Resource Management for Control Tasks Based on the Transient Dynamics of Closed-Loop Systems

Castañé, Rosa; Martí, Pau; Cervin, Anton; Henriksson, Dan

2006

Resource Management for Control Tasks Based on the Transient Dynamics of Closed-Loop Systems

Rosa Castañé, Pau Martí, Manel Velasco
Automatic Control Department
Technical University of Catalonia
{rosa.castane,pau.marti,manel.velasco}@upc.edu

Anton Cervin, Dan Henriksson
Department of Automatic Control
Lund University
{anton,dan}@control.lth.se

Abstract

This paper presents a resource management strategy for control tasks that maximizes control performance within the available resources by readjusting the task periods at run-time. A feedback scheduler is used to determine on-line the optimal task periods considering the response over a finite time horizon of the plants controlled by arbitrary linear control laws. We show how this problem can be expressed as an optimization problem, where the objective function relates the sampling periods to the transient responses of the controlled plants, and where restrictions are based on EDF schedulability constraints. For the general case, the solution of the optimization problem is computationally expensive, and thus, an approximate procedure to be executed on-line has been developed. We present simulation results that validate the presented approach.

1. Introduction

To achieve optimal performance in embedded control systems, the limited computational resources must be used as efficiently as possible. The traditional approach to real-time control [1, 2] is to implement each controller as a periodic task with a fixed period. This can be viewed as an open-loop approach, where no information about the run-time performance of the control loops is taken into account.

In this paper, we propose a resource management strategy for control tasks that maximizes control performance within the available resources by readjusting tasks periods at run-time. A feedback scheduling framework [3] is used to determine the optimal task periods on-line. The main idea behind the presented feedback scheduling approach is the observation that a controlled plant in a transient phase (e.g., during a set-point change or an external disturbance) may require more resources, that is, smaller sampling period, than a controlled plant in stationarity [4].

In the resource management approach taken in this paper, optimal periods for control tasks are chosen considering the response over a finite time horizon of the plants controlled by arbitrary linear control laws. This permits us to specify the response characteristics of all plants using, for example, pole-placement techniques. We show how this problem can be expressed as an optimization problem, where the objective function relates the transient response of controlled plants to the sampling periods, and where restrictions are based on EDF schedulability constraints. Since the solution of the optimization problem is computationally expensive in the general case, a simple procedure to be executed on-line is developed, which approximates the optimal solution.

Previous feedback scheduling approaches have neglected the impact of latency and jitter on the control system performance. Changing sampling periods on-line, the resulting delay patterns for different configurations are very difficult to predict. To solve this problem, the controllers considered in this paper are implemented using Control Servers [5]. A Control Server creates the abstraction of a control task with a specified period and fixed input-output latency (shorter than the period). I/O occurs at fixed predefined points in time, at which inputs are read or controller outputs become visible, thus removing I/O jitter.

The presented approach is validated by simulation of a set of controllers for three second-order plants. The simulations show that dynamic resource management based on cost functions can improve the performance of the controllers. Also, the efficiency of the optimization procedure is demonstrated.

1.1. Problem Formulation

The system we consider is a real-time system consisting of n control loops (see Figure 1). Each plant, i, is controlled by a control task with an adjustable sampling period, h_i, executing the corresponding control law, L_j^i, where j is the index of the currently used sampling interval of task i. The

Proceedings of the 18th Euromicro Conference on Real-Time Systems (ECRTS'06)
0-7695-2619-5 /06 $20.00 © 2006 IEEE

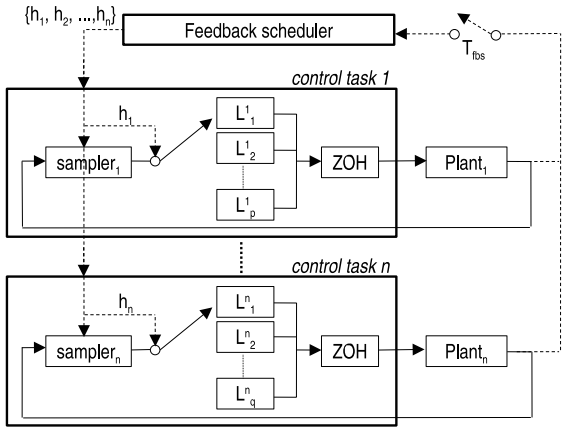

Figure 1. Model of the real-time system

number of control laws per task may not be the same. At the interface between the computer and the plant, the control signal is held constant between updates (zero-order hold).

