Integrated Real-Time Control and Scheduling for Safety Critical Cyber-Physical Systems

Marion Sudvarg[1], Andrew Clark[2], and Chris Gill[1]

[1]*Department of Computer Science and Engineering* [2]*Department of Electrical and Systems Engineering*
James McKelvey School of Engineering, Washington University in St. Louis
St. Louis, Missouri, United States
{*msudvarg, andrewclark, cdgill*}*@wustl.edu*

***Abstract*—Cyber-physical systems (CPS) must interact with varying environments at fine-grained time-scales, assuring control safety and stability while optimizing application-specific performance objectives. To address those requirements, co-design of real-time control and scheduling has received considerable attention over multiple decades, to allow rigorous assurance of system properties while enabling diverse forms of adaptation to changing operating conditions.**

In this paper, we present a new formalization of the periodicity requirements for control inputs to (1) guarantee reachability of safe (and avoidance of unsafe) portions of the system state space, (2) adaptively manage *dynamic* periodicity constraints that may change as the state space is traversed, and (3) express minimum periods to enable safe hand-offs between high-performance controllers and more conservative backup controllers. Our evaluations of this approach confirm that it is able to maintain system safety and stability while optimizing system performance.

***Index Terms*—Cyber-physical systems, real-time scheduling, control barrier functions, control and scheduling co-design**

I. Introduction

An increasingly connected world has driven rising demand for applications that respond to events in *real-time*. In cyber-physical systems (CPS), where software must sense and control physical interactions with the outside world, timely execution must be guaranteed: *temporal* correctness is as essential as *functional* correctness so that scheduling computations to meet deadlines is a first-class concern. Schedulability analysis has been a core focus of the real-time systems community, which has modeled the computational aspects of CPS using nuanced expansions of Liu and Layland's recurrent task model [1]: tasks are characterized by static properties like worst-case execution times, periods, and deadlines; a set of tasks is then deemed SCHEDULABLE or UNSCHEDULABLE according to these parameters for the chosen scheduling algorithm.

However, CPS increasingly execute in varying environments: microrobots [2]–[4] and other autonomous systems [5]–[7] must remain adaptable in diverse contexts. In safety-critical control systems, it is crucial that a system remain both ***schedulable*** and ***safe***, even as modes change in response to exogenous conditions or proximity to unstable or unsafe regions of the state space as the system navigates its environment. Adaptive scheduling and control are thus fundamentally coupled, and both should be configured to optimize performance within application-specific dynamic timing and safety constraints.

A key performance measure is *control cost*. In digital control systems, which discretize continuous control, control cost monotonically decreases with increasing invocation frequency. Seto et al. [8] considered scheduling digital controllers as a problem of selecting controller frequencies to minimize total control cost while *(i)* imposing a lower bound on each frequency to "guarantee the system behaves properly under digital control" and *(ii)* constraining the total utilization to remain within the bounds given by rate-monotonic (RM) or earliest deadline first (EDF) scheduling on a multiprocessor. However, many problems remain to be solved when implementing this as an adaptive framework for dynamic environments.

Safety: Selecting and enforcing constraints on digital controllers' minimum frequencies is vital to safe control system execution, and remains an open research question. Approaches include assigning multiples of a system's characteristic frequencies [8], with control barrier function constraints for robustness to sampling-related errors [9]–[12]. In this paper, we present a new formalization of the periodicity requirements for control inputs based on delay bounds on their outputs that guarantee reachability of safe (and avoidance of unsafe) portions of the system state space by ensuring positive invariance.

Moreover, in safety-critical systems, controllers that tend to provide better control performance may be coupled with more reliable high-assurance *backup controllers* that give stronger safety/stability guarantees [13], [14]. We extend our formalization to coupled controllers, presenting *(i)* an expression for the minimum safe period of each controller, and *(ii)* a check condition, based on a general expression of the safety regions of the state space, on which the transition from a performance to backup controller must occur. This also gives rise to an expression of the minimum frequency as a *dynamic* constraint that may change as the state space is traversed.

Adaptive Schedulability: The requirements to maintain safety in dynamic environments give rise to new scheduling problems. We formalize these as *constrained optimizations* over controller frequencies, with the objective to *minimize control cost* within formal *safety* constraints while guaranteeing *schedulability*. For systems we consider, offline calculation for online use may be infeasible: e.g., in a system of n controllers, there are 2^n possible combinations of performance/backup

activation; for large values of n, storing frequencies for all configurations quickly becomes prohibitive. Furthermore, frequency constraints may change as the system traverses different subregions of the state space. Moreover, execution mode changes may require the admission of additional tasks, affecting the schedulability constraints. Thus, the scheduling algorithm must remain *low-overhead*, and its execution time must be accounted for in schedulability analysis.

This paper proposes a framework for solving the optimization problem online, reassigning task periods to maintain safety, schedulability, and optimality while avoiding transient overload across controller transitions. It presents efficient algorithms to solve the optimization problem for a representative class of nonlinear cost functions via linearization with the Karush–Kuhn–Tucker (KKT) conditions, enabling $\mathcal{O}(n)$ period reassignment under utilization-based schedulability tests. We first consider RM and EDF scheduling on a uniprocessor and then extend the algorithms for fluid and partitioned EDF approaches to multiprocessor scheduling. We also demonstrate how this linearization gives rise to efficient numerical methods for optimization under more complex scheduling policies.

Organization: §II provides background on the system model. §III formalizes and solves the control problems. §IV formalizes and solves the scheduling problems, while considering how online adaptation can maintain safety and optimize performance across sub-regions of the state space and handoffs between controllers. §V evaluates the efficiency of our scheduling algorithms and performs a case study in aircraft control.[1] §VI positions this paper in the context of other related work. §VII presents conclusions and future work.

II. Preliminaries and System Model

We consider a cyber-physical system (CPS) running a set Γ of n real-time tasks. Each task $\tau_i \in \Gamma_{\text{CTRL}} \subseteq \Gamma$ controls a nonlinear system ς_i. We describe the task model and scheduling problems we address, then introduce the control model.

A. Scheduling

Each nonlinear system ς_i is controlled by an independent task $\tau_i \in \Gamma_{\text{CTRL}}$ running concurrently with non-control tasks: this gives rise to a scheduling problem for a task set Γ, with $\Gamma_{\text{CTRL}} \subseteq \Gamma$. We consider sporadic tasks $\tau_i = (C_i, T_i)$ according to Liu and Layland's model [1] where C_i is the task's worst-case execution time, and T_i is the minimum interarrival time between instances (or *jobs*) of the task. We consider only implicit deadlines $D_i = T_i$, but all schedulability analysis in this paper is *sustainable* [15], and so it is valid (though pessimistic) to treat a constrained deadline task τ_i with $D_i < T_i$ as if it has a period $T_i' = D_i$. Optimal analysis for constrained-deadline tasks is left for future work, though as we show in the following sections, our formalization of safe periodicity requirements maps naturally to an implicit-deadline task representation. Under this system model, a control task's period T_i is exactly the interval T_i between its discrete invocations as described in §II-B. Of primary concern to this work is the selection of frequencies $\omega_i = 1/T_i$ at which to run each controller. This gives rise to a problem of control and scheduling co-design, which we state as follows:

> Given a collection of tasks implementing computation of discrete digital controllers, assign a frequency to each task to *minimize control cost* within the constraints of *safety* and *schedulability*.

Each task has a minimum frequency $\omega_i^{\min}$ — corresponding to a maximum period $T_i^{\max}$ — below which system safety cannot be guaranteed. In §III, we present a novel derivation of this bound for a system ς_i under the control strategies described in §II-B. If the system is not *schedulable*, then it cannot guarantee this frequency is maintained, and it is therefore no longer safe.

Though a control task cannot safely be assigned a period above its maximum $T_i^{\max}$, it can be invoked at a *higher* frequency, thereby reducing the cost associated with controlling the corresponding system. In this work, we assume the cost $J_i(\omega_i)$ for controller ς_i is monotonically decreasing and convex, and that total cost is additive across controllers. This gives rise to the following constrained optimization problem:

$$\min_{\omega_i} \quad \sum_{\varsigma_i} J_i(\omega_i) \tag{1a}$$

$$\text{s.t.} \quad \forall i, \quad \omega_i \geq \omega_i^{\min} \tag{1b}$$

$$\text{and} \quad \text{The system is schedulable} \tag{1c}$$

In §IV, we derive a linear-time procedure to solve this efficiently for representative cost functions under uniprocessor, fluid, and partitioned EDF scheduling, thus ensuring safety and schedulability across online controller transitions.

B. Control

We consider a collection $\mathbf{S}$ of M nonlinear systems ς_i, each corresponding to a task $\tau_i \in \Gamma_{\text{CTRL}}$. The state of system ς_i at time t is described by $x_i(t) \in \mathbb{R}^{n_i}$ with control input $u_i(t) \in \mathbb{R}^{m_i}$. Dynamics are governed by

$$\dot{x}_i(t) = f_i\left(x_i(t)\right) + g_i\left(x_i(t)\right) u_i(t) \tag{2}$$

where $f_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ and $g_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i \times m_i}$ are polynomials [16]. From Eq. (2), the states of the systems are decoupled, but their corresponding *tasks* may be scheduled concurrently on shared resources, giving rise to the control/scheduling co-design problem described in §II-A.

Safety: The system governed by Eq. (2) operates under the safety constraints defined as follows. A subset $\mathcal{W}_i \subseteq \mathbb{R}^{n_i}$ is *positive invariant* under the dynamics in Eq. (2) for the given control input $u_i(t)$ if $x_i(t_0) \in \mathcal{W}_i$ implies that $x_i(t) \in \mathcal{W}_i$ for all $t \geq t_0$. In other words, for given dynamics, it describes a region of the state space for which, once the system enters that region, it will not leave that region in the future. This allows us to define formally a control system's *safety*.

Definition 1. *Let $\mathcal{W}_1, \ldots, \mathcal{W}_M$ be a given collection of sets representing safe regions of the state space with $\mathcal{W}_i \subseteq \mathbb{R}^{n_i}$. The nonlinear systems $x_1(t), \ldots, x_M(t)$ are* **safe** *if the sets $\mathcal{W}_1, \ldots, \mathcal{W}_M$ are positive invariant.*

[1] All simulation and evaluation code and data are available at https://github.com/McKelvey-Engineering-CSE/2025_rtas_control_codesign/.

In §III, we derive a temporal constraint for each controller that must be enforced by the chosen task scheduling policy to guarantee system safety. Control Barrier Functions (CBFs), defined as follows, give one approach for guaranteeing safety.

Definition 2. *A function $b_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is a* **Control Barrier Function** *(CBF) for nonlinear system ς_i if:*

(i) $\frac{\partial b_i}{\partial x_i} \neq 0$ *for all* x_i *with* $b_i(x_i) = 0$ *and*

(ii) there is a strictly increasing function $\alpha_i : \mathbb{R} \to \mathbb{R}$ *with* $\alpha_i(0) = 0$ *such that*

$$\inf \left\{ \frac{\partial b_i}{\partial x_i}(f_i(x_i) + g_i(x_i)u_i : u_i \in \mathbb{R}^{m_i} \right\} \geq -\alpha_i(b_i(x_i)).$$

A variety of techniques have been proposed for constructing CBFs [17]–[19]. The following result gives our main theoretical tool for proving safety of system ς_i.

