Proc. of the IEEE Conf. on Real-Time and Embedded Computing Systems and Applications (RTCSA'04), Gothenburg, Sweden, August 25-27, 2004.

The Jitter Margin and Its Application in the Design of Real-Time Control Systems

Anton Cervin[1], Bo Lincoln[2], Johan Eker[3], Karl-Erik Årzén[4], Giorgio Buttazzo[1]

[1] Department of Computer Engineering and Systems Science, University of Pavia, Italy
[2] Glaze AB, Limhamn, Sweden
[3] Ericsson Mobile Platforms AB, Lund, Sweden
[4] Department of Automatic Control, Lund Institute of Technology, Sweden
`anton@control.lth.se`

Abstract. Based on recent advances in control theory, we propose the notion of *jitter margin* for periodic control tasks. The jitter margin is defined as a function of the amount of constant delay in the control loop, and it describes how much additional time-varying delay can be tolerated before the loop goes unstable. Combined with scheduling theory, the jitter margin can be used to guarantee the stability and performance of the controller in the target system. It can also be used as a tool for assigning meaningful deadlines to control tasks. We discuss the need for best-case response-time analysis in this context, and propose a simple lower bound under EDF scheduling. Finally, a control-scheduling codesign procedure is given, where periods are assigned iteratively to yield the same relative performance degradation for each control task.

1 Introduction

1.1 Background and Motivation

In classical feedback control theory (e.g., [1]), notions such as *phase margin* and *gain margin* are used to describe how sensitive a control loop is towards various uncertainties in the plant. Nonnegative margins are required to ensure the stability of the closed-loop system. The margins are also used as practical stability measures, and there are various rules of thumb associated with them. For instance, it is typically recommended to have a phase margin of at least 30°–45° to ensure some degree of robustness and performance of the system.

When a controller is implemented as a task in a real-time system, a new kind of uncertainty is introduced—an *implementation uncertainty*. In this paper, we will focus on the specific problem of *output jitter*. Variability in the task execution time and preemption from other tasks can cause the controller to experience a different amount of input-output delay in each period. It is well known that such a jitter can degrade the control performance and in extreme cases even cause instability of the control loop (e.g., [2]). Although the present paper only considers jitter due to CPU scheduling, some of the results also carry over to networked control systems, where jitter due to variable transmission times is a major issue.

The majority of previous work on jitter in real-time control systems has focused on either scheduling theory or control theory. In the few instances where an integrated approach has been taken, the control analysis has been somewhat underdeveloped. By contrast, our analysis yields hard results and should hence be applicable to a wide range of systems, including safety-critical applications.

1.2 Contributions

Recently, a new stability theorem for control loops with time-varying input-output delays has been developed [3]. Based on this theorem, we propose the notion of *jitter margin* for control tasks. The jitter margin can be combined with real-time scheduling theory to guarantee the stability and performance of the controller in the target system. The jitter margin can also be used as a tool for assigning meaningful deadlines to control tasks.

It is noted that the jitter analysis can be improved if *best-case response times*, as well as worst-case response times, can be computed. For this purpose, we propose a lower bound on the best-case response time under EDF scheduling, where no such results are known to exist.

When designing a real-time control system, information about the task timing is needed in the control design, and information about the controller timing sensitivity is needed in the real-time design. Based on this insight, we propose an iterative control–scheduling codesign procedure, where the jitter margin is used as a central tool.

1.3 Outline

This paper is outlined as follows. In Section 2, the assumptions are given, and the jitter margin is defined. Its properties are discussed, and the problem of assigning control task deadlines is treated. In Section 3, we discuss jitter analysis under fixed-priority and EDF scheduling, and provide a simple but efficient lower bound on the minimum response time under EDF. In Section 4, a codesign procedure is proposed, where the goal is to implement a set of controllers such that they experience the same amount of performance degradation in the target system. A design example is given, in which the results under rate-monotonic and EDF scheduling are compared. Section 5 provides an overview of related work. Finally, in Section 6, the conclusions are given and future work is discussed.

2 The Jitter Margin

2.1 Preliminaries

Computer-controlled systems (e.g., [4]) are typically designed assuming periodic sampling and either zero or a constant computational delay. A real implementation, however, will introduce jitter at various points in the control loop.

In this paper, for analysis purposes, we will assume that the sampling is jitter-free, while the input-output delay may be time-varying. Jitter-free sampling can be achieved by programming the A-D converter to take samples periodically, or by requesting the A-D conversion when the control task is released.

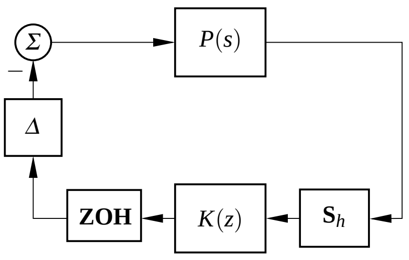


Fig. 1. Computer-controlled system with continuous-time plant $P(s)$, periodic sampler $\mathbf{S}_h$, discrete-time controller $K(z)$, zero-order hold, and time-varying delay Δ.

The control loop assumed in this paper is shown in Figure 1. The plant is described by the linear continuous-time system $P(s)$, and the plant output is sampled with the constant interval h. The controller is described by the linear discrete-time system $K(z)$. Following the zero-order hold, there is a time-varying delay Δ before the control signal is applied to the input of the plant.

