On Clairvoyant Control of Real-Time Control Systems subject to Deadline Overruns

Nils Vreman*, Anton Cervin*, Martina Maggio*†
* Department of Automatic Control, Lund University, Sweden
{name.surname}@control.lth.se
† Department of Computer Science Saarland University, Germany
{surname}@cs.uni-saarland.de

***Abstract*—Abstract here**

I. INTRODUCTION

A. Related Work

II. PROBLEM FORMULATION

In this section, we introduce the model of the control system and the timing issues the controller can experience. First, a brief introduction to the control system setup is provided, where we focus on how the physical components (i.e.,the plant, sensors, controller, and actuators) are interconnected. Second, we introduce the controller's computational model; in particular, the main timing issues that arise when the controller is implemented as a real-time task as well as how to address them.

A. Plant and Control Model

We consider all causal, linear time-invariant (LTI) plants $\mathcal{P}$. The plant is generally written on the following mathematical continuous-time state-space form:

$$\mathcal{P} := \begin{cases} \dot{x}(t) & = A\,x(t) + B\,u(t) + w(t) \\ y(t) & = C\,x(t) + v(t). \end{cases} \tag{1}$$

Here, $x(t)$ is the plant state, $u(t)$ the plant input, and $y(t)$ the plant output. The variables $w(t)$ and $v(t)$ represent respectively the process noise and the measurement noise. Both w and v are modelled as Gaussian white noise with known covariance matrices R_1^c and R_2^c. The plant is assumed to be both reachable and observable.

Since most modern feedback controllers are digital, the continuous measurement signal available from the plant needs to be sampled before used. Thus, the sensors sample the system periodically, with period T, described as follows:

$$y_k = Cx_k + v_k. \tag{2}$$

The notation with subscript k is used to represent the values sampled at time kT, e.g., $y_k = y(kT)$, and v_k is the discrete-time measurement noise with covariance $R_2 = {}^{R_2^c}\!/_{T}$.

To control the system, a periodic, discrete-time feedback controller $\mathcal{C}$ acting on the plant $\mathcal{P}$ is implemented. With the introduced notation the controller can generally be written on state-space form: ►**NV:** Add something about LET?◄

$$\mathcal{C} := \begin{cases} z_{k+1} & = F\,z_k + G\,y_k \\ u_k & = K\,z_k + H\,y_k \end{cases} \tag{3}$$

The design methodology used to formulate the controller varies; however, it is common to design a controller such that the following continuous, quadratic cost function is minimised:

$$J = \mathbb{E}\left[\int \begin{bmatrix} x^{\mathrm{T}}(\tau) \\ u^{\mathrm{T}}(\tau) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q_1^c(\tau) & Q_{12}^c(\tau) \\ (Q_{12}^c(\tau))^{\mathrm{T}} & Q_2^c(\tau) \end{bmatrix} \begin{bmatrix} x(\tau) \\ u(\tau) \end{bmatrix} d\tau \right]. \tag{4}$$

Here, Q_1^c is a (possibly) time-varying, positive semi-definite, weight matrix and Q_2^c is a (possibly) time-varying, positive definite, weight matrix. The control signal $u(t)$ is considered to be piece-wise constant between two successive updates of u_k. ►**NV:** Mention final cost Q_f◄

B. Control Task Model

Nominally, the controller is implemented in a real-time platform as a periodic task with period T. We will henceforth refer to the task executing the control algorithm as the *control task*. Every period of the control task, an instance is released and activated – called a *job* (denoted by j). We say that job j_k has an *activation time* a_k and a *completion time* f_k. Ideally, job j_k should complete before its corresponding *deadline* d_k, which is generally considered to occur when the next job activates, i.e., $f_k \leq d_k = a_{k+1} = a_k + T$.

For every job, the following steps are executed, chronologically:

1) the control command computed in the previous instance is updated at the actuator level;
2) the current plant measurement is sampled by the sensor and made available to the controller;
3) the controller updates its estimate of the current system state and predicts the system's state at the projected update instant; and
4) the new control command is computed and stored in memory.