Theorem 1 ([20], Theorem 2)**.** *Suppose that a region $\mathcal{W}_i$ is described by $\mathcal{W}_i = \{x_i : b_i(x_i) \geq 0\}$ for some CBF $b_i : \mathbb{R}^{n_i} \to \mathbb{R}$. Suppose that, at each time t, $u_i(t)$ satisfies*

$$\frac{\partial b_i}{\partial x_i}(f_i(x_i(t)) + g_i(x_i(t))u_i(t)) \geq -\alpha_i(b_i(x_i(t)) \quad (3)$$

where the partial derivative is evaluated at $x_i(t)$ and α_i satisfies the conditions of Definition 2. Then $\mathcal{W}_i$ is positive invariant.

Control Strategies: The safety of a system described by Eq. (2) thus depends on the choice of control signal $u_i(t)$. In this work, we consider two types of control strategies, each of which is based on a discrete-time implementation of a continuous feedback control law. Our goal is not to present new methods for constructing control barrier functions b_i or control input functions $u_i(t)$. Rather, it is to derive lower bounds on sampling frequency, and associated adaptive scheduling policies, to guarantee a system's safety under these strategies. The two strategies are described as follows.

1) Single Safe Control Law: In this control law, we are given a continuous function $\mu_i : \mathbb{R}^{n_i} \to \mathbb{R}^{m_i}$ that acts as a feedback control law. The function μ_i is assumed to satisfy Eq. (3) with strict inequality for some class-$\mathcal{K}$ function α_i. This property is satisfied, for example, by CBF-based controllers [20]. This control law is implemented in a sampled-data manner as a periodic task that releases jobs at a set of discrete times $r_{i,0} < r_{i,1} < \cdots$ with $r_{i,j+1} - r_{i,j} = T_i$ where T_i is the *period* of task τ_i. At a set of discrete times $t_{i,0} < t_{i,1} < \cdots$ with each $t_{i,j} \in [r_{i,j}, r_{i,j+1})$, the state $x_i(t_{i,j})$ is sampled and the control $\mu_i(x_i(t_{i,j}))$ is computed. The control signal is then given by $u_i(t) = \mu_i(x_i(t_{i,j}))$ over time interval $t \in [t_{i,j} + R_{i,j}, t_{i,j+1} + R_{i,j+1}]$, where $R_{i,j}$ is the response time required to compute $\mu_i(x_i(t_{i,j}))$ and send this command to the actuators.

2) Nominal and Safe Backup Control Laws: In this case, we assume that the controller has two modes, a *nominal* mode that prioritizes performance and a *safe backup* mode that prioritizes safety. The nominal and backup modes are characterized by continuous functions $\mu_i^N, \mu_i^B : \mathbb{R}^{n_i} \to \mathbb{R}^{m_i}$, respectively. Here, the backup mode is assumed to satisfy Eq. (3) with strict equality for some class-$\mathcal{K}$ function α_i. This class of control strategies has been described as having an *implicit or backup Control Barrier Function (CBF)* in the literature [21], [22], and assumes state-feedback controllers without internal states.

The control law is again implemented in a sampled-data manner as a task that releases jobs at a set of discrete times $r_{i,0} < r_{i,1} < \cdots$ with period $T_i = r_{i,j+1} - r_{i,j}$. At time $t_{i,j} \in [r_{i,j}, r_{i,j+1})$, the system samples state $x_i(t_{i,j})$, computes $b_i(x_i(t_{i,j}))$, and compares it with a threshold β_i representing a distance from the boundary of the safe region. If $b_i(x_i(t_{i,j})) \geq \beta_i$, the system stays in nominal mode, i.e., it computes $\mu_i^N(x_i(t_{i,j}))$ and the control signal is given by $u_i(t) = \mu_i^N(x_i(t_{i,j}))$ over the time interval $[t_{i,j} + R_{i,j}, t_{i,j+1} + R_{i,j+1}]$. However, if $b_i(x_i(t_{i,j})) < \beta_i$, then the control signal is given by $u_i(t) = \mu_i^B(x_i(t_{i,j}))$ over the interval $[t_{i,j} + R_{i,j}, t_{i,j+1} + R_{i,j+1}]$. Thus, any mode change is detected within a small fixed-length prefix (which is added to the task's worst-case execution time) of the controller's execution after sampling, and that mode is used for the rest of the job.

A key feature of this approach is that the period between job releases $T_i = r_{i,j+1} - r_{i,j}$ and the interval from sampling to applying the control command $R_{i,j}$ will depend on the mode of operation; we denote these T_i^N and $R_{i,j}^N$ when system ς_i operates in nominal mode and T_i^B and $R_{i,j}^B$ in backup mode.

Sum of Squares (SOS) Polynomials: Our procedure in §III for deriving constraints on controller frequencies requires checking for the existance of SOS polynomials.

Definition 3. *A polynomial $h(x)$ is a* ***sum-of-squares*** *(SOS) if it can be written in the form*

$$h(x) = \sum_{i=1}^{N} q_i(x)^2$$

for some polynomials $q_1, \ldots, q_N$.

The problem of selecting coefficients of a polynomial $h(x)$ to ensure that it is SOS can be represented as a semidefinite program, which is denoted as SOS optimization [23].

III. Control Problems

This section describes the scheduling constraints imposed by the safety requirements of the system. In order to ensure safety, a control task's period T_i must be sufficiently short. It also presents a procedure for upper-bounding the *delay interval* $\Delta_{i,j}$ to guarantee the safety of a control system ς_i.

Definition 4 (Delay Interval)**.** *Consider a nonlinear system ς_i governed by a discrete digital controller. Its state is sampled consecutively at times $t_{i,j}$, $t_{i,j+1}$ and a control signal is computed and applied at times $t'_{i,j}$, $t'_{i,j+1}$. The delay interval $\Delta_{i,j}$ for the sample taken at $t_{i,j}$ is the elapsed time $t'_{i,j+1} - t_{i,j}$.*

Intuitively, the delay interval is the elapsed time between the sampled state and the application of the *next* control signal, as shown in Fig. 1. Intuitively, a safe bound on this interval

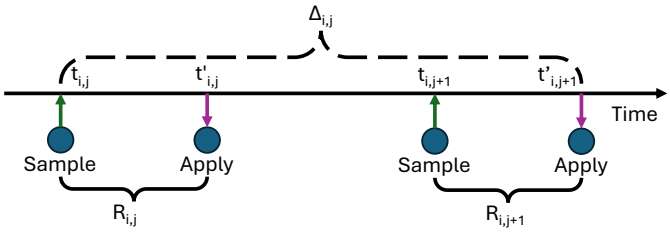


Fig. 1: The delay interval $\Delta_{i,j}$.

depends on how quickly the system may deviate from its current state under the chosen control law. Given this, keeping the state from reaching an unsafe region requires bounding the delay interval between when the current state is measured, and when the next measurement *and application of the resulting control signal* complete. We refer to the safe upper bound on $\Delta_{i,j}$ as the **delay bound** $\Delta_i^{\max}$ for the *system* ς_i. This section derives that bound; in the next section, we use this to derive a corresponding upper bound on the scheduling period for implicit-deadline tasks. We first consider a single safe controller, and then the case where there is a nominal controller and a safe backup controller, based on the two control strategies in §II-B.

A. Single Controller

From §II-B, safety of the system is guaranteed if Eq. (3) holds for all t. We observe that the control input at time t is given by $\mu_i(x_i(t_{i,j}))$. In other words, the control input is given by $\mu_i(x_i(t_{i,j-1}))$ while the next input $\mu_i(x_i(t_{i,j}))$ is being computed, and is equal to $\mu_i(x_i(t_{i,j}))$ thereafter. Our approach to ensuring Eq. (3) is to prove that

$$\frac{\partial b_i}{\partial x}\left(f_i(x_i)+g_i(x_i)\mu_i(x_i')\right) \geq -\alpha_i(b_i(x_i))$$

whenever $||x_i - x_i'||_2^2 \leq \rho_i^2$, or equivalently, whenever $x_i(t)$ is in the ball of radius ρ_i centered at x_i'. It then suffices to choose T_i so that $x_i(t)$ remains in the ball of radius ρ_i centered at $x_i(t_{i,j})$ for all $t \in [t_{i,j}, t_{i,j} + R_i + T_i]$.

Our approach to choosing T_i is based on lower-bounding the derivative $\frac{d}{dt}||x_i - x_i'||_2^2$ when x_i is in a ball of radius ρ_i centered at x_i'. To produce less conservative safety conditions, we compute a parameter ψ_i so that $\frac{d}{dt}||x_i(t) - x_i(t_{i,j})||_2^2 \leq \psi_i$ for $t \in [t_{i,j}, t_{i,j} + R_i]$ and a parameter θ_i so that $\frac{d}{dt}||x_i(t) - x_i(t_{i,j})||_2^2 \leq \theta_i$ for $t \in [t_{i,j}, t_{i,j} + R_i + T_i]$. The goal of our approach is to compute parameters ρ_i, θ_i, and ψ_i (Fig. 2), and then compute the maximum safe task period T_i as a function of the parameters.

Proposition 1. *Suppose that there exist nonnegative parameters ψ_i, θ_i, and ρ_i with $\theta_i \leq \psi_i$ such that the following conditions hold:*

1) For all x_i and x_i'' with $b_i(x_i'') \geq 0$ and $||x_i - x_i''||_2 \leq \rho_i$,

$$\frac{\partial b_i}{\partial x_i}\left(f_i(x_i)+g_i(x_i)\mu_i(x_i'')\right) \geq -\alpha_i(b_i(x_i)) \quad (4)$$

$$2(x_i - x_i'')^\top\left(f_i(x_i)+g_i(x_i)\mu_i(x_i'')\right) \leq \theta_i \quad (5)$$

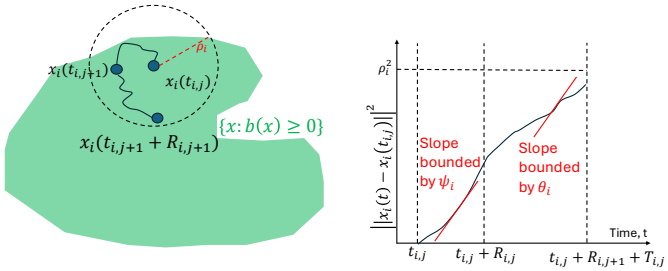


Fig. 2: Illustration of the parameters ψ_i, θ_i, and ρ_i.

2) For all x_i, x_i', x_i'' contained in the same ball of radius ρ_i with $b_i(x_i'') \geq 0$, we have

$$2(x_i - x_i')^\top\left(f_i(x_i)+g_i(x_i)\mu_i(x_i'')\right) \leq \psi_i \quad (6)$$

3) The parameters satisfy $\psi_i R_i + \theta_i(R_i + \Delta_i) < \rho_i^2$.

Then the sampled-data control policy with delay interval Δ_i ensures that $\mathcal{W}_i$ is positive invariant.

For ease of presentation, we prove this in two lemmas.

