Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Response Time Driven Design of Control Systems ⋆

Yang Xu, Karl-Erik Årzén, Enrico Bini, Anton Cervin

Department of Automatic Control, Lund University, Sweden (e-mail: yang, karlerik, bini, anton@control.lth.se).

Abstract: A correct design of controllers must necessarily account for the schedule of the controller task over the processor. Existing design techniques are based on the assumption that there is no delay, or the delay is constant over time. However, in practice, almost all controllers have time-varying delay, hence invalidating this assumption. In this paper, we introduce a period design policy, in which the controller delay is modelled by the distribution of the task response time. We show, via simulation, that our method can reduce the control cost compared to the state-of-art methods.

Keywords: Real-time systems; scheduling algorithms; control systems; LQG control; time delay

1. INTRODUCTION

The demand for efficient resource utilization in embedded and cyber-physical systems has led to an increased focus on co-design approaches, including control and scheduling co-design. The scenario studied here consists of a number of linear-quadratic-Gaussian (LQG) controllers, each implemented as a task executing on a single processor that uses preemptive fixed priority scheduling. The co-design problem consists of selecting the task periods and designing the individual controllers so that the total control performance is maximized.

As controllers are scheduled over a processor, their execution is subject to all effects that the task scheduler introduces.

- Since the computation of the control law consumes some execution time on the processor, there is always a *delay* between sampling and actuation (so called *input-output delay*).
- When a task can be preempted by the others then the amount of delay is not constant, since it depends on the (time-varying) number of times the task was actually preempted.

In earlier work, as later described in the related works section, the optimal task period selection problem has been approached either by ignoring the effects of the delay on the performance or by approximating the delay by a constant. In this work the delay is instead modeled by the task response time distribution, and the controller is designed to give optimal performance for a delay that varies according to such a distribution, i.e., a robust control design approach is used. This is done using the techniques by Nilsson et al. (1998) that enable the optimal design of controllers that are subject to an input-output delay, which is assumed to be a random variable with known distribution. One hypothesis behind this technique is that the delay of each job is *independent* from the others, which is certainly not the case in real systems. However, if the parameters of all tasks are known in advance, it is possible to determine the entire task schedule within the least common multiple of the periods (called *hyperperiod*), which is then repeated over time.

The picture, however, does drastically change if the controllers have to be designed, that is, if the task periods are *variables* of an optimization problem. In this case, the hyperperiod may not even exist (if any pair of task periods are incommensurable) and then the task schedule cannot be computed.

The design of controllers (which includes also the selection of their period) that are aware of the job delay pattern is a challenging problem, which is considered in this paper. In the next section, we review the related works.

Seto et al. (1996) studied the optimal task rate assignment such that a given performance index is maximized and the available computational resources are not overloaded. In this work, which is a milestone in the literature of real-time control co-design, however, the delay between sensing and actuation is still assumed to have no impact on the performance of the system. This was pointed out by Kim (1998). He expressed the cost as a function of both periods and delays, and then found the periods assuming that the delays are given. Then, the new delays are computed by simulating the schedule of all the tasks up to the hyperperiod, and iteratively the periods are computed again assuming the new values of delay. Clearly, this method can be extremely time consuming because it depends on the length of the hyperperiod. Another limitation of (Seto et al., 1996) was that they assumed as feasibility constraint the utilization upper bound from Liu and Layland (1973). However, the utilization upper bound is only a sufficient condition when a fixed priority scheduler is used. Bini and Di Natale (2005) proposed an algorithm

⋆ This work was supported in part by the LCCC Linnaeus Center, the eLLIIT Excellence Center at Lund University, and the Marie Curie Intra European Fellowship within the 7th European Community Framework Programme.

that finds the optimal period assignment of control tasks scheduled by fixed priority. In their work, the delay is guaranteed not to exceed the period for all tasks. Since the optimal method requires a time-consuming branch-and-bound algorithm to be executed, they also proposed a faster algorithm to find a sub-optimal period assignment taking advantage of some geometrical considerations in the space of feasible activation rates. Similarly to this paper, Bini and Cervin (2008) proposed an iterative period assignment routine based on an analytic solution of the linearised problem, by assuming a constant delay equal to an approximated response time. More recently, Aminifar et al. (2012) proposed a method to assign task periods so that the stability of the plants is guaranteed. The delay-related cost, however, is not taken into account.

