ON REAL-TIME CONTROL TASKS SCHEDULABILITY

P. Marti*, R. Villa*, J.M. Fuertes*, G. Fohler[†]

* Dep. of Automatic Control, Universitat Politècnica de Catalunya
Pau Gargallo 5, 08028 Barcelona, SPAIN
Ph.: 34-3-4011679, Fax: 34-3-4017045, Email: {pmarti,villa,pepf}@esaii.upc.es

[†] Dept. of Computer Engineering, Mälardalen Högskola
PO Box 883, SE-721 23 Västerås, SWEDEN
Ph: +46 21 103158, Fax: +46 21 10 31 10, Email: gerhard.fohler@mdh.se

Keywords: Algorithms and Software for Real-time Control, Scheduling, Sampled Data Systems, Time Varying and Periodic Systems

Abstract

In general, characteristics of classical control theory and properties of real-time scheduling algorithms may cause unexpected control system responses in the implementation of real-time computer-controlled systems. Revising real-time scheduling properties, we analyse which are the main timing problems that current scheduling algorithms may introduce in the execution of control loops. Next, we categorise those timing problems, that is, jitters on task-instances executions, in a control context. Afterwards, we show, by simulations, different types of control system performance degradation that these jitters may cause. Finally, we propose possible solutions that solves this degradation, based on irregular sampling discrete-time system models with varying time delays.

1 Introduction

Many control applications constitute real-time systems due to their strict timing constraints. Therefore, when implementing real-time computer controlled systems, we need to integrate control and real-time disciplines. Nevertheless, [13] has pointed out that there is a gap between the above disciplines due to their different theoretical and practical backgrounds when dealing with real-time control systems

The usual way of building a real-time computer-controlled system differentiates two separate stages: first, control design and then, its computer implementation. This staged procedure can lead to implementations that don't fulfil the stringent timing constraints that control applications require. On one hand, it is known that control theory assumes a highly deterministic timing of an implementation [5]. On the other hand, real-time scheduling algorithms introduce jitters in task-instance execution. This contradictory situation, jitters on task-instances execution vs. deterministic timing needs of an implementation, leads to final implementations that can suffer degradation in the control system performance and even lead to instability in the system.

A starting point for studying computer-controlled systems and for trying to overcome the gap between control and real-time disciplines is analysing the effects of real-time scheduling methods on control systems performance when real-time tasks are executing control loops. To do that, we firstly revise which are the time requirements imposed by control theory that control applications should meet. Then, we examine real-time scheduling properties to underline the main types of jitter that can appear in task- instances executions. Afterwards, by executing control loops through real-time tasks, we categorise these jitters in a control scenario. Next, we show by simulations, the degrading effects of these jitters have in the control performance of real-time computer-controlled systems.

In the final part of this paper, we propose control-based solutions that solve the control degradation introduced by jitters. We present a method based on accepting these jitters in the control design, using irregular sampling discrete time systems models with varying time delays.

This paper is structured as follows: in section 2, we revise relevant antecedents related to our work. In section 3, we give an overview on control loop timing analysis. Section 4 examines which real-time scheduling properties can introduce jitter on task-instance execution and section 5 characterises these jitters in a control scenario. Section 6 shows, using two examples, the effects of the applicability of real-time scheduling methods on the computer-controlled system performance and presents control-based solutions that solve the control degradation caused by jitters. Finally, we present conclusions and future work in section 7.

2 Related work

Control theory and real-time scheduling theory have been relatively independent research areas. Recently, work has been presented in literature, which addresses important issues in real-time control systems. [10] discussessome problems that can be found in real-time control, [3] gives an state-of-the-art on control systems, and scheduling and [13] revises fundamental aspects for implementing real-time control applications in distributed control systems, and studies mono-rate and multi-rate control systems and scheduling issues. [9] outlines the requirements of a real-time scheduling method to be applied in computer-controlled systems.

Specific scheduling-based solutions focusing on the jitter problem and its degrading effects can be found in [2,6]. However, in these works, jitters are not completely eliminated.

