A New Perspective on Quality Evaluation for Control Systems with Stochastic Timing

Maximilian Gaukler
Chair of Automatic Control
Friedrich-Alexander-Universität Erlangen-Nürnberg
max.gaukler@fau.de

Andreas Michalka
Chair of Automatic Control
Friedrich-Alexander-Universität Erlangen-Nürnberg
andreas.michalka@fau.de

Peter Ulbrich
Chair of Distributed Systems and Operating Systems
Friedrich-Alexander-Universität Erlangen-Nürnberg
peter.ulbrich@fau.de

Tobias Klaus
Chair of Distributed Systems and Operating Systems
Friedrich-Alexander-Universität Erlangen-Nürnberg
tobias.klaus@fau.de

ABSTRACT

As control applications are particularly sensitive to timing variations, the Quality of Control (QoC) is degraded by varying execution conditions of the underlying real-time system. In particular, transitions between different execution or environmental conditions pose a significant issue as they may impact the QoC unexpectedly.

So far, the QoC is usually evaluated either in a stationary, time-invariant way, which cannot analyze said transitions, or by simulation, which becomes inefficient when confronted with random influencing factors. In this paper, we propose a new perspective on QoC evaluation for modern, adaptive real-time systems with varying timing conditions. For this, we present a time-variant stochastic assessment approach that incorporates the effects mentioned before. Our results demonstrate that adaptive scheduling and runtime behavior considerably impacts the QoC. At the same time, the proposed scheme significantly outperforms a traditional simulation.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; *Real-time systems*; • **Computing methodologies** → *Systems theory*; • **Mathematics of computing** → Stochastic processes; • **Software and its engineering** → Scheduling.

KEYWORDS

Quality of Control, Jitter, Real-time Systems, Automatic Control

1 INTRODUCTION

Compliance with an application-specific physical specification is a primary design objective of real-time control systems: in a vehicle, this is, for example, to keep lane in a centimeter tolerance range. Further improvement (i. e., millimeter accuracy) does not lead to further increase in specification compliance or general benefit. Accordingly, from a control-theoretical point of view, the system must be designed and assessed to provide a sufficient Quality of Control (QoC) under all possible environmental conditions (e. g., wind). Typically, the QoC is quantified using a quadratic cost function

$$J = x^T Q x + u^T R u \tag{1}$$

based on the state error x and the control-signal u: a small deviation from the desired state $x = 0$ and small amount of actuation corresponds to minimum cost J and therefore the maximum QoC.

Automatic control is particularly sensitive to timing variations due to its close connection to the outside physical world. Any deviation from the assumed temporal properties may negatively impact the QoC [6, 19, 23]. Thus, the real-time system (i. e., operating system) is tuned for specific execution conditions (i. e., timing of computation and input/output) to provide an appropriate Quality of Service (QoS) to the control application running on top. Here, quality refers to accurate timing, such as deadline adherence or periodicity (i. e., absence of jitter), which are commonly used as assessment criteria. In practice, the prevailing point of view is that overall QoC should be optimized by maximizing the QoS, which boils down to tightening temporal bounds [14, 23].

In contrast, current trends in real-time systems foster a well-directed renouncement from this rigid interpretation by moving away from achieving the best possible QoC but rather one that is good enough: approaches such as dynamically reconfigurable systems or mixed-criticality scheduling trade QoS to boost average performance while easing system design as well as worst-case handling. For example, mixed-criticality scheduling [5, 22] provides multiple criticality levels, each with the expectation of a certain quality (here: QoC). Such approaches are, however, typically limited to QoS-guarantees in the form of adapted timing and execution conditions (e. g., control tasks may change priority or even be omitted) for each criticality level. Consequently, it is assumed that there is a static mapping between the QoS and the actually relevant QoC. This static assumption is, for example, also shared by feedback scheduling techniques [6, 8, 20]. Ultimately, deadlines may even be intentionally violated for runtime adaptivity. A vivid example being weakly-hard scheduling of automatic control tasks [3] such that in any window of m execution periods deadlines have only to be met for at least $n < m$ times.

1.1 Motivation

In summary, automatic control will be faced with more dynamic real-time computing systems, whose timing properties will be less predictable than they used to be. The question of which QoC results from these execution conditions is non-trivial. Although execution conditions (i. e., timing deviations) and environmental conditions are both determining factors for the QoC, the timing is typically not considered in the classical design process.

Existing approaches to evaluate the QoC under consideration of the actual runtime behavior, such as JITTERBUG [15], typically operate on stationary scenarios. This means that dedicated execution conditions (i. e., driving conditions) are considered individually and the effects are only evaluated time-averaged. Therefore, transitions between different conditions cannot be analyzed.

However, we showcased in previous work [13] that the dynamic behavior at these transitions is crucial for a sound assessment of the actual QoC if the environment or timing can vary over time. Thus, systemic evaluation schemes which capture time-varying conditions are urgently needed to take advantage of said advances in real-time system design, scheduling, and adaptivity without harming the actual objective: providing a *sufficient Quality of Control*.

1.2 Problem Statement

In this work we consider a system consisting of a physical plant and a digital controller (e. g., lane control in a car) that is implemented on a real-time operating system. We are looking for a numerical answer to the question: which Quality of Control results from the combined negative impacts of

- non-perfect input/output timing
- stochastic physical disturbance, such as side wind,
- measurement noise
- and the control situation, e. g., fast curve vs. straight road?

Because the system is subject to varying situations, these factors vary over time, resulting in a *time-dependent* QoC. Due to system dynamics (e. g., mechanical inertia) the dependency between QoC and situation is not a static but a *dynamic* function, as it depends on the history: a car, for example, cannot change its position instantaneously, but will only gradually head towards or away from the desired path, and therefore the current position accuracy heavily depends on the *prior* wind strength and input/output timing.

1.3 Contribution

In this paper, we present a novel approach to QoC evaluation for adaptive, non-deterministic real-time systems. We consider such an evaluation technique as a necessary and self-contained starting point towards a QoC-aware co-design of control application and real-time executive. We claim the following key contributions:

- New insights into the influence of runtime adaptive real-time systems on time-related QoC.
- A theoretically well-founded QoC evaluation scheme that is, to the best of our knowledge, the first to incorporate time-varying input and output delays and stochastic disturbances.
- An extended system model that facilitates the evaluation of Multiple-Input/Multiple-Output (MIMO) systems.
- A prototypical implementation of our approach that substantially outperforms traditional simulation-based approaches.

The remainder of this paper is structured as follows: First, Section 2 discusses related work. Section 3 and 4 introduce notation and underlying system model used in this document, respectively. Section 5 presents our primary approach for deterministic timing, which is subsequently extended to stochastic timing in Section 6. Section 7 gives an overview of essential parts of our implementation and exemplifies its performance. Section 8 concludes the main part and provides an outlook on current and upcoming work. Active readers find the main proofs detached in Section 9.

2 DISCUSSION OF RELATED WORK

With the aim of achieving both high control performance and resource efficiency, researchers have been considering many facets of the co-design of computing and control systems for more than 20 years. From a computing point of view, methods for flexible timing that respect the QoC have been developed, ranging from off-line optimization up to on-line adaptation to current environment conditions. From a control standpoint, the complementary approach of considering timing in the controller synthesis was proposed, with the aim of either passive robustness to unknown or active compensation of known delays. This development continued towards a co-synthesis of controller and static timing for individual environmental conditions. As an overarching approach, radically new architectures were proposed that combine controller and adaptive timing generator: With Event-Triggered Control, the control signal is only recomputed when the sensor value violates a certain condition. For Self-Triggered Control, the next sampling time is adjusted depending on the current error.

While these approaches offer efficiency improvements, they come at the expense of partly or completely moving away from well-established and easy-to-use control and real-time design methodologies. It is, therefore, an important question whether their use is necessary for a given system, especially when migrating existing, well-tested control designs to newer real-time architectures. This task of analysis and the underlying relation between timing and QoC has mostly been considered only implicitly.

One important point of distinction is the theoretical frameworks employed for analysis or synthesis: With the aim of addressing *typical* control performance, a linear-quadratic stochastic framework has been widely employed, such as in the work of Nilsson et al. for the synthesis of a delay-compensating optimal controller [18]. Most closely related to the aim of our work is the JITTERBUG toolbox by Lincoln and Cervin [10, 15], which explicitly computes the average QoC of a general linear system consisting of continuous- and discrete-time elements and noise sources. The delays are described by a Markov chain, also allowing for a controller dependent on current timing. However, only the stationary QoC is calculated, and all stochastic parameters have to remain constant, which means that time-varying external disturbance or timing influence cannot be modeled. For offline control-scheduling co-design, Fontanelli et al. [12] applied stochastic optimization to a similar framework.