Lemma 1. *Under the assumptions of Proposition 1, if $b_i(x_i(0)) > 0$, then **(i)** $||x_i(t) - x_i(0)|| \leq \rho_i$ for all $t \in [0, \Delta_i + R_{i,1}]$ where $R_{i,1}$ is the response time to compute and apply the control input at time $t_{i,1}$, and **(ii)** $b_i(x_i(t)) > 0$ for all $t \in [t_0, t_{i,1}]$.*

Proof. We first show **(i)**. Suppose condition **(i)** is violated, and let $t' = \inf\{t : ||x_i(t) - x_i(0)|| > \rho_i\}$. We have that

$$\begin{aligned}
&||x_i(t') - x_i(0)||_2^2 \\
&= \int_0^{t'} 2(x_i(\tau) - x_i(0))^\top\left(f_i(x_i(\tau)) + g_i(x_i(\tau))\mu_i(x_i(0))\right)\, d\tau \\
&\leq t'\theta_i \leq (\Delta_i + R_i)\theta_i < \rho_i^2,
\end{aligned}$$

a contradiction, where the first inequality follows from Eq. (5). Now, since **(i)** holds, we have that

$$\begin{aligned}
\frac{d}{dt}b_i(x_i(t)) &= \frac{\partial b_i}{\partial x_i}(f_i(x_i(t)) + g_i(x_i(t))\mu_i(x_i(0))) \\
&\geq -\alpha_i(b_i(x_i(t))
\end{aligned}$$

for $t \in [0, t_{i,1}]$ by Eq. (4), implying **(ii)** that $b_i(x_i(t)) > 0$ by the comparison lemma [24]. □

Lemma 2. *Let j be a positive integer. Under the assumptions of Proposition 1, if $b_i(x_i(t_{i,j})) > 0$ and $||x_i(t) - x_i(t_{i,j-1})||_2 \leq \rho_i$ for all $t \in [t_{i,j}, t_{i,j} + R_i]$, then **(i)** $||x_i(t_{i,j}) - x_i(t)||_2 \leq \rho_i$ for all $t \in [t_{i,j}, t_{i,j+1} + R_{i,j+1}]$ and **(ii)** $b_i(x_i(t)) > 0$ for all $t \in [t_{i,j}, t_{i,j+1}]$.*

Proof. Suppose that condition **(i)** fails, and let $t' = \inf\{t : ||x_i(t) - x_i(t_{i,j})||_2 > \rho_i\}$. First, suppose that $t' \leq t_{ij} + R_{ij}$. We have

$$\begin{aligned}
&||x_i(t') - x_i(t_{ij})||_2^2 \\
&= \int_{t_{ij}}^{t'} [2(x_i(\tau) - x_i(t_{ij}))^\top(f_i(x_i(\tau)) \\
&\quad + g_i(x_i(\tau))\mu_i(x_i(t_{i,j-1})))]\, d\tau \\
&\leq (t' - t_{ij})\psi_i \leq R_i\psi_i \leq \rho_i^2,
\end{aligned}$$

a contradiction. Now, suppose that $t' \in [t_{ij} + R_{ij}, t_{i,j+1} + R_{i,j+1}]$. We have

$$\begin{aligned}||x_i(t') - x_i(t_{ij})||_2^2 \leq \int_{t_{ij}}^{t_{ij}+R_{ij}} &[2(x_i(\tau) - x_i(t_{ij}))^\top \\ &\cdot(f_i(x_i(\tau)) + g_i(x_i(\tau))\mu_i(x_i(t_{i,j-1})))] \, d\tau \\ + \int_{t_{ij}+R_{ij}}^{t'} &[2(x_i(\tau) - x_i(t_{ij}))^\top \\ &\cdot(f_i(x_i(\tau)) + g_i(x_i(\tau))\mu_i(x_i(t_{i,j}))) \, d\tau \\ \leq R_i\psi_i + t'(\theta_i) \leq R_i\psi_i + (\Delta_i + R_i)\theta_i &< \rho_i^2\end{aligned}$$

a contradiction. Hence **(i)** holds. Condition **(ii)** then follows from Eq. (4) and the comparison lemma. □

Proof of Proposition 1. The proof is by induction on j. When $j = 0$, the result holds by Lemma 1. For positive j, the result holds by Lemma 2. □

The next step is a procedure for computing the parameters along with the maximum delay interval. Our procedure consists of the following steps:

A1) Using binary search, find the maximum value of ρ_i such that there exist SOS polynomials $\lambda_{i,j}(x_i, x_i')$, $j = 0, 1, 2$ where the following is SOS:

$$\begin{aligned}\lambda_{i,0}(\frac{\partial b_i}{\partial x_i}(f_i(x_i) + g_i(x_i)\mu_i(x_i')) + \alpha_i(b_i(x_i))) \\ - \lambda_{i,1}b_i(x_i') - \lambda_{i,2}(\rho_i^2 - ||x_i - x_i'||_2^2)\end{aligned} \quad (7)$$

A2) Using binary search, find the minimum value of θ_i such that there exist SOS polynomials $\xi_{i,j}(x_i, x_i')$ for $j = 0, 1, 2$ such that the following is SOS:

$$\begin{aligned}\xi_{i,0}(\theta_i - 2(x_i - x_i')^\top(f_i(x_i) + g_i(x_i)\mu_i(x_i'))) \\ - \xi_{i,1}b_i(x_i') - \xi_{i,2}(\rho_i^2 - ||x_i - x_i'||_2^2)\end{aligned} \quad (8)$$

A3) Using binary search, find the minimum value of ψ_i such that there exist SOS polynomials $\phi_{i,j}(x_i, x_i', x_i'')$ for $j = 0, 1, 2, 3, 4$ where the following is SOS:

$$\begin{aligned}&\phi_{i,0}(\psi_i - 2(x_i - x_i')^\top(f_i(x_i) + g_i(x_i)\mu_i(x_i''))) \\ &- \phi_{i,1}b_i(x_i'') - \phi_{i,2}(\rho_i^2 - ||x_i - x_i'||_2^2) - \phi_{i,3}(\rho_i^2 - ||x_i' - x_i''||_2^2) \\ &\qquad - \phi_{i,4}(\rho_i^2 - ||x_i - x_i''||_2^2)\end{aligned} \quad (9)$$

A4) Assign $\Delta_i^{\max}$ as

$$\Delta_i^{\max} = \frac{\rho_i^2 - \theta_i R_i - \psi_i R_i}{\theta_i}. \quad (10)$$

The response time R_i depends on guarantees given by the scheduler, and so it is not independent of the delay bound $\Delta_i^{\max}$. In §IV-A, we show how to transform Eq. (10) under the class of scheduling policies considered in this paper.

Proposition 2. *If $\Delta_i^{\max}$ is chosen such that Eqs.* (7)–(10) *are satisfied, then the control renders $\mathcal{W}_i$ positive invariant.*

Proof. Eqs. (7) and (8), respectively, ensure that Eqs. (4) and (5) hold for all x_i and x_i'' with $b_i(x_i'') \geq 0$ and $||x_i - x_i''||_2 \leq \rho_i$. Eq. (9) ensures that Condition 2) of Proposition 1 holds. Finally, Step A4) ensures that Condition 3) of Proposition 1 holds. □

To mitigate the computational complexity of SOS programming, sufficient conditions for Eqs. (5)–(6) can be derived. If $f_i(x_i) = F_i x_i$, $g_i(x_i) = G_i$, and $\mu_i(x_i) = K_i x_i$ for some matrices F_i, G_i, K_i, and if there exists $\gamma_i \geq 0$ such that $||x_i|| \leq \gamma_i$ for all x_i in the safe region, then

$$\min\{\theta_i, \psi_i\} \geq 2\rho_i(||F_i|| + ||G_i K_i||)\gamma_i$$

ensures that Eqs. (5)–(6) are satisfied.

B. Safe Backup Control Architecture

We now derive an approach for choosing periods T_i^N and T_i^B in order to compute the minimum sampling times for given values of β_i. We first present sufficient conditions for safety.

Proposition 3. *Suppose that there exist nonnegative parameters ρ_i^N, θ_i^N, ψ_i^N, ρ_i^B, θ_i^B, and ψ_i^B such that the following conditions are satisfied:*

(i) For all x_i and x_i' with $||x_i - x_i'|| \leq \rho_i^N$ and $b_i(x_i') \geq \beta_i$, conditions (4)–(5) *hold with $\theta_i = \theta_i^N$.*

(ii) For all x_i, x_i', x_i'' lying in a ball of radius ρ_i^N with $b_i(x_i'') \geq \beta_i$, condition (6) *holds with $\psi_i = \psi_i^N$.*

(iii) For all x_i and x_i' with $||x_i - x_i'|| \leq \rho_i^B$ and $b_i(x_i') \in [0, \beta_i]$, conditions (4)–(5) *hold with $\theta_i = \theta_i^B$.*

(iv) For all x_i, x_i', x_i'' lying in the same ball of radius ρ_i^B, condition (6) *holds with $\psi_i = \psi_i^B$.*

(v) We have $R_i^N\psi_i^N + (R_i^N + \Delta_i^N)\theta_i^N \leq (\rho_i^N)^2$ and $R_i^B\psi_i^B + (R_i^B + \Delta_i^B)\theta_i^B \leq (\rho_i^B)^2$.

Then the set $\mathcal{W}_i$ is positive invariant.

Proof. We will prove that, if $b_i(x_i(t_{i,j-1})) \geq 0$, then $b_i(x_i(t)) \geq 0$ for all $t \in [t_{i,j-1}, t_{ij}]$. Since $b_i(x_i(0)) \geq 0$, this is sufficient to imply positive invariance. We have that, if $b_i(x_i(t_{i,j-1})) \geq \beta_i$, then $b_i(x_i(t)) \geq 0$ for $t \in [t_{i,j-1}, t_{ij}]$ by (i), (ii), (v), and Proposition 1. Similarly, if $b_i(x_i(t_{i,j-1})) \in [0, \beta_i]$, then $b_i(x_i(t)) \geq 0$ for $t \in [t_{i,j-1}, t_{i,j}]$ by (iii)–(v) and Proposition 1. □

A modified version of the procedure from §III-A then can be used to compute the delay bounds $\Delta_i^{\max,N}$ and $\Delta_i^{\max,B}$ by repeating steps A1-A4 using the mode-specific variables $R_{i,j}^N$ when system ς_i operates in nominal mode and and $R_{i,j}^B$ in backup mode as discussed in §II. The delay bound $\Delta_i^{\max,N}$ denotes a safe upper bound on the elapsed time from $t_{i,j}$ when the state is sampled to the time $t_{i,j+1}'$ at which the next control signal is applied if the nominal controller is selected based on the sampled state at time $t_{i,j}$. $\Delta_i^{\max,B}$ is an upper bound on the elapsed time from the sampled state used by the backup controller to application of the next control signal. Thus, if from the state sampled at time $t_{i,j+1}$, a controller transition is triggered, the delay bound from the *previous* sample (i.e., the controller used at time $t_{i,j}$) must be respected.

IV. Optimal and Safe Scheduling

Given the scheduling model of §II-A and the control strategies outlined in §II-B, our goal now is to design policies that concurrently schedule control and non-control tasks so as to *(i)* minimize control cost within *(ii)* the safety constraints imposed

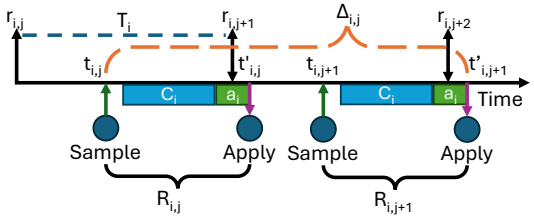


Fig. 3: The relationship between period T_i and delay interval $\Delta_{i,j}$.

by the controlled systems. In this section, we formulate this as a constrained optimization problem for systems of implicit-deadline tasks, then present a linear-time algorithm to solve the problem for utilization-based schedulability analysis.