Recently, some works have treated deficiencies in the computer system implementation of the control system with respect to time-variations and time-restrictions. [15, 16] investigate the effects of time varying delays on control system stability and performance. It is shown that time varying delays can cause instability and deteriorate performance. In [11, 15] an interpretation of time varying delays as computer induced-disturbances is given.

3 Control loop timing analysis

A control system has basically three main subsystems: sensory system, controller system and actuator system (Figure 1). In a distributed control system, each of these subsystems can be physically divided into separate units.

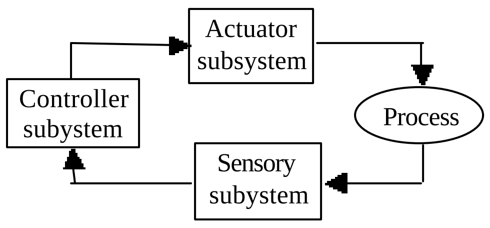

Figure 1. Conceptual control system

The general functionality of control systems can be described as follows: firstly, the sensory system collects data from the process to be controlled. Secondly, the control system, by means of a control law, processes this data and derives the needed action. Finally, the actuator system performs the action on the process. At the design stage, controllers are designed according to process dynamics and control performance specified requirements. Two different design approaches can be used: discrete-time controller design or discretization of a continuous-time controller design.

Discrete-time control theory considers the system through the values of the system inputs and outputs at the sampling instances. To do this, a sampled version of the continuous system model, that will be sampling period dependent, is derived. By doing this, well-known discrete-time system descriptions are obtained [5] and a wide range of discrete-time controller design methods can be applied in order to obtain the desired discrete-time controller.

Discretization of continuous-time design is to design the controller in the continuous time domain, and then, to approximate this design by an implementation (computer-based controller) through fast sampling.

At the end, in both cases, the discrete-time controller is a control computation algorithm to be executed at every sampling period h. Moreover, equidistant sampling and actuation instants are assumed. In addition, the control computation is commonly assumed to be instantaneous.

It has to be pointed out that the resulting controller is characterised by several design parameters that are highly dependent of the sampling period assumed at the design stage.

In this paper, we will focus on a control loop computer-based implementation where the sampling, the control computation and the actuation are performed in the same digital computer by the same task. Within the computer then, we will have a set of tasks sharing the CPU. Some of these tasks are control tasks, performing sampling, control computation and actuation in a sequential way.

4 Real-time basic properties

Real-time computing has been widely used in many areas, playing a key role in technology. There are many definitions of real-time computing, but the main idea [12] is that in such computing systems, the correctness of the system depends not only on the logical results of the computations but also on the time at which the results are produced.

Since the early work on real-time scheduling presented in [8], a wide variety of scheduling algorithms has been presented to schedule real-time tasks. One common feature of almost all those scheduling algorithms is that they introduce jitter on task-instances execution.

Real-time scheduling has provided general-purpose algorithms that mainly use general task models to express timing requirements. Control systems have specific timing requirements that should fit into these task models. Real-time systems usually assume task periods as fixed timing constraints. A real-time task τ_i can be characterised by a fixed period T_i, a deadline D_i and a worst-case computation time C_i. In real-time scheduling, tasks can suffer jitter in their instance execution due to the following reasons:

- After a task τ_i has been released, it has to delay its exection start because other higher priority tasks are executing.
- After a task τ_i has started execution, it can be preempted by other higher priority tasks or can be blocked when trying to acces shared resources other than the CPU.

This means that the start time instants of successive instances of a control task are not equidistant at run time. Neither are its successive finishing time instants. Notice that control systems assume equidistant sampling and actuation instants in a control system implementation.

For real-time computer-controlled systems, as we said in the previuis section, each control task-instance will perform the sampling (S) at the beginning of its execution and it will perform the actuation (A) at the end of its execution, after executin the control computation (C). The tasks model can be seen in figure 2.