From a safety perspective, stability or *worst-case* performance is analyzed to provide a lower bound on QoC. It should be noted that while this analysis is an important aspect of system design, it does not address the complementary question of performance in the typical, average case. The approach by Fontanelli et al. discussed before verifies stochastic stability in the mean-square sense. Lyapunov theory is used especially in the context of Event-Triggered Control. Based on an overapproximation of the set of possible system states (Reachability Analysis), Al Khatib et al. [1] present a method for stability analysis and timing synthesis of control systems with uncertain control periods, although their work is restricted to output feedback controllers and zero sampling-to-actuation delay.

A second key point of distinction is the approach to time discretization: Nilsson et al. use a classical one-step discretization in which the timestep is one full nominal control cycle, causing a difficult dependency between discrete transition matrix and delays. In JITTERBUG and the work of Fontanelli et al., this is simplified by limiting the Input/Output (I/O) timepoints to a grid, which is also used for discretization. The result is a discrete system of significantly higher sampling rate that switches between a finite number of transition matrices. Instead of an explicit discretization, Al Khatib et al. employ the formulation as a linear impulsive system, which is also used by many works on Event-Triggered Control.

3 NOTATION

We will use the following notation: The transpose of x is denoted x^T. I is the unity matrix and $e_j = \begin{bmatrix}0 & \dots & 0 & 1 & 0 & \dots & 0\end{bmatrix}^T$ the j-th unit vector, both of appropriate dimension. Intervals are denoted (a,b) for open, $[a,b]$ for closed and $(a,b] := \{x \,|\, a < x \leq b\}$: half-open interval. For a piecewise-continuous function $f(t)$, the left- and right-hand limits at $t = a$ are denoted $f(a^-)$ and $f(a^+)$. $\mathbb{N}$ is set of positive integers, $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$, and $\mathbb{R}$ the set of real numbers.

Random variables. For a random variable x, $\mathcal{E}\{x\}$ denotes the expectation and $\Pr(x = x_i)$ the probability, which is abbreviated as $\Pr(x_i)$. Undetermined random variables in an expectation may be annotated by a subscript as in $\mathcal{E}\{f(x)\} = \mathcal{E}_x\{f(x)\}$, which means that the sum $\mathcal{E}\{f(x)\} = \sum_i \Pr(x = x_i)\, f(x_i)$ iterates only over i. This subscript does *not* change the mathematical meaning; conditional expectations are always stated explicitly such as $\mathcal{E}_x\{f(x)|y\}$.

By $\operatorname{cov}\{x(t)\} := \mathcal{E}\left\{x(t)x^T(t)\right\}$ we define the time-varying covariance. For the sake of simplicity, we also apply this definition to signals with $\mathcal{E}\{x(t)\} \neq 0$, slightly abusing the name covariance.

4 SYSTEM MODEL

Plant. The physical system to be controlled is described as a linear time-invariant plant

$$\dot{x}_\mathrm{p}(t) = A_\mathrm{p} x_\mathrm{p}(t) + B_\mathrm{p} u(t) + G_\mathrm{p} d(t), \quad t > 0, \quad x_\mathrm{p}(0) = 0, \tag{2}$$

$$y(t) = C_\mathrm{p} x_\mathrm{p}(t) + w_\mathrm{p}(t) \tag{3}$$

with input $u(t) = \begin{bmatrix}u_1(t) & u_2(t) & \dots & u_m(t)\end{bmatrix}^T \in \mathbb{R}^m$, state $x_\mathrm{p}(t) \in \mathbb{R}^{n_\mathrm{p}}$ and output $y(t) = \begin{bmatrix}y_1(t) & \dots & y_p(t)\end{bmatrix}^T \in \mathbb{R}^p$. It is influenced by the stochastic disturbance $d(t) \in \mathbb{R}^{n_\mathrm{dist}}$ and measurement noise $w_\mathrm{p}(t) \in \mathbb{R}^p$, whose random properties are stated later.

Controller. The plant is controlled by the discrete-time controller

$$\begin{aligned} x_\mathrm{d}[k+1] &= A_\mathrm{d}[k] x_\mathrm{d}[k] + B_\mathrm{d}[k] y[k] + f_\mathrm{d}[k], \quad x_\mathrm{d} \in \mathbb{R}^{n_\mathrm{d}}, \\ u[k] &= C_\mathrm{d}[k] x_\mathrm{d}[k] + g_\mathrm{d}[k], \quad x_\mathrm{d}[0] = 0, \quad k \in \mathbb{N}_0. \end{aligned} \tag{4}$$

Beyond a standard linear controller, such as discretized PID or observer-based state feedback, this formulation also allows for time-varying controllers and, through the constant terms $f_\mathrm{d}[k]$ and $g_\mathrm{d}[k]$, for tracking a deterministic reference trajectory.

The given form has no feedthrough, which means that the control signal $u[k]$ only requires the previous measurement $y[k-1]$. This restriction is reasonable for control systems in which timing is relevant, as it permits a computation time of slightly below one control period, while with feedthrough any computation time would inevitably delay the output.

At this point, the time-variant formulation can be used to model arbitrary timing-dependent execution, especially controllers that compensate for varying delays, although this will be restricted later. As it will be seen in the derivations, the restriction to linear plants and controllers is crucial for the theoretical results that permit an efficient computation of the QoC.

I/O timing. We assume a fixed nominal controller period of T and asynchronous I/O behaviour, which means there is no special synchronization hardware that ensures that all sampling and actuation happens exactly at the start of the control period. Such a synchronization would require support for a global I/O clock signal, which is typically not available in ADC/DAC units integrated in microcontrollers and especially not for peripherals connected via buses such as I2C or CAN.

The main reason for nondeterministic I/O time variation is interference: resources of the real-time system, such as CPU and communication buses for digital sensors, are shared between control, I/O and other tasks, causing delays if a resource is already in use. While allocating fixed time slots based on worst-case execution times can solve this problem for simple systems, it is no longer feasible with growing system complexity and flexibility, and highly inefficient due to the high ratio between worst-case and average execution time. This is one argument for mixed-critical systems or dynamic allocation in general.

Nominally, the whole measurement vector $y[k]$ is sampled at kT. From this, the control signal $u[k+1]$ is computed and then emitted at $(k+1)T$. The real timing differs from this: The j-th control output component is delayed by $\Delta t_{\mathrm{u},j}[k]$ (where negative values represent a too early output), which can formally be stated as

$$u_j(t) = u_j[k], \quad kT + \Delta t_{\mathrm{u},j}[k] < t < (k+1)T + \Delta t_{\mathrm{u},j}[k+1]. \tag{5}$$

Respectively, the sample $y_j[k]$ of the j-th sensor is acquired with a time offset $\Delta t_{\mathrm{y},j}[k]$.

$$y_j[k] = y_j(kT + \Delta t_{\mathrm{y},j}[k]) \tag{6}$$

The timing of input, computation and output and its dataflow dependencies (double arrows) are shown in Figure 1. To avoid the need of extra buffers in the realization and corresponding buffer states in the resulting model, neighboring cycles k and $k+1$ are separated by a timing barrier $T_\mathrm{b}[k]$, which no event may cross:

$$\left.\begin{matrix} kT + \Delta t_{\mathrm{y},j}[k] \\ kT + \Delta t_{\mathrm{u},j}[k] \end{matrix}\right\} < T_\mathrm{b}[k] < \left\{\begin{matrix} (k+1)T + \Delta t_{\mathrm{y},j}[k+1] \\ (k+1)T + \Delta t_{\mathrm{u},j}[k+1] \end{matrix}\right. \quad \forall j,k \tag{7}$$

This barrier coincides with the controller computation and can equivalently be described by a task with zero execution time and four virtual dependencies as shown in Figure 1. For example, choosing $T_\mathrm{b}[k] = (k + \frac{1}{2})T$ limits all delays to $|\Delta t_{\dots}| < \frac{T}{2}$.

Without loss of generality the barrier is restricted to $T_\mathrm{b}[k] \in \big(kT,\ (k+1)T\big)$, which means that each control period k includes its nominal time kT and all delays are shorter than one period.

To simplify the timing model, we assume that no two input or output components are sampled at the same time, although this restriction will be lifted later.

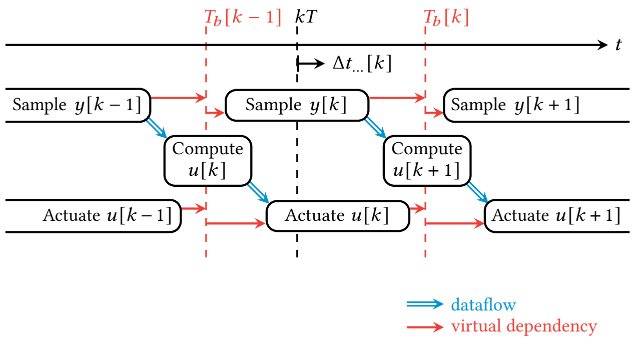

Figure 1: Timing model for I/O and computation with dataflow dependencies, the timing barrier $T_b[k]$ and its corresponding virtual dependencies.