Exact stability analysis of the closed-loop system is trivial if the delay Δ is either constant or varying according to a known, periodic pattern. If the delay varies randomly among a set of known delays, Lyapunov theory can be used to verify the stability of the closed-loop system. For freely time-varying delays, the analysis is considerably more difficult. The following theorem from [3] is only sufficient, but it guarantees stability for *any* delays in a given interval, including constant, periodic, and random delays:

Theorem 1 (Stability under output jitter). *The closed-loop system in Figure 1 is stable for any time-varying delays $\Delta \in [0, Nh]$, where $N > 0$ is a real number, if*

$$\left| \frac{P_{\text{alias}}(\omega)K(e^{i\omega})}{1 + P_{\text{ZOH}}(e^{i\omega})K(e^{i\omega})} \right| < \frac{1}{\tilde{N}\left|e^{i\omega} - 1\right|}, \quad \forall \omega \in [0, \pi], \tag{1}$$

where $\tilde{N} = \sqrt{\lfloor N \rfloor^2 + 2\lfloor N \rfloor g + g}$ and $g = N - \lfloor N \rfloor$; $P_{\text{ZOH}}(z)$ is the zero-order hold discretization of $P(s)$, and

$$P_{\text{alias}}(\omega) = \sqrt{\sum_{k=-\infty}^{\infty} \left| P\left(i(\omega + 2\pi k)\frac{1}{h} \right) \right|^2}. \tag{2}$$

Proof. *See [3].* □

2.2 Definitions and Properties

We now consider a periodic control task with the period $T = h$, executing in a real-time system. The plant is assumed to be sampled when the task is released, and the control signal is actuated when the task finishes.

The input-output delay experienced by the controller can be divided into two parts: a constant part, $L \geq 0$, and a time-varying part (the jitter), $J \geq 0$, see Figure 2. The

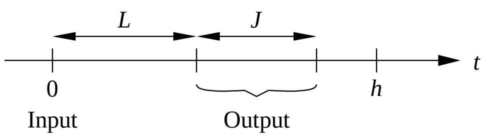


Fig. 2. The input-output delay can be divided into a constant delay, L, and a jitter, J.

minimum possible delay is hence given by L, and the maximum possible delay is given by $L+J$.

We will first recall the definition of the classical *delay margin* for the jitter-free case ($J=0$):

Definition 1 (Delay margin). *Given the system in Figure 1, the delay margin is defined as the largest number L_m for which closed-loop stability is guaranteed assuming a* constant delay $\Delta = L_m$.

Remark 1. For continuous-time control systems, the delay margin can be computed as

$$L_m = \varphi_m / \omega_c, \tag{3}$$

where φ_m is the *phase margin* and ω_c is the *crossover frequency* of the system. Due to aliasing effects, the exact computation is more complicated for computer-controlled systems (see [4]).

In systems with jitter, the delay and the jitter will both contribute to the destabilization of the system. Hence, we give the following definition of the *jitter margin*:

Definition 2 (Jitter margin). *Given the system in Figure 1, the jitter margin is defined as the largest number $J_m(L)$ for which closed-loop stability is guaranteed for any time-varying delay $\Delta \in [L, L+J_m(L)]$.*

Remark 2. Since Theorem 1 is only sufficient, it can only be used to compute a lower bound on the jitter margin. The theorem is not very conservative, however. To apply the theorem, we replace the plant $P(s)$ by its time-delayed version $P(s)e^{-sL}$ and let $N = J/h$.

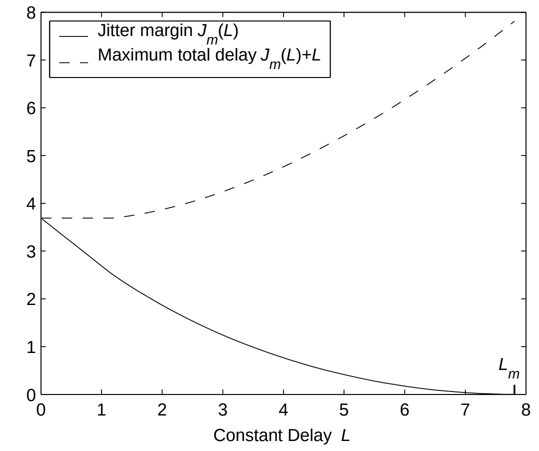
The reason for defining the jitter margin as a function of L is to make the stability test less conservative whenever a lower bound on L is available. It is obvious that, if a system is stable for any time-varying delay $\Delta \in [0, J]$, it must also be stable for any time-varying delay $\Delta \in [L, J]$, $0 < L \leq J$. Furthermore, in the latter case, the system might also be stable for longer delays. Based on this argument, the following properties of the jitter margin can be derived (the proofs are omitted):

Property 1. $J_m(L) = 0, \quad L \geq L_m$.

Property 2. $J_m(L) \leq L_m, \quad \forall L$.

Property 3. $J_m(L) + L$ is an increasing function of L.

(a)



(b)

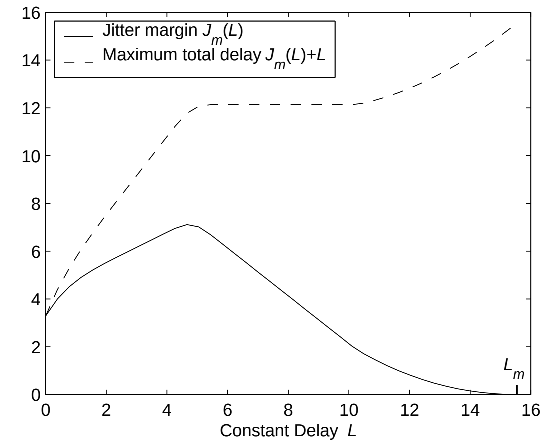


Fig. 3. Example of jitter margins $J_m(L)$: (a) PID controller with $h = 10$, (b) LQG controller with $h = 10$, designed for the delay $L = 5$. (All units are in ms.)

