Real-time and delay-dependent control co-design through feedback scheduling

Daniel Simon, Olivier Sename[1], David Robert and Olivier Testa
INRIA Rhône-Alpes, Équipe POP ART, 655 avenue de l'Europe, 38330 Montbonnot, France
{Daniel.Simon, Olivier.Sename}@inrialpes.fr

Abstract

Control systems are often designed using a set of cooperating periodic modules running under control of a real-time operating system. As a correct behaviour of the closed-loop controller requires that the system meets timing constraints like periods and latencies, the structure and timing parameters of the controller must be set according to control requirements. To take into account timing uncertainties, e.g. due to preemption, a delay-dependent feedback loop has been designed; the scheduling controller regulates the resource utilisation according to the estimated execution times. The aim is here to control the Quality of Service consisting of computing resources utilisation and quality of control contribution (evaluated here through the tracking error and the robustness w.r.t to control delay). The actuators are the tasks periods and a H_∞ control approach provides robustness w.r.t. modelling errors. A simulated example with two pendulums enlightens the approach.

1 Introduction

Most feedback control systems are essentially periodic, where the inputs (reading on sensors) and the outputs (posting on actuators) of the controller are sampled/hold at a predefined rate. While basic digital control theory deals with systems sampled at a single rate, it has been shown, e.g. [19], that the control performance of a non-linear system like a robot can be improved using a multi-rate controller : some parts of the control algorithm, e.g. updating parameters or controlling slow modes, can be executed at a pace slower than the one used for fast modes. In fact, a complex system involves sub-systems with different dynamics which must be further coordinated. Therefore the controller must run in parallel several control laws with different sampling rates inside a hierarchy of more or less tightly coordinated layers.

Digital control systems are often implemented as a set of tasks running on top of an off-the-shelf real-time operating system (RTOS) using fixed-priority and preemption. The performance of the control, e.g measured by the tracking error, and even more importantly its stability, strongly relies on the values of the sampling rates and computing latencies[1] (delays) [2]. Therefore it is essential that the implementation of the controller will respect an adequate temporal behaviour to meet the expected performance. However, control science and real-time computing most often evolved independently leading to misconceptions and lack of efficiency in the implementation of real-time control algorithms.

This paper deals with real-time scheduling and control co-design. First, as in [5], we are interested in feedback scheduling, i.e. we aim to adjust on-line the sampling periods of the controllers in order to meet the computing resource requirements. Thus, the process control parameters are changed on-line according to the required sampling periods. Furthermore the process control design takes into account control delays that are unavoidable in real-time control [3]. We here provide a state feedback control law for discrete systems with control delay, the parameters of which depend on the sampling period of the controllers.

The outline of the paper is the following. Section 2 recalls the state of control theory w.r.t. sampling and delays while section 3 enlightens some deficiencies between popular real-time paradigms and control requirements. Section 4 provides guidelines for a static implementation of closed-loop control systems. Section 5 presents the considered architecture for feedback scheduling and the Quality of Control (QoC) specifications. An illustrative example of a multi-tasks control system is presented in section 6 where the advantage of the proposed methodology is emphasised. The paper ends with some concluding remarks and further research directions.

2 Control and timing

Once a control algorithm has been designed, a first job consists in partitioning it into tasks and then in the assignment of timing parameters, i.e. periods of tasks and I/O latencies, so that the controller's im-

[1] on leave from Laboratoire d'Automatique de Grenoble, ENSIEG-BP 46, 38402 Saint Martin d'Hères Cedex, FRANCE

[1] the latency is the delay between the instant of a measure q_n on a sensor and the instant when the control signal $U(q_n)$ is sent to the actuators.

plementation satisfies the control objective. Control theory for linear systems sampled at fixed rates, possibly with *fixed and known pure time* delays, has been established a long time ago. Some more recent results deal with varying or unknown delays or sampling rates in control loops, still in the framework of linear systems, e.g. [13], [9]. Unfortunately most real-life systems are non-linear. The extrapolation of timing assignment through linearising often gives rough estimations of allowable periods and latencies or even can be meaningless. Thus slicing the control algorithm and setting adequate values for the timing parameters rapidly falls into case studies based on simulation and experiments. Such case-studies may benefit from off-line control/scheduling co-design, e.g. [16] using off-line iterative optimisation, to compute an adequate setting of periods, latencies and gains resulting in a requested control performance according to the available computing resource and implementation constraints.

