

# Optimal On-Line Sampling Period Assignment

Anton Cervin<sup>1</sup>, Manel Velasco<sup>2</sup>, Pau Martí<sup>2</sup>, and Antonio Camacho<sup>2</sup>

<sup>1</sup> Automatic Control Department, Lund University  
Lund, Sweden

<sup>2</sup> Automatic Control Department, Technical University of Catalonia  
Barcelona, Spain

Research Report: ESAII-RR-09-04

December 2009

## Abstract

*In embedded systems, the computing resources are often scarce and several control tasks may have to share the same computer. In this paper, we assume that a set of feedback controllers should be implemented on a single-CPU platform. We study the problem of optimal sampling period assignment, where the goal is to assign sampling rates to the controllers so that the overall control performance is maximized. We derive expressions relating the expected cost over a finite horizon to the sampling period, the computational delay, and the amount of noise acting on the plant. Based on this, we develop a feedback scheduler that periodically assigns new sampling periods based on estimates of the current plant states and noise intensities. The approach is evaluated in extensive experiments, where three double-integrator electronic circuits are controlled by three concurrent tasks executing in a small real-time kernel. A fourth task, acting as feedback scheduler, periodically assigns new sampling periods, considering the current plant states and noise characteristics. The experiments show that on-line sampling period assignment can deliver significantly better control performance than the state-of-the-art, static period assignment.*

# 1 Introduction

With the advent of the micro-controller in the 1970's, analog feedback controllers were gradually replaced by digital counterparts. Today, the vast majority of all controllers are implemented using computers, relying on periodic sampling and control. The sampling rate is normally fixed and is selected in relation to the speed of the system under control. Typical rules of thumb [1] suggest that the sampling rate should be chosen to be 10 to 30 times the bandwidth of the closed-loop system. Faster sampling implies better disturbance rejection and smaller sensitivity to modelling errors.

In embedded systems with very limited computational resources, it may not be possible to let the controller run at the desired rate. This can be true especially if there are many tasks competing for the same CPU. Hence, design of embedded systems always involves trade-offs between the various tasks in the system.

The trade-off between different feedback controllers in an embedded system was first formulated as an off-line optimization problem in a seminal paper by Seto *et al.* [2]. The performance of each controller was captured by a cost function that described the relationship between the sampling rate and the quality of control. Assuming that the cost vs rate for each controller can be approximated by an exponentially decreasing function, the paper gives an optimization algorithm that assigns optimal sampling rates to the controllers, subject to a CPU utilization constraint.

Later work has refined the model of Seto *et al.* and also moved the optimization algorithm from off-line use to on-line use. Eker *et al.* [3] considered linear quadratic (LQ) state feedback controllers and derived expressions relating the LQ-cost to the sampling interval. They also proposed an on-line optimization algorithm for sampling period assignment, that iteratively adjusted the sampling rates based on measurements of the CPU load. The approach was extended to linear controllers in [4].

Martí *et al.* [5] introduced feedback from the actual control performance by incorporating the current plant states into a heuristic cost function. The same authors presented an extension [6] where the relation between its cost function and standard quadratic cost functions was established. Henriksson *et al.* [7] further formalized this approach for linear quadratic controllers by incorporating the current plant states into a finite-horizon quadratic cost function, which also took the expected future plant noise into account. The extension to the general case of linear controllers was given in [8].

In a parallel line of research, Ben Gaid *et al.* [9] presented a complementary approach for the optimal integrated control and real-time scheduling

of control tasks. It combined non-preemptive optimal cyclic schedules according to the  $H_2$  performance criterion, with an efficient on-line scheduling heuristic in order to improve responsiveness to disturbances. An extension of this approach can be found in [10].

In this paper, we present an on-line sampling period assignment approach that extends the model of Seto *et al.* to capture several important properties of real controllers in the optimization problem:

- the performance of each controller is captured in a finite horizon cost function, taking into account (1) the sampling period, (2) the computational delay, and (3) the amount of noise acting on the plant.
- the cost function is developed for general linear controllers (and not only for state feedback controllers as in previous work).

Since finding an analytical solution to the optimization problem is not possible in the general case even for simpler cost functions [7], we present an algorithm that allows solving the optimization problem at run-time:

- By evaluating the cost function, it is noted that most of the terms can be computed off-line.
- Then, for convex functions on the sampling period, an on-line procedure that approximates the optimal solution is developed that allows identifying the set of optimal sampling periods.

Noting that the evaluation of the cost function strongly depends on the noise intensities, we complement the feedback scheduler with a noise estimator:

- Two different standard noise estimators are suggested, and their pros and cons discussed and evaluated.
- We show that the noise estimators are when the noise intensity changes are slower than the rate of execution of the period assignment algorithm.

Finally, we provide a proof-of-concept implementation, where the feedback scheduling approach is put to test in a real system:

- We study different key parameters of the proposed approach with respect to control performance and resource utilization.
- We provide a comparative performance analysis of our approach with respect to previous related work.

- In summary, we show, in real control experiments, that significant performance improvements can be achieved by on-line sampling rate assignment.

This paper extends the preliminary results presented in [8].

The rest of this paper is outlined as follows. In Section 2, the problem formulation is given. In Section 3, formulas for the quadratic cost of a general linear controller are derived. The algorithm for the on-line optimization is given in Section 4. The noise estimator that complements the feedback scheduler is discussed in Section 5. The experimental set-up is presented in Section 6 and results are reported in Section 7. Finally, Section 8 concludes the paper.

## 2 Problem Formulation

### 2.1 Plant and Controller Models

We assume a system in which  $n$  physical plants should be controlled by a computer with limited computational resources. The control is achieved by  $n$  concurrent control tasks executing in the computer, each task being responsible for the sampling, control computation, and actuation in one loop. There exists an additional task in the system, the Feedback Scheduler (FBS), that may change the sampling rates of the tasks depending on the current state of the system. The overall situation is depicted in Figure 1.

More formally, each plant  $i = 1 \dots n$  is described by a continuous-time linear stochastic system,

$$\begin{aligned} \frac{dx_i(t)}{dt} &= A_i x_i(t) + B_i u_i(t - \tau) + v_{c_i}(t) \\ y_i(t) &= C_i x_i(t) + e_i(t) \end{aligned} \quad (1)$$

where  $x_i$  is the plant state,  $u_i$  is the control signal,  $y_i$  is the measured output,  $A_i$ ,  $B_i$  and  $C_i$  are matrices of appropriate sizes,  $v_{c_i}$  is a continuous-time white noise process with intensity  $R_{1c_i}$ ,  $e_i$  is a discrete-time Gaussian white noise process with variance  $R_{2i}$ , and  $\tau_i = \alpha_i h_i$ ,  $0 < \alpha_i < 1$ , is a constant time delay representing the input-output delay in the implementation (see Section 2.2 below). To capture different operating conditions, we allow the process noise intensity to be time varying,

$$R_{1c_i}(t) = r_i(t) \bar{R}_{1c_i} \quad (2)$$

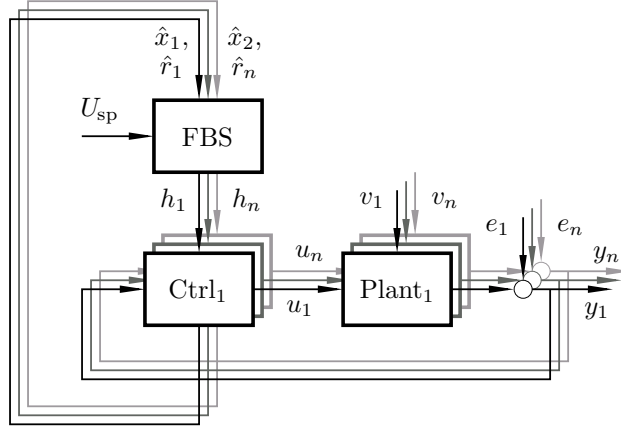


Figure 1: Dynamic resource allocation in multiple control loops using a feedback scheduler (FBS). The scheduler assigns sampling periods  $h_1 \dots h_n$ , based on state estimates  $\hat{x}_1 \dots \hat{x}_n$  and noise intensity estimates  $\hat{r}_1 \dots \hat{r}_n$ , to meet the utilization set-point  $U_{sp}$  and to optimize the overall performance of the control loops.

where  $r_i(t)$  is a positive scalar and  $\bar{R}_{1ci}$  is the nominal noise intensity. The controller can be constant, designed assuming a constant noise level, or time-varying, assuming different noise intensities given by different  $r_i(t)$  values.

Sampling the plant (1) with the interval  $h_i$  and control delay  $\tau_i = \alpha_i h_i$  yields the discrete-time system (see [1])

$$\begin{aligned} z_i[k+1] &= \Phi_i(h_i)z_i[k] + \Gamma_i(h_i)u_i[k] + v_i[k] \\ y_i[k] &= \Theta_i z[k] + e_i[k] \end{aligned} \quad (3)$$

where

$$\begin{aligned} z_i[k] &= \begin{bmatrix} x_i[k] \\ u_i[k-1] \end{bmatrix} \\ \Phi_i(h_i) &= \begin{bmatrix} \phi_i(h_i) & \phi_i(h_i - \alpha_i h_i) \gamma_i(\alpha_i h_i) \\ 0 & 0 \end{bmatrix} \\ \Gamma_i(h_i) &= \begin{bmatrix} \gamma_i(h_i - \alpha_i h_i) \\ I \end{bmatrix} \\ \Theta_i &= [C_i \ 0] \end{aligned}$$

and

$$\begin{aligned}\phi_i(t) &= e^{A_i t} \\ \gamma_i(t) &= \int_0^t \phi_i(s) B_i ds\end{aligned}$$

Assuming the noise level to be constant over the sampling interval, the covariance of the discrete-time white noise process  $v_i$  is given by

$$R_{1i}(h_i) = \begin{bmatrix} \int_0^{h_i} \phi_i(s) R_{1ci} \phi_i^T(s) ds & 0 \\ 0 & 0 \end{bmatrix} \quad (4)$$

Each controller  $i = 1 \dots n$  is described by a discrete-time linear system with adjustable sampling interval  $h_i$ ,

$$\hat{x}_i[k+1] = F_i(h_i) \hat{x}_i[k] + G_i(h_i) y_i[k] \quad (5)$$

$$u_i[k] = H_i(h_i) \hat{x}_i[k] + K_i(h_i) y_i[k] \quad (6)$$

where  $\hat{x}_i$  is the controller state (representing the plant state estimate) and  $F_i$ ,  $G_i$ ,  $H_i$  and  $K_i$  are matrices of appropriate sizes.