Each task period can take values in a predefined range $[h_i^{min} \dots h_i^{max}]$, determined by the control specifications. Due to resource constraints, periods for control tasks are chosen large enough to improve schedulability of other non-control tasks, to meet control performance requirements, as well as permitting control performance improvement when selecting short periods within the given ranges.

In addition, the real-time system has to be capable of assigning task periods $h_1, h_2, \dots, h_n$ at run-time considering the cost of each plant response with respect to the task periods. Therefore, the concept of feedback is used at two levels. Apart from the standard feedback used by the control tasks, the real-time system uses information from the plant responses to dynamically allocate resources between the competing tasks. It is assumed that the execution times of the control tasks are known and equal to $C_1, C_2, \dots, C_n$.

On-line sampling period assignment occurs at predefined time instants, which are given by the periodic execution of a high priority task, the feedback scheduler. The period of the feedback scheduler is denoted T_{fbs}.

The problem to be solved by the feedback scheduler is how to assign sampling periods to the set of n control tasks such that

- the overall control performance is maximized,
- the schedulability constraints are kept, and
- the introduced overhead due to period re-assignment is kept as small as possible.

The computation of the optimal periods can be formulated as a constrained optimization problem as follows:

$$\min_{h_1 \dots h_n} \sum_{i=1}^{n} J_i \qquad \text{subj. to} \sum_{i=1}^{n} \frac{C_i}{h_i} \leq U_{sp} \tag{1}$$

where the J_i are functions expressing the cost of running control loop i. The cost functions depend on the sampling interval, h_i, the controller design, and the transient and steady-state effect of external disturbances affecting the plant. The minimization of the aggregate cost is subject to a utilization constraint, where C_i are the worst-case execution times of the tasks. In our work we assume EDF schedulability constraints, that is, $U_{sp} = 1$.

1.2. Contributions of the Paper

As a first contribution, we use a cost function presented previously [3] but extend one of its capabilities. The original cost function and overall optimization procedure was presented for controllers designed using optimization techniques, i.e. linear-quadratic (LQ) controllers. In this paper, we allow for any linear controllers to be used, including those designed using pole-placement techniques.

As a second contribution, we deal with the computational delay and jitter in the implementation by using Control Servers [5]. The cost functions are computed taking the resulting constant delay into account. This means that the cost functions will more closely reflect the true performance in the implementation.

As a third contribution, we present an algorithm that allows solving the optimization problem at run-time. We show that, for the general case, the exact solution to the optimization problem requires extensive computations, which impair its implementation for a run-time resource allocator. Instead, we derive a simple algorithm that permits solving the problem at run-time with a feasible computational cost.

1.3. Related Work

Research on selecting optimal sampling periods for control tasks via optimization techniques can be divided into three main areas.

- First, those works concerned with the definition of the objective function (cost function). The objective function to be minimized (maximized) should define the relation between the sampling period and control performance, and therefore, capture how the dynamics of each control plant are influenced by the choice of the sampling period. It may also determine the way the controllers should be designed.

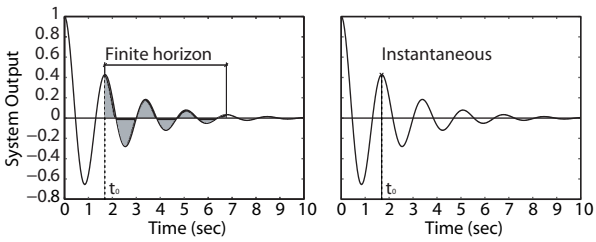

Figure 2. Finite-horizon (left) vs instantaneous (right) cost measurements

- Second, those works that focus on the domain of the optimization problem, i.e., the restrictions. For real-time systems, restrictions are imposed by the scheduling algorithm. Specifically, restrictions are usually given in the form of schedulability tests based on utilization factors.

- Third, those works that focus on finding solutions to the optimization problem. Important questions in this case are whether the optimization problem has a solution or not, whether the algorithm can always find a solution, and whether the optimization routine can be executed on-line.

The work in this paper focuses on the first and third area.

Selection of optimal sampling periods for control tasks via optimization techniques has been extensively treated in the literature.