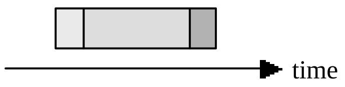

Figure 2. Task model

5 Jitter characterisation

With the presented task model for control tasks, to have not equidistant task-instance start times implies to have irregular sampling. That is, the separation between consecutive sampling instants is not constant; it varies form task-instance execution to task-instance execution. We call this variability in the sampling instants *sampling jitter* and denote it by h_k. Therefore, h_k is the sampling interval of the *k*-instance of a given control task.

In addition, to have not equidistant task-instance finishing times implies to have irregular actuation. That is, the separation between consecutive actuation instants is not constant, which also implies that the separation between sampling instants and actuation instants vary form task-instance execution to task-instance execution. We call this later variability *sampling-actuationdelays* and we denote it by t_k. Therefore, t_k is the sampling-actuation delay of the *k*-instance of a given control task.

To illustrate this jitter characterisation, assume a control task that is scheduled with another task using Rate Monotonic priority assignment [Liu]. The task set parameters are described in table 1.

	T_i	C_i	Description
τ_2	7	2	Control task
τ_1	4	2	Task

Table 1. Task set parameters

The set of tasks complies the RM scheuldabiltiy test:

$$U = \sum_{i=1}^{2} \frac{C_i}{T_i} = \frac{2}{4} + \frac{2}{7} \approx 0.78 < 2(2^{1/2} - 1) \approx 0.83$$

Suppose that the control task period has been chosen after the appropriate control analysis (T_2=7=h) and its deadline is supposed to be equal to its period. In addition, its computation time is 2, which means that sampling-actuation delays should be constant at run time.

The resulting schedule, over the hyperperiod, can be seen in Figure 3. Notice that some control task-instances have sampling periods longer or shorter than 7. For example, control task-instance starting at t=14 will have a sampling period of 8 (h_3) and control task-instance starting at t=2 will have a period of 5 (h_1). In addition, the sampling-actuation delay, that is supposed to be 2 for all task-instances, also varies from control task-instance execution to control task-instance execution. For example, control task-instance starting at t=7 will have a sampling-actuation delay of 4 (τ_2).

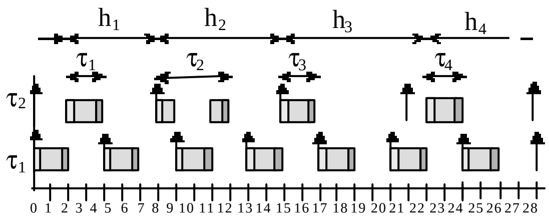

Figure 3. Schedule with sampling jitter and varying sampling-actuationdelays

6 Schedulability and control

In this section, we will show the possible effects that sampling jitter and varying sampling-actuation delays can cause on the control system performance. We will show by simulations, that the jitter presence in control tasks instances execution will degrade the performance of system, even causing a critical failure of the system. We will also show control-based solutions that solve the control performance degradation caused by jitters.

To do this, we will use two separate examples, the DC servo problem, controlled by a discretization of a continuous-time designed PID controller, and the inverted pendulum problem, controller by a state feedback controller obtained using pole placement observer design.

Firstly, we consider the servo problem, where the major goal is following the command signal. Consider the PID control of a DC servo described by the following continuous-time transfer function:

$$G(s) = \frac{1000}{s(0.5 \cdot s + 1)}$$

Following the specified requirements to have a percentage overshoot less that 15%, the PID parameters are tuned at the following values $K_p = 1.8$, $K_i = 0.1$ and $K_d = 0.09$. Once the PID controller has been designed in the continuous-time domain and with the appropriate sampling period (h=2ms), we obtained its PID discrete approximation:

$$e(t) = r(t) - y(t)$$
$$p(t) = K_p e(t)$$
$$i(t) = i(t-h) + \frac{K_i h}{2}(e(t) + e(t-h))$$
$$d(t) = \frac{K_d}{h}(e(t) - e(t-h))$$
$$u(t) = p(t) + i(t) + d(t)$$

We implement the obtained controller as a PID control task (with sampling period 2ms and execution time 0.2ms), which we will schedule.