The performance of each control loop  $i = 1 \dots n$  is measured by a finite-horizon continuous-time quadratic cost function,

$$J_i^{T_{\text{fbs}}} = \mathbb{E} \int_0^{T_{\text{fbs}}} \left\| \begin{matrix} x_i(t) \\ u_i(t) \end{matrix} \right\|_{Q_{ci}}^2 dt \quad (7)$$

where

$$Q_{ci} = \begin{bmatrix} Q_{1ci} & Q_{12ci} \\ Q_{12ci}^T & Q_{2ci} \end{bmatrix}$$

is a positive semidefinite matrix, and  $T_{\text{fbs}}$  is the period of the feedback scheduler. For this kind of cost function, a linear-quadratic Gaussian (LQG) controller [1] is optimal. Note that our theoretical framework does not require the controllers to be optimal, however.

## 2.2 Real-Time Scheduling Model

In contrast to previous work on optimal sampling period assignment [2, 3, 5, 7, 9], in this work we explicitly take the real-time scheduling and the computational delay into account in the optimization. It is assumed that the worst-case execution time of each controller  $i$  is described by

$$E_i = E_i^{\text{Calculate}} + E_i^{\text{Update}} \quad (8)$$

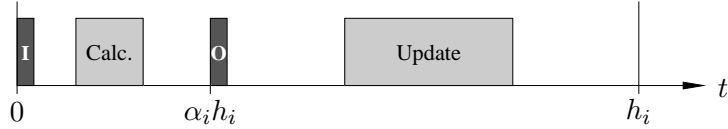


Figure 2: Scheduling model for the controllers. The input (I) and output (O) operations are scheduled at interrupt priority, while the Calculate and Update subtasks are scheduled using the Earliest-Deadline-First algorithm.

where  $E_i^{\text{Calculate}}$  is the worst-case execution time for calculating the control signal (essentially Eq. (6) above), and  $E_i^{\text{Update}}$  is the worst-case execution time for updating the controller state (Eq. (5) above).

In the implementation, the input operation (i.e., the A/D conversion of  $y[k]$ ), the Calculate portion of the code, the output operation (i.e., the D/A conversion of  $u[k]$ ), and the Update portion of the code are assumed to be scheduled as separate subtasks, see Fig. 2. The input and output operations are scheduled at interrupt priority and are assumed to take negligible time to execute. The Calculate and Update subtasks are scheduled using the Earliest-Deadline-First (EDF) algorithm [11, 12]. The Calculate subtask is released at the start of each period, with a relative deadline  $\alpha_i h_i$ , where

$$\alpha_i = \frac{E_i^{\text{Calculate}}}{E_i^{\text{Calculate}} + E_i^{\text{Update}}} \quad (9)$$

The Update subtask is released with the offset  $\alpha_i h_i$ , with a relative deadline  $(1 - \alpha_i) h_i$ . This implies that the input-output delay of controller  $i$  is constant and given by

$$\tau_i = \alpha_i h_i \quad (10)$$

Moreover, the organization of the code ensures that controller  $i$  has a constant processor demand [12] of

$$U_i = \frac{E_i}{h_i} \quad (11)$$

Schedulability of all subtasks can hence be guaranteed if and only if

$$\sum_{i=1}^n U_i \leq 1 \quad (12)$$

which is the standard schedulability condition for EDF.

### 2.3 Optimization Problem

The feedback scheduler is assumed to have knowledge of all plant and controller parameters, except the noise intensities, which are to be estimated on-line. The scheduler runs as a high-priority task with a fixed period  $T_{\text{fbs}} \gg h_i$  and execution time  $E_{\text{fbs}}$ . Upon invocation at time  $t_0$ , the scheduler is informed about the current controller states  $\hat{x}_1(t_0) \dots \hat{x}_n(t_0)$  (representing, for instance, estimates of the current plant states). The scheduler should then assign new sampling intervals  $h_1 \dots h_n$  such that the total expected cost over the next  $T_{\text{fbs}}$  units of time is minimized. This is formulated as the following optimization problem to be solved on-line:

$$\begin{aligned} \min_{h_1 \dots h_n} \quad & \sum_{i=1}^n J_i^{T_{\text{fbs}}} \\ \text{subj. to} \quad & \sum_{i=1}^n \frac{E_i}{h_i} \leq U_{\text{sp}} - U_{\text{fbs}} \end{aligned} \quad (13)$$

Here,  $U_{\text{sp}}$  is the CPU utilization set-point, which must be smaller than 1, and  $U_{\text{fbs}} = \frac{E_{\text{fbs}}}{T_{\text{fbs}}}$  is the processor demand of the feedback scheduler. The optimization problem is convex if the cost functions are convex functions of  $1/h_i$  (see [2]). Efficient on-line solution of the optimization problem is discussed in Section V.

## 3 Evaluation of the Cost Functions

To solve the optimization problem, we must be able to evaluate the cost function (7) for each control loop on-line. Note that the value of the cost function depends not only on  $T_{\text{fbs}}$ , but also on  $h_i$ ,  $\hat{x}_i(t_0)$ , the plant parameters (including the noise intensities), and the controller parameters. In this section, we therefore derive expressions for the stationary and transient cost of a linear discrete-time controller regulating a linear plant with a fixed time delay. Throughout this section, for clarity, we drop the plant/controller index  $i$ .

### 3.1 Sampling the Cost Function

Assuming that  $T_{\text{fbs}}$  can be evenly divided into  $h$ , the cost can be written as

$$J^{T_{\text{fbs}}} = \sum_{k=0}^{T_{\text{fbs}}/h-1} J[k] \quad (14)$$

where  $J[k]$  is the cost in the  $k$ th sampling interval and  $N = T_{\text{fbs}}/h$ . By sampling (7), this cost can be expressed as

$$\begin{aligned}
J[k] &= \mathbb{E} \left\{ \int_{kh}^{kh+\alpha h} \left\| \begin{matrix} x(t) \\ u(t) \end{matrix} \right\|_{Q_c}^2 dt + \int_{kh+\alpha h}^{kh+h} \left\| \begin{matrix} x(t) \\ u(t) \end{matrix} \right\|_{Q_c}^2 dt \right\} \\
&= \mathbb{E} \left\{ \int_0^{\alpha h} \left\| \begin{matrix} \phi(t)x[k] + \gamma(t)u[k-1] \\ u[k-1] \end{matrix} \right\|_{Q_c}^2 dt + \int_0^{h-\alpha h} \right. \\
&\quad \left. \left\| \begin{matrix} \phi(t)(\phi(\alpha h)x[k] + \gamma(\alpha h)u[k-1]) + \gamma(t)u[k] \\ u[k] \end{matrix} \right\|_{Q_c}^2 dt \right\} \quad (15) \\
&+ J_{\text{const}}(R_1, h) \\
&= \mathbb{E} \begin{bmatrix} z[k] \\ u[k] \end{bmatrix}^T \begin{bmatrix} Q_1(h) & Q_{12}(h) \\ Q_{12}^T(h) & Q_2(h) \end{bmatrix} \begin{bmatrix} z[k] \\ u[k] \end{bmatrix} + J_{\text{const}}(R_1, h)
\end{aligned}$$

where

$$\begin{aligned}
Q_1(h) &= [Q_{1a}(h) \quad Q_{1b}(h)] \\
Q_{1a}(h) &= \begin{bmatrix} q_1(\alpha h) + \phi^T(\alpha h)q_1(h-\alpha h)\phi(\alpha h) \\ q_{12}^T(\alpha h) + \phi(\alpha h) \end{bmatrix} \\
Q_{1b}(h) &= \begin{bmatrix} q_{12}(\alpha h) + \phi^T(\alpha h) \\ q_2(\alpha h) + \gamma^T(\alpha h)q_1(h-\alpha h)\gamma(\alpha h) \end{bmatrix} \\
Q_{12}(h) &= \begin{bmatrix} \phi^T(\alpha h)q_{12}(h-\alpha h) \\ \gamma^T(\alpha h)q_{12}(h-\alpha h) \end{bmatrix} \\
Q_2(h) &= q_2(h-\alpha h) \\
J_{\text{const}}(R_1, h) &= \text{tr} \, Q_{1c} \int_0^h \int_0^s \phi(s)R_{1c}\phi^T(s)ds dh
\end{aligned}$$

and

$$\begin{aligned}
q_1(t) &= \int_0^t \phi^T(s)Q_{1c}\phi(s) ds \\
q_{12}(t) &= \int_0^t \phi^T(s)(Q_{1c}\gamma(s) + Q_{12c}) ds \\
q_2(t) &= \int_0^t (\gamma^T(s)Q_{1c}\gamma(s) + 2\gamma^T(s)Q_{12c} + Q_{2c}) ds
\end{aligned}$$

The term  $J_{\text{const}}(R_1, h)$  is due to the intersample noise, see [13].

