Timing-Robust Control over the Cloud Using On-Line Parametric Optimization

Nyberg Carlsson, Max; Vreman, Nils; Cervin, Anton

Published in:
IFAC Proceedings Volumes (IFAC-PapersOnline)

DOI:
10.1016/j.ifacol.2023.10.457

2023

Timing-Robust Control over the Cloud Using On-Line Parametric Optimization ⋆

Max Nyberg Carlsson * **Nils Vreman** * **Anton Cervin** *

* *Department of Automatic Control, Lund University, Sweden*

Abstract: In this paper, we present a heuristic method for adapting a networked linear feedback controller to improve its robustness to timing complications, such as long delays, aborted computations, and dropped packets. The core concept of the approach is to log successful sampling and actuation events and then, at discrete time-points, use non-convex parametric optimization to improve the expected performance of the controller under the assumption that the future timing behavior will be similar to the current one. To reduce the time complexity of the optimization algorithm, automatic differentiation is integrated for efficient gradient descent. The approach is evaluated on a physical ball and beam plant, where both the controller and optimization algorithm can be located in the Cloud.

Keywords: Fault-tolerant, Data-driven robust control, Adaptive control

1. INTRODUCTION

Offloading feedback controllers to the Cloud is predicted to be a strong trend in the future (Abdelzaher et al., 2020). In the Cloud, more advanced control algorithms can utilize the extensive resources that are available in order to (among many others) compute global objectives for vehicle platoons, safe operations for swarms of drones, and global control of distributed nodes in a power grid. However, control over the Cloud also entails several challenges. Both the Cloud and the communication channels between the Cloud and the clients are powerful yet unreliable resources. To address these uncertainties, a Cloud-based controller must be robust against random changes in the computation and communication layers.

It is well known that extensive and time-varying delays in the control computation as well as communication channels may lead to degraded control performance, and in certain cases even instability (Cervin et al., 2003; Vreman et al., 2021). Despite this, robust control synthesis methods have typically relied on fault models with particular structure and known parameters, e.g., Markov chains, delay distributions, or packet loss probabilities. If these models are inaccurate for the specific system implementation, the stability and performance guarantees provided by the control design methods are void.

We instead propose an adaptive controller that is robust to multiple timing uncertainties, e.g., computational jitter, time-varying delays, and packet losses. The adaptive controller consists of three parts: (i) a *control law* that executes periodically, (ii) a *logger* that records the incoming packets, and (iii) an *optimizer*.

The controller adaptation is a stepping stone to mitigating the effects of communication delays between, e.g., self-driving autonomous vehicles and the cloud services hosting their corresponding digital twins. In such a setup, transmission delays are difficult to predict due to, for instance, network traffic, the vehicle's distance to a radio station, and transmission interference (among many other factors). With the adaptive controller proposed in this paper, we aim to robustify the control laws' implementation in remote cloud services, where network delays and packet drops are difficult to predict or estimate.

1.1 Related Work

The idea to design robust controllers via parametric optimization has been around for a long time (Ackermann, 1980). It is well known that many controller synthesis problems—including static output feedback and low-order dynamic feedback designs—lead to nonlinear optimization problems that can only be solved using numerical methods (Syrmos et al., 1997; Grimble and Johnson, 1999).

Optimization-based solutions have been proposed for a variety of non-standard LQR and LQG design problems. In Bernstein et al. (1986), an iterative projection algorithm is used to produce low-order sampled-data LQG controllers for systems with delays. A direct optimization approach is taken in Mäkilä and Norlander (2000), which produces optimal controllers for random delays.

The practical applicability of Cloud-based controllers were demonstrated in Skarin et al. (2018), where the authors show that running a controller in the 5G edge could improve the control performance in comparison to running it either locally or in a remote data center.

⋆ The authors are members of the ELLIIT Strategic Research Area at Lund University. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Number 871259 (ADMORPH project). This publication reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains. Corresponding author: max.nyberg_carlsson@control.lth.se

1.2 Contributions

Our objective is to explore how safety-critical control can be improved by exploring how state-of-the-art approaches within parametric optimization for LQG can be combined with Cloud-based control architectures. We propose a data-driven approach to compensate for delays and lost packets in networked control systems, utilizing the extensive resources that the Cloud provides. Compared to standard methods for designing LQG controllers for systems with delay, previous control signals are not stored as additional states. Instead, the approach adapts the controller parameters on-line based on a backlog of sensor packets and their corresponding timestamps. This makes the method applicable to large variety of scenarios, including constant, periodic, sporadic, or random patterns of timing and connectivity problems.