In the context of on-line assignment of periods to control task, Martí et al. (2004) and Henriksson and Cervin (2005) suggested to reallocate more resources to the control loop currently in need. However, none of these considered the control delay in their resource allocation schemes.

The work by Nilsson et al. (1998) assumed that the variability in delay was smaller than the sampling period. This limitation was later lifted by Lincoln and Bernhardsson (2000), who derived the LQG-optimal controller for networked control systems with arbitrarily long delays. However, no stationary solution was found, implying that the optimal controller involves quite heavy on-line computations. In this work, therefore, we design controllers based on truncated distributions so that the variability is always smaller than the period, even though this approach is suboptimal.

Closely related to our work, Samii et al. (2009) proposed a control-scheduling co-design procedure, where the control design step took the response-time distribution — found by schedule simulation — into account. However, the controllers were designed based on the average delay only. Also, the overall optimization was based on a genetic algorithm, which assigned controller periods within a fixed and given set only.

In this paper, we propose a design technique for controllers sharing the same processor. We show, assuming overall cost is a function of periods, how the local and global optimal period assignment can be found. Our method can also be used when the offset of job release time exists or when the execution time is distributed. The main contribution of this paper is the period assignment including the effects of delay and jitter by using response time analysis.

In Section 2.1 the task run-time model is presented. The controller and the definition for the control cost are introduced in Section 2.2. Validation for modeling the delay with the response-time distribution is provided in Section 3. In Section 4 the optimal period assignment problem is defined together a local sequential search-based method. In Section 5 this method is compared with a non-linear direct optimization method in a simulation evaluation where the cost is evaluated using the TrueTime simulation tool Cervin et al. (2003).

2. REAL-TIME CONTROL SYSTEM MODEL

2.1 Task run-time model

We consider a set of n independent control tasks, each one controlling a plant. The tasks run over the same shared processor. A task τ_i is characterized by the following independent parameters:

- the *worst-case computation time* C_i is the maximum execution requirement that a task can require. We highlight that the computation time of controllers is often quite static and predictable, since the code typically has not many conditional branches;
- the *period* of the task activations T_i is the time separation between two consecutive activations;
- the task *priority*. Without loss of generality we assume that the priority is implicitly assigned by the task ordering, such that τ_i has higher priority than τ_{i+1}.

Other characteristics of the tasks, which depend on the above mentioned parameters, are

- the *response time* R_i of the task is the time that elapses from the task activation to its finishing time. Since R_i varies over time depending on the interference from higher priority tasks, we represent it by a random variable with cumulative distribution function $F_i : [0, \infty) \to [0, 1]$. The value $F_i(r)$ is the probability $P\{R_i \leq r\}$ that any job released by τ_i has response time smaller than or equal to r. The distribution F_i depends on the parameters of the tasks $\{\tau_1, \tau_2, \ldots, \tau_i\}$, since they are the only ones which can affect the response time of τ_i;
- the task *utilization*
$$U_i = \frac{C_i}{T_i},$$
which measures the worst-case amount of computational resources required by the controller. If $\sum_i U_i > 1$, then we say that we are in *overload* conditions.

2.2 Model of the control cost

We assume that the plant to be controlled by each task is described by a linear system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + v_c(t) \\ y(t_k) &= Cx(t_k) + e(t_k) \end{aligned} \tag{1}$$

where x is the plant state, u is the controlled input, and v_c is a continuous-time white noise process. The output y is measured at discrete time instants t_k, with measurement noise e described by a discrete-time Gaussian white noise process. A, B, and C are matrices of appropriate sizes.

The controller τ_i then reads the output of the corresponding plant every period T_i, computes the control law, and then actuates the control input to the plant after a constant delay Δ_i due to its own computation as well as to the interference of higher priority tasks. To measure the cost of this action, we adopt a linear-quadratic Gaussian (LQG) framework (Åström and Wittenmark, 1997), in which the cost is measured by a standard quadratic cost function

$$J = \lim_{t \to \infty} \frac{1}{t} \mathrm{E} \left\{ \int_0^t \left(x^T(s) Q x(s) + \rho u^2(s) \right) ds \right\}. \tag{2}$$

The parameter $\rho > 0$ is a weight used to trade between the relative importance of input u and the state x.