### 3.2 Evaluation of the Cost

Combining (3) with (5) and (6), the closed-loop system can be written as

$$\begin{bmatrix} z[k+1] \\ \hat{x}[k+1] \end{bmatrix} = \Phi_{cl}(h) \begin{bmatrix} z[k] \\ \hat{x}[k] \end{bmatrix} + \Gamma_{cl}(h) \begin{bmatrix} v[k] \\ e[k] \end{bmatrix} \quad (16)$$

where

$$\begin{aligned} \Phi_{cl}(h) &= \begin{bmatrix} \Phi(h) + \Gamma(h)K(h)\Theta & \Gamma(h)H(h) \\ G(h)\Theta & F(h) \end{bmatrix} \\ \Gamma_{cl}(h) &= \begin{bmatrix} I & \Gamma(h)K(h) \\ 0 & G(h) \end{bmatrix} \end{aligned}$$

One difficulty here is that  $u[k]$  appears in the cost function, but not in the formulation of the closed-loop system. However, noting that

$$u[k] = H(h)\hat{x}[k] + K(h)\Theta z[k] + K(h)e[k] \quad (17)$$

the cost (15) can be rewritten as

$$\begin{aligned} J[k] &= \mathbb{E} \left\| \begin{bmatrix} z[k] \\ \hat{x}[k] \end{bmatrix} \right\|_{Q_{cl}(h)}^2 + K^T(h)Q_2(h)K(h)R_2 \\ &\quad + J_{\text{const}}(R_1, h) \end{aligned} \quad (18)$$

where

$$\begin{aligned} Q_{cl}(h) &= \begin{bmatrix} Q_{1cl}(h) & Q_{12cl}(h) \\ Q_{12cl}^T(h) & Q_{2cl}(h) \end{bmatrix} \\ Q_{1cl}(h) &= Q_1(h) + Q_{12}(h)K(h)\Theta + \Theta^T K^T(h)Q_{12}^T(h) + \\ &\quad \Theta^T K^T(h)Q_2(h)K(h)\Theta \\ Q_{12cl}(h) &= Q_{12}(h)H(h) + \Theta^T K^T(h)Q_2(h)H(h) \\ Q_{2cl}(h) &= H^T(h)Q_2(h)H(h) \end{aligned}$$

Finally, assume that the initial state is  $\begin{bmatrix} z[0] \\ \hat{x}[0] \end{bmatrix}$ . The cost can then be evaluated as

$$\begin{aligned} J^{T_{\text{fbs}}} &= \begin{bmatrix} z[0] \\ \hat{x}[0] \end{bmatrix}^T S[0] \begin{bmatrix} z[0] \\ \hat{x}[0] \end{bmatrix} \\ &\quad + \sum_{k=1}^{T_{\text{fbs}}/h} \text{tr} S[k] \Gamma_{cl}(h) \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} \Gamma_{cl}^T(h) \\ &\quad + \frac{T_{\text{fbs}}}{h} K^T(h)Q_2(h)K(h)R_2 + \frac{T_{\text{fbs}}}{h} J_{\text{const}}(R_1, h) \end{aligned} \quad (19)$$

where

$$S[k] = \sum_{m=0}^{T_{\text{fbs}}/h-k} (\Phi_{cl}^T(h))^m Q_{cl}(h) (\Phi_{cl}(h))^m$$

In summary, we have shown that the cost can be written as a sum of five terms, where the two first terms represent the cost due to the initial state, while the three last terms represent the expected cost due to future noise.

## 4 Efficient On-Line Optimization

Computing (19) on-line would be expensive. However, it can be noted that almost everything in (19) can be precomputed off-line and stored in a look-up table. On-line, we must only account for the current initial state, noise intensity, and sampling interval.

### 4.1 Approximate Evaluation of the Cost

In (19), the initial state  $m_0$  is somewhat problematic, since the plant state  $z[0]$  is not known by the feedback scheduler. However, if we assume that the controller contains a state observer and that  $\hat{x}[0]$  represents the current state estimate, we may use

$$m_0 \approx \begin{bmatrix} \hat{x}[0] \\ \hat{x}[0] \end{bmatrix} \quad (20)$$

Finally, under the assumption that the noise level will be constant during the scheduling period and equal to  $R_1 = r_1 \bar{R}_1$ , the expected cost can be written as

$$J^{T_{\text{fbs}}} \approx \hat{x}^T[0] S_x(h) \hat{x}[0] + r_1 J_1(h) + J_2(h) \quad (21)$$

where

$$\begin{aligned} S_x(h) &= S_{11}[0] + S_{12}[0] + S_{21}[0] + S_{22}[0] \\ J_1(h) &= \sum_{k=1}^{T_{\text{fbs}}/h} \text{tr} S[k] \Gamma_{cl}(h) \begin{bmatrix} \bar{R}_1 & 0 \\ 0 & 0 \end{bmatrix} \Gamma_{cl}^T(h) \\ &\quad + \frac{T_{\text{fbs}}}{h} J_{\text{const}}(\bar{R}_1, h) \\ J_2(h) &= \sum_{k=1}^{T_{\text{fbs}}/h} \text{tr} S[k] \Gamma_{cl}(h) \begin{bmatrix} 0 & 0 \\ 0 & R_2 \end{bmatrix} \Gamma_{cl}^T(h) \\ &\quad + \frac{T_{\text{fbs}}}{h} K^T(h) Q_2(h) K(h) R_2 \end{aligned}$$

Table 1: Ordered table with  $\Pi_{i,k}$  values for each plant and sampling periods.

	$h_1$	$h_2$	$\dots$	$h_{j-1}$	$h_j$
Plant 1	$\Pi_{1,1}$	$\Pi_{1,2}$	$\dots$	$\Pi_{1,(j-1)}$	$\Pi_{1,j}$
Plant 2	$\Pi_{2,1}$	$\Pi_{2,2}$	$\dots$	$\Pi_{2,(j-2)}$	$\Pi_{2,j}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
Plant n	$\Pi_{n,1}$	$\Pi_{n,2}$	$\dots$	$\Pi_{n,(j-1)}$	$\Pi_{n,j}$

Note that  $S_x(h)$  is a matrix function, while  $J_1(h)$  and  $J_2(h)$  are scalar-valued functions. They can be computed off-line for each plant  $i$  and sampling period  $h$ . Note that they do not depend on the current state estimate nor on the current noise level. Therefore, they can be stored in a table for run-time lookup. The on-line computations involves looking up the relevant values from the table and applying Eq. (21).

## 4.2 Structure

For the implementation, we construct a table as illustrated in Table 1 with entries  $\Pi(i, k) = \{S_x(i, k), J_1(i, k), J_2(i, k)\}$ , where  $i$  is the plant and  $k$  is the position of the sampling period inside an ordered vector with  $h_1$  being the smallest sampling period and  $h_j$  the largest. Note that the number of sampling periods  $h^*$  for each control task within the specified ranges  $[h_i^{min}, h_i^{max}]$  depends on the granularity  $H$ , defined as the difference between two consecutive periods,  $H = h_{k+1} - h_k$  (and  $T_{fbs} \bmod H = 0$ ).

## 4.3 Algorithm

Given estimates of the current noise levels ( $\hat{r}_i$ ) and plant states ( $\hat{x}_i$ ) the algorithm that we propose searches for the minimum of the cost function. The basic idea is to start with a non-schedulable solution where all task periods at their minimal values. Then the task periods are incremented successively in such a way that the cost function increment  $\Delta J(i)$  in each step is always minimal, until the task set is feasible. The algorithm is shown in Figure 3.

By doing so, the algorithm assumes that each task should run with its shortest period in order to achieve the minimum cost, i.e. the cost is a convex, increasing function on the sampling period. However, this is not a strict requirement. If the monotonicity assumption is removed, i.e., the cost is only convex, the stopping condition for the algorithm is to reach a set of

```

1:  $h = [h_1^{min}, h_2^{min}, \dots, h_n^{min}]$ 
2:  $C = [C_1, C_2, \dots, C_n]$ 
3:  $x = [x_1, x_2, \dots, x_n]$ 
4:  $r = [r_1, r_2, \dots, r_n]$ 
5: while  $\sum_{k=1}^n \frac{C_k}{h_k} > U_{sp} - U_{fbs}$  do
6:    $lower\_cost\_task = 1$ 
7:   for  $i = 1$  to  $n$  do
8:     if  $h(i) + H \leq h_i^{max}$  then
9:        $\Delta J(i) = J(h(i) + H, x(i), r(i)) - J(h(i), x(i), r(i))$ 
10:      if  $\Delta J(i) < \Delta J(lower\_cost\_task)$  then
11:         $lower\_cost\_task = i$ 
12:      end if
13:    end if
14:  end for
15:   $h(lower\_cost\_task) = h(lower\_cost\_task) + H$ 
16: end while
17: return  $h$ 

```

Figure 3: Algorithm for the implementation of the period assignment.

feasible periods (as before) and  $\forall i, \Delta J(i) > 0$ . The second condition forces to enlarge tasks periods as long as they imply a decrease in the cost.

Note that it is possible to simplify the complexity of the optimization algorithm down to  $O(kn)$ , where  $n$  is the number of plants and  $k$  is the number of possible sampling periods. In addition, the linear search applied to the table could be replaced by more effective search algorithms.

#### 4.4 A Note on Stability

It is well known that fast switching between stabilizing controllers may lead to an unstable closed-loop system [14]. In our case, however, the switching is assumed to be infrequent in comparison with the sampling intervals of the controllers, meaning that stability will not be an issue in practice. Still, if the control designer would like to assert stability for the family of controllers with different sampling intervals, it can be done using techniques such as finding a common Lyapunov function for the controllers [14].

#### 4.5 Numerical Example

Let us consider three control tasks, each one controlling an unstable plant whose model is given in (27). The task execution times are identical,  $C_1 = C_2 = C_3 = 2\text{ms}$ , and each task has three possible sampling periods,  $h_{1,2,3} =$

$\{20, 40, 60\}$ ms. The total available CPU utilization for control activities has been chosen to be  $U^{sp} = 0.19$ . Only two kinds of solutions can match these requirements: either all tasks must use the same (middle) sampling period,  $h_1 = h_2 = h_3 = 40$ ms, or each task must run at a different rate, e.g.  $h_1 = 20$ ms,  $h_2 = 40$ ms, and  $h_3 = 60$ ms.

For each control task and plant, three discrete-time LQG controllers have been designed for the different sampling period choices. For example, for a period of  $h = 20$ ms and a delay of  $\tau = h/10 = 2$ ms, taking into account a nominal noise intensity  $\bar{R}_{1c} = 0.0001BB^T$  in (2) and  $R_2 = 5 \cdot 10^{-5}$ , and assuming the following design matrices

$$Q_{1c} = 2C^T C = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}, \quad Q_{12c} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad Q_{2c} = 0.5$$

the discrete-time LQG regulator in the form (5)-(6) is characterized by

$$\begin{aligned} F &= \begin{bmatrix} -0.75548 & -0.32571 & 0.01451 \\ 0.72449 & 0.29775 & -0.01479 \\ 3.88150 & 1.63860 & -0.07661 \end{bmatrix} \\ G &= \begin{bmatrix} 1.6410 \\ -0.1901 \\ -0.1901 \end{bmatrix} \\ H &= [ 3.8815 \quad 1.6386 \quad -0.0766 ] \\ K &= [ -5.1284 ] \end{aligned}$$

The on-line selection of sampling periods for the three control tasks is done according to the cost function (21) and considering the algorithm shown in Figure 3. Suppose that the initial state estimates for each plant when the Feedback Scheduler is invoked are  $\hat{x}_1 = [0.5, 0.5, 0.5]^T$ ,  $\hat{x}_2 = [1, 1, 1]^T$ ,  $\hat{x}_3 = [1.5, 1.5, 1.5]^T$ , while the noise level is high for Plant 1,  $r_1 = 10$ , and low for Plant 2 and 3,  $r_2 = r_3 = 1$ .