1.3 Outline

The remainder of this paper is structured as follows. In Section 2 we outline the problem we are trying to solve and the assumptions put on the system setup. How the cost function of the leveraged toolbox `JitterTime` is calculated is described in Section 3. Section 4 describes the closed-loop system setup in more detail with focus on the controller adaptation in the Cloud. In Section 5, we present an empirical evaluation on a physical ball and beam process, which is controlled and robustified using the proposed method. Finally, Section 6 concludes the paper and discusses future research directions.

2. PROBLEM FORMULATION

In this paper, we address the timing problems that can occur when a control algorithm is executed in the Cloud. Of particular interest are time delays and packet losses that are induced by unreliable network communication.

Our objective is to improve the control robustness to *both* transmission delays over the network (i.e., latency) and packet losses due to unreliable communication channels. We want to avoid both imposing too restrictive constraints on the controller and altering the controller structure, such as introducing additional complexity or increasing its order. Finally, we set out to adapt the controller on-line without affecting the control performance or robustness negatively.

2.1 Assumptions

We consider a situation where a plant should be remotely controlled due to, for instance, limited local computing resources. Sampling and actuation are performed locally at the *client* side, while the control computations and parametric optimizations are carried out on a *server* in the Cloud. In the paper, we assume that the networked control loop operates in the following way:

(1) Measurement samples are taken periodically by the sensors on the client side, timestamped with the current time, and transmitted to the server.
(2) *If* and *when* a sample packet reaches the server,
    (a) a control computation is scheduled, and
    (b) the packet (including timestamp) is logged.
(3) *If* and *when* a control computation finishes, the response (i.e., the control signal) is transmitted back to the actuators on the client side.
(4) *If* and *when* a control packet reaches the client,
    (a) the actuator outputs the control signal to the plant, and
    (b) the client timestamps the control signal and sends it back to the server to be logged.

Limiting the scope of this work, the connection between server and actuator is assumed ideal. Furthermore, we assume that the messages sent over the network are timestamped using standard algorithms created for use in embedded systems. The additional overhead that is introduced by these algorithms is assumed to be negligible in relation to the round-trip and computation time. We assume that no additional overhead exists (e.g., from encrypting the network packets). Finally, we assume that there exists no a priori knowledge about packet loss probabilities or server response times; but the behavior is assumed to change slowly such that the adaptation to the network and computational conditions is feasible. We acknowledge that the assumptions are coarse; however, we see this work as a stepping stone to analysing more complex systems in the future.

2.2 Control Model and Objective

Given a process model with known parameters, a fixed-structure controller, a cost function, and a set of sampling and actuation timestamps collected over a time window, the objective is to minimize the expected future cost of the control loop. In the optimization, a major assumption is that the timing behavior in the near future will be similar to the recently logged behavior. To facilitate an efficient cost evaluation and optimization under variable timing patterns, we will assume a linear process with Gaussian disturbances, a linear controller, and a quadratic cost function.

3. COST FUNCTION EVALUATION

To leverage the power of parametric optimization, we need to be able to compute the expected future cost as a function of the feedback controller. For this purpose, we utilize the `JitterTime` toolbox, see Cervin (2019), which facilitates the evaluation of a quadratic cost function for a sampled-data linear stochastic system under arbitrary timing patterns. The toolbox allows for a number of continuous- and discrete-time systems driven by Gaussian white noise to be connected in a control loop. The cost is then calculated in a simulation phase, where the discrete systems can be executed in any order at given points in time. This makes it possible to evaluate the impact on performance of delays, jitter, lost samples, aborted computations, etc., in a given scenario.

The `JitterTime` model setup used for cost evaluation in this paper is shown in Figure 1. The linear continuous process $P(s)$ with control input $u(t)$ is disturbed by continuous-time white process noise $w(t)$ with intensity R_1, and its measurement signal $y(t)$ is corrupted by discrete-time Gaussian white noise $v(t)$ with variance

R_2. The discrete sampler $S(z) = I$ executes at sampling events and holds the current and old measurement values $y_k, \ldots, y_{k-d}$ between updates. The linear discrete controller $C(z)$ executes at actuation events and can use any of the current or old measurement samples as inputs to model an arbitrary real-valued input–output delay from 0 up to $d + 1$ sampling periods.

