Robustness Analysis for the Certification of Digital Controller Implementations*

Jerome Le Ny
University of Pennsylvania
3330 Walnut Street
Philadelphia, PA 19104, USA
jeromel@seas.upenn.edu

George J. Pappas
University of Pennsylvania
3330 Walnut Street
Philadelphia, PA 19104, USA
pappasg@seas.upenn.edu

ABSTRACT

Despite recent advances in the field of Networked Control Systems (NCS), the gap between the control design stage and the implementation stage on a physical platform remains significant. The simplifying assumptions made in the analysis of NCS are often not precise enough for realistic embedded control systems, and engineers must resort to time-consuming simulations and multiple redesign and testing phases before the performance of a system is judged adequate. Moreover, simulation-based methods do not typically provide rigorous performance or stability guarantees. We approach the problem of certifying a digital controller implementation from an input-output, robust control perspective. Following a standard method for analyzing sampled-data systems, we view the implementation step as a perturbation of a nominal linear time-invariant model. Nonlinearities and disturbances due to implementation effects are treated as uncertainty blocks and characterized via Integral Quadratic Constraints (IQCs), such as gain bounds. From our modeling discussion emerge some important types of uncertainties. We discuss some new gain bounds for one of them, namely an aperiodic sample-and-hold operator with uncertain sampling times. Two important features of the robust control approach are i) this approach is modular, i.e., the analysis of different uncertainty blocks can be done and refined separately, and the results combined in the study of a complete complex system; ii) the guarantees on the stability and performance of the implemented system can be obtained automatically via efficient computational tools.

Categories and Subject Descriptors

G.4 [**Mathematical Software**]: Reliability and Robustness; C.3 [**Special-Purpose and Application-Based Systems**]: real-time and embedded systems

1. INTRODUCTION

Networked and embedded system design places stringent requirements on the utilization of the three major classes of resources: computational resources (CPUs), communication resources, and the sensors and actuators. The implementation of control algorithms over these systems to form Networked Control Systems (NCS) poses challenging questions at the interface of several fields, in particular control theory, scheduling, and communications [29]. Usually the design of an embedded control system involves first control engineers, who model the plant and design the controller at a mathematical level of abstraction. The design is then handed off to an implementation team, who must translate it to software code implemented on a particular platform, that is, a hardware configuration, including communication networks and microcontrollers together with a Real-Time Operating System (RTOS) [2, 15, 16].

Control engineers use a simplified model of the implementation platform, chosen in order to facilitate the design of the control laws. For that purpose, the easiest model to handle assumes that the output signals of the plant are sampled periodically with no jitter in the sampling times and no quantization effect. Restricting the analysis to the sampling instants, the digital controller then sees a discrete-time time-invariant plant and discrete-time control techniques can be used [3]. A second major technique for the design of sampled-data control systems consists in starting with a continuous-time controller design, and discretizing it using a step-invariant or bilinear transformation for example [3, 5]. The artifacts introduced by the discretization can then be viewed as a perturbation of the nominal continuous-time system. The recent work on NCS essentially builds on these techniques, introducing additional elements to obtain more realistic models of the implementation platform. Some papers following the discrete-time approach, with possibly the introduction of a continuous-time component to study time-varying delays, include [14, 29, 39]. We follow the continuous-time approach, as in [26–28, 33, 36] for example.

It is important to increase the realism of the implementation platform model used at the control design stage, for several reasons:

1. Idealized models translate into hardware and scheduling requirements that are hard to meet or unclear at the implementation stage. For example, if quantization effects are neglected, how precise should the actual analog-to-digital converters be? Are the constraints on perfect periodic sampling times hard constraints (impossible to satisfy exactly) or can they be relaxed?
2. Most real-world embedded control systems have hard

*This work was supported by NSF award No. 0931239

ICCPS'10, April 13-15, 2010, Stockholm, Sweden.

cost constraints. Implementation-aware control design [2] has a direct impact on the cost of the final system, by allowing lower sampling frequencies and uncertain sampling times, less precise measurements, slower computations, lower power consumption, etc.

3. More realistic implementation models simplify the task of the implementation team, which must balance the control task requirements with those of other functions (e.g., a user interface) running on the same platform. They reduce the risk of finding the control requirements impossible to satisfy at the implementation stage, and can decrease the total time spent on system design, simulation and testing [22].

Despite recent advances, NCS models still tend to be significantly simplified versions of real-world embedded control systems, where multiple interacting control loops can involve shared sensors and actuators working at different update rates, and computations possibly performed on several Electronic Control Units (ECUs) according to a complicated schedule that must also accommodate tasks outside of the control functions. Available techniques tend to focus on one particular aspect of the implementation problem, which is unavoidable due to the variety of the possible issues. For example, the papers [28, 36] study the artifacts introduced by the scheduling delays of different sensor measurements over a shared communication network, but assume no computation delay at the controller, no propagation delay, and perfectly predictable sampling times. Propagation delays and uncertain sampling times are studied in [10, 24, 26], but there all the plant outputs are sampled at the same time and sent as one packet over the network, which can again be overly constraining from an implementation point of view. In general, there is no clear way to combine the various results available in order to study increasingly realistic models. Stability guarantees based on the construction of Lyapunov functions "by hand" for every possible communication protocol [28] is hard to generalize to the complicated and for all practical purpose non-deterministic schedules resulting from the utilization of modern RTOS and network protocols. Perhaps more importantly, the precise behavior of the scheduler is often not known at the control design stage, and constraining it too soon increases system integration issues. Finally, the importance of the notion of maximal allowable transmission interval (MATI), i.e., the time separating any two measurements received by the controller from any two sensors [36, 39], in systems involving sensors working at vastly different rates (e.g. a GPS at 20 Hz and an IMU at 1000 Hz) is not clear. In contrast, various software tools such as Ptolemy and Giotto are already available for the development of embedded control systems [16, 22] at a much greater level of detail than supported by the current theory of NCS. However, these tools are mostly used for simulation based approaches that cannot produce rigorous certificates of performance or stability. In [22] it is shown how uncertainties due to the RTOS behavior can result in loss of stability in a closed-loop system, even in the absence of a communication network.

In this paper we emphasize an input-output robust control approach to study the effects of a digital implementation on a continuous-time controller design. That is, we use a linear time-invariant (LTI) system as the nominal closed-loop model, and implementation artifacts and other nonlinearities can be included as uncertainty blocks and characterized using integral quadratic constraints (IQCs) [23], such as gain bounds. While this approach is potentially more restricted in the amount of nonlinearity it can handle in the final model compared to using a fully nonlinear nominal model [26–28], and potentially more conservative than analyzing directly a complete detailed model of the system, its has the following major advantages for our purpose:

- The stability conditions can be checked automatically with computational tools involving Linear Matrix Inequalities (LMI), via the Kalman-Yakubovich-Popov lemma.
- Most importantly, this approach is typically *modular*, and promising for the development of automated verification tools [19]. Adding disturbances and nonlinearities can be done by simply adding perturbation blocks to the nominal model. The analysis of different implementation disturbances can be done and refined separately, and combined to obtain a precise model of the chosen platform.

IQCs essentially motivated by sampled-data systems have been proposed before, see e.g. [13, 18, 20]. However, a significant part of this paper is spent on modeling aspects, and on isolating some important types of uncertainty blocks necessary for analyzing networked and embedded control systems. In addition, we provide new gain bounds complementing the result of of Mirkin [24] for aperiodic sample-and-hold blocks with bounds on the inter-sampling times.

2. MODELING

We consider the following resource utilization constraints:

1. Communication constraints. In typical embedded control applications, the signals exchanged between the plant and the computational devices (CPUs) must be transmitted via a shared communication medium, often a serial communication bus. A network protocol, such as the Controller Area Network (CAN) frequently used in automotive applications [6], is necessary to organize the access to the communication medium. This protocol is usually chosen based on all function requirements and the technology available, not just control loop constraints (indeed, the event triggered CAN with its time-varying delays is probably a bad idea from a control performance perspective).