The algorithm begins by assigning the shortest periods,  $h_1 = h_2 = h_3 = 20$ ms, as shown in Table 2. The period increment is  $H = 20$ ms. For this set of periods, the task set is not feasible,  $U = 0.3 > 0.19$ . We therefore start the first iteration of the algorithm by calculating  $J_i(h)$  and  $J_i(h + H)$ . We increment the sampling period for the control task of Plant2 because it implies the smallest cost increment. The task set with the new periods,  $h_1 = 20$ ms,  $h_2 = 40$ ms, and  $h_3 = 20$ ms, is still not feasible,  $U = 0.25 > 0.19$ . We therefore calculate a second iteration, and we conclude to increment the sampling period for the control task of Plant3 because it implies the

Iter.		$J_i(h)$	$J_i(h + H)$	$\Delta J_i$	CPU Utilization
Init	$h_{1,2,3} = 20$				$0.30 > 0.19$
1st	P1 ( $h_1 = 20$ )	0.554	1.197	0.644	
	P2 ( $h_2 = 20$ )	0.082	0.157	<b>0.075</b>	$\longrightarrow h_2 = 40$
	P3 ( $h_3 = 20$ )	0.116	0.205	0.090	
					$0.25 > 0.19$
2nd	P1 ( $h_1 = 20$ )	0.554	1.197	0.644	
	P2 ( $h_2 = 40$ )	0.157	0.274	0.117	
	P3 ( $h_3 = 20$ )	0.116	0.205	<b>0.090</b>	$\longrightarrow h_3 = 40$
					$0.20 > 0.19$
3th	P1 ( $h_1 = 20$ )	0.554	1.197	0.644	
	P2 ( $h_2 = 40$ )	0.157	0.274	<b>0.117</b>	$\longrightarrow h_2 = 60$
	P3 ( $h_3 = 40$ )	0.205	0.328	0.123	
					$0.18 \leq 0.19$

Table 2: Algorithm example

smallest cost increment. Still the task set with the new periods,  $h_1 = 20\text{ms}$ ,  $h_2 = 40\text{ms}$ , and  $h_3 = 40\text{ms}$ , is not feasible,  $U = 0.20 > 0.19$ . Hence, we perform a third iteration iteration and we conclude to increment the sampling period for the control task of Plant2 because it implies the smallest cost increment. Now, the task set with periods  $h_1 = 20\text{ms}$ ,  $h_2 = 60\text{ms}$ , and  $h_3 = 40\text{ms}$  is feasible,  $U = 0.18 \leq 0.19$ , and the algorithm stops and returns these periods.

## 5 Disturbance Estimators

The cost function (21) depends heavily on the variance of the process noise and the measurement noise. Consequently, feedback scheduling schemes assuming stationary noise processes are not ideally suited to handle situations when the level of noise changes dynamically. In the following we describe two different on-line estimators for the intensity of the process noise,  $R_{1c}$ . The noise estimates can then be used by the feedback scheduler to obtain updated finite-horizon cost functions, which are then used to compute the optimal task periods. The performance of two proposed estimators are evaluated using real data.

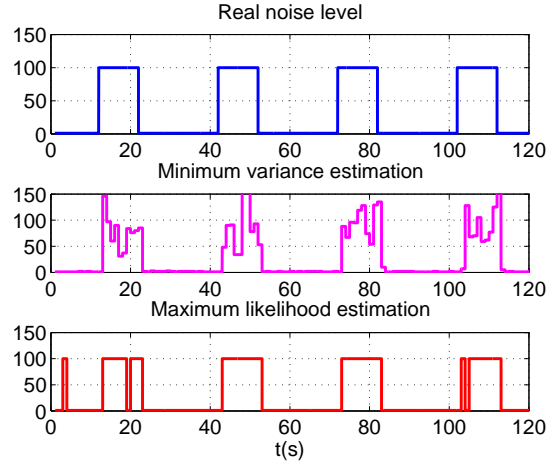


Figure 4: Evaluation of estimators.

## 5.1 Estimators

The task of the estimators is to produce estimates of the unknown process noise intensity,  $r_1$ , given measurements of the plant output,  $y$ . The estimators are based on the residuals,  $\tilde{y} = y - \hat{y}$ , for which the variance is given by

$$\sigma^2 = \mathbf{E}(\tilde{y}\tilde{y}^T) = C\tilde{P}C^T, \quad (22)$$

where

$$\tilde{P} = \mathbf{E}(xx^T) - 2\mathbf{E}(x\hat{x}^T) + \mathbf{E}(\hat{x}\hat{x}^T) \quad (23)$$

depends on the plant, the noise, and the controller design.

Table 3: Evaluation of the noise estimation error

Estimator	Plant	Mean	Standard deviation
Residual Variance Estimator	Plant1	2.9275	35.4591
	Plant2	4.0785	34.8543
	Plant3	4.5600	27.5753
Maximum-likelihood Estimator	Plant1	0.4950	27.1081
	Plant2	0.9900	27.9843
	Plant3	0.4950	25.2356

### 5.1.1 Residual Variance Estimator

The first method uses the observations of  $\tilde{y}$  collected during the last feedback scheduler period to compute the variance estimate

$$s^2 = \frac{\sum_1^N \tilde{y}^2}{N}, \quad (24)$$

where  $N = \frac{T_{\text{fbs}}}{h}$  is the number of recorded observations. The value of  $s^2$  is then used in a look-up table of pre-computed values of  $r_1$  to obtain the corresponding estimate,  $\hat{r}_1$ .

### 5.1.2 Maximum-Likelihood Estimator

In the second method, we again sum the squares of the residuals,  $\tilde{y}$ , in the feedback scheduler period. Using the fact that, if  $\tilde{y}_i \in N(0, \sigma)$ , then

$$\frac{1}{\sigma^2} \sum_1^N \tilde{y}_i^2 \in \chi^2(N), \quad (25)$$

a maximum-likelihood approach can be used to estimate which of several values of  $r_1$  that is most likely given the observations. This approach could be useful if it is a-priori known that the intensity of the noise can change between certain levels.

## 5.2 Evaluation

The evaluation is performed on the experimental setup described in Section 6, where three plants are controlled by three control tasks implementing an LQG controller with the specific settings given in Section 7.2. The intensity of the process noise  $r(t)$  changes abruptly between the values 1 and 100. The results of the noise intensities estimates  $\hat{r}_1$  for one of the plants are shown for both estimators in Figure 4.

By estimating the variance,  $\sigma^2$ , of the residuals,  $\tilde{y}$ , over a period of  $T_{\text{fbs}} = 1$  s, using Eq. (24) and translating the estimates to  $\hat{r}_1$ , we obtain the estimates shown in Figure 4. For the maximum-likelihood estimation we assume a-priori knowledge of the noise model, i.e., that the only two possible values of the process noise are 1 and 100. Given  $\tilde{y}^N = \sum_1^N \tilde{y}_i^2$  recorded also during 1 s, we choose the estimate  $\hat{r}_1 = 1$  if  $\tilde{y}^N < \tilde{y}^{\text{thres}}$  and  $\hat{r}_1 = 100$  otherwise, where  $\tilde{y}^{\text{thres}}$  is given by the intersection of the probability density functions for the two  $\chi^2$  distributions characterized by  $N = 25$  and  $h = 0.04$ s. Its estimates are also shown in Figure 4.

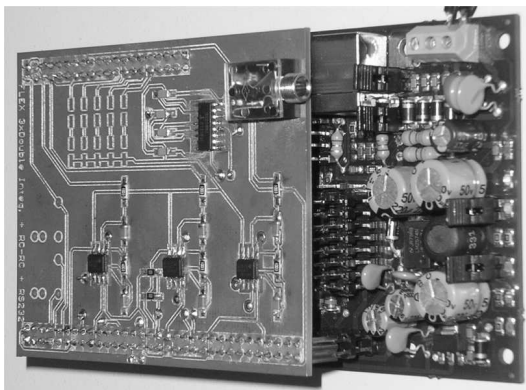


Figure 5: Experimental setup

Table 3 shows the mean value and standard deviation of the estimation error  $r(t) - \hat{r}(t)$  for the three plants. Both estimators are able to detect the changes in the noise intensity. In this case, the maximum likelihood estimator provides better numbers in terms of the mean and standard deviation of the error. However, if the noise changes were not that abrupt, the residual variance estimator would have performed better.

In addition, the implementation of the maximum likelihood estimator typically requires less resources. For both estimators,  $\tilde{y}^2$  can be computed by each control task in the controller update part, while the estimator is invoked by the feedback scheduler. However, in order to obtain accurate estimates using the residual variance estimator, the number of possible noise levels  $L$  used in the controller has to be large.

## 6 Experimental Set-up

As a prototype and performance demonstrator, a proof-of-concept implementation of the on-line sampling period assignment approach is presented. The setup consists of three plants in the form of double integrator electronic circuits that are controlled by three control tasks concurrently executing in the Erika real-time kernel [15] and scheduled under the EDF scheduling algorithm. The hardware platform is a Full Flex board [16] equipped with a dsPIC micro-controller. A fourth task, acting as feedback scheduler, periodically assigns new control task periods considering the current states and noise intensities of the plants.

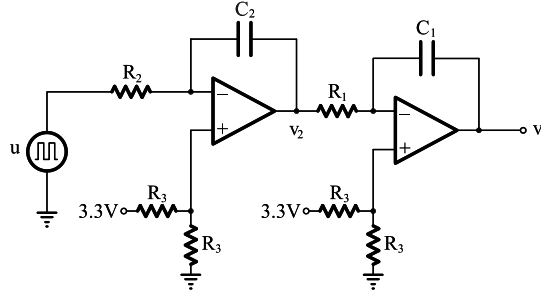


Figure 6: Electronic double integrator circuit

Component	Nominal value	Validated value
$R_{1/2}$	$1k\Omega$	$1k\Omega$
$R_1$	$100k\Omega$	$100k\Omega$
$R_2$	$100k\Omega$	$100k\Omega$
$C_1$	$470nF$	$420nF$
$C_2$	$470nF$	$420nF$

Table 4: Electronic components nominal and validated values

The three double integrators have been implemented on a custom daughter board that can be easily interfaced to the Full Flex base board (see details in the manual [17]), as shown in Figure 5. To debug and extract information from the micro-controller and plants, an RS232 link to a standard computer has been established. This link is managed from the computer side using Matlab<sup>®</sup>.

