A Formal Approach towards Safe and Stable Schedule Synthesis in Weakly Hard Control Systems

DEBARPITA BANERJEE*, Indian Statistical Institute, India
PARASARA SRIDHAR DUGGIRALA, The University of North Carolina at Chapel Hill, USA
BINEET GHOSH, The University of Alabama, USA
SUMANA GHOSH, Indian Statistical Institute, India

Real-time scheduling of multiple control tasks in a *weakly hard* setting is an emerging research direction, as it offers a more flexible and feasible environment for task scheduling. This is especially pertinent for resource-constrained embedded applications where tasks are allowed to miss a few deadlines for prudent sharing of computational resources. However, a control task missing its deadline could result in the system being unsafe or unstable. A significant amount of research efforts have been reported in the literature addressing the schedulability of control tasks while preserving the stability or safety. However, all of them focus on a stable schedule or a safe schedule, but not both the safety and stability aspects together. In this work, we ensure both control stability and safety to generate a *safe and stable schedule* for a weakly hard task system. In particular, we gradually endorse stability, safety, and schedulability, where we first synthesize a weakly hard constraint that preserves the desired stability of each control task. Next, we correlate stability with control safety and establish some mathematical results that guarantee control safety for an unbounded time horizon, unlike the existing methods. Finally, by leveraging *Satisfiability Modulo Theories (SMT)*, we synthesize the schedule that ensures control stability and safety while minimizing the *worst-case response time* of all the tasks, in a time-efficient way. To our knowledge, this is the first work to address *stability, safety, and schedulability* together for weakly hard task systems. We validate our method through extensive experiments using standard automotive benchmarks. In addition, we demonstrate the efficiency of the proposed method in comparison with some of the state-of-the-art techniques, as well as highlight its scalability, thereby establishing its applicability in real-world scenarios.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; **Embedded software**; • **Software and its engineering** → *Real-time schedulability*.

Additional Key Words and Phrases: Real-Time Embedded Systems, Control Stability, Control Safety, Scheduling

ACM Reference Format:
Debarpita Banerjee, Parasara Sridhar Duggirala, Bineet Ghosh, and Sumana Ghosh. 2025. A Formal Approach towards Safe and Stable Schedule Synthesis in Weakly Hard Control Systems. *ACM Trans. Embedd. Comput. Syst.* 1, 1, Article 1 (July 2025), 25 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Real-time task scheduling is one of the most prominent areas of research under the domain of embedded and cyber-physical systems [2, 12, 21]. Any real-time task has a *hard real-time requirement*

*This is the corresponding author

Authors' addresses: Debarpita Banerjee, debarpita2023_r@isical.ac.in, Indian Statistical Institute, Kolkata, India; Parasara Sridhar Duggirala, psd@cs.unc.edu, The University of North Carolina at Chapel Hill, USA; Bineet Ghosh, bineet@ua.edu, The University of Alabama, USA; Sumana Ghosh, sumana@isical.ac.in, Indian Statistical Institute, Kolkata, India.

ACM 1539-9087/2025/7-ART1
https://doi.org/10.1145/nnnnnnn.nnnnnnn

to complete its execution before its deadline to ensure the system's functionality. However, such hard requirement often leads to situations, where the resources can no longer be appropriately shared among the tasks, leading to an infeasible schedule of tasks in the shared processor. To improve the schedulability and resource allocation among multiple tasks, especially for resource-constrained embedded environments, recent design trends advocate migrating to a *weakly hard* setting [3]. Here the tasks can miss some of their deadlines occasionally, without hampering the system's performance. In this work, we consider such a weakly hard setup where each control task is only required to meet at least m deadlines in every sequence of k consecutive deadlines, following an $\binom{m}{k}$*-firm constraint*. Real-time control systems literature has explored the weakly hard framework in several contexts, such as resource and priority management [14, 19], control-scheduling co-design [10, 16], platform-level uncertainty management [15, 23]. In addition, control systems literature has accounted for weakly hard settings in other parallel contexts, and several seminal results exist in the related areas, e.g., network control systems [4, 12, 16], secure control systems [1]. However, in the presence of deadline misses, it is essential to ensure two major aspects, i.e., *control stability* and *control safety*, as it is implicit in the design of any hard real-time control task system. Intuitively, control stability refers to the ability of maintaining the desired state even in presence of external disturbances, whereas, control safety indicates that the system remains in a safe state despite timing uncertainties like deadline misses. Therefore, in our work, we carefully consider both stability and safety during schedule synthesis for a weakly hard control system to achieve the desired outcome—*a stable and safe schedule*. Existing methods in this direction explore schedulability of control tasks focusing either on the stability or on the safety aspect. For example, [7, 10, 16] focus on schedulability paired with stability, whereas safety is combined with schedulability in the recent works like [13, 23–25]. Some research efforts concentrate only on a single aspect at a time like stability [4, 17], or safety [15]. In contrast, in this work, we explore the triplet, ***(S_1: Stability, S_2: Safety, S_3: Schedulability)*** in the context of weakly hard control systems, for the first time in the literature. In particular, in this work, an *exponential stability* criterion is derived from the *settling time* requirement [10], where settling time indicates the specific time by which the system's output reaches the desired reference. On the other hand, the control safety is established by bounding the *deviation* between the ideal behavior (with no deadline miss) and the behavior of the system under deadline misses [23, 25]. In general, corresponding to any $\binom{m}{k}$-firm constraint, there is a collection of $f(m,k) = {}^{k}C_{m} + {}^{k}C_{m+1} + \cdots + {}^{k}C_{k}$ deadline hit-miss patterns. To obtain safe $\binom{m}{k}$-firm constraints, a *safety verification* process is applied on all the $f(m,k)$ patterns for *multiple* such $\binom{m}{k}$-constraints. This is done by verifying whether the deviation, between the system's state trajectory following each such pattern and the state trajectory corresponding to the ideal behavior, crosses the given bound. The existing methods account for time-consuming and extensive reachability algorithms [15, 23] or probabilistic techniques [8, 25] to perform this safety verification. We overcome the perplexity of handling the extensive verification process, for the combinatorial collection of patterns respecting multiple $\binom{m}{k}$-constraints, in the following way.

Novelty of the Proposed Method. We integrate the concepts of stability and safety to not only obtain a safe and stable schedule but also make the safety verification process [15, 23, 25] computationally efficient. We begin by ensuring stability through settling-time requirements, deriving *exactly one* stability-oriented $\binom{m}{k}$-firm constraint for a control task, with *significantly* smaller values of m and k. This specifically accelerates the safety verification process by avoiding multiple $\binom{m}{k}$s and also enhances the control performance. Furthermore, we ensure control safety over an infinite time horizon, which is not addressed in existing methods such as [23–25], despite adopting the same safety notion as ours. On ensuring both stability and safety, we then proceed to synthesize a safe and stable schedule by employing the *Satisfiability Modulo Theories (SMT)* [18].

The SMT-based approach has remained relatively underexplored in weakly hard control scheduling (with a few exceptions like [26]), mainly due to its inherent scalability challenges, as it requires exploring a vast search space of potential schedules. However, in our work, SMT is able to report a feasible schedule within a reasonable time frame because we significantly prune the search space of the SMT solver through two key steps: i) by deducing exactly one stability-oriented $\binom{m}{k}$-firm constraint for a control task and ii) by selecting one deadline hit-miss pattern for the task which obeys both stability and safety criteria. Furthermore, we generate a non-preemptive schedule by *minimizing the worst-case response time (WCRT)* of all the control tasks. This particularly helps in outperforming the existing preemptive EDF-based scheduling technique [10] and also the SMT-based scheduling approach with non-preemptive-EDF [26], used in a similar context as ours.

Outline of the Proposed Method. The proposed method generates a stable and safe schedule for a set of n control tasks sharing a common processor. The method has three main steps. First, to incorporate stability (S_1), we consider the standard control design parameter, *settling time*, to deduce exactly one stability-oriented $\binom{m}{k}$-firm constraint for a control task, complying with the exponential stability criterion. Once stability (S_1) is ensured, we proceed to the second step with n stability-oriented $\binom{m}{k}$-firm constraints for n tasks, to bring control safety (S_2) into effect. For a control task, safety is ensured by: i) constructing a stability-and-safety-oriented $\binom{m'}{k'}$-firm constraint, and ii) choosing a specific deadline hit-miss pattern p that meets the $\binom{m'}{k'}$-firm constraint, ensuring that the system's performance only deviates from the ideal behavior (where no deadline misses are allowed) by a bounded amount [25]. We also establish the control safety over an infinite time horizon and mathematically prove that it is sufficient to perform the safety verification till the settling time. The final step accounts for schedulability (S_3) to generate a stable and safe schedule, having the set of n hit-miss patterns, $\{p_1, p_2, \cdots, p_n\}$ following their respective stability-and-safety-oriented $\binom{m'}{k'}$-firm constraints, as the input. We develop an SMT-based schedule minimizing the WCRT. Figure 1 presents an overview of these three steps. We show that our proposed method not only guarantees the essential attributes of stability and safety, but also enhances control performance, accelerates the schedule synthesis process, and schedules a large number of jobs. The experimental observations thenceforward, justify the discussed steps and the constructions.

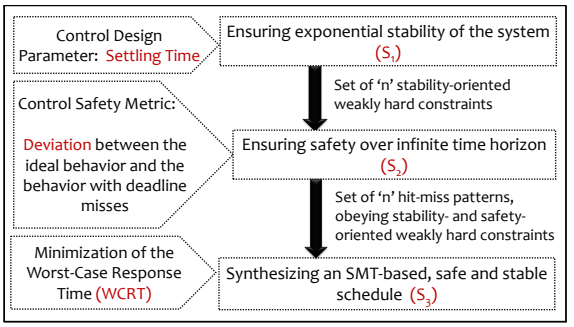


Fig. 1. Outline of the proposed method

Notable Contributions. Besides addressing the triplet (S_1, S_2, S_3) for the first time to introduce a safe and stable scheduling approach, our method enriches the existing literature as follows.

1) We correlate the control stability and control safety systematically by first deriving stability-oriented $\binom{m}{k}$ from the settling time and by deriving the stability-and-safety-oriented $\binom{m'}{k'}$ from $\binom{m}{k}$, with considerably smaller values of m, k, m' and k'. This helps to improve the control performance and also speeds up both the safety verification and schedule synthesis processes. Furthermore, we establish the control safety over an unbounded time horizon and also mathematically prove the sufficiency of actually performing the safety verification till the settling time.

2) We develop a novel SMT-based scheduling approach that minimizes the WCRT to generate a feasible schedule enhancing the scope of schedulability. We streamline the search space of the

SMT solver and significantly make it time- and compute-efficient by selecting exactly one deadline hit-miss pattern for a task, which obeys the stability and safety constraints for that task.

3) We conduct extensive experiments over 15 standard benchmark control systems from the automotive domain to highlight the scalability and pertinence of the proposed method in real-world applications. Specifically, we compare against four existing works [10, 23, 25, 26] to demonstrate the efficiency of the proposed method. When comparing our method's runtime to the approaches mentioned above, we observe significant improvements. For instance, our method successfully reports a feasible schedule within a highly reasonable time in most cases, while the method proposed in [10] encounters time-out issues. Additionally, we achieve a 92% improvement in stability and control performance compared to [23, 25], and a 21.57% improvement in safety compared to [10].

Organization. This paper is organized as follows. Section 2 lists down the related research work, highlighting their impacts and constraints. Section 3 discusses the background and Section 4 considers a case study to highlight the core challenges that the proposed approach aims to address. Section 5 delineates the entire method describing the three major steps. Section 6 presents experimental evaluations to showcase the efficiency and scalability of the proposed method. Finally, Section 7 summarizes the entire work with concluding remarks and future works.

2 RELATED WORK

Here, we present the related research work that addresses various issues including each of the three aspects, i.e., stability (S_1), safety (S_2) and schedulability (S_3), or their combinations.

S_1: Techniques for preserving **stability** in presence of deadline misses have been presented in [7, 10, 12, 17, 20]. The work explored in [17, 20] employs the idea of joint spectral radius, as a measure of stability. Asymptotically stable systems are considered in [7], whereas the exponential stability criterion, derived from the control design parameter, settling time, is dealt with in [10–12], which we too adopt in our work. The first three works discuss schedulability along with stability, and the line of work of [10] being similar to ours, we compare with them particularly, to prove the supremacy of the proposed method over theirs (ref. Section 6.3). None of the above methods constructs a single, stability-oriented $\binom{m}{k}$-firm constraint for each task, catalyzing both the safety verification process and the schedule synthesis.

S_2: Several types of **safety** analysis mechanisms have been explored in the last few years and many of them are applied in the scheduling context, too. Safe state sets (the state variables remain within a safety bound) are considered in [26], with which they develop a scheduling scheme and we show that our scheduling strategy is safe and also supersedes theirs. [13] establishes a co-design technique, DECNTR, which allows switching of sampling periods within a safe range and ensures the schedulability and robustness, maintaining the underlying safety. We do not follow this multi-mode control strategy based on the changes in the sampling period and also the co-design approach in our work. The quantitative safety notion, i.e., how much a trajectory, following a deadline hit-miss pattern, deviates from the ideal case with no miss, is addressed in [15, 23–25], which we too account for, in our work. The calculations for estimating an upper bound on this deviation through various reachability algorithms [15], iterative approaches [15, 23] and probabilistic methods [25], are complex and time-consuming. In contrast to these existing approaches, we use techniques from stability to deduce $\binom{m}{k}$-constraints that ensure safety. Also, we prove the safety till an infinite horizon, which is missing in these methods. Moreover, the weakly hard schedules generated by [23, 25] could potentially cause the systems to be unstable, in contrast to ours (ref. Section 6.3).