The performance of the control loop is measured by the continuous-time cost function

$$f_\tau = \mathrm{E} \int_\tau \left(x^\top(t) Q_1 x(t) + u^\top(t) Q_2 u(t) \right) \mathrm{d}t, \qquad (1)$$

where E denotes expected value, τ is the evaluation horizon, x is the process state vector, and Q_1 and Q_2 are positive definite weighting matrices. For regular sampling and actuation times, the cost calculation would be trivial and given by well-known formulas from sampled-data LQG theory (Åström and Wittenmark, 1997). Under irregular timing events, however, each interval between events must be considered separately.

Internally, the toolbox keeps track of the combined covariance of the continuous and discrete states of the model at all points in time. At the discrete events, either the sampler S or the controller C updates the covariance of their states and outputs according to discrete linear stochastic dynamics. Consider the discrete system

$$x(k+1) = \Phi_k x(k) + v(k) \qquad (2)$$

where $v(k)$ is zero-mean white-noise process with covariance R. When triggered, the covariance $P(k)$ is updated according to

$$P(k+1) = \Phi_k P(k) \Phi_k^\top + R. \qquad (3)$$

During the intervals between discrete events, the process P accumulates covariance and expected cost according to linear stochastic dynamics. Using the interval length as sampling time, these quantities are updated according to (3) and (1). The computational burden of a cost calculation is dominated by the matrix exponentials required for the sampling, and the total evaluation time grows linearly with the number of events during the evaluation horizon. For a more detailed description of the problem, we refer the interested reader to Åström and Wittenmark (1997) and Lincoln and Cervin (2002)

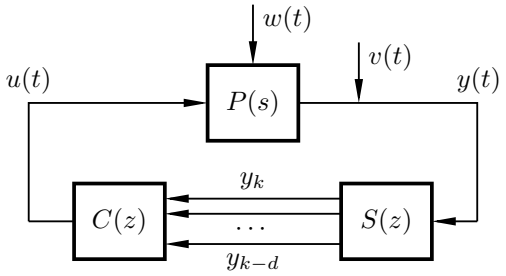

Fig. 1. `JitterTime` model for cost function evaluation. The continuous process P is connected to a discrete sampler S, which outputs both the current sample y_k as well as older samples to model long round-trip delays. The discrete controller C represents both the calculation and the actuation of the control signal.

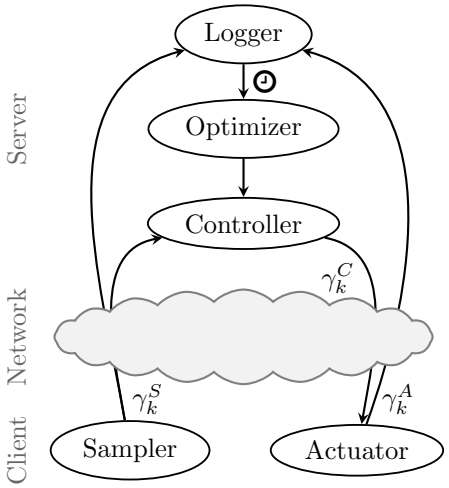

Fig. 2. An overview of the networked control system and the packets sent. Packets sent are: γ_k^S, from the sampler to the controller and Logger; γ_k^C, from the controller to the actuator; and γ_k^A, from the actuator to the Logger. Timestamps included in γ_k^S and γ_k^A are used to encode the network conditions, which is used to adapt the controller after a specified number of timestamps have been collected in the Logger.

4. IMPLEMENTATION

From the timestamps of the sampler and actuator packets, the round-trip delays and lost packets can sometimes be derived. Round-trip delay cause the difference between τ_k^S and τ_k^A. Lost γ_k^S packets cause time between received packets, in the server, to be more than one sampling period. If a γ_k^C or γ_k^A packet is dropped, the controller is updated but it is not known whether the system was actuated. For the implementation used in this paper, it is assumed only γ_k^S may be dropped. This forms a network architecture consisting of a client and server illustrated in Figure 2. The following paragraphs explain how the assumptions are used to adapt the controller.

At time k, the periodic sampler in the client sends the packet γ_k^S containing sensor measurements y_k and timestamp τ_k^S to the server. The server routes y_k to the controller routine and τ_k^S is stored in a queue denoted *Logger*, explained further below.