2. Computational constraints. A CPU is not dedicated to the computation of a single control law, but instead is shared by an increasing number of tasks of various criticality levels. An example of this architecture is the Integrated Modular Avionics concept for real-time embedded airborne software [37]. This trend will continue, driven in part by economic factors that give preference to using a smaller number of general purpose CPUs instead of a large number of specially manufactured microcontrollers. Note that it is also beneficial to the load on the communication network.

These constraints on communication and computational resources are of a similar nature. They impose additional delays to the transmission of various signals through the system. The communication protocol must schedule the trans-

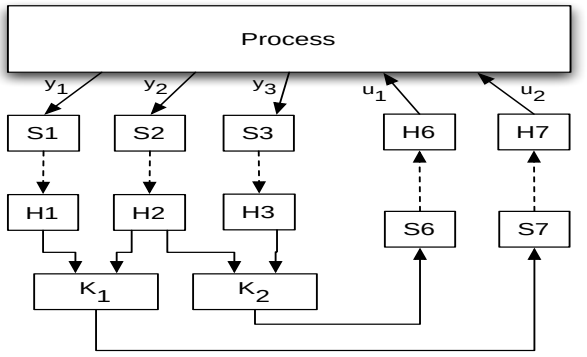


Figure 1: Mathematical model of a digital implementation of a controller on a single microcontroller. The blocks S_i represent sampling operations, the blocks H_i hold the latest received value to produce a continuous-time piecewise constant output. Discrete-time signals are represented by dashed arrows, continuous-time signals by continuous arrows.

missions of different samplers, and in addition the transmitted values also incur a potentially non-negligible propagation delay. Control input updates incur a delay due to the CPU utilization by other tasks and are scheduled by the RTOS. An important aspect of all these delays is that they are usually uncertain. Uncertainty could be due to transient perturbations on the communication network, sudden increase in communication and computational resource utilization by other non periodic tasks, etc. Even if we restrict ourselves to scheduling techniques that attempt to guarantee more precise timing properties, using for example time-triggered protocols [21], it is useful at the control design stage to give more freedom to the implementation team and guarantee that any choice of resource time-slots used in the final implementation and that satisfies certain conditions is guaranteed to be correct. At the control design level, such schedules on a time-triggered architecture could be represented by a non-deterministic automaton [38], and enlarging the set of allowed schedule greatly simplifies the integration of different subsystems using the same resources as well as their later updates. Time triggered protocols with relaxed clock synchronization requirements [4] are also motivated by cost reductions while introducing timing uncertainty.

2.1 Computational Resources

Our approach to the *mathematical* modeling of implementation disturbances can be illustrated by Fig. 1, where a continuous-time process has 3 measured outputs and 2 controlled inputs

$$y = [y_1, y_2, y_3]^T, \quad u = [u_1, u_2]^T,$$

and the control algorithm is implemented on a single microcontroller. All hold blocks in this paper represent zero-order holds. We view the implementation effects as perturbations of a nominal continuous-time Linear Time-Invariant (LTI) controller design $u = Ky$ (which can be dynamic). In this example, we have the (convolution) operator K of the form

$$K = \begin{bmatrix} k_{11} & k_{12} & 0 \\ 0 & k_{22} & k_{23} \end{bmatrix}, \; K_1 := [k_{11} \;\; k_{12}], K_2 := [k_{22} \;\; k_{23}].$$

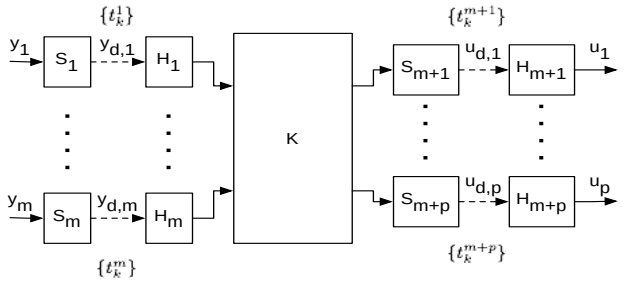


Figure 2: Mathematical model of a digital controller implementation. The times at which the signals are updated are written next to the sample and hold blocks.

For clarity of exposition, we start by assuming the absence of a shared communication network to transmit these signals between the plant and controller locations, as well as the absence of acquisition and conversion delays at the analog-to-digital and digital-to-analog converters (ADC and DAC). Moreover throughout the paper we neglect errors due to signal quantization. If all sample and hold blocks on the figure were operating synchronously and periodically, the resulting discrete-time controller would be the one obtained by the standard step-invariant transformation [5]. Here however, the different blocks are allowed to operate asynchronously, for example different signals can be sampled at different rates. Moreover there are resource constraints on the simultaneous operations of the different blocks, which are resolved by the RTOS scheduler on the microcontroller CPU.

With a fixed wiring structure, we can concentrate on the implementation of a single controller block, whose continuous-time nominal design is denoted again K in the following (for example $K = K_1$ on Fig. 1). The nominal controller K is an LTI controller of the form

$$\dot{x}_c = A_c x_c + B_c y \tag{1}$$
$$u = C_c x_c + D_c y, \tag{2}$$

where $y = [y_1, \ldots, y_m]^T$, $x_c \in \mathbb{R}^n$ and $u = [u_1, \ldots, u_p]^T$. The mathematical model of its digital implementation is represented on Fig. 2. We denote by $\{t_k^i\}$ the sequence of times at which signal y_i is sampled, and by $\{t_k^{m+i}\}$ the sequence of times at which the output signal u_i is updated. These times are the *actual physical times* when these events happen. Different implementation choices translate to various sources of uncertainty in the realization of the sequences $\{t_k^i\}, \{t_k^{m+i}\}$. We assume in general however that the times $\{t_k^i\}$ at which the sampling events happen can be recorded in variables $\mathtt{t}^{\mathtt{i}}$, transmitted to the CPU, where the value is stored in variables that we call again $\mathtt{t}^{\mathtt{i}}$. Such time-stamped samples are provided by recent standards, e.g. in the form of the "sampling ports" of the ARINC 653 specification for avionics application software [1]. In the following, we neglect the possible difference, which could be due to the acquisition time at the ADC, between the actual time t_k^i at which a physical signal changes value and the time recorded in $\mathtt{t}^{\mathtt{i}}$.

We now discuss a possible digital *implementation* of the mathematical model of Fig. 2. The goal is to be able to reproduce the outputs $u_j(t_k^{m+j})$ of the mathematical model given the samples $y_i(t_k^i)$, in order to justify the use of the

mathematical model to analyze the behavior of the system. The current value of the controller state is stored in a global variable $\mathtt{x_d}$, initialized with $\mathtt{x_d} = 0$, and we keep track of the last time at which $\mathtt{x_d}$ was updated using a global variable $\mathtt{t_x}$. Similarly the last sampled value for the input signal y_i is stored in a variable $\mathtt{y_{d,i}}$, its sampling time t_k^i in the variable $\mathtt{t^i}$, and we can initialize $\mathtt{y_{d,i}}$ to 0 for example before the first measurement becomes available. We denote $\mathtt{y_d} = [\mathtt{y_{d,1}}, \ldots, \mathtt{y_{d,m}}]^T$. The function processing samples for the signal y_i can be implemented as shown in Algorithm 1. That function has also the option to update a subset $\mathcal{O} \subset \{1, \ldots, p\}$ of the output signals, where the choice $\mathcal{O} = \emptyset$ is possible. If $\mathcal{O} \neq \emptyset$, then line 1 of the function must be executed at time t_k^i, i.e., the sampling times for signal y_i must coincide with the times at which the execution of the function starts. This can be accomplished using interrupts or by polling if the function directly activates the ADC. If $\mathcal{O} = \emptyset$, this requirement is not necessary, and the function of Algorithm 1 could process delayed but time-stamped samples. The notation $[M]_i, [M]^j$ denotes row i and column j of a matrix M, respectively. Note that the function first integrates the controller dynamics up to time t_k^i, and this state value update must occur before changing the value of the input signal stored in $\mathtt{y_{d,i}}$. Due to the fact that intersampling times may vary, the integration of the dynamics (1) requires the computation of matrix exponentials everytime a signal is sampled[1]. To compute the matrix $\mathtt{B_d}$ on line 3 of the algorithm, we also use matrix exponentials and the following lemma, see e.g. [5].