The first two steps are copy processes that can be handled close to instantaneously in a real platform. For this reason, their execution time is assumed to be negligible, and treated as instant at the start of each job. On the other hand, the state estimation, prediction, and the control command cannot be ignored and represents the majority of the control task's execution time. The prediction step compensates for the input-output (IO) delay of the controller; here, the IO delay corresponds to one period of the control task if the job completes its execution within the next activation[1]. This implementation choice improves predictability and simplifies the design of the controlled system.

C. Job Overruns

In this paper, we are interested in studying those cases where the periodic controller may experience a deadline *overrun*, i.e., when $f_k > d_k$. The analysis and synthesis presented in this paper is independent of the origin of the overrun; however, common causes for control job overruns are: preemption from higher priority tasks in the real-time system **[R]**, missing sensor data **[R]**, cache misses **[R]**, and temporary CPU overloads **[R]**. If the overrun is left unhandled, *self-interference* conditions would happen, i.e., a late completion of a job may cause subsequent jobs to start executing later and cause cascade overruns. For this reason, handling the overruns properly is required from a scheduler perspective. In particular, two strategies are considered: (i) *killing* the job that did not complete before its corresponding deadline, immediately releasing the subsequent job, or (ii) letting the job continue its execution until completion but *skipping* the subsequent jobs. These strategies are properly defined here.

Definition 1 (Kill strategy). *Under the Kill strategy, a job j_k that does not complete before its corresponding*

[1]This is commonly known as *logical execution time* (LET).

deadline d_k is immediately terminated and the subsequent job j_{k+1} is activated. We assume that the internal states of the controller are preserved and that the Kill process has negligible overhead. Formally, for job j_k it holds that either $f_k \leq d_k$ or $f_k = \infty$.

Definition 2 (Skip-Next strategy)**.** *Under the Skip-Next strategy, a job j_k that overruns its corresponding deadline is allowed to continue executing until completion and all subsequent jobs that would be activated while j_k is still active are* skipped. *Formally, for job j_k it holds that if it overruns its corresponding deadline d_k, it is assigned a new deadline $d_k^+ = d_{k+1}$ and the next job's activation time is set to $a_{k+1} = d_k^+$.*

In addition to the choice of overrun handling strategy, how to handle the control signal is also an important design parameter. Historically, mainly two *actuator modes* have been considered (also adopted by this paper): (i) *Zero*ing the control signal, or (ii) *Hold*ing it. The choice of actuation mode (sometimes referred to as strategies, but to avoid notational confusion we will exclusively denote them by modes) is highly dependent on the system dynamics, as no mode generally dominates the other one [1]. For systems with unstable dynamics, integrators, or high variance measurement noise, the Zero mode ($u_k = 0$ if the k-th job overruns its deadline) may be a safer option to avoid rapid state deviations. On the other hand, for stable systems or controllers with integrator dynamics the Hold mode ($u_k = u_{k-1}$ if the k-th job overruns its deadline) can keep the current trajectory, under the presumption that a non-zero control signal is required to keep the plant near its operating point. It is important to note that this is not an absolute truth, rather a rule of thumb for how to approach the problem.

The combination of deadline handling strategy and actuation mode has been shown to significantly affect the control performance [2]**[R]**. To simplify notation we will denote an arbitrary combination of strategy and actuation mode with $\mathcal{S}$ (henceforth denoted the control task's *policy*). Additionally, the strategies (Kill and Skip-Next) will respectively be denoted by the symbols **`K`** and **`S`** and the actuation modes Zero and Hold will be referred to by **`Z`** and **`H`**; for instance, a control task subject to the Kill strategy and Zero actuation will adhere to the policy $\mathcal{S} =$ **`KZ`**.

III. Clairvoyant Control

Overruns radically change the dynamics of the controlled systems. In this section, we first describe the system dynamics taking into account both overruns and the control task's policy, $\mathcal{S}$. We then use this model to derive an optimal controller for the case when we have perfect knowledge about the overrun pattern.

A. System Model Using Valid Jobs

When a job j_k experiences an overrun, the control output is not available at the activation instant of the next job. Moreover, jobs that are either skipped or killed do not contribute to the system dynamics since they are not producing any control output. For this reason, the analysis can be restricted to the jobs that correctly complete their execution and produce a control output; these jobs are henceforth referred to as *valid jobs*. We analyze their ordered sequence, where if j_k is a valid job, j_{k+1} is the immediately next valid job (which is not necessarily the job that is activated at time $a_k + T$). Since the periodicity of the control task is not assured, we need to introduce some variables to help us describe the system behavior.

