23]	Dense	Sparse		JSR[23]	Dense	Sparse	
	LB	UB	UB	×	LB	UB	UB	×
$m,\ k$	**Kill and zeroing**				**Kill and holding**			
1, 2	0.960	1.070	1.070	0.86	0.926	1.029	1.029	0.83
1, 3	0.920	**0.995**	**0.995**	0.83	0.894	**0.971**	**0.971**	0.77
1, 4	0.890	**0.945**	**0.996**	1.06	0.894	**0.957**	1.025*	1.25
1, 5	0.890	**0.922**	**0.983**	1.96	0.894	**0.948**	1.008*	2.25
1, 6	0.890	**0.920**	**0.975**	4.36	0.894	**0.942**	**0.995**	3.68
2, 3	0.983	1.124	1.124	0.67	0.956	1.085	1.085	0.80
2, 4	0.960	1.079	1.079	0.74	0.927	1.039	1.039	0.86
2, 5	0.939	1.039	1.142	2.09	0.905	1.002	1.105	1.58
2, 6	0.920	1.007	1.096	12.3	0.903	**0.974**	1.080	19.2
$m,\ k$	**Skip-Next and zeroing**				**Skip-Next and holding**			
1, 2	0.922	**0.924**	**0.924**	5.40	0.958	**0.958**	**0.958**	4.43
1, 3	0.898	**0.974**	**0.974**	10.5	0.917	**0.988**	**0.988**	10.4
1, 4	0.898	**0.963**	**0.963**	18.2	0.890	**0.940**	**0.940**	15.9
1, 5	0.898	**0.954**	**0.954**	17.6	0.890	**0.929**	**0.929**	20.8
1, 6	0.898	**0.946**	**0.947**	20.9	0.890	**0.927**	**0.927**	25.8
2, 3	0.953	1.034	1.039	4.45	0.982	1.070	1.076	5.91
2, 4	0.922	1.033	1.040	23.9	0.958	1.079	1.086	24.2
2, 5	0.898	**0.999**	1.005	77.8	0.937	1.038	1.043	58.1
2, 6	0.907	–*	1.007	–	0.917	–	**0.991**	–

Bold values represent stable systems under their corresponding EWHC, strategy, and actuator mode. Starred values represent stable systems inferred from Corollary 1. The `JSR toolbox` provides an accurate lower bound and a coarse upper bound. In contrast, the dense SOS method finds a better upper bound but takes more time. We compare the time to run both SOS methods, indicating with "×" the speedup factor to obtain the sparse bound w.r.t. the dense.

All the upper bounds computed by `JSR toolbox` are greater than 1, while all lower bounds are *below* 1, thus we cannot draw any conclusion using the `JSR toolbox`. For all EWHC, $\overline{\binom{m}{k}}\mathcal{H}$ where $m = 1$ and $2 < k \leq 6$ the SOS upper bounds allow us to infer that the system is stable for all combinations of strategy and actuator mode, and also for $k = 2$ under the Skip-Next strategy. From Theorem 1, the stability will hold also for all constraints that are harder to satisfy; in particular, Corollary 1 implies stability for all $\overline{\binom{m}{k}}\mathcal{H}$ with $m = 1$ and $k > 6$. The speedup ratio is growing when k increases, yielding a particularly high benefit of exploiting sparsity for the Skip-Next strategy with actuation zeroing.

VI. CONCLUSION

This letter proposes a switching stability analysis framework for LTI systems with arbitrary weakly-hard constraints, extending the weakly-hard model and providing an analytic stability bound. The analysis allows us to assess whether computational errors (present in industrial controllers) affect the stability of the controlled systems. Future work will focus on the performance loss due to the presence of deadline misses following the extended weakly-hard model.

REFERENCES

[1] B. Akesson, M. Nasri, G. Nelissen, S. Altmeyer, and R. I. Davis, "An empirical survey-based study into industry practice in real-time systems," in *Proc. Real-Time Syst. Symp.*, 2020, pp. 3–11.

[2] G. Bernat, A. Burns, and A. Liamosi, "Weakly hard real-time systems," *IEEE Trans. Comput.*, vol. 50, no. 4, pp. 308–321, Apr. 2001.

[3] A. Cervin, "Analysis of overrun strategies in periodic control tasks," *IFAC Proc. Vol.*, vol. 38, no. 1, pp. 219–224, 2005.

[4] H. Choi, H. Kim, and Q. Zhu, "Job-class-level fixed priority scheduling of weakly-hard real-time systems," in *Proc. Real-Time Embedded Technol. Appl. Symp.*, 2019, pp. 241–253.

[5] X. Dai, "A Gel'fand-type spectral radius formula and stability of linear constrained switching systems," *Lin. Algebra Appl.*, vol. 436, no. 5, pp. 1099–1113, 2012.

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

[7] M. Hertneck, S. Linsenmayer, and F. Allgöwer, "Efficient stability analysis approaches for nonlinear weakly-hard real-time control systems," *Automatica*, vol. 133, Nov. 2021, Art. no. 109868.

