Exploiting Job Response-Time Information in the Co-Design of Real-Time Control Systems

Yang Xu*, Karl-Erik Årzén*, Anton Cervin*, Enrico Bini†, Bogdan Tanasa‡
*Department of Automatic Control, Lund University, Sweden
Email: yang, karlerik, anton@control.lth.se
†Real-Time Systems Laboratory, Scuola Superiore Sant'Anna, Italy
Email: e.bini@sssup.it
‡Department of Computer and Information Science, Linköping University, Sweden
Email: bogdan.tanasa@liu.se

***Abstract*—We consider a real-time system of multiple tasks, each task having a plant to control. The overall quadratic control cost is to be optimized. We exploit the periodicity of the task response time, which corresponds to a periodic delay pattern in the feedback control loop. Perturbed periods are used as a tool to find a finite hyperperiod. We present an analytical procedure to design a periodic linear-quadratic-Gaussian (LQG) controller for tasks with fixed execution times as well as a numerical solution to the periodic–stochastic LQG problem for tasks with variable execution times. The controllers are evaluated using simulations in real-time scheduling and control co-design examples.**

I. Introduction

A. Motivation

Scheduling–control co-design is a fundamental problem that emerges from the combination of real-time computing and feedback control. Most real-time systems are developed for the aim of digital control. On the other hand, many control systems are implemented on top of real-time operating systems. The real-time systems parameters, such as task utilizations and deadlines, affect the periods and latencies of the control systems, and hence their control performance. The goal of control–scheduling co-design is to optimize the combined performance of all control tasks in the system, subject to a schedulability constraint. This is done by assigning suitable task parameters and designing feedback controllers that take the scheduling parameters into account.

In this paper, we focus on the response-time variation that arises from preemptive scheduling of a set of periodic tasks. Assuming constant execution times for all tasks, the response times will have a periodic regularity in cases where a hyperperiod exists. (Recall that the hyperperiod is the least common multiplier of the periods of all tasks.) If the response-time pattern is known, it can be exploited in the control design.

In reality, task execution times are not constant but may vary due to cache misses, unmodeled hardware interrupts, etc. We can model the execution time of each job of a control task as an independent random variable. The response time of each job will no longer be constant but can be characterized by a probability distribution. There is now a repetitive pattern of the response time distribution over the hyperperiod, and this can also be exploited in the control design.

The specific scenario considered in this paper is a uniprocessor with fixed-priority scheduling running a set of tasks where each task implements a controller for some physical plant. It is assumed that the plants and, hence, the controllers, are independent of each other. The goal is to select the task periods and design the controllers so that the global control performance is maximized. The performance of each control loop is measured by a quadratic cost function of the process states and the control signal. The global performance is simply the sum of the cost functions for the individual control loops, which then should be minimized. This choice of performance metrics leads naturally to LQG control since the design goal in LQG design is the minimization of a quadratic cost function.

B. Related Work

Much research has focused on the control–scheduling co-design problem. The seminal paper [1] presented an algorithm that optimizes controller task periods based on a cost function and then schedules the resulting tasks with the limited computing resources available. In [2], the importance of also considering the control delay is pointed out, and an iterative search algorithm is suggested to assign control task periods. [3] provided a procedure that finds the task activation rates that maximize a performance function within the deadline constraints in systems scheduled using fixed priorities. [4] presented an integrated approach for designing high-quality embedded control systems, while guaranteeing their stability in the worst case.

In [5] the authors of this paper solved the optimal period assignment problem assuming controller costs that depend linearly on the delay and where the delay is assumed to be constant and is estimated using an approximate response-time analysis. In the continuation paper, [6], the delay was instead modeled by the statistical distribution of the task response times, and then an optimization based approach was used find the task periods for controllers designed using stochastic LQG techniques. Periodic LQG control design in real-time systems has been considered before [7], but for the case of variable sampling intervals and not for varying delays.

For joint ECU and bus scheduling in mixed-criticality systems, [8] formulated a integer linear programming problem to optimize linear or quadratic control performance functions. For controller rate selection in wireless networks, [9] formulated a constrained non-linear optimization problem and use simulated annealing to find a near-optimal solution. Ripoll and Ballester-Ripoll [10] investigated the problem of selecting

the task periods within some given intervals, such that the corresponding hyperperiod is minimized.

C. Contributions

The approach taken in current paper is to exploit the periodic delay pattern that results if the task periods are perturbed slightly, obtaining a finite hyperperiod. This is combined with so called periodic LQG design techniques, where the periodic delay pattern is taken explicitly into account in the control design. The contributions of this paper are the following:

- We introduce a method to perturb task periods in order to achieve a short and finite hyperperiod.
- We perform job response time analysis to discover the delay distribution, useful for the control design.
- We propose an analytical method for *periodic* LQG control design, suitable for tasks with constant execution times.
- We propose a numerical method for *periodic–stochastic* LQG control design, suitable for tasks with random execution times.
- We conduct evaluations on control–scheduling problems using the Jitterbug and TrueTime toolboxes [11].

D. Outline

The outline of the rest of this paper is as follows: The real-time control system model is presented in Section II. Section III gives a method to achieve a finite hyperperiod. Assuming this method, a periodic LQG control design procedure is given in Section IV. Section V evaluates the periodic LQG control design on three different examples. Section VI discusses statistical response-time analysis vs schedule simulations to find the response-time probability distributions. In Section VII, a numerical method to design a periodic–stochastic LQG controller is presented. Section VIII evaluates the periodic–stochastic LQG control design on three different examples. Finally, Section IX offers some concluding remarks.

II. Real-Time Control System Model

A. Control Task Model

We consider a real-time system composed of n control tasks. The tasks execute on a single processor under fixed priority preemptive scheduling. Each task is characterized by the following parameters:

- The *worst-case execution time* C_i is the maximum length of time the task could take to execute.
- For the *actual execution time* of each job of the task, we will consider two different cases:
  1) The execution time is constant and equal to C_i.
  2) The execution time of each job is statistically independent and drawn from a probability distribution with maximum value C_i.
- The *period* T_i of the task is the time between the release of two consecutive jobs of the task.
- The task *priority* is assumed to be implicitly assigned by the task ordering, such that the ith task has higher priority than the $(i+1)$th task.

There are further characteristics of the tasks, which depend on the above mentioned parameters:

- The *response time* R_{ij} of the jth job of task i is the time that elapses from the job release time to its finishing time.
- The task *utilization* $U_i = C_i/T_i$ measures the worst-case amount of computational resources required by the controller. We denote the *total utilization* of all control tasks by $U = \sum_i^n U_i$.

We assume throughout the paper that the task phasing is known, and that all tasks are released simultaneously at time zero. However, the proposed approach would also work for other phasings.