LEMMA 1. *Let A_{11} and A_{22} be square matrices and define*

$$\begin{bmatrix} F_{11}(t) & F_{12}(t) \\ 0 & F_{22}(t) \end{bmatrix} := \exp\left\{ t \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \right\}.$$

Then $F_{11}(t) = e^{tA_{11}}$, $F_{22}(t) = e^{tA_{22}}$, and

$$F_{12}(t) = \int_0^t e^{(t-\tau)A_{11}} A_{12} e^{\tau A_{22}} d\tau.$$

In other words, the term $\int_0^{\mathtt{t^i}-\mathtt{t_x}} e^{A_c\tau} d\tau$ can be obtained as F_{12} with

$$F = \exp\left\{ (\mathtt{t^i} - \mathtt{t_x}) \begin{bmatrix} 0 & I \\ 0 & A_c \end{bmatrix} \right\},$$

and moreover this computation also provides $\mathtt{A_d} = F_{22}$.

Some trivial modifications are possible. For example, we can use a single call to Algorithm 1 to process the samples of several input signals.

2.1.1 Additional Output Updates and Constraints on Valid Schedules

We also provide the option to update some controller outputs $\mathtt{u_{d,i}}$ without sampling inputs first, as described in Algorithm 2. An additional integration step is performed in order to be consistent with the model of Fig. 2. Note that the sequence of values of the global variable $\mathtt{t_x}$ should be monotonically increasing, which places some constraints on the order of the calls of the various functions by the RTOS scheduler. In general, using the functions of Algorithms 1 and 2 we can implement the model of Fig. 2, assuming that

1. the processing of samples using the functions specified by Algorithm 1 follows the same order as the sampling times $\{t_k^i\}_{k,1\leq i\leq m}$.
2. If the output update function of Algorithm 2 is executed while there are still unprocessed samples, then these samples must be discarded. In this case the time sequences $\{t_k^i\}$ of Fig. 2 correspond to the samples that are effectively processed. Algorithm 1 with output update ($\mathcal{O} \neq \emptyset$) can be used to perform frequent control updates and avoid wasting samples.
3. the computational delays introduced by lines 2-6 of Algorithm 1 or lines 2-5 of Algorithm 2 are assumed to be negligible here.

Assumptions 1 and 2 guarantee that the sequence of updates of the variable $\mathtt{t_x}$ is valid (monotonically increasing). As noted earlier, the updates of the variables $\mathtt{t}^i$ correspond to the actual sampling times of the physical input signals. However, if computational delays cannot be neglected, then Algorithms 1 and 2 do not quite correspond to the model of Fig. 2, as discussed next.

Algorithm 1 Function processing sample $y_i(t_k^i)$

Require: time-stamped samples
1: $(\mathtt{tmp}, \mathtt{t^i}) \leftarrow \mathtt{read\_input\_i}()$
2: $\mathtt{A_d} \leftarrow e^{A_c(\mathtt{t^i}-\mathtt{t_x})}$
3: $\mathtt{B_d} \leftarrow \left(\int_0^{\mathtt{t^i}-\mathtt{t_x}} e^{A_c s} ds\right) B_c$
4: $\mathtt{x_d} \leftarrow \mathtt{A_d x_d} + [\mathtt{B_d}]^i \mathtt{y_{d,i}}$
5: $\mathtt{y_{d,i}} \leftarrow \mathtt{tmp}$
6: $\mathtt{u_{d,i}} \leftarrow [C_c]_i \mathtt{x_d} + [D_c]_i \mathtt{y_d},\ i \in \mathcal{O}$ {$\mathcal{O}$ can be $\emptyset$}
7: $\mathtt{update\_output\_i}(\mathtt{u_{i,d}}),\ i \in \mathcal{O}$ {$\mathcal{O}$ can be $\emptyset$}
8: $\mathtt{t_x} \leftarrow \mathtt{t^i}$

Algorithm 2 Function updating output u_i at t_k^{m+i}

1: $\mathtt{t^{m+i}} \leftarrow \mathtt{request\_time}()$
2: $\mathtt{A_d} \leftarrow e^{A_c(\mathtt{t^{m+i}}-\mathtt{t_x})}$
3: $\mathtt{B_d} \leftarrow \left(\int_0^{\mathtt{t^{m+i}}-\mathtt{t_x}} e^{A_c s} ds\right) B_c$
4: $\mathtt{x_d} \leftarrow \mathtt{A_d x_d} + [\mathtt{B_d}]^i \mathtt{y_{d,i}}$
5: $\mathtt{u_{d,i}} \leftarrow [C_c]_i \mathtt{x_d} + [D_c]_i \mathtt{y_d}$
6: $\mathtt{update\_output\_i}(\mathtt{u_{d,i}})$
7: $\mathtt{t_x} \leftarrow \mathtt{t^{m+i}}$

2.1.2 Computational Delays

If delays due to the computations on lines 2-6 of Algorithm 1 or lines 2-5 of Algorithm 2 are significant, we need to modify the model of Fig. 2 or the functions in Algorithm 1 and 2 to make them consistent with each other. The discrepancy comes from the fact that the output produced by the implementation has value $u_i(\mathtt{t^i}) = u_i(t_k^i)$ in Algorithm 1 or $u_i(\mathtt{t^{m+i}})$ in Algorithm 2, but the output signal changes only at times $\mathtt{t^i} + \delta^{c,i}$ or $\mathtt{t^{m+i}} + \delta^{c,i}$ respectively, where $\delta^{c,i}$ is the computation delay. Hence in general we need to add a delay block after each output hold block on Fig. 2, which captures the properties of the delay $\delta^{c,i}$.

If however the computation time $\delta^{c,i}$ is perfectly known, a modification of the implemented functions can avoid the in-

[1]This is a form of jitter compensation. Although there are efficient methods for computing such exponentials, a computationally cheaper but more conservative method would be to use fixed matrices $\mathtt{A_d}, \mathtt{B_d}$, and model the error in the mathematical model using additional time-varying delays on the intersampling times.

troduction of delay blocks in the mathematical model. For this purpose, we replace $\mathtt{t}^{\mathtt{m+i}}$ by $\mathtt{t}^{\mathtt{m+i}} + \delta^{c,i}$ in lines 2, 3 and 7 of Algorithm 2. Moreover, Algorithm 1 should not be used to make output updates, that is, we should take $\mathcal{O} = \emptyset$ there. Since assumption 2 above must be satisfied, this way of compensating computational delays might not always be convenient, in which case the introduction of additional blocks in the mathematical model to account for the (perfectly known) delay cannot be avoided. If delay compensation is feasible, then with the modification of Algorithm 2 just described the output value $u_i(\mathtt{t}^{\mathtt{m+i}} + \delta^{c,i})$ is produced exactly at time $\mathtt{t}^{\mathtt{m+i}} + \delta^{c,i}$, hence we avoid the introduction of delay blocks in the mathematical model. The times at the output sample and hold devices in the mathematical model of Fig. 2 must be changed from t_k^{m+i} to $t_k^{m+i} + \delta^{c,i}$ however.