Example 1 (Jitter margin). Figure 3 reports the jitter margin as computed by Theorem 1 for the plant $P(s) = 1000/(s(s+1))$ and two different controllers. Both controllers are designed with the sampling interval $h = 10$ [ms]. In (a), a PID controller is used. The delay margin is $L_m = 7.8$, and the jitter margin has the maximum value $J_m(0) = 3.7$. In (b), an LQG controller designed for a constant delay $L = 5$ is used. Here, the delay margin is $L_m = 15.5$, and the jitter margin has the maximum value $J_m(4.8) = 7.1$. It can be seen that the jitter-margin function can have different shapes for different controllers, but the maximum total delay, $J_m(L) + L$, is always an increasing function. $\square$

2.3 Verifying Stability and Performance

If we know the constant delay L and the jitter J of a control task, stability of the closed-loop system is guaranteed if

$$J_m(L) > J. \tag{4}$$

Often, it is not enough to just guarantee stability—there must also be some margins that guarantee performance. In classical control theory, the phase margin is sometimes used as a performance and robustness measure. Unfortunately, the phase margin is only defined for systems without jitter. It is, however, possible to generalize the concept via an extended definition of the delay margin. Hence, we start by defining a delay margin for systems with delay and jitter:

Definition 3 (Delay margin for systems with delay and jitter). *Given the system in Figure 1, assuming some constant delay L and jitter J, the delay margin is defined as the largest number L_m for which closed-loop stability is guaranteed for any time-varying delay $\Delta \in [L + L_m, L + L_m + J]$.*

Remark 3. For systems without jitter, this definition is equivalent to Definition 1.

Expressed in terms of the jitter-margin function $J_m(L)$, the delay margin is given by the smallest L_m that solves

$$J_m(L + L_m) = J. \tag{5}$$

For the control designer, it is often more convenient to think in terms of phase margin, since that measure is independent of time. For systems without jitter, the relationship between phase margin and delay margin is approximately given by (3). Based on this observation, we propose the notion of *apparent phase margin:*

Definition 4 (Apparent phase margin). *Given the system in Figure 1, assuming the constant delay L and the jitter J, the apparent phase margin is defined as the largest number $\hat{\varphi}_m$ for which closed-loop stability is guaranteed for any time-varying delay $\Delta \in [L + \hat{\varphi}_m/\omega_c, L + \hat{\varphi}_m/\omega_c + J]$, where ω_c is the crossover frequency of the system if assuming only the constant delay L.*

Similar to above, expressed in terms of the jitter-margin function $J_m(L)$, the apparent phase margin is given by the smallest $\hat{\varphi}_m$ that solves

$$J_m(L + \hat{\varphi}_m/\omega_c) = J. \tag{6}$$

A system with the apparent phase margin $\hat{\varphi}_m \leq 0^\circ$ can be interpreted as a system for which stability cannot be guaranteed, while any $\hat{\varphi}_m > 0^\circ$ can be interpreted as a performance guarantee. For systems without jitter, the apparent phase margin is equal to the classical phase margin.

2.4 Deadline Assignment

In the real-time literature, task deadlines are often considered as given parameters. Using the jitter margin, we can derive *real* hard deadlines that guarantee closed-loop stability. For instance, given that we have a lower bound on the constant delay L in the target system, we can guarantee stability by assigning the relative deadline

$$D = L + J_m(L). \tag{7}$$

(It is of course also required that all deadlines are really met during run-time.) Note that, if no estimate of L is available, assuming $L = 0$ yields a more conservative deadline.

Similarly, we can assign deadlines that guarantee a certain apparent phase margin in the target system. Given a lower bound on the constant delay L in the target system and a desirable apparent phase margin $\hat{\varphi}_m < \omega_c(L_m - L)$, we can guarantee a level of performance by assigning the deadline

$$D = L + J_m(L + \hat{\varphi}_m/\omega_c). \tag{8}$$

Example 2 (Deadline assignment). Consider the LQG controller in Example 1, whose jitter margin is shown in Figure 3(b). Without jitter, assuming $L = 5$, the phase margin is $\varphi_m = 34.9^\circ$ and the crossover frequency is $\omega_c = 57.9$ rad/s. Suppose that we require an apparent phase margin of $\hat{\varphi}_m = 20^\circ$. The allowable jitter is then given by

$$J_m(5 + 20^\circ/57.9\ \text{rad}) = J_m(11.0) = 1.4,$$

and we should hence assign the relative deadline

$$D = L + J_m(11.0) = 6.4.$$

□

An interesting problem here is that, depending on the scheduling policy, the constant delay might depend on the deadline which we are trying to compute. For instance, under deadline-monotonic scheduling, the assigned deadline will affect the priority of the task, which might in turn affect the constant delay. The problem could possibly be addressed using an iterative deadline assignment procedure, but this is left as future work.

3 Output Jitter Analysis

In order to apply the stability and performance analysis of the previous section, we need to be able to compute the constant delay and the jitter for each control task in the system. This can be done using response-time analysis. Let R_i and R_i^b denote, respectively, the worst-case and best-case response times of task i. The constant delay, L_i, and the jitter, J_i, are then given by

$$L_i = R_i^b, \tag{9}$$

$$J_i = R_i - R_i^b. \tag{10}$$

Often, the true values of R_i and R_i^b cannot be obtained. First, if the task phasing is unknown, one must assume *worst-case phasing* when computing R_i and *best-case phasing* when computing R_i^b. It is not certain that R_i and R_i^b can *both* occur during the lifetime of the system. Second, depending on the scheduling policy and the task set, exact analysis for the worst-case and the best-case response times may not be available.