Each control task implements a feedback controller, for which we define the following timing parameters:

- The *sampling interval* h_{ij} is the time difference between the sampling operation of job j and job $j-1$ of task i. We assume that sampling always is performed at the job release time; hence, $h_{ij} = T_i$, $\forall j$.
- The *delay* $\tau_{i,j}$ is the time interval between the sampling operation and the output operation of job j of task i. We assume that the output operation is performed when the job finishes; hence, $\tau_{i,j} = R_{i,j}$. This is the main bridge between real-time theory and control theory in this paper.

B. Plant Model and Co-Design Problem Formulation

Each control task i controls a linear time-invariant (LTI) continuous time plant

$$\begin{aligned} \dot{x}_i(t) &= A_i x_i(t) + B_i u_i(t) + v_i(t) \\ y_i(t_k) &= C_i x_i(t_k) + e_i(t_k) \end{aligned} \tag{1}$$

where $x_i(t)$ is the state vector of the plant, $u_i(t)$ is the control input, $y_i(t)$ is the system output, and A_i, B_i, C_i are constant matrices. The disturbance $v_i(t)$ is continuous time white noise, while the measurement noise $e_i(t)$ is discrete time white noise.

The co-design problem involves assigning task periods $[T_1, \ldots, T_n]$ yielding a total utilization $U = \sum_{i=1}^n U_i \leq 1$ and designing n LQG controllers in order to minimize the total cost

$$J = \sum_{i=1}^{n} J_i$$

where

$$J_i = \int_0^\infty \begin{pmatrix} x_i^T(t) & u_i^T(t) \end{pmatrix} Q_{ci} \begin{pmatrix} x_i(t) \\ u_i(t) \end{pmatrix} dt \tag{2}$$

and where $Q_{ci} = \begin{pmatrix} Q_{1ci} & Q_{12ci} \\ Q_{12ci}^T & Q_{2ci} \end{pmatrix}$ is a symmetric positive definite matrix. The cost J is defined to penalize state deviation and control effort. The LQG controllers may have time-varying parameters but are to be designed off-line.

III. Task Period Perturbation to Achieve a Finite Hyperperiod

Since the task periods are not necessarily integers, we extend the definition of the hyperperiod of the set of tasks, as follows.

TABLE I: Example of approximate hyperperiod $\hat{H}$ when $T_1 = \sqrt{2} \approx 1.4142$ and $T_2 = \pi \approx 3.1416$.

ϵ	$\hat{H}$	$[k_1, k_2]$	$[\hat{T}_1, \hat{T}_2]$
0	∞	$[\infty, \infty]$	$[\sqrt{2}, \pi]$
0.001	28.284	$[20, 9]$	$[1.4139, 3.1420]$
0.1	9.8995	$[7, 3]$	$[1.3690, 3.1943]$
1	$\max_i\{T_i\} = \pi$	$[1, 1]$	$[2.5658, 2.5658]$

Definition 1: The *hyperperiod* of a set of tasks with periods $[T_1, \ldots, T_n]$, is the smallest $H > 0$ such that

$$\forall i, \ \exists k_i \in \mathbb{N} : \quad k_i T_i = H. \tag{3}$$

Since we assume that the initial task periods are any real numbers, the hyperperiod H of (3) may not exist. In this case we set it to $H = \infty$. We observe that the above definition does not necessarily require the task periods to be integers. For example, if $T_1 = 2\sqrt{2}$ and $T_2 = 3\sqrt{2}$, then according to Definition 1 we have that the hyperperiod of $[T_1, T_2]$ is $H = 6\sqrt{2}$. However, Definition 1 requires the periods to be commensurate: if $T_1 = 1$ and $T_2 = \sqrt{2}$, then there is no hyperperiod H satisfying (3), and we set $H = \infty$.

When the hyperperiod H is large or infinity, it is not practical or feasible to investigate the job response-time pattern over the hyperperiod. We therefore propose a method to perturb the periods to obtain a finite and short hyperperiod.

A. Finding an Approximate Hyperperiod

When the periods are the solution of an optimization problem such as in control-scheduling co-design, it may indeed happen that the task periods may be real values or, at least, machine-representable "real" values. In these cases it may be useful to find a suitable substitute for the hyperperiod. We propose the following definition.

Definition 2: Given a tolerance $\epsilon \in (0, 1)$, let $[k_1, \ldots, k_n] \in \mathbb{N}^n$ be such that

$$1 - \frac{\min_i\{k_i T_i\}}{\max_i\{k_i T_i\}} \leq \epsilon. \tag{4}$$

The *approximate hyperperiod* $\hat{H}$ of the task periods $[T_1, \ldots, T_n]$ is then

$$\hat{H} = \max_i\{k_i T_i\}. \tag{5}$$

As $\epsilon \to 0$, the approximate hyperperiod of Definition 2 tends to the actual hyperperiod of Definition 1, if the limit exists.

In Table I we illustrate an example when $T_1 = \sqrt{2}$ and $T_2 = \pi$. The third column reports the values $[k_1, k_2]$ that makes Eq. (4) true. It can be observed that, as the tolerance ϵ increases the corresponding approximate hyperperiod $\hat{H}$ decreases.

B. Control Task Period Assignment

Let us assume that an optimization-based control-scheduling co-design has been performed, e.g., using the method in [5] or [6]. This leads to a set of real-valued task periods, $\mathbf{T} = [T_1, \ldots, T_n]$, that give good control performance but for which, typically, no finite hyperperiod exists. Also such a solution is always fully utilizing the processor, that is:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} = 1. \tag{6}$$

The goal we have is to find some other task periods $\hat{\mathbf{T}} = [\hat{T}_i, \ldots, \hat{T}_n]$ which are "close" to the original values and which have a finite hyperperiod that is not too large.

We propose the following method for period assignment:

1) Set a value of ϵ of desired proximity between $\mathbf{T}$ and $\hat{\mathbf{T}}$. A typical value could be between 10^{-3} and 10^{-2}.
2) Compute the set of integers $[k_1, \ldots, k_n]$ such that (4) holds;
3) Calculate the modified periods as
$$\hat{T}_i = \frac{\sum_{j=1}^{n} k_j C_j}{k_i} \tag{7}$$
This choice implies that even the tasks with the modified periods $\hat{T}_i$ are fully utilizing the processor.
4) Redesign the controllers taking the obtained periodicity explicitly into account using a periodic LQG control design scheme. The results of this will be that for each controller the controller parameters will depend on the current job in the hyperperiod.

For the simple example of Table I, if $C_1 = \sqrt{2}/3$ and $C_2 = 2\pi/3$, which implies that $\sum_i \frac{C_i}{T_i} = 1$, then the task periods $\hat{T}_i$ modified according to (7) are reported in the fourth column.