Seto *et al.* [6] first introduced the idea of sampling rate optimization under CPU utilization constraints. A performance index expressed as a function of the sampling rate was used as basis for the optimization. However, their approach focused on off-line optimization. Along the same lines, [7], developed a general period optimization approach for fixed-priority scheduling.

Palopoli *et al.* [8] studied the problem of joint period selection and feedback design for multiple control loops, using the stability radius rather than a quadratic cost function as the performance criterion.

Martí *et al.* [4] presented the idea of dynamic resource allocation based on state feedback from the controlled processes. This work, however, based the optimization on the definition of state-dependent cost functions, and not on the state evolution over a period of time.

Henriksson *et al.* [3] used LQ controllers, and expressions related to the expected cost to the sampling period and the plant states were derived and used for on-line sample-rate adjustments. However, its feasibility from a computational overhead point-of-view was not addressed.

2. Cost Function Design

The cost functions must capture the design goals and must measure the application performance. The goal of the feedback scheduling architecture is to maximize the overall control performance, which is traditionally measured in terms of the deviation of the plant response with respect to a given set-point, and in terms of the energy spent to correct such deviations. For both metrics, the smaller the measure, i.e., the smaller deviation and the smaller the energy spent, the better the control performance. Therefore, maximization of control performance becomes minimization of a cost function stated in terms of the plant response deviation (error) and the energy spent.

In the rest of this section, we construct the cost function, justifying all the decisions taken in its development.

2.1. Motivating Discussion

For each plant, the cost function can be based on instantaneous measures or on finite (or infinite) time-horizon measures (note that the latter will not be a measure, but rather a prediction from a measure based on a model of the plant). For example, [4] considered optimization based on a instantaneous cost function. On the other hand, [3] considered optimization based on a finite-horizon cost function.

Figure 2 illustrates the difference. Each sub-figure represents the response of a controlled plant over time. At time instant t_0, a measure of the control performance is taken. In the right figure, an instantaneous measure is taken, that is, the response value at time t_0 is the measure. On the other hand, in the left figure, the measure is the integral of the absolute value of the response from t_0 over a finite time horizon.

Optimization at time t_0 could be based on any of them. However, consider the following scenario illustrated in Figure 3. Here we show the response of two plants that are controlled by two control tasks. At time t_0, both have the same instantaneous error. Therefore, optimization based on the instantaneous measure will not be able to favor any of the two tasks in terms of period re-assignment. However, the foreseen evolution of each response, which can be captured by a time-horizon measure, is completely different in terms of expected deviations with respect to the equilibrium point 0. Here, the optimization will be able to favor one of the two tasks. It has to be pointed out that the evaluation horizon could also be infinite. Motivated by this, we focus our effort on constructing the cost function taking into account finite-horizon costs.

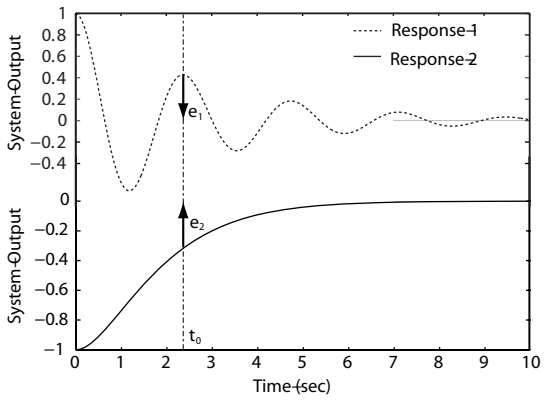

Figure 3. Illustrative example

2.2. Intuitive Approach

Each plant is assumed to be described by a linear system,

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t) + \upsilon(t) \\ y(t) &= Cx(t) \end{aligned} \tag{2}$$

where $x(t)$ is the plant state, $u(t)$ is the control signal, $y(t)$ is the measurement signal, and $\upsilon(t)$ is a zero mean white noise process with variance R that captures the influence of future disturbances. A, B, and C are constant matrices that describe the dynamics of the plant. The objective of the control is to keep the output y as close to zero as possible, despite the presence of external disturbances.

