Jitter Evaluation of Real-Time Control Systems

Lluesma Camps, Manuel; Cervin, Anton; Balbastre, Patricia; Ripoll, Ismael; Crespo, Alfons

2006

Jitter Evaluation of Real-Time Control Systems*

Manuel Lluesma†, Anton Cervin‡, Patricia Balbastre†, Ismael Ripoll† and Alfons Crespo†
†Technical University of Valencia, Spain
‡Lund University, Sweden

Abstract

The real-time implementation of a controller typically introduces artefacts like delay and jitters that have not been considered at the design stage. As a consequence, the system behaves in a non-periodic manner, and the real performance is degraded with respect to the expected response. This paper proposes a hybrid task model to reduce the impact of the scheduling on the control performance. For a large batch of typical plants, we analyze how sensitive the control system is to jitter when the sampling rate is slow or fast compared to the bandwidth of the system.

1. Introduction

Industrial control applications are usually developed in two distinct phases: control design and real-time implementation. In the control design stage a regulator is obtained, which is later translated into a control task in the implementation phase. Traditionally, these two phases have been carried out in relative isolation. In the control community, the real-time implementation is often dismissed as a non-issue. However, a real implementation will always introduce delay and jitter at various points in the control loop. These delays cause the system to behave in a non-periodic manner, causing the real performance to be degraded compared to the expected response.

To remedy this problem, some works have pointed out the necessity of integrating the control design and the scheduling design [11, 5, 2, 3, 9]. This integration allows, among other things, the reduction of the negative influence of the computational platform.

In this work, we evaluate the ability of various task models to reduce the negative influence of the computational platform. Also, a new hybrid task model is proposed in order to reduce the amount of delay and jitter even further. The control performance is evaluated using task set simulation followed by stochastic control analysis. We investigate how the sensitivity to jitter depends on the sampling period in relation to the bandwidth of the closed-loop system. The experimental evaluation is carried out on a test batch with 27 processes, classified into stable, marginally stable, and unstable plants.

*This work is partially supported by EU/FP6/ARTIST2

2. Task models to reduce control jitter

Several works in the real-time literature deal with the problem of jitter in control systems. Two different approaches have been taken: to compensate for the jitter or to try to minimize it. In the latter case, one of the solutions from the scheduling point of view is to change the task model, raising the priority of the task for which we want to minimize the jitter. Two of these works are the CO_US task model [3] and the IMF task model [5]. In this section, these two models are outlined, and a hybrid model, called ICOFU, is proposed.

Let $\mathcal{T} = \{T_1, ..., T_n\}$ be a periodic task system. Each control task is described by the parameters $T_i = (C_i, D_i, P_i, O_i)$, were P_i is the period, D_i is the relative deadline, C_i is the worst-case execution time ($C_i \leq D_i$), and O_i is the relase offset. Preemptive, uniprocessor earliest deadline first (EDF) [8] scheduling is assumed throughout.

2.1. CO_US and IMF models

In order to minimize input-output delay and jitter, Cervin [3] proposes the following subtask partitioning. Each control task is split into two subtasks: *Calculate Output* (CO) and *Update State* (US). Since the CO subtask is more time-critical, it is assigned a shorter deadline than the US subtask. This model will be called CO_US.

To reduce output jitter and input jitter, the IMF (Initial, Mandatory and Final) task model is proposed in [5] for fixed-priority schedulers and in [2] for dynamic-priority scheduling. In this model, each control task T_i is split into three subtasks: T_{ii} corresponds to the data acquisition subtask (Initial task), T_{im} corresponds to the control computation (including the state update) (Mandatory task) and

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

T_{if} corresponds to the control action delivery subtask (Final task). As final tasks are often more critical (since they correspond to the control action delivery), this proposal assigns highest priority to final tasks, medium priority to initial tasks, and the lowest priority to mandatory tasks.

2.2. ICOFU model

Since both models achieve good results, we propose a hybrid model, taking advantage of both task models. The main idea is to split the mandatory subtask of the IMF model into a *calculate output* subtask and an *update state* subtask. The initial and final tasks remain the same as in the original IMF model. Taking into account that the update state task can be executed after the final task (and even after the initial task of the next iteration) the task parameters of the ICOFU model are

$$T_{ii} = (C_{ii}, D_{ii}, P_i, 0) \quad T_{ico} = (C_{ico}, D_i, P_i, 0)$$

$$T_{if} = (C_{if}, D_{if}, P_i, O_{if}) \quad T_{iu} = (C_{iu}, 2D_i - \epsilon, P_i, 0)$$

where ϵ is an arbitrarily small positive number.