Definition 3 (Input-output delay σ_k)**.** *The input-output delay σ_k of a valid job j_k is the time interval between its sensing instant and the control output update (i.e., the activation instant of the next valid job).*

Definition 4 (Sample interval s_k)**.** *The sample interval s_k is the time interval between the activation of j_k and the activation of the next valid job j_{k+1}.*

Definition 5 (Hold interval h_k)**.** *The hold interval h_k is the time interval between the update of the control output computed by j_k and the one of of the next valid job.*

The triplet (σ_k, s_k, h_k) can be used univocally to extract the optimal controller for each valid job j_k. In fact, if perfect knowledge about the valid jobs is accessible, the optimal controller can be designed based on the following steps:

- update the current state estimate, using the most recent plant measurement and the system evolution from the past measurement (s_k);
- predict the state at the output instant using the input-output delay (σ_k); and
- compute the next control output knowing the hold interval up to the next successful update (h_k).

Graphical examples showing (σ_k, s_k, h_k) are presented in Figures 1–4 for the different policies $\mathcal{S} = \{\mathbf{KH}, \mathbf{SH}, \mathbf{KZ}, \mathbf{SZ}\}$. In the figures, the blue bars represent valid jobs (with corresponding activation times a_k), red bars are terminated jobs, and the parameters are depicted with interval arrows below the schedule traces. In general, the following relations

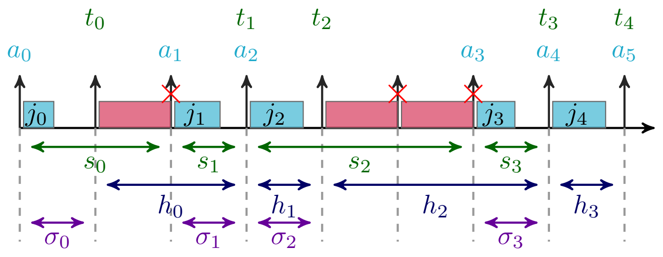

Fig. 1. $\mathcal{S} = \mathbf{KH}$

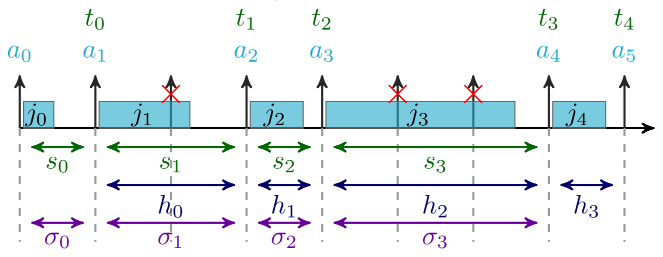

Fig. 2. $\mathcal{S} = \mathbf{SH}$

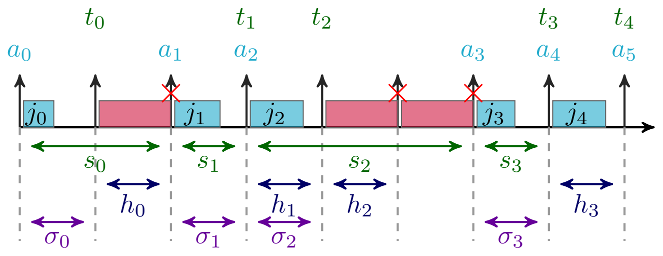

Fig. 3. $\mathcal{S} = \mathbf{KZ}$

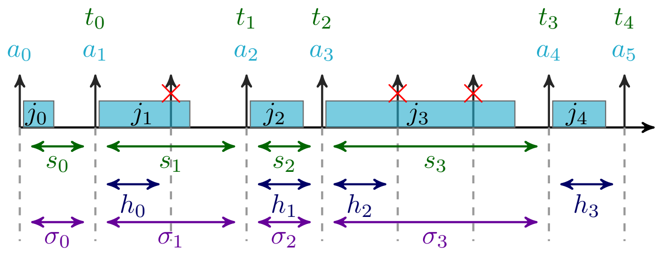

Fig. 4. $\mathcal{S} = \mathbf{SZ}$

hold:

$$\begin{aligned} s_k &= a_{k+1} - a_k \\ h_k &= \begin{cases} s_k, & \text{if } \mathbf{KH} \\ s_{k+1}, & \text{if } \mathbf{SH} \\ T, & \text{otherwise.} \end{cases} \\ \sigma_k &= \begin{cases} T, & \text{if } \mathbf{K} \\ s_k, & \text{if } \mathbf{S}. \end{cases} \end{aligned} \tag{5}$$