2.1.3 Static Controller Case

The implementation of a static controller ($u = D_c y$) is significantly simpler, since no integration of the controller dynamics is required. Note moreover that a static controller block K commutes with the hold and sampling blocks in the mathematical model of figure 2, and that for two blocks S, H that are synchronized (i.e., sample or change value at the same instants $\{t_k\}$), we have $SH = I$. These properties can be used to simplify the mathematical model. For example, consider a static controller with two inputs sampled simultaneously, and one output. Its implementation model, mapping continuous-time signals y_1, y_2 to the continuous-time, piecewise constant signal u_1, is $(HS)_{out} K (HS)_{in}$. Assume now that the output is updated right after sampling new values of the inputs, and that the computation time is neglected. This corresponds to using only Algorithm 1 for sampling and output update. Then the blocks H and S in the mathematical model are all synchronized. By the preceding remarks, we have $HSKHS = HK(SH)S = HKS = HSK = KHS$. The representation HSK is more efficient from a modeling and analysis point-of-view however, because it introduces the disturbance HS on a one-dimensional signal u_1 whereas KHS introduces it on the two dimensional signal y.

2.2 Effects of a Communication Network

Let us now consider the situation where a communication channel connects the plant to the controller, which we assume to be a shared communication bus (in contrast to, say, a multi-hop wireless network). A medium access control (MAC) protocol, such as the Controller Area Network (CAN) in automotive applications, must be implemented on the various resources accessing the channel (microcontrollers, sensors, actuators). Since the samples are time-stamped, the network delays do not impact the processing of samples by Algorithm 1, assuming that we take $\mathcal{O} = \emptyset$ and that the samples are processed in the same order as the sampling times $\{t_k^i\}_{k,1 \le i \le m}$. Hence the network protocol should schedule the transmission of signal samples in the order in which they become available.

Perfectly known transmission delays $\delta^{p,i}$ between the time at which an output is updated in line 6 of Algorithm 2 and the time at which it reaches the plant could potentially be compensated in the same way as known computational delays, by replacing $\mathtt{t}^{\mathtt{m+i}}$ with $\mathtt{t}^{\mathtt{m+i}} + \delta^{p,i}$ in Algorithm 2 and using $\{t_k^{m+i} + \delta^{p,i}\}$ in the mathematical model of Fig. 2. Again, the requirement of assumption 2 of subsection 2.1 can lead to wasting too many samples. We must then again account for propagation delays of the output updates by introducing delay blocks at the output of the mathematical model of Fig. 2 instead. Delay blocks can also be used to model uncertain and time-varying delays.

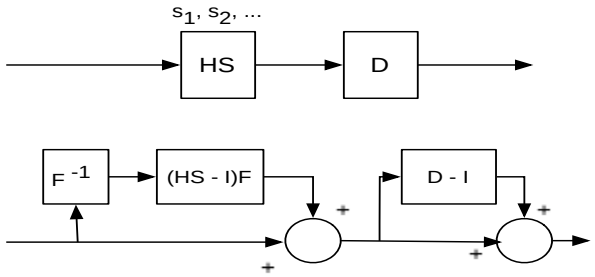


Figure 3: Implementation effects on a continuous-time signal. HS is a sample-and-hold operation at times $\{s_k\}_{k \ge 0}$, which can be uncertain. The block D models a delay, possibly uncertain and time-varying, and can be added at the outputs of Fig. 2 to model computational and communication delays. The second equivalent representation views the operation of these blocks as perturbations of the nominal signals. Note that the signals entering the delay block are piecewise constant. A low-pass filter is included in the uncertain sampling block to ensure that this block has finite energy gain.

2.3 Robustness Analysis and Important Uncertainty Blocks

We have seen in the previous sections how the mathematical model of Fig. 2 can capture the software implementation of a controller using sampling and output functions as described in Algorithms 1 and 2, with possibly additional delay blocks to model acquisition, transmission and computation delays. The mathematical model of the implementation is then used for closed-loop system analysis and potentially refinement of the control design. Note that the analysis of the mathematical model involves the sequences $\{t_k^i\}, \{t_k^{m+i}\}$ of times at which the input signals are sampled and the output signals are updated. Certain assumptions are made on these sequences at the control design stage, but various sources of uncertainty such as computational and communication delays make it hard for the scheduler to guarantee a precise control over these times. For example, consider a microcontroller executing Algorithm 1, and requesting a new sample of signal y_i. If a communication network is present, the time at which the controller request reaches the sensor might be uncertain, resulting in an uncertain sampling time t_k^i. In general, the analysis and design then involve a trade-off:

- Making more constraining assumptions on the uncertainty blocks of the mathematical model, such as negligible delays, perfectly synchronized input and output blocks with perfectly known and periodic sampling times, facilitates the control design step but typically translates into specifications that are costly and very hard or even impossible to implement on a given architecture.

- Relaxing these assumptions by increasing the class of allowed uncertainties due to implementation effects gives more flexibility to the implementation team but leads to potentially overly conservative control laws and perhaps controllers that are more complex in order to cope with these uncertainties. However, an implementation aware analysis at the initial control stage can detect earlier if certain system specifications are impossible to meet.

We have seen that viewing implementation effects as disturbances leads us to studying the following uncertainty blocks, see Fig. 3:

- Operators of the form $D - I$, where D is a delay, possibly time-varying and uncertain.
- Operators of the form $HS - I$, where HS is a sample-and-hold device, with typically time-varying and uncertain sampling times.

The trade-off mentioned above manifests itself in the control analysis by making more or less restrictive assumptions on the class of uncertain operators D and HS considered, such as variations in the sampling periods and delays, size of the delays, etc.

Robustness analysis results for systems with delay blocks that are uncertain and time-varying can be found for example in [17, 20]. In our situation, we note that all inputs to the delay blocks are piecewise constant signals, an information which could potentially be exploited. Indeed, consider a piecewise constant signal changing value at times $\{s_k\}_{k\geq 0}$ and entering an uncertain delay block. Let $\{\delta_k\}_{k\geq 0}$ be the incurred delays, and let $t_k = s_k + \delta_k$. It is not hard to see that as long as the delay block preserves the order of the sequence of switching times, a valid representation of the effect of this delay can be obtained by using the time-varying piece-linear delay model

$$w(t) = Dv(t) \Leftrightarrow w(t) = v(t - \delta(t))$$

with v piecewise constant, and $\delta(t)$ such that

$$\delta(t_k) = \delta_k,\ \frac{d\delta}{dt}(t) = 1, t_k \leq t < t_{k+1}.$$

In other words $\delta(t) = \delta_k + t - t_k$ and $w(t) = v(t_k - \delta_k)$ for $t \in [t_k, t_{k+1})$. Here the delay signal has constant derivative but is discontinuous. Another valid representation, again using the fact that the input signal is piecewise constant, is to take for $\delta(t)$ a continuous piecewise linear signal that linearly interpolates the points (t_k, δ_k). This allows us to use the results of [20].

Note that these two types of uncertainty blocks (uncertain delay and uncertain sample-and-hold) are by no means the only ones that should be considered. For example, we have not included in the discussion above any quantization effects at the sampling devices nor the fact that digital computations are performed in finite-precision arithmetic. In the rest of the paper, we consider aperiodic sample-and-hold blocks in more details.