## 6.1 Plant Details

Each plant is an electronic double integrator, as illustrated in Figure 6. Note that in the integrator configuration, the operational amplifiers require positive and negative input voltages. Otherwise, they will quickly saturate. However, since the circuit is powered by the dsPIC, and thus no negative voltages are available, the 0V voltage ( $V_{ss}$ ) in the non-inverting input has been shifted from GND to half of the value of  $V_{cc}$  (3.3V) by using a voltage divider  $R_{1/2}$ . Therefore, the operational amplifier differential input voltage can take positives or negatives values. The nominal electronic components values are shown in table 4.

The operational amplifier in integration configuration can be model by

$$V_{\text{out}} = \int_0^t -\frac{V_{\text{in}}}{RC} dt + V_{\text{initial}} \quad (26)$$

where  $V_{\text{initial}}$  is the output voltage of the integrator at time  $t = 0$ , and ideally  $V_{\text{initial}} = 0$ , and  $V_{\text{in}}$  and  $V_{\text{out}}$  are the input and output voltages of the integrator, respectively.

Taking into account (26), and the scheme shown in Figure 6, the double integrator plant dynamics can be modeled by

$$\begin{aligned} \frac{dv_2}{dt} &= \frac{-1}{R_2 C_2} u \\ \frac{dv_1}{dt} &= \frac{-1}{R_1 C_1} v_2 \end{aligned}$$

In state space form, the model is

$$\begin{aligned} \begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} &= \begin{bmatrix} 0 & \frac{-1}{R_1 C_1} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{-1}{R_2 C_2} \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \end{aligned}$$

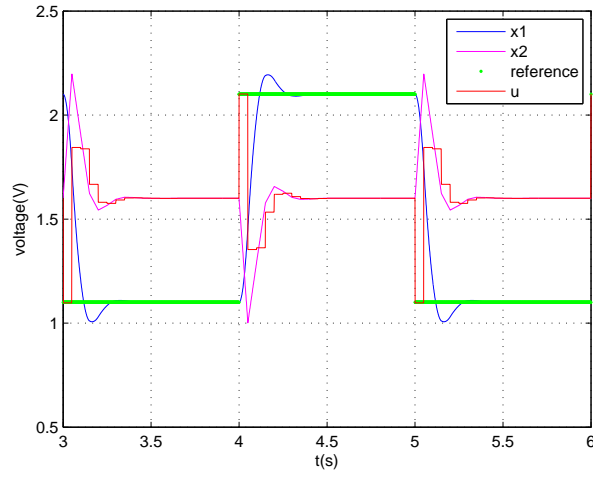
Taking into account the tolerances in the electronics components (5% for resistors and 25% for capacitors), the mathematical model that best matches the circuit dynamics is given by the validated values listed in table 4. The model validation has been performed applying a standard control algorithm with a sampling period of  $h = 50\text{ms}$ , with reference changes, and comparing the theoretical results obtained from a Simulink<sup>©</sup> model with those obtained from the plant.

With the validated values for the components, the model used for controller design is given by

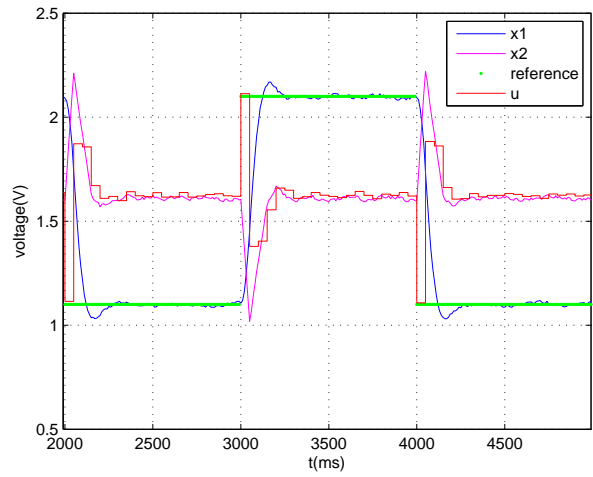
$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & -23.809524 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ -23.809524 \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x \end{aligned} \quad (27)$$

where the state vector is  $x = [v_1 \ v_2]^T$ .

Figure 7 shows the results of this validation. In particular, the controller gain designed via pole placement locating the desired closed loop poles at  $\lambda_{1,2} = -15 \pm 20i$  is  $K = [0.5029 \ -0.9519]$ . Since the voltage input of the operational amplifier is 1.6V (which is half  $V_{cc}$ : the measured  $V_{cc}$  is



(a) Theoretical simulated plant response



(b) Experimental plant response

Figure 7: Model validation.

3.2V although it is powered by 3.3V), the tracked reference signal has been established to be from 1.1V to 2.1V ( $\pm 0.5V$  around 1.6V). For the tracking, the feedforward matrix  $N_x$  is zero and  $N_x = [ 1 \ 0 ]$ .

The goal of the controller is to make the circuit output voltage ( $v_1$  in Figure 6) to track a reference signal by giving the appropriate voltage levels (control signals)  $u$ . Both states  $v_1$  and  $v_2$  can be read via the Analog-to-Digital-Converter (ADC) port of the micro-controller and  $u$  is applied to the plant through the Pulse-Width-Modulation (PWM) port.

## 6.2 Software Organization

Table 5 details the main software tasks required for the implementation. For each control task, the execution model presented in Section 2.2 is achieved by implementing two subtasks, `Calculate_Control` and `Update_Control`, and by configuring two interrupts in charge of the sampling and actuation operations, `Read_State` and `Apply_Control`.

The `Feedback_Scheduler` implements the algorithm shown in Figure 3, whose input parameters are the current states and noise intensities of each plant, and whose outputs are the set of new sampling periods. Two key parameters of this task are the period  $T_{\text{fbs}}$  and the granularity  $H$  of the algorithm.

Each plant is associated with a `Noise_Change` task, that facilitates the injection of noise with variable intensities  $r(t)$  and a configurable period  $h_{\text{noise}}$ . The period of the `Noise_Change` tasks as well as the activation times with respect to the feedback scheduler periodicity are also key parameters. In addition, the `Noise_Estimator` task is also implemented. Whenever used, each `Update_Control` task, apart from updating the controller state, computes  $\tilde{y}^2$  at each execution in order to speed up the execution of the `Noise_Estimator` task.

Finally, the `Send` task for debugging and monitoring has to have a period short enough to obtain the monitoring data with enough accuracy.

## 7 Experimental Results

Extensive experiments have been performed. Although only a few specific cases are summarized, the experiments are general and the results are not limited to these or any other special cases.

## 7.1 Key Parameters

The online period assignment approach has several key parameters that influence the achievable control performance and the resource demands:

- The period of the feedback scheduler task,  $T_{\text{fbs}}$ : It is assumed that  $T_{\text{fbs}} \gg h_i$ . A high value of  $T_{\text{fbs}}$  implies an implementation, which is less responsive in face of changes in the controlled plants (perturbations and noise) but also less computationally demanding. Finally, the relation of  $T_{\text{fbs}}$  and the frequency of the noise level changes must be also considered.
- The utilization set-point,  $U_{\text{sp}}$ : It must be large enough to accommodate all the control tasks as well as the feedback scheduler task.
- The number of sampling periods for each control task,  $h^*$ , and the granularity of the optimization algorithm  $H$ ; these parameters are highly related. The  $H$  granularity determines the number of sampling periods  $h^*$  for each control task. From a control perspective, it relates to the accuracy of the optimization algorithm, that is, the smaller the  $H$  (the bigger the number of possible periods  $h^*$ ), the better is the approximation to the exact solution of the optimization problem. (To get the optimal solution we need  $H \rightarrow 0$ .) On the other hand, the smaller the  $H$ , the bigger the computational overhead of the search algorithm and the memory requirements for storing the search table.
- Controller design strategy: The controller gains can be designed for a constant noise level or for several noise levels. The memory demands of the strategies are different. For a nominal noise level, the number of controllers to be stored in memory for a control task is  $h^*$ , while it is  $h^*L$  for the case of  $L$  different noise levels.
- The frequency of change of the noise intensities,  $h_{\text{noise}}$ : if the noise intensities change too frequently, the contribution of the feedback scheduling strategy with respect to noise attenuation will be negligible and the algorithm will perform similar to [5]. Hence, it is required to have  $h_{\text{noise}} \geq T_{\text{fbs}}$ .
- The number of noise intensity levels  $L$ : this determines the type of estimator to be used, as well as the number of controllers to be stored in memory and the parameters required by the estimator itself (translation table or thresholds for the variance or maximum likelihood estimator, respectively).

- Noise level estimation: The used estimator depends on the noise pattern present in the plant.

In addition to the implementation of the optimal sampling period selection approach, a complementary implementation has been also coded into the Erika kernel. It corresponds to the traditional static, periodic case: the three control tasks always execute with the same sampling period, irrespective of noise changes. In some of the evaluated scenarios, small modifications have been also applied to the periodic case in order to provide more illustrative performance numbers.

In the following subsections we characterize the performance of the proposed approach considering a base scenario, and then we modify several of the listed parameters to achieve a complete performance analysis.