Control systems are often cited as examples of "hard real-time systems" where deadline violations are strictly forbidden. In fact experiments show that this assumption may be false for closed-loop control. Any practical feedback system is designed to obtain some stability margin and robustness w.r.t. the plant parameters uncertainty. This also provides robustness w.r.t. timing uncertainties: closed-loop systems are able to tolerate some amount of sampling period and computing delays deviations, jitter and occasional data loss with no loss of stability or integrity, but only disturbances. The hard real-time assumption should be changed for a "weakly hard" one, where absolute deadlines would be replaced by statistical ones, e.g. the allowable output jitter compliant with the desired control performance. Even if computing such statistics is out of the scope of current control theory, this intrinsic robustness of closed-loop controllers gives an additional degree of freedom which can comply with Quality of Service (QoS) computation and flexible scheduling design.

3 Traditional real-time models

Usually, real-time systems are modelled by a set of recurrent tasks assigned to one or several processors and a worst case response times technique is used to analyse fixed-priority real-time systems as initiated in [10]. Well known scheduling policies, such as Rate Monotonic for fixed priorities and EDF for dynamic priorities [23], assign priorities according to timing parameters, respectively sampling periods and deadlines. They are said "optimal" as they maximise the number of tasks sets which can be scheduled with respect of deadlines, under some restrictive assumptions. They are very popular but they must not be used blindly.

They hardly take into account precedence and synchronisation constraints which naturally appear in a control algorithm. The relative urgency or criticality of the control tasks can be unrelated with the timing parameters. Thus, the timing requirements of control systems w.r.t. the desired control goal expressed as a performance index do not fit well with scheduling policies purely based on schedulability tests. It has been shown through experiments, e.g. [3], that a blind use of such traditional scheduling policy can lead to an inefficient controller implementation; on the other hand a scheduling policy based on application's requirements, associated with a right partition of the control algorithm into real-time modules may give better results.

Another example of unsuitability between computing and control requirements arises when using priority inheritance or priority ceiling protocols to bypass priority inversion due to mutual exclusion, e.g. to ensure the integrity of shared data. While they are designed to avoid dead-locks and minimise priority inversion lengths, such protocols jeopardise at run-time the initial schedule which was carefully designed to meet control requirements. As a consequence latencies along some control paths can be increased to values far over their desired value, leading to a poor control performance or even instability.

Design and off-line schedulability analysis rely on a right estimation of the tasks worst case execution time. Even in embedded systems the processors use caches and pipelines to improve the average computing speed while decreasing the timing predictability. Another source of uncertainty may come from some pieces of the control algorithm. For example, the duration of a vision process highly depends on incoming data from a dynamic scene. Also some algorithms are iterative with a badly known convergence rate, so that the time before reaching a predefined threshold is unknown (and must be bounded by a timeout). Finally, in a dynamic environment some control activities can be suspended or resumed and control algorithms with different costs can be scheduled according to various control modes leading to large variations in the computing load. Thus real-time control design based on worst case execution and strict deadlines inevitably leads to a low average usage of the computing resource.

4 Control systems and off-the-shelf RTOS

The controller is most often implemented as a set of modules, each of them encoding a piece (function) of the control algorithm. At run-time the control modules are basically periodic and are scheduled using the basic features of the RTOS, i.e. priority based preemption and synchronisation primitives.

4.1 Priority assignment

A control system for a robot, and more generally for process control, can be split into several calculation paths [1] : the direct control path computes control set-points from tracking errors and must run with a small period and a low latency to ensure the process stability. The respect of the above timing constraints is critical w.r.t. the control performance, i.e. this part of the controller has a high relative urgency. Others urgent activities are critical tasks which monitor the system's activity and which are triggered by the detection of deviations from the nominal behaviour. In fact some of these error recovery procedures can be better triggered by interrupts rather than by periodic polling.

Other tasks are used to update slowly varying parameters of the non-linear plant model. These tasks are often data-handling intensive, e.g. using trigonometric functions or matrix inversion. Their duration can be far longer than the period assigned to the direct path, but delaying their ending instants has a weak effect on the controller performance, e.g. the control jitter or the system's stability. Thus they can be assigned a low priority so that their execution is preempted by every execution of the direct path calculation [19].

The whole system is usually run over a single CPU, or on a limited number of CPUs with a static partition of the tasks. *Priorities must be assigned to tasks according to their relative urgency*, so that a high criticality module can preempt the execution of a less urgent one when it becomes runnable. This also ensures that in case of overload the tasks which are serviced on time will be the most critical ones ; obviously provision must be made to avoid an operating system's crash in case of overload or repeated deadlines miss. As far as the control algorithm does not change, the relative urgency of modules does not change either so that priorities inside the control algorithm can be statically assigned. Anyway the relative urgency between tasks may change during a restructuring of the system, e.g. for admission of new control tasks ; in that case this is done at a rate imposed by the controlled system and environment's dynamics, not at the scheduler's rate like in EDF.