If the delay of the controller varies as a random variable with a given probability density, it is also possible to determine the optimal control law (Nilsson et al., 1998), which minimizes the cost of (2). In this case, it is done through an iterative solution of the corresponding stochastic Riccati equation. MATLAB and the Jitterbug toolbox (Lincoln and Cervin, 2002) can be used to compute both the optimal control feedback and the corresponding cost.

Since the cost J_i of the controller τ_i depends on the response time distribution, which in turn depends on the period of the higher priority tasks, then it can be written as

$$J_i(T_1, \ldots, T_n) \tag{3}$$

with

$$\frac{\partial J_i}{\partial T_k} = 0, \qquad \forall k = i+1, \ldots, n. \tag{4}$$

The model used assumes that the sampling is performed at the task release times, i.e., without any jitter. This can, e.g., be achieved by performing the sampling in the the clock interrupt service routine, or in a dedicated high-priority task that then communicates the sample to the controller task, using, e.g., a mailbox. Modelling the delay of the controller using the task response time assumes that the actuation is performed at the end of the task. Normally, this is not the way a controller is implemented. Instead the code is structured in two section: a *Calculate-Output* section that contains the calculations that directly depend on the current sample, and an *UpdateState* section where the internal states of the controller are updated. The actuation is then performed as soon as the *CalculateOutput* section is completed. However, since the approach in this paper is based on simulating the total task schedule it is straightforward to extend it to this case instead. An additional assumption implicitly made is that the task execution time is relatively constant from one job to the next. Although this is not true for general tasks, the assumption is more valid for controller tasks, since the code size is relatively small and the amount of branches is low. Furthermore, it is assumed the kernel allows the task periods to have arbitrary real-valued values. This implies a so-called tick-free kernel.

3. VALIDATING THE MODEL OF THE TASK DELAY

3.1 Choice of the response time distribution

The design approach in this paper is based on the possibility to design the controller taking the distribution of the delay caused by the task scheduling into account. This is done by modelling the delay by the response time distribution. However, when the tasks have constant execution times, the delay is actually deterministic, although following a pattern, which is not easily characterizable (Lehoczky, 1990). It is then necessary to validate the appropriateness of modelling the delay as stochastic.

Consider the following example. Assume a task set consisting of two tasks defined as

Task	T_i	C_i	Priority
τ_1	0.24	0.12	High
τ_2	0.3	0.12	Low

where task τ_2 implements a LQG-controller with sampling period, $T_2 = 0.3$, controlling an inverted pendulum process modelled by the Laplace-transfer function $P(s) = 1/(s^2 - 1)$. The continuous-time input noise has the covariance $R1_c = 1$ and the discrete-time measurement noise has the covariance $R2 = 0.01$. The cost function that the controller should minimize is given by Equation (2), with $\rho = 0.001$ and

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 0.01 \end{pmatrix}.$$

The schedule of the tasks is illustrated in Figure 1. The schedule repeats every *hyperperiod*, that is the least

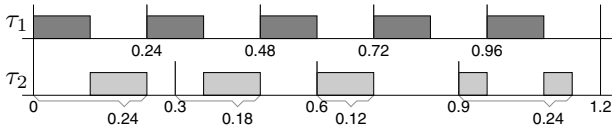

Fig. 1. Schedule of the two control tasks.

common multiple of the task periods (1.2 for these tasks). The task execution times are assumed to be constant and given by C_i.

As it can be noticed in the figure, the response time for task τ_2 will have the following repetitive cycle $R_2 = 0.24, 0.18, 0.12, 0.24, \ldots$ (the interested reader can find more details in the work by Lehoczky (1990)). Hence, the worst-case response time is 0.24, the best-case response time is 0.12, and the average-case response time is 0.195.