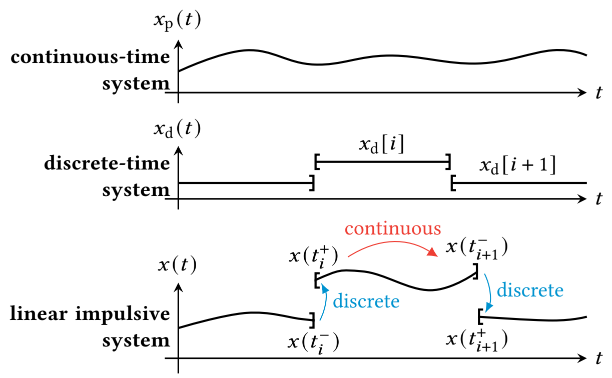

Figure 2: A linear impulsive system allows for both continuous-time and discrete-time dynamics.

Random Variables. The system is influenced by three independent sources of randomness: disturbance $d(t) \in \mathbb{R}^{n_{\text{dist}}}$, measurement noise $w_p(t) \in \mathbb{R}^p$ and the timing sequence $\mathcal{T}$.

The timing sequence $\mathcal{T} := (t_i, A_i, N_i)_{i\in\mathbb{N}}$ consists of I/O event times t_i and event matrices (A_i, N_i), which encode the type of event, such as "sample the fifth sensor", and a possibly timing-dependent execution. A formal definition of A_i and N_i will be given later.

Disturbance and measurement noise are subsumed as noise $\mathcal{N} := (d(\cdot), w_p(\cdot))$, which formally is a tuple of functions. The disturbance $d(t)$ is modeled as average-free white noise with mean $\mathcal{E}\{d(t)\} = 0$ and autocorrelation function

$$\mathcal{E}\left\{d(t)d^T(t+\tau)\right\} = H(t)\delta(\tau), \quad H(t) \in \mathbb{R}^{n_{\text{dist}}\times n_{\text{dist}}}, \tag{8}$$

where δ is the Dirac delta functional. As a mild technical assumption, $H(t)$ is assumed to be piecewise constant and may only change at event times t_i, although this may be weakened by introducing virtual (no-operation) events. $H(t)$ itself is deterministically known.

Continuous-time white noise is the idealization of a signal with constant spectral density within a very large bandwidth. While it greatly simplifies the calculations, a mathematically strict definition of the delta functional and the differential equations containing it is intricate. This topic is further elucidated in Section 9.1.1.

The measurement noise after sampling is assumed to be discrete-time white noise. Its fictitious continuous-time source is the average-free continuous-time noise $w_p(t)$ of constant and finite covariance:

$$\mathcal{E}\left\{w_p(t)\right\} = 0, \quad \mathcal{E}\left\{w_p(t)w_p^T(t+\tau)\right\} = \begin{cases} N_p, & \tau = 0, \\ 0, & \text{else.} \end{cases} \tag{9}$$

The previous assumption that sensor channels are sampled one at a time implies that all samples have uncorrelated noise and the matrix $N_p \in \mathbb{R}^{p\times p}$ can be chosen as diagonal without loss of generality.

Quality of Control. As a measure of the time-dependent QoC we use the instantaneous quadratic cost

$$\begin{aligned} J(t) = {} & (x_p(t) - x_r(t))^T \tilde{Q} (x_p(t) - x_r(t)) \\ & + (u(t) - u_r(t))^T \tilde{R} (u(t) - u_r(t)), \end{aligned} \tag{10}$$

which weights state and control deviations from a reference trajectory $x_r(t), u_r(t)$ with $\tilde{Q} \in \mathbb{R}^{n_p\times n_p}$ and $\tilde{R} \in \mathbb{R}^{m\times m}$, respectively. The higher this cost, the worse the current QoC. The quadratic form allows derivations within a linear-quadratic stochastic framework and is commonly used for optimal control (e. g., Riccati controller).

5 APPROACH FOR DETERMINISTIC TIMING

Our approach is based on reformulating the presented system model as a linear impulsive system (Section 5.1). Under the assumption of deterministic timing, we time-discretize the system (Section 5.2) and evaluate the QoC (Section 5.3). The results will subsequently be extended to stochastic timing in Section 6. To improve readability, main proofs are detached to Section 9.

5.1 Reformulation as Linear Impulsive System

To obtain a compact formulation of the closed-loop dynamics, we introduce the total system state

$$x := \left[x_p^T(t) \;\; x_d^T(t) \;\; y_d^T(t) \;\; u^T(t) \;\; 1\right]^T \in \mathbb{R}^n, \tag{11}$$

which combines the continuous-time plant state x_p with piecewise-constant "discrete-time" states x_d, y_d and u for controller, sensors and actuators. To simplify the treatment of signals with nonzero average, a constant 1 is added as the last state entry, yielding a total order of $n = n_p + n_d + p + m + 1$. The block matrices given in the following sections are divided according to these state components.

Using this state, the dynamics can be rewritten as a continuous-time linear system with input d that is interrupted by linear jumps at the time instants t_i, $i \in \mathbb{N}$, $t_{i+1} > t_i > 0$:

$$\dot{x}(t) = Ax(t) + Gd(t), \quad t > 0, \quad t \neq t_i, \quad x(0) = x_0, \tag{12}$$

$$\begin{aligned} & x(t_i^+) = A_i x(t_i^-) + N_i^{1/2} v_i, \\ & \mathcal{E}\{v_i\} = 0, \quad \mathcal{E}\left\{v_i x^T(t_i^-)\right\} = 0, \quad \mathcal{E}\left\{v_i v_i^T\right\} = I. \end{aligned} \tag{13}$$

As illustrated in Figure 2, combining continuous-time and discrete-time states in one state vector x results in this linear impulsive system with both continuous and discrete dynamics. Here, the continuous dynamics (12) model the plant, while the discrete jumps (13) represent sampling, actuation and the discrete-time controller.

The random increment per jump is factorized as $N_i^{1/2} v_i$ to separate its time-varying covariance N_i from the actual randomness, which is represented by a random variable v_i of unity covariance. $N_i^{1/2}$ is defined by the Cholesky decomposition $N_i^{1/2} N_i^{1/2^T} := N_i$, which is not necessarily unique, but always exists because any covariance matrix is symmetric and positive semidefinite [4].

In the following, the system matrices for the closed-loop control are determined. From there on, only two cases need to be handled: a discrete jump from $x(t_i^-)$ to $x(t_i^+)$ and the continuous transition from $x(t_i^+)$ to the state $x(t_{i+1}^-)$ just before the next jump. Together, these two cases describe the evolution from $x(t_i^+)$ to $x(t_{i+1}^+)$, which equals a time discretization with time steps placed at the I/O events.

5.1.1 Continuous Dynamics. Between two events the discrete-time states are constant, that is

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x_{\mathrm{d}}^T(t) & y_{\mathrm{d}}^T(t) & u^T(t) & 1 \end{bmatrix}^T = 0. \tag{14}$$

Combining this with the plant ODE (3) yields

$$A = \begin{bmatrix} A_{\mathrm{p}} & 0 & 0 & B_{\mathrm{p}} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} G_{\mathrm{p}} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{15}$$

5.1.2 Discrete Events. The discrete transition and noise matrices for the k-th control cycle are given in the following.

Sample. The j-th sensor's value is acquired as

$$y_{\mathrm{d},j}(t_i^+) = e_j^T\left(C_{\mathrm{p}}x_{\mathrm{p}}(t_i^-) + w_{\mathrm{p}}(t_i^-)\right), \quad t_i = kT + \Delta t_{\mathrm{y},j}[k], \tag{16}$$

where the unit vector e_j used to select one output channel is of length p. As the other entries of y_{d} remain unchanged, it holds that

$$\begin{aligned} y_{\mathrm{d}}(t_i^+) &= (I - e_je_j^T)y_{\mathrm{d}}(t_i^-) + e_jy_{\mathrm{d},j}(t_i^+) & (17)\\ &= \underbrace{(I - e_je_j^T)y_{\mathrm{d}}(t_i^-) + e_je_j^TC_{\mathrm{p}}x_{\mathrm{p}}(t_i^-)}_{\tilde{A}_ix(t_i^-)} + \underbrace{e_je_j^Tw_{\mathrm{p}}(t_i^-)}_{\tilde{v}_i}. & (18)\end{aligned}$$

It can be shown that because of its stochastic properties, the measurement noise $w_{\mathrm{p}}(t_i^-)$ and therefore also $\tilde{v}_i$ is independent of the current state $x(t_i^-)$. Therefore (18) can be extended to the whole state x and noise vector v so that it matches (13). Extending $\tilde{A}_i$ and $\tilde{N}_i = \operatorname{cov}\{\tilde{v}_i\}$ accordingly results in the event matrices

$$A_i = \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ e_je_j^TC_{\mathrm{p}} & 0 & I - e_je_j^T & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{19}$$

$$N_i = \begin{bmatrix} 0 & & & & \\ & 0 & & & \\ & & e_je_j^TN_{\mathrm{p}}e_je_j^T & & \\ & & & 0 & \\ & & & & 0 \end{bmatrix}. \tag{20}$$