If we were interested in an instantaneous performance measure at time t_0, a cost function such as $J = |y(t_0)|$ would suffice. However, we are interested in the system evolution over a time interval. Therefore, we need to obtain the solution of the the differential equation (i.e., the plant response) and evaluate its deviation with respect to the equilibrium point over the time interval. Looking at sub-figure left in Figure 2, this would mean computing the grey area.

We assume that the control signal $u(t)$ is given by an arbitrary state feedback gain L, that is, $u(t) = -Lx(t)$. Disregarding $\upsilon(t)$ and assuming continuous feedback, the solution $y(t)$ of (2) is given by

$$y(t) = Ce^{(A-BL)(t-t_0)}x_0 \tag{3}$$

where $x_0 = x(t_0)$ is the initial state of the system, and $A - BL$ is the closed-loop system matrix.

To evaluate the expected performance, i.e., to compute the grey area, we can integrate $|y(t)|$ over a finite time horizon to give the measure we are looking for,

$$J = \int_{t_0}^{t_f} |y(t)|dt \tag{4}$$

where t_f is the end of the evaluation period. However, computing such an integral is expensive. Moreover, it will differ from the real response if large sampling periods are used. To capture the influence of the sampling period on performance, a sampled version of the plant and of the performance measure is needed. Sampling the plant with the hold interval h gives a discrete-time system in the form

$$\begin{aligned} x(kh+h) &= \Phi(h)x(kh) + \Gamma(h)u(kh) \\ y(kh) &= Cx(kh) \end{aligned} \tag{5}$$

Considering a control gain $L(h)$ obtained with an arbitrary controller design method, the closed-loop plant dynamics will have the form

$$x(kh+h) = \Big(\Phi(h) - \Gamma(h)L(h)\Big)x(kh) \tag{6}$$

From here, two different approaches can be used to approximate the cost function and to develop a suitable performance measure:

- Absolute value operator: The integral approximation could be expressed as the absolute values of $y(kh)$ over the time horizon $t_0 + Nh$, as in

$$J_a = \sum_{k=0}^{N-1} |y(t_0 + kh)| \tag{7}$$

Introducing $\Phi_{cl}(h) = (\Phi(h) - \Gamma(h)L(h))$, we get

$$\begin{aligned} x(t_0+h) &= \Phi_{cl}(h)x_0 \\ x(t_0+2h) = \Phi_{cl}(h)x(t_0+h) &= \Phi_{cl}^2(h)x_0 \\ &\vdots \\ x(t_0+Nh) &= \Phi_{cl}^N(h)x_0 \end{aligned} \tag{8}$$

Expression (7) can be rewritten in terms of (8) as

$$J_a = \sum_{k=0}^{N-1} |C\Phi_{cl}^k(h)x_0| \tag{9}$$

- Quadratic operator: instead of using the absolute value, a quadratic operator could be used. Expressing the sum of all quadratic values of $y(kh)$, we get

$$J_b = \sum_{k=0}^{N-1} y(t_0 + kh)^T y(t_0 + kh), \tag{10}$$

which, using (8), gives

$$J_b = \sum_{k=0}^{N-1} (C\Phi_{cl}^k(h)x_0)^T C\Phi_{cl}^k(h)x_0 \tag{11}$$

or the equivalent

$$J_b = \sum_{k=0}^{N-1} x_0^T \Phi_{cl}^k(h)^T C^T C\Phi_{cl}^k(h)x_0 \tag{12}$$

Both cost functions J_a and J_b will serve for evaluating the deviation of the plant response with respect to the equilibrium point over a finite time horizon. It turns out, however, that the expression (12) is much easier to evaluate and minimize with respect to the optimal control law than (9). Motivated by this, we will use a quadratic cost function for the feedback scheduler. If we denote $Q_1 = C^T C$, expression (12) will be in the form

$$J = \sum_{k=0}^{N-1} x(t_0 + kh)^T Q_1 x(t_0 + kh) \tag{13}$$

which is the simplest cost function used in optimal control problems [2]. In next section, we develop the cost function used in our architecture.

2.3. Theoretical Approach

As outlined in Section 1.1, the concept of feedback is used at two levels: at the control loop level and at the feedback scheduler level.

Starting at the control loop level, the first step is to establish the proper relationship between the sampling period and the finite-horizon cost function for each controller. To simplify the notation we assume that $t_0 = 0$ from now on. As noted in [3], apart from the sampling interval, there are two other factors influencing the expected performance of a control loop: the current state x_0 of the plant, and the expected future disturbances affecting the plant.