As a second example we consider the inverted pendulum problem, where the major goal is to maintain the desired vertical position of the inverted pendulum. The inverted pendulum process is given by the following simplified continuous-time state-space description:

$$\frac{dx}{dt} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot u(t)$$
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot x(t)$$

Following the closed loop requirements and with the appropriate sampling period, we designed a discrete-time state-feedback control law using pole placement observer design. We implement the obtained controller as a state feedback control task, which we will schedule.

In the following, simulations will be done using the real-time control systems simulator presented in [7].

6.1 Single task schedulability impact in control

In this section, we are going to study the schedulability effects on control performance if the PID control task or the state feedback control task are executed in isolation on a CPU.

In both examples, the respective control task is executing with equidistant sampling and actuation instants, without any jitter. Therefore, in this case, no degrading effects appear in the system response

The square pulse response of the DC servo controller by the PID task can be seen in figure 4, fulfilling the specified requirements.

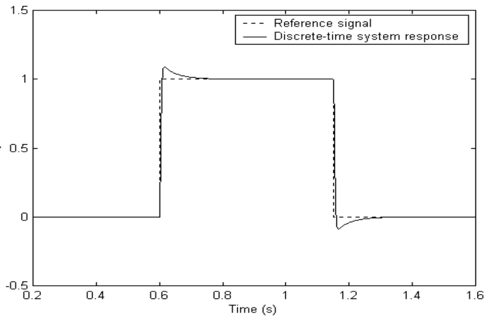

Figure 4. DC servo discrete-time system response

The closed loop step transient response of the inverted pendulum controlled by the state feedback control task can be seen in figure 5, fulfilling the control requirements.

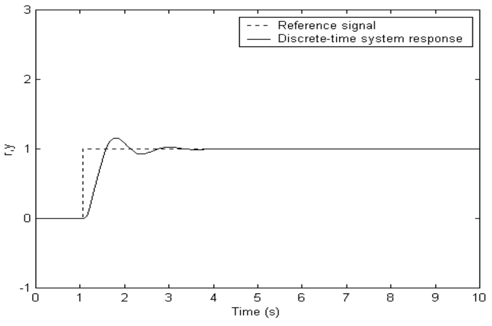

Figure 5. Inverted pendulum discrete-time system response

6.2 Multiple task schedulability impact in control

Now we are going to study the schedulability effects on the control performance if the PID control task or the state feedback control task has to share the CPU with other tasks. We use RM as a scheduling algorithm and before running the system, we have verified the schedules feasibility.

Figure 6 shows the effects of multiple task schedulability on the DC servo system performance.

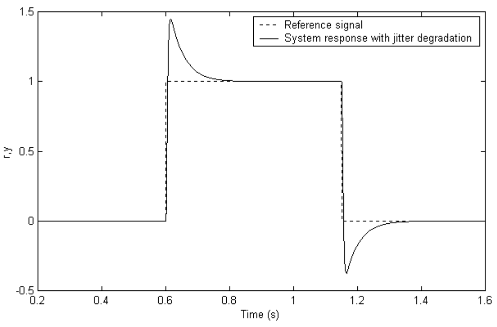

Figure 6. DC servo response with jitter degradation

The system response, which was to track the square pulse input, suffers important degradation, which drives the system response out of the specified requirements. This degradation is caused by two reasons. First, control actions are calculated using the PID algorithm, where the parameter h is constant (2ms). However, due to sampling jitter, at run time, sampling intervals for each PID control task-instance are not constant anymore. They vary from 1.5ms to 2.5 ms. Second, sampling-actuation delays were supposed to be constant (0.2ms). However, at run time, they also vary.

Similarly, figure 7 shows the effects of multiple task schedulability on the inverted pendulum system performance.