A. Safe Scheduling

We map the safety conditions in §III to a policy for task parametrization and scheduling, such that a digital control system remains *safe* if its tasks are deemed *schedulable*.

Safe Parameters: For a control task $\tau_i \in \Gamma_{\text{CTRL}}$, C_i is assigned as the worst-case execution time to sample the system's state and compute the control signal. From this we have the following theorem, which constrains the period T_i:

Theorem 2. *If a control task τ_i is deemed schedulable when invoked at a period $T_i \leq \frac{\Delta_i^{\max} - a_i}{2}$, then the corresponding system ς_i is guaranteed to remain safe. Here, $\Delta_i^{\max}$ is the delay bound, defined as in Definition 4, and derived in §III for the considered control strategies; and a_i is the worst-case time to physically apply the computed control signal.*

Proof. Consider a job $J_{i,j}$ of task τ_i released at time $r_{i,j}$. Since the task is to sample the system state and compute the corresponding control signal, this will complete before its deadline $r_{i,j} + T_i$ if the task is schedulable, and so the sample will be obtained at some time $t_{i,j}$ for $t_0 \leq t_{i,j} \leq t_0 + T_i$. Similarly, the next job $J_{i,j+1}$ is released at time $r_{i,j+1} = r_{i,j} + T_i$ with deadline $r_{i,j} + 2T_i$, and so if the task is schedulable, the next control signal is computed at some time $c_{i,j+1}$ for $r_{i,j} + T_i + C_i \leq c_{i,j+1} \leq r_{i,j} + 2T_i$. Then the time $t'_{i,j}$ at which the control is physically applied is $c_{i,j+1} + a_i$, so $t'_{i,j+1} \leq r_{i,j} + 2T_i + a_i$. Then since $t_{i,j} \geq r_{i,j}$, we have $\Delta_{i,j} = t'_{i,j+1} - t_{i,j} \leq r_{i,j} + 2T_i + a_i - r_{i,j} = 2T_i + a_i$. Then if $T_i \leq \frac{\Delta_i^{\max} - a_i}{2}$, we have $\Delta_{i,j} \leq 2\frac{\Delta_i^{\max} - a_i}{2} + a_i = \Delta_i^{\max}$. Since $\Delta_{i,j} \leq \Delta_i^{\max}$, the system ς_i remains safe. □

We refer to this upper bound on the period T_i of task τ_i as $T_i^{\max}$. The above proof is illustrated in Fig. 3.

Schedulability: Theorem 2 says that a collection $\mathbf{S}$ of systems ς_i will remain safe if their controllers are scheduled under the following conditions:

- C1) Every task $\tau_i \in \Gamma_{\text{CTRL}}$ that implements control of system ς_i is assigned a period $T_i \leq T_i^{\max}$.
- C2) All tasks $\tau_i \in \Gamma$ are assigned implicit deadlines $D_i = T_i$.
- C3) Tasks are scheduled according to these parameters.
- C4) Sustainable analysis for the chosen scheduling algorithm deems the task set Γ to be schedulable with properly-characterized worst-case execution times C_i.

It is important to emphasize that this analysis guarantees safety even in the presence of delays between job release $r_{i,j}$ and sampling $t_{i,j}$, or delays in execution due to preemption by higher priority jobs. Such delays are illustrated in Fig. 3.

Computing the Delay Bound: Under the stated schedulability conditions, the time R_i to compute and apply a signal from controller ς_i is upper-bounded by $T_i^{\max} + a_i$. For $T_i \leq \frac{\Delta_i^{\max} - a_i}{2}$, $R_i \leq \frac{\Delta_i^{\max} + a_i}{2}$. Substituting into Eq. (10),

$$\Delta_i^{\max} = \frac{\rho_i^2 - \theta_i(\Delta_i^{\max} + a_i)/2 - \psi_i(\Delta_i^{\max} + a_i)/2}{\theta_i}.$$

Solving for $\Delta_i^{\max}$,

$$\Delta_i^{\max} = \frac{2\rho_i^2 - (\theta_i + \psi_i)a_i}{3\theta_i + \psi_i} \tag{11}$$

B. Minimizing Control Cost

Although the system's safety constraints impose a *maximum* period $T_i^{\max}$ (equivalently, a *minimum frequency* $\omega_i^{\min} = 1/T_i^{\max}$) on each control task, it may be desirable to invoke the controller at a *higher* frequency to reduce control cost. This gives rise to a constrained optimization problem of the form in Eq. (1), where periods T_i (equivalently, frequencies $\omega_i = 1/T_i$) are assigned to each control task $\tau_i \in \Gamma_{\text{CTRL}}$ to minimize control cost within the safety constraints.

Convex, Non-Increasing Control Cost: If the control cost $J(\omega)$ is monotone non-increasing as a function of frequency, it may be solved using the method of Lagrange multipliers. Here, we derive a solution that gives rise to a linear-time algorithm for a representative class of cost functions of the form:

$$J_i(\omega_i) = A_i e^{-B_i \omega_i} \tag{12}$$

where $A_i \in \mathbb{R}^+$ and $B_i \in \mathbb{R}^+$ are constant parameters of the nonlinear system ς_i and its chosen controller. Cost functions of this form have been used in the prior work [14] as approximations of real-world control objectives [25], e.g., minimizing tracking error in radar systems, loss in total work produced by a mechanical system, or minimizing energy use [8] via more frequent control inputs that allow gentler corrective actions.

Utilization-Based Schedulability Analysis: We primarily consider scheduling algorithms with utilization-based analysis, where a sufficient test compares the total system utilization to a constant bound U_D. For multiprocessor scheduling, each task's utilization is additionally constrained to not exceed 1.

C. Solution Derivation

1) Feasibility: We note that:

$$J_i'(\omega_i) = -B_i A_i e^{-B_i \omega_i}$$

Because the first derivative is negative for all $\omega_i > 0$, the cost function is monotone decreasing. Moreover:

$$J_i''(\omega_i) = B_i^2 A_i e^{-B_i \omega_i}$$

The second derivative is non-negative, so the cost function is convex. The objective (1a) is the sum of monotone decreasing, convex functions in the feasible region, and is therefore itself monotone decreasing and convex.

$$F\left(\{\omega_i\}, \lambda, \{\lambda_{i,1}\}, \{\lambda_{i,2}\}\right) = \sum_i A_i e^{-B_i \omega_i} + \lambda \left(\sum_i C_i \omega_i - U_D \right) + \sum_i \lambda_{i,1} \left(\omega_i^{\min} - \omega_i \right) + \sum_i \lambda_{i,2} \left(\omega_i - 1/C_i \right) \tag{13}$$

$$\frac{\partial F}{\partial \omega_i} = -B_i A_i e^{-B_i \omega_i} + C_i \lambda - \lambda_{i,1} + \lambda_{i,2} = 0 \tag{14a}$$

$$\sum_i C_i \omega_i - U_D \leq 0 \tag{14b}$$

$$\omega_i^{\min} - \omega_i \leq 0 \tag{14c}$$

$$\omega_i - 1/C_i \leq 0 \tag{14d}$$

$$\lambda \geq 0 \tag{14e}$$

$$\lambda_{i,1} \geq 0 \tag{14f}$$

$$\lambda_{i,2} \geq 0 \tag{14g}$$

$$\lambda \left(\sum_i C_i \omega_i - U_D \right) = 0 \tag{14h}$$

$$\lambda_{i,1} \left(\omega_i^{\min} - \omega_i \right) = 0 \tag{14i}$$

$$\lambda_{i,2} \left(\omega_i - 1/C_i \right) = 0 \tag{14j}$$

Fig. 4: Lagrangian and Karush–Kuhn–Tucker (KKT) Conditions.

2) KKT Conditions: This lets us use the method of Lagrange multipliers [26], [27]. The Lagrangian function corresponding to our stated objective in Eq. (13), and the Karush-Kuhn-Tucker (KKT) conditions in Eq. (14), are grouped for ease of reference in Fig. 4.

3) Solution Derivation: We assume that for all i, $\omega_i^{\min} < 1/C_i$. If $\omega_i^{\min} = 1/C_i$, then ω_i must be $1/C_i$, so the frequency of controller ς_i is not variable and can be removed from the problem, reducing U_D by 1. If $\omega_i^{\min} > 1/C_i$, then task τ_i would have a utilization greater than 1; we consider only sequential tasks, so we do not allow this under our system model. For completeness, we also address these trivial cases:

- If $U_D \geq n$ then $\omega_i = 1/C_i$ for all i.
- If $U_D = \sum_i C_i \omega_i^{\min}$ then $\omega_i = \omega_i^{\min}$ for all i.
- If $U_D < \sum_i C_i \omega_i^{\min}$ then no feasible solution exists, and so safety of the system cannot be guaranteed.

Otherwise, from the complimentary slackness condition (14j), we know that if $\omega_i < 1/C_i$ for any τ_i, then $\lambda_{i,2}$ is 0. Similarly, since $\lambda_{i,1} \geq 0$ (14f) the stationarity condition (14a) requires that $\lambda > 0$ since the first term is negative, and so from (14h), $\sum_i C_i \omega_i = U_D$. Intuitively, this says that if any controller's utilization is less than 1, then the total utilization is U_D — otherwise, we can still increase the frequency of some controller to achieve a better result.

Let us assume first that $\omega_i^{\min} < \omega_i < 1/C_i$ for some τ_i. The stationarity condition (14a) reduces to:

$$-B_i A_i e^{-B_i \omega_i} + C_i \lambda = 0 \tag{15}$$

Solving for ω_i:

$$\omega_i = \frac{1}{B_i} \left(\ln \left(\frac{A_i B_i}{C_i} \right) - \ln \lambda \right) \tag{16}$$

We introduce terms $\ell_i = \ln \left(\frac{A_i B_i}{C_i} \right)$ and $z = -\ln \lambda$. Then, from the primal constraints on ω_i (14c) and (14d), we have:

$$\omega_i(z) = \max \left(\min \left(\frac{1}{B_i} (\ell_i + z), \omega_i^{\max} \right), \omega_i^{\min} \right) \tag{17}$$

where $\omega_i^{\max} = 1/C_i$. It remains to find the value z for which $\sum_i C_i \omega_i(z) = U_D$, from which ω_i and T_i are computed.