S_3: **Scheduling** of weakly hard control tasks, without accounting for stability and safety, is studied in [6, 14, 19]. The work in [14] uses an EDF scheduling policy to bound the distribution of deadline misses for uniprocessor systems and typical worst-case analysis (TWCA) serves as a

tool to compute the weakly-hard constraints. Fixed-priority preemptive schedulers are explored in [6, 19]. The former associates each job with a job-class, and the job's response time along with the job-class patterns aid to the scheduling process. However, the schedulability analysis for a set of periodic tasks, is based on an MILP formulation in [19]. Some work considering the blend of stability and schedulability includes [10], which solves an ILP to generate a stable schedule using EDF as the underlying scheduling algorithm. We compare with their method particularly, to prove the efficiency of our scheduling strategy (ref. Section 6.3). The authors of [7] construct an online state-aware scheduling approach, guaranteeing the system's stability and control performance. They schedule all the critical and some of the non-critical jobs reporting low schedulability ratios, for not too high utilization values. In contrast, we report a schedule even with much higher processor utilization in most cases. Safety is combined with schedulability to design a safe schedule in [23–25], where an automata-based scheduler, confined by the limitation of having equal sampling periods for all systems, is considered mostly. Our technique generates a schedule without the constraint of identical sampling periods. SMT-based scheduling, though well-studied in other areas of real-time systems [5, 21], is a research direction that is actively being explored in this domain. The only existing SMT-based scheduler that too tackles safety is [26]. However, this method suffers from higher complexity issues due to the iterative process of counter-example-guided refinements to obtain a safe schedule. On the other hand, the methods dealing with WCRT analysis [6, 22] work in different settings; either in multiprocessor systems or in circumstances handling worst-case temporal interference on a job-class. The framework considered in our work is different from these kinds of existing methods, as we determine the job's start and finish time by minimizing its response time, and its arrival time is computable from the safety and stability criteria. Such an idea, merged with SMT-based scheduling, is also a salient contribution of the proposed method.

3 BACKGROUND

This section discusses the system layout, consisting of the plant-controller model, the control performance, stability and safety paradigms followed by the description of the fundamentals of the weakly hard control task set.

3.1 Plant-Controller Pair

The dynamical system under consideration is referred to as the *plant* and there is a stabilizing *feedback controller*, which on discerning the plant output/plant state, regulates the control input periodically. Both of these together constitute the *plant-control closed loop system*, commonly known as a *feedback control loop*. The plant's dynamics are specified by a set of differential equations, also known as state equations, given as, $\dot{\boldsymbol{x}}(\boldsymbol{t}+\boldsymbol{1}) = \boldsymbol{Ax}(\boldsymbol{t}) + \boldsymbol{Bu}(\boldsymbol{t})$, $\boldsymbol{y}(\boldsymbol{t}) = \boldsymbol{Cx}(\boldsymbol{t})$. The constant time gap by which the plant's dynamics are checked by the controller is referred to as the *sampling period* of the controller. The choice of sampling period plays an important role in designing an appropriate controller. In this work, we consider discrete linear time-invariant (LTI) systems and the state equations are discretized to obtain the discrete LTI dynamics of the plant, given as follows.

$$\boldsymbol{x}[\boldsymbol{k}+\boldsymbol{1}] \;=\; \boldsymbol{A_d x}[\boldsymbol{k}] + \boldsymbol{B_d u}[\boldsymbol{k}], \quad \boldsymbol{y}[\boldsymbol{k}] \;=\; \boldsymbol{C_d x}[\boldsymbol{k}]. \tag{1}$$

Here, the vectors $\boldsymbol{x}[\boldsymbol{k}]$, $\boldsymbol{y}[\boldsymbol{k}]$ and $\boldsymbol{u}[\boldsymbol{k}]$ represent the plant state, output and the control input respectively, at the $\boldsymbol{k}$-th sampling instant, or at time $\boldsymbol{t} = \boldsymbol{kh}$, where $\boldsymbol{k} \in \mathbb{N}$ and $\boldsymbol{h}$ is the sampling period. $\boldsymbol{A_d}$, $\boldsymbol{B_d}$, and $\boldsymbol{C_d}$ are the system matrices. The LTI dynamics of the controller is given as,

$$\boldsymbol{u}[\boldsymbol{k}] = \boldsymbol{K x}[\boldsymbol{k}-\boldsymbol{1}]. \tag{2}$$

where $\boldsymbol{K}$ is the feedback control gain. The correct working of the controller relies upon the suitable values of the gain. In this work, we consider *static controllers*, i.e., controllers where the control

action is a function of only the most recent state measurement. Also, we consider a one-sample delayed system, as modeled in Eq. (2), and we use the standard *optimal control* technique, *Linear Quadratic Regulator* (LQR) for the controller design.

The discrete-time controller is generally implemented as a *software control task* in the underlying embedded architectural platform, where it gets executed periodically with a period of $\boldsymbol{h}$. The control task needs to complete its execution before a certain time, which is known as its *deadline* and this is its real-time requirement to be satisfied. In this work, we follow the logical execution time paradigm, i.e., the system's state, sampled at timestep $\boldsymbol{k-1}$, is used to obtain the new control input at timestep $\boldsymbol{k}$ (as in Eq. (2)). The new input is applied at the deadline (equal to the sampling period here) of the control job [17].

3.2 Control Performance and Stability

Both the control performance and the system's stability are cardinal features when dealing with a feedback control loop. We consider the control performance metric as the *settling time*. It is defined as the required time by which the system output reaches and remains around the reference value (e.g., within 5 % error band), after responding to a sudden change in the input (e.g., step-like reference change), under the assumption that the system is asymptotically stable when no deadlines are missed. The settling time is a standard control design parameter and we use it to deduce the exponential stability criterion in our method (ref. Section 5.1). We state below the idea of exponential stability that we consider as the notion of the system's stability [10].

Definition 1 ($(\boldsymbol{l},\boldsymbol{\epsilon})$*-Exponential Stability Criterion). A dynamical system, given by Eq.* (1)*, is said to be* $(\boldsymbol{l},\boldsymbol{\epsilon})$*-exponentially stable, if for a given* $\boldsymbol{\epsilon \in (0,1)}$ *and an* $\boldsymbol{l} \in \mathbb{N}$, $\frac{||x[k+l]||}{||x[k]||} < \epsilon$*, for every* $\boldsymbol{k} \in \mathbb{N}$*, i.e., every* $\boldsymbol{l}$*-length ratio of the state norm* $||\boldsymbol{x}||$ *(2-norm) decreases by a damping factor of* $\boldsymbol{\epsilon}$*.*

Intuitively, the above definition states that a reduction in the ratio of norms by $\boldsymbol{\epsilon}$ at every $\boldsymbol{l}$ sampling intervals leads the system's norm to become small over time (tending to zero) and ensures exponential stability. In discrete-time LTI systems, like we consider in our work, exponential stability is equivalent to other forms of stability, e.g., Lyapunov stability and bounded-input, bounded-output (BIBO) stability. Hence, these forms of stability are also accounted for in our method.

3.3 Weakly Hard Control Systems

In a resource-constrained embedded environment, allowing the tasks to miss a few of their deadlines occasionally, favors the task scheduling process and reduces resource overload, but the system performance and stability must not be inhibited [10, 16]. In a hard real-time setting, where all the control tasks need to meet their respective deadlines, it often becomes arduous to abide by such stringent requirements, and therefore, the concept of *weakly hard systems* eventuates. It is a much more lenient and flexible setting, provided that the number of deadline misses in a sequence of task invocations is bounded. For example, in a series of $\boldsymbol{k}$ consecutive deadlines, at least $\boldsymbol{m}$ of them must be met, which is popularly termed as the $\binom{\boldsymbol{m}}{\boldsymbol{k}}$*-firm weakly hard constraint*. There are many other such familiar constraints in this model, but we consider the $\binom{\boldsymbol{m}}{\boldsymbol{k}}$-firm constraint in our work.

The deadline miss is generally manifested as a *control execution skip*. We adhere to the *hold-and-kill* policy [15, 17] in this work, which means, for a control execution skip in a sampling interval $\boldsymbol{[k-1,k)}$, no new control input is computed and the plant state is updated using the last control input from the preceding iteration, i.e., $\boldsymbol{u[k] = u[k-1]}$. Considering the augmented state vector (both state variables and control input) as $\boldsymbol{z[k]} = \begin{bmatrix} \boldsymbol{x[k]} \\ \boldsymbol{u[k]} \end{bmatrix}$, the *closed loop dynamics matrices* $\mathcal{A}_{ns}$ and $\mathcal{A}_s$ in situations of *no execution skip* and an *execution skip*, are obtained as, $\mathcal{A}_{ns} = \begin{bmatrix} A_d & B_d \\ K & 0 \end{bmatrix}$

and $\mathcal{A}_s = \begin{bmatrix} A_d & B_d \\ 0 & I \end{bmatrix}$ respectively. Conforming to this, we define a *Control Execution Sequence (CES)* as a pattern of deadline hits and misses (equivalently, a pattern of no execution skip and execution skip) in a sequence of consecutive task invocations, where a deadline hit and miss are denoted by $\mathbf{1}$ and $\mathbf{0}$ respectively. For example, if $m = 2$ and $k = 5$, then one CES following the $\binom{3}{5}$-firm constraint, could be, $s = 11010$, for a time horizon of 5. Note, here the execution skips occurred at the third and fifth sampling intervals and the system evolves with this CES as follows, $z[k+5] = \mathcal{A}_s\mathcal{A}_{ns}\mathcal{A}_s\mathcal{A}_{ns}\mathcal{A}_{ns}z[k]$.

3.4 Control Safety

In case of weakly hard real-time settings, since execution skips are allowed deliberately, it is essential to determine how much the system deviates from its ideal behavior (i.e., following the sequence $111\cdots$), when it evolves following any weakly hard constraint. The notion of control safety that we consider in our work, is satisfying a given *safety bound* [15, 25] all the time. Ensuring only the system's stability is not sufficient for safety-critical systems that rely upon some weakly hard constraints. Therefore, we pursue integrating it with control safety and thereby designing a stable and safe system. The following terminologies defined below formalize the above arguments.

Definition 2 (*Nominal Trajectory). A nominal trajectory $\mathcal{N}$ is a sequence of state vectors $x[0], x[1], \cdots, x[T] \in \mathbb{R}^q$, or the state evolutions up to a finite time length T, where $x[0]$ is the initial state and $x[j]$ is calculated using Eq.* (1) *and* (2)*, for $j = 1$ to T.*

Note that the state evolution based on the ideal sequence $\mathbf{1}^T$ forms the nominal trajectory. Similarly, when execution skips are allowed following any $\binom{m}{k}$-firm constraint, we have a *CES-based trajectory* $\mathcal{C}$, where the system evolves with a CES that follows the given $\binom{m}{k}$-firm constraint (ref. the example at the end of the previous section). Note that the initial state of any such CES-based trajectory $\mathcal{C}$ is the same as that of $\mathcal{N}$. Let $x^n = (x_1^n, x_2^n, \cdots, x_q^n)$ and $x^c = (x_1^c, x_2^c, \cdots, x_q^c)$ denote the state vectors for trajectories $\mathcal{N}$ and $\mathcal{C}$ respectively, and *dis(.)* be the *Euclidean distance* on $\mathbb{R}^q$ (i.e., using the $||.||_2$-norm). Taking into account the infinite norm over time, the deviation of $\mathcal{C}$ from $\mathcal{N}$ is expressed as,

$$\Delta(\mathcal{N}, \mathcal{C}) = \max_{0 \le j \le T} dis(\mathcal{N}[j], \mathcal{C}[j]) = \max_{0 \le j \le T} ||x^n[j] - x^c[j]|| = \max_{0 \le j \le T} \sqrt{\sum_{i=1}^{q} (x_i^n[j] - x_i^c[j])^2} \quad (3)$$

Definition 3 (*Safety Requirement). A CES meets the safety requirement, if its trajectory $\mathcal{C}$ obeys the criterion: $\Delta(\mathcal{N}, \mathcal{C}) \le d^{safe}$, where d^{safe} is the given safety bound and $\mathcal{N}$ is the nominal trajectory.*

Since Δ is the maximum of the deviations at all time-points, thus, the quantity *dis($\mathcal{N}[j], \mathcal{C}[j]$)* is less than or equal to d^{safe}, at each time-point j.

3.5 Control Task Set

We aim to synthesize a safe and stable schedule for a set of n independent control tasks in a uniprocessor system, sharing a common processor. Let $T_C = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n\}$ be the set of n control tasks that are executed together following a set of CESs $\mathcal{P} = \{p_1, p_2, \cdots, p_n\}$. These CESs are obtained by satisfying some stability and safety criteria, which will be discussed in subsequent sections. Each task $\mathcal{T}_i$ is characterized by its *arrival time* (a_i), *sampling period* (h_i), *worst-case execution time (WCET)* (c_i) and *relative deadline* (d_i), where d_i is considered to be equal to the sampling period h_i. The position of deadline hits, i.e., a $\mathbf{1}$ in a CES p_i or $p_i[j] = 1$, indicates a job instance j of that task, whose arrival time $(a_{i,j})$, sampling period $(h_{i,j})$, WCET $(c_{i,j})$, relative deadline $(d_{i,j})$ and absolute deadline $(a_{i,j} + d_{i,j})$ are given by $j \times h_i$, h_i, c_i, d_i and

$(j+1) \times h_i$ (as $d_i = h_i$), respectively. For example, let $\mathcal{T}$ be a task with sampling period 3 ms and $p = 101$ be its CES, conforming to the stability and safety criteria. The jobs corresponding to the positions $p[0]$ and $p[2]$, have the arrival time, sampling period, relative and absolute deadlines as, (0 ms, 3 ms, 3 ms, 3 ms) and (6 ms, 3 ms, 3 ms, 9 ms), respectively. Moreover, for each job instance, we have three other parameters, namely, the *start time* (when a job begins execution), *finish time* (when a job completes execution) and the *response time* (difference between the finish time and the arrival time). For the j-th job of task $\mathcal{T}_i$, these parameters are denoted by $st_{i,j}$, $fin_{i,j}$ and $res_{i,j}$, and thus by definition, $res_{i,j} = fin_{i,j} - a_{i,j}$. The *worst-case response time (WCRT)* of a task $\mathcal{T}_i$ is the maximum of the response time of all its jobs, i.e., WCRT($\mathcal{T}_i$) = $\max_{j \in J(\mathcal{T}_i)} res_{i,j}$, where $J(\mathcal{T}_i)$ denotes the set of jobs of task $\mathcal{T}_i$. For the task $\mathcal{T}_i$ to be schedulable, WCRT($\mathcal{T}_i$) $\leq d_i$ must hold, for each i. The relevance of the parameters is elaborated while discussing schedulability in Section 5.3.

An Important Note: In this work, we consider an offline schedule, where we determine the ordering of task execution at the design time. Also, we use static controllers and do not re-design them by updating the sampling period to enhance the control performance or reduce the resource overload. Rather, keeping the sampling period fixed, we synthesize an $\binom{m}{k}$-firm constraint for each control task that ensures both the system's stability and safety while enhancing the control performance. Having the sampling periods and WCETs as fixed inputs, if we try to schedule all the n tasks in a hard real-time setting, then it may often lead to a scenario where all the tasks can no longer be scheduled on one processor. Even in a weakly hard setting, if the scheduler allows some of the jobs to miss their deadlines in order to obtain a feasible schedule, then the deadline hit-miss sequence returned by the scheduler may not satisfy the underlying stability and safety requirements. This may lead to diminished control performance. Hence, considering resource-constrained systems, we intentionally allow some execution skips for a control task to obtain its CES following an $\binom{m}{k}$-constraint. This CES complies with both the stability and safety criteria (ref. Definitions 1 and 3 respectively). Each of the n tasks is then scheduled on the processor following its CES.