The controller is triggered by receiving y_k from the sampler. The internal controller states get updated and the control signal u_k is sent in packet γ_k^C back to the client once calculated, where it is routed to the actuator.

By receiving u_k, the actuator activates and changes the input to the system by holding a constant signal. A timestamp τ_k^A of when the actuation changed is sent to the server in packet γ_k^A, routed to the Logger.

Whenever both τ_k^S and the corresponding τ_k^A have arrived at the Logger, a counter is increased. Once a certain number of timestamps have been stored the timestamps, and possibly more data depending on implementation specifics, are sent to an optimizing routine. Based on the recorded network conditions, a cost function is evaluated using a digital twin of the system. This step can be

expanded to store additional logged data in order to adapt the controller.

4.1 Client-Server Setup

At the core of the presented approach is the client-server architecture, in Figure 2. The client is located near the controlled process, it consists of the sampler and actuator and interacts with the physical process. With the main requirement being network connection, the hardware could be a computationally weak microcontroller. The server process can be run remotely, such as in a virtual machine in the Cloud, where the control signal is calculated and controller optimization is performed.

The client-server communication was implemented via the WebSocket protocol [1]. With each packet sent from the client (i.e. γ_k^S or γ_k^A), a timestamp is included; the timestamps are used for the on-line adaptation by representing the network conditions.

Round-trip delays can be emulated by waiting between calculating the u_k and sending it to the actuator. Packet losses are emulated by simply not sending them.

4.2 Optimization Problem

Let $\tau = (\tau_0, \tau_1)$ be the time interval that the adaptation of the controller C should be based on. In other words, all timestamp pairs $\tau_k = \left(\tau_k^S, \tau_k^A\right)$, such that $\tau_k^S \geq \tau_0$ and $\tau_k^A \leq \tau_1$ are true, will be used to adapt the controller.

Let $\tilde{f}_\tau(C_K)$ be the value of the cost function (1) when using the specific controller C_K, parameterized by the vector of controller parameters $K \in \mathbb{R}^n$. With no further constraints, the formulation of the optimization problem therefore is simply

$$\min_K \tilde{f}_\tau(K). \tag{4}$$

Due to the real-time nature of the problem, the adaptation should be performed within a limited time from from the timestamp measurements. Thus the optimization problem not being solved sufficiently fast is a problem which must be addressed. As such, a limited number of optimization iterations are performed each time the optimization routine is performed. How many iterations to execute depends on hardware performance, deadline for the optimization routine, complexity of the system, number of timestamps used in the cost function, etc.

4.3 Real-Time Optimization in Julia

To allow for evaluation and optimization of the cost function (1), we combined the `JitterTime` [2] with the `ForwardDiff` package (Revels et al., 2016) to leverage *automatic differentiation* of the `JitterTime` simulation. Automatic differentiation exploits the composition of functions with known derivatives; derivatives of more advanced functions, consisting of these elementary functions, can thus be calculated by applying differentiation rules (particularly the chain rule) (Hoffmann, 2016). Since a `JitterTime` simulation consists of relatively simple operations such as products, additions, and matrix exponentials, it is possible to compute an exact gradient of $\tilde{f}_\tau$ with respect to the controller parameters K. The gradient can then be used to efficiently solve the optimization problem using gradient descent, following for instance the Armijo rule for deciding step lengths. The number of iterations in the real-time optimization can be adjusted to the amount of available compute resources as well as the time windows used to log timing events.

[1] For a specification of the protocol, see Melnikov and Fette (2011).
[2] Translated from its original version (written in Matlab) to Julia: `https://github.com/X-N-C/JitterTime.jl`

5. PRACTICAL EVALUATION

To evaluate the proposed adaptive approach, a remotely controlled ball and beam process was used. The ball and beam (Wellstead et al., 1978) is a classical control example, where a ball rolls freely on a beam, which in turn is rotated by a speed-controlled DC motor. The goal is to make the ball position follow a reference trajectory (viewed as an output process disturbance) by adjusting the motor voltage. The available measurement signals are the beam angle and the ball position. The process dynamics is essentially described by a triple integrator. For our particular device, a state-space model of the process is given by

$$\dot{x}(t) = \begin{bmatrix} 0 & 0 & 0 \\ -10 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 4.5 \\ 0 \\ 0 \end{bmatrix} u(t) + w(t)$$
$$y(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} x(t) + v(t)$$

where the three components of x represent the beam angle, ball velocity, and ball position, respectively. The process noise w is assumed to have the intensity $R_1 = \mathrm{diag}(1000, 50, 1)$, while the measurement noise v has the variance $R_2 = \mathrm{diag}(0, 1)$. The cost matrices are specified as $Q_1 = \mathrm{diag}(1, 0, 20)$ and $Q_2 = 100$. The reference position for the ball is chosen as a sine signal that covers half of the beam length.