Algorithm 1: ASSIGN_FREQUENCIES(Γ_{CTRL},U_D)

```
1  Input: A set Γ_CTRL of control tasks, Desired total utilization U_D
2  Output: A set {T_i} of period assignments
3  ▷ Check trivial cases
4  U^min ← 0, U^max ← 0
5  forall τ_i ∈ Γ_CTRL do
6      U^min ← U^min + C_i ω_i^min, U^max ← U^max + C_i ω_i^max
7  if U_D ≥ U^max then return {1/ω_i^max}
8  if U_D = U^min then return {1/ω_i^min}
9  if U_D < U^min then return INFEASIBLE
10 ▷ Create sorted list of minimum and maximum
     values of z for each task
11 Z ← ∅
12 forall τ_i ∈ Γ_CTRL do
13     z_i^min ← B_i ω_i^min − ℓ_i
14     z_i^max ← B_i ω_i^max − ℓ_i
15     Insert z_i^min and z_i^max into Z
16 ▷ Solve for z
17 U_D^* ← U_D − U^min
18 A ← 0, B ← 0
19 forall z_i ∈ Z do
20     if z_i is z_i^min then
21         U_D^* ← U_D^* + C_i ω_i^min
22         A ← A + ℓ_i C_i / B_i, B ← B + C_i / B_i
23     else if z_i is z_i^max then
24         U_D^* ← U_D^* − C_i ω_i^max
25         A ← A − ℓ_i C_i / B_i, B ← B − C_i / B_i
26     z ← (U_D^* − A) / B
27     if z ≤ next z_i ∈ Z then break
28 ▷ Compute periods T_i
29 forall τ_i ∈ Γ_CTRL do
30     Compute ω_i(z) per Eq. (17)
31     T_i = 1/ω_i
32 return {T_i}
```

D. Solving for z

From the KKT conditions, we derived Eq. (17), showing that our optimization problem reduces to a linear program with a single degree of freedom z. Finding optimal task frequencies within the system's safety and schedulability constraints thus requires us to solve for a single value of z for which the total utilization does not exceed the schedulable utilization bound U_D. Alg. 1 outlines such a procedure. It takes the set Γ_{CTRL} of control tasks and the desired utilization U_D; we assume the utilization demand of non-control tasks (which have fixed execution times and periods) is subtracted from the system's schedulable utilization bound in a pre-processing step.

Lines 3–9 check the trivial cases outlined previously. Here,

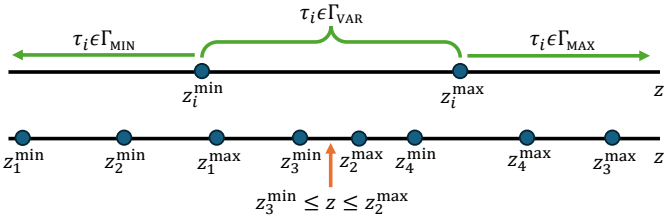


Fig. 5: Task partitions into Γ_{MIN}, Γ_{VAR}, and Γ_{MAX} with respect to z. **Top:** If $z \le z_i^{\min}$, then $\tau_i \in \Gamma_{\text{MIN}}$; if $z \ge z_i^{\max}$, then $\tau_i \in \Gamma_{\text{MAX}}$; and if $z_i^{\min} < z < z_i^{\max}$, then $\tau_i \in \Gamma_{\text{VAR}}$. **Bottom:** For the displayed ordering of $z_i^{\min}$ and $z_i^{\max}$ values, if $z_3^{\min} < z < z_2^{\max}$, then $\tau_4 \in \Gamma_{\text{MIN}}$, $\tau_2, \tau_3 \in \Gamma_{\text{VAR}}$, and $\tau_1 \in \Gamma_{\text{MAX}}$.

$\omega_i^{\min} = 1/T_i^{\max}$, where $T_i^{\max}$ is derived from $\Delta_i^{\max}$ according to Theorem 2. For sequential tasks, ω_i must be no greater than $1/C_i$ to avoid the utilization of τ_i from exceeding 1. For generality, we denote the upper bound on ω_i as $\omega_i^{\max}$, since the semantics of the corresponding control system ς_i might impose a different upper bound on invocation frequency.

If none of the trivial cases are met, we know that the value of z for which $\sum_i C_i \omega_i(z) = U_D$ (we call this z^*) is such that $\omega_i^{\min} < \omega_i(z^*) < \omega_i^{\max}$ for some τ_i. We denote $z_i^{\min}$ as the value of z for which $\omega_i(z) = \omega_i^{\min}$ and similarly for $z_i^{\max}$. Then since $\omega_i(z)$ is linear, if none of the trivial cases are met, it follows that $\min_i z_i^{\min} < z^* < \max_i z_i^{\max}$.

This means that, for the optimal value z^* of z, we can partition tasks $\tau_i \in \Gamma_{\text{CTRL}}$ into three subsets. Γ_{MIN} are those for which $z^* \le z_i^{\min}$ and so $\omega_i(z^*) = \omega_i^{\min}$. Γ_{MAX} are similarly those for which $z^* \ge z_i^{\max}$ and so $\omega_i(z^*) = \omega_i^{\max}$. And finally Γ_{VAR} are those for which $z_i^{\min} \le z^* \le z_i^{\max}$, so ω_i still remains variable with respect to z. It follows that Γ_{VAR} must be non-empty if none of the trivial cases are satisfied.

Given such a partition, from Eq. (17) we can solve for z^*:

$$\sum_{\tau_i \in \Gamma_{\text{VAR}}} C_i \left(\frac{1}{B_i} (\ell_i + z^*) \right) + \sum_{\tau_i \in \Gamma_{\text{MIN}}} C_i \omega_i^{\min} + \sum_{\tau_i \in \Gamma_{\text{MAX}}} C_i \omega_i^{\max} = U_D$$

For brevity, we introduce the following notation:

$$U_D^* = U_D - \sum_{\tau_i \in \Gamma_{\text{MIN}}} C_i \omega_i^{\min} - \sum_{\tau_i \in \Gamma_{\text{MAX}}} C_i \omega_i^{\max} \tag{18}$$

From this we get:

$$z^* = \frac{U_D^* - \sum_{\tau_i \in \Gamma_{\text{VAR}}} \frac{\ell_i C_i}{B_i}}{\sum_{\tau_i \in \Gamma_{\text{VAR}}} \frac{C_i}{B_i}} \tag{19}$$

The problem now is that the partition of tasks into sets Γ_{VAR}, Γ_{MIN}, and Γ_{MAX} is not known a priori, so we may need to test each possible partition. However, only some partitions are feasible, because *tasks move between these partitions in order of their $z_i^{\min}$ and $z_i^{\max}$ values*, as illustrated in Fig. 5. Lines 11–15 of Alg. 1 compute and sort these values into a list Z, defining the $2n-1$ possible task partitions.

For each partition, Alg. 1 solves for z according to Eq. (19). If z is too large, i.e., if z would imply a different task partition based on the values of $\omega_i^{\min}$ and $\omega_i^{\max}$, then the next partition is checked. Otherwise, the value of z is the optimal value z^*, and task periods are assigned accordingly.

Execution Time Optimization: Naïvely, for n tasks, computing z^* for a given partition takes time $\mathcal{O}(n)$, and with $2n-1$ partitions, solving for z^* takes time quadratic in the number of tasks. However, given the sorted list Z of $z_i^{\min}$ and $z_i^{\max}$ values, our algorithm finds a solution in *linear time* by tracking each term in Eq. (19); these can be updated in constant time as a task moves between partitions.

We assume that all tasks start in Γ_{MIN}, so U_D^* is initialized to $U_D - \sum_{\tau_i \in \Gamma_{\text{MIN}}} C_i \omega_i^{\min}$ (Line 17), while $\sum_{\tau_i \in \Gamma_{\text{VAR}}} \frac{\ell_i C_i}{B_i}$ (denoted $\mathcal{A}$) and $\sum_{\tau_i \in \Gamma_{\text{VAR}}} \frac{C_i}{B_i}$ (denoted $\mathcal{B}$) are initialized to 0. The algorithm then iterates over values of z in Z. For values $z_i^{\min}$, task τ_i is moved from Γ_{MIN} to Γ_{VAR}; U_D^* is therefore increased by $C_i \omega_i^{\min}$ since the frequency of τ_i is now variable (Line 21), and $\mathcal{A}$ and $\mathcal{B}$ are increased by the appropriate amounts (Line 22). Values $z_i^{\max}$ represent moving task τ_i from Γ_{VAR} to Γ_{MAX}; U_D^* is therefore decreased by $C_i \omega_i^{\max}$ since the frequency of τ_i is now fixed (Line 24), and $\mathcal{A}$ and $\mathcal{B}$ are reduced accordingly (Line 25).

Once values U_D^*, $\mathcal{A}$, and $\mathcal{B}$ are updated, z can be computed in constant time (Line 26). For n tasks, execution time is therefore $\mathcal{O}(n \log n)$ to compute and sort values in Z, $\mathcal{O}(2n-1) = \mathcal{O}(n)$ time to solve for z^*, and finally $\mathcal{O}(n)$ time to compute periods; total time is thus $\boldsymbol{\mathcal{O}(n \log n)}$.

E. Application to Utilization-Based Scheduling Algorithms

We now discuss how to use Alg. 1 to assign periods to control tasks scheduled with utilization-based analysis.

Uniprocessor Scheduling: The Liu and Layland model of periodic, implicit-deadline tasks [1] tells us that for preemptive EDF scheduling, the utilization bound $U_D = 1$. For task-level fixed-priority scheduling, rate-monotonic priority assignment is optimal, and achieves a utilization bound of $U_D = n(\sqrt[n]{(2)} - 1)$ for a set Γ of n tasks. In both cases, $U_D \le 1$, granting a refinement that improves the execution time of Alg. 1 if the maximum frequency of each task τ_i is simply restricted by the requirement that individual utilizations do not exceed 1, i.e., $\omega_i^{\max} = 1/C_i$. In this case, the optimal value z^* satisfies $z^* < z_i^{\max}$ for all τ_i, so Z can be constructed to only include values $z_i^{\min}$.

Fluid Scheduling: Under the fluid scheduling paradigm [28], individual tasks τ_i are assigned a fraction f_i of a processor at each instant in time. Implementations exist to approximate it, e.g., under the RT-FAIR scheduling framework in LITMUS^RT [29]. It is a convenient abstraction under which implicit-deadline tasks are schedulable on m cores so long as no single task has a utilization exceeding 1, and so long as the total utilization does not exceed the number of processor cores. Alg. 1 therefore can be applied directly.

Partitioned EDF Scheduling: While the fluid paradigm is a convenient abstraction for multicore scheduling, it often remains impractical in real systems [30]. Partitioned scheduling is an attractive alternative that is straightforward to implement. Under partitioned EDF, tasks are first assigned to processor cores, then each core independently schedules its tasks according to EDF. The task system is thus schedulable if there exists

an assignment of tasks to cores such that the total utilization on each core does not exceed 1. This determination is equivalent to bin-packing, and is therefore NP-hard in the strong sense. Nonetheless, the first-fit and best-fit descending heuristics are guaranteed to produce a schedulable allocation on m cores so long as total utilization does not exceed $U_D = \frac{m+1}{2}$.

This gives rise to two natural approaches toward assigning task periods with low control cost while remaining safe under partitioned EDF. The first is to invoke Alg. 1 with a utilization bound of $U_D = \frac{m+1}{2}$. If UNSCHEDULABLE is not returned, a heuristic is guaranteed to find a schedulable partition for the assigned periods. At this point, Alg. 1 can be invoked for *each task partition*, re-assigning frequencies to achieve a utilization of up to $U_D = 1$ on each core. Alternatively, one can simply partition tasks according to their minimum safe frequencies, then invoke Alg. 1 for each individual core.

F. Numerical Method for Alternative Schedulability Analysis

From Eq. (17), $\omega_i(z)$ is piecewise linear, non-decreasing on z. Because control cost decreases with ω_i, we can state our optimization problem as one of *finding the maximum value of z for which the task system is schedulable*. To do so, we can perform a linear or binary search in the interval $z \in [\min_i z_i^{\min}, \max_i z_i^{\max}]$ at a chosen granularity. For every value of z that is tested, $\omega_i(z)$ is calculated for each task, and the system is tested for schedulability.