Compute. After all sensors have been sampled, their measurements, which were saved in y_{d}, are used to update the controller state x_{d} so that the new output is available for actuation:

$$x_{\mathrm{d}}(t_i^+) = x_{\mathrm{d}}[k+1] = A_{\mathrm{d}}[k]x_{\mathrm{d}}(t_i^-) + B_{\mathrm{d}}[k]y_{\mathrm{d}}(t_i^-) + f_{\mathrm{d}}[k]\cdot 1 \tag{21}$$

$$\Leftrightarrow \quad A_i = \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & A_{\mathrm{d}}[k] & B_{\mathrm{d}}[k] & 0 & f_{\mathrm{d}}[k] \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad N_i = 0. \tag{22}$$

The update formally takes place at $t_i = T_{\mathrm{b}}[k]$, which is the timing barrier between the control cycles k and $k+1$ as introduced in (7). As the computation itself has no physical effect, the time of this event is not relevant as long as the order of events is preserved. Therefore, it may be merged with the preceding or successive event.

Actuate. Similarly to sampling, emitting the j-th actuator's value

$$u_{\mathrm{d},j}(t_i^+) = e_j^T(C_{\mathrm{d}}[k]x_{\mathrm{d}}(t_i^-) + g_{\mathrm{d}}[k]\cdot 1), \quad t_i = kT + \Delta t_{\mathrm{u},j}[k], \tag{23}$$

where the unit vector e_j is of length m, is described by

$$A_i = \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & e_je_j^TC_{\mathrm{d}}[k] & 0 & I - e_je_j^T & e_je_j^Tg_{\mathrm{d}}[k] \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad N_i = 0. \tag{24}$$

Merging concurrent events. The restriction that no I/O events happen simultaneously can be lifted easily: To sample multiple sensors $j_{1,2,\dots}$ at once, e_j can be replaced with the matrix $\begin{bmatrix} e_{j_1} & e_{j_2} & \dots \end{bmatrix}$. The same holds for multiple actuators. If two events, such as sampling and actuation, occur at the same time $t_1 = t_2$, they can be merged into one with $A_i = A_2A_1$ and $N_i = N_2 + A_2N_1A_2^T$.

5.2 Stochastic Discretization for Deterministic Timing

For evaluating the performance without computing a multitude of simulations with different random noise inputs, we use the time-varying state covariance $P(t) := \mathcal{E}\{x(t)x^T(t)\}$ as internal representation. For now, the timing will be assumed to be deterministic, although this restriction will be removed in Section 6.

Start. The initial state is deterministically known as

$$x(0) = x_0 = \begin{bmatrix} 0 & \dots & 0 & 1 \end{bmatrix}^T \quad \Rightarrow P(0) = x_0x_0^T. \tag{25}$$

The zero entries of x_0 may be changed without loss of generality. An extension to random initial states is also possible.

Discrete events. From the discrete system equation (13) and the stochastic properties of v_i, the covariance update can be derived as

$$P(t_i^+) = \mathcal{E}\Bigg\{ A_i x(t_i^-) x^T(t_i^-) A_i^T + N_i^{1/2} \underbrace{v_i v_i^T}_{\mathcal{E}\{\cdot\}=I} N_i^{1/2^T} + \underbrace{v_i x^T(t_i^-)}_{\mathcal{E}\{\cdot\}=0} A_i^T + A_i \underbrace{x(t_i^-) v_i^T}_{\mathcal{E}\{\cdot\}=0} N_i^{1/2^T} \Bigg\} \tag{26}$$

$$= A_i P(t_i^-) A_i^T + N_i + 0 + 0 =: \mathrm{jump}(P(t_i^-), A_i, N_i). \tag{27}$$

Continuous dynamics. The continuous evolution from $x(t_i^+)$ to $x(t_{i+1}^-)$ between two discrete jumps can be described by the discrete covariance update equation

$$\begin{aligned} P(t_{i+1}^-) =& \Phi(t_{i+1}^- - t_i^+)\, P(t_i^+)\, \Phi^T(t_{i+1}^- - t_i^+) \\ &+ \Psi(t_{i+1}^- - t_i^+, H(t_i^+)) \\ =:& \mathrm{elapse}\left(P(t_i^+),\ t_{i+1}^- - t_i^+,\ H(t_i^+)\right) \end{aligned} \tag{28}$$

$$\text{with } \Phi(\Delta) = \mathrm{e}^{A\Delta}, \quad \Psi(\Delta, H) = \int_0^\Delta \mathrm{e}^{A\tau} G H G^T (\mathrm{e}^{A\tau})^T \,\mathrm{d}\tau. \tag{29}$$

The proof was detached to Section 9.1.2. This result makes use of the technical assumption that the noise covariance $H(t)$ remains constant in the concerned timespan (t_i, t_{i+1}). A faster method of calculating Φ and Ψ is obtained using the results of Van Loan [17]:

$$\Phi = M_3^T,\ \Psi = M_3^T M_2 \tag{30}$$

$$\text{with } \begin{bmatrix} M_1 & M_2 \\ 0 & M_3 \end{bmatrix} = \exp\left(\Delta \begin{bmatrix} -A & GHG^T \\ 0 & A^T \end{bmatrix}\right),\ M_{1,2,3} \in \mathbb{R}^{n\times n}. \tag{31}$$

By these calculations, the continuous covariance update can be mapped to an equivalent discrete covariance update, as it holds that

$$\mathrm{jump}(P, \Phi(\Delta), \Psi(\Delta, H)) = \mathrm{elapse}(P, \Delta, H). \tag{32}$$

Therefore, the result can be interpreted as a discrete-time stochastic system. This discretization is not an approximation.

Combined result. Following the alternating chain of discrete events (jump($\cdot$)) and continuous evolution inbetween (elapse($\cdot$)), the covariance $P(t)$ can be calculated at every event and, if needed, also at arbitrary time points. This stochastic evaluation immediately yields the covariance without computing individual random trajectories as in a Monte-Carlo simulation.

We denote the covariance evolution from $t = a^+$ to $t = b^+$ as

$$P(b^+) =: \underset{(a,b]}{F}\left(P(a^+), \mathcal{T}\right), \quad b > a, \tag{33}$$

which is the combined result of the stochastic discretization presented in this section and, implicitly, the event matrices given in Section 5.1. An implementation is given in Algorithm 1.

Two important properties of F will be used later: Firstly, $F(P, \mathcal{T})$ is a quadratic matrix form in P just as the functions jump($\cdot$) and elapse($\cdot$) it is composed of, that is

$$\underset{(a,b]}{F}(P, \mathcal{T}) = M_1^T(\mathcal{T}, a, b)\, P\, M_1(\mathcal{T}, a, b) + M_2(\mathcal{T}, a, b), \tag{34}$$

where $M_{1,2}$ are matrix-valued functions.

Secondly, the evolution from $t = 0$ to a later time $t = b^+$ can be split at any intermediate time point $t = a^+$, where $b > a > 0$:

$$P(b^+) = \underset{(0,b]}{F}(x_0 x_0^T, \mathcal{T}) = \underset{(a,b]}{F}\Bigg(\underbrace{\underset{(0,a]}{F}\left(x_0 x_0^T, \mathcal{T}_{(0,a]}\right)}_{P(a^+)}, \mathcal{T}_{(a,b]}\Bigg) \tag{35}$$

Algorithm 1 Covariance computation for deterministic timing

```
Input: Start covariance $P(t = a^+)$, time range $(a, b]$ with $b > a > 0$
    and list of events $\mathcal{T} = (t_i, A_i, N_i)_i$.
Output: Covariance $P(t = b^+)$.
  $t \leftarrow a$
  $P \leftarrow P(t = a^+)$
  for all $i = 1, 2, \ldots, \infty$ with $t_i \in (a, b]$ do
    $P \leftarrow$ elapse($P$, $t_i - t$, $H(t^+)$)
    $t \leftarrow t_i$
    $P \leftarrow$ jump($P$, $A_i$, $N_i$)
  end for
  if $t < b$ {No event at the end time.} then
    $P \leftarrow$ elapse($P$, $b - t$, $H(b)$) {Elapse the remaining time.}
  end if
  $P(t = b^+) \leftarrow P$
```

This can be seen from the causal iterative structure of Algorithm 1, in which P can be interpreted as stochastic state: $P(b^+)$ only depends on an initial value $P(a^+)$ and the subsequent timing $\mathcal{T}_{(a,b]}$, which is defined as the subsequence of the complete timing $\mathcal{T}$ with $t_i \in (a, b]$. In Equation (35) the timing dependencies are narrowed because the evolution $F_{(a,b]}$ only processes the events $\mathcal{T}_{(a,b]}$ within the considered time range.