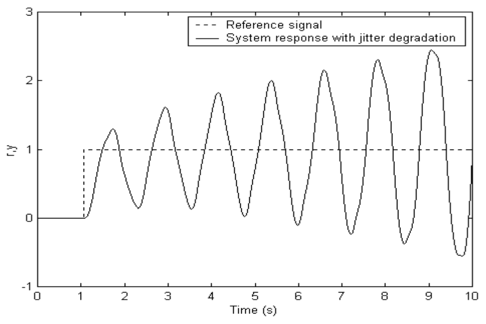

Figure 7. Inverted pendulum response with jitter degradation

The system response, which was to keep the vertical position of the inverted pendulum, goes to instability. This degradation is cause again due to sampling jitter and sampling-actuation delays.

A preliminary conclusion it that in the implementation of computer-controlled systems, real-time scheduling algorithms and classical control theory cannot be developed separately because unexpected system performance may occur. Two main solutions can be adopted: to use specific scheduling-based algorithms to minimise jitters, or to accept these jitters in the controller design and to compensate for the degradation they can introduce. We adopt the later solution, that is, to accept jitters that the schedule introduces and to compensate for it at run time with the appropriate control law implementation, in what we call the *compensation approach*.

6.3 Jitter aware control

The main idea behind the compensation approach, that was suggested in [14,1,4], is to compensate for the degradation on the control system response due to variations from sample to sample. What we propose is to adopt this technique for both jitters, that is, to adjust at runtime the controller parameters at each control task-instance execution to account for both sampling jitter and sampling-actuation delays. Therefore, at run time, controller parameters must be updated at each control task-instance execution according to the actual sampling jitter (h_k) and the sampling-actuation delay (τ_k). To do that, for state space models, we can use the following irregular sampling discrete time system models with varying time delays:

$$x(\bar{h}_{k+1}) = \Phi(h_k)x(\bar{h}_k) + \Gamma_0(h_k, t_k)u(\bar{h}_k) + \Gamma_1(h_k, t_k)u(\bar{h}_{k-1})$$
$$y(\bar{h}_k) = Cx(\bar{h}_k) + Du(\bar{h}_k)$$
$$u(\bar{h}_k) = -L(h_k)x(\bar{h}_k)$$
$$\bar{h}_k = \sum_0^k h_k$$

where $\Phi(h_k) = e^{Ah_k}$
$$\Gamma_0(h_k, t_k) = \int_0^{h_k - t_k} e^{As} ds B$$
$$\Gamma_1(h_k, t_k) = e^{A(h_k - t_k)} \int_0^{t_k} e^{As} ds B$$

Applying this technique to the DC servo problem, we can see, in figure 8, the system response if the compensation approach is used in the PID control task. It is clear that in the system response performance the previous degradation due to jitters is eliminated. This elimination is due to the fact that in this case, at each control task-instance execution, control actions are calculated according to actual jitters.

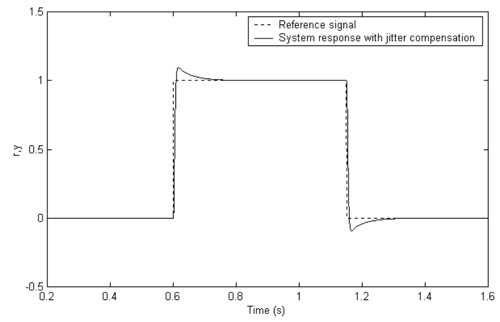

Figure 8. DC servo response with jitter compensation

Applying this technique to the inverted pendulum problem, we can also see (Figure 9) that in the system response, if the compensation approach is used in the state feedback control task, the jitter degradation is eliminated.

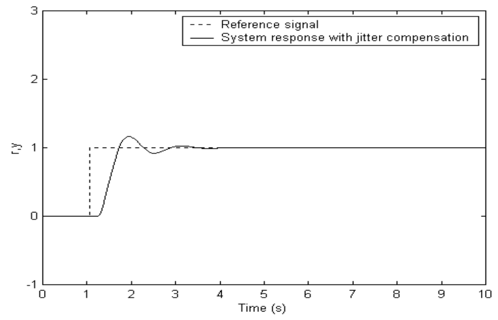

Fig.9. Inverted pendulum response with jitter compensation

As a conclusion, simulations results show the effectiveness of this approach. However, a deeper control formalisation of the models we use is needed, as well as a stability analysis.