Partitioned EDF: This observation gives us a third approach for partitioned EDF scheduling. For every value of z that is tested, a heuristic may be used to find a partition. The largest value of z for which a schedulable partition is still found defines the assignment of tasks to cores; again, Alg. 1 then may be invoked for tasks on each individual core to further increase their frequencies. §V-A compares the three stated approaches.

G. Online Adaptation

A key distinction for the second control strategy introduced in §II-B is that a system may transition between a nominal controller and a backup controller, depending on its distance from an unsafe region. In such cases, the control cost function coefficients A_i, B_i, delay bound $\Delta_i^{\max}$, and execution time C_i to compute the control signal may change, requiring new periods be assigned to minimize cost while maintaining safety.

Consider a control task τ_i that at time $t_{i,j+1}$ samples the state of system ς_i and determines it must transition from nominal to safe backup mode. The execution time and maximum period therefore change from C_i^N and $T_i^{\max,N}$ to C_i^B and $T_i^{\max,B}$, respectively. Fig. 1 illustrates the challenge: the delay interval is defined as the time from sampling to the application of the control signal computed from the *next* sample. Therefore, although the new controller has an updated delay bound $\Delta_i^{\max,B}$, the *previous* delay bound $\Delta_i^{\max,N}$ applies and *its* deadline must be met. To avoid transient overload, for each task we assign C_i as the maximum computation time among its own control law and that of any controller it may transition to. Although pessimistic, this is sufficient to allow C_i to subsume the execution time of a controller after a transition; additional analysis based on, e.g., online algorithms or semi-clairvoyant mixed-criticality theory [31] to derive necessary conditions and lower-bound C_i are deferred to future work.

We now consider how to reassign task periods across a controller transition. Without loss of generality, assume a system ς_i transitions from nominal to safe backup mode based on the state sampled at time $t_{i,j+1}$ (the same arguments apply for transitions from backup to nominal mode). The constrained optimization problem now may be solved using the new input parameters, e.g., by using Alg. 1, to obtain new periods for each task. How and when the new periods are assigned is case-dependent, and relies on the observation in [32] that it is safe to *increase* a task's period at any time, and to *decrease* it at the next job activation, but not necessarily before.

If the transitioning control task's period *decreases*, then because of the small fixed-length prefix during which the transition condition is checked and the optimization problem is solved, the new period is selected during active job execution. Therefore, the current job's period (and deadline) must be kept the same, and it is not until the release of the *next* job at time $r_{i,j+2}$ that the new period is applied. However, observe from Fig. 1 that this still respects the delay bound from the previous state sampled at time $t_{i,j}$. For all other tasks, it is also safe to delay a period decrease until the next job release, and period increases can be applied immediately since new periods still respect the unchanged delay interval used as a constraint in the optimization problem. On the other hand, if the transitioning control task's period *increases*, the extended period/deadline can only be applied immediately if the resulting worst-case delay interval still respects the delay bound from the previous state sampled at time $t_{i,j}$. Otherwise, we must wait until the next job release at time $r_{i,j+2}$ to extend the period. This means that all *other* tasks τ_k in the system must continue to execute at their previous periods until that time. If the reassigned period is shorter, this new period will be applied at the first release $r_{k,j}$ of τ_k that occurs after $r_{i,j+2}$.

V. EVALUATION

A. Evaluation of Scheduling

We first evaluate our scheduling strategies over a broad space of parameters using randomly-generated synthetic task sets. We aim to gauge how efficiently Alg. 1 runs for online reassignment of task frequencies during controller mode transitions. We also compare the schedulability rates and control costs achieved by our multiprocessor allocation strategies.

Algorithm Execution Times: To quantify the overhead incurred by Alg. 1, we measure its execution time for a large number of synthetic task sets. We generate sets of tasks τ_i of size n from 2–50, and with total minimum safe utilizations $U_{\text{SUM}}^{\min} = \sum_{\tau_i} C_i/T_i^{\max}$ in the range 0.1–0.9 in steps of 0.1. For each combination of $(n, U_{\text{SUM}}^{\min})$, we generate 1000 task sets. Parameters A_i and B_i of the representative control cost function in Eq. (12) are randomly drawn from a uniform distribution over $[0, 1)$, while $T_i^{\max}$ is selected from the range 1–1000 using a log-uniform distribution as recommended in [33].

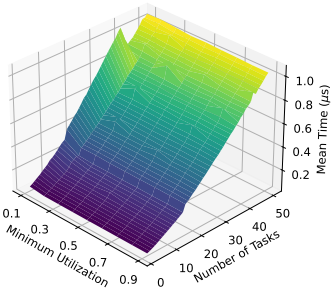


Fig. 6: Mean execution times of Alg. 1.

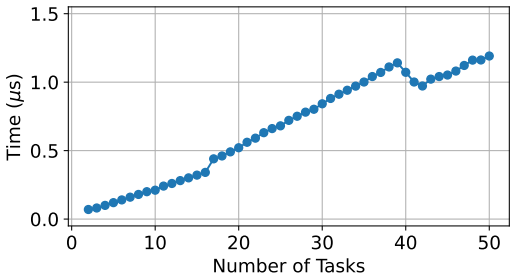


Fig. 7: 99^{th}-percentile execution times of Alg. 1.

The Dirichlet Rescale (DRS) Algorithm [34] distributes the total minimum utilization $U_{\text{SUM}}^{\min}$ in an unbiased random fashion across individual tasks; from these and the $T_i^{\max}$ values, execution times C_i are derived. The value $\ell_i = \ln(\frac{A_i \cdot B_i}{C_i})$ is precomputed for each task.

These synthetic task sets are intended to explore a large space of possible combinations of key task parameters without reference to any specific controller. The results generalize to systems with co-scheduled non-control tasks, as these reduce the processor utilization available to the set Γ_{CTRL} of control tasks, but do not fundamentally change the optimization problem. Guided by the results presented in this paper, as future work we will design new evaluations using specific parameters for particular controllers, including mode-specific variations.

We implement Alg. 1 in C++, compiling with GCC optimization level `-O3` and static linking. We run it on a single core of an AMD Threadripper PRO 7985WX with 128GB of RAM running Linux 6.8.0-45-generic. Fig. 6 plots a surface showing mean execution times for each combination $(n, U_{\text{SUM}}^{\min})$. We observe that the algorithm is very efficient, executing in about 1.2μs on average for sets of 50 tasks. For lower values of $U_{\text{SUM}}^{\min}$, execution time tends to increase slightly because there is more room to increase task frequencies from their minimum safe values, so the algorithm must iterate over more task partition boundaries $z_i^{\min}$.

Nonetheless, there is not a strong dependence, so we also look at the 99^{th}-percentile execution times aggregated across all values $U_{\text{SUM}}^{\min}$ for each n. These are plotted in Fig. 7. 99^{th}-percentile times remain below 1.2μs overall, reinforcing that this algorithm is suitable for online transitions without inducing substantial overhead. Notably, though its overhead tends to increase with n, there is a slight *decrease* from 39 to 42 tasks. We found this to be consistent over multiple reruns, and traced this to the C++ standard library's sort routine.

Multiprocessor Allocation: Next, we consider the co-scheduling of a large number of control tasks on a multicore system. We generate systems of control tasks of size n in the range 32–100 in steps of 2, and numbers of cores m in the range 2–16. For each combination (n, m), we draw $U_{\text{SUM}}^{\min}$ uniformly from the interval $[1, m)$, then use DRS [34] to distribute this without exceeding 1 for an individual task. Parameters A_i, B_i, $T_i^{\max}$, and C_i are assigned as before.

We evaluate the three strategies outlined in §IV-E, §IV-F for allocating tasks to processors and assigning periods under partitioned EDF: *(i)* **P-EDF**, where tasks are first partitioned according to $T_i^{\max}$, then Alg. 1 is invoked for each core. *(ii)* **P-EDF-U**, where Alg. 1 is invoked with $U_D = \frac{m+1}{2}$, tasks are partitioned, then periods are re-assigned on each core. *(iii)* **P-EDF-OPT**, where the largest value of z for which task frequencies ω_i computed per Eq. (17) achieve a schedulable partition is found by linear search, then periods are again re-assigned on each core by Alg. 1. We use the first-fit descending (by utilization) bin packing heuristic, and for P-EDF-OPT, we search for z with a step size $(\max_i z_i^{\max} - \min_i z_i^{\min})/1000$.

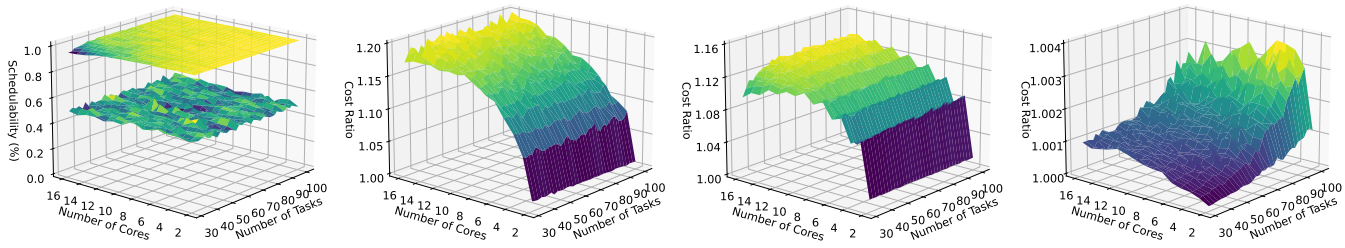
We compare these approaches to fluid scheduling, which provides us with theoretically-optimal bounds on schedulability and control cost. Fig. 8a shows the schedulability rates achieved by each approach. The lower surface is P-EDF-U, which successfully schedules about half of the evaluated task sets. This makes sense, as our method for generating utilizations achieves an average total of $\frac{m+1}{2}$, exactly the guaranteed bound. As both P-EDF and P-EDF-OPT first attempt to partition tasks according to their maximum safe periods, their schedulability rates are equivalent, and are reflected by the upper surface; even with 16 tasks on 32 cores, the heuristic successfully schedules 94.6% of the task systems.

Fig. 8b–d show the median ratio of the control cost achieved by each approach to that of fluid scheduling; a ratio of 1 is therefore optimal. Notice that P-EDF-U obtains a slightly better control cost than P-EDF; P-EDF-U should therefore be preferred over P-EDF unless the system is deemed unschedulable by the utilization bound. However, P-EDF-OPT achieves the lowest cost ratio by far; despite its higher runtime complexity, it is the preferred approach for offline allocation.

B. Evaluation of Control Safety

We evaluated the safety provided by our approach on a nonlinear flight dynamics model. The flight dynamics of an aircraft can be decomposed into lateral and longitudinal components. The longitudinal component has state variable $x_1(t) \in \mathbb{R}^4$, where the states represent airspeed, angle of attack, body pitch rate, and pitch angle. The control input is scalar and equal to the elevator command. We use the textbook longitudinal dynamics given by [35]:

$$\dot{x}_1(t) = \begin{pmatrix} Z_\alpha/V & Z_\alpha & 0 & Z_\delta \\ M_\alpha/Z_\alpha & M_q & \left(M_\delta - M_\alpha \frac{Z_\delta}{Z_\alpha}\right) & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\omega^2 & -2\zeta\omega \end{pmatrix} x_1(t) + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \omega^2 \end{pmatrix} u_1(t) \quad (20)$$



(a) Schedulability Rates **(b)** Cost Ratio: P-EDF **(c)** Cost Ratio: P-EDF-U **(d)** Cost Ratio: P-EDF-OPT.