5.3 QoC Evaluation

To simplify the following derivations, the QoC defined in (10) can be rewritten as the quadratic form $J(t) = x^T(t)\, Q(t)\, x(t)$ with

$$Q(t) = \begin{bmatrix} \tilde{Q} & 0 & 0 & 0 & -\tilde{Q} x_\mathrm{r}(t) \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \tilde{R} & -\tilde{R} u_\mathrm{r}(t) \\ -x_\mathrm{r}^T(t)\tilde{Q} & 0 & 0 & -u_\mathrm{r}^T(t)\tilde{R} & x_\mathrm{r}^T(t)\tilde{Q}x_\mathrm{r}(t) + u_\mathrm{r}^T(t)\tilde{R}u_\mathrm{r}(t) \end{bmatrix}. \tag{36}$$

It should be noted that $Q(t)$ is time-varying, but deterministically known. The resulting cost $J(t)$ is random. For a classical simulation, one has to choose a particular random sequence for disturbance and measurement noise. To obtain a representative value for comparison, we instead take the expectation $\mathcal{E}\{J(t)\}$ with regard to the noise $\mathcal{N}$ to obtain a noise-averaged, but time-dependent QoC.

This expectation can be computed from the covariance using

$$\mathcal{E}\{J(t)\} = \mathcal{E}\left\{x^T(t)\, Q(t)\, x(t)\right\} = \mathrm{trace}\Big(Q(t) \underbrace{\mathcal{E}\left\{x(t) x^T(t)\right\}}_{P(t)}\Big) \tag{37}$$

[21, p. 11]. The required covariance is calculated by Algorithm 1.

For ease of computation, the cost is evaluated at one discrete sampling point per cycle. While other choices or finer discretizations are possible, we choose this point as the timing barrier $T_\mathrm{b}[k]$ to ensure that timing changes cannot move the update of u across the evaluation point. Furthermore, it does not matter for this choice whether $T_\mathrm{b}^+[k]$ or $T_\mathrm{b}^-[k]$ is considered, as this would only make a difference for the controller state, which is not weighted in Q.

6 EXTENSION TO STOCHASTIC TIMING

In this section, we generalize the previous results to stochastic timing. Therefore, we first address arbitrary stochastic timing in

Section 6.1, which is then specialized for piecewise-independent timing to enable a practical implementation in Section 6.2. To improve readability, the main proof is again detached to Section 9.

6.1 General Stochastic Timing

Under the restriction of deterministic timing, the previous sections give results for the covariance dynamics F, which stochastically describe the state evolution:

$$\underset{\mathcal{N}}{\mathcal{E}}\left\{x(t^+)x^T(t^+)\right\} = \underset{(0,t]}{F}\left(\underset{\mathcal{N}}{\mathcal{E}}\left\{x(0)x^T(0)\right\}, \mathcal{T}\right) \text{ for known } \mathcal{T} \quad (38)$$

Because noise $\mathcal{N}$ and timing $\mathcal{T}$ are independent, this can be generalized as the conditional expectation

$$\mathcal{E}\left\{x(t^+)x^T(t^+)|\mathcal{T}\right\} = \underset{(0,t]}{F}\left(\mathcal{E}\left\{x(0)x^T(0)|\mathcal{T}\right\}, \mathcal{T}\right) \quad (39)$$

also valid for uncertain timing, which is the starting point for the following derivation of a general result for nondeterministic timing.

For any function $f(\alpha, \beta)$ of random variables α, β it holds that

$$\mathcal{E}\{f(\alpha,\beta)\} = \underset{\alpha}{\mathcal{E}}\left\{\underset{\beta}{\mathcal{E}}\{f(\alpha,\beta)|\alpha\}\right\} \quad (40)$$

[21, Lemma 2.2]. The evolution of $x(t)$ depends on the noise $\mathcal{N}$ and the timing $\mathcal{T}$. Therefore, the covariance dynamics for stochastic timing can be split as

$$\mathcal{E}\left\{x(t^+)x^T(t^+)\right\} \overset{(40)}{=} \underset{\mathcal{T}}{\mathcal{E}}\left\{\underset{\mathcal{N}}{\mathcal{E}}\left\{x(t^+)x^T(t^+)|\mathcal{T}\right\}\right\} \quad (41)$$

$$\overset{(39)}{=} \underset{\mathcal{T}}{\mathcal{E}}\left\{\underset{(0,t]}{F}\left(\underset{\mathcal{N}}{\mathcal{E}}\left\{x(0^+)x^T(0^+)|\mathcal{T}\right\}, \mathcal{T}\right)\right\} \quad (42)$$

As the start state $x(0^+) = x(0)$ is certain, it holds that

$$\mathcal{E}\left\{x(0^+)x^T(0^+)|\mathcal{T}\right\} = \mathcal{E}\left\{x(0^+)x^T(0^+)\right\} = x_0x_0^T \quad (43)$$

and therefore

$$\mathcal{E}\left\{x(t^+)x^T(t^+)\right\} \overset{(42),(43)}{=} \underset{\mathcal{T}}{\mathcal{E}}\left\{\underset{(0,t]}{F}\left(x_0x_0^T, \mathcal{T}\right)\right\}, \quad (44)$$

which means that the covariance under timing uncertainty is the timing-probability-weighted average of the covariance for certain timing. Assuming discrete-valued timing, this allows for a first, naive implementation by individually computing the covariance for each possible timing sequence, as described in Section 5, and then taking the weighted average. Unfortunately, the number of timing sequences and therefore the complexity of this implementation grows exponentially with the length of the analyzed timespan.

6.2 Piecewise-Independent Stochastic Timing

To allow for a feasible implementation, we assume that the timing is independent across deterministically known *separation points* $t_{\text{s},k}$, i. e., the timing $\mathcal{T}_{(0,\, t_{\text{s},k}]}$ up to and including each separation point is independent of the timing $\mathcal{T}_{(t_{\text{s},k},\, t_{\text{s},k+1}]}$ after it.

Independent control periods are the most important case of piecewise-independent timing. Here, the timing barriers are separation points ($t_{\text{s},k} = T_{\text{b}}[k]$). This assumption is motivated by the observation that dependencies are typically confined to a single control period: For example, if two sensors cannot be sampled concurrently, their timing is dependent within one period. Consequently, we excluded cross-period dependencies in the first analysis step for performance reasons, since they are typically rare, indirect and therefore weaker.

The new timing barriers allow iterating the covariance from one separation point to the next, as will be derived in the following. We reintroduce the previous notation $P(t) := \mathcal{E}\{x(t)x^T(t)\}$, where the expectation is now also computed with regard to random timing.

$$P(t^+_{\text{s},k+1}) = \mathcal{E}\left\{x(t^+_{\text{s},k+1})x^T(t^+_{\text{s},k+1})\right\} \quad (45)$$

$$= \underset{\mathcal{T}}{\mathcal{E}}\left\{\underset{(0,\, t_{\text{s},k+1}]}{F}\left(x_0x_0^T, \mathcal{T}_{(0,\, t_{\text{s},k+1}]}\right)\right\}. \quad (46)$$

Inside the expectation, the timing is no longer random, but a determined variable of integration. Therefore, (35) may be applied to split the evolution at $t_{\text{s},k}$, even though it was only derived for deterministic timing:

$$P(t^+_{\text{s},k+1}) \overset{(35)}{=} \underset{\mathcal{T}_{(0,\, t_{\text{s},k}]}, \mathcal{T}_{(t_{\text{s},k},\, t_{\text{s},k+1}]}}{\mathcal{E}}\left\{\underset{(t_{\text{s},k},\, t_{\text{s},k+1}]}{F}\left(\underset{(0,\, t_{\text{s},k}]}{F}\left(x_0x_0^T, \mathcal{T}_{(0,\, t_{\text{s},k}]}\right), \mathcal{T}_{(t_{\text{s},k},\, t_{\text{s},k+1}]}\right)\right\}. \quad (47)$$

As the timings before and after the separation point, $\mathcal{T}_{(0,\, t_{\text{s},k}]}$ and $\mathcal{T}_{(t_{\text{s},k},\, t_{\text{s},k+1}]}$, are independent, we get

$$P(t^+_{\text{s},k+1}) = \underset{\mathcal{T}_{(t_{\text{s},k},\, t_{\text{s},k+1}]}}{\mathcal{E}}\left\{\underset{\mathcal{T}_{(0,\, t_{\text{s},k}]}}{\mathcal{E}}\left\{\underset{(t_{\text{s},k},\, t_{\text{s},k+1}]}{F}\left(\underset{(0,\, t_{\text{s},k}]}{F}\left(x_0x_0^T, \mathcal{T}_{(0,\, t_{\text{s},k}]}\right), \mathcal{T}_{(t_{\text{s},k},\, t_{\text{s},k+1}]}\right)\right\}\right\}. \quad (48)$$