The nominal, baseline controller for the process is an LQG regulator with the sampling period of 50 ms, consisting of a Kalman filter with 3×2 gain matrix L and a state feedback with a 1×3 gain vector K. After some preliminary experiments, where it was seen that the Kalman filter was not so impacted by delay, it was decided that only the state feedback gain K should be adapted.

We considered a situation where both long transmission delays and random sample packet drops could occur simultaneously. Sampling and actuation events were logged, and once 100 timestamp pairs have been stored in the Logger, the optimization routine is called; i.e., if no dropout occured the controller was adapted every 5 seconds. As a simplification of the method implementation, only packets of type γ_k^S are considered droppable. These are lost with a uniformly distributed dropout probability; as such the controller state is not updated without a corresponding actuation.

After verifying that the server can run from a remote data center and successfully control the process under nominal conditions (less than 10 ms round-trip delay and no dropped packets), the robustness of optimizing the controller was investigated. For ease of configuration

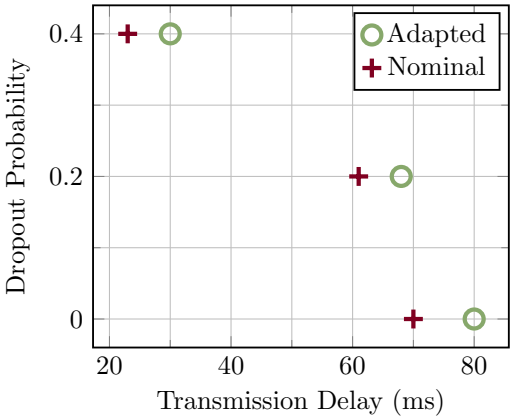

Fig. 3. For too large time delays or packet losses, the closed loop system becomes unstable. For three different packet loss probabilities, the adapted controller manages to keep the system stable for approximately 10 ms longer delays. The system is sampled with a period of 50 ms.

during the performance evaluation, both client and server were run locally.

Due to the nature of random packet drops, the controlled process was evaluated for 5 minutes. If it remained stable, the emulated round-trip delay was increased and the experiment repeated until the threshold delay was identified.

For different packet drop probabilities, the time delay thresholds for when the system becomes unstable are shown in Figure 3. The adapted controller handles time delays better than the nominal, remaining stable for circa 10 ms longer time delays in each case.

6. CONCLUSION AND FUTURE WORK

The idea explored in this paper is based on using previous network conditions as the best guess for the future, thus performance depends on how quickly the environment changes compared to how far back in time the controller is adapted to. If the update rate is too slow, the controller may be exceedingly conservative and limit performance. Although updating too quickly might require unnecessarily many computations for barely any improvement.

In this exploration, all parameters in the state feedback vector are decision variables. For larger systems, no complexity analysis of how the optimization loop scales has been done. To remedy this, maybe a subset of parameters could be optimized independently and the automatic differentiation should be in reverse mode for better efficiency. The gradient of the cost function, when the simulation is run for 100 samples, typically becomes large and the line search must be performed appropriately.

When implementing LQG controllers, an extra state for earlier control signals can be added to account for time delays, with one additional state for each sample period of delay. The controller structure of the method presented here, where the controller parameters are adapted, does not change with the time delay.

The controller has been implemented and manages to stabilize an unstable process, demonstrating how the approach can work in practice. Although there exists analytical solutions to LQG problems for systems with time delays, none exist for dropped packets.

This paper focused on the novelty of on-line adaptation of a controller based on network conditions and the implementation on a real system. There are multiple ways forward from this.