[8] C. Huang, K.-C. Chang, C.-W. Lin, and Q. Zhu, "Saw: A tool for safety analysis of weakly-hard systems," in *Proc. Int. Conf. Comput. Aided Verification*, 2020, pp. 543–555.

[9] C. Huang, W. Li, and Q. Zhu, "Formal verification of weakly-hard systems," in *Proc. 22nd ACM Int. Conf. Hybrid Syst. Comput. Control*, 2019, pp. 197–207.

[10] V. Kozyakin, "The Berger–Wang formula for the Markovian joint spectral radius," *Lin. Algebra Appl.*, vol. 448, pp. 315–328, May 2014.

[11] J. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM J. Optim.*, vol. 11, no. 3, pp. 796–817, 2001.

[12] H. Liang, Z. Wang, R. Jiao, and Q. Zhu, "Leveraging weakly-hard constraints for improving system fault tolerance with functional and timing guarantees," in *Proc. Conf. Comput. Aided Design*, 2020, p. 101.

[13] H. Liang, Z. Wang, D. Roy, S. Dey, S. Chakraborty, and Q. Zhu, "Security-driven codesign with weakly-hard constraints for real-time embedded systems," in *Proc. Int. Conf. Comput. Design*, 2019, pp. 217–226.

[14] S. Linsenmayer and F. Allgower, "Stabilization of networked control systems with weakly hard real-time dropout description," in *Proc. Conf. Decis. Control*, 2017, pp. 4765–4770.

[15] S. Linsenmayer, M. Hertneck, and F. Allgöwer, "Linear weakly hard real-time control systems: Time-and event-triggered stabilization," *IEEE Trans. Autom. Control*, vol. 66, no. 4, pp. 1932–1939, Apr. 2021.

[16] M. Maggio, A. Hamann, E. Mayer-John, and D. Ziegenbein, "Control-system stability under consecutive deadline misses constraints," in *Proc. Euromicro Conf. Real-Time Syst.*, 2020, pp. 1–24.

[17] P. Parrilo and A. Jadbabaie, "Approximation of the joint spectral radius using sum of squares," *Lin. Algebra Appl.*, vol. 428, pp. 2385–2402, May 2008.

[18] P. Pazzaglia, A. Hamann, D. Ziegenbein, and M. Maggio, "Adaptive design of real-time control systems subject to sporadic overruns," in *Proc. Design Autom. Test Europe Conf. Exhibition*, 2021, pp. 1887–1892.

[19] P. Pazzaglia, C. Mandrioli, M. Maggio, and A. Cervin, "Deadline-miss-aware control," in *Proc. Euromicro Conf. Real-Time Syst.*, 2019, pp. 1–24.

[20] M. Philippe, R. Essick, G. E. Dullerud, and R. M. Jungers, "Stability of discrete-time switching systems with constrained switching sequences," *Automatica*, vol. 72, pp. 242–250, Oct. 2016.

[21] G. Rota and W. G. Strang, "A note on the joint spectral radius," *Indagationes Mathematicae*, vol. 63, pp. 379–381, Apr. 1960. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1385725860500461

[22] L. Schenato, "To zero or to hold control inputs with lossy links?" *IEEE Trans. Autom. Control*, vol. 54, no. 5, pp. 1093–1099, May 2009.

[23] G. Vankeerberghen, J. Hendrickx, and R. Jungers, "JSR: A toolbox to compute the joint spectral radius," in *Proc. Int. Conf. Hybrid Syst. Comput. Control*, 2014, pp. 1–6.

[24] N. Vreman, A. Cervin, and M. Maggio, "Stability and performance analysis of control systems subject to bursts of deadline misses," in *Proc. Euromicro Conf. Real-Time Syst.*, 2021, pp. 1–23.

[25] N. Vreman, R. Pates, and M. Maggio, "Weaklyhard.jl: Scalable analysis of weakly-hard constraints," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp.*, 2022, pp. 228–240.

[26] J. Wang, M. Maggio, and V. Magron, "SparseJSR: A fast algorithm to compute joint spectral radius via sparse SOS decompositions," in *Proc. Amer. Control Conf.*, 2021, pp. 2254–2259.

[27] J. Wang, V. Magron, and J. Lasserre, "TSSOS: A moment-SOS hierarchy that exploits term sparsity," *SIAM J. Optim.*, vol. 31, no. 1, pp. 30–58, 2021.

[28] W. Xu, Z. Hammadeh, A. Kröller, R. Ernst, and S. Quinton, "Improved deadline miss models for real-time systems using typical worst-case analysis," in *Proc. Euromicro Conf. Real-Time Syst.*, 2015, pp. 247–256.

[29] X. Xu and B. Açikmeşe, "Approximation of the constrained joint spectral radius via algebraic lifting," *IEEE Trans. Autom. Control*, vol. 66, no. 7, pp. 3386–3392, Jul. 2021.

[30] X. Xu and Y. Hong, "Matrix expression and reachability analysis of finite automata," *J. Control Theory Appl.*, vol. 10, pp. 210–215, Apr. 2012.