However, using only preemption is not enough to accurately specify the controller, in particular it cannot efficiently take into account the precedence constraints between subsets of the control algorithm.

4.2 Synchronisation and communication

A partial synchronisation of tasks allows for the specification of precedence constraints and thus improves the control performance by minimising the latency on critical paths.

Generally, the best data to be used in a *closed-loop* control algorithm is the last one produced, thus the buffers between modules may have only one slot, and the incoming data overwrites the old one. The following basic communication/synchronisation schemes are of particular interest to implement closed-loop controllers :

- Synchronising the starting of a module on the end of another module (data production) is of prime interest to specify data dependency and to ensure latency minimisation on some critical control paths [19]; such one way synchronisation with a system's clock is also a good way to provide for periodic tasks.

- Asynchronous communication must be used between modules with unrelated sampling rates. As mutual exclusion on shared data using mutex introduces side effect run-time synchronisation (priority inversion), lock-free shared data protection mechanisms, e.g. [20], must be preferred;

- Strong synchronisation (the well known rendezvous) must be sparingly used, and its appropriateness w.r.t. the application requirements must be carefully considered as it is a very efficient dead-lock generator.

Some comprehensible rules, based on control algorithm analysis, layers of static priority and careful synchronisation, allows for the design of rather complex controllers taking into account the control requirements; they also lead to off-line analysis independently of the priority assignment, e.g. [18].

However, the design and schedulability analysis of systems based on static values of the scheduling parameters assumes the *a priori* knowledge of the worst case execution time of the tasks [15]. This is always difficult to measure, and can lead to a severe under-use of the computing power when the computing load has large variations, e.g. in vision-based control. Another source of timing uncertainty arises from variable communication delays when the controller is distributed over a local area network, leading to measuring jitter. Perturbations on the system's computing load also arise during restructuring due to admission or cancellation of control activities upon occurrence of events coming from a dynamic environment. Adaption against timing uncertainties could be provided by more flexible scheduling policies.

5 Further results on feed-back scheduling

It is expected that an on line adaption of the scheduling parameters of the controller may increase its overall efficiency w.r.t. timing uncertainties coming from the unknown controlled environment. Also we know from control theory that closing the loop may increase performance and robustness against disturbances when properly designed and tuned (otherwise it may lead to instability). Thus the idea of feed-back scheduling recently arose both from the control side

[6, 5] and from the real-time computing side [22, 11]. Anyway the design of efficient feed-back schedulers must start with a safe design of a real-time implementation based on control requirements as previously described (i.e. algorithm partition, precedence constraints and priority assignment).

Figure 1 gives an overview of a feed-back scheduler where an outer loop (the *scheduling controller*) has been added to the process controller to adapt in real-time the scheduling parameters ; it takes as input measures taken on both the controlled process output (e.g. the tracking error) and on the computer's activity (e.g. the computing load). Besides this controller working periodically at a rate which is itself to be determined, the system's structure may evolve along a discrete time scale upon occurrence of events, e.g. for new tasks admission or exception handling. These decisional processes are handled by another real-time task, the *scheduling manager*, which is not further detailed in this paper. Preliminary studies and experiments show that the tools needed to handle the measurements and actuation tasks, e.g. precise time-stamping of events, already exist in most off-the-shelf RTOS ; thus these housekeeping tasks can be built in a middleware layer between the kernel and the application tasks and we do not need to patch an existing real-time kernel.

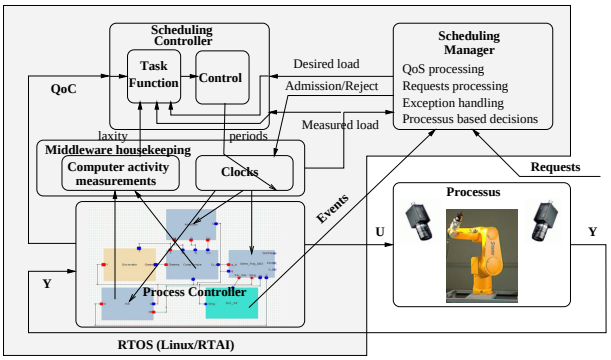

Figure 1: Feed-back scheduler structure

The problem can thus be stated as QoC (Quality of Control) optimisation under constraint of available computing resources. During experiments an estimate of the current utilisation may be computed as:

$$U = \sum_{i=1}^{n} \frac{C_i}{h_i}$$

where C_i is the estimated execution time of the task i, and h_i the sampling period assigned to task i.