From a stability perspective, it is always safe to overestimate R_i and to underestimate R_i^b. This will make L_i smaller and J_i larger, causing the apparent phase margin to decrease.

Below, a brief outline of the available results in response-time analysis under fixed-priority and EDF scheduling is given. For EDF, a new lower bound on best-case response times is proposed.

3.1 Worst-Case Response Time Analysis

Under fixed-priority scheduling, assuming $D_i \leq T_i$, the worst-case response time of task i is given by the well-known equation [5]

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j. \tag{11}$$

Exact analysis also exists for task sets with release offsets as well as deadlines $D > T$ [6, 7].

Under EDF scheduling, worst-case response-time analysis is more complicated. Assuming $D_i \leq T_i$, the worst-case response time of task i is given by [8, 9]

$$R_i = \max\Big\{C_i, \max_{a \geq 0} \{L_i(a) - a\}\Big\}, \tag{12}$$

where the busy interval $L_i(a)$ is given by the equation

$$L_i(a) = W_i\big(a, L_i(a)\big) + \left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) C_i, \tag{13}$$

and the higher-priority workload $W_i(a,t)$ is given by

$$W_i(a,t) = \sum_{j \neq i, D_j \leq a + D_i} \min\left\{ \left\lceil \frac{t}{T_j} \right\rceil, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor \right\} C_j. \tag{14}$$

It should be noted that only a finite number of values of a must be checked when evaluating (12). The analysis has also been generalized to arbitrary deadlines [8].

3.2 Best-Case Response Time Analysis

Under fixed-priority scheduling, exact best-case analysis has recently been developed for the case $D \leq T$ [10]. The best-case response time of task i is given by the equation

$$R_i^b = C_i^b + \sum_{j \in hp(i)} \left\lceil \frac{R_i^b}{T_j} - 1 \right\rceil C_j^b, \tag{15}$$

where C_i^b denotes the *best-case execution time* of task i.

Under EDF scheduling, no exact best-case analysis is known to exist. A trivial lower bound $\underline{R}_i$ on the best-case response time of task i is given by

$$\underline{R}_i^b = C_i^b. \tag{16}$$

This is actually a quite good bound for the shortest-period tasks. The longest-period tasks can, however, have much longer best-case response times, especially if the system load is high.

A tighter lower bound on the best-case response time can be obtained by interference analysis, see Appendix A. Our proposed lower bound, $\underline{R}_i$, is given by the equation

$$\underline{R}_i^b = C_i^b + \sum_{\forall j: D_j < \underline{R}_i^b} \left\lceil \frac{\min\left\{\underline{R}_i^b, D_i - D_j\right\}}{T_j} - 1 \right\rceil C_j^b, \tag{17}$$

which can be solved by recursion from above (cf. [10]).

The results obtained with the proposed bound have been compared to results obtained by simulation, where the shortest response time of each task was recorded. (Note that the latter constitutes an *upper bound* on the real best-case response time.) The bounds were evaluated for loads ranging from $U = 0.5$ to $U = 0.99$. For each load case, 100 random task sets were generated. The number of tasks in each set was integer-uniformly distributed between 2 and 10. The task periods were exponentially distributed with mean 1, and the fraction of the execution time to the period was uniformly distributed between 0 and 1. The execution times were uniformly rescaled to give the task set the desired utilization. Throughout, $D_i = T_i$ and $C_i^b = C_i$ were assumed.

For each task set, the system was simulated for 1000 s, and the minimum response time of the longest-period task (task n) was recorded. The result was compared with the bounds (16) and (17). Figure 4 shows the mean of R_n^b/C_n over the task sets for different bounds and different loads. It is seen that the proposed bound performs quite well up to a load of $U = 0.95$. The bound is not tight since it does not consider initial interference, see Appendix A.

4 A Codesign Procedure

To illustrate how the jitter margin could be applied in the design of real-time control systems, we describe an iterative control–scheduling codesign procedure.

It is assumed that a set of independent controllers should be implemented in the same processor. The controllers are designed in continuous time, and should be discretized and implemented as periodic tasks with different periods. The goal of the codesign procedure is to choose sampling periods such that the controllers will experience the same relative performance degradation in the target system, taking the jitter into account. The performance of the continuous-time controller is measured by its original phase margin φ_m, and the performance of the control task is measured by its apparent phase margin $\hat{\varphi}_m$ (see Section 2.3). The goal of the procedure is to make the ratio $\hat{\varphi}_m/\varphi_m$ as equal as possible among the tasks.

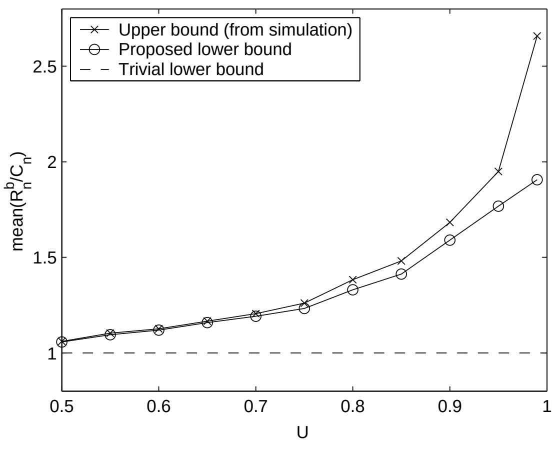


Fig. 4. Comparison of bounds on the best-case response time under EDF. The results are shown for the longest-period task.