4 A MOTIVATIONAL EXAMPLE

We now consider a case study of a *DC motor speed control system (MS)* and discuss its stability and safety constraints, to illustrate how the proposed method incorporates these aspects in task scheduling. MS is a second-order dynamical system that controls the rotational speed of the motor (output) by regulating the motor terminal voltage (control input). Since, it is a safety-critical application, an uncontrolled motor speed can lead to overheating and hazardous outcomes, hence, it is crucial to maintain a stable speed, within a safety limit. The system dynamics matrices, A, B and C (ref. Section 3.1) of the control system are given as, $A = \begin{bmatrix} -10 & 1 \\ -0.02 & -2 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$, $C = [1\ 0]$. The state variables denote the rotational speed and armature current. Suppose that MS is designed to achieve a reference speed of 0.5 rad s^{-1}. If there is a sudden increase in the speed (say it reaches to 1.2 rad s^{-1} at maximum) due to some external input, the speed should decrease and reach the reference, i.e., 0.5 rad s^{-1}, within the settling time (ref. Section 3.2), say 0.3 s. The discrete-time controller for this MS is implemented as a software control task in the embedded platform and it is sampled periodically with a time gap of h, say 10 ms (sampling period).

To ensure stability in a weakly hard real-time setting in our proposed method; we deduce the (l, ϵ)-exponential stability criterion (ref. Definition 1) from the settling time, maximum disturbance at input and the reference values, thereby deriving exactly one stability-oriented $\binom{m}{k}$-firm constraint (ref. Section 5.1). This guarantees that, if the system's state evolution follows any CES corresponding to that $\binom{m}{k}$, then the output reaches the desired reference within the settling time, after responding to a sudden change in the input. Figure 2 depicts scenarios of stable and unstable output responses

corresponding to $\binom{7}{10}$ and $\binom{5}{10}$ constraints respectively, when the system is simulated till a time horizon of 2 s. Suppose that the $\binom{7}{10}$-constraint is derived from the (l, ϵ)-stability criterion and we consider any CES following $\binom{7}{10}$ to simulate the system. We observe that the corresponding output (marked with red) reaches the reference, 0.5 rad s^{-1}, within the settling time, 0.3 s, after responding to a sudden increase in the speed to 1.2 rad s^{-1}. On the other hand, considering the $\binom{5}{10}$-constraint, the output not only fails to reach the reference within 0.3 s, but also shows a highly transient response initially (marked with blue), which may lead to undesirable performance. This implies that the underlying stability is not hampered if **3** or lesser deadlines are missed in a sequence of **10** deadlines (i.e., the $\binom{7}{10}$-constraint), however the system tends to be unstable on missing **5** or more deadlines. This substantiates the fact that the proposed method ensures the system's stability in a weakly hard setting, by confirming that the system stabilizes quickly after responding to sudden changes. Also, the exponential stability is guaranteed (complying with the (l, ϵ)-exponential stability criterion).

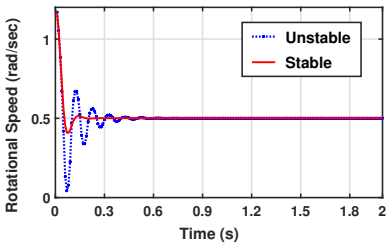


Fig. 2. Stable and unstable responses in MS

However, it is vital to ensure that the system's behavior should not deviate significantly from the ideal/nominal behavior (no deadline misses) before meeting the settling time requirement, i.e., during the transient phase. The notion of control safety considered in our work is characterized by bounding such deviations. In a weakly hard scenario, our goal is to ensure that the state trajectory $\boldsymbol{C}$, corresponding to some CES allowing a few deadline misses, always deviates from the nominal trajectory $\boldsymbol{\mathcal{N}}$, only by a bounded amount, i.e., $\Delta\ (\boldsymbol{\mathcal{N}}, \boldsymbol{C}) \leq \boldsymbol{d^{safe}}$ (ref. Definition 3). Let us here consider $\boldsymbol{d^{safe}}$ as **0.08**. Figure 3 outlines the behaviors of two trajectories for two different CESs, following the same stable $\binom{7}{10}$-constraint. The trajectory following the CES **1110111010** (marked with green) remains within the safety envelope of the nominal trajectory, whereas, the trajectory of another CES, **1000111111** (marked with red), fails to abide by the safety criterion as the corresponding deviation crosses the safety bound $\boldsymbol{d^{safe}}$. This may lead to fatal consequences in safety-critical applications. The unsafe behavior mainly arises due to the consecutive three deadline misses at the beginning for the second CES, leading to diminished control performance. We propose a strategy that prudently avoids such consecutive misses and we consider CESs that follow both stability and safety criteria (ref. Definitions 1,3).

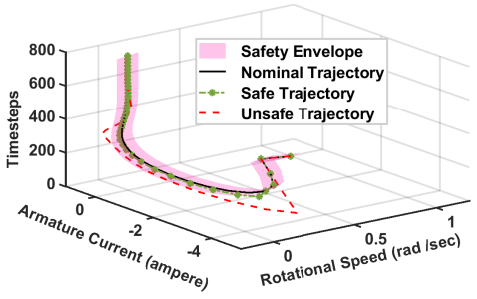


Fig. 3. Safe and unsafe state trajectories in MS

Since this work aims to generate a stable and safe schedule for a set of $\boldsymbol{n}$ control tasks in a uniprocessor system, hence for each task we ensure that the CES, following which the task is scheduled, should always obey the underlying stability and safety criteria. We finally obtain a feasible schedule for the $\boldsymbol{n}$ control tasks by minimizing the WCRT of each control task.

5 THE PROPOSED METHOD

We first formally define the problem statement and then discuss the proposed solution.

Problem Statement: For a set of $\boldsymbol{n}$ plant-control systems, given the following inputs: i) settling time and output reference of each control system, ii) a set of $\boldsymbol{n}$ control tasks (corresponding to the $\boldsymbol{n}$

plant-control systems) in a uniprocessor system, each characterized by its sampling period, WCET and safety bound; synthesize a feasible schedule of the tasks (if one exists), that ensures system's stability and safety over an infinite time horizon.

We solve the following two sub-problems to find a solution to the above problem.

Sub-problem 1: With the given set of inputs, determine a CES for each task that conforms with the stability and safety criteria, i.e., it ensures that the closed loop is $(\boldsymbol{l}, \boldsymbol{\epsilon})$-exponentially stable (ref. Definition 1 for stability) and the system's behavior under deadline misses, deviates from the nominal behavior (with no deadline miss) only by a bounded amount (ref. Definition 3 for safety).

Sub-problem 2: Given the set of $\boldsymbol{n}$ CESs (comply with stability and safety criteria in Sub-problem 1) for the $\boldsymbol{n}$ control tasks as input, generate a feasible uniprocessor schedule (if one exists), that is optimal w.r.t. WCRT while ensuring the system's stability and safety over an infinite time horizon.

The proposed method consists of three main steps. Figure 4 gives a synopsis of it, showing how the three aspects, *stability* (S_1), *safety* (S_2) and *schedulability* (S_3) are taken into consideration in our work. Steps 1 and 2 propose a solution to Sub-problem 1, while Step 3 offers a solution to Sub-problem 2. We demonstrate each of these steps below.

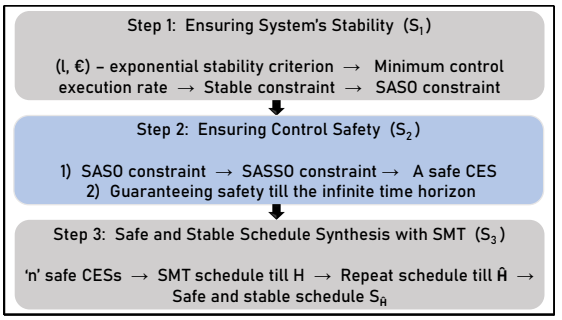


Fig. 4. Stepwise summary of the method.
[*SASO:* Scheduling-Affable Stability-Oriented Constraint,
SASSO: Scheduling-Affable Stability-and-Safety-Oriented Constraint.]

5.1 Step 1: Ensuring System's Stability

The first step is initiated with the concepts and theory of stability (S_1). We consider the notion of $(\boldsymbol{l}, \boldsymbol{\epsilon})$-exponential stability in this work. In order to exhibit its correlation with the settling time [10], we proceed as follows.

As mentioned earlier, settling time is the required time by which the system's output reaches and remains around the reference value (e.g., within 5 % error band). It is analyzed using the system's step response, i.e., how quickly the system responds to a sudden change in the input and then stabilizes towards the actual reference. Let $\boldsymbol{x} = (x_1, x_2, \cdots, x_q)$ be the state vector. Without loss of generality, we consider that in the presence of some external input or in an unsettled scenario, the system may operate in a larger region $||x|| = \sqrt{x^T x} \leq \xi_s + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is the maximum amount of deviation allowed in an unsettled situation. However, by the settling time, the operating region must be confined only to $||x|| = \sqrt{x^T x} \leq \xi_s$, i.e., the operating region in the settled scenario is $\sqrt{x^T x} \leq \xi_s$ ($||.||_2$ is referred to as $||.||$). Here, ξ_s is the radius of the reference region and it is obtained from the output reference (the desired value that the system tries to achieve) as follows. Let the output vector be $\boldsymbol{y} = (x_1, x_2, \cdots, x_w)$, for some some $\boldsymbol{w} \leq \boldsymbol{q}$, and the output reference vector be $\tilde{\xi} = (\hat{\xi_1}, \hat{\xi_2}, \cdots, \hat{\xi_w})$. Obeying the respective output references $\hat{\xi_i}$, ξ_s is chosen as $\sqrt{\sum_{i=1}^{w} \hat{\xi_i}^2}$. Note that for the case of a single output variable (i.e., $\tilde{\xi}$ is a scalar), ξ_s is equal to the output reference $\tilde{\xi}$.

Now, we establish the $(\boldsymbol{l}, \boldsymbol{\epsilon})$-exponential stability criterion (ref. Definition 1) from the settling time, which is primarily motivated by [10]. We consider that the system evaluation starts (i.e., at time-point **0**) in some perturbed/unsettled scenario. As the system reaches the desired output reference within the settling time, hence, by the construction of ξ_s as described above, the operating region, $||x|| \leq \xi_s$, is reached from the region, $||x|| \leq \xi_s + \boldsymbol{\delta}$, within the settling time T_s, i.e., within $N_h = \lceil \frac{T_s}{h} \rceil$ samples ($\boldsymbol{h}$ is the sampling period). This implies, $||x[N_h]|| \leq \xi_s$ with $||x[0]|| = \xi_s + \boldsymbol{\delta}$ (as initially the maximum value $\xi_s + \boldsymbol{\delta}$ is reached), and hence, $\frac{||x[N_h]||}{||x[0]||} \leq \frac{\xi_s}{\xi_s + \boldsymbol{\delta}}$. Thus, we consider

the damping ratio as $\frac{\xi_s}{\xi_s+\delta}$. Based on the values of T_s and h, N_h can be arbitrarily large, hence, to work with a computationally viable quantity, N_h is tuned by a constant, $f > 1$, and we set $l = \lceil \frac{N_h}{f} \rceil$. Intuitively, rather than considering a larger length of N_h to ensure that $\frac{||x[N_h]||}{||x[0]||} \leq \frac{\xi_s}{\xi_s+\delta}$ holds, we consider smaller intervals, like $[0, l), [l, 2l), \cdots$, to guarantee that, for some $\epsilon < 1$ (to be determined), $\frac{||x[l]||}{||x[0]||} < \epsilon$, $\frac{||x[2l]||}{||x[l]||} < \epsilon$, etc. holds. This implies $\frac{||x[N_h]||}{||x[0]||} < \epsilon^f$. To ensure that $\frac{||x[N_h]||}{||x[0]||} \leq \frac{\xi_s}{\xi_s+\delta}$ holds, we tune the damping ratio, $\frac{\xi_s}{\xi_s+\delta}$, to obtain the modified damping ratio, $\epsilon = (\frac{\xi_s}{\xi_s+\delta})^{\frac{1}{f}}$. Note that ensuring this (l, ϵ)-pair, the $(N_h, \frac{\xi_s}{\xi_s+\delta})$-stability criterion is automatically satisfied, since (l, ϵ) is much stricter than $(N_h, \frac{\xi_s}{\xi_s+\delta})$. This is how we derive the (l, ϵ)-exponential stability pair from the settling time.

Next we construct an $\binom{M}{K}$-firm constraint that conforms with the (l, ϵ)-stability criterion. Let r be the *rate of successful control execution* or the *percentage of deadline hits* over an infinite horizon and it is derived as follows. If β is the *minimum decay factor* (a tuning parameter for the damping ratio ϵ), h is the sampling period, ρ is the spectral radius of a matrix and $\chi_0 = \rho(\mathcal{A}_s)^2$, $\chi_1 = \rho(\mathcal{A}_{ns})^2$, $\beta = \frac{\ln(\frac{1}{\epsilon})}{l \times h}$, then the inequality $\frac{2\ln(\beta)+\ln(\chi_0)}{\ln(\chi_0)-\ln(\chi_1)} \leq r \leq 1$ holds [10]. The min value of r is $\frac{2\ln(\beta)+\ln(\chi_0)}{\ln(\chi_0)-\ln(\chi_1)}$.