Because $F(P, \mathcal{T})$ is a quadratic form of P, the expectation can be moved inside of F. The proof for this, which is given in Section 9.2, relies on the linearity of controller and plant.

$$P(t^+_{\text{s},k+1}) \overset{(67)}{=} \underset{\mathcal{T}_{(t_{\text{s},k},\, t_{\text{s},k+1}]}}{\mathcal{E}}\left\{\underset{(t_{\text{s},k},\, t_{\text{s},k+1}]}{F}\left(\underbrace{\underset{\mathcal{T}_{(0,\, t_{\text{s},k}]}}{\mathcal{E}}\left\{\underset{(0,\, t_{\text{s},k}]}{F}\left(x_0x_0^T, \mathcal{T}_{(0,\, t_{\text{s},k}]}\right)\right\}}_{P(t^+_{\text{s},k})\ (44)}, \mathcal{T}_{(t_{\text{s},k},\, t_{\text{s},k+1}]}\right)\right\} \quad (49)$$

The resulting iteration equation

$$P(t^+_{\text{s},k+1}) = \underset{\mathcal{T}_{(t_{\text{s},k},\, t_{\text{s},k+1}]}}{\mathcal{E}}\left\{\underset{(t_{\text{s},k},\, t_{\text{s},k+1}]}{F}\left(P(t^+_{\text{s},k}), \mathcal{T}_{(t_{\text{s},k},\, t_{\text{s},k+1}]}\right)\right\} \quad (50)$$

allows to compute the covariance and therefore the QoC iteratively with linear complexity regarding the time horizon and the number of timing possibilities within one separation period.

7 IMPLEMENTATION AND EXAMPLE

To verify the presented results, the stochastic QoC-model was implemented in MATLAB and compared to a classical Simulink simulation of the underlying system model from Section 4. The implementation and further examples are available at https://doi.org/10.5281/zenodo.1145674, for future versions see http://qronos.de/.

7.1 Simulation

For the simulation, a fixed-step solver with a small step size of $T_{\text{sim}} \ll T$ was used and accordingly the I/O timing was restricted to multiples of T_{sim}. Continuous-time white noise with $\mathcal{E}\{d(t)d(\tau)\} = \delta(t-\tau)$ was approximated by discrete-time white noise of variance $1/T_{\text{sim}}$ as per [16]. The model does not prescribe a noise probability density, so we chose it as Gaussian for the simulation.

One simulation run corresponds to one realization of the random process. To simulate the mean cost $\mathcal{E}\{J(T_{\text{b}}[k])\}$, which the QoC-model can directly compute, but a deterministic simulation cannot, M simulation runs with different pseudorandom noise and timing sequences were averaged. This was implemented by cyclically repeating the process and taking the cyclic average.

7.2 Example Setup

As a system, we chose the linearized inverse pendulum, a widely used benchmark for control systems. Its parameters

$$A_{\text{p}} = \begin{bmatrix} 0 & 1 \\ \omega_0^2 & -2\xi\omega_0 \end{bmatrix},\ B_{\text{p}} = \begin{bmatrix} 0 \\ \omega_0/9.81 \end{bmatrix},\ G_{\text{p}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ C_{\text{p}} = \begin{bmatrix} 1 & 0 \end{bmatrix},$$
$$H = 10^{-3},\ N_{\text{p}} = 10^{-6},\ \omega_0 = \pi,\ \xi = 0.5,\ T = 0.2 \tag{51}$$

are based on [2]. To keep the pendulum in the unstable upright position ($x_{\text{r}} = 0, u_{\text{r}} = 0$), an observer-based state feedback controller

$$\hat{x}[k+1] = A_{\text{p,d}}\hat{x}[k] + B_{\text{p,d}}u[k] + L(y[k] - Cx[k]) \tag{52}$$
$$u[k] = K\hat{x}[k] \tag{53}$$

is designed by pole placement for the discretized plant $(A_{\text{p,d}}, B_{\text{p,d}})$. The continuous-time equivalent poles are chosen as $\{-10, -11\}$ for the controller and $\{-20, -22\}$ for the observer, and discretized by $\lambda_{\text{d}} = e^{\lambda T}$. It will be seen that this naive choice of very fast poles compared with the plant poles of $\{1.94, -5.08\}$ is problematic, as the controller becomes especially sensitive to larger timing deviations.

The cost weighting factors $\tilde{Q} = 550 \cdot I$ and $\tilde{R} = 0.8$ were chosen such that x_{p} and u contribute an equal part of the total cost, which is 1 if no delays are present. The timing barrier was chosen as just before the nominal I/O point kT.

The system was subject to deterministic piecewise-constant delays in the range of $[0, T/2)$ for sampling and actuation, which could result from switching between execution modes with static timing.

It should be noted that, for the sake of clarity, this example only covers a limited set of features, but the results will nevertheless show important effects. The source package contains additional examples for stochastic timing, time-varying disturbance parameters, multiple inputs and outputs and reference trajectory tracking.

7.3 Results

The chosen sequence of delays and the resulting cost, which is the opposite of QoC, are shown in Figure 3. The plot underlines the importance of the dynamic aspect of the QoC: changing to worse execution conditions, such as increased actuator delay at $t = 10$, has a very limited immediate effect, but instead causes a gradual increase of cost, corresponding to a decrease of the QoC. Even after better timing has been restored ($t = 20$), aftereffects of the previous worse conditions still have to decay ($t = 20 \ldots 22$) before the QoC is as good as before. Additionally, the dynamics may be

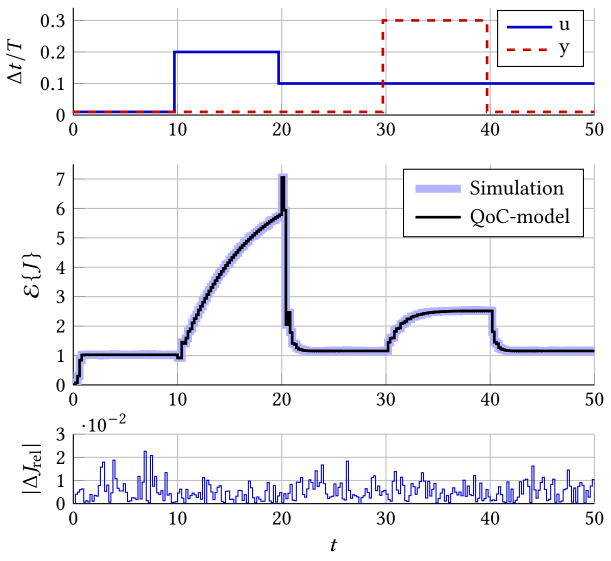

Figure 3: QoC evaluation for a controlled inverse pendulum for time-varying actuation and sampling delays: comparison between our model and a classical simulation.

counterintuitive in the sense that switching to better execution timing may incur a short-time negative effect ($t = 20$). A static approximation of QoC as a function of the timing, an assumption often underpinning embedded control systems design [7, 9, 11], fails to describe these memory-like behaviors, as it would instead predict a piecewise-constant QoC for the given timing. A similar result as for the actuator delay is obtained for varying sensor delays ($t = 30 \ldots 40$), although the impact on QoC is lower. This underlines that each I/O component has to be treated separately and models with only one delay parameter cannot be enough.

While the model was computed in less than one second, the classical simulation with $T_{\text{sim}} = 10^{-4}$ and $M = 3 \cdot 10^4$ runs took over five hours on an Intel Core i7-4790 processor, which exemplifies the computational efficiency gained by the stochastic evaluation. Simulation and model match within a relative deviation[1] of $|\Delta J_{\text{rel}}| < 0.03$, which can be attributed to the remaining randomness after averaging a finite number of simulation runs.

8 CONCLUSION AND OUTLOOK

Conclusion. Overall, our results confirm that the previous assumption of a simple, static relationship between timing and QoC is highly questionable. With our evaluation, we were able to demonstrate that the temporal progression and state history of the system (i. e., scheduling and control) can have a massive impact on the resulting Quality of Control. Moreover, our system model takes into account MIMO systems with heterogeneous sensor and actuator delays. It could, therefore, be used to extend existing approaches with more limited timing models, such as [1].

For the first time, we offer a systematic approach for evaluating the temporal progression of the QoC that accounts for both, varying environmental and execution conditions. We consider this a

[1]defined as $|\Delta_{\text{rel}}(a,b)| := |a-b|/\min(|a|,|b|)$ for $a, b \neq 0$, but also $\Delta_{\text{rel}}(0,0) := 0$.

vital step towards an accurate usage of QoC as an evaluation metric in highly dynamic and adaptive real-time settings, such as mixed-criticality scheduling, and as a basis for further research on co-design of control and real-time executive. Since our approach takes traditionally-designed control systems as input, it can be applied to existing systems for the retrospective evaluation of execution-condition impact. In future research, it could be used to tailor adaptive real-time approaches by systematically limiting their adaptivity so that they no longer jeopardize the overall QoC.