We assume that a feedback scheduler should execute in the system with a period of T_{fbs}. It is then natural to use T_{fbs} as the finite horizon in the cost function. To capture the inter-sample behavior of the control system, the cost function is formulated in continuous time as

$$J = \mathrm{E}_v \int_0^{T_{fbs}} y^2(t) dt$$

where E_v denotes expected value with regard to the noise process v. The cost function is then sampled with the interval h to give the equivalent discrete-time form

$$J = \mathrm{E}_v \Big\{ \sum_{k=0}^{N-1} \Big(x(kh)^T Q_1 x(kh) + 2x(kh)^T Q_{12} u(kh) + u(kh)^T Q_2 u(kh) + J_v(h) \Big) \Big\}, \tag{14}$$

where $N = \lceil T_{fbs}/h \rceil$ is assumed to be an integer, and J_v is an additional cost term due to the inter-sample noise. This cost function accounts for the plant response deviation, the sampled noise process (a discrete-time noise process with variance $R_1(h)$, see [2] for more details), and energy spent for the control over the feedback scheduler period. Note that the cost (14) resembles the cost (13) obtained intuitively in Section 2.2.

If optimal control were used, the optimal cost would be

$$J = x_0^T S x_0 + \sum_{k=0}^{N-1} \Big(\mathrm{tr}\, S(h) R_1(h) + J_v(h) \Big), \tag{15}$$

where S is the solution to the algebraic Riccati equation

$$S = \Phi^T S \Phi + Q_1 - (\Phi^T S \Gamma + Q_{12})(\Gamma^T S \Gamma + Q_2)^{-1} (\Gamma^T S \Phi + Q_{12}^T) \tag{16}$$

Note that Φ, Γ, Q_1, Q_{12}, Q_2, J_v, R_1, and S all depend on the sampling interval h.

It is, however, also possible to evaluate the cost function (14) for an arbitrary (i.e., non-optimal) state-feedback control law $u(kh) = -Lx(kh)$, as long as the closed-loop system is stable. In this case, the expression for S in (15) is replaced by the solution S to the Lyapunov equation (see, e.g., [10])

$$S = (\Phi - \Gamma L)^T S (\Phi - \Gamma L) + Q_1 - Q_{12} L - L^T Q_{12}^T + L^T Q_2 L \tag{17}$$

Note that, in this case, no optimization of the cost (15) has taken place. It will be the goal of the feedback scheduler to optimize the aggregate of the costs (15) for all controlled plants.

To see what terms in the cost that can be computed offline, we note that (15) can be rewritten as

$$J = x_0^T S x_0 + T_{fbs} \bar{J} \tag{18}$$

where $\bar{J} = \frac{1}{h} \big(\mathrm{tr}\, S(h) R_1(h) + J_v(h) \big)$.

3. The Period Assignment Problem

We consider n control tasks, each one described by a constant execution time C_i, an adjustable period h_i, and a cost function $J_i(x_0, h_i, T_{fbs})$ (as defined by (18)). When invoked at time t_0, the feedback scheduler is informed about the plant state vectors $x_1(t_0), \ldots, x_n(t_0)$. It should then assign new sampling periods $h_1, \ldots, h_n$ such that the total expected cost over the next T_{fbs} units of time is minimized. This is formulated as the following optimization problem:

$$\begin{aligned} &\min_{h_1 \ldots\, h_n} \sum_{i=1}^{n} J_i(x_i(t_0), h_i, T_{fbs}) \\ &\text{subj. to} \sum_{i=1}^{n} \frac{C_i}{h_i} \leq 1 \end{aligned} \tag{19}$$

To find a common analytical solution for this problem in the general case is not possible, because J_i depends on h_i,

which determines $S(h)$, which also depends on the dynamics of each plant. Moreover, the relationships are non-linear. For this reason, we will solve this optimization problem approximately, mapping the cost to be minimized to a table, which will be a function of the plants and the sampling periods. This calculus will be done *off-line*.

As the range of possible sampling periods are known off-line, i.e., $\{h_1 \dots h_n\}$, stability for each control loop under arbitrary switching sequences can be checked by looking at all possible closed-loop matrices [11].