By tuning ϵ we can control the length of the hyperperiod and then the length of the pattern of job response times. Also, ϵ is used to control the magnitude of the perturbation of the original periods. From (4), it follows that

$$\forall i, \quad (1-\epsilon)\frac{\max_j\{k_j T_j\}}{T_i} \leq k_i \leq \frac{\max_j\{k_j T_j\}}{T_i},$$

which enables to find lower and upper bounds to the perturbed $\hat{T}_i$ of (7), as follows

$$\forall i, \quad (1-\epsilon) T_i \sum_{j=1}^{n} U_j \leq \hat{T}_i \leq \frac{1}{1-\epsilon} T_i \sum_{j=1}^{n} U_j.$$

If the original periods $[T_1, \ldots, T_n]$ are fully utilizing the processor, which is always the case if they are the solution of an optimal real-time control co-design problem [5], [6], then the bounds to $\hat{T}_i$ become

$$\forall i, \quad (1-\epsilon) T_i \leq \hat{T}_i \leq \frac{1}{1-\epsilon} T_i. \tag{8}$$

Eq. (8) provides an insightful interpretation of ϵ. It states that by perturbing the periods according to (7), with k_i defined by (4), then the amount of perturbation can be controlled by ϵ.

In the real-time control co-design problem, selecting ϵ is a trade-off between the tolerable variation of the task periods from the solution of the continuous optimization problem and the length of the hyperperiod. A long hyperperiod requires more memory to store the controller parameters and a larger (off-line) computational effort to design the controllers.

IV. Periodic LQG Control Design for Deterministic Job Response Times

In a hyperperiod, fixed actual execution times implies that the delays are known, but variable. This section establishes the LQG control design for the resulting periodic system. The modified hyperperiod proposed in the previous section makes it possible to realize this design procedure. Note that existing LQG design methods for time-delay systems do not apply because the delays are either assumed to be constant or in the form of a probability distribution. Here we allow the delays to vary according to a deterministic pattern over a hyperperiod.

A. Sampling the Periodic Time Delay System

For task i,

$$H = l_i h_i$$

where H is the hyperperiod, h_i is the period, and l_i is the number of jobs in one hyperperiod. The controlled plant is given by Equation (1). The delays arrive in the deterministic pattern $\tau_{i,1}$, $\tau_{i,2}$, ..., τ_{i,l_i}. The delays can be less than, equal to, or greater than period. We know however that $\tau_{i,j} < \tau_{i,j+1} + h_i,\ j \in \{1, 2, \ldots, l_i - 1\}$, because in a single-CPU system, the finishing times of jobs cannot be disordered. The control signal is assumed to be zero-order hold, i.e. piecewise constant between update instants (see Figure 1).

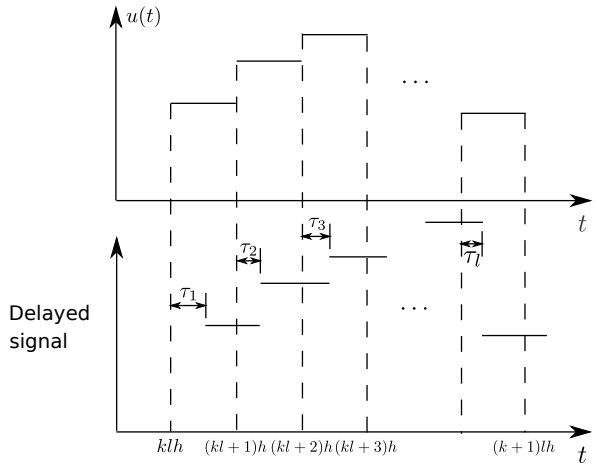


Fig. 1: Delayed control signals in one hyperperiod.

Because the dynamics repeats in every hyperperiod, we only need to investigate the state space model over one hyperperiod. Actually, if there exists a smaller contiguous repeating sub-sequence than the full sequence in the hyperperiod, the controller can be recalculated for the shorter length. The size of matrices in state space model is smaller and the cost reformulation is simpler. However, the controller is the same as the full hyperperiod design. For convenience, the subscripts indicating task i are omitted in the rest of this section.

The integration of Equation (1) over one hyperperiod is shown in Appendix A. Following this, an extended state-space model can be introduced,

$$\begin{pmatrix} x((k+1)lh) \\ u'(klh) \end{pmatrix} = \begin{pmatrix} \Phi & \Gamma_1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x(klh) \\ u'((k-1)lh) \end{pmatrix} + \begin{pmatrix} \Gamma_0 \\ I \end{pmatrix} u'(klh)$$

Here, l control signals over the hyperperiod are included in the extended state. This reformulation changes the system from continuous time infinite form to discrete time finite form.

B. Sampling the Loss Function

We have so far sampled the plant model, but we also need to sample the cost function. The cost of the ith task is

$$J_i = \sum_{k=0}^{\infty} J_i(k) = \sum_{k=0}^{\infty} J_i(kl_i h_i)$$

$$\begin{aligned} J(klh) &= \int_{klh}^{(k+1)lh} x^T(t)Q_{1c}x(t) + 2x^T(t)Q_{12c}u(t) \\ &\quad + u^T(t)Q_{2c}u(t)dt \\ &:= x^T(klh)Q_1x(klh) + 2x^T(klh)Q_{12}u'(klh) \\ &\quad + u'^T(klh)Q_2u'(klh) + J_v(lh) \end{aligned}$$

The discrete time cost matrices Q_1, Q_2 and Q_{12} are calculated as shown in Appendix B. Given the extended state space model, the standard linear quadratic control design method can be applied as shown in Appendix C.

V. Periodic LQG Control Evaluation

In this section we first apply the periodic LQG design procedure to a simple example and then to two real time scheduling and control co-design problems. In the simple example, the design approach will guarantee the correctness of periodic LQG design, without having to consider the exact multi-tasking behavior in the real-time system. The exact costs can be calculated using Jitterbug toolbox. The second example will illustrate how the perturbed period method works. The third evaluation example will provide a systematic approach for periodic LQG design in the real time multi tasks environment. The latter examples involve joint scheduling and control simulations and are evaluated using the TrueTime toolbox.

A. A Simple Example

Assume that a plant under control is given by

$$G(s) = \frac{1}{s^2 - 1}$$

The cost function is given by Equation (2), in which the continuous time cost matrix is

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

and $\rho = 0.01$. The continuous time state noise covariance matrix is $R_{1c} = BB^T$, and the discrete time measurement noise covariance matrix is $R_2 = 0.01\text{tr}\{R_{1c}\}$.

The sampling period is $h = 0.3$, and the hyperperiod $H = 4h$. The delays in one hyperperiod are assumed to be $\tau_1 = 0.24$, $\tau_2 = 0.18$, $\tau_3 = 0.12$, $\tau_4 = 0.24$.

LQG controllers are then designed by the following three methods:

- Constant delay: Using Jitterbug, a time-invariant LQG controller is designed assuming a constant delay equal to the average delay.
- Stochastic LQG: Using Jitterbug, a time-invariant LQG controller is designed, viewing the delay as a

random variable with known probability mass distribution. Riccati equation.

- Periodic LQG: Design a sequence of l LQG controllers using the periodic LQG control design procedure proposed in Section IV.