Fig. 8: Comparison of partitioned EDF core allocation and period assignment approaches.

Consistent with [35], we choose Z_α=−1.053, Z_δ=−0.0343, M_α=−2.33, M_q=−1.033, M_δ=−1.17, V=329, ω=26π, and ζ=0.6. The lateral component has state variable $x_2(t) \in \mathbb{R}^4$, where the states represent yaw rate, roll rate, roll angle, and yaw angle. The control inputs $u(t) \in \mathbb{R}^2$ are the aileron and rudder angles. The nonlinear lateral dynamics are given by

$$\dot{x}_2(t) = \begin{pmatrix} \phi_1(x) \\ \frac{1}{V_0}(Y_\beta[x_2]_2 + Y_p[x_2]_3 + Y_r[x_2]_4) + \frac{g\cos\theta_0}{V_0}\sin[x_2]_1 \\ L_\beta[x_2]_2 + L_p[x_2]_3 + L_r[x_2]_4 \\ N_\beta[x_2]_2 + N_p[x_2]_3 + N_r[x_2]_4 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ \frac{Y_{ail}}{V_0} & \frac{Y_{rud}}{V_0} \\ L_{ail} & L_{rud} \\ N_{ail} & N_{rud} \end{pmatrix} u_2(t) \quad (21)$$

where

$$\phi_1(x) = [x_2]_3\cos\alpha_0 - r_s\sin\alpha_0 + \tan\theta_0 q_0 \sin[x_2]_1 + ([x_2]_4\cos\alpha_0 + [x_2]_3\sin\alpha_0)\cos[x_2]_1$$

and $[x_2]_j$ denotes the j^{th} component of $[x_2]$. We define the safety constraint for the systems as $x_1(t)^\top x_1(t) \leq 1$ and $x_2(t)^\top x_2(t) \leq 1$, modeling the requirement that both the lateral and longitudinal components must remain sufficiently close to the desired equilibrium point. We consider a scenario in which a single processor is used to compute the control inputs for the lateral and longitudinal modes. We use the backup safe controller architecture defined in §III-B. The nominal and backup controllers for each subsystem are constructed as follows. For each $i = 1, 2$, we construct a stabilizing control law $u_i = -K_i x_i$ by solving an infinite-horizon LQR problem

$$\int_0^\infty x_i(t)^\top Q x_i(t) + u_i(t)^\top R u_i(t)\ dt \quad (22)$$

with $Q = 10^{-3}I$ and $R = I$. For the nonlinear lateral dynamics, we solve Eq. (22) using the linearization of Eq. (21) around the origin. For the longitudinal dynamics, we assume that the goal of the system is to track a step reference $z = (0\ 0\ 0.5\ 0)^\top$, and hence define the nominal control law $\mu_1^N(x_1) = -K_1 x_1 + N_1 z$, where N_1 is chosen to ensure conference to the desired reference value (see, e.g., [36]). The safe backup control law is given by $\mu_1^B(x_1) = -K_1 x_1$. For the lateral dynamics, we assume that the goal of the system is to stabilize the system state to 0, and hence define both the nominal and backup control laws as $\mu_2^N(x_2) = \mu_2^B(x_2) = -K_2 x_2$.

To construct the control barrier functions, for each $i = 1, 2$, we let $b_i(x_i) = c_i - x_i^\top P_i x_i$, where P_i denotes the positive definite solution to the Algebraic Riccati Equation [37]

$$F_i^\top P_i + P_i F_i - P_i G_i R^{-1} G_i^\top P_i + Q = 0.$$

The term $x_i^\top P_i x_i$ is equal to the value function of the infinite-horizon LQR problem (22), while c_i is chosen as the largest possible value of c satisfying $\{x : x^\top P_i x \leq c\} \subseteq \{x : x^\top x \leq 1\}$. For more details on this approach for constructing CBFs, see [38]. We chose representative control cost parameters as $A_i = B_i = 1$ for both systems. For switching to the safe backup mode, we chose a threshold representing close proximity to the edge of the safe region: $\beta_i = 0.9c_i$. For these efficient controllers, we assume an execution time $C_i = 10\mu$s.

We first computed the control parameters for this system model according to the methods in §III. For the longitudinal dynamics, the nominal controller parameters are ρ_1^N=0.5, θ_1^N=3.6 × 10^3, ψ_1^N=3.6 × 10^3, $T_1^{\max,N}$=5 × 10^{-5}s and the backup controller parameters are ρ_1^B=0.467, θ_1^B=237, ψ_1^B=237, and $T_1^{\max,B}$=9 × 10^{-4}. For the lateral dynamics, nominal parameters are ρ_2^N=0.5, θ_2^N=2.1826, ψ_2^N=2.1826, $T_2^{\max,N}$=0.0573s, and β_2^N=2.185 · 10^-4 and backup parameters are ρ_2^B=0.5, θ_2^B=2.231, ψ_2^B=2.231, and $T_2^{\max,B}$=0.056s.

We then simulated the system trajectories under this control model. The simulation results are shown in Figure 9. The value of $||x_1(t)||_2$ is shown in Figure 9(a). Since the target state lies outside the safe region $\{x : b_1(x) \geq 0\}$, the nominal controller will steer the system towards the safety boundary. When $b_1(x_1(t)) < \beta_1$, the backup controller is engaged to steer the system back towards the interior of the safe region. This leads to the oscillating behavior observed in the figure. The lateral dynamics, shown in Figure 9(b), are governed by the stabilizing LQR controller and remain within the safe region over the duration of the simulation.

VI. Related Work

As early as 1996, Seto et al. considered how to ensure feasible scheduling of real-time control tasks while optimizing control performance [8]. They presented an integrated approach to control and scheduling design, in which relationships between task periods and control performance are modeled explicitly, along with lower bounds on tasks' periods (upper bounds on frequency). Although they also considered cost functions of the form given by Eq. (12), the work presented in our paper considers both upper and lower bounds on task periods, explicitly framing ranges of safely exploitable

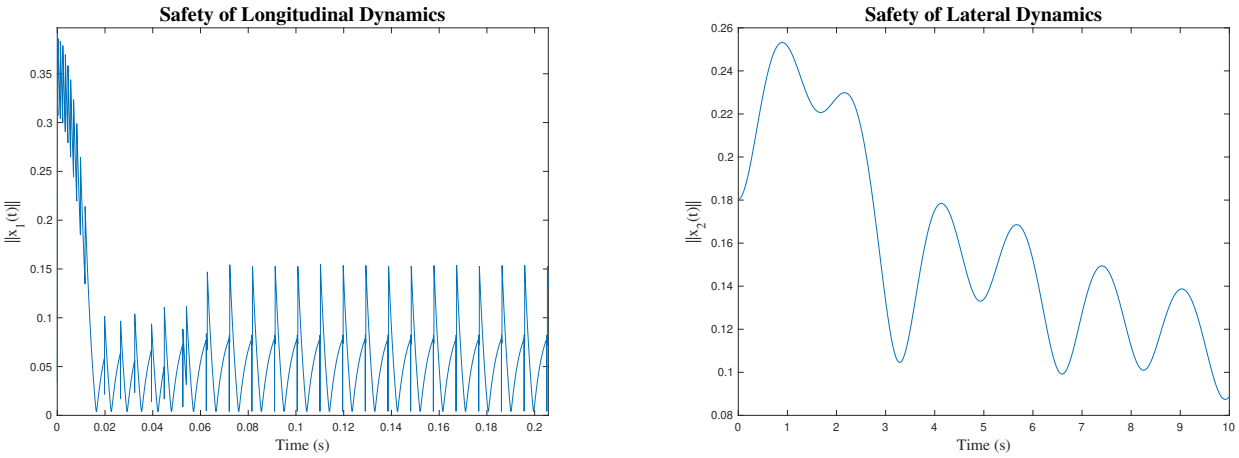


Fig. 9: Simulation of a nonlinear aircraft dynamics model in which the decomposed longitudinal and lateral dynamics are controlled by the same processor. For both longitudinal states $x_1(t)$ and lateral states $x_2(t)$, the safety constraint is to satisfy $||x_1(t)||_2 \leq 1$ and $||x_2(t)||_2 \leq 1$ for all t. The longitudinal states $x_1(t)$ have an additional reference tracking objective. **Left**: Plot of $||x_1(t)||$ for the longitudinal dynamics. The controller switches between nominal and safe backup modes as it approaches the safety envelope. **Right**: Plot of lateral dynamics. The agent states remain within the safe region.

adaptation. In 2001, Aydin et al. [39] considered concave reward functions more generally, but only provided linear-time optimization techniques for linear reward functions. By using the KKT conditions shown in Fig. 4 to linearize the problem, we enable the numerical method in §IV-F for schedulability analysis beyond a static utilization bound.

Recently, Gifford et al. published new results [40] on co-design of real-time control and scheduling for multi-mode systems and provided a summary of other related work on co-design more broadly across the intervening decades: [41]–[54]. In [55], [56], conditions on the sampling frequency were introduced to ensure closed-loop stability of a linear system, which is distinct from the safety properties considered in our work. A more complex scheduling/co-design formulation was proposed in [57], [58], treating the control inputs and sample times as optimization variables in a non-convex optimization problem. In this paper, we model control and scheduling parameters jointly, so as to minimize control cost while maintaining schedulability and control safety, both (1) within a single controller as it transits sub-regions of the system state space in which the parameters may differ, and (2) across transitions between high-performance controllers and more conservative backup controllers. Also recently, Baruah et al. considered control-scheduling co-design involving mitigative controllers, which can accommodate delays in one iteration of the control loop through corrective actions in the subsequent one(s) [59]. That introduces a novel form of workload elasticity that they show can be integrated rigorously with real-time scheduling assurance. In this paper we model period-elastic adaptation, though extending the constrained-optimization approach presented here to consider both period-elastic and workload-elastic adaptation appears achievable as future work.

Safety verification of cyber-physical systems also has seen recent interest [19], [20]. Much such work assumes a CPS either operates in continuous or discrete time with a fixed sampling interval. The most closely related works present control barrier functions for sampled-data systems [9]–[12] but assume the sample periods are given and aim for robustness bounds on the CBFs to ensure that sampling effects do not compromise safety. We instead present constraints on the sampling frequency to ensure safety with a given control law, which is new. Moreover, our SOS-based safety constraints do not to our knowledge appear in the existing literature. Our approach is also novel in bounding delay between sampled state (at time $t_{i,j}$) and application of a control signal based on the *next* sample (at time $t'_{i,j+1}$). Most existing work on delay-based period assignment, e.g. in [60], bounds the sample/apply delay within a single job interval $t_{i,j}$ to $t'_{i,j}$.

VII. Conclusions and Future Work

This paper has presented a new formalization of the periodicity requirements for control tasks to guarantee avoidance of unsafe portions of the system state space. Control tasks also must be *schedulable* so that these requirements are met. This reinforces the fundamental connection between schedulability and safety, with a framework to guarantee that control signals are applied in a timely manner, even when the system transitions online between nominal and more conservative backup controllers. We express this as a constrained optimization problem, where task periods are assigned to *minimize control cost* within the implied safety and schedulability constraints.