We now compare the costs for four different LQG-controllers:

- LQG_B The controller is designed assuming a constant delay equal to the best-case response time.
- LQG_A The controller is designed assuming a constant delay equal to the average-case response time.
- LQG_W The controller is designed assuming a constant delay equal to the worst-case response time.
- LQG_S The controller is designed assuming that the delay is governed by a stochastic variable with a discrete-time distribution function corresponding to the above cycle, i.e., the likelihood is 50% that the delay is equal to 0.24, 25% that it is equal to 0.18, etc.

The costs will be evaluated for two execution scenarios:

- E_S The delay is a stochastic variable with the distribution function defined previously.
- E_D The delay varies from job to job according to the deterministic repetitive cycle defined previously. This scenario corresponds to the true execution according to the schedule.

The costs for the eight cases are given below.

Controller	E_S	E_D
LQG_B	0.6561	0.7112
LQG_A	0.5959	0.6200
LQG_W	0.6413	0.6169
LQG_S	0.5891	0.6125

In this example, for both execution scenarios the controller designed using the delay distribution gives the lowest cost, which speaks in favour of the approach adopted in this paper. Also, the cost for the true deterministic execution scenario is quite close to the cost for the stochastic execution scenario. This also speaks in favour of the proposed approach. Furthermore, when as in the current case the task periods are derived through numerical optimization it is unlikely that there will exist any hyperperiod, i.e., there is no repeating pattern for the delay. As will be shown by the evaluations presented in the following the error introduced by assuming a stochastic delay is in general quite minor.

3.2 Computation of the response time distribution

In Section 3.1 we showed that adopting the response time distribution as model for delay provides results that are quite similar to the ones obtained by considering the exact pattern of job delays. However, to best of our knowledge, today there is no analytical method, which provides the response time distribution as a function of the task parameters. Hence, we can only proceed by simulation.

Our simulation-based computation of the response time distribution must necessarily use a finite number of jobs for which the response time is computed. Next, we investigate the impact of the number of jobs on the accuracy of the response time distribution.

Let $R(k)$ denote the random variable of the job response time extracted among the first k jobs, and $F_{R(k)}(r)$ its cumulative distribution function (CDF). With this notation in mind and for the purpose of measuring how sensitive the distribution of $R(k)$ is to k, we define the following *incremental normalized distance*:

$$d(k) = \frac{1}{T} E\{|R(k) - R(k+1)|\} \tag{5}$$

with $E\{\cdot\}$ denoting the mathematical expectation, and T being the period of the task. Intuitively, $d(k)$ represents the diversity of $R(k+1)$ w.r.t. $R(k)$. The factor $1/T$ is added to properly normalize such a distance. We observe that $d(k)$ is zero if and only if $R(k)$ and $R(k+1)$ are coincident almost everywhere. Moreover, we have

$$\lim_{k\to\infty} d(k) = 0$$

since as k grows, the two random variables $R(k)$ and $R(k+1)$ tend to coincide.

Such a quantity $d(k)$ can be used to determine the number of jobs after which the simulation can be stopped. In fact, if $d(k)$ is small it means that adding new samples of the job response time is not going to affect significantly the new response time distribution. As shown by Vallender (1974), the value $d(k)$ of (5) can be computed as

$$d(k) = \int_{-\infty}^{\infty} |F_{R(k)}(r) - F_{R(k+1)}(r)|\, dr. \tag{6}$$

In Figure 2, we report the value $d(k)$ computed for the response time of the lowest priority task, as the number of jobs increases. The plot is reported for several task sets, each denoted in the format $[(C_1, T_1), (C_2, T_2), \ldots, (C_n, T_n)]$ in Fig. 2. Observe that when the task periods are integers

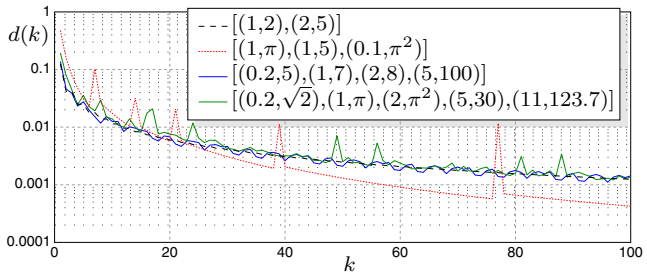

Fig. 2. $d(k)$ as a function of the number k of jobs.