The costs evaluated in Jitterbug are given in Table II. The cost using periodic LQG is better than the other two. This is because it takes all the information about system dynamics, especially about delays, into account in control design procedure. The larger the delay variability, the larger the improvement over previous design techniques will be.

TABLE II: Costs in the simple example

	Cost
Constant delay design	3.0124
Stochastic LQG design	2.9528
Periodic LQG design	2.2194

B. Three-Plant Example

Assume a set of three tasks controlling three plants

$$P_1(s) = \frac{1}{s^2 + 0.549s - 0.1979}$$
$$P_2(s) = \frac{1}{s^2 - 0.9947s + 0.2366}$$
$$P_3(s) = \frac{1}{s^2 + 0.711s + 0.0252}$$

We start by assigning initial periods with the approach proposed in [5]. The initial periods are shown in Table III, where all periods have been rounded to four decimal places. The hyperperiod is infinite. Then we modify the periods by selecting the tolerance as $\epsilon = 0.1$ or $\epsilon = 0.05$. Now the hyperperiods are finite. As shown in Section III, the lower the ϵ value, the longer the hyperperiod. The utilization is kept at $U = 0.98$ in order to avoid excessive control delays.

TABLE III: Perturbed periods and costs

	T_1	T_2	T_3	H	J_{ini} [5]	J_{LQG}
Initial	0.2795	0.3509	0.2792	∞	-	-
$\epsilon = 0.1$	0.2740	0.3653	0.2740	1.0959	5.5697	5.2188
$\epsilon = 0.05$	0.2837	0.3404	0.2837	1.7021	5.5697	5.2061

We then perturb the periods to obtain a finite hyperperiod, calculate the actual job response times and redesign the controllers according to the periodic LQG method. The results are shown in Table III. The initial method from [5] uses a constant delay LQG design based on the approximate average response time. The periodic LQG method gives better results than the initial method, and even better performance when the tolerance ϵ is small. The latter implies a longer hyperperiod but enables a closer-to-optimal resource distribution.

C. Evaluation on Randomly Generated Plants

In this co-design example, we generate plants randomly 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 $a_1, a_2 \in \mathrm{U}(0, 1)$, $\omega \in \mathrm{U}(0, 1)$, $\zeta \in \mathrm{U}(0, 1)$.

- 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 $a_1, a_2 \in \mathrm{U}(-1, 1)$, $\omega \in \mathrm{U}(0, 1)$, $\zeta \in \mathrm{U}(-1, 1)$.

- 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 $a_1, a_2, a_3 \in \mathrm{U}(-1, 1)$, $\omega \in \mathrm{U}(0, 1)$, $\zeta \in \mathrm{U}(-1, 1)$.

We assume Q_c, $\rho = 0.01$, $R_{1c} = BB^T$, and R_2 to be the same as those in Section V-A.

The evaluation examines systems with $n = 3$ control tasks. The nominal task utilizations U_i^{nom} are generated randomly using an n-dimensional uniform distribution with total utilization 1. The worst-case execution time was drawn from $C_i \in \mathrm{U}(0.04, 0.4)/n$. The task priorities were assigned using rate-monotonic ordering based on the periods returned by [1].

The initial task periods are assigned by the method in [5] with utilization 0.99, and the initial controllers are designed based on a constant delay equal to the average response-time. To avoid numerical errors, the utilization is set to 0.99.

Then perturbed periods are calculated to get a finite hyperperiod, and the delays in a period are derived from a schedule simulation. Finally, a periodic LQG controller is designed for each plant.

The two control design methods are evaluated by Monte Carlo simulations, where the plants, the controllers, and the scheduler are simulated in parallel using TrueTime. From each family of plants, 10 random plants are generated. After the controller design, the plants and controllers are simulated for 1000 s, and the total cost, J, was recorded. All the costs are given in Table IV.

The cost with periodic LQG controller has lower value than cost with gain scheduling method. It is reasonable, because periodic LQG controller takes more information about system dynamics into account.

TABLE IV: Evaluation of costs for co-design for random plants

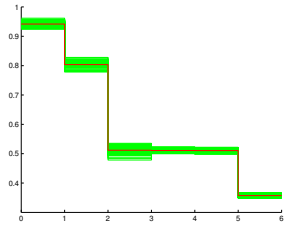
	Family I	Family II	Family III
Initial design [5]	3.23	4.56	15.41
Periodic LQG	3.18	4.41	12.66

D. Limitations of Periodic LQG Control

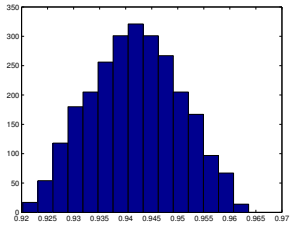
As shown in the previous evaluations, periodic LQG control gives good performance as long as the task execution times are constant. However, when the execution times vary, then a periodically repeating delay pattern no longer exists, and the performance obtained using the periodic LQG controller decreases.

The extent of this performance decrease depends on the shape of the delay distribution function, which in turn depends on the execution time distribution functions of the task under investigation and all higher-priority tasks. Consider the following examples. Assume that a certain task of low priority has six jobs in a hyperperiod. In Fig. 2a the job response times, i.e., delays, are plotted for a large number of hyperperiods. The x-axis shows the current job index in the hyperperiod and the y-axis shows the corresponding delay. The red line shows the average response time. Fig. 2b shows the response time distribution of the first job in the hyperperiod. With this narrow, unimodal shape of the distribution it is likely that a periodic LQG controller designed for the average delay in each job would still work reasonable well. However, in another example, using slightly different task parameters, the results may be as presented in Fig. 2c and Fig. 2d. Here the response-time distribution is bimodal and the average delay is not a good approximation of the true delay and it is less likely that a periodic LQG control will perform well.

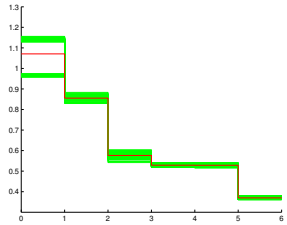
In the following sections we will investigate how a periodic–stochastic LQG control design can be used to cope with this problem. In this type of LQG controller the delay is modeled by a probability distribution for each job in the hyperperiod. Hence, rather than having delays that repeat periodically, we will have delay distributions that repeat periodically. In order to use this technique, the delay distribution for each job must be known at design time. This is studied next.



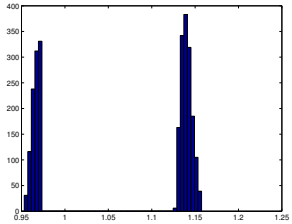
(a) Job response times over a hyperperiod—first example



(b) Response time distribution for job 1 in the first example



(c) Job response times over a hyperperiod—second example



(d) Response time distribution for job 1 in the second example

Fig. 2: Two examples showing different job response time distributions

VI. Calculation of the Job Response Time Distributions

There are two approaches to obtaining the job response time cumulative distribution functions. The first and most accurate way is to calculate them analytically using some statistical response time analysis tool. The alternative is to simulate the task schedule using real-time schedule simulator for sufficiently long time and measure the individual delays. Similar to measurement-based WCET analysis the latter approach always runs the risk of not encountering response times that have low probability. Here, both will be investigated. The response time analysis framework considered is [12].