Example 1. *Let us consider a cruise control system, where the reference speed (output variable) of the car is* $25\,\mathrm{km\,h^{-1}}$ *(ξ_s) and the speed can increase to* $70\,\mathrm{km\,h^{-1}}$ *($\xi_s + \delta$) at maximum. If the settling time (T_s) is* 1 s *and the speed goes high ($\leq 70\,\mathrm{km\,h^{-1}}$), the cruise control should guarantee that it decreases to* $25\,\mathrm{km\,h^{-1}}$ *within* 1 s*, i.e., within* $N_h = \lceil \frac{T_s}{h} \rceil = \lceil \frac{1\,\mathrm{s}}{0.01\,\mathrm{s}} \rceil = 100$ *samples, assuming* $h = 0.01$ s*. As* 100 *is a large value, we select a smaller and feasible value like* $l = \lceil \frac{N_h}{f} \rceil = \lceil \frac{100}{5} \rceil = 20$ *(with the tuning parameter* $f = 5$*). The modified damping ratio is* $\epsilon = (\frac{\xi_s}{\xi_s+\delta})^{\frac{1}{f}} = (\frac{25}{70})^{\frac{1}{5}} = 0.812$*, and hence the norm* $||x[k]||$ *decreases by a factor of* 0.812 *in every* $l = 20$ *sampling intervals. The minimum decay factor is obtained as,* $\beta = \frac{\ln(\frac{1}{\epsilon})}{l \times h} = 1.044$*. For* $\chi_0 = 1.007$ *and* $\chi_1 = 0.890$*, we obtain* $\frac{2\ln(\beta)+\ln(\chi_0)}{\ln(\chi_0)-\ln(\chi_1)} = 0.753$*. Thus,* $0.753 \leq r \leq 1$ *and min value of* r *is* 0.753*.*

We set forth the next definition with the above expositions.

Definition 4 (*Stable Constraint). The $\binom{M}{K}$-firm constraint obtained from the (l, ϵ)-stability criterion is said to be a stable constraint, when* $M = \lceil (\frac{2\ln(\beta)+\ln(\chi_0)}{\ln(\chi_0)-\ln(\chi_1)}) \times l \rceil$ *and* $K = l$*.*

This clearly shows that allowing $K - M$ execution skips in a sequence of K executions does not inhibit the system's stability. In Example 1, we obtain $\binom{16}{20}$ as the stable constraint for the cruise control system. Since $\beta = \frac{\ln(\frac{1}{\epsilon})}{l \times h}$, as mentioned above, hence $\epsilon = e^{-lh\beta}$. As $\frac{||x[k+l]||}{||x[k]||} < \epsilon$ for every $k \in \mathbb{N}$, as per the (l, ϵ)-exponential stability criterion (ref. Definition 1), hence the norm $||x[k]||$ tends to zero exponentially. Thus, the stable constraint $\binom{M}{K}$ makes the system exponentially stable.

Since our work focuses on synthesizing a safe and stable schedule, we develop a strategy to *quicken the scheduling process, maintaining the underlying stability and enhancing the control performance* simultaneously, specifically by constructing another $\binom{m}{k}$ from the stable $\binom{M}{K}$-constraint. The parameter l in the (l, ϵ)-pair is generally large (in range 20-40) and working with such large values of M and K (since $K = l$), becomes computationally intractable. For this, we construct another $\binom{m}{k}$ from the stable $\binom{M}{K}$-constraint, such that m divides M, k divides K and both m and k are much smaller than M and K respectively. Let us consider the stable constraint, $\binom{M}{K} = \binom{16}{20}$, for the cruise control system, as shown in Example 1. Let $\binom{m}{k} = \binom{4}{5}$ be the smaller constraint and p be a CES corresponding to $\binom{4}{5}$. If p is repeated 4 times, we obtain a CES of length 20 satisfying $\binom{16}{20}$, since p obeys the $\binom{4}{5}$-constraint. Note that the $\binom{4}{5}$-constraint aids to the stability of the system

by generating the stable constraint $\binom{16}{20}$, complying with the above argument. The construction of such an $\binom{m}{k}$ helps in two ways, first by rendering smaller values of $\boldsymbol{m}$ and $\boldsymbol{k}$ (**4** and **5** here), thereby making the scheduling method time-efficient (discussed in Section 5.3). Secondly, it expunges the case of consecutive **0**s indicating the fact of continuous deadline misses which may increase the chance of violating the safety criterion (crossing the bound $\boldsymbol{d^{safe}}$) and diminish the control performance. In our example, consecutive four 0s could occur in a CES following the $\binom{16}{20}$ constraint, but this can never occur if the CES $\boldsymbol{p}$ is repeated **4** times. Keeping all these factors in mind, we design this new $\binom{m}{k}$-constraint and formally define it below.

Definition 5 (*Scheduling-Affable Stability-Oriented Constraint (SASO)). An $\binom{m}{k}$-firm constraint is called as SASO if it is obtained from the stable constraint $\binom{M}{K}$, such that, $\boldsymbol{m}$ and $\boldsymbol{k}$ are relatively small divisors of $\boldsymbol{M}$ and $\boldsymbol{K}$ respectively.*

As the SASO constraint makes the scheduling process time-efficient (ref. Section 5.3) and also contributes to the system's stability, hence we use the terms 'scheduling-affable' and 'stability-oriented' in its nomenclature. The SASO constraint is an input to Step 2, as discussed next.

5.2 Step 2: Ensuring Control Safety

Now, we proceed to the second step of the method, which enforces control safety ($\boldsymbol{S_2}$) on top of control stability. Considering a SASO constraint $\binom{m}{k}$ for a control system, we first examine whether there exists a CES following that $\binom{m}{k}$, with exactly $\boldsymbol{(k\text{-}m)}$ execution skips, which satisfies the safe bound $\boldsymbol{d^{safe}}$. This is done by checking whether the deviation between any such CES-based state trajectory and the nominal trajectory $\boldsymbol{\mathcal{N}}$ (ref. Eq. (3)), is less than or equal to $\boldsymbol{d^{safe}}$. If such a CES is found, then it is evident that this $\binom{m}{k}$ potentially contributes to the system's safety. The term '*potentially*' highlights the fact that at least one of the CESs conforming to that $\binom{m}{k}$ satisfies the safety bound but not all the CESs, hence, we may not declare the entire SASO constraint to be safe, rather we define the next two terminologies based on these explanations.

Definition 6 (*Safe CES). A CES $\boldsymbol{p}$ is called as safe if its corresponding state trajectory $\boldsymbol{C_p}$ deviates from the nominal trajectory $\boldsymbol{\mathcal{N}}$ by an amount, that does not exceed the safe bound $\boldsymbol{d^{safe}}$, i.e., $\boldsymbol{\Delta\,(\mathcal{N}, C_p) \leq d^{safe}}$.*

Definition 7 (*Scheduling-Affable Stability-and-Safety-Oriented Constraint (SASSO)). An $\binom{m}{k}$-firm constraint is called as SASSO, if it is SASO and there exists at least one safe CES with exactly $\boldsymbol{m}$ **1**s in a length of $\boldsymbol{k}$.*

As $\binom{m}{k}$ denotes *at least* $\boldsymbol{m}$ deadline hits in a sequence of $\boldsymbol{k}$ consecutive control executions, thus $\binom{m'}{k}$ always implies $\binom{m}{k}$, for any $\boldsymbol{m < m' \leq k}$. Therefore, if no safe CES with exactly $\boldsymbol{m}$ **1**s in a length of $\boldsymbol{k}$ is obtained, after probing through ${}^{k}C_{m}$ such CESs, we increase the number of **1**s to $\boldsymbol{m+1}$ and iterate the same process of safety verification with the $\binom{m+1}{k}$ constraint. Starting with a SASO constraint $\binom{m}{k}$, continuing the same process with an increasing value of $\boldsymbol{m}$, let $\binom{m'}{k}$ be the constraint for which finally a safe CES with $\boldsymbol{m'}$ **1**s in a length of $\boldsymbol{k}$ is found. We then consider $\binom{m'}{k}$ as a SASSO constraint. For example, the CES $\boldsymbol{p = 11011}$ following the SASO constraint $\binom{4}{5}$ (obtained from the stable constraint $\binom{16}{20}$ in Example 1) obeys the safety bound, $\boldsymbol{d^{safe} = 0.06}$, for the cruise control system, since its trajectory $\boldsymbol{C_p}$ satisfies inequality $\boldsymbol{\Delta(\mathcal{N}, C_p) \leq d^{safe}}$ ($\boldsymbol{\Delta(.)}$ as in Eq. (3)). Hence, **11011** is a safe CES and $\binom{4}{5}$ is SASSO constraint. Herewith, we obtain a solution to Sub-problem 1, because the safe CES obtained from a SASSO constraint, for each control task, complies with both the stability and safety criteria.

Now, we discuss our mathematical findings related to control safety. Eq. (3) states that the safety verification is performed till the time horizon T, ensuring that the distance between the two trajectories is lesser or equal to d^{safe}, for each $j = 0$ to T. In this regard, we try to address the following three research questions.

Q1. Is the control safety requirement guaranteed merely till the time-point T or over an infinite time horizon?

Q2. Is there any upper bound on the time horizon length for verifying the control safety (since we cannot do it over an unbounded horizon)?

Q3. Can we obtain the exact time-point T, which is sufficient to guarantee control safety?

To answer Q1, we establish Theorem 1, where we compute an upper bound on the deviation, $\Delta(\mathcal{N}, C)$ (ref. Eq. (3)), for any safe CES-based trajectory C and the nominal trajectory $\mathcal{N}$. For example, as mentioned above, the safe CES **11011** corresponds to the SASSO constraint $\binom{4}{5}$, which is deduced from the stable $\binom{M}{K} = \binom{16}{20}$ (ref. Example 1). Hence, $l = K = 20$ and the state norm $||x||$ reduces by a factor of ϵ in every 20 sampling intervals. We use this (l, ϵ)-stability criterion to derive an upper bound on the distance between the two trajectories, i.e., $dis(\mathcal{N}[j], C[j])$, at each time-point j. Finally, we prove that this upper bound tends to zero as time increases, hence the actual distance also approaches zero. This clarifies that both the trajectories merge and the corresponding deviation becomes zero after some time, thereby establishing control safety over an infinite time horizon.

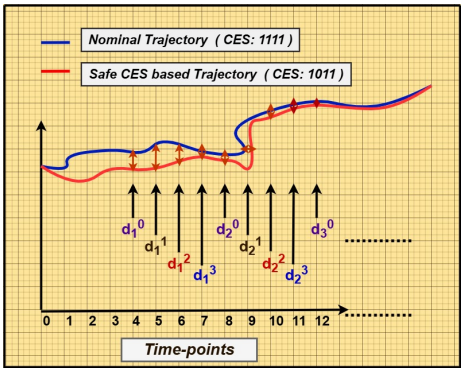


Fig. 5. Distance d_i^τ between the trajectories for $l = 4$, $\mu = 3$

Theorem 1. *A safe CES p obtained from a SASSO constraint $\binom{m}{k}$, that conforms with the stability and safety criteria, furnishes control safety over an infinite time horizon.*

Proof. To prove the claim, we first express all the arguments till a finite time-point $T_{ub} = \mu \times l$, for some $\mu \in \mathbb{N}$ and the parameter l comes from the (l, ϵ) pair. Next, we establish the control safety till the infinite horizon. Let $\mathcal{N}$ and C_p, respectively, be the nominal trajectory and a CES-based trajectory for a safe CES p, with exactly m 1s in length k, where $\binom{m}{k}$ is a SASSO constraint. Let $x^n[j] = \{x_1^n[j], x_2^n[j], \cdots, x_q^n[j]\}$ and $x^c[j] = \{x_1^c[j], x_2^c[j], \cdots, x_q^c[j]\} \in \mathbb{R}^q$, represent the respective state vectors for the two trajectories $\mathcal{N}$ and C_p respectively, at the j-th time-point. Also, the initial states of both the trajectories are equal, i.e., $x^n[0] = x^c[0]$. Since, $\mathbb{R}^q$ is a metric space, at any time-point j, the distance between $\mathcal{N}$ and C_p is written in terms of the vector norms as,

$$dis(\mathcal{N}[j],\ C_p[j]) = \sqrt{\sum_{v=1}^{q}(x_v^n[j] - x_v^c[j])^2} = ||\ x^n[j] - x^c[j]\ || \ \leq \ ||\ x^n[j]\ || + ||\ x^c[j]\ ||\ . \quad (4)$$

Here, we introduce the notation d_i^τ, such that, $d_i^\tau = dis(\mathcal{N}[il + \tau],\ C_p[il + \tau])$ represents the distances at various time-points. For the time-points $j = l, 2l, \cdots, \mu l$, the distances are calculated as $d_1^0, d_2^0, \cdots, d_\mu^0$. Similarly, at time-points $j = l{+}1, 2l{+}1, \cdots, (\mu{-}1)l{+}1$, the distances are $d_1^1, d_2^1, \cdots, d_{\mu-1}^1$, respectively. Continuing in this fashion, at time-points $j = l{+}(l{-}1), 2l{+}(l{-}1), \cdots, (\mu - 1)l{+}(l{-}1)$, d_i^τ becomes $d_1^{l-1}, d_2^{l-1}, \cdots, d_{\mu-1}^{l-1}$, respectively. Let us illustrate this by considering an example with $l = 4$. We represent the distances d_i^τ for $\mu = 3$ (say), i.e., up to the length $T_{ub} = 3l = 12$. Therefore, d_i^τ becomes d_1^0, d_2^0, d_3^0 at $j = 4, 8, 12$; d_1^1, d_2^1 at $j = 5, 9$; d_1^2, d_2^2 at $j = 6, 10$;

and d_1^3, d_2^3 at $j = 7, 11$, respectively. Figure 5 clarifies the same. Note that the (l, ϵ)-exponential stability criterion in Definition 1 mentions that the norm of the state vector is reduced by a factor of ϵ in every l sampling intervals, i.e, $\frac{||\ x[k+l]\ ||}{||\ x[k]\ ||} < \epsilon$, for every $k \in \mathbb{N}$. Hence, to capture the (l, ϵ)-criterion on the distances d_i^τ, the length l has to be reached at least. Using In-Eq. (4) and Definition 1, we formally write the set of inequalities, for $j = l, 2l, \cdots, \mu l$ as follows.

$$d_1^0 = dis(\mathcal{N}[l], C_p[l]) \leq ||\ x^n[l]\ || + ||\ x^c[l]\ || \leq \epsilon\ (\ ||\ x^n[0]\ || + ||\ x^c[0]\ ||\ )$$
$$= 2\epsilon ||\ x^n[0]\ ||\quad (as,\ x^n[0] = x^c[0])\ .$$

$$d_2^0 = dis(\mathcal{N}[2l], C_p[2l]) \leq ||\ x^n[2l]\ || + ||\ x^c[2l]\ || \leq \epsilon(\ ||\ x^n[l]\ || + ||\ x^c[l]\ ||\ )$$
$$\leq \epsilon^2(\ ||\ x^n[0]\ || + ||\ x^c[0]\ ||\ ) = 2\epsilon^2 ||\ x^n[0]\ ||.$$