Preliminary studies [6] suggest that a direct synthesis of the scheduling regulator as an optimal control problem leads, when it is tractable, to a solution too costly to be implemented in real-time[2]. Practical solutions will be found in the available control toolbox or in enhancements and adaptation of current control theory. For instance the calculation of the new task periods can be done by the rescaling [4]:

$$h_i^{new} = h_{i_{nominal}} \frac{U}{U_{sp}}$$

where U_{sp} is the utilisation set-point.

The feedback scheduler then controls the processor utilisation by assigning task periods that optimise the overall control performance.

5.1 A new feedback scheduling architecture

In order to adjust on-line the scheduling parameters of the control tasks, a control-scheme should be established for the scheduler, as done in [5]. As any control design problems, the important issues are therefore the specification of control inputs, measurement outputs and control structure.

5.1.1 Control structure: Feedback scheduling is a dynamic approach allowing to better use the computing resources, in particular when the workload changes e.g. due to the activation of an admitted new task. We propose in figure 2 a hierarchical control structure. The feedback scheduler controls the CPU activity according to the computing resource availability (measured through some computing load metric) by adjusting the periods of the tasks used in the process controller(s).

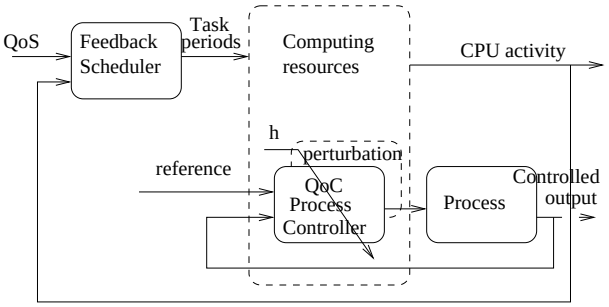

Figure 2: Hierarchical control structure

On the other hand the internal process controller is here designed to take into account timing uncertainties, e.g. due to preemptions which are unavoidable in real-time control and difficult to accurately predict in a dynamic environment. Indeed unknown input-output latencies can deteriorate the process performances and stability; thus the considered Quality of Control measure is composed of the usual tracking error and the robustness w.r.t to control delays.

Note that in this preliminary study this structure is simplified compared with the one of figure 1 : ideally the QoC measured from the controlled process would be also fed back to the scheduler and be taken into account in the QoS computation.

[2] recall that the feed-back scheduler will be itself a real-time task loading the shared computing resource...

5.1.2 Sensors and actuators: As stated in section 4, priorities must be assigned to control tasks according to their relative urgency ; this ordering remains the same in the case of a dynamic scheduler. Dynamic priorities, e.g. as used in EDF, only alter the interleaving of running tasks and will fail in adjusting the computing load w.r.t. the control requirements measured by the QoC. In consequence we have elected the tasks periods to be the main actuators of the system running on top of a fixed priority scheduler[3]. Anyway, to be compliant with control requirements the control algorithms must be first adequately sliced into sets of ordered synchronised and communicating real-time tasks as in the static case.

Our aim is to adjust on-line the sampling periods of the controllers in order to meet the computing resource requirements. The control inputs are then the periods of the control tasks. The measured output is the CPU utilisation, estimated through the execution time, in a similar way as in [5]. Thus, for each period of the scheduler h_S, the CPU utilisation is estimated from job execution-time measurements of the control tasks, as:

$$\begin{aligned} \overline{U}(kh_S) &= \sum_{i=1}^{n} \frac{\overline{c}_i(kh_S)}{h_i((k-1)h_S)} \\ \hat{U}(kh_S) &= \lambda \hat{U}((k-1)h_S) + (1-\lambda)\overline{U}(kh_S) \quad (1) \end{aligned}$$

where, for each period of the scheduler, $h_i(kh_S)$ is the sampling period currently assigned to the control task i, and $\overline{c}_i(kh_S)$ is here the mean of the measured execution-times of the control task i during each period of the scheduler. Samples for the measured output (i.e. CPU utilisation) are taken here at the period of the scheduler to be controlled, which is usual for identification purpose. In (1)λ is a forgetting factor, chosen as $\lambda = 0.3$, which ensures a fast estimation.