A further core aspect of our work is the stochastic evaluation of the QoC model, which fundamentally differs from and outperforms current brute-force simulation approaches. In this work, we have limited our model to linear systems, which significantly simplifies computation. While this is a restriction in principle, we consider it a compelling starting point for the validation of our approach and a good match for a broad range of practical applications.

Outlook. We are currently working on further reduction of the computational effort, with the aim of using the model at runtime for QoC-aware timing adaptation. In addition, we plan to generalize our research to nonlinear systems, albeit at the cost of highly increased computation time. This will allow addressing adaptive controllers, which are inherently nonlinear as they depend on amplitude.

In this work, we derived our final result under the assumption of piecewise-independent timing, which is appropriate for typical I/O jitter that has no cause outside of the current control period. However, a main limitation is that external timing disturbances with longer time horizons cannot be represented. To address systems where such dependencies become significant, future research will be concerned with extending the provided results towards Markov-chain timing models. This extension is expected to yield a time-varying generalization of [15], which would facilitate a direct comparison of both frameworks.

9 PROOFS

In this section, two key results used in Sections 5 and 6 will be derived: firstly, for the stochastic discretization, and secondly, for the expectation of affine matrix functions.

9.1 Stochastic discretization

9.1.1 Notes about the delta functional. Modeling the disturbance as white noise greatly simplifies the calculations, because it is uncorrelated at different time points. However, the infinite signal power of white noise makes the system description (12) a stochastic differential equation, which gives rise to unnecessary mathematical difficulties. As an engineer's approach, the delta functional $\delta(t)$ can be seen as shorthand for a short rectangular impulse of area 1,

$$\delta(t) = \begin{cases} h, & |t| < \frac{1}{2h} \\ 0, & \text{otherwise} \end{cases} \qquad \text{with } h \to \infty, \tag{54}$$

or any other function whose integral approaches

$$\int_{t_0}^{t_1} \delta(t)\,\mathrm{d}t \to \begin{cases} 1, & t_0 < 0 < t_1 \\ 0, & 0 \notin (t_0, t_1) \end{cases} \qquad \text{for } t_0 < t_1. \tag{55}$$

The limit $h \to \infty$ then has to be taken after inserting the definition (54) into the system equations, which roughly corresponds to exciting the system with bandwidth-limited noise of a bandwidth significantly larger than the closed-loop bandwidth.

The "sifting property" (55) implies the two-dimensional version

$$\int_{t_1}^{t_2}\int_{t_3}^{t_4} f(\alpha)\,g(\beta)\,\delta(\alpha-\beta)\,\mathrm{d}\alpha\,\mathrm{d}\beta = \int_{(t_1,t_2)\cap(t_3,t_4)} f(\tau)\,g(\tau)\,\mathrm{d}\tau \qquad \text{for } t_1 < t_2 \text{ and } t_3 < t_4. \tag{56}$$

9.1.2 Stochastic Discretization for Deterministic Timing. For the known start and end time t_0, t_1, the continuous system

$$\dot{x}(t) = Ax(t) + Gd(t) \tag{57}$$

has the general solution

$$x(t_1) = \mathrm{e}^{A(t_1-t_0)}x(t_0) + \int_{t_0}^{t} \mathrm{e}^{A(t_1-\tau)}Gd(\tau)\,\mathrm{d}\tau, \quad t_1 > t_0 \geq 0. \tag{58}$$

It is excited by independent white noise $d(t)$ with constant covariance within the time range:

$$\mathcal{E}\{d(t)\} = 0, \quad \mathcal{E}\Big\{d(t)d^T(t+\tau)\Big\} = H\delta(\tau) \quad \text{for } t_0 < t < t_1 \tag{59}$$

Taking the covariance of the general solution yields

$$\begin{aligned}\operatorname{cov}\{x(t_1)\} =& \mathcal{E}\Bigg\{\int_{t_0}^{t_1} \mathrm{e}^{A(t_1-\tau)}Gd(\tau)\,\mathrm{d}\tau \int_{t_0}^{t_1} d^T(\tilde{\tau})G^T(\mathrm{e}^{A(t_1-\tilde{\tau})})^T\,\mathrm{d}\tilde{\tau} \\ &+ \mathrm{e}^{A(t_1-t_0)}x(t_0)x^T(t_0)(\mathrm{e}^{A(t-t_0)})^T + M + M^T\Bigg\}\end{aligned} \tag{60}$$

where the mixed term

$$M = \underbrace{\int_{t_0}^{t_1} \mathrm{e}^{A(t_1-\tau)}Gd(\tau)\,\mathrm{d}\tau}_{M_1}\; x^T(t_0)(\mathrm{e}^{A(t_1-t_0)})^T \tag{61}$$

has zero expectation as we will show now by splitting off independent factors: $M_1 = x(t_1)|_{x(t_0)=0}$ is the random influence of noise $d(t)$ on the state $x(t_1)$. As $d(t)$ is white noise, this influence is independent of the initial state $x(t_0)$. For deterministic t_0, t_1 we get

$$\mathcal{E}\{M\} = \int_{t_0}^{t_1} \mathrm{e}^{A(t_1-\tau)}G\underbrace{\mathcal{E}\{d(\tau)\}}_{0\ (59)}\,\mathrm{d}\tau\;\mathcal{E}\Big\{x^T(t_0)\Big\}\Big(\mathrm{e}^{A(t_1-t_0)}\Big)^T = 0. \tag{62}$$

Continuing the calculations from (60), it follows that

$$\begin{aligned}\operatorname{cov}\{x(t_1)\} =& \mathrm{e}^{A(t_1-t_0)}\operatorname{cov}\{x(t_0)\}\,(\mathrm{e}^{A(t_1-t_0)})^T \\ &+ \int_{t_0}^{t_1}\int_{t_0}^{t_1} \mathrm{e}^{A(t_1-\tau)}G\,\underbrace{\mathcal{E}\Big\{d(\tau)d^T(\gamma)\Big\}}_{H\delta(\tau-\gamma)\ (59)} \\ &\cdot G^T(\mathrm{e}^{A(t_1-\gamma)})^T\,\mathrm{d}\tau\,\mathrm{d}\gamma.\end{aligned} \tag{63}$$

With $\Delta := t_1 - t_0$, the 2D sifting property (56) leads to

$$\begin{aligned}\operatorname{cov}\{x(t_1)\} =& \mathrm{e}^{A(t_1-t_0)}\operatorname{cov}\{x(t_0)\}\,(\mathrm{e}^{A(t_1-t_0)})^T \\ &+ \int_{t_0}^{t_1} \mathrm{e}^{A(t_1-\tau)}GHG^T(\mathrm{e}^{A(t_1-\tau)})^T\,\mathrm{d}\tau \end{aligned} \tag{64}$$

$$= \underbrace{\mathrm{e}^{A\Delta}}_{=:\Phi(\Delta)}\operatorname{cov}\{x(t_0)\}\underbrace{(\mathrm{e}^{A\Delta})^T}_{\Phi^T(\Delta)} + \underbrace{\int_0^{\Delta} \mathrm{e}^{A\tau}GHG^T(\mathrm{e}^{A\tau})^T\,\mathrm{d}\tau}_{=:\Psi(\Delta)} \tag{65}$$

$$= \Phi(\Delta)\operatorname{cov}\{x(t_0)\}\,\Phi^T(\Delta) + \Psi(\Delta). \tag{66}$$

This matches the result given in [21, pp. 86 ff.] for systems with constant sampling period Δ.

9.2 Expectation of an affine matrix function

Theorem 9.1. *Any matrix-valued function $F : \mathbb{R}^{n\times n}\times\mathcal{Y} \mapsto \mathbb{R}^{n\times n}$ that is affine in its first argument $X \in \mathbb{R}^{n\times n}$ and may possess an arbitrary second argument $y \in \mathcal{Y}$ fulfills*

$$\underset{X,y}{\mathcal{E}}\{F(g(X),y)\} = \underset{y}{\mathcal{E}}\left\{F\left(\underset{X}{\mathcal{E}}\{g(X)\}, y\right)\right\} \tag{67}$$

if X, y are independent random variables (or y is deterministic) and therefore

$$\Pr(X|y) = \Pr(X) \tag{68}$$

holds. Here, $g(X) : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ is an arbitrary function.