3. APERIODIC SAMPLE AND HOLD OPERATORS

In section 2 we argued that it is important to study operators of the form $\Delta = HS - I$, where HS is a sample-and-hold operation with uncertain sampling times determined partially by the RTOS as well as communication and perhaps computational delays. This operator has been the topic of recent research papers, somewhat implicitly in e.g. [9–11, 25, 32]. Its input-output behavior, which is our main interest here, is considered explicitly in [13, 24]. Let us denote its sampling times as $\{t_k\}_{k\geq 0}$, $t_0 = 0$, and the inter-sampling times

$$h_k := t_{k+1} - t_k,\ k \geq 0.$$

We consider a situation where for control design and analysis purposes, the implementation platform and scheduler can guarantee an upper and lower bound h_u, h_l on the the inter-sampling times, that is[2]

$$0 < h_l \leq h_k \leq h_u, \forall k \geq 0. \tag{3}$$

Such bounds can be extracted from the analysis of the behavior of the scheduler deciding the execution times of the sample processing and output update functions.

Robustness analysis is based on the quantitative characterization of the uncertainties, for example via $\mathcal{L}^2$-gain bounds. In particular a recent result of Mirkin [24] can be immediately rephrased as

$$\left\| \Delta \circ \frac{1}{s} \right\| \leq \frac{2}{\pi} h_u, \tag{4}$$

where $\frac{1}{s}$ is the Laplace transform of an integrator, and throughout the paper $\|\cdot\|$ for a system S denotes the $\mathcal{L}^2$ power gain, i.e., the infimum of the set of $\gamma \geq 0$ such that

$$\inf_{T\geq 0} \int_0^T \gamma^2 |v(t)|^2 - |w(t)|^2 dt > -\infty,$$

for all locally square integrable signals v, w such that $w(t) = (Sv)(t)$ for all t.

3.1 Flexibility of the Input-Output Approach

Using an input-output approach and gain bounds such as (4) allows the study of complicated systems in a modular way. Consider for example the closed-loop system

$$\dot{x} = Ax + Bu = \begin{bmatrix} 0 & 1 \\ 0 & -0.1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u$$
$$u = Kx = -\begin{bmatrix} 3.75 & 11.5 \end{bmatrix} x,$$

which is studied in several papers, e.g. [12, 24, 25, 31, 39]. In these references, it is assumed that both outputs x_1, x_2 are sampled simultaneously. In this case, it is currently known that the sampled system is stable for uncertain inter-sampling intervals with $h_u = 1.69s$ for example [9] (the maximum *constant* sampling period for which the system is stable is $1.729s$). Note that since K is a static controller, we can use a single sample-and-hold block around the controller as discussed in subsection 2.1.3, and to simplify computations we can place this block after the single output of the controller. The result (4) together with the small-gain theorem then shows that as long as the nominal closed loop system $G(s) = sKP/(1 - KP)$ with $P = (sI - A)^{-1}B$ has $\mathcal{L}^2$ gain

[2]the lower bound h_l is introduced for technical reasons at this point, but we conjecture that increasing it does not help in obtaining tighter results in proposition 1 below.

($\mathcal{H}^\infty$-norm) less than $\pi h_u/2$, the sampled system is stable. This gives a maximum value $h_u = 1.3659s$ guaranteeing stability, which is somewhat conservative.

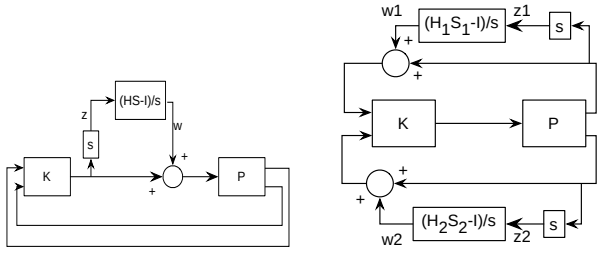
As discussed in the following subsections, it is possible to refine the characterization of the input-output behavior of the block Δ, hence there might still be room for reducing the conservativeness of the input-output approach. But in any case the main advantage of this approach in our view is the possibility of immediately considering more flexible architectures as in Fig. 2. For example, the input-output gain result (4) also allows us to consider the situation where the two plant outputs x_1, x_2 are sampled at different times with corresponding upper bounds $h_{u,1}, h_{u,2}$ on their inter-sampling times, see Fig. 3.1. The nominal system has now two inputs $w = [w_1, w_2]^T$ and two outputs $z = [z_1, z_2]^T$ using the notation of the figure, and the perturbation is block-diagonal

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} (H_1S_1 - I) \circ \frac{1}{s} & 0 \\ 0 & (H_2S_2 - I) \circ \frac{1}{s} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}. \qquad (5)$$

The transfer function of the nominal system, properly scaled, is then

$$G(s) = \begin{bmatrix} 2h_{u,1}/\pi & 0 \\ 0 & 2h_{u,2}/\pi \end{bmatrix} s(I - PK)^{-1}PK,$$

where the scaling by $D = \begin{bmatrix} 2h_{u,1}/\pi & 0 \\ 0 & 2h_{u,2}/\pi \end{bmatrix}$ is introduced to make sure that the uncertain operator defined by (5) and multiplied by D^{-1} has $\mathcal{L}^2$-gain at most 1. Applying the scaled small gain theorem [8, p.248] then yields a Linear Matrix Inequality (LMI) which is feasible if and only if $\|G\| < 1$, a sufficient condition for stability of the perturbed system. We can then verify for example that the system is stable for $h_{u,1} = 1.85$s and $h_{u,2} = 1.2$s. Hence we see that the system can accommodate sensors working at fairly different rates and a variety of schedules that do not require simultaneous sampling of the two plant output signals.



(a) Simultaneous sampling of the two outputs. (b) Distinct sampling times.

Figure 4: Example with two sampling configurations. F is a static controller so (a) can model the simultaneous sampling of the two plant outputs. In the configuration (b), the output of the controller changes every time one of the signal is sampled at the input (computational delays are neglected).

3.2 $\mathcal{L}^2$-gain Computations

In view of the flexibility of the input-output approach outlined in the previous paragraph, we consider here some additional results regarding the characterization of uncertainty blocks of the form $\Delta = HS - I$, where the sampling times are uncertain but satisfy (3). Note that a (strictly proper) pre-filter such as $1/s$ is necessary in (4) because the $\mathcal{L}^2$-gain of the unfiltered operator Δ is infinite [5]. The choice of the filter $1/s$ has been previously motivated by the study of Δ from the point-of-view of time-delay systems [10], but as pointed out by Mirkin [24] other choices are possible, which could be interesting depending on the dynamics of the nominal closed-loop system. In this subsection, we compute the $\mathcal{L}^2$-gain of the operator $\Delta \circ F$, where $F = C(sI - A)^{-1}B$ is a pre-filter with A is stable, complementing the result (4) of Mirkin who considers the case $F = 1/s$.

We can study the operator $\Delta \circ F$ by generalizing the lifting technique used for periodically sampled systems [5], as in e.g. [7, 34]. Corresponding to a sequence $\{t_k\}_{k\geq 0}$ of time instants, we have a lifting operator $L : \mathcal{L}^2[0,\infty) \to \mathcal{K}$, where $\mathcal{K}$ as a set consists of sequences $\{s_k\}_{k\geq 0}$ with $s_k \in \mathcal{K}_k := \mathcal{L}^2[0, h_k)$. L is defined by

$$Lf = \{f_k\}_{k\geq 0}, \text{ with } f_k(t) = f(t_k + t), \forall t \in [0, h_k).$$