It is interesting to note that, in order to completely characterize j_k, information about the next valid job j_{k+1} is necessary – such as activation time a_{k+1} and completion time f_{k+1}.

This implies that a truly optimal controller needs to know the future to choose the right output.

B. Clairvoyant LQG Control

Assuming that we have no job overruns in the system, the optimal solution to Equation (4) is provided by a linear-quadratic-Gaussian (LQG) controller consisting of a Kalman filter and a linear-quadratic regulator (LQR). If the timing behavior of all jobs was completely known in advance, we would be able to design, by looking offline at the schedule, an optimal time-varying LQG controller that also minimizes the cost function (4). We call this our *clairvoyant* controller.

1) Kalman filter: ▶**NV:** To be written◀

Time-varying Kalman filter:

$$\begin{aligned}\hat{x}_{k|k} &= \hat{x}_{k|k-1} + L_k^f(y_k - C\hat{x}_{k|k-1})\\ \hat{x}_{k+1|k} &= \Phi(s_k)\hat{x}_{k|k} + \Gamma_0(s_k,\sigma_k)u_k + \Gamma_1(s_k,\sigma_k)u_{k-1}\\ &= \Phi(s_k)\hat{x}_{k|k-1} + \Gamma_0(s_k,\sigma_k)u_k + \Gamma_1(s_k,\sigma_k)u_{k-1} +\\ &+ L_k(y_k - C\hat{x}_{k|k-1})\end{aligned} \tag{6}$$

where (source: see KJ's book)

$$\begin{aligned}L_k^f &= P_{k|k-1}C^{\mathrm{T}}\left(CP_{k|k-1}C^{\mathrm{T}} + R_2\right)^{-1}\\ L_k &= \Phi(s_k)L_k^f\end{aligned} \tag{7}$$

and

$$\begin{aligned}P_{k+1|k} = \Phi(s_k)P_{k|k-1}\Phi^{\mathrm{T}}(s_k) + R_1(s_k) +\\ - L_k\left(CP_{k|k-1}C^{\mathrm{T}} + R_2\right)L_k^{\mathrm{T}}\end{aligned} \tag{8}$$

and

$$\begin{aligned}R_1(t_1) &= \int_0^{t_1} e^{As}R_1^c e^{A^{\mathrm{T}}s}ds\\ R_2 &= \frac{R_2^c}{T}\end{aligned} \tag{9}$$

2) LQR: To minimise the *continuous* cost function (4) using a *discrete* controller, we design an LQR controller that instead minimises the continuous cost function's discrete equivalent:

$$J = \mathbb{E}\left[\sum_{k=0}^{\infty} x_k^{\mathrm{T}}Q_1x_k + 2x_k^{\mathrm{T}}Q_{12}u_k + u_k^{\mathrm{T}}Q_2u_k\right] \tag{10}$$

Here, Q_1, Q_2, and Q_{12} are the discrete equivalents of Q_1^c, Q_2^c, and Q_{12}^c. ▶**NV:** Mention final cost Q_f◀

$$x_{k+1} = \Phi(s_k)x_k + \Gamma_0(s_k,\sigma_k)u_k + \Gamma_1(s_k,\sigma_k)u_{k-1} \tag{11}$$

where

$$\begin{aligned}\Phi(t_1) &= e^{At_1}\\ \Gamma_0(t_1,t_2) &= \int_0^{t_1-t_2} e^{As}ds\,B\\ \Gamma_1(t_1,t_2) &= e^{A(t_1-t_2)}\int_0^{t_2} e^{As}ds\,B\\ \Gamma(t_1,t_2) &= e^{A(t_1-t_2)}\int_0^{t_2} e^{As}ds\,B\end{aligned} \tag{12}$$