The inputs to the codesign procedure are a set of n continuous-time plants, $P(s)$, a set of n continuous-time controllers, $K(s)$, estimates of the best-case and worst-case execution times of the control algorithms, C and C^b, and a scheduling policy where worst-case as well as best-case response time analysis is available.

The procedure is outlined is the following steps:

1. Initialize by assigning initial (nominal) sampling periods h for the controllers. (A common rule of thumb [4] is to choose the sampling period such that $\omega_b h \in [0.2, 0.6]$, where ω_b is the bandwidth of the closed-loop continuous system.)
2. Rescale the periods linearly such that the task set becomes schedulable under the given scheduling policy. (Here, a suitable sufficient schedulability test can be used.)
3. Discretize the controllers using the assigned sampling periods, yielding the set of discrete-time controllers $K(z)$.
4. For each task, compute worst-case and best-case response times, R and R^b. (Here, the analysis in Section 3 is applicable.)
5. For each task, compute the jitter margin using Theorem 1 and the apparent phase margin $\hat{\varphi_m}_i$ from (6), assuming the constant delay $L_i = R_i^b$ and the jitter $J_i = R_i - R_i^b$.
6. For each task, compute the relative performance degradation $r_i = \hat{\varphi_m}_i / \varphi_{m_i}$. Also, compute their mean value, $\overline{r} = \sum r_i / n$.
7. For each task, adjust the period according to
$$h_i := h_i + k h_i (r_i - \overline{r}) / \overline{r},$$
where $k < 1$ is a gain parameter.
8. Repeat from 2 until no further improvement is given. A suitable stop criterion is when sum of the performance differences, $\sum |r_i - \overline{r}|$, is no longer decreasing.

The period adjustment mechanism in step 7 is intended to decrease the periods of controllers with bad performance, and to increase the periods of controllers with good performance. Choosing the gain parameter can be difficult. A small k will give slow adaptation, while a large k can cause instability.

The iterative procedure tries to solve a highly nonlinear optimization problem. Hence, it is not certain that it will converge to an optimal solution. For instance, under rate-monotonic scheduling, a small period adjustment may change the task priorities, and this can in turn have a huge impact on the jitter. Neither is it certain that a completely equal performance degradation can be achieved.

Example 3 (Codesign). We consider an example where three controllers should be implemented in a single CPU. Both rate-monotonic and EDF scheduling is considered. The execution times of the control algorithms are assumed to be equal and constant and are given by $R = R^b = 0.15$ [ms]. The plants to be controlled are given by

$$\begin{aligned} P_1(s) &= \frac{8\cdot 10^5}{s(s+1000)}, \\ P_2(s) &= \frac{4\cdot 10^4}{(s-200)(s+200)}, \\ P_3(s) &= \frac{5\cdot 10^7}{s(s^2+100s+2.5\cdot 10^5)}, \end{aligned} \tag{18}$$

and the continuous-time controllers are given by

$$\begin{aligned} K_1(s) &= \frac{4.88\cdot 10^3(s+2\cdot 10^5)(s+1295)}{(s+5000)(s^2+7.325\cdot 10^4 s+2.573\cdot 10^9)}, \\ K_2(s) &= \frac{2.57\cdot 10^3(s+2\cdot 10^5)(s+259.1)}{(s+3000)(s^2+1.645\cdot 10^4 s+1.35\cdot 10^8)}, \\ K_3(s) &= \frac{478(s+2\cdot 10^5)(s^2+160.6s+1.655\cdot 10^5)}{(s+2740)(s+1000)(s^2+2494s+7.109\cdot 10^6)}. \end{aligned} \tag{19}$$

Table 1 reports the bandwidth ω_b and the original phase margin φ_m of each control loop. It is seen that the loops have different bandwidths, which suggests that the controllers would require different sampling intervals. The differences in bandwidth are also visible in Figure 5, which shows the system responses for the different continuous-time loops.

To initialize the procedure, nominal sampling periods are chosen by the rule of thumb $\omega_b h = 0.2$. This results in a CPU utilization of $U = 1.30$. Hence, slower sampling must be used in the target system. For the controller discretization, the Tustin method is used.

Table 1. Bandwidths and phase margins of the original continuous-time control loops

Loop	ω_b	φ_m
$P_1(s), K_1(s)$	960 rad/s	74.1°
$P_2(s), K_2(s)$	599 rad/s	49.5°
$P_3(s), K_3(s)$	179 rad/s	69.7°

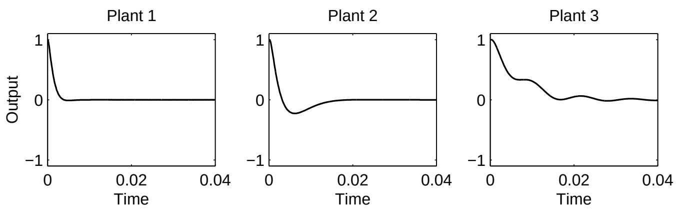


Fig. 5. System responses of the original continuous-time control loops.

First, rate-monotonic scheduling is assumed. The target utilization is chosen as $U = 0.78$. The adaptation gain is chosen as $k = 0.2$. The results of the codesign procedure after one and ten iterations are shown in Table 2. After the initial iteration, where the nominal sampling periods have been simply rescaled, loop 3 has a small negative apparent phase margin. That means that stability cannot be guaranteed for that loop. After ten iterations, the periods have been adjusted such that they are nearly equal, resulting in a somewhat more equal performance degradation (as measured by the ratio $\hat{\varphi}_m/\varphi_m$).