We have relaxed the deadline of the update state subtask, but to a certain limit. Indeed, activation k of task T_{iu} must finish before activation $k+1$ of task T_{ico}. Therefore, we can assign the deadline to the update state task as $D_{iu} = 2D_i - \epsilon$. Deadlines for initial and final tasks are calculated as in the IMF model, that is, iterativelly assigning the task deadline as the worst case response time of the task.

It is also necessary to assign an offset to the final task, $O_{if} = WCRT_{im} - C_{jf}$. Here, $WCRT_{im}$ is the worst case response time of the mandatory subtask.

As stated in [10] an asynchronous task set is feasible if the corresponding synchronous task set is feasible. In an asynchronous task set, tasks are allowed to have a start time different from zero (O_i). The corresponding synchronous task set has the same timing characteristics except the offset which is reset to zero ($O_i = 0$). We will use this result to check the feasibility of the ICOFU model.

Let $\mathcal{T}_{icofu}$=$\{T_{1i}$,T_{1co},T_{1f},T_{1u}, ..., T_{ni},T_{nco},T_{nf},$T_{nu}\}$ be the ICOFU task set, and let $\mathcal{T}'_{icofu}$ be the corresponding synchronous task set. Then, by [10], if $\mathcal{T}'_{icofu}$ is feasible then $\mathcal{T}_{icofu}$ is also feasible. This is a pessimistic test in the sense that some task sets may be feasible even when the corresponding synchronous task set is unfeasible.

3. Experimental set-up

We consider a system where three independent periodic tasks should be implemented in a computer with limited computational resources. Task periods are 20, 29, 35 while computation times for subtasks are: CO: 2–6; US: 3–6; I: 0.5 ;F: 0.5.

The simulated system consists of three independent plants with different initial parameters. The system is controlled by a computer with limited computational resources. So, a linear digital controller is designed for each plant. The three plants are implemented as real-time tasks such that the overall control performance is optimised. Figure 1 shows the system characterisation using the TrueTime [7] toolbox.

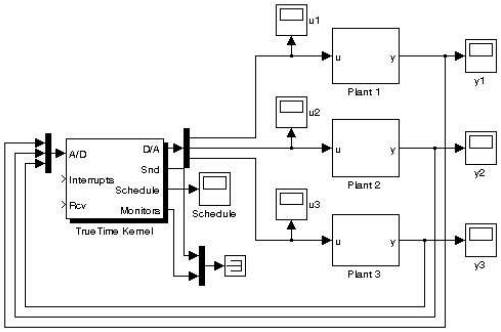

Figure 1. TrueTime model to control multiple plants

The main goal is to analyse the behaviour of the plants from a cost function point of view, studying how variations in sampling latency and input-output lantency affect the system's performance. In this case, the *cost* of the system is specified as (1)

$$J_c = \lim_{T\to\infty} \frac{1}{T} \int_0^T (y^2(t) + \rho u^2(t))dt \qquad (1)$$

where ρ is a weight, u is the inner control signal and y is the plant response. This cost function could in theory be evaluated numerically using very long simulations with TrueTime. A better alternative is to use the Jitterbug [4] toolbox, where the cost function can be computed analytically without losing much accuracy.

The timing model of each plant consists of three nodes. The first node is periodic and represents the release of the control task. There is a random delay τ_1 until the second node where sampling operation *H1(z)* is updated, and another random delay τ_2 where the computation and actuation of the control signal *H2(z)* is updated. While τ_1 represents the input sampling (L_s), the input-output latency of the system is given by $L_{io} = \tau_1 + \tau_2$. Thus, L_s and L_{io} are not independent variables and can never reach their maximum values simultaneously. So, τ_2 is a matrix delay distribution which models the dependency between L_s and L_{io} (this ensures that L_{io} never exceeds the period).

In order to evaluate the CO_US, IMF and ICOFU task models, a plant test batch for real-time control design and performance evaluation has been designed. This batch is inspired by [6], where a test batch for process control and PID

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

tunning was presented. Compared to their batch, plants with excessive dead-times have not been considered, and some more difficult-to-control resonant and unstable plants have been added. There are 27 plants in total (see Apendix A). To account for process disturbances, measurement noise and fixed delays in the control design, the LQG (Linear-Quadratic-Gaussian) control design methods has been used. This regulator consists of an optimal state-feedback gain and a Kalman state estimator.