5.1.3 Control design and implementation: Our aim is here to provide a new control scheme for the feedback scheduler. First one should note that, if the execution times are constant, then the relation, $U = \sum_{i=1}^{n} C_i f_i$ (where $f_i = 1/h_i$ is the frequency of the task) is a linear function (while it would not be as a function of the task periods). For the considered application with two control tasks, illustrated in section 6, we obtain the following static map. This linear characteristic of the static behaviour of the scheduler implies that we can consider a linear model for the scheduling controller design. Now, using (1), the estimated requested CPU load is:

$$\hat{U}(kh_S) = \frac{(1-\lambda)q^{-1}}{1-\lambda q^{-1}} \sum_{i=1}^{n} \overline{c}_i(kh_S) f_i(kh_S) \quad (2)$$

where q stands for the shift operator. For the control design, we have chosen to consider a "normalised"

[3]Possible secondary actuators are variants of the control algorithms, with different QoS contributions to the whole system. Such variants should be handled by the scheduling manager working on a discrete events time scale

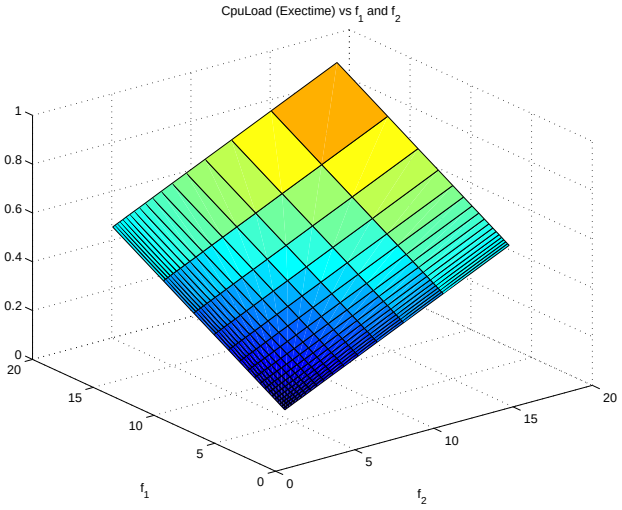

Figure 3: CPU utilisation vs control tasks frequencies

control model (i.e independent on the execution time) as $H(z^{-1}) = \frac{(1-\lambda)z^{-1}}{1-\lambda z^{-1}}$. As illustration, in the two control tasks system presented in the following section, the control scheme is therefore as in figure 4 where the estimated execution-times are used on-line to adapt the gain of the controller for the original CPU system (2).

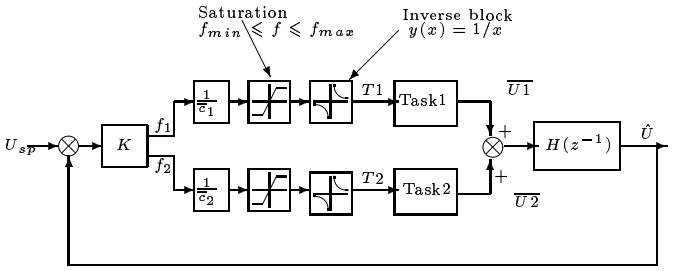

Figure 4: Control scheme for CPU resources

According to this control scheme, the design of the controller K can be made using any advanced control methodology. For the considered application (see section 6), we have chosen the H_∞ control theory which can lead to a robust controller w.r.t modelling errors (see [21] for details on H_∞ control). Moreover it provides good properties in the presence of external disturbance, as it is emphasised in the illustrative example.

5.2 A QoC criterion: robustness w.r.t. delays

In most of computer-controlled systems, the computer must do in each period: sampling of the process output, executing of the control algorithm, sending the new control signal to the process. This implies that the control task is supposed to have a fixed period and that the input-output latency (i.e the control delay) is small and without jitter. If not, this should be considered in the control design. In fact such delays may be of two-fold: first the communication be-

tween the sensor and the controller, and between the controller and the actuator, as well as the computational delay corresponding to the control computation cost. The latter is generally less than a sampling period. However when the control task is preempted by higher priority tasks, this may lead to delays larger than a sampling period. In this framework, these delays may be lumped in a single input delay. Hence, we will consider in the sequel discrete time-delay systems as:

$$\Sigma_d : \quad x(k+1) \quad = \quad Ax(k) + Bu(k-d) \quad (3)$$

where $x(k) \in \mathbb{R}^n$ is the state vector assumed to be measured, $u(k) \in \mathbb{R}^r$ is the control input vector, d is the positive *unknown* delay value, A and B are real matrices of appropriate dimensions. Let us recall that in this formulation, d is the input unknown delay.