Proceeding likewise, we get,

$$d_\mu^0 = dis(\mathcal{N}[\mu l], C_p[\mu l]) \leq\ 2\epsilon^\mu ||\ x^n[0]\ ||\ . \tag{5}$$

Similarly, we consider the time-points, $j = l{+}1, 2l{+}1, \cdots, (\mu{-}1) \times l{+}1$, and formulate another series of in-equations, using the fact that, either $x^c[1] = \mathcal{A}_s x^c[0]$ or $x^c[1] = \mathcal{A}_{ns} x^c[0]$ based on there is/isn't an execution skip. We assume without loss of generality that $||\mathcal{A}_{ns}|| \geq ||\mathcal{A}_s||$. Hence, for an execution skip,

$$d_1^1 = dis(\mathcal{N}[l+1],\ C_p[l+1]) \leq ||\ x^n[l+1]\ || + ||\ x^c[l+1]\ || \leq \epsilon\ (\ ||\ x^n[1]\ || + ||\ x^c[1]\ ||\ )$$
$$\leq \epsilon\ (\ ||\mathcal{A}_{ns}||\ ||\ x^n[0]\ || + ||\mathcal{A}_s||\ ||\ x^c[0]\ ||\ ) \leq \epsilon\ ||\mathcal{A}_{ns}||\ (\ ||\ x^n[0]\ || + \frac{||\mathcal{A}_s||}{||\mathcal{A}_{ns}||}\ ||\ x^c[0]\ ||\ )$$
$$\leq 2\epsilon\ ||\mathcal{A}_{ns}||\ ||\ x^n[0]\ ||\qquad (since,\ ||\mathcal{A}_{ns}|| \geq ||\mathcal{A}_s||\ \ implies\ \ \frac{||\mathcal{A}_s||}{||\mathcal{A}_{ns}||} \leq 1)$$

For no execution skip, $d_1^1 \leq \epsilon\ (\ ||\mathcal{A}_{ns}||\ ||\ x^n[0]\ || + ||\mathcal{A}_{ns}||\ ||\ x^c[0]\ ||\ ) = 2\epsilon\ ||\mathcal{A}_{ns}||\ ||\ x^n[0]\ ||\ .$

Thus, for both the cases of execution skip and no skip, we can state, $d_1^1 \leq 2\epsilon\ ||\mathcal{A}_{ns}||\ ||\ x^n[0]\ ||$, and with similar arguments, $d_2^1 \leq 2\epsilon^2\ ||\mathcal{A}_{ns}||\ ||\ x^n[0]\ ||$ and finally,

$$d_{\mu-1}^1 \leq 2\epsilon^{\mu-1}\ ||\mathcal{A}_{ns}||\ ||\ x^n[0]\ ||\ . \tag{6}$$

Generalizing the above pattern and using In-Eq. (5) and (6), we obtain,

$$d_i^\tau = dis(\mathcal{N}[il+\tau],\ C_p[il+\tau]) \leq\ 2\epsilon^i\ ||\mathcal{A}_{ns}||^\tau\ ||\ x^n[0]\ ||\ . \tag{7}$$

where, $i = 1$ to μ, for $\tau = 0$ and $i = 1$ to μ-1, for $\tau = 1$ to l-1. If $||\mathcal{A}_s|| \geq ||\mathcal{A}_{ns}||$, then In-Eq. (7) is modified as $d_i^\tau \leq 2\epsilon^i ||\mathcal{A}_s||^\tau ||x^n[0]||$. For very large values of μ, i.e., when $\mu \to \infty$, as $\epsilon < 1$, the factor $\epsilon^i \to 0$ (the max value of i is μ for $\tau = 0$ and μ-1 for $\tau = 1$ to l-1). Hence, the distance d_i^τ eventually approaches zero. This proves that trajectories $\mathcal{N}$ and C_p merge after a time-point. □

In this fashion, we not only establish the safety till the infinite horizon but also obtain an upper bound on the distance d_i^τ using the (l, ϵ)-exponential stability criterion. As, d_i^τ has to be less than d^{safe} to ensure the control safety, this exhibits a correlation between safety and stability. Next, to answer Q2, i.e., for obtaining the value of T_{ub}, we proceed as follows.

Corollary 1. *The upper bound on the time horizon, for verifying that $\Delta(\mathcal{N}, C) \leq d^{safe}$ holds, is the time-point T_{ub}, where T_{ub} is given by,* $\left\lceil \frac{\ln(2\ ||\mathcal{A}_{ns}||^{l-1}\ ||x^n[0]||\ ) - \ln(d^{safe})}{|\ln \epsilon|} \right\rceil \times l.$

Proof. Generally, $||\mathcal{A}_{ns}|| > 1$ and hence, $||\mathcal{A}_{ns}||^{\tau}$ increases for larger values of τ, the largest value being $||\mathcal{A}_{ns}||^{l-1}$ (as $\tau \leq l$-1). This shows, amongst the time-points, $j = (\mu\text{-}1)l+1, (\mu\text{-}1)l+2, \cdots, (\mu\text{-}1)l+(l\text{-}1)$, the upper bound, on the distance d_i^{τ} (from In-Eq. (7)), is maximum at the point $(\mu\text{-}1)l+(l\text{-}1)$, i.e., for $d_{\mu\text{-}1}^{l\text{-}1}$. Though it is a loose upper bound, still we specify it to be less than the given bound d^{safe}, in order to evaluate T_{ub}. Following this, we write,

$$d_{\mu-1}^{l-1} \leq 2\epsilon^{\mu-1} ||\mathcal{A}_{ns}||^{l-1} || x^n[0] || \leq d^{safe} \Rightarrow (\mu-1) \times \ln \epsilon \leq ln \left(\frac{d^{safe}}{2 ||\mathcal{A}_{ns}||^{l-1} || x^n[0] ||} \right)$$

$$\Rightarrow (\mu-1) \geq \frac{\ln \left(\frac{2 ||\mathcal{A}_{ns}||^{l-1} ||x^n[0]||}{d^{safe}} \right)}{\ln \frac{1}{\epsilon}} \Rightarrow \mu \geq \left\lceil \frac{\ln (2 ||\mathcal{A}_{ns}||^{l-1} ||x^n[0]||) - \ln (d^{safe})}{|\ln \epsilon|} \right\rceil = \alpha .$$

At the start of the proof of Theorem 1, we initially considered the finite horizon $T_{ub} = \mu \times l$ for any $\mu \in \mathbb{N}$. Now, using the above result, the min value of μ becomes α and we set $T_{ub} = \alpha \times l$, thereby completing the proof. □

Note that the value of T_{ub} is not the exact one, but rather a loose approximation of the time length for the control safety verification. This is because the triangle inequalities (for norms) are used to obtain the upper bound of In-Eq. (7). In practice, it is observed that the trajectories merge much before $T_{ub} = \alpha \times l$, when the system is not in its transient phase, i.e., when it is closer to the settling time. Yet, a key point here is that, T_{ub} is the maximum time-point for verifying the control safety. It is absolutely superfluous to check beyond that and this is how we come up with a theoretical upper bound on the time horizon length in Corollary 1. Nonetheless, we derive another riveting result that helps in avoiding the safety verification till $\alpha \times l$ and do it till the settling time only. This leads to the answer to Q3. The upper bound in Corollary 1 is directly obtained as a consequence of Theorem 1, correlating the stability and safety notions. Now, we establish a connection between the control safety and the settling time to provide an exact value of the horizon length, which is much less than $\alpha \times l$.

Theorem 2. *Settling time is the exact length of the time horizon for performing the safety verification, i.e., to check whether the deviation $\Delta(\mathcal{N}, \mathcal{C})$, between the nominal trajectory $\mathcal{N}$ and any CES-based trajectory $\mathcal{C}$, is less than the bound d^{safe}.*

Proof. When the system is in a settled and disturbance-free mode, i.e., for all time-points $t \geq T_s$, where T_s is the settling time, it remains within the region $\sqrt{x^T x} \leq \xi_s$ (ref. Section 5.1). Since we deduce the stable constraint $\binom{M}{K}$ from the (l, ϵ)-stability criterion, which in turn is derived using the settling time and output reference values, hence, $\sqrt{x^T x} \leq \xi_s$ holds for all $t \geq T_s$ and for all trajectories corresponding to a stable constraint $\binom{M}{K}$, i.e., the nominal trajectory $\mathcal{N}$ and also any CES-based trajectory $\mathcal{C}$ following this $\binom{M}{K}$. Let the corresponding state vectors for $\mathcal{N}$ and $\mathcal{C}$ be $x^n = (x_1^n, x_2^n, \cdots, x_q^n)$ and $x^c = (x_1^c, x_2^c, \cdots, x_q^c)$ respectively. Without loss of generality, let the output variable be the first variable of the state vector, i.e., output $y(t) = x_1^n(t) = x_1^c(t) = \xi_s, \forall t \geq T_s$. Applying inequality $\sqrt{x^n(t)^T x^n(t)} \leq \xi_s$, for $\mathcal{N}$, we get,

$$\sum_{i=1}^{q} x_i^n(t)^2 \leq \xi_s^2 \Rightarrow \xi_s^2 + \sum_{i=2}^{q} x_i^n(t)^2 \leq \xi_s^2 \Rightarrow \sum_{i=2}^{q} x_i^n(t)^2 \leq 0$$

$$\Rightarrow x_i^n(t) = 0, \;\; 2 \leq i \leq q, \;\; \forall t \geq T_s \;\; (as, \; x_1^n(t) = \xi_s) \tag{8}$$

Similarly for $\mathcal{C}$, $\sqrt{x^c(t)^T x^c(t)} \leq \xi_s$ implies $x_i^c(t) = 0, \; 2 \leq i \leq q, \forall t \geq T_s$. Hence, the deviation between $\mathcal{N}$ and $\mathcal{C}$ is actually zero after the settling time T_s. Similar arguments also work for multiple output variables, i.e., for $y(t) = (x_1(t), x_2(t), \cdots, x_w(t))$, for some $w \leq q$.

Also, it may be the case that the output is of the form $\boldsymbol{y(t) = \zeta_1 x_1(t) + \zeta_2 x_2(t) + \cdots + \zeta_w x_w(t)}$, for some $\boldsymbol{w \leq q}$ and $\boldsymbol{\zeta_i}$ are given constants. Let the output reference be $\hat{\xi}_s$, such that, the affine hyperplane $\mathcal{P} : \zeta_1 x_1(t) + \zeta_2 x_2(t) + \cdots + \zeta_w x_w(t) = \hat{\xi}_s$ remains within the operating region $\sqrt{x(t)^T x(t)} \leq \xi_s$ for all $t \geq T_s$ and for all trajectories following a stable constraint $\binom{M}{K}$. Thus, for all $t \geq T_s$, for any point $(x_1(t), x_2(t), \cdots, x_w(t))$ on $\mathcal{P}$, the output $y(t)$ reaches $\hat{\xi}_s$. We consider the point $(\frac{\hat{\xi}_s}{w\,\zeta_1}, \frac{\hat{\xi}_s}{w\,\zeta_2}, \cdots, \frac{\hat{\xi}_s}{w\,\zeta_w})$, which lies on $\mathcal{P}$. Next, we ensure that the output variables $x_1(t)$, $x_2(t), \cdots, x_w(t)$, reach the references, $\frac{\hat{\xi}_s}{w\,\zeta_1}, \frac{\hat{\xi}_s}{w\,\zeta_2}, \cdots, \frac{\hat{\xi}_s}{w\,\zeta_w}$, respectively, for $t \geq T_s$. This implies that the output $y(t)$ reaches the reference $\hat{\xi}_s$ for $t \geq T_s$. With similar equations like Eq. (8), $x_1^n(t) = x_1^c(t) = \frac{\hat{\xi}_s}{w\,\zeta_1}$, $x_2^n(t) = x_2^c(t) = \frac{\hat{\xi}_s}{w\,\zeta_2}, \cdots, x_w^n(t) = x_w^c(t) = \frac{\hat{\xi}_s}{w\,\zeta_w}$, and $x_{w+1}^n(t) = x_{w+1}^c(t) = 0, \cdots, x_q^n(t) = x_q^c(t) = 0$, for all $t \geq T_s$. Thus, $\Delta(\mathcal{N}, C) = 0 < d^{safe}$ after the settling time T_s. □

With all the above deductions and findings, we conclude this phase and proceed next to construct a schedule, which is safe and stable over an infinite time horizon.

5.3 Step 3: Construction of a Safe and Stable Schedule with SMT

Fully utilizing the processor bandwidth in a uniprocessor platform while considering a hard real-time setting (i.e., when no execution skip/deadline miss is allowed), leads to a non-schedulable situation more often. That is why the weakly hard real-time setting (allowing occasional deadline misses) becomes a promising alternative for task scheduling. Having our focus on weakly hard control systems, given the sampling periods and WCET as inputs, we construct a schedule for the weakly hard control tasks guaranteeing the underlying stability and safety conditions. To generate such a feasible schedule, we leverage formal methods, particularly the concept of *Satisfiability Modulo Theories (SMT)*, a widely used constraint-solving technique. More specifically, we formulate an optimization problem to find a schedule that *minimizes the worst-case response time (WCRT)* of control tasks. This propels the jobs to execute as soon as they arrive if there is a scope to do so, and thereby increases the interval between the completion time and the absolute deadline of the job. Such a way of scheduling facilitates other jobs to execute during idle time and obtain a schedule even with very high processor utilization, which we show in Section 6.2 through extensive experimental evaluation. Like the response time, there is another analogous parameter, named *lateness*, which is either zero or negative for a feasible schedule. Larger negative values of the lateness of a job instance signify that the job is scheduled much earlier than its deadline, hence, minimized WCRT indicates the least value of lateness.

The scheduling problem is formulated as a set of constraints (clauses) and furnished to the SMT solver (of type *optimizer*), which in turn returns a feasible schedule as the output if the set of constraints is *satisfiable*. All such constraints are listed below, along with their descriptions. For the j-th job of the i-th task, the four parameters, namely, arrival time ($a_{i,j}$), sampling period ($h_{i,j}$), WCET ($c_{i,j}$) and relative deadline ($d_{i,j}$) are the given inputs (ref. Section 3.5) and the parameters, *start time* ($st_{i,j}$), *finish time* ($fin_{i,j}$) and *response time* ($res_{i,j}$), are determined by the solver in order to generate the feasible schedule.

Input to the SMT-Optimizer: Sampling period h_i, WCET c_i of task $\mathcal{T}_i$, $i = 1$ to n (ref. Section 3.5) and the set of n safe CESs coupled with the corresponding SASSO constraints.

Output from the SMT-Optimizer: A feasible schedule.