This controller has been designed for each plant using the Jitterbug [4] command `lqgdesign`, which provides a discrete-time LQG controller for a continuous time plant G. It is assumed a constant latency equal to the fixed delay which has to be compensated (each task's model has a different minimum fixed delay). The LQG design method minimizes the cost function (1). Typically, a controller can be executed periodically over a range of sampling periods given that the control system performs well. In this way, the sampling interval could be chosen according to the rule of thumb $0.2 < \omega_b P < 0.6$ presented in [1]. Here, ω_b is the bandwidth of the closed-loop system (smaller values produce jitter reduction but higher utilisation). New books on sample-data control recommend sampling even faster, so the interval $0.1 < \omega_b P < 1$ is evaluated.

4. Results

The main goal is to evaluate how sensitive a plant is to jitter when the system is forced to sample very slowly or very fast compared to the bandwidth of the system under different task models. The plants in the test batch are classified into stable, unstable and marginally stable systems. Results have been obtained for different sampling periods according to the rule of thumb defined before.

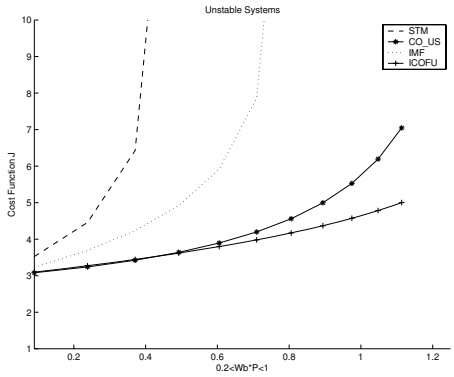

Figure 2. Cost evaluation. Unstable Systems

Figure 2 shows the average cost evaluation when the system consists of three unstable plants. With regard to sampling period P, when it is increased (slow sampling - product $\omega_b P$ near or slightly higher than 1) the system performance becomes degraded (higher J_c) for all task models. This is expected by control designers, since, in general, slow sampling causes worse performance. The ICOFU model achieves the best performance for most sampling periods. Furthermore, it presents a near-linear behaviour and the slower the sampling, the better the response with respect to the CO_US model. STM and IMF don't work quite as well (when the sampling period is increased only a bit) and although the IMF model behaves better than STM the final performance is not that good in any of them. As shown in Figure 2, when $\omega_b P \approxeq 0.4$ and $\omega_b P \approxeq 0.8$ for STM and IMF respectively, the cost J_c goes to ∞. This means that the process cannot be controlled under those models.

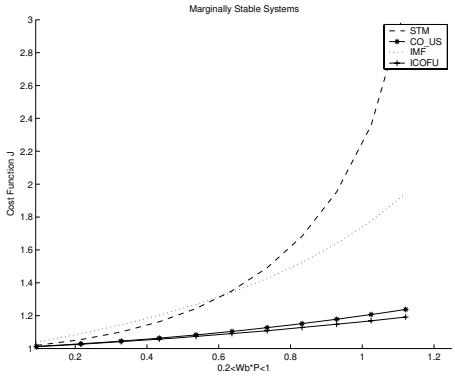

Figure 3. Cost evaluation. Marginally Stable Systems

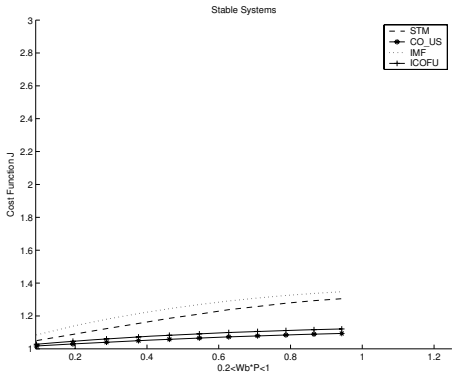

Figure 4. Cost evaluation. Stable Systems

Figures 3 and 4 show the average cost evaluation for stable and marginally stable plants respectively when delay transitions are fixed to the $<L_s, L_{io}>$ generated for the longest-period task (it is the task which suffers more interferences from other tasks and therefore, it has the most pessimistic delay distributions). Since there is a slightly bet-

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

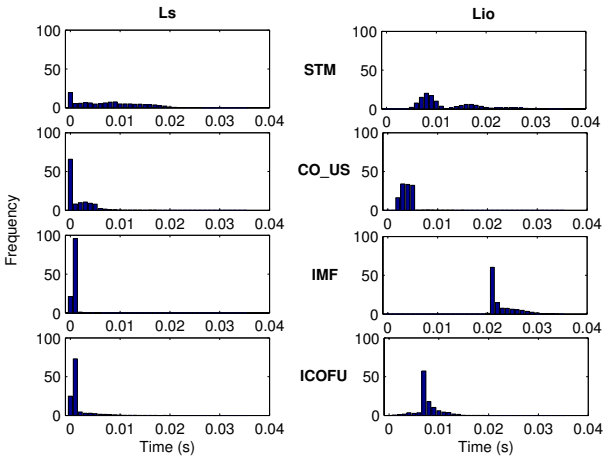

Figure 5. L_s and L_{io} distributions for STM, CO_US, IMF and ICOFU models

ter difference between ICOFU and CO_US with respect to STM and IMF, in stable plants all the analyzed control systems achieve stability quickly. With regard to marginally stable plants, ICOFU and CO_US present a near-linear behaviour too, whereas STM and IMF degrade the system's performance when the sampling period is increased.