Remark 1 *Control theory for linear systems sampled at fixed rates, including fixed and known delays, has been much studied for ten years. Let us cite [2] where an augmentation approach is used for sampling a system with time-delay and obtain an equivalent free delay representation. However if the delay vary the dimension of the non-delayed system will vary accordingly, which is not acceptable. Since the delay is here assumed to be unknown, such an approach cannot be applied. This motivates the need in real-time control, to consider discrete-time representations with time-delay. On the other hand specific methods to time-delay systems could be used to derive a discrete-time representation with delays [7] from a continuous-time one.* □

For systems (3), the following family of state feedback control laws is considered:

$$u(k) = Kx(k) \quad (4)$$

Using the control law (4), the closed-loop system is then:

$$x(k+1) = Ax(k) + BKx(k-d)$$

Since the closed-loop system is a state-delayed system, specific method to study the stability of such systems must be used in order to design the feedback gain K. Two kinds of stability results can be obtained: either delay-independent, or delay-dependent ones. For discrete-time systems let us cite the papers of [9, 12] as they consider systems with unknown delay, with uncertainties, and eventually a disturbance. In the considered framework of real-time control with delays, experiments may allow delay measurements (see for instance [14]). Even if the exact delay values are unknown, it can be possible to estimate a bound on the control delay, i.e. a maximal delay. We then focus here on delay-dependent methods for discrete time-delay systems, that ensure stability for a maximum allowable delay. In [17] an Linear Matrix Inequality approach has been used to design a memoryless delay-dependent state feedback control law of the form (4) that ensures asymptotically stability for the closed-loop system (3-4) for any time-delay d satisfying $0 \leqslant d \leqslant \bar{d}$.

6 Illustrative example

In this part, we describe our methodology for the feedback-scheduling of control tasks in the case of two linearised pendulum systems presented in [6], i.e.

$$\left\{ \begin{array}{lcl} \dot{x}(t) & = & Ax(t) + Bu(t) \\ y(t) & = & Cx(t) \end{array} \right. \quad (5)$$

where $A = \left[\begin{array}{cc} 0 & 1 \\ \omega_0^2 & -2\xi\omega_0 \end{array} \right]$, $B = \left[\begin{array}{c} 0 \\ \omega_0/g \end{array} \right]$ and $C = \left[\begin{array}{cc} 1 & 0 \end{array} \right]$, with $\xi = 0.2$, $g = 9.81$, $\omega_0 = 3.77$ for the first pendulum, $\omega_0 = 4.08$ for the second one. y is the controlled output and x is here assumed to be measured. The corresponding system is the pendulum in the upright position (i.e an unstable open-loop system).

According to the choice of a sampling period h, and including a control delay d, the following discrete model is obtained:

$$x(k+1) \quad = \quad A_h x(k) + B_h u(k-d) \quad (6)$$

where A_h and B_h are the corresponding system matrices.

In this case, the nominal sampling period is $0.1sec$, and is assumed to be changed between $0.02s$ and $0.5s$ by steps of $0.02s$. Note that this is larger than required by the usual criterion for the choice of sampling period [2]: $0.1 \leqslant \omega h \leqslant 0.6$. The corresponding saturation block in figure 4 is then such that $f_{min} = 0.02$ and $f_{max} = 0.5$. To this model, the control delay must be added to lead to the considered system representations (3).

6.1 Process control design

The methodology described in section 5.2 has been used to design delay dependent state feedback control laws for system (6). This methodology is compared to a classical pole placement control law, designed assuming there is no control delay.

As precised in [5], on-line recalculations of the control law are often too costly, particularly here where convex optimisation (LMI) is used. The considered solution is then to calculate off-line the parameters of the controller for a range of sampling periods, and store them in a table, as we have done for the considered range of sampling period, i.e. $[0.02 - 0.5]$ with a step of $0.02s$.

6.2 Feedback scheduling design

The feedback scheduler is here implemented as an application task that runs in parallel with the control

task, with a higher priority. It executes as a periodic task, with a period h_S, larger than the sampling period of the control task, in order not to change two often the sampling period of the control tasks, i.e. the control parameters. We have chosen here $h_S = 2s$.

As precised before in section 5.1, The H_∞ control theory is here applied . Such a control method uses some weighting functions that have to be chosen to satisfy the performance specifications, i.e.here mainly a closed-loop system with a rise time of $4s$ and a module margin higher than 0.5 for robustness. Using the classical methods available in control design softwares, the solution of the H_∞ control problem gives the controller $K = [K_1 \quad K_2]$ with $K_1(z) = K_2(z)$ and:

$$K_1(z) = \frac{-0.1938622 + 0.4519570z + 0.6475011z^2}{0.8412522 - 0.1573669z + z^2}$$