To verify the results of the procedure, the complete real-time system (including plants, controllers, and scheduler) was also simulated using the MATLAB/Simulink toolbox TrueTime [11]. The actual control system responses after one and ten design iterations are shown in Figure 6. It is seen that, after one iteration, loop 3 is close to unstable, as predicted by the negative apparent phase margin. After ten iterations, the performance degradation of loop 3 is visibly smaller.

Next EDF scheduling is assumed. The target utilization is chosen as $U = 0.95$. The results of the codesign procedure after one and ten iterations are shown in Table 3. After the initial iteration, task 3 has a large negative apparent phase margin, implying that the control loop might be unstable. After ten iterations, the performance degradation is quite even among the controllers. Again, the results were also verified in simulations. Figure 7 shows the system response after one and ten iterations.

Table 2. Codesign results under rate-monotonic scheduling: (a) after one iteration, (b) after ten iterations.

(a)

Task	h	R	R^b	J	$J_m(R^b)$	$\hat{\varphi}_m$	$\hat{\varphi}_m/\varphi_m$
1	0.35	0.15	0.15	0	1.08	60.8°	0.82
2	0.56	0.30	0.15	0.15	1.17	27.9°	0.56
3	1.87	0.90	0.15	0.75	0.47	−4.8°	−0.07

(b)

Task	h	R	R^b	J	$J_m(R^b)$	$\hat{\varphi}_m$	$\hat{\varphi}_m/\varphi_m$
1	0.56	0.15	0.15	0	0.96	56.5°	0.76
2	0.57	0.30	0.15	0.15	1.17	27.7°	0.56
3	0.60	0.45	0.15	0.30	1.18	27.9°	0.40

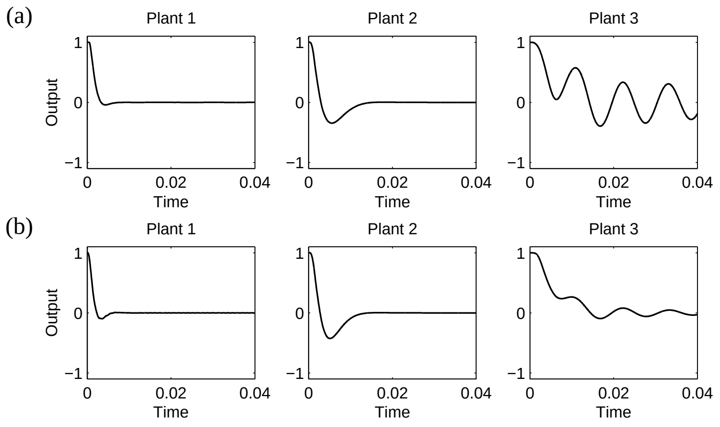


Fig. 6. Control system responses under rate-monotonic scheduling: (a) after one iteration, (b) after ten iterations.

Table 3. Codesign results under EDF scheduling: (a) after one iteration, (b) after ten iterations.

(a)

Task	h	R	R^b	J	$J_m(R^b)$	$\hat{\varphi}_m$	$\hat{\varphi}_m/\varphi_m$
1	0.28	0.16	0.15	0.01	1.11	64.0°	0.86
2	0.46	0.34	0.15	0.19	1.21	33.4°	0.67
3	1.53	1.35	0.60	0.75	0.03	−18°	−0.27

(b)

Task	h	R	R^b	J	$J_m(R^b)$	$\hat{\varphi}_m$	$\hat{\varphi}_m/\varphi_m$
1	0.40	0.31	0.15	0.16	1.04	43.1°	0.58
2	0.50	0.40	0.15	0.25	1.19	26.7°	0.54
3	0.54	0.45	0.15	0.30	1.20	29.8°	0.43

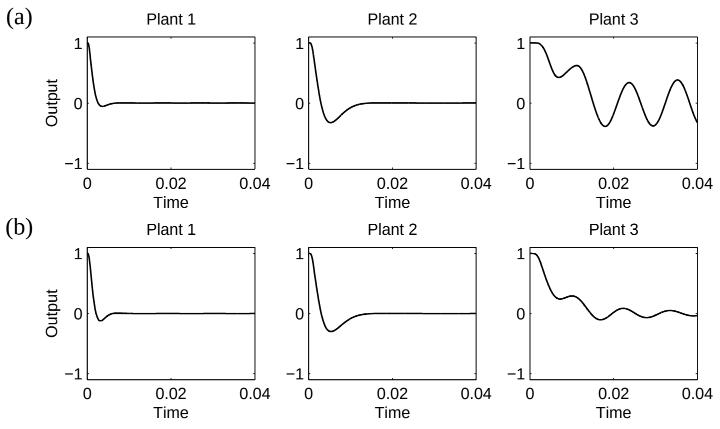


Fig. 7. Control system responses under EDF scheduling: (a) after one iteration, (b) after ten iterations.

The final design results under rate-monotonic scheduling and EDF scheduling are quite similar. Under EDF, slightly shorter periods could be used, due to the higher level of schedulability of EDF. It can also be noted that, under EDF, the jitter is more evenly distributed among the tasks. This makes it possible to achieve a more even performance degradation among the control loops.