Feasibility-Related Constraints: For a CES with m 1s in a length of k, the m 1s indicate m job instances and we schedule each such job instance. For a schedulable job, the feasibility conditions are, that its start time must succeed its arrival time ($\mathcal{L}_1$), the finish time must precede its absolute deadline ($\mathcal{L}_2$) and the difference between the finish and start time must be equal to its WCET ($\mathcal{L}_3$).

For example, let $\boldsymbol{p = 1011}$ is a CES specifying a control task $\mathcal{T}_i$ with a sampling period and WCET as 4 ms and 2 ms respectively. Following $\boldsymbol{p}$, the three job instances arrive at the first, third and fourth sampling instants. The second job arrives at the third sampling instant, i.e., at $\boldsymbol{t = 8}$ ms (since the first job arrives at $\boldsymbol{t = 0}$ ms). Thus its parameters, $\boldsymbol{a_{i,2}}$, $\boldsymbol{h_{i,2}}$, $\boldsymbol{c_{i,2}}$ and $\boldsymbol{a_{i,2} + d_{i,2}}$ are equal to 8 ms, 4 ms, 2 ms and 12 ms respectively. Hence, it should begin execution after $\boldsymbol{t = 8}$ ms, execute for 2 ms and finish before $\boldsymbol{t = 12}$ ms. The three constraints below describe the above explanation.

$$\mathcal{L}_1 : \boldsymbol{st_{i,j} \geq a_{i,j}}\,, \qquad \mathcal{L}_2 : \boldsymbol{fin_{i,j} \leq a_{i,j} + d_{i,j}}\,, \qquad \mathcal{L}_3 : \boldsymbol{fin_{i,j} - st_{i,j} = c_{i,j}}.$$

Response Time-Related Constraints: For a feasible schedule, WCRT($\mathcal{T}_i$)= $\max_{j \in J(\mathcal{T}_i)} \boldsymbol{res_{i,j} \leq d_i}$ must hold (ref. Section 3.5). Since, for any $\boldsymbol{j}$-th job $\in \boldsymbol{J(\mathcal{T}_i)}$, $\boldsymbol{d_{i,j} = d_i}$, thus the schedule is feasible if the response time of each job is less than or equal to its deadline. For the $\boldsymbol{j}$-th job of task $\mathcal{T}_i$, the related constraints are,

$$\mathcal{L}_4 : \boldsymbol{res_{i,j} = fin_{i,j} - a_{i,j}}\,, \qquad \mathcal{L}_5 : \boldsymbol{0 \leq res_{i,j} \leq d_{i,j}}.$$

The *lateness* of the $\boldsymbol{j}$-th job of task $\mathcal{T}_i$ is defined as, $\boldsymbol{late_{i,j} = fin_{i,j} - (a_{i,j} + d_{i,j})}$. Clearly, it is either zero or a negative quantity following constraint $\mathcal{L}_2$.

Conflict Removing Constraints: For any two job instances $\boldsymbol{j_1}$ and $\boldsymbol{j_2}$ of two different control tasks, their execution intervals should be disjoint, since, exactly one job instance can be scheduled at a time-point, in a uniprocessor system. All possible cases of overlaps between $\boldsymbol{j_1}$ and $\boldsymbol{j_2}$, (ref. Figure 6), are outlined below as: *left partial overlap* ($\mathcal{L}_6$), *right partial overlap* ($\mathcal{L}_7$) and the *full overlap* ($\mathcal{L}_8$). These instances are avoided by the solver to remove the conflicts between two jobs, hence we write constraints in negation.

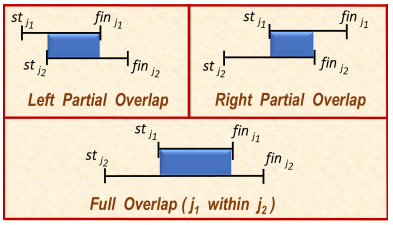


Fig. 6. Overlaps between jobs $\boldsymbol{j_1}$ and $\boldsymbol{j_2}$

$$\mathcal{L}_6 : \boldsymbol{Not(st_{j_1} < st_{j_2} < fin_{j_1})},\ \mathcal{L}_7 : \boldsymbol{Not(st_{j_1} < fin_{j_2} < fin_{j_1})},\ \mathcal{L}_8 : \boldsymbol{Not(st_{j_2} \leq st_{j_1}, fin_{j_1} \leq fin_{j_2})}$$

Objective Function for Optimization: For each task $\mathcal{T}_i$, $\boldsymbol{1 \leq i \leq n}$, WCRT($\mathcal{T}_i$) = $\max_{j \in J(\mathcal{T}_i)} \boldsymbol{res_{i,j}}$ is to be minimized to generate the schedule and this is formally expressed as follows.

$$\mathcal{L}_9 : \boldsymbol{Minimize} \max_{\substack{i = 1 \text{ to } n \\ j \in J(\mathcal{T}_i)}} \boldsymbol{res_{i,j}}.$$

Here, $\boldsymbol{J}$ denotes the set of jobs of a task. The entire set of constraints provided to the solver is the consolidated formula of the form of $\mathcal{L}$, given as follows,

$$\mathcal{L} = \left(\bigwedge_{1 \leq i \leq n} \bigwedge_{j \in J(\mathcal{T}_i)} \bigwedge_{1 \leq \gamma \leq 5} \mathcal{L}_\gamma \right) \wedge \left(\bigwedge_{j_1 \in J(\mathcal{T}_{i_1}),\ j_2 \in J(\mathcal{T}_{i_2}),\ i_1 < i_2} \bigwedge_{6 \leq \gamma \leq 8} \mathcal{L}_\gamma \right) \wedge \mathcal{L}_9.$$

Here, the first term encompasses constraints $\mathcal{L}_1 - \mathcal{L}_5$, for all job instances of each task in the set $\boldsymbol{T = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n\}}$. The second term exhibits the constraints, $\mathcal{L}_6 - \mathcal{L}_8$, for each pair of jobs $\boldsymbol{j_1}$ and $\boldsymbol{j_2}$, belonging to two different tasks $\mathcal{T}_{i_1}$ and $\mathcal{T}_{i_2}$ respectively. We eliminate the redundancy of checking the same set of constraints twice, for the pairs $(\boldsymbol{j_1}, \boldsymbol{j_2})$ and $(\boldsymbol{j_2}, \boldsymbol{j_1})$, hence we add the condition $\boldsymbol{i_1 < i_2}$. The last term indicates the optimization constraint $\mathcal{L}_9$. If $\mathcal{L}$ is satisfiable, then the SMT-optimizer reports a feasible schedule.

We generate the schedule $\boldsymbol{S_H}$ till a finite time-point, $\boldsymbol{H}$ = $\text{lcm}_{1 \leq i \leq n}(\boldsymbol{k_i \times h_i})$ where, $\binom{m'_i}{k_i}$ and $\boldsymbol{h_i}$ are the SASSO constraints and the sampling period of the $\boldsymbol{i}$-th control task respectively. Since, each pattern $\boldsymbol{p_i}$ repeats after a length of $\boldsymbol{k_i}$, it occurs an integer number of times within the horizon $\boldsymbol{H}$. Recall that a SASSO constraint $\binom{m'}{k}$ ensures the control safety over its underlying SASO constraint, which in turn is generated from a stable constraint $\binom{M}{K}$ for enhancing the time complexity of the

scheduling method, without compromising the stability. Also, this SASO constraint is a special case of the stable constraint, by construction. Therefore, to ensure both the safety and stability of a schedule, we should not restrict ourselves to consider the schedule S_H till H only, rather we should proceed towards $\hat{H} > H$, where the length K of the stable constraint $\binom{M}{K}$ is confirmed to be reached (since, the system norm reduces by the damping factor ϵ after every l sampling intervals by Definition 1 and $K = l$). This holds true for all the n control systems. Consequently, to acquire a safe and stable schedule, we consider the larger horizon, or the *hyper-period* $\hat{H} = \text{lcm}_{1 \leq i \leq n}(K_i \times h_i)$, which is some multiple of H, say b_H. The schedule $S_{\hat{H}}$, obtained till $\hat{H}$, is both safe and stable over an infinite time horizon, and it is just b_H many repetitions of S_H. Thus, the construction of the schedule till H suffices to obtain $S_{\hat{H}}$. Basically, we schedule a smaller number of job instances till the length H and precisely by replicating this ordering of the jobs, a larger number of job instances are actually scheduled over the larger length $\hat{H}$. This simultaneously makes the scheduling process faster and offers room to work with multiple tasks and job instances, which we also showcase through experimental evaluation. Apart from increasing the control performance (in comparison to that of a stable constraint $\binom{M}{K}$), the SASO constraints aid to the efficiency of the scheduling mechanism, in the above way. The obtained schedule $S_{\hat{H}}$ is optimal w.r.t. WCRT and also ensures the system's safety and stability over an infinite time horizon, thereby it furnishes a solution to Sub-problem 2. This wraps up the entire methodology and next, we substantiate the scalability and the efficiency of our proposed method, with some experimental observations.

6 EXPERIMENTAL RESULTS

This section describes the experimental details, along with the results and analysis, to establish the efficacy of the proposed method. At first, we delineate the proposed method using a case study considering **5** benchmark control systems from the automotive domain. Next, we expand on the scalability analysis of the proposed technique by considering **15** benchmark control systems, followed by the experimental comparison with the state-of-the-art works. Our objective is to demonstrate that the proposed method offers improved schedulability with a lower runtime, while ensuring the underlying stability and safety. Such features are accomplished by the proposed method specifically due to the following two factors: i) minimized WCRT while scheduling, and ii) pruned SMT search space to a large extent with the safe CESs obtained from the SASSO constraints. All the experiments described are carried out on a 64-bit Windows OS in a 2.10 GHz Intel Core-i5 machine, with 32 GB of RAM and we use MATLAB version R2020a and Z3 SMT solver [18] with Python API for the experiments.

6.1 Working Execution of the Proposed Method

For this experiment, the control systems considered are, a second-order *DC motor speed* (MS) control, a third-order *cruise control* (CC), a fourth-order *suspension control* (SC), a second-order *resistor-capacitor network* (RC) and a second-order lane-following controller for an *F1-tenth model car* (F1) that adjusts the steering angle [25]. The inputs to our method, viz., the system dynamics, sampling period (h), WCET (c), settling time (T_s), reference value (ξ_s), maximum deviation (δ) (ref. Section 5.1), safety bound (d^{safe}), are mentioned in Columns **2**-**8** of Table 1, respectively. If we consider a hard real-time setting, i.e., all jobs of all the tasks meet their respective deadlines, then by using the data of Columns **3** and **4**, the utilization value would result in, $U = \sum_{1 \leq i \leq 5} \frac{c_i}{h_i} = 1.17 > 1$ leaving some tasks unscheduled. Thus, a few execution skips are required to schedule the jobs in a uniprocessor system and this is one rationale for considering the weakly hard task setting. Hence, for this task setup, we need to formulate the corresponding weakly hard constraints. With the inputs in Columns **3**-**7**, the (l, ϵ)-pair and the control execution rate r are first calculated

(ref. Section 5.1) to generate the stable $\binom{M}{K}$ constraint (ref. Definition 4). Subsequently, following Definitions 5, 7 and 6, the SASO $\binom{m}{k}$, the SASSO $\binom{m'}{k}$ constraints and the safe CES $\boldsymbol{p}$ (using $\boldsymbol{d^{safe}}$ in Column 8 for the last two parameters) are evaluated, respectively. Columns **9-13** of Table 1 report all these parameters, which are computed in less than 1 ms, since obtaining stable and SASO constraints are constant time actions and the safety checking for ${}^{k}C_{m}$ many CESs, corresponding to an $\binom{m}{k}$, takes minimal amount of time for small values of $\boldsymbol{m}$ and $\boldsymbol{k}$. The deviation between the nominal and CES-based trajectories is computed for multiple initial states (**8-10** random points on the unit circle). The one leading to the minimum deviation is selected.

1	2	3	4	5	6	7	8	9	10	11	12	13
Plant	**System Dynamics**	**h (ms)**	**c (ms)**	T_s **(s)**	ξ_s	δ	d^{safe}	(l, ϵ)	$\binom{M}{K}$	$\binom{m}{k}$	$\binom{m'}{k}$	p
F1	A = [0 6.5; 0 0] B = [0; 19.685], C = [1 0]	20	4	0.3	0.1	0.04	0.56	(15, 0.7143)	$\binom{10}{15}$	$\binom{2}{3}$	$\binom{2}{3}$	**101**
SC	A = [0 1 0 0; -8 -4 8 4; 0 0 0 1; 80 40 -160 -60] B = [0; 80; 20; -1120], C = [1 0 0 0]	15	3	0.3	2	1.717	0.8	(20, 0.7361)	$\binom{15}{20}$	$\binom{3}{4}$	$\binom{3}{4}$	**0111**
CC	A = [0 1 0 ; 0 0 1; -6.0476 -5.2826 -0.238] B = [0; 0; 2.4767], C = [1 0 0]	10	2	1.2	30	75.5	0.06	(30, 0.7302)	$\binom{25}{30}$	$\binom{5}{6}$	$\binom{5}{6}$	**101111**
MS	A = [-10 1; -0.02 -2] B = [0; 2], C = [1 0]	20	5	0.6	0.1	0.101	0.1	(15, 0.7053)	$\binom{10}{15}$	$\binom{2}{3}$	$\binom{2}{3}$	**101**
RC	A = [-6 1; 0.2 -0.7] B = [5; 0.5], C = [1 0]	15	4	1.0	0.5	1.8	0.07	(16, 0.7452)	$\binom{12}{16}$	$\binom{3}{4}$	$\binom{3}{4}$	**1011**

Table 1. Input and output parameters of the control systems

Here, the horizon of the schedule, $H = \text{lcm}_{1 \le i \le 5}\{k_i \times h_i\}$ and the hyper-period $\hat{H} = \text{lcm}_{1 \le i \le 5}\{K_i \times h_i\}$ are equal to **60** and **1200** respectively. The synthesized schedule S_{60} has **15** job instances and since the safe and stable schedule S_{1200} is just **20** repetitions of S_{60}, hence, **300** job instances occur in total in S_{1200}. On receiving the input tuples $\{(h_i, c_i, \binom{m'_i}{k_i}, p_i)\}_{i=1}^{5}$, the solver synthesizes S_{60}, which has an average lateness value of **-8.067** (average over the lateness values of all the **15** jobs), ensuring that all the five tasks (specified by the SASSO constraints) are schedulable. The total time taken to obtain the schedule S_{60} (or S_{1200}) is 0.08 s.