6.3 Simulations

Simulations have been performed using Truetime, a Matlab/Simulink toolbox for real-time control [8]. Here the first pendulum has an higher priority than the second one. On figure 6 the presented results are then only given for this lower priority pendulum. Moreover we assume here that the control delay is half a nominal sampling period, i.e. $0.05s$.[2mm] The scenario used here is the following. At $t = 0$ the control tasks begin. At $t = 2$, the scheduling controller is switched on, and the feedback control of the computing resources is realised around 60% of utilisation (which is the nominal CPU load). At $t = 25s$, a reference step input is sent, representing an decrease of 30% of ressource availability, leading to a increase of the control sampling periods. In both cases (classical and delay dependent control laws) the performances are few affected by this change, due to the adaptation of the control parameters (stored in tables). At $t = 50$, the set-point of resources availability is set back to 60%. At $t = 70s$, a disturbance task, with an higher priority than the control tasks, but for which we cannot measure its execution time, appears. As seen on figure 5, this task implies important preemptions in the lower pendulum control task, leading to an increase of the control delay. As shown in figure 6 the classical pole placement control law, design without account for delay, becomes unstable. On the contrary, the memoryless delay-dependent control law ensures robust stability of the system, despite the presence of unknown and varying control delay. This generates some temporal uncertainties in the execution of the control task These results point out the interest of feedback scheduling, allowing an adaptation of the control law parameters under resource availability, as well as the importance of taking into account the control delay in the design of the process control law, as done in part 5 in the process modelling and control steps. If not the process can become unstable due to such temporal uncertainties.

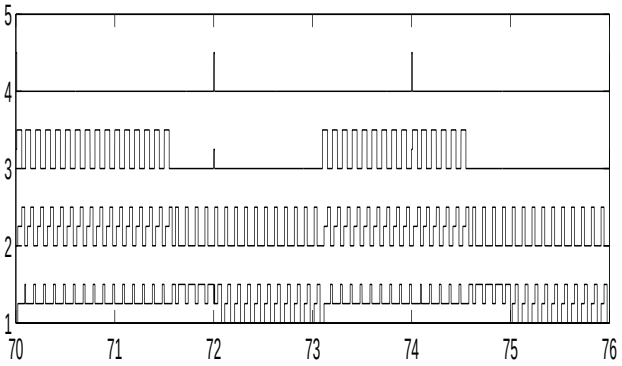

Figure 5: Zoom on process control behaviours

7 Conclusion

In this paper, a new methodology for scheduling-control co-design is proposed. We have provided a new architecture and design method for feedback scheduling. Indeed the control synthesis of the feedback scheduler has been provided using the H_∞ control theory, and the gain of the controller is adapted online using the measured execution times of the control tasks. An integrated control-scheduling approach is proposed, where the state feedback control law has been designed according to a range of sampling periods (for adaptation under computer resources requirements), and taking into account some control delay (mainly due here to the preemption in the controllers). Some simulation results have been given, which emphasises the interest of this approach.

Obviously simulations are not accurate enough to model a real system, and ongoing experiments running on top of an actual RTOS must be further developed to better assess this new approach; in particular the computing overload due to such method must be carefully evaluated and must be integrated in the overall QoS computation. However even simulations show that setting the scheduling controller period, estimating the CPU load and choosing the QoC criterion are not trivial tasks. Finally, besides control related aspects, the role and structure of the scheduling manager must be detailed to efficiently integrate exception handling and control modes in a safe and flexible control system.

References

[1] K.-E. Arzén, A. Cervin, J. Eker, and L. Sha. An introduction to control and scheduling co-design. In *39th IEEE Conference on Decision and Control*, Sydney, Australia, december 2000.

[2] K.J. Astrom and B. Wittenmark. *Computer-Controlled Systems.* Information and systems sciences series. Prentice Hall, New Jersey, 3rd edition, 1997.

[3] A. Cervin. Towards the integration of control and real-time scheduling design. Technical Report Li-