As future work, we will also consider schedule-driven control, developing controller models informed by the computational resources of the system. We will explore other co-design principles and control strategies, including controllers for which glitches in internal state tracking may need to be detected and corrected. Moreover, we will address fundamental scheduling questions raised by our approach, e.g., how the delay bound changes if tasks' deadlines may be shorter than their periods, and does this enable extended periods and thus better schedulability; what other forms of control cost functions can be treated similarly; and can we leverage predictions or semi-clairvoyant mixed-criticality theory to be more optimistic in our assignment of execution times across transitions?

ACKNOWLEDGMENTS

This research was supported by NSF grants CNS-2303563, CMMI-2418806, CNS-2229290, NASA award 80NSSC21K1741, and a WashU OVCR seed grant.

REFERENCES

[1] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[2] J. Solem, "The application of microrobotics in warfare," 9 1996. [Online]. Available: https://www.osti.gov/biblio/369704

[3] J. J. Abbott, Z. Nagy, F. Beyeler, and B. J. Nelson, "Robotics in the small, part i: Microbotics," *IEEE Robotics & Automation Magazine*, vol. 14, no. 2, pp. 92–103, 2007.

[4] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood, "Controlled flight of a biologically inspired, insect-scale robot," *Science*, vol. 340, no. 6132, pp. 603–607, 2013. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1231806

[5] J. Kim, H. Kim, K. Lakshmanan, and R. Rajkumar, "Parallel scheduling for cyber-physical systems: Analysis and case study on a self-driving car," in *2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2013, pp. 31–40.

[6] A. Li, H. Liu, J. Wang, and N. Zhang, "From timing variations to performance degradation: Understanding and mitigating the impact of software execution timing in slam," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[7] A. Li, J. Wang, S. Baruah, B. Sinopoli, and N. Zhang, "An empirical study of performance interference: Timing violation patterns and impacts," in *2024 Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2024.

[8] D. Seto, J. Lehoczky, L. Sha, and K. Shin, "On task schedulability in real-time control systems," in *17th IEEE Real-Time Systems Symposium*, 1996, pp. 13–21.

[9] G. Bahati, P. Ong, and A. D. Ames, "Sample-and-hold safety with control barrier functions," in *2024 American Control Conference (ACC)*. IEEE, 2024, pp. 5169–5176.

[10] J. Breeden, K. Garg, and D. Panagou, "Control barrier functions in sampled-data systems," *IEEE Control Systems Letters*, vol. 6, pp. 367–372, 2021.

[11] L. Niu, H. Zhang, and A. Clark, "Safety-critical control synthesis for unknown sampled-data systems via control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6806–6813.

[12] P. S. Oruganti, P. Naghizadeh, and Q. Ahmed, "Robust control barrier functions for sampled-data systems," *IEEE Control Systems Letters*, 2023.

[13] L. Sha, "Dependable system upgrade," in *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No.98CB36279)*, 1998, pp. 440–448.

[14] R. Chandra, X. Liu, and L. Sha, "On the scheduling of flexible and reliable real-time control systems," *Real-Time Systems*, vol. 24, no. 2, pp. 153–169, Mar 2003. [Online]. Available: https://doi.org/10.1023/A:1021726418716

[15] S. Baruah and A. Burns, "Sustainable scheduling analysis," in *2006 27th IEEE International Real-Time Systems Symposium (RTSS'06)*. IEEE, 2006, pp. 159–168.

[16] S. Sastry, *Nonlinear systems: analysis, stability, and control*. Springer Science & Business Media, 2013, vol. 10.

[17] A. Clark, "Verification and synthesis of control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6105–6112.

[18] W. Zhao, T. He, T. Wei, S. Liu, and C. Liu, "Safety index synthesis via sum-of-squares programming," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 732–737.

[19] O. So, Z. Serlin, M. Mann, J. Gonzales, K. Rutledge, N. Roy, and C. Fan, "How to train your neural control barrier function: Learning safety filters for complex input-constrained systems," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 11 532–11 539.

[20] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[21] Y. Chen, M. Jankovic, M. Santillo, and A. D. Ames, "Backup control barrier functions: Formulation and comparative study," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6835–6841.

[22] A. R. Kumar, K.-C. Hsu, P. J. Ramadge, and J. F. Fisac, "Fast, smooth, and safe: implicit control barrier functions through reach-avoid differential dynamic programming," *IEEE Control Systems Letters*, 2023.

[23] A. Papachristodoulou and S. Prajna, "A tutorial on sum of squares techniques for systems analysis," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 2686–2700.

[24] H. Khalil, *Nonlinear systems*. Prentice Hall, 2002.

[25] K. Shin, C. Krishna, and Y.-H. Lee, "A unified method for evaluating real-time computer controllers and its application," *IEEE Transactions on Automatic Control*, vol. 30, no. 4, pp. 357–366, 1985.

[26] W. Karush, "Minima of functions of several variables with inequalities as side constraints," *M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago*, 1939.

[27] H. KUHN, "Nonlinear programming," in *Proc. 2nd Berkeley Symposium, 1951*. Univ. of California Press, 1951, pp. 481–492.

[28] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, "Proportionate progress: A notion of fairness in resource allocation," *Algorithmica*, vol. 15, no. 6, pp. 600–625, Jun 1996. [Online]. Available: https://doi.org/10.1007/BF01940883

[29] J. M. Calandrino, H. Leontyev, A. Block, U. C. Devi, and J. H. Anderson, "$LITMUS^{RT}$: A testbed for empirically comparing real-time multiprocessor schedulers," in *2006 27th IEEE International Real-Time Systems Symposium (RTSS'06)*, 2006, pp. 111–126.

[30] M. Sudvarg, C. Gill, and S. Baruah, "Improved implicit-deadline elastic scheduling," in *Proceedings of the 14th IEEE International Symposium on Industrial Embedded Systems (SIES 2024)*. IEEE, 2024. [Online]. Available: https://sudvarg.com/publications/SIES2024_improved_implicit_elastic.pdf

[31] K. Agrawal, S. Baruah, and A. Burns, "Semi-clairvoyance in mixed-criticality scheduling," in *2019 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2019, pp. 458–468.

[32] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic scheduling for flexible workload management," *IEEE Transactions on Computers*, vol. 51, no. 3, pp. 289–302, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1109/12.990127

[33] P. Emberson, R. Stafford, and R. Davis, "Techniques for the synthesis of multiprocessor tasksets," in *WATERS workshop at the Euromicro Conference on Real-Time Systems*, Jul. 2010, pp. 6–11.

[34] D. Griffin, I. Bate, and R. I. Davis, "Generating Utilization Vectors for the Systematic Evaluation of Schedulability Tests," in *2020 IEEE Real-Time Systems Symposium (RTSS)*, 2020, pp. 76–88.

[35] E. Lavretsky and K. A. Wise, "Robust adaptive control," in *Robust and adaptive control: With aerospace applications*. Springer, 2012, pp. 317–353.

[36] C.-T. Chen, *Linear system theory and design*. Oxford University Press, 1984.

[37] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2004.

[38] A. Clark, "A semi-algebraic framework for verification and synthesis of control barrier functions," *To appear in IEEE Transactions on Automatic Control (TAC)*, 2024.

[39] H. Aydin, R. Melhem, D. Mossé, and P. Mejia-Alvarez, "Optimal reward-based scheduling for periodic real-time tasks," *IEEE Transactions on Computers*, vol. 50, no. 2, pp. 111–130, 2001.

[40] R. Gifford, F. Galarza-Jimenez, L. Phan, and M. Zamani, "Decntr: Optimizing safety and schedulability with multi-mode control and resource allocation co-design," in *30th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2024.

[41] K. Arzen, A. Cervin, J. Eker, and L. Sha, "An introduction to control and scheduling co-design," in *Conference on Decision and Control (CDC)*, vol. 5, 2000, p. 4865–4870.

[42] M. Schmitz, B. Al-Hashimi, and P. Eles, "A co-design methodology for energy-efficient multi-mode embedded systems with consideration of mode execution probabilities," in *Design, Automation and Test in Europe*, 2003, p. 960–965.

[43] D. Simon, D. Robert, and O. Sename, "Robust control/scheduling codesign: application to robot control." in *11th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2005, p. 118–127.

[44] M. Gaid, A. Cela, and Y. Hamam, "Optimal integrated control and scheduling of networked control systems with communication constraints: application to a car suspension system," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 4, p. 776–787, 2006.

[45] G. Buttazzo, M. Velasco, and P. Marti, "Quality-of-control management in overloaded real-time systems," *IEEE Transactions on Computers*, vol. 56, no. 2, p. 253–266, 2007.

[46] R. Schneider, D. Goswami, S. Zafar, M. Lukasiewycz, and S. Chakraborty, "Constraint-driven synthesis and tool-support for flexray based automotive control systems," in *IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, 2011, p. 139–148.

[47] D. Soudbakhsh, L. Phan, A. Annaswamy, O. Sokolsky, and I. Lee, "Co-design of control and platform with dropped signals," in *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2013.

[48] H. Chwa, K. Shin, and J. Lee, "Closing the gap between stability and schedulability: A new task model for cyber-physical systems," in *24th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2018, p. 327–337.

[49] A. Gujarati, M. Nasri, and B. Brandenburg, "Quantifying the resiliency of fail-operational real-time networked control systems," in *Euromicro Conference on Real-Time Systems (ECRTS)*, 2018.

[50] D. Soudbakhsh, L. Phan, A. Annaswamy, and O. Sokolsky, "Codesign of arbitrated network control systems with overrun strategies," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, p. 128–141, 2018.

[51] Q. Zhu and A. Sangiovanni-Vincentelli, "Codesign methodologies and tools for cyber–physical systems," *Proceedings of the IEEE*, vol. 106, no. 9, p. 1484–1500, 2018.

[52] L. Scheuvens, A. H¨oßler, T. Barreto, and G. Fettweis, "Wireless control communications co-design via application-adaptive resource management," in *5G World Forum (5GWF)*, 2019, p. 298–303.

[53] X. Dai, S. Zhao, Y. Jiang, X. Jiao, X. Hu, and W. Chang, "Fixed priority scheduling and controller co-design for time-sensitive networks." in *Conference on Computer-Aided Design*, 2020, p. 1–9.

[54] D. Roy, S. Ghosh, Q. Zhu, M. Caccamo, and S. Chakraborty, "Goodspread: Criticality-aware static scheduling of cps with multi-qos resources," in *2020 IEEE Real-Time Systems Symposium (RTSS)*, 2020, pp. 178–190.

[55] S. Reimann, W. Wu, and S. Liu, "Real-time scheduling of pi control tasks," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 1118–1125, 2015.

[56] P. Marti, C. Lin, S. A. Brandt, M. Velasco, and J. M. Fuertes, "Draco: Efficient resource management for resource-constrained control tasks," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 90–105, 2008.

[57] M. E. M. B. Gaid, A. S. Cela, and Y. Hamam, "Optimal real-time scheduling of control tasks with state feedback resource allocation," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 2, pp. 309–326, 2008.

[58] D. Gorges, M. Izak, and S. Liu, "Optimal control of systems with resource constraints," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 1070–1075.

[59] S. Baruah, M. Hosseinzadeh, I. Kolmanovsky, and B. Sinopoli, "Adaptive scheduling for real-time control," in *Proceedings of the 32nd International Conference on Real-Time Networks and Systems*, ser. RTNS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1–10.

[60] E. Bini and A. Cervin, "Delay-aware period assignment in control systems," in *2008 Real-Time Systems Symposium*, 2008, pp. 291–300.