4. Optimization Procedure

The optimization procedure starts by assuming that each task should run with its shortest period in order to achieve the minimum cost. However, due to schedulability constraints, this may not be feasible. The optimization procedure then increases the period of one task at a time, minimizing the corresponding increment in cost, until the task set becomes schedulable.

Although the basic idea for this procedure is similar to the solution to the optimization problem presented in [6], a fundamental difference exists. Our solution holds for cost functions that do not require *(a)* the existence of an analytical solution, and *(b)* the definition of the cost function as an exponential function. In other words, the solution presented in [6] can not be applied to our problem in the general case. But our solution can be applied to the problem formulated in [6].

4.1. Basic Idea

The required calculations to minimize the objective function in (19) can be divided into one off-line part and one on-line part. Given a set of sampling periods and associated controllers, the factors $S(h)$ and $\bar{J}(h) = \text{tr}\ S(h)R_1(h) + J_v(h)$ can be calculated off-line for each plant. (These factors will be generically called $S_{i,j}$, where i identifies the task and j the sampling period). Therefore, given a plant and a sampling period, the required on-line computations to evaluate the cost function in one point are essentially reduced to one vector/matrix multiplication and one vector/vector multiplication.

	h_1	h_2	...	h_{j-1}	h_j
Plant 1	$S_{1,1}$	$S_{1,2}$	...	$S_{1,(j-1)}$	$S_{1,j}$
Plant 2	$S_{2,1}$	$S_{2,2}$	...	$S_{2,(j-2)}$	$S_{2,j}$
...	...	...	...	...	...
Plant n	$S_{n,1}$	$S_{n,2}$	...	$S_{n,(j-1)}$	$S_{n,j}$

Table 1. Ordered table with $S_{i,j}$ values for each plant and sampling periods.

Taking advantage of the above facts, we construct a table with entries $S_{i,j}$, as displayed in Table 1. Here, the number of periods for all control tasks within the specified ranges depends on the difference between two consecutive periods, $H = h_{i+1} - h_i$. The lower the value of H, the higher the number of periods, and thus, the larger the table to be stored, but also the better the approximation of the optimization procedure to the exact optimal solution. Therefore, H becomes a design parameter.

Assuming that each cost function (15) is monotonically increasing in h, the table has the following characteristics: *a)* by construction, the period increases from left to right, *b)* $S_{i,j}$ increase from left to right: $S_{i,j} < S_{i,j+1}$, and *c)* since each task has a constant execution time, the CPU utilization decreases from left to right. To minimize the aggregated cost, the idea is to search the table from left to right, increasing the period of one controller at a time until the task set becomes schedulable. The exact algorithm is given next.

4.2. Algorithm

The algorithm searches for the minimum cost for all plants from left to right by at each iteration selecting the period h that gives the minimal cost increment $\Delta J(i)$.

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: **while** $\sum_{k=1}^{n} \frac{C_k}{h_k} > U^{sp}$ **do**
5: $lower\_cost\_task = 1$
6: **for** $i = 1$ to n **do**
7: **if** $h(i) + H \leq h_i^{max}$ **then**
8: $\Delta J(i) = J(h(i) + H, x(i)) - J(h(i), x(i))$
9: **if** $i \leq 2$ and $\Delta J(i) < \Delta J(lower\_cost\_task)$ **then**
10: $lower\_cost\_task = i$
11: **end if**
12: **end if**
13: **end for**
14: $h(lower\_cost\_task) = h(lower\_cost\_task) + H$
15: **end while**
16: **return** h

Input parameters are the minimum sampling period and the computation time of each task, as well as the states of all plants at the time at which the feedback scheduler is invoked. The complexity of the algorithm is $O(j \cdot n)$, where n is the number of tasks and j is the number of periods. This simplifies to $O(n)$.

4.3. Simple Example

Suppose that we should choose optimal sampling periods for two controllers of first-order plants that, for simplicity,

are not subject to noise. Therefore, in (18), $T_{fbs}\bar{J} = 0$. The look-up table will, thus, only contain the factors $S(h)$. Assume the following table, where the difference between two consecutive periods is 0.4:

h	0.1	0.5	0.9
Plant 1	0.1	0.2	0.5
Plant 2	0.2	0.4	0.9

Suppose that the initial conditions for the plants are $x_1 = x_2 = 1$, and that the execution times are $C_1 = 0.1$, $C_2 = 0.4$. The sampling period of the feedback scheduler is $T_{fbs} = 5$.