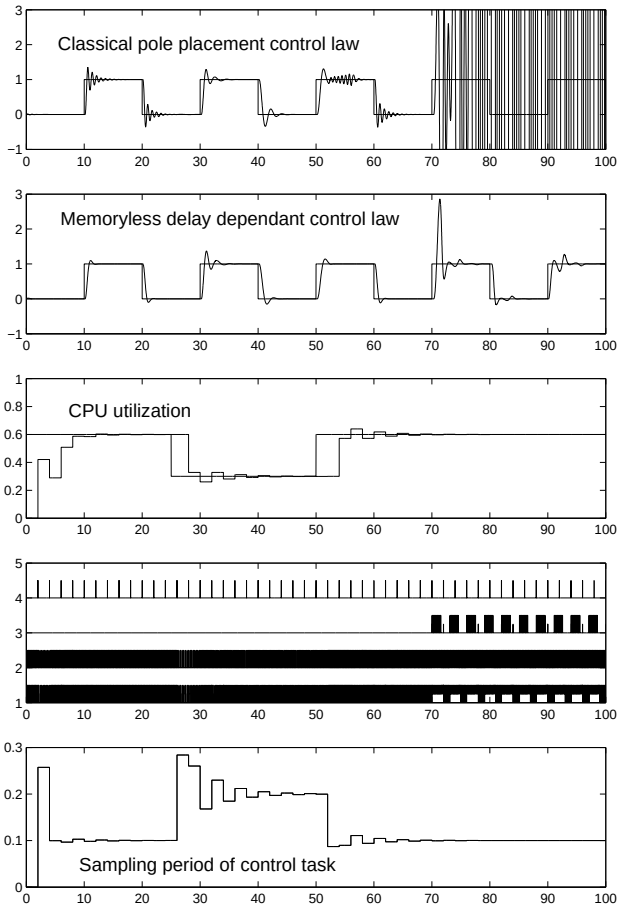

Figure 6: Feedback scheduling and process control behaviours

centiate thesis ISRN LUTFD2/TFRT–3226–SE, Department of Automatic Control, Lund Institute of Technology, Sweden, May 2000.

[4] A. Cervin and J. Eker. Feedback scheduling of contol tasks. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, December 2000.

[5] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén. Feedback-feedforward scheduling of control tasks. *Real Time Systems*, 23(1):25–54, 2002.

[6] J. Eker, P. Hagander, and K-E. Årzén. A feedback scheduler for real-time controller tasks. *Control Engineering Practice*, 8(12):pp 1369–1378, 2000.

[7] A. Fattouh, O. Sename, and J.-M. Dion. Pulse controller design for linear time-delay systems. In *IFAC Symposium on System Structure and Control*, Prague, 2001.

[8] D. Henriksson, A. Cervin, and K.-E. Årzén. Truetime: Simulation of control loops under shared computer resources. In *Proceedings of the 15th IFAC World Congress on Automatic Control*, Barcelona, Spain, July 2002.

[9] Y. S. Lee and W. H. Kwon. Delay-dependent robust stabilization of uncertain discrete-time state-delayed systems. In *IFAC 15th World Congress*, Barcelone, Spain, 2002.

[10] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in hard real-time environment. *Journal of the ACM*, 20(1):40–61, 1973.

[11] C. Lu, J.-A. Stankovic, G. Tao, and S.-H. Son. Feedback control real-time scheduling: Framework, modeling, and algorithms. *Real Time Systems*, 23(1):85–126, 2002.

[12] M. S. Mahmoud. Robust h_∞ control of discrete systems with uncertain parameters and unknown delays. *Automatica*, 36:627–635, 2000.

[13] P. Marti, J. Fuertes, G. Fohler, and K. Ramamritham. Jitter compensation for real-time control systems. In *22nd IEEE Real-Time Systems Symposium*, London, UK, 2001.

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

[15] P. Puschner and A. Burns. Guest editorial : A review of worst-case execution-time analysis. *Real Time Systems*, 18(2-3):pp 115–128, 2000.

[16] M. Ryu, S. Hong, and M. Saksena. Streamlining real-time controller design: from performance specifications to end-to-end timing constraints. In *IEEE Real Time Systems Symposium*, 1997.

[17] O. Sename, D. Simon, and D. Robert. Feedback scheduling for real-time control of systems with communication delays. *submitted to IEEE International Conference on Emerging Technologies and Factory Automation, Lisbon, Portugal*, september 2003.

[18] D. Simon and F. Benattar. Design of real-time periodic control systems through synchronisation and fixed priorities. Technical Report RR4677, INRIA, december 2002.

[19] D. Simon, E. Castillo, and P. Freedman. Design and analysis of synchronization for real-time closed-loop control in robotics. *IEEE Trans. on Control Systems Technology*, 6(4):445–461, july 1998.

[20] H.R. Simpson. Multireader and multiwriter asynchronous communication mechanisms. *IEE Proceedings-Computer and Digital Techniques*, 144(4):241–244, 1997.

[21] S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control: analysis and design.* John Wiley and Sons, 1996. http://www.chembio.ntnu.no/ skoge/.

[22] J. Stankovic, C. Lu, S. H. Son, and G. Tao. The case for feedback control real-time scheduling. In *11th Euromicro Conference on Real-Time Systems*, York, England, 1999.

[23] J.A. Stankovic, M.Spuri, M. Di Natale, and G. Buttazzo. Implications of classical scheduling results for real-time systems. *IEEE Computer*, 28(6):16–25, 1995.