- For this paper a state feedback controller with a Kalman filter was used. To fully utilize Cloud resources, adaption of more advanced control structures could be explored.
- For step changes in round-trip delays, investigate how quickly the system must be able to adapt in order to handle delay differences of a given length.
- The current implementation of the client was run on a personal desktop computer; the client should execute on a basic microcontroller, capable of the bare minimum such as network communication.
- The currently formulated optimization problem (4) consists of only a cost function. Constraints could be added, ensuring desired properties to be obeyed. For the implementation, the suitability of more advanced methods than gradient descent could be explored.
- After each optimization cycle, the controller is updated like a jump linear system. Study properties such as stability margins when these switches occur.
- The current cost function is, based on the covariance, the total cost after a full simulation. Is this the most suitable cost function?
- Only packets with sampled sensor data are assumed losable in the setup of this paper. In the future all packet types should be handled appropriately.
- The method actuates a constant control signal during sequences of lost packets. A combination of the proposed adaption and e.g. letting the actuation fade to 0 could be investigated.
- Although the evaluated process is unstable, only the servo problem of following a trajectory was evaluated. Robustness to external physical disturbances, e.g. load disturbances, should be investigated.

REFERENCES

Abdelzaher, T., Hao, Y., Jayarajah, K., Misra, A., Skarin, P., Yao, S., Weerakoon, D., and Årzén, K.E. (2020). Five challenges in cloud-enabled intelligence and control. *ACM Trans. Internet Technol.*, 20(1).

Ackermann, J. (1980). Parameter space design of robust control systems. *IEEE Transactions on Automatic Control*, 25(6), 1058–1072. doi:10.1109/TAC.1980.1102505. URL `https://ieeexplore.ieee.org/document/1102505`.

Åström, K.J. and Wittenmark, B. (1997). *Computer-Controlled Systems (3rd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Bernstein, D., Davis, L., and Greeley, S. (1986). The optimal projection equations for fixed-order, sampled-data dynamic compensation with computation delay. *IEEE Transactions on Automatic Control*, 31(9), 859–862. doi:10.1109/TAC.1986.1104407. URL `https://ieeexplore.ieee.org/document/1104407`.

Cervin, A. (2019). JitterTime 1.0—Reference manual. Technical Report TFRT-7658, Department of Automatic Control, Lund University, Sweden.

Cervin, A., Henriksson, D., Lincoln, B., Eker, J., and Årzén, K.E. (2003). How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime. *IEEE Control Systems*, 23(3), 16–30.

Grimble, M. and Johnson, M. (1999). Algorithm for PID controller tuning using LQG cost minimization. In *Proceedings of the 1999 American Control Conference*. doi:10.1109/ACC.1999.786393. URL `https://ieeexplore.ieee.org/document/786393`.

Hoffmann, P.H.W. (2016). A hitchhiker's guide to automatic differentiation. *Numerical Algorithms*, 72(3), 775–811.

Lincoln, B. and Cervin, A. (2002). Jitterbug: A tool for analysis of real-time control performance. In *Proc. 41st IEEE Conference on Decision and Control*. Las Vegas, NV.

Melnikov, A. and Fette, I. (2011). The WebSocket Protocol. RFC 6455. doi:10.17487/RFC6455. URL `https://www.rfc-editor.org/info/rfc6455`.

Mäkilä, P. and Norlander, T. (2000). Parametric linear quadratic control and random delays. *IET Control Theory & Applications*, 147(6).

Revels, J., Lubin, M., and Papamarkou, T. (2016). Forward-mode automatic differentiation in Julia. *arXiv:1607.07892 [cs.MS]*. URL `https://arxiv.org/abs/1607.07892`.

Skarin, P., Tärneberg, W., Årzen, K.E., and Kihl, M. (2018). Towards mission-critical control at the edge and over 5G. In *2018 IEEE International Conference on Edge Computing (EDGE)*. doi:10.1109/EDGE.2018.00014. URL `https://ieeexplore.ieee.org/document/8473376`.

Syrmos, V., Abdallah, C., Dorato, P., and Grigoriadis, K. (1997). Static output feedback—A survey. *Automatica*, 33(2), 125–137. doi:https://doi.org/10.1016/S0005-1098(96)00141-0.

Vreman, N., Cervin, A., and Maggio, M. (2021). Stability and Performance Analysis of Control Systems Subject to Bursts of Deadline Misses. In B.B. Brandenburg (ed.), *33rd Euromicro Conference on Real-Time Systems (ECRTS 2021)*, volume 196. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ECRTS.2021.15.

Wellstead, P.E., Chrimes, V., Fletcher, P.R., and R. Moody, A.J.R. (1978). The ball and beam control experiment. *The International Journal of Electrical Engineering & Education*, 15(1).