(such as in the 1-st and 3-rd case) the distribution of the response time does not vary roughly over time. Instead, it is interesting to notice that when any pair of periods is incommensurable (such as in the 2-nd and 4-th cases) the response time of some job does not conform to the distribution computed until that point in time.

Finally, we remark that the goal of this paper is not to determine the most accurate possible distribution of the task response time. Instead, we aim at determining a response time distribution, which models the task delay well enough to allow an improved selection of the task periods. As we will show later in the experiments, selecting the first 100 jobs for computing the response time distribution is a suitable choice for determining task periods which more aggressively reduce the control cost.

4. PERIOD ASSIGNMENT

The period selection problem can be formulated as follows:

- find the controller periods $[T_1, \ldots, T_n]$,
- such that the cost

$$J = \sum_{i=1}^{n} J_j, \tag{7}$$

  with J_i defined in (2), is minimized, and
- the set of controllers is feasible, that is

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1. \tag{8}$$

Since the cost J_i of (2) has no explicit form and can only be computed numerically (through Jitterbug, in our case), we solve this problem with numerical optimization. Below we provide more details for the optimization procedure.

4.1 Choice of the initial periods

Since the problem is non-convex, the choice of the initial solution (initial periods in our case) may affect the final solution. Three methods to obtain initial values were evaluated:

- equal periods, that is setting $\forall i,\ T_i = \sum_j C_j$,
- even utilization assignment, that is setting $T_i = nC_i$, and
- random initialization.

However, all three methods gave initial periods that were far from the optimal solution.

Instead the delay-aware period assignment method (Ri-Approx) from Bini and Cervin (2008) was used to provide

initial task periods. In this approach a linear approximation of the cost function $J = \sum \alpha_i T_i + \beta_i \Delta_i$ is minimized assuming that the delay Δ_i equals an approximation of the average response time.

4.2 Cost calculation

The change of T_i affects the response time distribution of task j when $j \geq i$. And the change of response time distribution affects the cost of the current task. So the overall cost J is a function of each period T_i of tasks. For the given computation time C_i and period T_i of each task with fixed priority, the differentiation process for the overall J with respect to each T_i consists of the following steps:

(1) Compute the response time distribution of each task i using the method in Section 3.2;
(2) Design the LQG controller using the response-time distribution as the delay distribution, for each task i using the method in Section 2.2;
(3) Calculate the LQG cost J_i of each task i;
(4) Calculate the overall cost J by (7).

For the example in 3.1, we plotted the overall cost J as a function of periods T_1 and T_2 (Fig.3). The function is non-linear and non-convex, even though there are only two tasks.

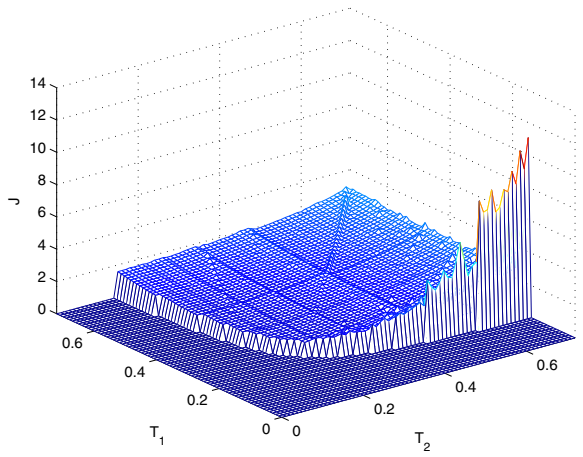

Fig. 3. J as a function of periods T_1 and T_2.

4.3 Local optimal solution

Gradient based optimization methods could be used to find the local optimal overall cost J, but they are time-consuming. So instead, we introduce a sequential search-based optimization method, which is derivative free. The method is presented in Algorithm 1. Since low priority tasks are disturbed by interference from high priority tasks the algorithm starts by finding the optimal periods for higher priority tasks and fix them, before proceeding to lower priority tasks. Since larger periods of high priority tasks give more utilization to the low priority tasks, the algorithm begins by trying larger periods and then smaller ones. The utilization requirement $U \leq 1$ is checked before assignment of smaller periods. This heuristic approach is called sequential search in the sequel.