5 Related Work

Several works have considered scheduling solutions to reduce output jitter in general. In [12] and [13], it is suggested to use dedicated high-priority output tasks to reduce the jitter. This has the disadvantages of a more complex implementation and longer delays on average. [14] considers jitter reduction under deadline-monotonic and EDF scheduling. Output jitter reduction under EDF is also the topic of [15] and [16]. It can be noted that, in these papers, the jitter is defined between successive periods, rather than over the lifetime of the system (as in this paper).

There have also been some efforts to specifically minimize jitter in control tasks. The papers [17, 18] define the *control action interval*, which is just another term for output jitter. The proposed solution introduces high-priority tasks for the input and output actions. Again, this has the disadvantage of longer delays on average. Also, the resulting control performance is not analyzed. [19] proposes a subtask scheduling method for control tasks, where the main part of the control algorithm are scheduled at different priorities. The scheme attempts to reduce both the delay and the jitter. The performance improvements are verified by simulations.

Jitter compensation in control has been the subject of much research. In [20], an optimal jitter-compensating LQG controller is derived in the context of networked control loops. The controller uses timestamps to track the sensor-to-controller and controller-to-actuator delays. The performance is measured by a quadratic cost function and is evaluated by stochastic analysis. [21] considers jitter compensation in state feedback controllers. No specified scheduling algorithm is considered, but it is assumed that the delays are known a-priori. Also, full state information is assumed. The performance improvements are verified by simulations. In [22] a more realistic approach is taken, where the output jitter experienced in one period is compensated for in the next period. The resulting jitter-compensating controller can be viewed as a generalization of the well-known Smith predictor.

In the area of control–scheduling codesign, [23] studies computational delays in computer-controlled systems. Hard constraints on the controlled variables (e.g., physical constraints) are used to derive maximum allowable control latencies in different regions of the statespace. It is noted that the hard deadline may be a random variable due to stochastic disturbances acting on the process. The approach is extended in [24] where the stability of the closed-loop system is also considered. Sampling period selection for control tasks is the topic of [25]. The performance of the control loops are described using cost functions, and the period assignment problem is formulated as an optimization problem. The combined effect of period and delay on control performance is studied in [26], where simulations are used to evaluate the performance. None of these papers considers jitter, however.

6 Conclusion

This paper has proposed the notion of *jitter margin* and showed how it can be applied in the design of real-time control systems. The stability test is based on worst-case assumptions about the jitter, and hence produces hard stability results. We have also linked the control analysis to scheduling analysis, showing how output jitter analysis can be used together with the jitter margin. An extensive codesign example has been presented, where many of the concepts introduced in the paper have been applied.

This paper has only treated output jitter. In some applications, sampling jitter is also an issue. We are investigating if the stability analysis can be extended to also handle this case.

The topic of best-case response-time analysis needs to be investigated further. For instance, exact best-case response-time analysis under EDF could be developed. It would also be interesting to consider jitter analysis where the same task phasing is assumed for the best-case and the worst-case response-time analysis.

The suggested codesign approach is only one of many possible. It would be interesting to also consider direct digital design, where the controller is designed to compensate for the constant delay. In this case, a quadratic cost function is probably a better performance measure than the apparent phase margin.

A A Lower Bound on the Best-Case Response Time under EDF

Consider a set of periodic tasks scheduled under EDF. Each task i has a period T_i, a relative deadline $D_i \leq T_i$, and a best-case execution time C_i^b. It is assumed that the task set is schedulable. Let R_i be the response time of an instance of task i that is released at time 0, and let task j be a potentially interfering task. We will construct a lower bound on R_i by shifting each task j such that minimum interference is obtained.

First, consider a task j with $D_j \geq R_i$. It is obvious that the task can be phased such that it does not interfere with task i.

For each task j with $D_j < R_i$ we must consider two different cases, see Figure 8. In case (a), $D_i - D_j > R_i$, and each instance of task j released within the interval $[0, R_i]$ will have higher priority than task i. Minimum interference is obtained when task j is phased such that one release occurs at time R_i. The number of complete preemptions from task j is hence given by $\lceil R_i/T_j - 1 \rceil$.

In case (b), $D_i - D_j \leq R_i$, and instances of task j will only have higher priority if released within the interval $[0, D_i - D_j]$. Minimum interference is obtained when task j is phased such that one release occurs at time $D_i - D_j$. The number of complete preemptions from task j is hence given by $\lceil (D_i - D_j)/T_j - 1 \rceil$.

Each complete preemption from task j will contribute C_j to the response time. Combining the two cases above, a lower bound, $\underline{R}_i^b$, on the minimum response time of task i is given by

$$\underline{R}_i^b = C_i^b + \sum_{\forall j : D_j < \underline{R}_i^b} \left\lceil \frac{\min\left\{\underline{R}_i^b, D_i - D_j\right\}}{T_j} - 1 \right\rceil C_j^b$$

This expression provides only a lower bound, since it does not take any initial (partial) interference from task j into account (see for instance Figure 8(a)). It is possible to

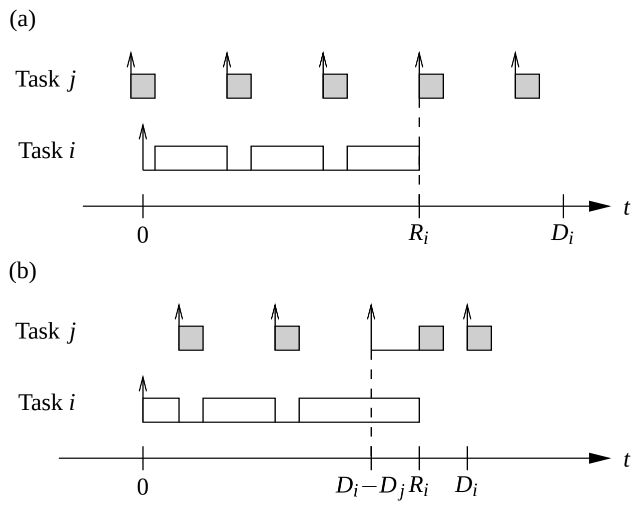


Fig. 8. Different cases where task j causes minimum interference for task i: (a) $D_i - D_j > R_i$, (b) $D_i - D_j \leq R_i$.

improve the formula slightly by including some obvious cases where initial interference must occur. It is conjectured, however, that the expression for the *exact* best-case response time is as complex as the formula for exact worst-case response time under EDF.