▶**NV:** Note that $\Gamma(t_1,t_2)$ can be used to describe both how the "old" and "new" control signals affect the next state in a system with a time delay. In particular, if $t_1 = t_2$ it implies that $\Gamma(t_1,t_1)$ represent the effect of the "new" control signal on the state evolution and similarly if $t_2 = 0$. ◀

The optimal control signal to be applied in the hold interval h_k is given by

$$u(t_k) = -K_kx(t_k) \;\Leftrightarrow\; u_k = -K_kx_k, \tag{13}$$

where the sequence of feedback gain matrices $\{K_k\}$ are obtained as the solution to a discrete-time Riccati equation (DTRE). The feedback matrices can be calculated off-line and stored in a table for on-line use.

The control law (13) cannot be implemented as it stands, however, since the actuated control action is computed based on a state measurement that is σ_k time units old. Hence, to compensate for the IO delay, we add a prediction term to the control computation

$$\hat{x}_k^p = \Phi(\sigma_k)\hat{x}_{k|k} + \Gamma_0(\sigma_k,0)u_{k-1}. \tag{14}$$

Replacing x_k in Equation (13) with $\hat{x}_k^p$ provides us with the LET adopting control law

$$u_k = -K_k\Phi(\sigma_k)\hat{x}_{k|k} - K_k\Gamma_0(\sigma_k,0)u_{k-1} \tag{15}$$

Obtaining the LQR gains K_k involve solving a discrete-time Riccati equation. Normally, this is a simple problem to solve; however, the presence of both deadline misses and different actuation modes complicate the solution. We therefore present the solution separately for the two actuation modes, i.e., Hold and Zero.

▶**NV: SUPER IMPORTANT NOTE:** I became a bit confused/worried that this is incorrect now since we use the innovation term ($L_k^f y_k$) in the control signal u_k, but then it is not proper LET? What I'm trying to say: Do we calculate the correct gain K_k for each interval since we actuate u_k based on current data (combined with the JitterTime simulations)?◀

HOLD: The feedback gain is given by a Riccati equation: ▶**NV:** NOTE that $\Phi(x)$ should be in between actuation instants but $\Gamma(x,y)$ should be affected by actuation mode as well. Also note that for the *hold* case $\Delta_k = h_k$, $\forall k$.

$$\Delta_k = t_{k+1} - t_k$$

◀

$$\begin{aligned} S_k &= \Phi^{\mathrm{T}}(\Delta_k) S_{k+1} \Phi(\Delta_k) + Q_1(\Delta_k) \\ &\quad - \left(\Phi^{\mathrm{T}}(\Delta_k) S_{k+1} \Gamma(\Delta_k, h_k) + Q_{12}(\Delta_k, h_k)\right) K_k \\ K_k &= \left(Q_2(\Delta_k, h_k) + \Gamma(\Delta_k, h_k)^{\mathrm{T}} S_{k+1} \Gamma(\Delta_k, h_k)\right)^{-1} \\ &\quad \times \left(\Gamma(\Delta_k, h_k)^{\mathrm{T}} S_{k+1} \Phi(\Delta_k) + Q_{12}^{\mathrm{T}}(\Delta_k, h_k)\right) \end{aligned} \tag{16}$$

where $Q_1(t_1)$, $Q_2(t_1,t_2)$, and $Q_{12}(t_1,t_2)$ are calculated according to

$$\begin{aligned} Q_1(t_1) &= \int_0^{t_1} \Phi^{\mathrm{T}}(s) Q_1^c \Phi(s) ds \\ Q_2(t_1,t_2) &= \int_0^{t_1} \left[\Gamma^{\mathrm{T}}(s,s) Q_1^c \Gamma(s,s) + 2\Gamma^{\mathrm{T}}(s,s) Q_{12}^c + Q_2^c\right] ds \\ Q_{12}(t_1,t_2) &= \int_0^{t_1} \Phi^{\mathrm{T}}(s) \left[Q_1^c \Gamma(s,s) + Q_{12}^c\right] ds \end{aligned} \tag{17}$$

▶**NV:** The definitions of $Q_2(t_1,t_2)$ and $Q_{12}(t_1,t_2)$ are novel for the zero case◀