A. A Framework for Probabilistic Response-Time Analysis

We now describe the framework used to accurately compute the response time distributions of tasks scheduled on uniprocessor platforms based on non-idling, preemptive scheduling policies [12]. With respect to scheduling policies, the framework is general and covers both static and dynamic scheduling algorithms like fixed priority scheduling and earliest deadline first scheduling.

Besides the scheduling policy, the framework takes as inputs the relevant tasks parameters, which include **i)** periods and **ii)** variable execution times. The main assumption on which the framework is built is that the tasks' jobs are independent from each other. Thus, independent random variables can be used to model the variable execution times of all jobs. As such, jobs belonging to the same task will be modeled with independent identically distributed random variables. The framework also assumes that **i)** the execution times of the tasks are described by probability density functions (PDFs) and that **ii)** the maximum CPU utilization does not exceed 100%. Given all these assumptions the framework accurately computes the response time distributions of the tasks by conducting a job-level analysis over all the jobs in the hyperperiod.

Internally, the framework tightly approximates all the PDFs with polynomial functions. This is because polynomial functions can be integrated analytically in polynomial time given their degrees. Also, for the uniform PDFs used in this paper, the polynomial approximations are exact. For each job in the hyperperiod, the framework identifies a set of non-idling scenarios where each such scenario covers a continuous interval of possible response times that the job under analysis might have. In this way, the response time distribution of the job under analysis can be evaluated by integrating the joint execution time function of those jobs that it interacts with (i.e. jobs with higher priority) over the points in the current interval. We illustrate the idea with the help of an example below.

TABLE V: Task parameters in the probabilistic response-time analysis example

Task Index	T	D	$C^{\min}$	$C^{\max}$
1	5	5	1	2
2	6	6	1	2
3	9	9	1	2

B. Response-Time Analysis Example

We provide an example describing the functionality of the tool used to compute the response time distributions of the tasks scheduled on uniprocessor platforms based on non-idling, preemptive scheduling policies. Let us assume a task set with three tasks. The tasks' parameters are presented in Table V and it is assumed that the jobs are scheduled with fixed priorities implied by task ordering and that the random execution times are uniformly distributed. Given the parameters, a total of 43 jobs are released in any hyperperiod. In the following we show the response time distribution of the 10^{th} job, corresponding to the second job of the third task.

The cumulative response time distribution of the 10^{th} job (see Figure 3) has been evaluated in two ways. First, we have conducted simulations based on the TrueTime simulator and, later, we have used the analytical tool. The red line depicts the response time distribution obtained from task simulations while the blue line denotes the analytical distribution. It can be seen that the two graphs match each other very well apart from one exception. The simulations have shown that the response time distribution of the 10^{th} job spreads over the interval $[2..3]\cup[4..6]$. On the other hand, the analytical tool reports that the same distribution has the domain $[2..3]\cup[4..6]\cup[7..9]$. Such discrepancies are likely to occur as, in our case, the probability for the response time of the 10^{th} to lie in the interval $[7..9]$ is very small (less then 0.01).

In this section, it has been shown that the response time distribution calculated using TrueTime schedule simulation is similar to the analytical solution. With this as motivation we will use TrueTime simulation to derive the response time PDFs in the following sections. The reason for not using the analytical calculation framework is that for certain task parameter values the memory requirements are too large.

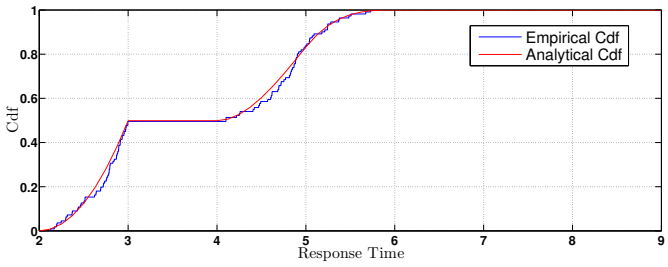


Fig. 3: Cumulative response time distribution of the 10^{th} job in the probabilistic response-time analysis example.

VII. Periodic–Stochastic LQG Control Design for Random Job Response Times

This section presents a numerical method for LQG control design when the execution time for each task is varying randomly. It is assumed that a hyperperiod exists and that the response time PDF has been derived for each job of the task. As usual, the LQG design can be split into two parts: state feedback design and state estimator design.

A. State Feedback Design

In the state feedback design, it is assumed that the full state vector x is known at each sampling instant. The goal is to design a time-varying control law

$$u(k) = -L(k)x(k)$$

where the state feedback gain L varies over the hyperperiod. Given knowledge of the PDFs of the delays τ_k, we formulate a periodic–stochastic Riccati equation as

$$\begin{aligned} S(k) =& \mathrm{E}_{\tau_{k+1}}\{\Phi^T S(k+1)\Phi + Q_1 - (\Phi^T S(k+1)\Gamma + Q_{12}) \\ & (\Gamma^T S(k+1)\Gamma + Q_2)^{-1}(\Gamma^T S(k+1)\Phi + Q_{12}^T)\} \end{aligned}$$

where matrices Φ, Γ and Q are all functions of τ_{k+1}. They can be calculated using the method in Section IV-B. The Riccati equation can be solved iteratively by calculating $S(l)$, $S(l-1)$, $\ldots$, $S(1)$, and then repeating again from $S(l)$ and so on, until the sequence of matrices $S(k)$, $k = 1 \ldots l$, converges. The state feedback gain is then obtained as

$$L(k) = \mathrm{E}_{\tau_{k+1}}(\Gamma^T S(k+1)\Gamma + Q_2)^{-1}(\Gamma^T S(k+1)\Phi + Q_{12}^T)$$

B. State Estimator Design

Since the system we are considering in (1) is linear and time-varying, and v_i, e_i are Gaussian noises, a time-varying Kalman filter is the optimal estimator. Since the sampling period is fixed, the Kalman filter gain K can actually be obtained by solving a regular Riccati equation, yielding the standard update equation

$$\hat{x}(k \mid k) = \hat{x}(k \mid k-1) + K(y(k) - C\hat{x}(k \mid k-1))$$

where $\hat{x}$ is the estimated state vector. The state prediction equation will however be time-varying and is obtained by taking the expected value over the delay for each job:

$$\hat{x}(k+1 \mid k) = \mathrm{E}_{\tau_k}\Phi\hat{x}(k \mid k) + \mathrm{E}_{\tau_k}\Gamma u(k)$$

Combining the state estimator with the state feedback, we finally obtain the periodic LQG control law

$$u(k) = -L(k)\hat{x}(k \mid k)$$

VIII. Periodic–Stochastic LQG Control Evaluation

In this section we first apply the periodic–stochastic LQG design to a simple example and calculate the exact costs using Jitterbug. We then look at two co-design examples where performance was evaluated using simulations in TrueTime.

A. A Simple Example

The plant is an inverted pendulum, which is the same as in Section V-A. We also have the same cost matrix Q_c, and noise covariance matrices R_{1c} and R_2. The sampling period is $h = 0.3$, and the hyperperiod is $H = 3h$. The response times of every job in one hyperperiod are given in the form of probability mass functions as

$$\tau_1 = \begin{pmatrix} 0.12 & 0.4 \\ 0.15 & 0.2 \\ 0.18 & 0.4 \end{pmatrix}$$
$$\tau_2 = \begin{pmatrix} 0.15 & 0.5 \\ 0.18 & 0.5 \end{pmatrix}$$
$$\tau_3 = \begin{pmatrix} 0.12 & 1 \end{pmatrix}$$

Here, the left column is the length of response time, and the right column is the corresponding probability.

LQG controllers are designed by the following three methods:

- Periodic–stochastic LQG: Design a set of controllers using the the method proposed in Section VII.
- Periodic LQG: Using the average response time as the delay in each period, design a set of LQG controllers using the method from Section IV.
- Stochastic LQG: Design a time-invariant controller based on the overall delay distribution.

The costs, evaluated in Jitterbug, are shown in Table VI. The periodic–stochastic LQG has the lower cost than the other three methods, because it takes all the information about response times into account in the control design procedure. The other two methods work with different kinds of approximations of the response times and are hence suboptimal.

TABLE VI: Costs in the simple example

	Cost
Periodic–stochastic LQG design	0.92
Periodic LQG design	1.01
Stochastic LQG design	0.99

B. Three-Plant Example

Assume a set of three tasks controlling three plants

$$P_1(s) = \frac{1}{s^2 - 0.068s - 0.007}$$
$$P_2(s) = \frac{1}{s^2 + 1.048s - 0.640}$$
$$P_3(s) = \frac{1}{s^2 - 0.442s + 0.397}$$

to be controlled in three tasks in a real-time system. P_1 has the highest priority, while P_3 has the lowest priority.

We start by assigning the initial periods with the approach proposed in [5]. We then use perturbed periods to obtain a finite hyperperiod. The utilization is set to 0.95, in order to avoid excessive control delays. We assume that the actual execution times are random variables, where the execution time of each job is drawn from $\mathrm{U}(0.9C, C)$. The job response time PDFs are then derived from TrueTime simulations. We evaluate the same three methods as outlined in the previous subsection. VIII-A. The costs are from the evaluation in TrueTime and are shown in Table VII. Again it is seen that the periodic–stochastic LQG outperforms the other two methods.

TABLE VII: Costs in the three-plant example

	Cost
Periodic–stochastic LQG design	5.42
Periodic LQG design	6.84
Stochastic LQG design	16.57

C. Evaluation on Randomly Generated Plants

We here consider control and scheduling co-design for sets of randomly generated plants. The same kind of plants as in Section V-C are used, and again the evaluation examines systems of $n = 3$ control tasks. The nominal task utilization U_inom are is generated from an n-dimensional uniform distribution with total utilization 0.95. The worst-case execution time is generated from $C_i \in \mathrm{U}(0.04, 0.4)/n$. The task priorities are assigned based on the periods returned by [1].

Then the initial task periods are assigned by the method in [5]. The periods are then perturbed get a finite hyperperiod. The utilization is throughout kept at 0.95 to avoid excessive control delays. The controllers are again designed by the three methods outlined in Section VIII-A.

The different control design methods are evaluated by Monte Carlo simulation, where the plants, the controllers, and the scheduler are simulated in parallel using TrueTime. From each family of plants, 10 random plants are generated. After control design, the plants and controllers are simulated for 1000 s, and total cost, J, is recorded. All the costs are given in Table IV. The actual job execution times are random variables drawn from $\mathrm{U}(0.9C, C)$.

TABLE VIII: Evaluation of the costs for random plants

	Family I	Family II	Family III
Periodic–stochastic LQG design	3.71	5.36	13.07
Periodic LQG design	4.01	9.00(5)	13.49(3)
Stochastic LQG design	5.48(1)	7.91(5)	22.74(2)

The numbers in the parenthesis indicate how many times the cost is infinity due to an unstable control loop. The mean value of cost does not include the infinity costs. We can see that the cost with periodic–stochastic LQG controller has lower value than costs with the other two methods, even when the unstable cases have been discounted. The periodic–stochastic LQG is the only method to obtain 100% stable systems for families II and III.

IX. Conclusions and Future Work

We have proposed new periodic and periodic–stochastic LQG control designs for minimizing the overall cost in real-time control systems. The approaches rely on knowledge of the response-time pattern of each task. In the case of periodic–stochastic LQG, also knowledge of the PDF of the response time of each job is required. To target large systems, there is a need for more efficient tools for statistical response-time analysis. Also, non-control tasks with possibly unknown phasings should ideally also be included in the analysis.

Several other issues remain open. One is to extend the ideas to the situation where the worst-case response time is larger than sampling period. Another issue is the consideration of

multiprocessor case is also interesting. Finally, to further prove the usefulness of the proposed method, it would be interesting to perform evaluations with hardware and communication overheads.

APPENDIX A
SAMPLING THE PERIODIC TIME DELAY SYSTEM

For convenience, all the subscripts indicating the ith task are omitted. Integration of Equation (1) over one hyperperiod is given as

$$\begin{aligned}
&x((k+1)lh)\\
=&e^{Alh}x(klh)+\int_{klh}^{klh+\tau_1}e^{A((k+1)lh-s)}dsBu((kl-1)h)\\
&+\sum_{j=1}^{l-1}\int_{(kl+j-1)h+\tau_j}^{(kl+j)h+\tau_{j+1}}e^{A((k+1)lh-s)}dsBu((kl+j-1)h)\\
&+\int_{((k+1)l-1)h+\tau_l}^{(k+1)lh}e^{A((k+1)lh-s)}dsBu(((k+1)l-1)h)\\
=&\Phi x(klh)+\Gamma_0\begin{pmatrix}u(klh)\\u((kl+1)h)\\\vdots\\u(((k+1)l+1)h)\end{pmatrix}\\
&+\Gamma_1\begin{pmatrix}u((k-1)lh)\\u(((k-1)l+1)h)\\\vdots\\u((kl-1)h)\end{pmatrix}\\
=&\Phi x(klh)+\Gamma_0u'(klh)+\Gamma_1u'((k-1)lh)
\end{aligned}\tag{9}$$

where

$$\Phi=e^{Alh}$$

$$\Gamma_0=\begin{pmatrix}\int_{klh+\tau_1}^{(kl+1)h+\tau_2}e^{A((k+1)lh-s)}dsB\\\int_{klh+\tau_2}^{(kl+1)h+\tau_3}e^{A((k+1)lh-s)}dsB\\\vdots\\\int_{((k+1)l-2)h+\tau_{l-1}}^{((k+1)l-1)h+\tau_l}e^{A((k+1)lh-s)}dsB\end{pmatrix}^T$$

$$\Gamma_1=\begin{pmatrix}0&0&\cdots&0&\int_{klh}^{klh+\tau_1}e^{A((k+1)lh-s)}dsB\end{pmatrix}$$

Here, the vector $u'(klh)$ contains all the control signals from time klh to time $(k+1)lh$, and the length of it is l.

APPENDIX B
SAMPLING THE LOSS FUNCTION

In order to calculate Q_1, Q_{12} and Q_2, the cost matrices of the loss function are sampled using zero-order hold. For Equation (1), when v_i, e_i are zero, and form time b to time a control $u_i(t)$ is constant, the discrete time cost matrices are

$$Q_1^0(a,b)=\int_b^a\Phi^T(s,b)Q_{1c}\Phi(s,b)ds$$

$$Q_{12}^0(a,b)=\int_b^a\Phi^T(s,b)(Q_{1c}\Gamma(s,b)+Q_{12c})ds$$

$$Q_2^0(a,b)=\int_b^a\Gamma^T(s,b)Q_{1c}\Gamma(s,b)+2\Gamma^T(s,b)Q_{12c}+Q_{2c}ds$$

Due to the periodicity, the cost matrices calculation is only performed from time 0 to time lh, which is one hyperperiod. Then

$$Q_1=Q_1^0(lh,0)$$

We further have

$$Q_2=\begin{pmatrix}Q_2^{(1,1)}&Q_2^{(1,2)}&\cdots&Q_2^{(1,l)}\\Q_2^{(1,2)T}&Q_2^{(2,2)}&\cdots&Q_2^{(2,l)}\\\vdots&\vdots&\ddots&\vdots\\Q_2^{(1,l)T}&Q_2^{(2,l)T}&\cdots&Q_2^{(l,l)}\end{pmatrix}$$

where

$$\begin{aligned}
&Q_2^{(1,1)}\\
=&Q_2^0(h+\tau_2,\tau_1)+\sum_{i=1}^{l-2}\Gamma^T(h+\tau_2,\tau_1)\Phi^T(ih+\tau_{i+1},h+\tau_2)\\
&Q_1^0((i+1)h+\tau_{i+2},ih+\tau_{i+1})\Phi(ih+\tau_{i+1},h+\tau_2)\\
&\Gamma(h+\tau_2,\tau_1)+\Gamma^T(h+\tau_2,\tau_1)\Phi^T((l-1)h+\tau_l,h+\tau_2)\\
&Q_1^0(lh,(l-1)h+\tau_l)\Phi((l-1)h+\tau_l,h+\tau_2)\Gamma(h+\tau_2,\tau_1)
\end{aligned}$$

$$\begin{aligned}
&Q_2^{(1,2)}=\Gamma^T(h+\tau_2,\tau_1)(\sum_{i=1}^{l-1}\Gamma^T(ih+\tau_{i+1},h+\tau_2)\\
&Q_1^0((i+1)h+\tau_{i+2},ih+\tau_{i+1})\Gamma(ih+\tau_{i+1},h+\tau_2)+\\
&\Gamma^T((l-1)h+\tau_l,h+\tau_2)Q_1^0(lh,(l-1)h+\tau_l)\\
&\Gamma((l-1)h+\tau_l,h+\tau_2))
\end{aligned}$$

$$\begin{aligned}
Q_2^{(1,l)}=&\Gamma^T(h+\tau_2,\tau_1)\Phi^T((l-1)h+\tau_l,h+\tau_2)\\
&Q_{12}^0(lh,(l-1)h+\tau_l)
\end{aligned}$$

$$\begin{aligned}
&Q_2^{(2,2)}=Q_2^0(2h+\tau_3,h+\tau_2)+\Gamma^T(2h+\tau_3,h+\tau_2)\\
&(\sum_{i=2}^{l-2}\Phi^T(ih+\tau_{i+1},2h+\tau_3)Q_1^0((i+1)h+\tau_{i+2},ih+\tau_{i+1})\\
&\Phi(ih+\tau_{i+1},2h+\tau_3)+\Phi^T((l-1)h+\tau_l,2h+\tau_3)\\
&Q_1^0(lh,(l-1)h+\tau_l)\Phi((l-1)h+\tau_l,2h+\tau_3))\\
&\Gamma(2h+\tau_3,h+\tau_2)
\end{aligned}$$

$$\begin{aligned}
Q_2^{(2,l)}=&\Gamma^T(2h+\tau_3,h+\tau_2)\Phi^T((l-1)h+\tau_l,2h+\tau_3)\\
&Q_{12}^0(lh,(l-1)h+\tau_l)
\end{aligned}$$

$$Q_2^{(l,l)}=Q_2^0(lh,(l-1)h+\tau_l)$$

And

$$Q_{12}=\begin{pmatrix}Q_{12}^1\\Q_{12}^2\\\vdots\\Q_{12}^l\end{pmatrix}$$

where

$$\begin{aligned}
&Q_{12}^1=\begin{pmatrix}\Phi(\tau_1,0)&\Gamma(\tau_1,0)\end{pmatrix}^T(Q_{12}^0(h+\tau_2,\tau_1)+\\
&(\sum_{i=2}^{l-1}\Phi^T((i-1)h+\tau_i,\tau_1)Q_1^0(ih+\tau_{i+1},(i-1)h+\tau_i)\\
&\Phi((i-1)h+\tau_i,h+\tau_2)+\Phi^T((l-1)h+\tau_l,\tau_1)\\
&Q_1^0(lh,(l-1)h+\tau_l)\Phi((l-1)h+\tau_l,h+\tau_2))\\
&\Gamma(h+\tau_2,\tau_1))
\end{aligned}$$

$$
\begin{aligned}
&Q_{12}^2 \\
&= \begin{pmatrix} \Phi(\tau_1, 0) & \Gamma(\tau_1, 0) \end{pmatrix}^T \Phi^T(h+\tau_2, \tau_1)(Q_{12}^0(2h+\tau_3, h+\tau_2) \\
&+ (\sum_{i=3}^{l-1} \Phi^T((i-1)h+\tau_i, h+\tau_2) \\
&Q_1^0(ih+\tau_{i+1}, (i-1)h+\tau_i)\Phi((i-1)h+\tau_i, 2h+\tau_3)+ \\
&\Phi^T((l-1)h+\tau_l, h+\tau_2)Q_1^0(lh, (l-1)h+\tau_l) \\
&\Phi((l-1)h+\tau_l, 2h+\tau_3))\Gamma(2h+\tau_3, h+\tau_2))
\end{aligned}
$$

$$
\begin{aligned}
Q_{12}^l = &\begin{pmatrix} \Phi(\tau_1, 0) & \Gamma(\tau_1, 0) \end{pmatrix}^T \Phi^T((l-1)h+\tau_l, \tau_1) \\
&Q_{12}^0(lh, (l-1)h+\tau_l)
\end{aligned}
$$

The matrix Q_1 is positive semidefinite, and Q_2 is positive definite, which will be relaxed in the next subsection. The cross term Q_{12} is general not zero even if $Q_{12c} = 0$.

Appendix C
Linear Quadratic Control Design

Using the extended state space matrices Φ, Γ from Section IV-A and cost matrix Q from Section IV-B, we solve the standard discrete time algebraic Riccati equation

$$
\begin{aligned}
&Q_1 + \Phi_e^T P \Phi_e - \\
&(\Phi_e^T P \Gamma_e + Q_{12})(\Gamma_e^T P \Gamma_e + Q_2)^{-1}(\Gamma_e^T P \Phi_e + Q_{12}^T) - P = 0
\end{aligned}
$$

where

$$
\Phi_e = \begin{pmatrix} \Phi & \Gamma_1 \\ 0 & 0 \end{pmatrix}, \quad \Gamma_e = \begin{pmatrix} \Gamma_0 \\ I \end{pmatrix}
$$

The linear quadratic state feedback vector is given by

$$
L = (Q_2 + \Gamma_e^T P \Gamma_e)^{-1}(2Q_{12}^T + \Gamma_e^T P \Phi_e)
$$

$Q_2 + \Gamma_e^T P \Gamma_e$ needs to be positive definite. It means that the requirement in Section IV-B about Q_2 is relaxed. The control signals over the hyperperiod are obtained as

$$
u'(klh) = \begin{pmatrix} u(klh) \\ u((kl+1)h) \\ \vdots \\ u((k+1)lh) \end{pmatrix} = -Lx(klh) = -\begin{pmatrix} L_1 \\ L_2 \\ \vdots \\ L_l \end{pmatrix} x(klh)
$$

The state feedback vector is based on the extended state $x(klh)$, which means the sampling happens every hyperperiod. But in fact, it should happen every period. So the state feedback vector must be reformulated according to

$$
\begin{pmatrix} u(klh) \\ u((kl+1)h) \\ \vdots \\ u((k+1)lh) \end{pmatrix} = -\begin{pmatrix} L_1' x(klh) \\ L_2' x((kl+1)h) \\ \vdots \\ L_l' x((k+1)lh) \end{pmatrix}
$$

For any $i \in \{1, 2, \cdots, l\}$, the extended state at time $(kl+i)h$ is

$$
\begin{aligned}
x((kl+i)h) &= (\Phi_{e,i-1} - \Gamma_{e,i-1} L_{i-1}')x((kl+i-1)h) \\
&= \prod_{j=1}^{i-1} (\Phi_{e,i-j} - \Gamma_{e,i-j} L_{i-j}')x(klh)
\end{aligned}
$$

where $\Phi_{e,i}$ and $\Gamma_{e,i}$ are the ith extended state space matrices in a hyperperiod. So

$$
\begin{aligned}
L_i x(klh) &= L_i' x((kl+i-1)h) \\
&= L_i' \prod_{j=2}^{i-1} (\Phi_{e,i-j} - \Gamma_{e,i-j} L_{i-j}')x(klh)
\end{aligned}
$$

The reformulated state feedback vectors are finally calculated recursively by

$$
L_i' = \begin{cases} L_1 & i = 1 \\ L_i \prod_{j=1}^{i-2} (\Phi_{e,j} - \Gamma_{e,j} L_j')^{-1} & i > 1 \end{cases}
$$

Acknowledgments

The authors would like to thank the LCCC Linnaeus Center and the ELLIIT Excellence Center at Linköping and Lund.

References

[1] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, "On task schedulability in real-time control systems," in *Proceedings of the 17th IEEE Real-Time Systems Symposium*, Washington, DC, USA, Dec. 1996, pp. 13–21.

[2] B. K. Kim, "Task scheduling with feedback latency for real-time control systems," in *Proceedings of the 5th International Workshop on Real-Time Computing Systems and Applications*, Hiroshima, Japan, Oct. 1998, pp. 37–41.

[3] E. Bini and M. Di Natale, "Optimal task rate selection in fixed priority systems," in *Proceedings of the 26th IEEE Real-Time Systems Symposium*, Miami, FL, Dec. 2005, pp. 399–409.

[4] A. Aminifar, S. Samii, P. Eles, Z. Peng, and A. Cervin, "Designing high-quality embedded control systems with guaranteed stability," in *Proceedings of the 33rd IEEE Real-Time Systems Symposium*, San Juan, Puerto Rico, 2012, pp. 283–292.

[5] E. Bini and A. Cervin, "Delay-aware period assignment in control systems," in *Proceedings of the 29th IEEE Real-Time Systems Symposium*, Barcelona, Spain, Dec. 2008, pp. 291–300.

[6] Y. Xu, K.-E. Årzén, E. Bini, and A. Cervin, "Response time driven design of control systems," in *Proceedings of the 19th World Congress of the International Federation of Automatic Control*, Cape Town, South Africa, Aug. 2014, pp. 6098–6104.

[7] P. Ramanathan, "Graceful degradation in real-time control application using (m,k)-firm guarantee," in *Proceedings of the 27th Annual International Symposium on Fault-Tolerant Computing*, Seattle, WA, 1997, pp. 132–141.

[8] D. Goswami, M. Lukasiewycz, R. Schneider, and S. Chakraborty, "Time-triggered implementations of mixed-criticality automotive software," in *Proceedings of the Conference on Design, Automation and Test in Europe*, Dresden, Germany, 2012, pp. 1227–1232.

[9] A. Saifullah, C. Wu, P. B. Tiwari, Y. Xu, Y. Fu, C. Lu, and Y. Chen, "Near optimal rate selection for wireless control systems," *ACM Transactions on Embeddded Computing Systems*, vol. 13, no. 4s, pp. 1–25, Apr. 2014.

[10] I. Ripoll and R. Ballester-Ripoll, "Period selection for minimal hyperperiod in periodic task systems," *Computers, IEEE Transactions on*, vol. 62, no. 9, pp. 1813–1822, 2013.

[11] A. Cervin, D. Henriksson, B. Lincoln, and K.-E. Årzén, "Jitterbug and TrueTime: Analysis tools for real-time control systems," in *Proceedings of the 2nd Workshop on Real-Time Tools*, Copenhagen, Denmark, Aug. 2002.

[12] B. Tanasa, U. D. Bordoloi, P. Eles, and Z. Peng, "Probabilistic response time and joint analysis of periodic tasks," in *Proceedings of the 27th Euromicro Conference on Real-Time Systems*, Lund, Sweden, 2015.