At initialization, all tasks have the shortest period: $h = \begin{pmatrix} 0.1 & 0.1 \end{pmatrix}$. For this set of periods, however, the CPU utilization is larger than 1: $U = \frac{0.1}{0.1} + \frac{0.4}{0.1} = 5 > 1$. We therefore calculate $J_i(h)$ and $J_i(h+H)$ for the first step:

	$J_i(h)$	$J_i(h+H)$	ΔJ_i
Task 1	0.1	0.2	0.1
Task 2	0.2	0.4	0.2

We then increment the sampling period for Task 1, because that will give the smallest cost. The new set of periods that we have is $h = \begin{pmatrix} 0.5 & 0.1 \end{pmatrix}$, and the new CPU utilization is: $U = \frac{0.1}{0.5} + \frac{0.4}{0.1} = 4.2 > 1$.

We thus have to do another iteration. We again calculate $J_i(h)$ and $J_i(h+H)$:

	$J_i(h)$	$J_i(h+H)$	ΔJ_i
Task 1	0.2	0.5	0.3
Task 2	0.2	0.4	0.2

Finally, increasing the period for Task 2, the CPU utilization will be $U = \frac{0.1}{0.5} + \frac{0.4}{0.5} = 1 \leq 1$ and the set of optimal sampling periods found are $h = \begin{pmatrix} 0.5 & 0.5 \end{pmatrix}$.

5. Simulation Results

We have simulated the feedback scheduling strategy in the TrueTime control and scheduling co-simulation tool [9]. EDF is used as the scheduling algorithm for all control tasks as well as for the feedback scheduler task.

5.1. Implementation Details

The optimization approach is exemplified on a set of controllers for three different second-order systems, a ball-and-beam process, a DC motor process, and a harmonic oscillator process. The plants have the following state-space representations in continuous time:

Ball and Beam:

$$\frac{dx}{dt} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u, \quad y = \begin{pmatrix} 1 & 0 \end{pmatrix} x$$

DC Motor:

$$\frac{dx}{dt} = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u, \quad y = \begin{pmatrix} 0 & 1 \end{pmatrix} x$$

Harmonic Oscillator:

$$\frac{dx}{dt} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u, \quad y = \begin{pmatrix} 1 & 0 \end{pmatrix} x$$

Controllers for the three plants were designed using the pole placement method. The desired continuous-time poles for the three systems are the following:

	poles
Ball and Beam	$-10 \pm i$
DC Motor	$-7 \pm i$
Harmonic Oscillator	$-5 \pm i$

The set of possible sampling periods has been chosen from 0.05 s to 0.5 s with increments of 0.01 s. That is, 46 different sampling periods were considered for each controller.

Figure 4 shows the TrueTime simulation setup. The TrueTime kernel simulates the execution of the four periodic real-time tasks under EDF. Three tasks correspond to the three controllers with periods as defined before, and execution times $C_1 = C_2 = C_3 = 0.05$ s, and the fourth task is the feedback scheduler, with a period of 5 s and execution time of 0.05 s. Clearly, all controllers cannot execute with the shortest period due to EDF schedulability constraints.

The TrueTime kernel outputs, through the digital-to-analog (D/A) converter, are the three control signals that are forwarded to each plant. The output (state) of each plant is fed back to the TrueTime kernel through the analog-to-digital (A/D) converter. The state is used both for each control task to calculate the next control signal, and for the feedback scheduler to reassign periods.

The input to each plant is also affected by a stationary noise process as well as by further disturbances that model for example set point changes.

Finally, the TrueTime kernel has two more outputs, the schedule and the set of assigned sampling periods. The first gives the schedule produced by the EDF algorithm, and the second gives information on the periods of each task. Figure 5 shows the evaluation of the cost function to be minimized (18) for each plant. The cost functions in the figure are evaluated for the allowed range of sampling periods and for a common initial condition, with an assumed noise variance of unity. Note that the cost functions are monotonically increasing, but with different slopes.