Algorithm 1 Local optimization

procedure SEQUENTIALSEARCH($T_1, T_2, ..., T_n$)
  calculate J at the initial value $T_1, T_2, ..., T_n$
  for $i = 1 : n$ **do**
    $T_i' \leftarrow T_i + \Delta T_i$
    calculate J' at $T_1, T_2, ..., T_i', .., T_n$
    $moved \leftarrow 0$
    while $J' < J$ **do**
      $T_1 \leftarrow T_i'$
      $J \leftarrow J'$
      $T_i' \leftarrow T_i + \Delta T_i$
      calculate J' at $T_1, T_2, ..., T_i', .., T_n$
      $moved \leftarrow 1$
    end while
    if $moved = 0$ **then**
      $T_i' \leftarrow T_i - \Delta T_i$
      $U \leftarrow \sum \frac{C_i}{T_i'}$
      if $U \leq 1$ **then**
        calculate J' at $T_1, T_2, ..., T_i', .., T_n$
        while $J' < J$ **do**
          $T_1 \leftarrow T_i'$
          $J \leftarrow J'$
          $T_i' \leftarrow T_i - \Delta T_i$
          calculate J' at $T_1, T_2, ..., T_i', .., T_n$
        end while
      end if
    end if
  end for
  return $T_1, T_2, ..., T_n$ ▷ The local minimum
end procedure

4.4 Global optimal solution

Our interest here is not only in the locally optimal solution. According to the characteristics of the LQG cost as a function of periods, we are certain to say it is a typical global optimization problem. So several optimization methods could be applied for its solution. As mentioned, the derivative-based optimization method were avoided, since they are very time-consuming in the current problem.

The DIRECT optimization method (Jones et al., 1993) is a sampling-based algorithm. It is a global derivative-free method. In the given domain, DIRECT optimization samples some points without initial value, then decides where to do the next search based on the known information. It converges to a global minimum value. It is widely used in 'black box' problem, in which the relation between inputs and outputs is complex.

5. EVALUATION

In this section the proposed response time driven period assignment method is evaluated comparing with the delay-aware period assignment method (RiApprox) from Bini and Cervin (2008). To better understand the properties of the proposed period assignment method, tasks sets are randomly generated to control randomly generated plants with varying characteristics. The evaluation is performed using the TrueTime toolbox.

5.1 Simulation setup

Plant sets were randomly generated from the following three different plant families.

- Family I: All plants have two stable poles and each plant is drawn from $P_1(s)$ and $P_2(s)$ with equal probability where
$$P_1(s) = \frac{1}{(s+a_1)(s+a_2)}$$
$$P_2(s) = \frac{1}{s^2+2\zeta\omega s+\omega^2}$$
with $a_1, a_2 \in U(0,1)$, $\omega \in U(0,1)$, $\zeta \in U(0,1)$.
- Family II: All plants have two stable or unstable poles, with each plant drawn with equal probability from
$$P_3(s) = \frac{1}{(s+a_1)(s+a_2)}$$
$$P_4(s) = \frac{1}{s^2+2\zeta\omega s+\omega^2}$$
with $a_1, a_2 \in U(-1,1)$, $\omega \in U(0,1)$, $\zeta \in U(-1,1)$.
- Family III: All plants have three stable or unstable poles, with each plant drawn with equal probability from
$$P_5(s) = \frac{1}{(s+a_1)(s+a_2)(s+a_3)}$$
$$P_6(s) = \frac{1}{(s^2+2\zeta\omega s+\omega^2)(s+a_3)}$$
with $a_1, a_2, a_3 \in U(-1,1)$, $\omega \in U(0,1)$, $\zeta \in U(-1,1)$.

For the LQG controllers, $\rho = 0.01$, $R_{1c} = BB^T$, and $R_2 = 0.01\text{tr}\{R_{1c}\}$ were used. Robust LQG controllers were designed with the assigned period T and response time distribution. The LQG cost is computed using the same LQG parameters.

The evaluation examined systems of $n \in \{3, 5\}$ control tasks. The nominal task utilization U_i^{nom} were generated using an n-dimensional uniform distribution with total utilization 1. The execution time was given by $C_i \in U(0.04, 0.4)/n$. The task priorities were assigned based on the periods returned by Seto et al. (1996).