Moreover $\mathcal{K}$ has the structure of a Hilbert space, with inner product

$$\langle u, v\rangle = \sum_{k=0}^{\infty} \langle u_k, v_k\rangle_k := \sum_{k=0}^{\infty} \int_0^{h_k} u_k(t)^* v_k(t) dt,$$

for $u = \{u_k\}_{k\geq 0}, v = \{v_k\}_{k\geq 0}$. We denote the corresponding norm on $\mathcal{K}$ simply by $\|\cdot\|_{2,\mathcal{K}}$. The lifting operator L is an isometry between $\mathcal{L}^2[0,\infty)$ and $\mathcal{K}$, which allows us to compute $\mathcal{L}^2$ gains in the lifted domain. In the following, we use the notation $\mathcal{E}$ to denote a generic Euclidean space $\mathbb{R}^s$ with appropriate dimension s. By immediate extension of the lifting techniques developed for periodically sampled systems [5], the lifted version of the operator $(HS - I) \circ F$ is an operator mapping $\mathcal{K}$ to $\mathcal{K}$ that can be represented by the linear parameter-varying discrete-time and infinite-dimensional system

$$\begin{aligned} \xi_{k+1} &= \underline{A}(h_k)\xi_k + \underline{B}(h_k)u_k, \\ y_k &= [\underline{C}(h_k) - H(h_k)C]\xi(k) + \underline{D}(k)u_k, \end{aligned} \qquad (6)$$

with $\xi_k = x(t_k)$ and the operators

$$\underline{A}(h_k) : \mathcal{E} \to \mathcal{E},\ \underline{A}(h_k) = e^{h_k A},$$

$$\underline{B}(h_k) : \mathcal{K}_k \to \mathcal{E},\ \underline{B}(h_k)u_k = \int_0^{h_k} e^{A(h_k - \tau)} B u_k(\tau) d\tau,$$

$$\underline{C}(h_k) : \mathcal{E} \to \mathcal{K}_k,\ (\underline{C}(h_k)x)(t) = Ce^{At}x,\ \forall t \in [0, h_k),$$

$$\underline{D}(h_k) : \mathcal{K}_k \to \mathcal{K}_k,\ (\underline{D}(h_k)u_k)(t) = C \int_0^t e^{A(t-\tau)} B u_k(\tau) d\tau,$$

$$H(h_k) : \mathcal{E} \to \mathcal{K}_k,\ (H(h_k)x)(t) = x,\ \forall t \in [0, h_k).$$

Define $\hat{\underline{C}}(h_k) = [\underline{C}(h_k) - H(h_k)C] : \mathcal{E} \to \mathcal{K}_k$, so that

$$(\hat{\underline{C}}(h_k)\, x)(t) = C(e^{At} - I)x,\ \forall t \in [0, h_k).$$

Finally, consider for $h > 0$ and $\gamma > \|\underline{D}(h)\|$ (the induced norm of the operator $\underline{D}(h) : \mathcal{L}^2[0,h) \to \mathcal{L}^2[0,h)$) a choice of *matrices* $\tilde{A}_{\gamma,h}, \tilde{B}_{\gamma,h}$ and $\tilde{C}_{\gamma,h}$ satisfying

$$\begin{aligned} \tilde{A}_{\gamma,h} &= \underline{A}(h) + \underline{B}(h)\underline{D}^*(h)(\gamma^2 I - \underline{D}(h)\underline{D}^*(h))^{-1}\underline{C}(h) \\ \tilde{B}_{\gamma,h}\tilde{B}_{\gamma,h}^T &= \gamma^2 \underline{B}(h)(\gamma^2 I - \underline{D}^*(h)\underline{D}(h))^{-1}\underline{B}^*(h) \\ \tilde{C}_{\gamma,h}^T\tilde{C}_{\gamma,h} &= \gamma^2 \hat{\underline{C}}^*(h)(\gamma^2 I - \underline{D}^*(h)\underline{D}(h))^{-1}\hat{\underline{C}}(h). \end{aligned}$$

Due to space constraints, we do not include here a discussion of the procedure for computing the matrices $\tilde{A}_{\gamma,h}, \tilde{B}_{\gamma,h}, \tilde{C}_{\gamma,h}$ for fixed γ, h, but it follows closely the discussion in [5, chapter 13]. Note that the choice for $\tilde{B}_{\gamma,h}, \tilde{C}_{\gamma,h}$ is not unique and these matrices can be obtained for example from a Cholesky decomposition once the right-hand side of the relations above is computed.

PROPOSITION 1. *The system $\Delta \circ F$ has $\mathcal{L}^2$-gain less than γ if $\gamma \geq \|\underline{D}(h_u)\|$ and the following parameter-dependent LMI has a feasible solution $Q \succ 0$ such that for all $h \in [h_l, h_u]$*

$$\begin{bmatrix} \tilde{A}_{\gamma,h}^T Q A_{\gamma,h} - Q + \tilde{C}_{\gamma,h}^T \tilde{C}_{\gamma,h} & \tilde{A}_{\gamma,h}^T Q \tilde{B}_{\gamma,h} \\ \tilde{B}_{\gamma,h_u}^T Q \tilde{A}_{\gamma,h} & \tilde{B}_{\gamma,h}^T Q \tilde{B}_{\gamma,h}^T - \gamma^2 I \end{bmatrix} \prec 0. \quad (7)$$

PROOF. The system has $\mathcal{L}^2$-gain less than γ if it is dissipative with respect to the supply rate $s_k(u_k, y_k) = \gamma^2 \|u_k\|_k^2 - \|y_k\|_k^2$ [35]. Hence we consider the inequality

$$S(\xi_{k+1}) - S(\xi_k) \leq \gamma^2 \|u_k\|_k^2 - \|y_k\|_k^2, \ \forall h_k, \forall u_k, \forall \xi_k,$$

where S is a storage function. Looking for a quadratic storage function $S(\xi) = \xi^T Q \xi$ leads to the operator-valued parameter dependent LMI

$$\begin{bmatrix} \underline{A}_h^T Q \underline{A}_h - Q + \hat{\underline{C}}_h^* \hat{\underline{C}}_h & \underline{A}_h^T Q \underline{B}_h + \hat{\underline{C}}_h^* \underline{D}_h \\ \underline{B}_h^* Q \underline{A}_h + \underline{D}_h^* \hat{\underline{C}}_h & \underline{B}_h^* Q \underline{B}_h + \underline{D}_h^* \underline{D}_h - \gamma^2 I \end{bmatrix} \prec 0.$$

All operators involved in this LMI have finite rank. We can then use the techniques of [5, chap. 13], [34], to transform it into the matrix-valued LMI of the proposition. We also use the fact that $\|\underline{D}(h)\|$ increases monotonically with h, which is easy to see [24]. □

CONJECTURE 1. *The parameter-dependent LMI (7) can be solved by sampling it at a finite number of values for h. By continuity of the coefficients as functions of h, for a sufficiently large number of sample points, the feasibility of the sampled system and the parameter dependent system are equivalent. More quantitative results could be obtained by using a perturbation analysis of the matrix exponential as in [12, 31]. However we conjecture that it is in fact sufficient to verify (7) for $h = h_u$, i.e., the feasibility of the LMI for $h = h_u$ implies the feasibility of the LMI for all $0 < h \leq h_u$.*

The behavior of the gain-bound of $\Delta \circ F$ as a function of h_u is illustrated on Fig. 5. The linear scaling of the bound for $F = 1/s$ (see (4)) seems to be particularly useful for nominal systems that tolerate only a small value of h_u, whereas using a stable filter F is more useful for systems that tolerate relatively larger inter-sampling times.

4. EXAMPLE

Consider the control system shown on Fig. 6, where P is the linear system

$$\dot{x} = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -20.2 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 20.2 \end{bmatrix} u$$

$$\begin{bmatrix} \alpha \\ q \end{bmatrix} = \begin{bmatrix} 57.2958 & 0 & 0 \\ 0 & 57.2958 & 0 \end{bmatrix} x.$$

Without the sample-and-hold blocks, this system is the continuous-time model of an aircraft pitch controller described in [30]. Here α is the angle of attack and q the pitch rate.