We define "affine" in this context that F can be written as

$$F(X,y) = \operatorname{unvec}(M(y)\operatorname{vec}X) + C(y) \tag{69}$$

with the matrices $M(y) \in \mathbb{R}^{n^2\times n^2}$ and $C(y) \in \mathbb{R}^{n\times n}$, which is a generalization of the scalar case $f(x) = mx + c$. Here,

$$\operatorname{vec}\underbrace{\begin{bmatrix}v_1 & \cdots & v_n\end{bmatrix}}_{V\in\mathbb{R}^{a\times b}} := \begin{bmatrix}v_1^T & \cdots & v_n^T\end{bmatrix}^T, \tag{70}$$

is the vectorization operator, which concatenates the column vectors v_i of V into one column vector. The corresponding inverse operator is defined by $\operatorname{unvec}(\operatorname{vec}(V)) := V$.

A particular case of (69) used in this paper is the quadratic form

$$F(X,y) = A^T(y)XA(y) + B(y), \quad A(y), B(y), X \in \mathbb{R}^{n\times n}. \tag{71}$$

This can be seen from the relation

$$\operatorname{vec}(A^T XA) = (A^T \otimes A^T)\operatorname{vec}X, \tag{72}$$

where $\otimes$ is the Kronecker product [4].

Proof. To prove (67) under the assumption (68), we split the expectation using (40) and then use the linearity of vec, unvec and the matrix multiplication, which permits

$$\operatorname{unvec}\left(M(y)\operatorname{vec}\left(\sum_i \alpha_i X_i\right)\right) = \sum_i \alpha_i \operatorname{unvec}(M(y)\operatorname{vec}X_i) \tag{73}$$

with the dimensions as stated above and $\alpha_i \in \mathbb{R}$.

$$\underset{X,y}{\mathcal{E}}\{F(g(X),y)\} \overset{(40)}{=} \underset{y}{\mathcal{E}}\left\{\underset{X}{\mathcal{E}}\{F(g(X),y)|y\}\right\} \tag{74}$$

$$\overset{(69)}{=} \underset{y}{\mathcal{E}}\left\{\underset{X}{\mathcal{E}}\{\operatorname{unvec}(M(y)\operatorname{vec}g(X)) + C(y)|y\}\right\} \tag{75}$$

$$= \underset{y}{\mathcal{E}}\left\{\sum_i \Pr(X = X_i|y)\cdot(\operatorname{unvec}(M(y)\operatorname{vec}g(X)) + C(y))\right\} \tag{76}$$

$$\overset{(68),(73)}{=} \underset{y}{\mathcal{E}}\left\{\operatorname{unvec}\left(M(y)\operatorname{vec}\left(\underbrace{\sum_i \Pr(X = X_i)\, g(X_i)}_{\mathcal{E}_X\{g(X)\}}\right)\right)\right.$$

$$\left. + \underbrace{\sum_i \Pr(X = X_i)}_{1}\, C(y)\right\} \tag{77}$$

$$\overset{(69)}{=} \underset{y}{\mathcal{E}}\left\{F\left(\underset{X}{\mathcal{E}}\{g(X)\}, y\right)\right\} \tag{78}$$

□

REFERENCES

[1] Mohammad Al Khatib, Antoine Girard, and Thao Dang. 2016. Verification and Synthesis of Timing Contracts for Embedded Controllers. In *Proc. of the 19th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC '16)*. ACM, New York, NY, USA, 115–124. https://doi.org/10.1145/2883817.2883827

[2] Karl-Erik Årzén, Bo Bernhardsson, et al. 1999. *Integrated control and scheduling.* Department of Automatic Control, Lund Institute of Technology. http://www.control.lth.se/documents/1999/arz+99t.pdf

[3] Guillem Bernat, Alan Burns, and Alberto Liamosi. 2001. Weakly hard real-time systems. *IEEE Trans. on Computers* 50, 4 (April 2001), 308–321. https://doi.org/10.1109/12.919277

[4] Dennis S. Bernstein. 2005. *Matrix Mathematics: Theory, Facts, and Formulas with Application to Linear Systems Theory.* Princeton University Press.

[5] Alan Burns and Robert Davis. 2016. *Mixed Criticality Systems – A Review.* Technical Report 9. Edition. Department of Computer Science, University of York. https://www-users.cs.york.ac.uk/burns/review.pdf

[6] Giorgio Buttazzo and Anton Cervin. 2007. Comparative Assessment and Evaluation of Jitter Control Methods. In *Proc. of the 15th Intl. Conf. on Real-Time and Network Systems (RTNS '07).* 163–172. https://hal.inria.fr/inria-00168530/document

[7] Giorgio Buttazzo, Manel Velasco, and Pau Marti. 2007. Quality-of-Control Management in Overloaded Real-Time Systems. *IEEE Trans. on Computers* 56, 2 (2007), 253–266. https://doi.org/10.1109/TC.2007.34

[8] Anton Cervin and Johan Eker. 2000. Feedback Scheduling of Control Tasks. In *Proc. of the 39th IEEE Conf. on Decision and Control (CDC '00)*, Vol. 5. IEEE Press, New York, NY, USA, 4871–4876. https://doi.org/10.1109/CDC.2001.914702

[9] Anton Cervin, Johan Eker, Bo Bernhardsson, and Karl-Erik Årzén. 2002. Feedback-Feedforward Scheduling of Control Tasks. *Real-Time Systems* 23, 1-2 (2002), 25–53. https://doi.org/10.1023/A:1015394302429

[10] Anton Cervin, Dan Henriksson, Bo Lincoln, Johan Eker, and Karl-Erik Årzén. 2003. How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime. *IEEE Control Systems Magazine* 23, 3 (2003), 16–30. https://doi.org/10.1109/MCS.2003.1200240

[11] Felicioni Flavia, Jia Ning, Françoise Simonot-Lion, and Song YeQiong. 2008. Optimal on-line (m,k)-firm constraint assignment for real-time control tasks based on plant state information. In *IEEE Intl. Conf. on Emerging Technologies and Factory Automation (ETFA '08).* IEEE, 908–915. https://doi.org/10.1109/ETFA.2008.4638504

[12] Daniele Fontanelli, Luca Greco, and Luigi Palopoli. 2013. Soft real-time scheduling for embedded control systems. *Automatica* 49, 8 (2013), 2330–2338. https://doi.org/10.1016/j.automatica.2013.04.036

[13] Tobias Klaus, Florian Franzmann, Maximilian Gaukler, Andreas Michalka, and Peter Ulbrich. 2016. Poster Abstract: Closing the Loop: Towards Control-aware Design of Adaptive Real-Time Systems. In *Proc. of the 37th Real-Time Systems Symp. (RTSS '16).* IEEE, 363–363. https://doi.org/10.1109/RTSS.2016.042

[14] Hermann Kopetz. 1997. *Real-Time Systems: Design Principles for Distributed Embedded Applications* (first ed.). Kluwer Academic Publishers.

[15] Bo Lincoln and Anton Cervin. 2002. JITTERBUG: a tool for analysis of real-time control performance. In *Proc. of the 41st IEEE Conf. on Decision and Control (CDC '02).* IEEE, 1319–1324. https://doi.org/10.1109/cdc.2002.1184698

[16] Keith Herbert Lloyd. 1982. *On the implementation of noise in the discrete simulation of continuous systems.* Weapons Systems Research Laboratory, Australia. http://www.dtic.mil/dtic/tr/fulltext/u2/a142055.pdf

[17] Charles Van Loan. 1978. Computing integrals involving the matrix exponential. *IEEE Trans. on Automatic Control* 23, 3 (Jun 1978), 395–404. https://doi.org/10.1109/TAC.1978.1101743

[18] Johan Nilsson, Bo Bernhardsson, and Björn Wittenmark. 1998. Stochastic analysis and control of real-time systems with random time delays. *Automatica* 34, 1 (1998), 57 – 64. https://doi.org/10.1016/S0005-1098(97)00170-2

[19] Asok Ray. 1994. Output Feedback Control Under Randomly Varying Distributed Delays. *Guidance, Control, and Dynamics* 17, 4 (1994), 701–711. https://doi.org/10.2514/3.21258

[20] Daniel Simon, Alexandre Seuret, and Olivier Sename. 2012. On real-time feedback control systems: Requirements, achievements and perspectives. In *Systems and Computer Science (ICSCS), 2012 1st Intl. Conf. on.* https://doi.org/10.1109/IConSCS.2012.6502458

[21] Torsten Söderström. 2002. *Discrete-Time Stochastic Systems* (second ed.). Springer.

[22] Steve Vestal. 1997. MetaH Support for Real-Time Multi-Processor Avionics. In *Proc. of 5th Intl. Work. on Parallel and Distributed Real-Time Systems and 3rd Work. on Object-Oriented Real-Time Systems.* IEEE, 11–21. https://doi.org/10.1109/WPDRTS.1997.637858

[23] Björn Wittenmark, Johan Nilsson, and Martin Törngren. 1995. Timing Problems in Real-time Control Systems. In *Proc. of the American Control Conf.* New York, NY, USA, 2000–2004. https://doi.org/10.1109/ACC.1995.531240