Summary:

$$\begin{cases} (1)\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + L_k^f (y_k - C\hat{x}_{k|k-1}) \\ (2)\ \hat{x}_{k+1|k} &= \Phi(s_k)\hat{x}_{k|k-1} + \Gamma_0(s_k,\sigma_k)u_k + \\ &\quad +\Gamma_1(s_k,\sigma_k)u_{k-1} + L_k(y_k - C\hat{x}_{k|k-1}) \\ (3)\ \hat{x}_k^p &= \Phi(\sigma_k)\hat{x}_{k|k} + \Gamma_0(\sigma_k,0)u_{k-1} \\ (4)\ u_k &= -K_k \hat{x}_k^p \end{cases} \tag{18}$$

Combining (1), (3), and (4) yields

$$\begin{aligned} u_k &= -K_k \Phi(\sigma_k)\left(I - L_k^f C\right)\hat{x}_{k|k-1} - \\ &\quad - K_k \Gamma_0(\sigma_k, 0) u_{k-1} - \\ &\quad - K_k \Phi(\sigma_k) L_k^f y_k. \end{aligned} \tag{19}$$

Finally, combining the expression for u_k with (2) yields

$$\begin{aligned} \hat{x}_{k+1|k} &= \left[\Phi(s_k) - L_k C - \Gamma_0(s_k,\sigma_k) K_k \Phi(\sigma_k)\left(I - L_k^f C\right)\right]\hat{x}_{k|k-1} + \\ &\quad + \left[\Gamma_1(s_k,\sigma_k) - \Gamma_0(s_k,\sigma_k) K_k \Gamma_0(\sigma_k,0)\right] u_{k-1} - \\ &\quad + \left[L_k - \Gamma_0(s_k,\sigma_k) K_k \Phi(\sigma_k) L_k^f\right] y_k. \end{aligned} \tag{20}$$

ZERO: The feedback gain is given by a Riccati equation: ▶**NV:** NOTE that $\Phi(x)$ should be in between actuation instants but $\Gamma(x,y)$ should be affected by actuation mode as well. Also note that for the *zero* case $\Delta_k \neq h_k$, $\forall k$.

$$\Delta_k = t_{k+1} - t_k$$

◀

$$\begin{aligned} S_k &= \Phi^{\mathrm{T}}(\Delta_k) S_{k+1} \Phi(\Delta_k) + Q_1(\Delta_k) \\ &\quad - \left(\Phi^{\mathrm{T}}(\Delta_k) S_{k+1} \Gamma(\Delta_k, h_k) + Q_{12}(\Delta_k, h_k)\right) K_k \\ K_k &= \left(Q_2(\Delta_k, h_k) + \Gamma(\Delta_k, h_k)^{\mathrm{T}} S_{k+1} \Gamma(\Delta_k, h_k)\right)^{-1} \\ &\quad \times \left(\Gamma(\Delta_k, h_k)^{\mathrm{T}} S_{k+1} \Phi(\Delta_k) + Q_{12}^{\mathrm{T}}(\Delta_k, h_k)\right) \end{aligned} \tag{21}$$

where $Q_1(t_1)$, $Q_2(t_1,t_2)$, and $Q_{12}(t_1,t_2)$ are calculated according to

$$\begin{aligned} Q_1(t_1) &= \int_0^{t_1} \Phi^{\mathrm{T}}(s) Q_1^c \Phi(s) ds \\ Q_2(t_1,t_2) &= \int_0^{t_2} \left[\Gamma^{\mathrm{T}}(s,s) Q_1^c \Gamma(s,s) + 2\Gamma^{\mathrm{T}}(s,s) Q_{12}^c + Q_2^c\right] ds \\ &\quad + \int_{t_2}^{t_1} \left[\Gamma^{\mathrm{T}}(t_1,s) Q_1^c \Gamma(t_1,s) + 2\Gamma^{\mathrm{T}}(t_1,s) Q_{12}^c + Q_2^c\right] ds \\ Q_{12}(t_1,t_2) &= \int_0^{t_2} \Phi^{\mathrm{T}}(s) \left[Q_1^c \Gamma(s,s) + Q_{12}^c\right] ds \\ &\quad + \int_{t_2}^{t_1} \Phi^{\mathrm{T}}(s) \left[Q_1^c \Gamma(t_1,s) + Q_{12}^c\right] ds \end{aligned} \tag{22}$$