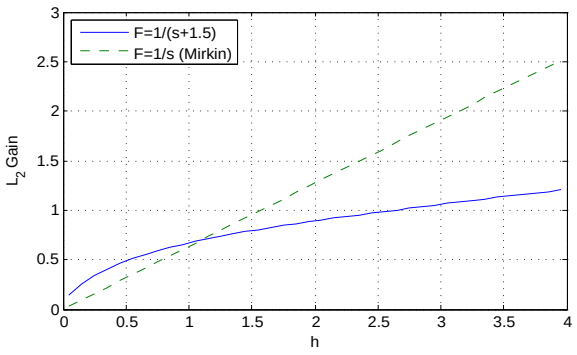


Figure 5: $\mathcal{L}^2$-gain bound of $(HS-I)\circ F$ as a function of h_u, for $F = 1/s$ [24], and $F = 1/(s+1.5)$ (proposition 1).

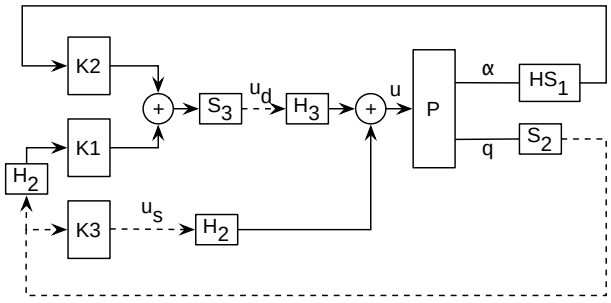


Figure 6: Mathematical model of a digital implementation of a pitch controller with multiple sampling and control update rates.

A continuous-time design, which does not take into account implementation effects, leads to a controller composed of 3 blocks

$$K_1(s) = \frac{k_1}{s}, \ K_2(s) = \frac{k_2}{s + p_2}, \ K_3(s) = k_3$$

$$k_1 = 1.361, \ k_2 = 0.807, \ p_2 = 10, \ k_3 = 0.475.$$

In [30] a digital implementation of this controller is suggested, where all sampling and hold devices are assumed to be synchronized. A sampling period for all these devices of $0.025s$ is proposed heuristically. We can now gain some insight into the behavior of a more realistic implementation with devices that are not synchronized and operate with different sampling rates, according to the discussion of section 2.

Let us consider the following digital implementation of this controller. We neglect any network and computation delay in this example, hence consider only uncertain sample and hold blocks. A single CPU implements digitally the controllers K_1, K_2, K_3, and the mathematical model of the digital implementation is obtained by adding the sample and hold blocks in the configuration shown on Fig. 6. The sampling times for the three sample and hold devices are denoted $\{t_k^1\}, \{t_k^2\}$ and $\{t_k^3\}$ respectively. This model can represent the following implementation. The signals α and q are sampled at times $\{t_k^1\}$ and $\{t_k^2\}$ respectively. There

are two scalar variables $\mathtt{x}_1$ and $\mathtt{x}_2$ that keep track of the internal states of the blocks K_1, K_2 respectively, and that are stored together with the last time $\mathtt{t}_{x1}, \mathtt{t}_{x2}$ at which they were updated. Additional variables $\mathtt{u}_d$ and $\mathtt{u}_s$ store the current value of the output of the dynamic and static parts of the controller respectively. A first function f_1 reads the latest sample value of α, stores its time-stamp in $\mathtt{t}^\alpha$, and updates the value of $\mathtt{x}_2$ as described in Algorithm 3 by integrating the dynamics of K_2 since the last update of $\mathtt{x}_2$.

Algorithm 3 Function f_1 for processing samples of α

1: $(\mathtt{tmp}, \mathtt{t}^\alpha) \leftarrow \mathtt{read\_input\_}\alpha()$
2: $\mathtt{x}_2 \leftarrow e^{-p_2(\mathtt{t}^\alpha - \mathtt{t}_{x2})}\mathtt{x}_2 + \frac{\alpha}{p_2}(1 - e^{-p_2(\mathtt{t}^\alpha - \mathtt{t}_{x2})})$
3: $\alpha \leftarrow \mathtt{tmp}$
4: $\mathtt{t}_{x2} \leftarrow \mathtt{t}^\alpha$

A second function f_2 is in charge of requesting and processing samples for the signal q. In addition, it performs an integration step for K_1, and updates the output by computing the output of the static controller K_3 and combining it with the current value $\mathtt{u}_d$ of the output of the dynamic part of the controller. The output port with the analog control signal changes value at the times $\{t_k^2\}$ as well because computational delays are neglected, see subsection 2.1.2. Finally, a third function f_3 only updates the controller output, after performing integration steps for the dynamic controllers according to Algorithm 2. An output update can be requested by the scheduler even if no new measurement sample was collected, i.e., the update results only from the integration of the the controller dynamics. However, since function f_3 integrates the dynamics of the controller K_2 in particular, executing it at time t requires to discard any older available sample of the signal α that has not yet been processed by the function f_1 at that time, as discussed in subsection 2.1.1.

Algorithm 4 Function f_2 for processing samples of q

1: $(\mathtt{tmp}, \mathtt{t}^\mathtt{q}) \leftarrow \mathtt{read\_input\_q}()$
2: $\mathtt{x}_1 \leftarrow k_1(\mathtt{t}^\mathtt{q} - \mathtt{t}_{x1})\mathtt{q}$
3: $\mathtt{q} \leftarrow \mathtt{tmp}$
4: $\mathtt{u}_s \leftarrow k_3\mathtt{q}$
5: $\mathtt{update\_output}(\mathtt{u}_d + \mathtt{u}_s)$
6: $\mathtt{t}_{x1} \leftarrow \mathtt{t}^\mathtt{q}$

Algorithm 5 Function f_3 for additional updates of the controller output

1: $\mathtt{t}^3 \leftarrow \mathtt{request\_time}()$
2: $\mathtt{x}_2 \leftarrow e^{-p_2(\mathtt{t}^3 - \mathtt{t}_{x2})}\mathtt{x}_2 + \frac{\alpha}{p_2}(1 - e^{-p_2(\mathtt{t}^3 - \mathtt{t}_{x2})})$
3: $\mathtt{x}_1 \leftarrow k_1(\mathtt{t}^3 - \mathtt{t}_{x1})\mathtt{q}$
4: $\mathtt{u}_d \leftarrow \mathtt{x}_1 + \mathtt{x}_2$
5: $\mathtt{update\_output}(\mathtt{u}_d + \mathtt{u}_s)$
6: $\mathtt{t}_{x1} \leftarrow \mathtt{t}^3$
7: $\mathtt{t}_{x2} \leftarrow \mathtt{t}^3$

Using the $\mathcal{L}^2$-gain analysis outlined above, we can for example verify at the control design stage that as long as the scheduler can guarantee that the sequences $\{t_k^1\}, \{t_k^2\}, \{t_k^3\}$ satisfy $h_{u,1} = 4s, h_{u,2} = 0.049s$ and $h_{u,3} = 0.455s$ for example, the system is stable. In particular, we require a relatively faster update rate of the static controller output $\mathtt{u}_s$ in order to guarantee stability. These values where obtained using the setup shown on Fig. 7. The nominal closed-loop system has 3 inputs w_1, w_2, w_3 and 3 outputs z_1, z_2, z_3, and the pre-filters used are $F_1 = \frac{1}{s+1}$, $F_2 = \frac{1}{s}$, $F_3 = \frac{1}{s}$. In addition to stability, performance measures expressed in terms of input-output power gain can be studied in a straightforward way.

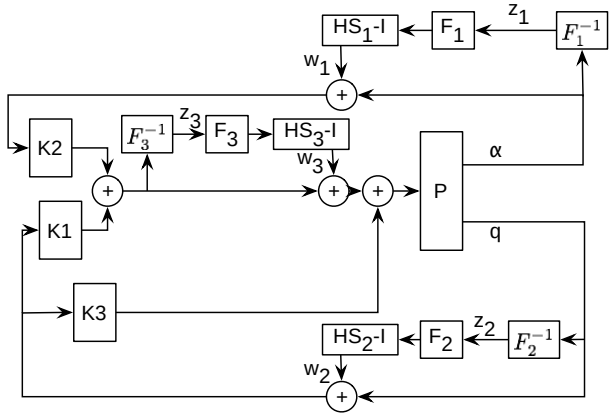


Figure 7: Setup for Robustness Analysis.