CASE 1: 5 plants: (CC, MS, F1, SC, RC) H **= 60 ,** J_H**=15 ,** $\hat{H}$ **= 1200,** $J_{\hat{H}}$ **= 300**						
Util. range	0.7 - 0.8		0.8 - 0.9		0.9 - 1.0	
Actual Util.	0.73	0.76	0.80	0.82	0.9	0.93
Min. Time (s)	0.120	0.100	0.100	0.070	Not	
Avg. Time (s)	0.126	0.108	0.118	0.080	Schedulable	

CASE 2: 7 plants: (CC, MS, LC, F1, SC, RC, TTC) H **= 60 ,** J_H**=21 ,** $\hat{H}$ **= 1800 ,** $J_{\hat{H}}$ **= 630**						
Util. range	0.7 - 0.8		0.8 - 0.9		0.9 - 1.0	
Actual Util.	0.75	0.78	0.82	0.87	0.93	1.0
Min. Time (s)	0.481	0.421	0.343	0.450	2.094	4.097
Avg. Time (s)	0.55	0.448	0.552	0.494	2.582	6.058

CASE 3: 9 plants: (RC, SC, LC, F1, CC, MS1, TTC, VDC, MS2) H **= 60 ,** J_H**=23 ,** $\hat{H}$ **= 37,800 ,** $J_{\hat{H}}$ **= 14,490**						
Util. range	0.7 - 0.8		0.8 - 0.9		0.9 - 1.0	
Actual Util.	0.70	0.75	0.80	0.88	0.93	0.97
Min. Time (s)	0.030	0.334	0.578	1.438	1.969	3.201
Avg. Time (s)	0.070	0.350	0.724	1.604	2.1615	3.556

CASE 4: 11 plants: (DCS, RC, LC, SC, F1, CC, LK, MS1, TTC, VDC, MS2) H **= 60 ,** J_H**=29 ,** $\hat{H}$ **= 37,800 ,** $J_{\hat{H}}$ **= 18,270**						
Util. range	0.7 - 0.8		0.8 - 0.9		0.9 - 1.0	
Actual Util.	0.73	0.78	0.82	0.87	0.90	0.98
Min. Time (s)	1.515	3.578	8.312	12.031	91.594	161.031
Avg. Time (s)	1.547	3.620	8.328	12.219	92.724	161.836

CASE 5: 13 plants: (DCS, RC, LC, SC, F1, CC, LK1, MS1, TTC, VDC, MS2, ACC, LK2) H **= 60 ,** J_H**=33 ,** $\hat{H}$ **= 37,800 ,** $J_{\hat{H}}$ **= 20,790**						
Util. range	0.7 - 0.8		0.8 - 0.9		0.9 - 1.0	
Actual Util.	0.72	0.77	0.82	0.87	0.92	0.97
Min. Time (s)	15.781	29.907	30.234	63.422	102.172	399.421
Avg. Time (s)	16.167	30.537	32.867	63.953	102.610	400.053

CASE 6: 15 plants: (DCS, RC, LC, SC, F1-1, CC, LK, MS1, TTC, VDC, MS2, ACC, LK2, SC2, F1-2) H **= 60 ,** J_H**=37 ,** $\hat{H}$ **= 75,600 ,** $J_{\hat{H}}$ **= 46,620**						
Util. range	0.7 - 0.8		0.8 - 0.9		0.9 - 1.0	
Actual Util.	0.72	0.77	0.83	0.87	0.92	0.96
Min. Time (s)	478.125	661.719	2075.531	4235.438	7568.125	> 3 hrs
Avg. Time (s)	485.305	669.047	2230.242	4261.399	8517.031	Timed out

Table 2. Total runtime for various configurations of plants and corresponding task utilization

6.2 Scalability Analysis of the Proposed Method

In order to demonstrate the scalability and the real-world applicability of the proposed method, we carried out the experiments as mentioned above by increasing the number of plants and the corresponding control tasks. Some other benchmarks considered here are, a fifth-order *vision-based lateral control* (LC) system [9], a second-order *vehicle dynamic controller* (VDC), a second-order *trajectory tracking controller* (TTC) [1], a fifth-order *lane keeping* (LK) system, a third-order *adaptive cruise control* (ACC) and a second-order *DC-servo* (DCS) control system (supplement of [7]).

Objective: Here, we examine how the *total runtime* of the proposed method varies for different values of the *total processor utilization*, for various configurations of plant-control systems.

Design of the Experiment: We set up six cases by considering various numbers of plant-control systems, and for each such case, we devise several sub-cases by varying the total utilization (obtained via altering the input parameter WCET of the tasks). Table 2 reports the entire data set for the above cases. To get various plant options, in some instances the same system is considered with different sampling periods, e.g., *MS-1* and *MS-2*. The number of jobs scheduled till horizons $\boldsymbol{H}$ and $\hat{\boldsymbol{H}}$ are denoted as $\boldsymbol{J_H}$ and $\boldsymbol{J_{\hat{H}}}$ respectively. We report both the minimum (Min. time) and average (Avg. Time) time (for multiple runs) to synthesize the safe and stable schedule, for different total utilization values (Actual Util. = $\sum_{1 \leq i \leq n} \frac{m'_i \times c_i}{k_i \times h_i}$) in the range of **[0.6-1.0]** (Util. Range). Being an offline scheduling mechanism, for the proposed method, we fix a time-out value of **3** hours after which we declare the set of tasks to be *non-schedulable* in the processor.

Observations: We conclude the following points from the above experiment.

i) The number of jobs in $\hat{\boldsymbol{H}}$, i.e., $\boldsymbol{J_{\hat{H}}}$, is remarkably large for a higher number of plants, especially for Cases **3**-**6**. Moreover, the WCRT minimization facilitates obtaining a schedule, regardless of very high utilization values (e.g., **0.93**-**1**), in almost all the cases.

ii) It is also worth noting that the time taken to generate the schedule is fairly less for all utilization values in Cases **1**-**3**, and for Actual Util. $\boldsymbol{<}$ **0.9** in Case **4**. Almost for all task and utilization configurations, except Case 6, the average time is much less than 1 h; the highest is around 7 min for the utilization of **0.97** in Case 5. With 15 plants, in Case 6, the schedules are synthesized within a reasonable time (in the range of 7 min-38 min) up to the Actual Util. $\boldsymbol{<}$ **0.85**. For Actual Util. $\boldsymbol{>}$ **0.85**, we too can obtain the schedule by fixing the time-out a little higher.

This proves that the proposed method, despite of considering a higher number of control tasks (corresponding to the plant-control systems) and job instances, can still report a feasible schedule in a fairly reasonable time while utilizing the processor bandwidth as much as possible.

Improved Schedulability: We know that for a feasible schedule, the lateness should be either negative or zero, and for an infeasible schedule, the lateness value becomes a positive quantity. A higher negative value of the lateness signifies that the jobs are scheduled as soon as they arrive if there are empty slots to do so. This indicates a lesser WCRT, which is the objective function of our underlying optimization problem. Specifically, this improves the scope of schedulability in our method and thus it can schedule a large number of job instances at higher processor utilization values.

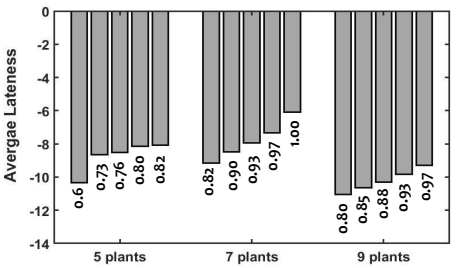


Fig. 7. Average lateness vs utilization

With an increase in the total utilization, the number of such empty slots decreases and the lateness increases, becoming less negative. The average lateness for various sub-cases of Cases **1**-**3** in Table 2, is shown in Figure 7. We achieve feasible schedules with a minimized WCRT, which is distinctly

justified in the figure in terms of negative average lateness. The respective utilization values are mentioned on top of all the bar graphs.

6.3 Comparison with the Existing Methods

Since in this work, we leverage *formal methods* for synthesizing the *safe and stable* schedule, we refer to our proposed method as *FMSS*, in the rest of the paper. As our focus, in this work, is on the triplet, (S_1*: stability,* S_2*: safety,* S_3*: schedulability*), we conduct the following set of rigorous experimental comparisons with the state-of-the-art methods exploring all these three aspects, specifically emphasizing the schedulability (S_3) aspect. As mentioned earlier, the literature lacks any existing method that focuses on S_1, S_2, and S_3 simultaneously, unlike our proposed one, therefore, we can mostly compare existing methods considering one (two) aspect(s) at a time.

(1) Comparison focusing on S_2, S_3: The closest works that also consider stability and schedulability together in the same context is [10]. Like [10], we consider the exponential stability criterion, but we take into account control safety to construct a safe and stable schedule, minimizing WCRT. On the other hand, [10] solves an ILP to generate a *Pattern Guided Stable* schedule, using EDF as the underlying scheduling algorithm, but has not addressed safety analysis. We name their method as *PGS* and compare with it, to prove that PGS not only lacks control safety but also falls short of our standards, in regards to other metrics too.

A. Comparing FMSS with PGS w.r.t. Schedulability (S_3):

Objective: To highlight the efficiency of FMSS, we compare it with PGS in terms of schedulability and the total runtime to generate the schedule.

Design of the Experiment: This comparison is carried out by providing two different types of inputs to PGS: a) $\boldsymbol{n}$ stable $\binom{M}{K}$ constraints, and b) $\boldsymbol{n}$ SASO constraints for $\boldsymbol{n}$ plant-control systems, as exhibited in Columns **4** and **5** in Table 3 respectively. In [10], PGS considers $\boldsymbol{n}$ stable $\binom{M}{K}$ constraints as inputs to generate a feasible schedule. Note that the setup with SASO constraints is not a part of PGS, rather we have designed the SASO in FMSS to generate a feasible schedule faster, along with enhancing the control performance. Since FMSS considers SASSO constraints (derived from SASO constraints) as inputs, hence, we also examine the schedulability of PGS with SASO constraints as inputs (SASSO cannot be an input to PGS, since it is deduced based on the safety criterion and safety analysis is not addressed by PGS in [10]). The ✓/✗ marks in Columns **3** and **5** denote the cases where a schedule could/couldn't be synthesized by FMSS and PGS, respectively.

n	Util.	Time (s) taken by FMSS	Time (s) taken by PGS with stable constraints	Time (s) taken by PGS with SASO constraints
5	0.76	0.100	80.625	0.456
5	0.82	0.070	207.547	1.344
7	0.78	0.421	10.234	0.100
7	**0.82 - 1.0**	✓	> 1 h (Timed Out)	✗
9	0.70	0.030	26.219	0.077
9	**0.72 - 0.97**	✓	> 1 h (Timed Out)	✗
11	**0.72 - 0.98**	✓	> 1 h (Timed Out)	✗
13	**0.72 - 0.92**	✓	> 1 h (Timed Out)	✗
15	**0.72 - 0.85**	✓	> 1 h (Timed Out)	✗

Table 3. Comparing FMSS with the existing method PGS

Observations: FMSS considerably outperforms PGS both w.r.t. the runtime and schedulability.

i) We observe that with stable $\binom{M}{K}$ constraints as inputs, PGS fails to report a schedule within an hour (time-out taken as **1** h), even for utilization values close to **0.72**, for **9** plants or higher (Column **4**). In contrast, FMSS reports a feasible schedule within 0.03 s-0.5 s in such cases (Column **3**). This mainly motivates us to construct another case by selecting the SASO constraints as inputs, which we follow in our proposed method to work with a smaller subset.

ii) The consideration of SASO constraints results in a smaller input space for the ILP in PGS. If $\binom{18}{30}$ is a stable constraint, then for any CES following the SASO $\binom{3}{5}$, if repeated to a length of **30**, forms a subset of the collection of ${}^{30}C_{18}$ different CESs, obtained from $\binom{18}{30}$. With this setup, the runtime of PGS improves considerably; for all the cases it takes nearly equal or a little higher runtime than our FMSS method, as shown in Column **5**. Yet, in the majority of the cases,

for increasing plant numbers and the corresponding task utilization values around the range $(0.72 - 1)$, PGS fails to report a feasible schedule, declaring the tasks to be non-schedulable by EDF. In contrast, FMSS guarantees schedulability with utilization up to **95%**, on average, for all these cases (the runtime detail for each individual case is reported in Table 2). Therefore, the construction of the SASO and minimization of the WCRT in FMSS is substantial, and clearly [10] fails to provide better schedulability with a smaller runtime.

B. Comparing FMSS with PGS w.r.t. Safety (S_2):

Objective: Since control safety is not addressed by PGS in [10], we compare w.r.t. the metric, safety, by exhibiting the violation of the safety bound, d^{safe}, in PGS in certain cases, as opposed to ours.

Observations: A schedule generated for a set of five tasks turns out to be *unsafe* in the case of PGS, as for one of the systems, F1-tenth model car, the state trajectory obtained by following the underlying CES **1101011010** generated by the ILP solver, deviates from the nominal trajectory (when execution skip is not allowed), by a value (**0.714**) more than d^{safe} (**0.56**). On the other hand, FMSS generates the safe CES as **10011**, which when repeated to a length of **10**, forms a stable and safe CES **1001110011**. With this CES the respective deviation from the nominal trajectory is a maximum of **0.462**. Figure 8 explains this pictorially, where the deviation at each sampling instant (multiples of 20 ms) is shown for both the methods.

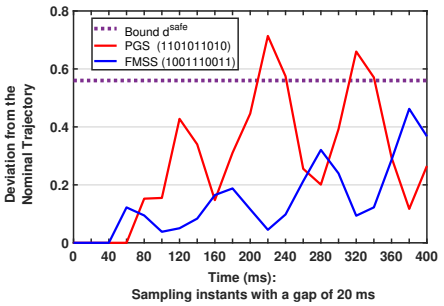


Fig. 8. Comparing control safety with PGS

This clarifies that the proposed method FMSS, surpasses PGS in terms of the metrics: schedulability, total runtime and control safety.