Not sure if the above equations are correct or if it should be

$$\begin{aligned} Q_2(t_1,t_2) &= \int_0^{t_1} \left[\Gamma^{\mathrm{T}}(t_1,s) Q_1^c \Gamma(t_1,s) + 2\Gamma^{\mathrm{T}}(t_1,s) Q_{12}^c + Q_2^c\right] ds \\ Q_{12}(t_1,t_2) &= \int_0^{t_1} \Phi^{\mathrm{T}}(s) \left[Q_1^c \Gamma(t_1,s) + Q_{12}^c\right] ds \end{aligned} \tag{23}$$

▶**NV:** The definitions of $Q_2(t_1,t_2)$ and $Q_{12}(t_1,t_2)$ are novel for the zero case◀

Summary:

$$\begin{cases} (1)\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + L_k^f (y_k - C\hat{x}_{k|k-1}) \\ (2)\ \hat{x}_{k+1|k} &= \Phi(s_k)\hat{x}_{k|k-1} + \Gamma_0(s_k,\sigma_k)u_k + \\ &\quad +\Gamma_1(s_k,\sigma_k)u_{k-1} + L_k(y_k - C\hat{x}_{k|k-1}) \\ (3)\ \hat{x}_k^p &= \Phi(\sigma_k)\hat{x}_{k|k} + \Gamma_0(\sigma_k,0)u_{k-1} \\ (4)\ u_k &= -K_k \hat{x}_k^p \end{cases} \tag{24}$$

Combining (1), (3), and (4) yields

$$\begin{aligned} u_k &= -K_k \Phi(\sigma_k)\left(I - L_k^f C\right)\hat{x}_{k|k-1} - \\ &\quad - K_k \Gamma_0(\sigma_k, 0) u_{k-1} - \\ &\quad - K_k \Phi(\sigma_k) L_k^f y_k. \end{aligned} \tag{25}$$

Finally, combining the expression for u_k with (2) yields

$$\begin{aligned} \hat{x}_{k+1|k} &= \left[\Phi(s_k) - L_k C - \Gamma_0(s_k,\sigma_k) K_k \Phi(\sigma_k)\left(I - L_k^f C\right)\right]\hat{x}_{k|k-1} + \\ &\quad + \left[\Gamma_1(s_k,\sigma_k) - \Gamma_0(s_k,\sigma_k) K_k \Gamma_0(\sigma_k,0)\right] u_{k-1} - \\ &\quad + \left[L_k - \Gamma_0(s_k,\sigma_k) K_k \Phi(\sigma_k) L_k^f\right] y_k. \end{aligned} \tag{26}$$

DISCRETISED COST MATRICES IN THE ZERO CASE: Recall that:

$$\begin{aligned} \Phi(t_1) &= e^{At_1} \\ \Gamma(t_1,t_2) &= e^{A(t_1-t_2)}\int_0^{t_2} e^{As}ds\,B \\ \Delta_k &= t_{k+1}-t_k \\ h_k &= \begin{cases} \Delta_k, & \text{if } \mathtt{H} \\ T, & \text{if } \mathtt{Z} \end{cases} \end{aligned} \tag{27}$$

►NV: NOTE that the equations above hold for both the Hold, i.e., $h_k = \Delta_k$, and Zero cases, i.e., $h_k = T$.◄

Then it follows that (for Zero actuation):

$$\begin{aligned} x_{k+1} &= \Phi(h_k)x_k + \Gamma(h_k,h_k)u_k \\ x(t_{k+1}) &= \Phi(\Delta_k - h_k)x_{k+1} \\ u(t) &= \begin{cases} u_k, & \text{if } t \le h_k \\ 0, & \text{otherwise} \end{cases} \\ \implies x(t) &= \begin{cases} \Phi(t)x_k + \Gamma(t,t)u_k, & \text{if } t \le h_k \\ \Phi(t)x_k + \Gamma(t,h_k)u_k, & \text{otherwise} \end{cases} \end{aligned} \tag{28}$$