## 7.2 Base Scenario

For the results presented in this section, the following parameters characterizing the base scenario have been applied:

- The feedback scheduler task period is  $T_{\text{fbs}} = 1000\text{ms}$ .
- The utilization set-point is  $U_{\text{sp}} = 0.0135$ .
- Sampling period choices for each controller are defined as follows. Considering that the feedback scheduler execution time is  $E_{\text{fbs}} = 1.6\text{ms}$ , the available CPU utilization for the three control tasks is  $U_{\text{sp}} - U_{\text{fbs}} = 0.0135 - \frac{1.6}{1000} = 0.0119$ . Considering that the execution time of each controller is  $E_{1,2,3} = 0.180\text{ms}$  (0.083ms for the calculate part and 0.097ms for the update part), and the minimum period is specified to be  $h_{1,2,3}^{\text{min}} = 20\text{ms}$ , the longest period is  $h_{1,2,3}^{\text{max}} = 90\text{ms}$ . Hence, for the online period assignment, sampling period choices for each control task range from 20 to 90ms. For the baseline periodic implementation, sampling periods for control tasks are always 40ms, which gives the same utilization factor  $U_{\text{sp}}$ .
- The granularity of the optimization algorithm is set to  $H = 10\text{ms}$ . Hence, each task can run at  $h^* = 8$  different sampling periods.
- Noise intensities, periodicity and noise estimator: Two noise levels named low and high are injected to the plant,  $r_i(t) = 1, 100$ . The artificial tasks injecting noise are configured in such a way that

- during an interval of 3000ms, one plant is injected a high intensity noise, i.e.  $r(t) = 100$ , while the others two are injected a low intensity noise, i.e.  $r(t) = 1$ . Hence  $R_{1c,low} = 1 \cdot R_{1,base} B^T B = [0 \ 0; 0 \ 0.1]$  and  $R_{1c,high} = 100 \cdot R_{1,base} B^T B = [0 \ 0; 0 \ 10]$ , where  $R_{1,base} = 0.0002209$ . The plant affected with the high intensity noise varies cyclically every 3000ms. We refer to this noise pattern as a noise with a period of  $h_{noise} = 3000$ ms.
  - Changes in noise intensities happen at the beginning of the executions of the feedback scheduler task, that is, noise and feedback scheduler are fully synchronized.
  - An ideal noise estimator is considered.
- Each control task implements an LQG controller considering the plant model (27), the cost function (7) with

$$Q_{1c} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad Q_{12c} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad Q_{2c} = 1,$$

the noise level as described above with  $R_2 = 5 \cdot 10^{-5}$  and parametrized according to the sampling period that applies among the possible choices. For the case of the controllers for the periodic baseline strategy, the parameters are the same but the noise level is specified to be the highest level that applies.

Each execution run lasts 200s and the monitoring task that sends the plant states and control signals through the RS232 link executes with a period of 5ms.

### 7.3 Performance Metric

The evaluation of the optimal on-line sampling period assignment and the periodic case has been performed for each plant according to the cost function

$$J = \sum_{k=0}^{200} \int_{0.005k}^{0.005(k+1)} \left\| \begin{matrix} x_i(t) \\ u_i(t) \end{matrix} \right\|_{Q_{ci}}^2 dt \quad (28)$$

Therefore, the cost evaluation for each plant is done using the states and control signals that are taken every 5ms. These values, evaluated in the cost function during 200s give the cost of each plant. Summing the three costs for the three plants, we obtain the accumulated overall cost.

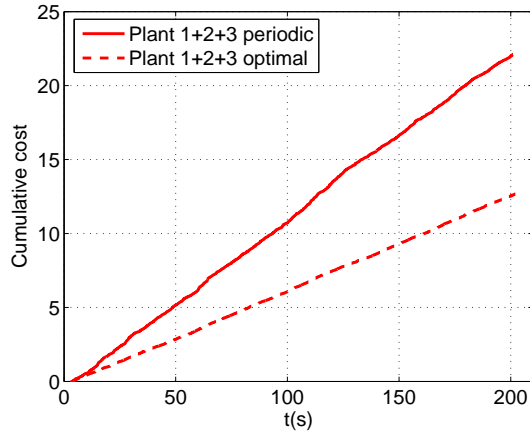


Figure 8: Cumulative cost of the optimal sampling period assignment approach, named “optimal”, vs. the standard periodic approach, named “Periodic”.

#### 7.4 Base Scenario Evaluation

The result for the base scenario is shown in Figure 8. The figure shows the accumulated cost for the three controllers using the on-line sampling period assignment presented in this paper, named “Optimal”, compared to the standard approach, named “Periodic”: the lower the curve, the lower the cost, i.e. the better the approach. As it can be seen, after 200s, the cost improvement of the optimal with respect to the static is around 40%. It is important to stress that depending on different tuning parameters such as those specified in Section 7.2, the exact percentage of improvement can be different.

Figure 9 shows details of the control performance of the standard periodic policy (sub-figure 9a) and the on-line period assignment policy (sub-figure 9b). Each sub-figure shows the dynamics of the three plants, each one affected by a high noise level during an interval of 3000ms. For the standard policy, the dynamics are strongly affected by the noise intensities. However, the optimal policy has the ability of mitigating the effects of the noise due to the online sampling period adjustment. Therefore, the cost for the optimal is lower than the cost for the periodic.

To further illustrate the behavior of the presented approach, Figure 10 shows the details of the periodic (sub-figure 10a) and optimal (sub-figure 10b)

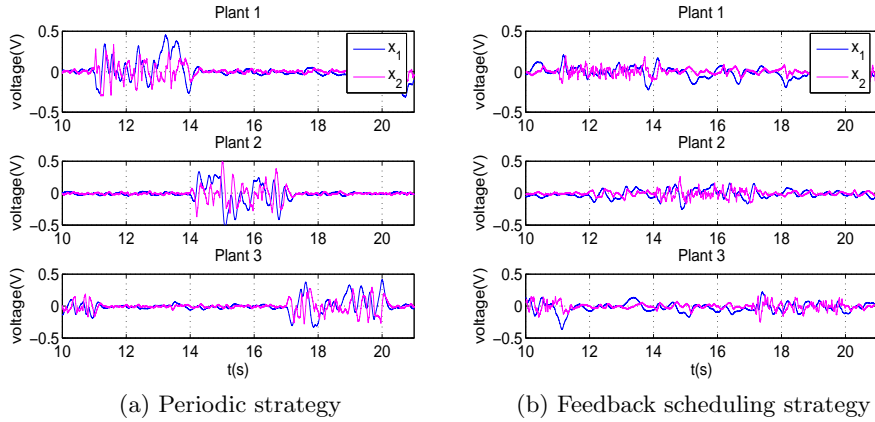


Figure 9: Plant dynamics for the three plants

approach for a given plant. Each sub-figure shows the plant response in the top and the sequence of sampling periods in the bottom. In the periodic, regardless of the noise that is affecting the plant, the sampling period is always the same, 40ms. In particular, during the time interval from 11s to 14s, the noise intensity for Plant1 is high. However, for the on-line policy, during the same time interval having a high intensity noise, the controller task period is changed by the feedback scheduler to 20ms, and its immediate effect is to mitigate the noise effects in the plant output. In addition, when the noise in the plant changes to a low intensity at time 14s, the controller period is reset to 80ms and later on to 90ms, giving room to other tasks that will need more CPU.

## 7.5 Single Controller

In the previous experiment, the three controllers for the optimal strategy were updating the control parameters according off course to the sampling period but also to the noise level intensity. However, the three controllers for the periodic strategy were constant.

Figure 11 shows the combined analysis of control performance and memory demands for the periodic and for the feedback scheduling approach following different controller design strategies. The first two “cost/memory” bars on the left show the performance of the periodic approach for two cases, where both controllers always execute at 40ms. The first cost/memory bar on the left (labeled “Periodic 1C”) is the case when only one controller gain applies, which corresponds to the upper curve in Figure 8. The memory

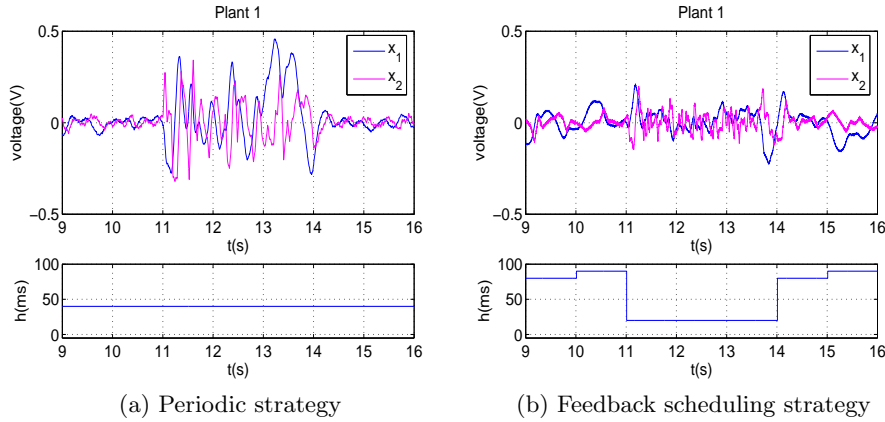


Figure 10: Plant dynamics and sampling periods for one plant

demand in this case is for storing one controller, which means storing matrices  $F$ ,  $G$ ,  $H$ , and  $K$ . In our case, the matrices contain 16 floats, that is, 64 bytes. The second cost/memory bar on the left (labeled “Periodic 2C”) is also the standard periodic case with the difference that two sets of control parameters apply depending on the noise level. The cost is a little bit better (lower) than the first case. The memory demand in this case is for storing two controllers, that is 128 bytes.

The two “cost/memory” bars on the right of Figure 11 show the performance of the optimal approach for two cases. The first cost/memory bar (labeled “Optimal 1C”) is the case when the controller gain is designed for one noise level for each sampling period. The cost is much better than the previous two periodic strategies. The memory demand in this case is for storing  $h^* = 8$  controllers, that is, 512bytes. The second cost/memory bar (labeled “Optimal 2C”) is the case when the control parameters that apply also depend on the noise level. The cost in this case is the best, which is the one also shown by the lower curve of Figure 8. The memory demand in this case is for storing  $h^*L$  controllers, that is 1 Kb.

The application of the feedback scheduling approach is clearly beneficial in terms of control performance at the expenses of using more memory.

## 7.6 Granularity $H$

The granularity  $H$  of the search table used by the optimization algorithm specifies the number of sampling period choices for each controller. In the base experiment,  $H = 10$  ms.