Finally, Figure 5 shows L_s and L_{io} distributions for each model. As is shown, the STM model gives the highest sampling jitter and input-output jitter, which causes the bad performance of unstable plants. In order to reduce those variabilities, IMF proposes a subtask partitioning and introduces a fixed delay (Offset). Even though sampling and actuation are done periodically (at 0.5 and 21.5 ms respectively) the fixed delay is very large and therefore performance of the unstable control systems is degraded, even if the delay is compensated by the controller. Fixed delay is reduced to 7.5ms when using the ICOFU model and the jitter is kept very low, achieving the best results. On the other hand, the CO_US model sends the control action as soon as possible but with a higher input-output jitter again negatively affecting the performance of unstable systems (when the sampling period is increased).

5. Conclusions

In this paper two task models (CO_US and IMF) that reduce control jitter and latencies have been evaluated. Furthermore, a new task model called ICOFU, which combines the advantages of both CO_US and IMF task models, has been proposed and analyzed. The evaluation of these three task's models has been made for a large number of different control systems. In conclusion, ICOFU and CO_US models are more suitable for controlling both unstable and marginally plants. This is due to the fact that those systems are more sensitive to jitter (to a lesser extent in marginally stable plants). Furthermore, ICOFU model takes the lead against CO_US when the sampling rate is slow compared to the bandwidth of the system. So, decisions made in the real-time design affect the control design, and vice versa. For instance, the choice of scheduling polices influences the latency distributions in the control loop. Therefore, this should be taken into account in the control design.

References

[1] K. Astrom and B. Wittenmark. *Computer-Controlled Systems*. Prentice Hall, 1997.

[2] P. Balbastre, I. Ripoll, J. Vidal, and A. Crespo. A task model to reduce control delays. *The Journal of Real-time Systems*, 27(3), September 2004.

[3] A. Cervin. Improved scheduling of control tasks. In *Proceedings of the 11th Euromicro Conference on Real-Time Systems*, 1999.

[4] A. Cervin and B. Lincoln. Jitterbug 1.1-reference manual. Tech. report, Lund Institute of Technology, Sweden, 2003.

[5] A. Crespo, I. Ripoll, and P. Albertos. Reducing delays in rt control: the control action interval. *IFAC World Congress Beijing*, 1999.

[6] T. Hagglund and K. J. Astrom. Revisiting the ziegler-nichols step response method for pid control. *Journal of Process Control*, pages 635–650, 2004.

[7] D. Henriksson and A. Cervin. Truetime 1.13-ref manual. Tech. report, Lund Institute of Technology, Sweden, 2003.

[8] C. Liu and J.W.Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *JACM*, 23:46–68, 1973.

[9] P. Martí, J. M. Fuertes, and G. Fohler. Jitter compensation for real-time control systems. In *IEEE RTSS*, 2001.

[10] I. Ripoll, A. Crespo, and A. Mok. Improvement in feasibility testing for real-time tasks. *Journal of Real-Time Systems*, 11:19–40, 1996.

[11] D. Seto, J. Lehoczky, and L. Sha. Task period selection and schedulability in real-time systems. In *IEEE RTSS*, 1998.

A. Plant batch definition

$P_1(s) = \frac{\omega^2}{s^2-\omega^2}$ $w = 9.9, 7.0, 5.7$; $P_2(s) = \frac{1}{s^2}$ $n = 1, 2, 3$

$P_3(s) = \frac{1}{(s+1)^n}$ $n = 3, 4, 5$; $P_4(s) = \frac{1-\alpha s}{(s+1)}$ $\alpha = 0.1, 0.3, 0.5$

$P_5(s) = \frac{1}{(s+1)(1+sT)}$ $T = 0.01, 0.1, 0.2, 0.5$

$P_6(s) = \frac{1}{(1+s)(1+\alpha s)(1+\alpha^2 s)(1+\alpha^3 s)}$ $\alpha = 0.1, 0.5, 0.9$

$P_7(s) = \frac{1}{s^2+2\zeta s+1}$ $\zeta = 0.7, 0.4, 0.1, 0$

$P_8(s) = \frac{1}{s(s^2+2\zeta s+1)}$ $\zeta = 0.7, 0.4, 0.1, 0$

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE