LQG-Based Control and Scheduling Co-Design

LQG-Based Control and Scheduling Co-Design ⋆

Yang Xu[1], **Karl-Erik Årzén**[1], **Enrico Bini**[2], **Anton Cervin**[1]

[1] *Department of Automatic Control, Lund University, Sweden, e-mail: {yang,karlerik,anton}@control.lth.se*
[2] *Department of Computer Science, University of Torino, Italy*

Abstract: Control and scheduling co-design becomes an issue when several controller tasks share the same execution platform and disrupt the ideal sampling and actuation patterns. In co-design the objective is to optimize the combined performance of all the controllers on the platform, subject to schedulability constraints. In the paper four LQG-based co-design methods are reviewed and evaluated: delay-aware, stochastic, periodic, and harmonic LQG co-design.

Keywords: real-time system, linear-quadratic control, fixed-priority scheduling

1. INTRODUCTION

For cost-saving reasons it is not uncommon to let multiple feedback controllers share the same computational platform. The controllers are usually implemented as periodic tasks, executing under the control of a real-time operating system with fixed-priority preemptive scheduling. At the design stage it is typically assumed that the sampling period and the delay between sampling and actuation (the so-called input-output latency) are constant. However, in a single-CPU, multitasking system, scheduling-induced interference from high-priority tasks make the above assumptions invalid. This is the motivation for control and scheduling co-design. The objective of the co-design is to optimize the total combined performance of all the controllers on the platform, subject to a schedulability constraint. This is done by selecting task parameters, i.e, periods and priorities, and designing the controllers so that they take the timing effects caused by the scheduling into account.

Control performance can be measured in multiple ways. Here the performance of each controller will be measured using a quadratic cost function of the plant states and the control signals. The total performance of the system is then given by the sum of the cost functions for all the control loops. This performance metric naturally leads to the use of LQG (linear-quadratic-Gaussian) design techniques.

The aim of this paper is to compare four LQG-based co-design methods that have been developed by the authors. In the first method, originally called *RiApprox* (Bini and Cervin, 2008) but here referred to as *delay-aware LQG co-design*, an approximate response-time analysis is used to estimate the average input-output delay. The LQG controllers are then designed assuming constant delays. In the second method, *stochastic LQG co-design* (Xu et al., 2014), the delay distributions are obtained at design-time using schedule simulation and then stochastic LQG control design is employed, i.e., the LQG controllers are designed assuming these delay distributions. The task periods obtained using the above two methods generally give rise to infinite hyperperiods. In the third method, *periodic LQG co-design* (Xu et al., 2015), the idea is to perturb the periods slightly in order to obtain a finite hyperperiod, which will correspond to a periodic delay pattern for the control loops. This periodicity is then explicitly accounted for by using periodic LQG control design, resulting in a periodic sequence of feedback gains for each controller. Finally, the fourth method, *harmonic LQG co-design* (Xu et al., 2016), again perturbs the task periods but this time to make the periods harmonic. The scheduling-induced delays will be constant and again ordinary LQG design can be applied in the same way as in the first method.

1.1 Related Work

Scheduling is fundamental in real-time control system implementation. In the seminal paper (Seto et al., 1996), a cost was defined as a function of task period. The design goal was to minimize the total cost by choosing optimal periods. The paper (Eker et al., 2000) presented how the cost function of LQ control depends on the sampling period and proposed an on-line adjustment method. Standard discrete-time LQG control theory is explained in (Åström and Wittenmark, 1997). The optimal LQG design problem for delay probability distributions was solved in (Nilsson et al., 1998) and has also been implemented in the Jitterbug toolbox (Lincoln and Cervin, 2002).

Many variants of the control and scheduling co-design problem have been proposed. In (Zhang et al., 2008), an H_∞ control design method for real-time scheduling is proposed. (Samii et al., 2009a) proposed control scheduling co-design procedures for static-cyclic and priority-based scheduling. (Samii et al., 2009b) considered synthesis of multi-mode embedded control systems. (Goswami et al., 2012) synthesized schedules that optimize control performance, satisfying timing requirements by solving an integer linear programming problem. (Bund and Slomka,

⋆ This work was supported in part by the LCCC Linnaeus Center and the ELLIIT Excellence Center at Lund University.

2013) introduced delay density to derive the delay specification for control system. (Aminifar et al., 2013) modified periods and priorities to optimize performance while guaranteeing stability. Harmonic period assignment algorithms have been presented in e.g. (Bonifaci et al., 2013), (Nasri et al., 2014).

1.2 Outline

Section 2 contains the problem formulation. Section 3 contains brief descriptions of the four co-design methods. More detailed presentations are available in the references. Finally, in Section 4 the four methods are evaluated.

2. PROBLEM FORMULATION

2.1 Real-Time System Model

A real-time system, consisting of n independent tasks running on a single processor under preemptive fixed-priority scheduling, is considered. The ith task, denoted by τ_i, is characterized by the following parameters:

- The *execution time* C_i is the length of time the task τ_i takes to execute. In this paper, unless otherwise stated, a constant C_i is assumed.
- The *period* T_i is the constant time interval between two consecutive releases of task τ_i.
- The task *priority* is implicitly given by the task index so that τ_i has higher priority than τ_{i+1}.

Furthermore, the following task parameters are defined:

- The *start latency* S_i is the time interval between the release time and the start time of task τ_i.
- The *response time* R_i is the time interval between the release time and the finish time of task τ_i.
- The task *utilization* $U_i = C_i/T_i$ measures the fraction of computational resources required by the controller. The *total utilization* of all control tasks is $U = \sum_{i=1}^{n} U_i$, which should be less than or equal to 1.
- The *hyperperiod* H is the least common multiplier of all the task periods.

2.2 LQG Control

Each real-time task τ_i defined above is used to implement a discrete-time controller, for which the following timing parameters are defined:

- The *sampling interval* h_i is the time difference between two sampling operations of task τ_i. Here we assume that sampling jitter has been eliminated by sampling at the release time; hence, $h_i = T_i$.
- The *delay* L_i is the time interval between the sampling and actuation of task τ_i. The actuation is performed at the end of execution; hence, $L_i = R_i$.

The plant to be controlled by each controller is described by a continuous-time linear single-input, single-output system

$$\dot{x}(t) = Ax(t) + Bu(t) + v_c(t) \quad (1)$$

$$y(t_k) = Cx(t_k) + e(t_k) \quad (2)$$

where x is the plant state, u is the controlled input, and v_c is a continuous-time white noise process with intensity R_{1c}. The output y is measured at discrete time instants t_k, with measurement noise e described by a discrete-time Gaussian white process with variance R_2. A, B, and C are matrices of appropriate sizes.

For each plant, an LQG controller should be designed to minimize the quadratic cost function

$$J_i = \lim_{t\to\infty} \frac{1}{t} \mathrm{E}\left\{ \int_0^t \left(x^T(s) Q_{1c} x(s) + \rho u^2(s) \right) ds \right\} \quad (3)$$

The matrix Q is a symmetric positive definite weighting matrix that penalizes state deviations and $\rho > 0$ is a weight used to trade off the relative importance between the state x and the input u. The global objective is to minimize the overall cost subject to schedulability:

$$\text{minimize } J = \sum_{i=1}^{n} J_i, \quad \text{s.t. } U \le 1 \quad (4)$$

Sampling the state equation (1) with the interval h and a time-varying delay L^k, a time-varying sampled-data system is obtained:

$$x(t_{k+1}) = \Phi(L^k)x(t_k) + \Gamma_0(L^k)u(t_k) + \Gamma_1(L^k)u(t_{k-1}) \quad (5)$$

Similarly, the cost function (3) is translated into a time-varying, discrete-time counterpart. Based on the sampled model, a state feedback law can be designed, which is based on either the average value of L^k, the distribution of L^k, or the pattern of L^k over the hyperperiod. Finally, a Kalman filter for the sampled model is designed and it is combined with the state feedback to obtain a complete LQG controller.

3. CO-DESIGN METHODS

3.1 Delay-Aware LQG Co-Design

In the first co-design method, (Bini and Cervin, 2008), the delay of the controller is approximated as

$$L_i = R_i^{\text{approx}} = \frac{C_i}{1 - \sum_{j=1}^{i-1} U_j} \quad (6)$$

This constant value is an approximation of the actual average delay over all task instances. Then, for each control task τ_i, the cost function (3) is approximated as a linear function of the period and the approximated delay:

$$J_i = \alpha_i T_i + \beta_i L_i \quad (7)$$

From this the periods that solve (4) can be found analytically. In reality, though, the cost functions are nonlinear but smooth. Then an iterative algorithm can be used to find the optimal periods.

Two priority assignment strategies are proposed. One is Seto's rate-monotonic priority assignment (Seto et al., 1996). The other one is a brute force method: All $n!$ priority assignments are tested and the one giving the smallest cost is selected. In this paper the first approach is taken. The periods and priorities returned by this method are also used as a starting point for the other methods.

3.2 Stochastic LQG Co-Design

The second co-design method, (Xu et al., 2014), assumes knowledge of the delay distributions of the controller tasks. They can in principle be obtained in two different

ways: using probabilistic response-time analysis or by a sufficiently long schedule simulation. Here, the second approach has been chosen.

For a given set of task parameters, the procedure to compute the overall cost J as a function of the periods consists of the following steps:

(1) Compute response-time distributions for each task i;
(2) Design an LQG controller using the response-time distribution as delay distribution for each task i;
(3) Calculate the LQG cost J_i of each task i;
(4) Calculate the overall cost J.

The overall cost function for a two-task example is shown in Fig. 1. As can be seen, the function is both non-linear and non-convex. Relying on nonlinear optimization, (4) is solved. Two different methods for the period assignment have been considered:

- **Local solution.** A heuristic derivative-free optimization method is proposed to find the local optima. This is called the Sequential Search method.
- **Global solution.** The DIRECT sampling-based optimization method, (Jones et al., 1993), is used to search for the global optimum.

The initial periods are in both cases obtained using the delay-aware LQG method.

3.3 Periodic LQG Co-Design

Assuming constant execution times, the task response times vary from instance to instance, but they have a pattern that repeats over the hyperperiod. Knowledge of this pattern makes it possible to use periodic LQG design techniques to design controllers that are aware of the delay periodicity, see (Xu et al., 2015). The starting point for the design method is again the task period obtained from the delay-aware LQG co-design method. Since these periods are real-valued, the hyperperiods may be large or infinite. In order to apply periodic LQG design, the following method to perturb the periods has been proposed to obtain a finite and short hyperperiod:

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

$$1 - \frac{\min_i\{k_i T_i\}}{\max_i\{k_i T_i\}} \leq \epsilon. \tag{8}$$

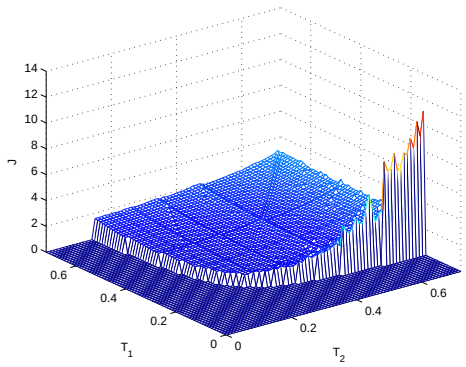

Fig. 1. Example with two control tasks showing the total cost J as a function of periods T_1 and T_2. The cost function is non-smooth and non-convex.

The *approximate hyperperiod* $\hat{H}$ is then given by

$$\hat{H} = \max_i\{k_i T_i\}. \tag{9}$$

The following method is proposed for period assignment:

(1) Set a value of ϵ of desired proximity. A typical value could be between 10^{-3} and 10^{-2}.
(2) Compute the set of integers $[k_1, \ldots, k_n]$ such that (8) holds;
(3) Calculate the modified periods as

$$\hat{T}_i = \frac{\sum_{j=1}^{n} k_j C_j}{k_i} \tag{10}$$

This choice implies that the tasks with the modified periods $\hat{T}_i$ are fully utilizing the processor.

Via a schedule simulation, it is straightforward to obtain the sequence of delay values L^k over a hyperperiod. A corresponding periodic LQG controller can then be designed by solving a periodic Riccati equation.

In a real system, the execution times are typically not constant. One approach then is to use the average response time for each job, and reformulate the problem into the constant execution time case. However, if the execution time variations are large, this will not yield a good approximation. In that case it is possible to design a *periodic-stochastic LQG controller*. To design such a controller, the response time distributions for every job over the hyperperiod needs to be calculated first. Also here one can either use probabilistic response time analysis, e.g., (Tanasa et al., 2015), or simulate the schedule with randomly distributed execution times for sufficiently long and record the response times. Given this the periodic-stochastic LQG controller can be obtained by iteratively solving the corresponding Riccati equations.

3.4 Harmonic LQG Co-Design

In industry, harmonic periods are often selected for ease of implementation. The harmonic LQG co-design method, (Xu et al., 2016), exploits this but now also for control performance reasons. Consider the simple two-task example shown in Fig. 2. In the upper plot the tasks have periods $\{3, 5\}$ and constant execution times $\{1, 3\}$. It is assumed

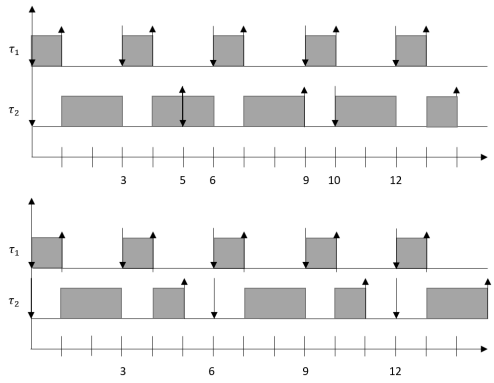

Fig. 2. Example of two tasks with non-harmonic periods (upper plot) and harmonic periods (lower plot). The job arrivals are shown with down-arrows and the job finishing times with up-arrows.

that these periods have been chosen to give good total control performance. From the figure one can see that the response time of task τ_2 varies according to the pattern $5, 4, 4, 5, 4, 4 \ldots$. If one changes the period of task τ_2 to 6, i.e., harmonize the periods, then, as shown in the lower plot, the response time will be always equal to 5, i.e., more deterministic than in the first case. This will most likely lead to worse control performance since both the period and the average delay are larger than before. However, if one also introduces an offset of 1 for τ_2, i.e., perform the sampling at the start of the job rather than at the release time, then the response time will always be equal to 4. The question is then whether the decrease in control performance of the controller executing as task τ_2 caused by the longer sampling period is compensated for by the increased performance caused by the shorter and constant delay obtained through the period harmonization. As will be shown in Section 4 this is very often the case.

Harmonic task sets have several nice properties (Lehoczky et al., 1989). For instance, schedulability is guaranteed as long as the total utilization is less than or equal to 1. Further, under the assumption that execution times are constant for each task, it holds that

- The response time for each task is constant.
- The start latency for each task is constant, and is given by the response time of task τ_{i-1}, with τ_i being the current task.

The co-design method starts by calculating initial task periods using the stochastic LQG co-design method. The task periods are then harmonized using Theorem 1 in (Xu et al., 2016). The theorem finds the closest harmonic periods to a set of initial periods in the Euclidean sense. For n tasks there will be 2^{n-1} sets of possible periods. All possible assignments are evaluated, and the one that gives the best control performance is selected. It should however be noted that the harmonic period assignment with the lowest control cost is not necessarily restricted to the above sets. The control cost function could have a form so that the harmonic period assignment with the lowest cost is not among the above candidates. However, as shown in the general evaluation the proposed approach obtained gives considerably better control performance than the non-harmonic case.

Finally a release offset is added to each task except the highest priority task. The length of the offset is the start latency of each task. Then a set of LQG controllers are designed for the obtained constant delays.

4. EVALUATION

In this section, the four co-design methods outlined above are evaluated.

4.1 Design and Evaluation Tools: Jitterbug and TrueTime

Jitterbug is a MATLAB based toolbox used to design and evaluate controllers under various timing conditions (Lincoln and Cervin, 2002). Jitterbug can be used to design LQG controller with constant delays or delay probability distributions. For various controllers, it can be used to evaluate LQG costs by modeling different delay sequences. In this paper, Jitterbug is used to design LQG controllers and to evaluate LQG control performance, in cases where Jitterbug supports this.

TrueTime is a MATLAB/Simulink-based simulation toolbox for embedded and networked control systems (Cervin et al., 2003). For real-time systems simulation, TrueTime provides fixed-priority scheduling with preemption and release offset. Using Monte Carlo simulations, TrueTime can be used to evaluate the total system cost. Schedule simulations are performed and the LQG control costs is evaluated in TrueTime, in the cases not supported by Jitterbug.

4.2 A Simple Example

As a simple co-design example, assume three plants,

$$P_1(s) = \frac{2}{s^2}, \quad P_2(s) = \frac{1}{s^2 - 3}, \quad P_3(s) = \frac{1}{s(s+1)},$$

that should be controlled by three tasks τ_1, τ_2, τ_3. τ_1 has the highest priority and τ_3 has the lowest priority. The constant execution times are given as $C_1 = 0.1$, $C_2 = 0.12$, $C_3 = 0.14$. The LQG cost function parameters are given as $Q_{1c} = C^T C$, and $\rho = 0.01\mathrm{tr}\,(Q_{1c})$.

The evaluation procedure is performed as follows:

- **Delay-aware LQG co-design.** The LQG cost of each plant i is approximated by a linear function of the period and delay as in Eq. (7). The sensitivity coefficients α_i and β_i are evaluated at the point $T_i = C_i$, $L_i = C_i$ using Jitterbug. Then the period assignment method in (Bini and Cervin, 2008) is used to minimize the LQG cost under the simplifying assumption that these are the true cost functions.
- **Stochastic LQG co-design.** Using the delay-aware LQG periods as a starting point, first 100 response times are calculated to obtain the delay probability distributions for each controller. This data is then used to design a set of stochastic LQG controllers (Xu et al., 2014). The whole procedure is repeated in a sequential search for the optimal task periods.
- **Periodic LQG co-design.** Using the stochastic LQG periods as a starting point, the solution is perturbed with the tolerance $\epsilon = 0.05$ to yield a finite hyperperiod (Xu et al., 2015). A set of time-varying LQG controllers are then designed based on the resulting cyclic schedule.
- **Harmonic LQG co-design.** Using the delay-aware LQG periods as a starting point, the $2^{n-1} = 4$ closest harmonic task period sets are investigated (Xu et al., 2016). For each period set, the fixed start latencies and response times are computed and used to design standard LQG controllers. Jitterbug is then used to find the minimal total cost among these four cases.

The resulting periods and the total LQG cost are shown in Table 1, where all numbers have been rounded to two decimal places. The stochastic LQG has lower cost than the delay-aware LQG method, because more response time information is taken into account in the design procedure. The periodic LQG and harmonic LQG methods have better performance than delay-aware LQG and stochastic LQG. For periodic LQG the reason is that with a finite hyperperiod the delay variation is smaller than it using

Table 1. Results for the simple example

	T_1	T_2	T_3	J
Delay-aware LQG	0.26	0.33	0.56	2.36
Stochastic LQG	0.30	0.41	0.45	2.18
Periodic LQG	0.26	0.34	0.58	1.85
Harmonic LQG	0.29	0.29	0.58	1.33

a potential infinite hyperperiod. For harmonic LQG the reason is that using harmonic periods and task offsets the schedule will give rise to both constant and short delays.

In the references that this paper is based on more elaborate evaluations are presented. However, only partial comparisons have been made so far. In the next subsection, a larger evaluation is performed using randomly generated plant dynamics and showing the variability and confidence of the results obtained.

4.3 Randomly Generated Examples

To see whether the results for the simple example above holds in more general cases, sets of three plants are randomly generated for large-scale evaluation from the following three plant families:

- Family I: All plants have two stable poles and are 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 $a_1, a_2, \zeta \in \text{unif}(0,1)$, $\omega \in \text{unif}(0,1)$.
- 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 $a_1, a_2, \zeta \in \text{unif}(-1,1)$, $\omega \in \text{unif}(0,1)$.
- 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 $a_1, a_2, a_3, \zeta \in \text{unif}(-1,1)$, $\omega \in \text{unif}(0,1)$.

20 sets of plants are randomly generated for each family. For the LQG controllers, the design parameters are $Q_{1c} = C^T C$, and $\rho = 0.01\text{tr}\,(Q_{1c})$. The task execution times were randomly generated from $C_1 \in \text{unif}(0.09, 0.11)$, $C_2 \in \text{unif}(0.11, 0.13)$, $C_3 \in \text{unif}(0.13, 0.15)$. Task 1 has the highest priority, while task 3 has the lowest priority.

The optimization procedure to assign initial non-harmonic periods is the same as in the previous section. The overall LQG costs are evaluated for delay-aware LQG, stochastic LQG co-design, periodic LQG co-design, harmonic LQG co-design. The overall results, averaged over 20 generated plant sets for each family, are summarized in Table 2.

The delay-aware LQG costs are normalized to 1 for each plant set, then each cost for stochastic LQG, periodic LQG, harmonic LQG, is normalized and compared with corresponding delay-aware LQG cost. The box plots are shown in Figs. 3, 4 and 5.

Table 2. Average total costs in the randomly generated examples

Family	I	II	III
Delay-aware LQG	3.35	11.93	34.87
Stochastic LQG	2.92	5.98	22.10
Periodic LQG	2.88	6.48	21.06
Harmonic LQG	2.03	3.89	13.84

In Family III, the likelihood that the plants are unstable, and, hence, more sensitive to delays and delay jitter, is larger, and therefore the total cost is considerably higher than for Family I and II. The values of stochastic LQG and periodic LQG are lower than delay-aware LQG values. The reason for this is that more information about delay are taken into account when designing controllers. The best results are obtained for the harmonic tasks with offsets. In this case the increase in cost caused by the period perturbation is small compared to the decrease in cost caused by the smaller and jitter-free delays.

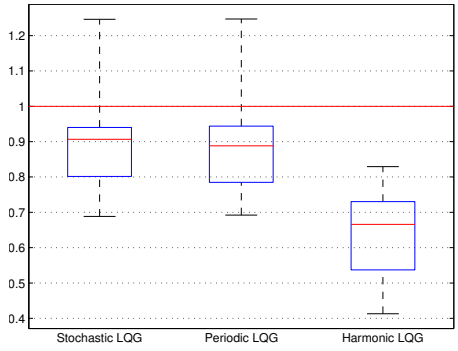

Fig. 3. Normalized costs for Family I

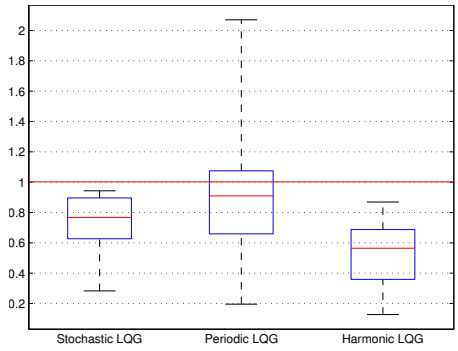

Fig. 4. Normalized costs for Family II

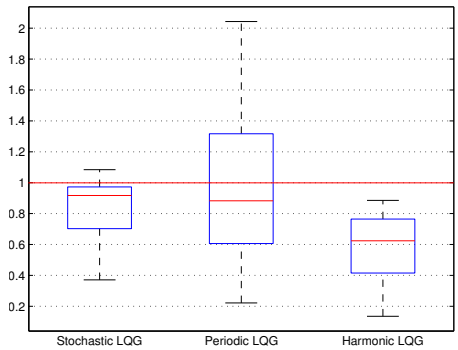

Fig. 5. Normalized costs for Family III

5. CONCLUSION

Several recent results on LQG-based control and scheduling co-design for real-time control systems have been presented and compared. The methods differ in the way that they treat the (possibly time-varying) scheduling-induced control delays. The delay-aware method only considers an approximation of the mean delay. The stochastic method treats the delay as a sequence of independent and identically distributed random variables. The periodic method looks at the deterministic delay pattern over a hyperperiod. Finally, the harmonic method eliminates the time variability all together by enforcing harmonic task periods. The evaluation showed that the last method produces the best results for randomly generated design examples. Eliminating the jitter allows for standard LQG design software to be used, which is a further benefit of the harmonic approach.

Throughout the paper several simplifying assumptions have been made. It was assumed that there is no sampling jitter, that all execution times are constant, and that the task priorities are fixed and given. All of these assumptions may be relaxed, but this leads to more complicated co-design problems. The issue of varying execution times has been treated in (Xu et al., 2015), leading to a *periodic–stochastic* LQG formulation.

Overall, the issue of robustness deserves more attention, both from a scheduling and a control point of view. The methods in the paper are based on a standard quadratic cost function that only captures the mean control performance. What happens to the total system performance in the worst case, when there are unmodeled tasks in the system, when execution times are uncertain or time-varying, or when the plant models are imperfect?

REFERENCES

Aminifar, A., Eles, P., Peng, Z., and Cervin, A. (2013). Control-quality driven design of cyber-physical systems with robustness guarantees. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*, 1093–1098.

Åström, K.J. and Wittenmark, B. (1997). *Computer-Controlled Systems. Theory and Design*. Prentice Hall, third edition.

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

Bonifaci, V., Marchetti-Spaccamela, A., Megow, N., and Wiese, A. (2013). Polynomial-time exact schedulability tests for harmonic real-time tasks. In *34th Real-Time Systems Symposium (RTSS)*, 236–245.

Bund, T. and Slomka, F. (2013). A delay density model for networked control systems. In *Proceedings of the 21st International conference on Real-Time Networks and Systems*, 205–212.

Cervin, A., Henriksson, D., Lincoln, B., Eker, J., and Årzén, K.E. (2003). How does control timing affect performance? *IEEE Control Systems Magazine*, 23(3), 16–30.

Eker, J., Hagander, P., and Årzén, K.E. (2000). A feedback scheduler for real-time controller tasks. *Control Engineering Practice*, 8(12), 1369–1378.

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

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

Lehoczky, J.P., Sha, L., and Ding, Y. (1989). The rate-monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the* 10^{th} *IEEE Real-Time Systems Symposium*, 166–171. Santa Monica (CA), U.S.A.

Lincoln, B. and Cervin, A. (2002). 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.

Nasri, M., Fohler, G., and Kargahi, M. (2014). A framework to construct customized harmonic periods for real-time systems. In *26th Euromicro Conference on Real-Time Systems (ECRTS)*, 211–220.

Nilsson, J., Bernhardsson, B., and Wittenmark, B. (1998). Stochastic analysis and control of real-time systems with random time delays. *Automatica*, 34(1), 57–64.

Samii, S., Cervin, A., Eles, P., and Peng, Z. (2009a). Integrated scheduling and synthesis of control applications on distributed embedded systems. In *Proc. Design, Automation & Test in Europe (DATE)*.

Samii, S., Eles, P., Peng, Z., and Cervin, A. (2009b). Quality-driven synthesis of embedded multi-mode control systems. In *Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE*, 864–869.

Seto, D., Lehoczky, J.P., Sha, L., and Shin, K.G. (1996). On task schedulability in real-time control systems. In *Proceedings of the* 17^{th} *IEEE Real-Time Systems Symposium*, 13–21. Washington, DC, USA.

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

Xu, Y., Årzén, K.E., Bini, E., and Cervin, A. (2014). Response time driven design of control systems. In *Proceedings of the* 19^{th} *World Congress of the International Federation of Automatic Control*, 6098–6104. Cape Town, South Africa.

Xu, Y., Årzén, K.E., Cervin, A., Bini, E., and Tanasa, B. (2015). Exploiting job response-time information in the co-design of real-time control systems. In *21st International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 247–256.

Xu, Y., Cervin, A., and Årzén, K.E. (2016). Harmonic scheduling and control co-design. In *22nd Int'l Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*.

Zhang, F., Szwaykowska, K., Wolf, W., and Mooney, V. (2008). Task scheduling for control oriented requirements for cyber-physical systems. In *IEEE Real-Time Systems Symposium, 2008*, 47–56.