Acknowledgments

The work in this paper has been partially supported by EU/ARTIST (http://www.artist-embedded.org) and by SSF/FLEXCON (http://www.control.lth.se/FLEXCON).

References

1. Franklin, G., Powell, D., Emami-Naeini, A.: Feedback Control of Dynamic Systems. 4th edn. Prentice Hall (2002)
2. Törngren, M.: Fundamentals of implementing real-time control applications in distributed computer systems. Real-Time Systems **14** (1998)
3. Kao, C.Y., Lincoln, B.: Simple stability criteria for systems with time-varying delays. Automatica (2004) To appear in September 2004. Preprint available at http://www.control.lth.se.
4. Åström, K.J., Wittenmark, B.: Computer-Controlled Systems. Prentice Hall (1997)
5. Joseph, M., Pandya, P.: Finding response times in a real-time system. The Computer Journal **29** (1986) 390–395
6. Audsley, N., Tindell, K., Burns, A.: The end of the line for static cyclic scheduling. In: Proc. 5th Euromicro Workshop on Real-Time Systems. (1993)
7. Tindell, K., Burns, A., Wellings, A.J.: An extendible approach for analyzing fixed priority hard real-time tasks. Real-Time Systems **6** (1994) 133–151

8. George, L., Rivierre, N., Spuri, M.: Preemptive and non-preemptive real-time uniprocessor scheduling. Technical Report 2966, Institut National de Recherche en Informatique et en Automatique (1996)
9. Stankovic, J.A., Spuri, M., Ramamritham, K., Buttazzo, G.C.: Deadline Scheduling for Real-Time Systems—EDF and Related Algorithms. Kluwer Academic Publishers (1998)
10. Redell, O., Sanfridson, M.: Exact best-case response time analysis of fixed priority scheduled tasks. In: Proc. 14th Euromicro Conference on Real-Time Systems, Vienna, Austria (2002)
11. Henriksson, D., Cervin, A., Årzén, K.E.: TrueTime: Simulation of control loops under shared computer resources. In: Proceedings of the 15th IFAC World Congress on Automatic Control, Barcelona, Spain (2002)
12. Locke, C.D.: Software architecture for hard real-time applications: Cyclic vs. fixed priority executives. Real-Time Systems **4** (1992) 37–53
13. Klein, M.H., Ralya, T., Pollak, B., Obenza, R., Gonzalez Härbour, M.: A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems. Kluwer Academic Publisher (1993)
14. David, L., Cottet, F., Nissanke, N.: Jitter control in on-line scheduling of dependent real-time tasks. In: Proc. 22nd IEEE Real-Time Systems Symposium. (2001)
15. Baruah, S., Buttazzo, G., Gorinsky, S., Lipari, G.: Scheduling periodic task systems to minimize output jitter. In: Proc. 6th International Conference on Real-Time Computing Systems and Applications. (1999)
16. Kim, T., Shin, H., Chang, N.: Deadline assignment to reduce output jitter of real-time tasks. In: Proc. 16th IFAC Workshop on Distributed Computer Control Systems. (2000)
17. Crespo, A., Ripoll, I., Albertos, P.: Reducing delays in RT control: The control action interval. In: Proc. 14th IFAC World Congress. (1999) 257–262
18. Balbastre, P., Ripoll, I., Crespo, A.: Control task delay reduction under static and dynamic scheduling policies. In: Proc. 7th International Conference on Real-Time Computing Systems and Applications. (2000)
19. Cervin, A.: Improved scheduling of control tasks. In: Proceedings of the 11th Euromicro Conference on Real-Time Systems, York, UK (1999) 4–10
20. Nilsson, J.: Real-Time Control Systems with Delays. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Sweden (1998)
21. Marti, P., Fohler, G., Ramamritham, K., Fuertes, J.M.: Jitter compensation for real-time control systems. In: Proc. 22nd IEEE Real-Time Systems Symposium. (2001)
22. Lincoln, B.: Jitter compensation in digital control systems. In: Proceedings of the 2002 American Control Conference. (2002)
23. Shin, K.G., Krishna, C.M., Lee, Y.H.: A unified method for evauating real-time computer controllers and its applications. IEEE Transactions on Automatic Control **30** (1985) 357–366
24. Shin, K.G., Kim, H.: Derivation and application of hard deadlines for real-time control systems. IEEE Transactions on Systems, Man, and Cybernetics **22** (1992) 1403–1413
25. Seto, D., Lehoczky, J.P., Sha, L., Shin, K.G.: On task schedulability in real-time control systems. In: Proc. 17th IEEE Real-Time Systems Symposium, Washington, DC (1996) 13–21
26. Ryu, M., Hong, S., Saksena, M.: Streamlining real-time controller design: From performance specifications to end-to-end timing constraints. In: Proc. 3rd IEEE Real-Time Technology and Applications Symposium. (1997) 91–99