7 Conclusions

In this paper we have identified a gap between real-time systems and control systems. To deal with computer-controlled systems that need real-time computing, an integrated approach must be done in order to avoid degraded system performance.

We have characterised the real-time scheduling jitters in a control scenario. We have seen the degrading effects of task schedulability in the control system response. Moreover, we have proposed control-based solutions that solve this degradation by designing controllers that readjust its parameters at each execution according to actual jitters.

Future work will focus on the formalisation of the compensation approach and stability analysis. In addition, its full applicability in current real-time scheduling algorithms has to be analysed in detail.

Acknowledgements

The presented research has received support from Spanish CICYT project ref. DPI2000-1760-C03-01.

References

[1] P. Albertos and J. Salt. “Digital Regulators Redesign with Irregular Sampling”, *11th IFAC World Congress* (Preprints), vol 8, pp 157-161 (1990)

[2] P. Albertos, A. Crespo, I. Ripoll, M. Vallés and P. Balbastre. “RT control scheduling to reduce control performance degrading”, *39th IEEE Conf. on Decision and Control*. Sydney, Australia, December (2000)

[3] K-E. Årzen, B. Bernhardsson, J. Eker, A. Cervin, K. Nilsson, P. Persson and L. Sha. “Integrated Control and Scheduling”, Research report ISSN 0820-5316. Dept. Automatic Control, Lund Institute of Technology (1999)

[4] K.-E. Årzen, A. Cervin, J. Eker and L. Sha. “An Introduction to Control and Scheduling Co-Design”, *39th IEEE Conference on Decision and Control*, Sydne*y*, Australia, December (2000)

[5] K. J. Åström and B. Wittenmark. *Computer-Controlled Systems. Third edition*. Prentice Hall. (1997)

[6] A. Cervin. “Improved Scheduling of Control Tasks”, *Proceedings of the 11th Euromicro Conference on Real-Time Systems*, York, England, June (1999)

[7] J. Eker and A. Cervin. “A Matlab Toolbox for Real-Time and Control Systems Co-Design”, *in Proc. 6th Int. Conference on Real-Time Computing Systems and Applications*, Hong Kong, China, December (1999)

[8] C. Liu and J. Layland. “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment”, *J.ACM*, **20**, 46-61. (1973)

[9] P. Marti, R. Villa, J.M. Fuertes and G. Fohler. “Real Time Methods Requirements in Distributed Control Systems”, *Procc. 25th IFAC Workshop on Real-Time Programming*, Spain, 101-108 (2000)

[10] J. Nilsson. “Some Topics in Real-Time Control”. *ACC*, June 24-26. Philadelphia. (1998)

[11] K. Shin and X. Cui. “Computing Time Delay and its Effects on Real-Time Control Systems”, *IEEE Transactions on Control Systems Technology.* **Vol. 3**, N.2, p.218-224 (1996)

[12] J.A. Stankovic. “Misconceptions About Real-Time Computing: A Serious Problem for Next Generation Systems”, *IEEE Computer*. **21**(10):10-19 (1988).

[13] M. Törngren. “Fundamentals of Implementing Real-time Control Applications in Distributed Computer Systems”, *J. of Real-Time Systems*, **14**, 219-260, Kluwer Academic Publishers. (1998)

[14] B. Wittenmark and K.J. Åström. "Simple Self-tuning Controllers". In Unbehauen, Ed. Methods and Applications in Adaptive Control, number 24 in Lecture Notes in Control nd Information Sciences, pp 21-29. Springer-Verlag, Berlin, 1980

[15] B. Wittenmark, J. Nilsson and M. Törngren. “Timing Problems in Real-Time Control Systems: Problem Formulation”, *Proc. Of the American Control Conference*, Seattle, Washington. (1995)

[16] B. Wittenmark, B. Bastian and J. Nilsson “Analysis of Time Delays in Sinchronous and Asynchronous Control Loops”. *Proc. Of the 37th Conf. On Decision andControl*, Tampa, FL. US. (1998)