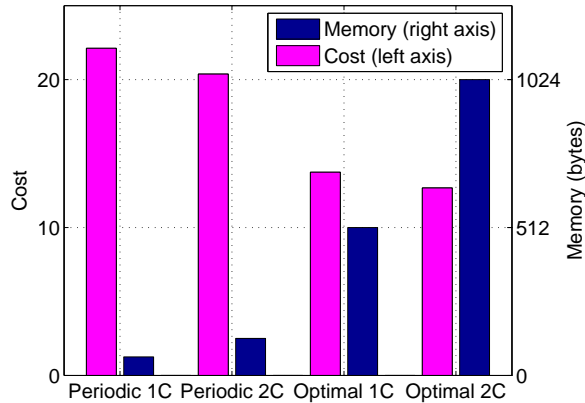
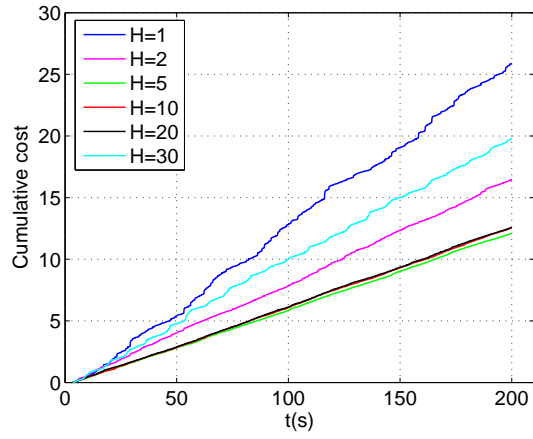


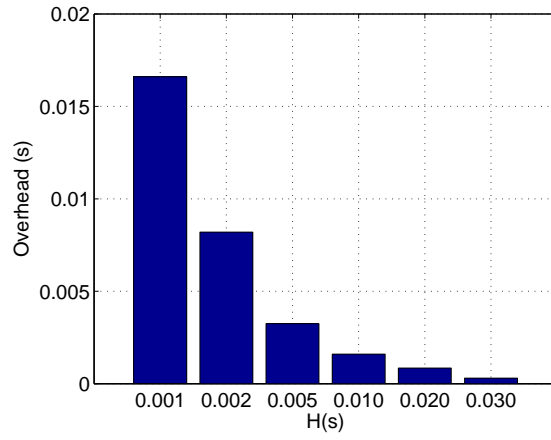
Figure 11: Cumulative cost and memory demand analysis of the optimal sampling period assignment approach, named “Optimal”, vs. the standard periodic approach, named “Periodic” with different controller gain design strategies.

Figure 12a shows the performance of the feedback scheduling strategy for different values of  $H$ . When the granularity is small, say  $H = 1$  ms (or  $H = 2$  ms), a number of  $\frac{90-20}{1} + 1 = 71$  (or 36) entries appears in the search table of the optimization algorithm. As a consequence the execution time of the feedback scheduler increases, and the additional overhead that this represents leaves no room for the control tasks to speed up their rate of execution to improve control performance, as shown by the top curve (or third highest curve). When the granularity is high, say  $H = 30$  ms, the number of period choices is very limited, and therefore, although the overhead of the feedback scheduler is very small, control tasks can not take advantage of the available CPU as efficiently, as shown by the second performance curve from the top. Hence, from a control point of view, the granularity  $H$  has to be chosen carefully, avoiding introducing significant overhead in the feedback scheduler but still giving enough period choices for each task. Good values in this experiment are  $H = 5, 10$  or  $20$ , which basically provides the same performance (the lower three curves).

Figure 12b complements the previous analysis by showing the CPU time consumed by the feedback scheduler for each  $H$  choice. It can be seen that as  $H$  decreases, the overhead increases exponentially. It has to be pointed out that better search algorithms in the optimization procedure of the feedback



(a) Control performance



(b) Computational overhead

Figure 12: Performance analysis of the algorithm granularity  $H$

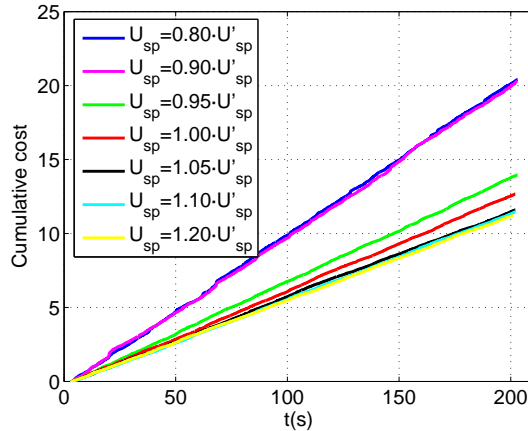


Figure 13: Cumulative cost of the optimal sampling period assignment approach for different CPU utilization set-points with respect  $U'_{sp} = 0.0135$ .

scheduler than the linear search that we have applied could result in lower computational overhead.

## 7.7 CPU Utilization Setpoint

In the base experiment, the CPU utilization set point was  $U_{sp} = 0.0135$ . Figure 13 shows the performance that can be achieved when the CPU utilization set-point varies. As expected, the higher the set-point, the lower the cost. However, as it can be seen in the figure, there is a point where having more CPU dedicated to the online assignment approach does not provide increased benefit in terms of control performance. This is due to the fact that with a high CPU set-point, all the controllers can run at their highest rate,  $h_{1,2,3} = 20\text{ms}$ , and therefore, no further improvement can be achieved with an even higher CPU set-point.

In addition, although not shown in the figures, it is worth noting that the higher the CPU set-point, the faster is the search algorithm because finding a schedulable solution is easier.

## 7.8 Noise Levels

The base experiment assumed only two noise levels. Figure 14 shows the performance of the feedback scheduling (labeled “Optimal”) and the periodic

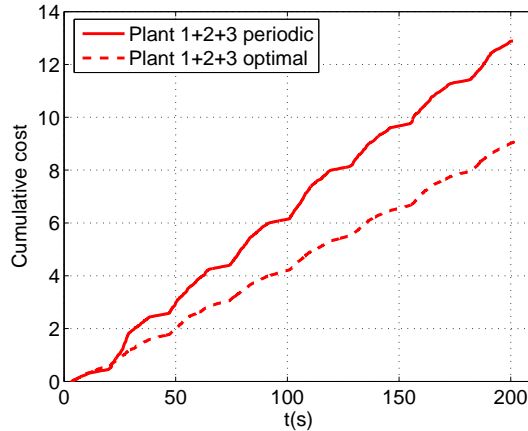


Figure 14: Cumulative cost of the optimal sampling period assignment approach when 4 noise levels are considered:  $r_1(t) = 1$  (extra-low),  $r_2(t) = 10$  (low),  $r_3(t) = 50$  (high),  $r_4(t) = 100$  (extra-high).

approach when the three plants are affected by four different noise levels. As it can be seen, the feedback scheduling approach achieves a substantial improvement. The achieved improvement is around 30% while in the case of two noise levels it was around 40%. The more noise levels, the less aggressive are the resulting controllers, and therefore, less control performance improvement can be expected.

## 7.9 Feedback Scheduler Period vs Noise Period

In the base experiment, the period of the feedback scheduler was  $T_{\text{fbs}} = 1000\text{ms}$  while the period of the noise intensity changes was  $h_{\text{noise}} = 3000\text{ms}$ . Figure 15 shows the behavior of the online period assignment approach with respect to different values of the period of the feedback scheduler while keeping the same noise intensity rate of change to  $h_{\text{noise}} = 3000\text{ms}$ .

As it can be seen in the figure, as long as  $T_{\text{fbs}}$  is smaller than  $h_{\text{noise}}$ , the presented approach delivers good numbers. However, when this relation does not hold, as for the case of  $T_{\text{fbs}} = 4\text{s}$ , then, the performance drastically decreases. The reason is that for these cases, the feedback scheduler is not responsive enough, and the given task periods are not adjusted anymore to the correct real noise levels because they also change within a  $T_{\text{fbs}}$ . Hence, the presented approach requires having  $T_{\text{fbs}} \leq h_{\text{noise}}$  in order to provide good

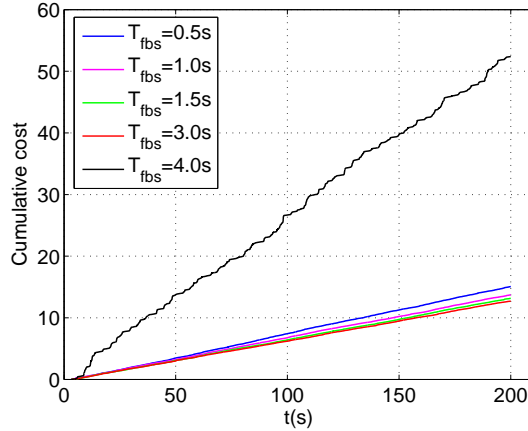


Figure 15: Cumulative cost of the optimal sampling period assignment approach for different period values of the feedback scheduler.

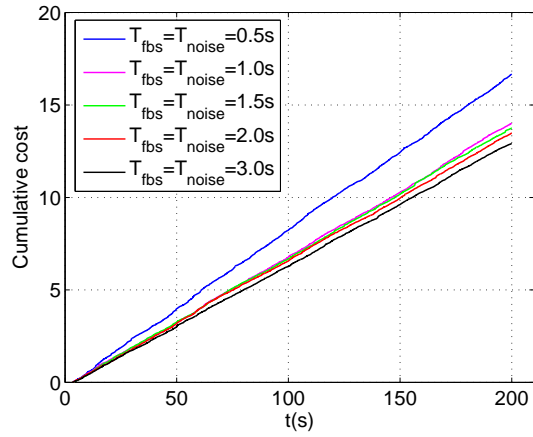
control performance results. Note also that a short noise period requires then a short  $T_{fb_s}$ , and therefore, the computational overhead of the feedback scheduler may also jeopardize the behavior of the approach, where also  $H$  plays an important role.

### 7.10 Noise Period

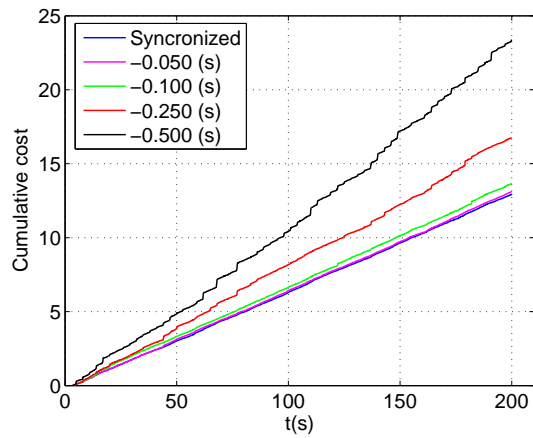
Up to now (except for the previous experiment) the noise changes occurred exactly when the feedback scheduler task starts executing, that is, when  $h_{noise} = T_{fb_s}$  or  $h_{noise} = m \cdot T_{fb_s}$ , i.e. they are synchronized, representing the best possible case. In this case, the feedback scheduler always assigns periods according to the correct noise intensities.