(2) Comparison focusing on S_1, S_3: The closest existing methods that consider control safety and schedulability aspects together for a weakly hard setting are [23] and [25]. Though these methods consider the same notion of control safety as ours, they do not account for stability in their works. Therefore, we compare FMSS and [23, 25] w.r.t. control stability (S_1) and schedulability (S_3) (specifically, in terms of the runtime to generate a safe schedule). The work in [23] develops the concept of *Safe Constraint Synthesis* to generate a safe schedule. The method described in [25] accounts for *Deterministic* verification of a schedule that is constructed with safe constraints, obtained through a probabilistic method named *Statistical Hypothesis Testing (SHT)* [8]. Based on their underlying methodical structures, we refer to these methods as *SCS* and *DSHT*, respectively. Both these methods have the limitation of ensuring control safety over an infinite time horizon and additionally consider an automata-based construction of schedulers that also rely on equal sampling periods for all the control tasks, making the real-world application space restricted. FMSS overcomes both these limitations while guaranteeing the desired stability of the systems.

A. Comparing FMSS with DSHT and SCS w.r.t. Stability (S_1):

Objective: We compare w.r.t. stability to highlight the fact that, even though the safe constraint synthesis ensures a minimum deviation from the ideal behavior (i.e., the nominal state trajectory), but the underlying systems can still remain unstable.

Design of the Experiment: We consider cruise control (CC) and suspension control (SC) systems for this comparative experiment. The controllers for CC and SC are designed using the LQR technique with a sampling period of 20 ms, as chosen in SCS and DSHT. For plant CC, DSHT reports $\binom{1}{6}$ as safe and both SCS and DSHT declare $\binom{1}{5}$ to be safe. However, FMSS states $\binom{5}{6}$ and $\binom{3}{5}$ as the SASSOs and **111011** and **11010** as the safe CESs, for lengths **6** and **5** respectively. Similarly,

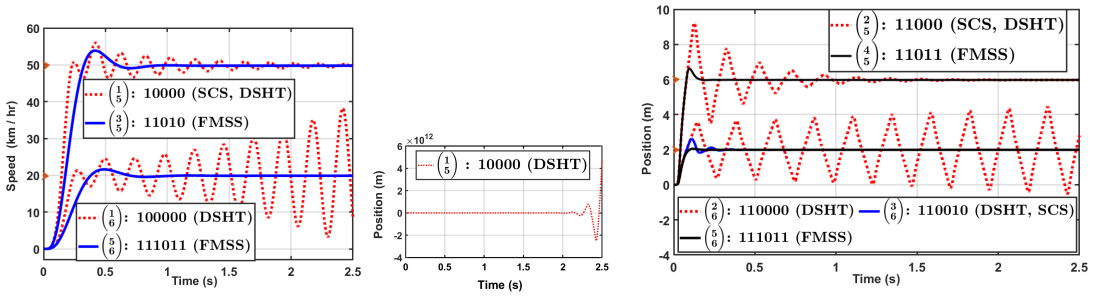
for plant SC, according to DSHT, the constraints $\binom{1}{5}$, $\binom{2}{6}$ are safe, and according to both DSHT and SCS, $\binom{2}{5}$, $\binom{3}{6}$ are safe (as reported in [23, 25]). With FMSS, we get $\binom{5}{6}$ and $\binom{4}{5}$ as SASSOs and **111011** and **11011** as safe CESs. For this experiment, we choose the CES for an $\binom{m}{k}$ provided by DSHT and SCS, which is the one amongst all possible combinations that gives the best output response.

Observations: On experimenting over the CC and SC systems, we observe noticeable differences in their output responses while selecting a safe constraint for state evolution, in contrast to the selection of a safe CES corresponding to our SASSO constraint derived from a stable constraint. Even with the best CES, the safe constraints of DSHT and SCS fail to ensure stability, in most cases.

i) For plant CC, Figure 9a portrays this observation for both the cases of length **5** (plots in the top with a reference of 50 km h^{-1}) and length **6** (plots in the bottom with a reference of 20 km h^{-1}). It is worth mentioning that the settling time achieved for the constraint $\binom{1}{5}$ returned by both SCS and DSHT (top, red) is 4 s, whereas for $\binom{3}{5}$ as returned by FMSS (top, blue), the settling time becomes 0.8 s only. In this case, the reference is considered as 50 km h^{-1} and marked in the plot in Figure 9a. On the other side, for $\binom{1}{6}$ as returned by DSHT, the system doesn't settle (bottom, red) at all, while in contrast, for $\binom{5}{6}$ in FMSS the system settles within 1 s (bottom, blue). In this case, the reference is 20 km h^{-1}. Hence, FMSS offers **80%** and **100%** improvements in the settling time respectively.

ii) For plant SC, considering constraint $\binom{1}{5}$ as returned by DSHT, the system shows extremely inconsistent and unstable behavior (ref. Figure 9b), with an overshoot in the range $\mathbf{10^{12}}$m, when its reference is 2 m. Also, with $\binom{2}{6}$ (reported by DSHT) the system becomes highly unstable (ref. Figure 9c, bottom, red). Although the system settles for constraints $\binom{2}{5}$ (top, red), $\binom{3}{6}$ (bottom, blue), as returned by DSHT and SCS, the settling times are 2.5 s and 0.5 s respectively. In contrast, for $\binom{4}{5}$ (top, black), $\binom{5}{6}$ (bottom, black), as reported by FMSS, the system settles at 0.2 s and 0.1 s respectively. With this, there is an improvement of **92%** and **80%** respectively, in FMSS, for plant SC.

Stability plays a pivotal role in designing efficient control systems and FMSS excels considerably in regard to this metric.



(a) Plant CC (b) Plant SC (c) Plant SC

Fig. 9. Comparing output responses with unstable systems of SCS and DSHT

B. Comparing FMSS with DSHT and SCS w.r.t. Schedulability (S_3):

Objective: Here, we compare with the DSHT and SCS methods w.r.t. metric, total runtime required to generate a safe schedule, to point out the efficiency of FMSS.

Design of the Experiment: For this comparison, we take the same set-up of **5** plants (ref. Table 1) as considered in [23, 25]. The only change we add is that we allow the sampling periods to be different since we do not have any restriction to consider them equal, as opposed to [23, 25].

Observations: As per the data reported in [25], the total time required to obtain a safe schedule by DSHT and SCS is 13.5 s and 748.37 s, respectively. In contrast, despite considering the same set-up for **5** plants (ref. Table 1), FMSS takes only 0.08 s to report the schedule, offering an improvement of nearly **99.41%** and **100%** in runtime respectively. Moreover, unlike FMSS, for these methods, dealing with a higher number of plants would be a practical challenge because of their heavy-duty computations; specifically, the reachability analysis and the iterative execution of the SHT framework (counter-example guided refinement on not obtaining a safe schedule). Also, the iterative process of synthesizing safe $\binom{m}{k}$s by calculating the upper bound on the deviation with the *bounded-tree algorithm* [15], makes the complexity even worse in SCS. These justifications point to an exponential time rise in the runtime for these methods, with an increasing number of plants and the corresponding tasks. Hence, FMSS is undoubtedly superior and efficient in comparison with [23, 25], also, it is quite comprehensible from the scalability analysis study reported in Table 2 of Section 6.2 (this type of scalability analysis with a higher number of plants is not reported in [23, 25]).

(3) Comparative Comments on Other Works: The only recent work that considers the SMT-based approach for generating safe schedules for weakly hard control systems is [26]. Here, the authors use refinements to reiterate the process of constraint solving, on obtaining a spurious trace, until a safe schedule is observed. Instead, FMSS aims to generate a safe schedule in exactly one iteration if it exists, which makes FMSS more effective. For a set-up of **4** control tasks, [26] reports the runtime and memory consumption as 40 s and 48 MB for the *hold-and-kill* policy (that we too consider, ref. Section 3.3) using Z3 solver; while with FMSS, Z3 generates the schedule in 0.04 s having a 20 MB memory requirement, for a 4-task setup. Since the safety notion considered in [26] is completely different from ours, there is no scope for fair comparison w.r.t. this aspect. Moreover, as reported in [26], their work is limited to medium-sized systems; we anticipate that [26] may suffer from higher time-complexity and memory consumption (due to the iterative refinement process) issues while dealing with a higher number of systems simultaneously, as opposed to ours.

7 CONCLUSION AND FUTURE DIRECTIONS

This work proposes a novel approach for generating a safe and stable schedule for a set of weakly hard control tasks preserving the desired control stability and safety guarantee. To the best of our knowledge, this is the first work which addresses the triplet, (stability, safety, schedulability), in the real-time control context. The state-of-the-art methods have studied either stability or safety but not both the aspects with a harmony of real-time scheduling of control tasks. To this end, we develop a scheduler, which even though permits tasks to miss some of their deadlines intermittently, we still do not allow the underlying stability and safety to be compromised. Moreover, we establish the control safety over an infinite time horizon, unlike some previous methods that consider a bounded time horizon. To synthesize the schedule, we develop an SMT-based scheduling approach which minimizes the worst-case response time. In contrast to existing methods, the SMT-based approach proves itself to be time-efficient in our work. This is because we reduce the search space of the SMT solver by selecting exactly one deadline hit-miss pattern for a task, which conforms to a weakly hard constraint obeying both the stability and safety criteria. Additionally, with some rigorous case studies, we demonstrate the scalability of the proposed method and its efficiency in comparison to some existing methods, which have addressed either stability or safety. Dealing with both stability and safety for scheduling in non-linear control systems could be an interesting next step of this work. Moreover, establishing efficient techniques to generate an SMT-based schedule in a reasonable time frame for a significantly large number of control tasks, is also an essential aspect to explore in the near future. Another riveting future direction could be considering dependencies in the task model while designing a safe and stable schedule.

ACKNOWLEDGMENTS

The authors sincerely thank the anonymous reviewers for their constructive feedback.

REFERENCES

[1] Sunandan Adhikary et al. 2020. Skip to secure: Securing cyber-physical control loops with intentionally skipped executions. In *Proc. on CPS&IoT Security and Privacy (CPSIoTSec)*. 81–86.
[2] Sanjoy Baruah, Pontus Ekberg, and Abhishek Singh. 2022. Fixed-parameter analysis of preemptive uniprocessor scheduling problems. In *Proc. Real-Time Systems Symposium (RTSS)*. 185–196.
[3] G. Bernat et al. 2001. Weakly hard real-time systems. *IEEE Transactions on Computers (TC)* 50, 4 (2001), 308–321.
[4] Rainer Blind and Frank Allgöwer. 2015. Towards networked control systems with guaranteed stability: Using weakly hard real-time constraints to model the loss process. In *Proc. Conference on Decision and Control (CDC)*. IEEE, 7510–7515.
[5] Zhuo Cheng, Haitao Zhang, Yasuo Tan, and Yuto Lim. 2017. SMT-based scheduling for overloaded real-time systems. *IEICE Transactions on Information and Systems* 100, 5 (2017), 1055–1066.
[6] Hyunjong Choi, Hyoseung Kim, and Qi Zhu. 2021. Toward practical weakly hard real-time systems: A job-class-level scheduling approach. *IEEE Internet of Things Journal (IoT-J)* 8, 8 (2021), 6692–6708.
[7] Hoon Sung Chwa et al. 2018. Closing the gap between stability and schedulability: A new task model for cyber-physical systems. In *Proc. Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 327–337.
[8] Bineet Ghosh et al. 2022. Statistical hypothesis testing of controller implementations under timing uncertainties. In *Proc. Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 11–20.
[9] Sumana Ghosh. 2023. Delay-Aware Control for Autonomous Systems. In *Proc. VLSI Design and Embedded Systems (VLSID)*. 1–6.
[10] Sumana Ghosh et al. 2017. A Structured Methodology for Pattern Based Adaptive Scheduling in Embedded Control. *ACM Transactions on Embedded Computing Systems (TECS)* 16, 5s (Sept. 2017), 189:1–189:22.
[11] Sumana Ghosh, Soumyajit Dey, and Pallab Dasgupta. 2019. Performance and Energy Aware Robust Specification of Control Execution Patterns under Dropped Samples. *IET Computers & Digital Techniques (IET-CDT)* 13 (2019), 493–504.
[12] Sumana Ghosh, Soumyajit Dey, and Pallab Dasgupta. 2020. Pattern guided integrated scheduling and routing in multi-hop control networks. *ACM Transactions on Embedded Computing Systems (TECS)* 19, 2 (2020), 1–28.
[13] Robert Gifford et al. 2024. Decntr: Optimizing safety and schedulability with multi-mode control and resource allocation co-design. In *Proc. Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 306–319.
[14] Zain AH Hammadeh, Sophie Quinton, and Rolf Ernst. 2019. Weakly-hard real-time guarantees for earliest deadline first scheduling of independent tasks. *ACM Transactions on Embedded Computing Systems (TECS)* 18, 6 (2019), 1–25.
[15] Clara Hobbs et al. 2022. Safety analysis of embedded controllers under implementation platform timing uncertainties. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 41, 11 (2022), 4016–4027.
[16] Ling Lyu et al. 2017. Co-design of stabilisation and transmission scheduling for wireless control systems. *IET Control Theory & Applications* 11, 11 (2017), 1767–1778.
[17] Martina Maggio et al. 2020. Control-system stability under consecutive deadline misses constraints. In *Proc. Euromicro Conference on Real-Time Systems (ECRTS)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.
[18] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *Proc. Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 337–340.
[19] Youcheng Sun and Marco Di Natale. 2017. Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks. *ACM Transactions on Embedded Computing Systems (TECS)* 16, 5s (2017), 1–19.
[20] Nils Vreman, Paolo Pazzaglia, Victor Magron, Jie Wang, and Martina Maggio. 2022. Stability of linear systems under extended weakly-hard constraints. *IEEE Control Systems Letters (L-CSS)* 6 (2022), 2900–2905.
[21] Shimin Wang et al. 2020. An improved smt-based scheduling for overloaded real-time systems. *Engineering Letters* 28, 1 (2020), 112–122.
[22] Guoqi Xie, Wei Wu, and Renfa Li. 2021. Carry-out interference optimization in WCRT analysis for global fixed-priority multiprocessor scheduling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 41, 3 (2021), 478–491.
[23] Shengjie Xu et al. 2023. Safety-aware flexible schedule synthesis for cyber-physical systems using weakly-hard constraints. In *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*. 46–51.
[24] Shengjie Xu et al. 2023. Safety-Aware Implementation of Control Tasks via Scheduling with Period Boosting and Compressing. In *Proc. Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 196–205.
[25] Shengjie Xu et al. 2023. Statistical approach to efficient and deterministic schedule synthesis for cyber-physical systems. In *Proc. International Symposium on Automated Technology for Verification and Analysis (ATVA)*. Springer, 312–333.
[26] Anand Yeolekar, Ravindra Metta, and Samarjit Chakraborty. 2024. SMT-based Control Safety Property Checking in Cyber-Physical Systems under Timing Uncertainties. In *Proc. on VLSI Design and Embedded Systems (VLSID)*. 276–280.