In the optimization procedure, the cost values and the LQG controllers were calculated using the Jitterbug toolbox. However, Jitterbug can only design LQG controllers when the delay variation is less than the period. Therefore the obtained delay distributions were truncated from below if the delay variation was larger than the period. The truncation was performed in such way that the average delay remained the same before and after the truncation. The upper limit was set to worst-case response time (R_i^{wc}), and the lower limit was set to $R_i^{\text{wc}} - T_i$.

In general, the stability of the plants under control decides the sensitivity towards delay and jitter. Therefore, it can be expected that the cost obtained for Family I has the smallest value, while the cost for Family III has the highest value. This is also shown by the evaluation.

5.2 Evaluation results

The RiApprox method was used to compute the initial periods. The different period assignment methods were evaluated by Monte Carlo simulation, where the plants (including the disturbances), the controllers, and the scheduler were simulated in parallel using TrueTime. From each family of plants, 10 random plants were generated. After the period assignment the plants and controllers were simulated for 1000 s, and total cost, J, was recorded.

	$n=3$	$n=5$
RiApprox	4.0519	5.3972
Initial	4.0110	5.3415
Sequential search	3.8557	5.2496
DIRECT optimization	3.8552	5.2300

Table 1. Evaluation of the cost: Family I

	$n=3$	$n=5$
RiApprox	19.5711(1)	7.3223(3)
Initial	5.2573	7.0249
Sequential search	4.9206	6.6345
DIRECT optimization	4.9206	6.6310

Table 2. Evaluation of the cost: Family II

	$n=3$	$n=5$
RiApprox	15.0433(1)	24.7611(3)
Initial	14.3951	23.2661
Sequential search	12.3179	20.0072
DIRECT optimization	12.3179	19.8083

Table 3. Evaluation of the cost: Family III

The utilization requirement $U \leq 1$ should be always fulfilled. In sequential search algorithm, when the step has been changed, the utilization is checked to avoid the situation that the utilization is larger than 1. In DIRECT optimization algorithm, if the utilization is more than 1, the cost is set to a very large number to enforce the optimization result away from this situation.

The evaluation results are based on $n \in \{3, 5\}$ plants and controllers. In each family, the cost values are compared for four period assignment methods:

- RiApprox by Bini and Cervin (2008), in which the LQG controllers are designed for a constant delay and the task periods are obtained from a linear approximation of the cost function;
- Initial, in which the task periods obtained from RiApprox are used but the LQG controllers are redesigned based on the response time distributions;
- Sequential Search is the local search algorithm described in Algorithm 1, initialized with the periods computed in the previous method; and
- DIRECT optimization that is the global optimization approach proposed by Jones et al. (1993) and available in Matlab.

The total costs for the three plant families are shown in Tables 1, 2, and 3.

For Family III the RiApprox method sometimes gives rise to unstable control loops, i.e., infinite cost. Therefore, the total cost is divided by the number of plants for which the cost is finite. In parentheses, the number of times (out of 10) that RiApprox gives an infinite cost is shown.

The evaluation shows that the response time driven period assignment methods gives better LQG performance than the RiApprox method in all cases. The difference is larger

		Sequential search	DIRECT method
Family I	$n = 3$	8.2912	114.2827
	$n = 5$	18.4958	198.2194
Family II	$n = 3$	3.4655	123.5941
	$n = 5$	11.9846	215.4092
Family IIII	$n = 3$	5.0514	128.8757
	$n = 5$	14.4181	266.7887

Table 4. Run-time of the methods.

when the task set is large or when the plants are unstable. The sequential search-based method gives results that in all cases are very close to the results obtained by the global DIRECT method.

However, the sequential search-based method requires much fewer computations than the DIRECT method. In order to evaluate this, for each plant family, n plants were randomly generated and n corresponding controllers were designed. The computation times of Sequential search and DIRECT optimization were compared using the `tic` and `toc` method in Matlab. The computation times are shown in Table 4 As can be seen difference is approximatively a factor of 2.