Figure 16a shows the control performance of the presented approach for different  $T_{fb_s}$  values while keeping the relation  $h_{noise} = T_{fb_s}$ . As it can be seen, the performance is satisfactory except when the period is small, due to the overhead introduced by many executions of the feedback scheduler.

Figure 16b shows the control performance of the presented approach for  $T_{fb_s} = 1000\text{ms}$  while not keeping the relation  $h_{noise} = T_{fb_s}$ . In this scenario, the synchronization is lost for different amounts of time. As it can be seen, as the synchronization decreases, the cost increases. Therefore, the successful operation of the online period assignment requires knowing the correct noise intensities at the right time. This requires good accuracy for



(a) Synchronized



(b) Unsynchronized

Figure 16: Cumulative cost of the optimal sampling period assignment approach for different patterns of the feedback scheduler periodicity and noise scheduler periodicity

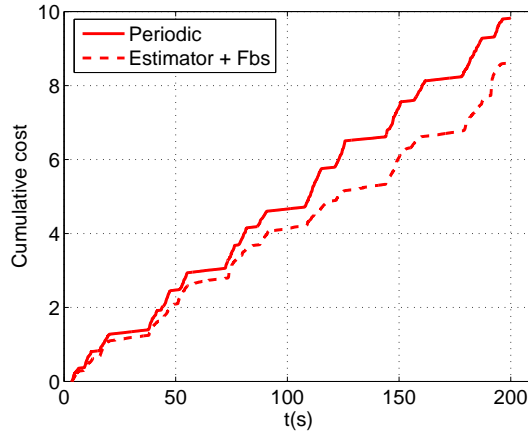


Figure 17: Cumulative cost of the optimal sampling period assignment approach for random noise periodicity and application of the noise estimator

the noise estimator, and a slow noise intensity rate of change.

### 7.11 Noise Estimator Performance

Up to now, the noise estimator was considered ideal, that is, the feedback scheduler had the exact noise intensity levels at each execution and during the  $T_{\text{fbs}}$  interval, and the noise levels did not change.

In Figure 17 we show the performance of the presented approach with a more realistic setting where noise intensities are estimated for each plant and their rate of change is randomly generated. In particular, the random  $h_{\text{noise}}$  is forced to be higher than the period of the feedback scheduler, that is, we injected random but slow noise intensity changes. In addition, the noise intensity estimation was performed by the maximum likelihood estimator executed by the feedback scheduler task. In terms of overhead, the maximum likelihood estimator only adds 0.015ms to each feedback scheduler execution and requires storing two thresholds for computing the correct estimates, one for each noise level. As it can be seen in the figure, the performance improvement with respect the static case is still significant (around 15%).

It must be noted that the performance improvement shown in Figure 17 should not be directly compared to previous performance numbers because the number of noise intensity changes is different, as well as the percentage time during which the high noise intensity applies. In addition, all the

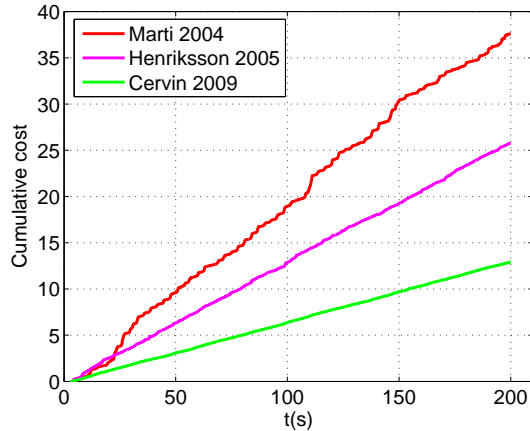


Figure 18: Cumulative cost of the optimal sampling period assignment approach with respect previous work.

previous case studies in terms of computational and memory overhead, and in terms of control performance also apply to this “real” scenario with the appropriated scaling. However, in order to be able to characterize which benefit is a consequence of a particular parameter, the ideal estimator was more suited for the detailed performance analysis.

## 7.12 Policies Comparison

To finalize the performance analysis of the online period assignment, the proposed approach is compared to two previous relevant approaches, Martí *et al.* 2004 [5] and Henriksson *et al.* 2005 [7]. The first approach considers cost function that only depends on current plant information. The second approach uses finite-horizon cost as in our approach but only considers stationary noise levels.

Using the same control settings for all the approaches as explained in the base experiment, the performance numbers are shown in Figure 18. As expected, Martí *et al.* 2004 [5] provide the lowest performance because only the current state is accounted for in the optimization and the noisy states highly jeopardize the operation of the approach. Then, Henriksson *et al.* 2005 [7] provides a medium performance curve because varying noise is not accounted for, and then the noise states predominate in the finite horizon optimization solution. Finally, the presented approach, labeled “Cervin 2009”,

provides the best performance curve.

## 8 Conclusions

In this paper, we have demonstrated how feedback from the control applications can be used to on-line optimize the performance. The feedback scheduler evaluates the expected cost over a finite time horizon, taking into account the noise intensity, the sampling interval, and the computational delay for each control loop. Based on this information, optimal sampling intervals are assigned to the control tasks.

As for the theoretical contribution of the paper, we have shown how to efficiently evaluate a quadratic cost function for a given linear plant and a general linear controller, taking the computational delay into account. The practical contribution of the paper is that we have demonstrated, on a set of real plants, how a real-time implementation with feedback scheduling can improve the control performance quite substantially compared to a state-of-the-art periodic implementation. The practical results agree well with the those predicted by the theory.

## References

- [1] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems*. Prentice Hall, 1997.
- [2] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, “On task schedulability in real-time control systems,” in *Proc. 17th IEEE Real-Time Systems Symposium*, 1996.
- [3] J. Eker, P. Hagander, and K.-E. Årzén, “A feedback scheduler for real-time control tasks,” *Control Engineering Practice*, vol. 8, no. 12, pp. 1369–1378, 2000.
- [4] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén, “Feedback-feedforward scheduling of control tasks,” *Real-Time Systems*, vol. 23, no. 1–2, pp. 25–53, Jul. 2002.
- [5] P. Martí, C. Lin, S. A. Brandt, M. Velasco, and J. M. Fuertes, “Optimal state feedback based resource allocation for resource-constrained control tasks,” in *Proc. 23rd IEEE Real-Time Systems Symposium*, 2004.

- [6] P. Martí, C. Lin, S. A. Brandt, M. Velasco, and J. M. Fuertes, “Draco: Efficient resource management for resource-constrained control tasks,” *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 90–105, January 2009.
- [7] D. Henriksson and A. Cervin, “Optimal on-line sampling period assignment for real-time control tasks based on plant state information,” in *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*, Seville, Spain, Dec. 2005.
- [8] R. Castañé, P. Martí, M. Velasco, A. Cervin, and D. Henriksson, “Resource management for control tasks based on the transient dynamics of closed-loop systems,” in *Proceedings of the 18th Euromicro Conference on Real-Time Systems*, Dresden, Germany, Jul. 2006.
- [9] M.-M. B. Gaid, A. Çela, Y. Hamam, and C. Ionete, “Optimal scheduling of control tasks with state feedback resource allocation,” in *American Control Conference 2006*, June 2006.
- [10] M.-M. B. Gaid, A. Çela, and Y. Hamam, “Optimal real-time scheduling of control tasks with state feedback resource allocation,” *IEEE Transactions on Control Systems Technology*, vol. 17, no. 2, pp. 309 – 326, mar 2009.
- [11] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment,” *Journal of the ACM*, vol. 20, no. 1, pp. 40–61, 1973.
- [12] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo, *Deadline Scheduling for Real-Time Systems—EDF and Related Algorithms*. Kluwer, 1998.
- [13] K. J. Åström, *Introduction to Stochastic Control Theory*. New York: Academic Press, 1970.
- [14] D. Liberzon, *Switching in Systems and Control*. Birkhäuser, 2003.
- [15] “Erika enterprise, 2008, <http://www.evidence.eu.com/content/view/27/254/>,” 2008, [Online; accessed 10-December-2008].
- [16] “Flex full base board, <http://www.evidence.eu.com/content/view/154/207/>,” 2008, [Online; accessed 10-December-2008].

- [17] “Flex, modular solution for embedded applications, verion 1.0.1, october 2008, [http://www.evidence.eu.com/download/manuals/pdf/flex\\_refman\\_1.0.1.pdf](http://www.evidence.eu.com/download/manuals/pdf/flex_refman_1.0.1.pdf),” 2008, [Online; accessed 10-December-2008].

Functionality	Name of task	Type	Comments
<b>Feedback Scheduler</b>	<code>Feedback_Scheduler</code>	RT TASK	Periodic real-time task with period $T_{\text{fbs}}$ that computes suitable periods for the three control tasks
<b>For each controller</b>	<code>Calculate_Control</code>	RT TASK	Periodic real-time task that computes the control signal with the period $h$ given by the feedback scheduler and a relative deadline equal to $h/10$
	<code>Update_Control</code>	RT TASK	Periodic real-time task that updates the controller state with the same period as the <code>Calculate_Control</code> task but with a relative deadline equal to $h$
	<code>Read_State</code>	RX Interrupt	Time-triggered interrupt that samples the plant. Its period is the same as the <code>Calculate_Control</code> task
	<code>Apply_Control</code>	RX Interrupt	Time-triggered interrupt that applies the control signal to the plant. Its period is the same as the <code>Calculate_Control</code> task but with an offset of $h/10$
<b>Noise</b>	<code>Noise_Change</code>	RT TASK	Periodic real-time task with a tunable period $h_{\text{noise}}$ that injects noise to each plant
	<code>Noise_Estimator</code>	RT TASK	Periodic real-time task that implements the noise intensity estimator. It is triggered by the <code>Feedback_Scheduler</code> task.
<b>Debugging</b>	<code>Send</code>	RT TASK	Periodic real-time tasks that sends data through the RS232 link for logging and debugging purposes every 5ms

Table 5: Software tasks in each board.