Note that

$$\Phi(t-h_k)x_{k+1} = \Phi(t)\,x_k + \Gamma(t,h_k)\,u_k,\ t > h_k$$

. Thus:

$$\begin{aligned} J_k &= \int_0^{\Delta_k} x^{\mathrm{T}}(t)Q_1^c x(t) + 2x^{\mathrm{T}}(t)Q_{12}^c u(t) + u^{\mathrm{T}}(t)Q_2^c u(t)\,dt \\ &= \int_0^{\Delta_k} x^{\mathrm{T}}(t)Q_1^c x(t)\,dt + \underbrace{\int_0^{h_k} 2x^{\mathrm{T}}(t)Q_{12}^c u(t) + u^{\mathrm{T}}(t)Q_2^c u(t)\,dt}_{u(t>h_k)=0} \\ &= \int_0^{\Delta_k} \left[x_k^{\mathrm{T}}\Phi^{\mathrm{T}}(t) + u_k^{\mathrm{T}}\Gamma^{\mathrm{T}}(\square,\square)\right] Q_1^c \left[\Phi(t)x_k + \Gamma(\square,\square)u_k\right] dt + \\ &+ \int_0^{h_k} 2\left[x_k^{\mathrm{T}}\Phi^{\mathrm{T}}(t) + u_k^{\mathrm{T}}\Gamma^{\mathrm{T}}(t,t)\right] Q_{12}^c u_k + u_k^{\mathrm{T}}Q_2^c u_k\,dt \end{aligned}$$

Split first integral in two cases

$$\begin{aligned} J_k &= \int_{h_k}^{\Delta_k} \left[x_k^{\mathrm{T}}\Phi^{\mathrm{T}}(t) + u_k^{\mathrm{T}}\Gamma^{\mathrm{T}}(t,h_k)\right] Q_1^c \left[\Phi(t)x_k + \Gamma(t,h_k)u_k\right] dt + \\ &+ \int_0^{h_k} \left[x_k^{\mathrm{T}}\Phi^{\mathrm{T}}(t) + u_k^{\mathrm{T}}\Gamma^{\mathrm{T}}(t,t)\right] Q_1^c \left[\Phi(t)x_k + \Gamma(t,t)u_k\right] dt + \\ &+ \int_0^{h_k} 2\left[x_k^{\mathrm{T}}\Phi^{\mathrm{T}}(t) + u_k^{\mathrm{T}}\Gamma^{\mathrm{T}}(t,t)\right] Q_{12}^c u_k + u_k^{\mathrm{T}}Q_2^c u_k\,dt \end{aligned}$$

Combine for the same parameters:

$$\begin{aligned} J_k &= x_k^{\mathrm{T}} \underbrace{\int_0^{\Delta_k} \Phi^{\mathrm{T}}(t)Q_1^c\Phi(t)\,dt}_{Q_1(\Delta_k)}\, x_k + \\ &+ 2x_k^{\mathrm{T}} \underbrace{\left[\int_0^{h_k} \Phi^{\mathrm{T}}(t)\left(Q_1^c\Gamma(t,t) + Q_{12}^c\right)dt + \int_{h_k}^{\Delta_k} \Phi^{\mathrm{T}}(t)Q_1^c\Gamma(t,h_k)\,dt\right]}_{Q_{12}(\Delta_k,h_k)} u_k \\ &+ u_k^{\mathrm{T}} \underbrace{\left[\int_0^{h_k} \left[\Gamma^{\mathrm{T}}(t,t)Q_1^c\Gamma(t,t) + 2\Gamma^{\mathrm{T}}(t,t)Q_{12}^c + Q_2^c\right] dt + \int_{h_k}^{\Delta_k} \left[\Gamma^{\mathrm{T}}(t,h_k)Q_1^c\Gamma(t,h_k)\right] dt\right]}_{Q_2(\Delta_k,h_k)} u_k \end{aligned}$$

IV. CONCLUSIONS

REFERENCES

[1] L. Schenato. To zero or to hold control inputs with lossy links? *IEEE Transactions on Automatic Control*, 54(5):1093–1099, 2009.

[2] N. Vreman, A. Cervin, and M. Maggio. Stability and Performance Analysis of Control Systems Subject to Bursts of Deadline Misses. In B. B. Brandenburg, editor, *33rd Euromicro Conference on Real-Time Systems (ECRTS 2021)*, volume 196 of *Leibniz International Proceedings in Informatics (LIPIcs)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.