5. CONCLUSION

In this paper, we have discussed the use of robust control techniques based on input-output analysis to study the effect of digital implementations on nominal controller designs. The analysis of control laws implemented over networks is currently a very active area of research, and similar resource constraint issues arise when one considers computational constraints. Input-output control techniques offer a promising direction of research for the analysis of networked and embedded control systems, due to the modular nature of the approach. Starting from a particular nominal control system design, a careful implementation respecting a few rules can be modeled mathematically by adding decoupled uncertainty blocks. Stability and performance guarantees can then be obtained automatically using efficient computational methods for a set of allowable schedules, instead of requiring the exact specification of the scheduler behavior at the control design stage. Adding or removing uncertainty blocks, corresponding to different implementation choices, does not change the framework of the analysis. These aspects make the input-output approach a good candidate for the development of automated tools, capable of certifying the properties of a nominal control system design for a range of realistic implementation platforms, thereby simplifying subsequent system integration and update issues.

6. REFERENCES

[1] R. Alena, J. Ossenfort, K. Laws, A. Goforth, and F. Figueroa. Communications for integrated modular avionics. In *Proceedings of the IEEE Aerospace Conference*, pages 1–18, 2007.

[2] K.-E. Arzen, A. Cervin, and D. Henriksson. Implementation-aware embedded control systems. In D. Hristu-Varsakelis and W. Levine, editors, *Handbook of Networked and Embedded Control Systems*, 2005.

[3] K. Astrom and B. Wittenmark. *Computer-Controlled Systems: Theory and Design*. Prentice Hall, 3rd edition, 1997.

[4] A. Benveniste, P. Caspi, P. L. Guernic, H. Marchand, J.-P. Talpin, and S. Tripakis. A protocol for loosely time-triggered architectures. In *Proceedings of the Second International Conference on Embedded Software*, volume 2491 of *Lecture Notes In Computer Science*. Springer, 2002.

[5] T. Chen and B. Francis. *Optimal Sampled-Data Control Systems*. Springer, 1995.

[6] R. Davis, A. Burns, R. Bril, and J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):1573–1383, April 2007.

[7] G. Dullerud and S. Lall. Asynchronous hybrid systems with jumps - analysis and synthesis methods. *Systems and Control Letters*, 37:61–69, 1999.

[8] G. Dullerud and F. Paganini. *A Course in Robust Control Theory: A Convex Approach*. Springer, 2000.

[9] E. Fridman. A refined input delay approach to sampled-data control. *Automatica*, 2009. In Press.

[10] E. Fridman, A. Seuret, and J.-P. Richard. Robust sampled-data stabilization of linear systems: An input delay approach. *Automatica*, 40:1441–1446, 2004.

[11] E. Fridman, U. Shaked, and V. Suplin. Input/output delay approach to robust sampled-data H_∞ control. *Systems and Control Letters*, 271-282(54), 2005.

[12] H. Fujioka. A discrete-time approach to stability analysis of systems with aperiodic sample-and-hold devices. *Transactions on Automatic Control*, 54(10):2440–2445, October 2009.

[13] H. Fujioka. Stability analysis of systems with aperiodic sample-and-hold devices. *Automatica*, 45:771–775, 2009.

[14] V. Gupta, B. Hassibi, and R. Murray. Optimal LQG control across packet-dropping links. *System and Control Letters*, 56(6):439–449, June 2007.

[15] H. Hanselmann. Implementation of digital controllers - a survey. *Automatica*, 23(1):7–32, 1987.

[16] T. Henzinger, B. Horowitz, and C. Kirsch. Giotto: A time-triggered language for embedded programming. *Proceedings of the IEEE*, 91(1):84–99, 2003.

[17] M. Jun and M. Safonov. IQC robustness analysis for time-delay systems. *International Journal of Robust and Nonlinear Control*, 11:1455–1468, 2001.

[18] C.-Y. Kao and B. Lincoln. Simple stability criteria for systems with time-varying delays. *Automatica*, 40:1429–1434, 2004.

[19] C.-Y. Kao, A. Megretski, U. Jönsson, and A. Rantzer. A MATLAB toolbox for robustness analysis. In *Proceedings of the IEEE conference on computer aided control systems design*, Taipei, Taiwan, 2004.

[20] C.-Y. Kao and A. Rantzer. Stability analysis of systems with uncertain time-varying delays. *Automatica*, 43:959–970, 2007.

[21] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.

[22] J. Liu, J. Eker, J. Janneck, and E. Lee. Realistic simulations of embedded control systems. In *Proceedings of the 15th IFAC World Congress*, 2002.

[23] A. Megretski and A. Rantzer. System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, 42(6):819–830, June 1997.

[24] L. Mirkin. Some remarks on the use of time-varying delay to model sample-and-hold circuits. *IEEE Transactions on Automatic Control*, 52(1109-1112), 2007.

[25] P. Naghshtabrizi, J. Hespanha, and A.R.Teel. Exponential stability of impulsive systems with application to uncertain sampled-data systems. *Systems and Control Letters*, 57(5):378–385, May 2008.

[26] P. Naghshtabrizi, J. Hespanha, and A. Teel. Stability of delay impulsive systems with application to networked control systems. *Transactions of the Institute of Measurement and Control, Special Issue on Hybrid and Switched Systems*, 2009. To appear, available at http://www.ece.ucsb.edu/ hespanha/published.

[27] D. Nesic and D. Liberzon. A unified framework for design and analysis of networked and quantized control systems. *IEEE Transactions on Automatic Control*, 54(4):732–747, 2009.

[28] D. Nesic and A. Teel. Input-output stability properties of networked control systems. *IEEE Transactions on Automatic Control*, 49(10):1650–1667, October 2004.

[29] J. Nilsson. *Real-Time Control Systems with Delays*. PhD thesis, Dept. Automatic Control, Lund Institute of Technology, Lund, Sweden, January 1998.

[30] B. Stevens and F. Lewis. *Aircraft Control and Simulation*. Wiley, 2nd edition, 2003.

[31] Y. Suh. Stability and stabilization of nonuniform sampling systems. *Automatica*, 44:3222–3226, 2008.

[32] V. Suplin, E. Fridman, and U. Shaked. Sampled-data H_∞ control and filtering: Nonuniform uncertain sampling. *Automatica*, 43:1072–1083, 2007.

[33] M. Tabbara, D. Nesic, and A. Teel. Networked control systems: Emulation-based design. In F.-Y. Wang and D. Liu, editors, *Networked Control Systems: Theory and Applications*, pages 57–94. Springer, 2008.

[34] K. Tan, K. Grigoriadis, and F. Wu. Output-feedback control of LPV sampled-data systems. *International Journal of Control*, 75(4):252–264, 2002.

[35] A. van der Schaft. *L2-Gain and Passivity Techniques in Nonlinear Control*. Springer, 2nd edition, 2000.

[36] G. Walsh, H. Ye, and L. Bushnell. Stability analysis of networked control systems. *IEEE Transactions on Control Systems Technology*, 10(3):438–446, 2002.

[37] C. Watkins and Walter. Transitioning from federated avionics architectures to integrated modular avionics. In *Proceedings of the 26th Digital Avionics Systems Conference*, 2007.

[38] G. Weiss and R. Alur. Automata based interfaces for control and scheduling. In *Proceedings of the 10th International Conference on Hybrid Systems: Computation and Control*, 2007.

[39] W. Zhang, M. Branicky, and S. Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, 21(1):84–99, 2001.