6. CONCLUSIONS AND FUTURE WORKS

In this paper, we have introduced response time driven design of LQG controllers. The major contribution of the paper is the proposed period assignment method based on the task response time distribution and the sequential search-based algorithm. Through a simulation-based evaluation it was shown that the method may lead to very good results. The control performance, as measued by the LQG cost, was improved compared to previous approaches and the local search-based heuristic algorithm gave results that were very close to the global optimum, with 50% less computation effort compared to the global optimization method.

In future work, the task priority assignment will be integrated with the period assignment. Also, the Earliest-Deadline-First (EDF) scheduling case, which is really relevant for safety-related control systems, will be investigated.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the suggestion to adopt the Wesserstein's distance to compare the distribution of two random variables, by Giacomo Como.

REFERENCES

Amir Aminifar, Soheil Samii, Petru Eles, Zebo Peng, and Anton Cervin. Designing high-quality embedded control systems with guaranteed stability. In *Proceedings of the* 33^{rd} *IEEE Real-Time Systems Symposium*, pages 283–292, San Juan, Puerto Rico, 2012.

Karl Johan Åström and Björn Wittenmark. *Computer-Controlled Systems. Theory and Design.* Prentice Hall, third edition, 1997.

Enrico Bini and Anton Cervin. Delay-aware period assignment in control systems. In *Proceedings of the* 29^{th} *IEEE Real-Time Systems Symposium*, pages 291–300, Barcelona, Spain, December 2008.

Enrico Bini and Marco Di Natale. Optimal task rate selection in fixed priority systems. In *Proceedings of the* 26^{th} *IEEE Real-Time Systems Symposium*, pages 399–409, Miami (FL), U.S.A., December 2005.

Anton Cervin, Dan Henriksson, Bo Lincoln, Johan Eker, and Karl-Erik Årzén. How does control timing affect performance? *IEEE Control Systems Magazine*, 23(3): 16–30, June 2003.

Dan Henriksson and Anton Cervin. Optimal on-line sampling period assignment for real-time control tasks based on plant state information. In *Proceedings of the* 44^{th} *IEEE Conference on Decision and Control and European Control Conference*, Seville, Spain, December 2005.

Donald R. Jones, Cary D. Perttunen, and Bruce E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

Byung Kook Kim. Task scheduling with feedback latency for real-time control systems. In *Proceedings of the* 5^{th} *International Workshop on Real-Time Computing Systems and Applications*, pages 37–41, Hiroshima, Japan, October 1998.

John P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadline. In *Proceedings of the* 11^{th} *IEEE Real-Time Systems Symposium*, pages 201–209, Lake Buena Vista (FL), U.S.A., December 1990.

Bo Lincoln and Bo Bernhardsson. Optimal control over networks with long random delays. In *Proceedings of the International Symposium on Mathematical Theory of Networks and Systems*, January 2000.

Bo Lincoln and Anton Cervin. Jitterbug: A tool for analysis of real-time control performance. In *Proceedings of the* 41^{st} *IEEE Conference on Decision and Control*, Las Vegas, NV U.S.A., December 2002.

Chung Laung Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.

Pau Martí, Caixue Lin, Scott A. Brandt, Manuel Velasco, and Josep M. Fuertes. Optimal state feedback based resource allocation for resource-constrained control tasks. In *Proceedings of the* 25^{th} *IEEE Real-Time Systems Symposium*, pages 161–172, Lisbon, Portugal, December 2004.

Johan Nilsson, Bo Bernhardsson, and Björb Wittenmark. Stochastic analysis and control of real-time systems with random time delays. *Automatica*, 34(1):57–64, January 1998.

Soheil Samii, Anton Cervin, Petru Eles, and Zebo Peng. Integrated scheduling and synthesis of control applications on distributed embedded systems. In *Proc. Design, Automation & Test in Europe (DATE)*, April 2009.

Danbing Seto, John P. Lehoczky, Lui Sha, and Kang G. Shin. On task schedulability in real-time control systems. In *Proceedings of the* 17^{th} *IEEE Real-Time Systems Symposium*, pages 13–21, Washington, DC, USA, December 1996.

S. S. Vallender. Calculation of the Wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications*, 18(4):784–786, 1974.