5.2. Validity of the Feedback Architecture

In Figure 6, we show an example of the progression of a simulation run over 20 seconds. The top sub-figure, *Sched-*

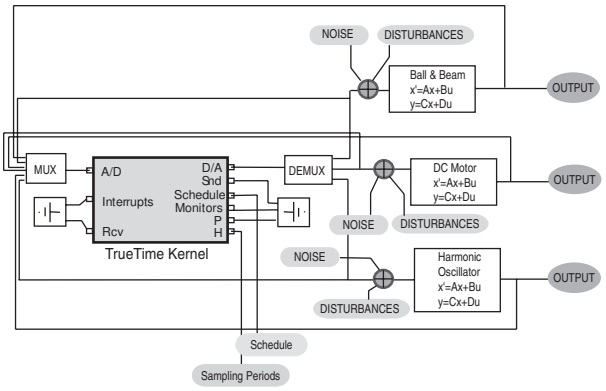

Figure 4. Simulation setup

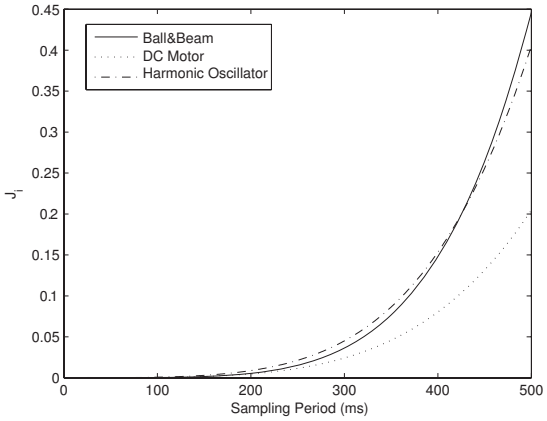

Figure 5. Cost functions

uler, shows the resulting schedule, that is, the rate of execution of each control task. Although it is difficult to appreciate all the details, it can be clearly seen that the density of job executions changes over time. The higher the density, the more jobs are executed.

The three sub-figures below the schedule are the responses of the three plants. They oscillate due to the noise process, and each arrow marks the arrival of a perturbation (e.g., a set-point change). Each perturbation increases the deviation of each response with respect to the equilibrium point.

Finally, the bottom sub-figure shows how each control task period varies over time. This sub-figure complements the schedule sub-figure, for a detailed interpretation.

Let us describe the behavior of the overall setup using Figure 6. At the simulation start, all plants are given the same initial conditions (zero) as well as similar sampling periods established by the execution of the feedback scheduler task. The next execution of the feedback scheduler will

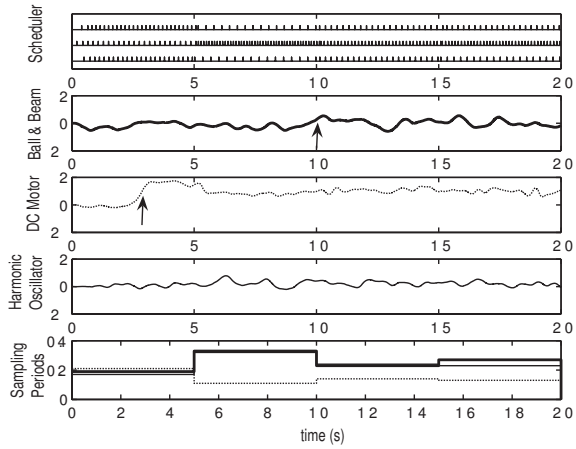

Figure 6. Task periods and plant responses

be after time 5 s.

During the first two seconds, plant responses oscillate due to the introduced noise. At time 2.5 s, a set-point change in the form of a unit step arrives to the DC motor plant. This produces a transient error that lasts long enough to produce a noticeable period reassignment at the execution of the feedback scheduler at time 5 s. At this point in time, task 2 is given a shorter period (see *Sampling Periods* sub-figure), which means a higher density of job executions (see *Scheduler* sub-figure). To balance this, the other two control tasks are given a larger period.

The period reassignment occurs at each feedback scheduler task execution. As a consequence, every 5 seconds, periods are changed according to each plant state via the optimization procedure.

In Figure 7 we exemplify the possible improvement due to the feedback scheduler. The figure shows the accumulated cost of the plant responses, which permits one to better appreciate the achieved improvement. In this figure, the lower the value, the better the performance.

Let us focus on the *without FBS* and *FBS+algorithm* lines. It can be clearly seen the the accumulated error drastically decreases when our approach is used.

5.3. Validity of the Optimization Procedure

Looking again at Figure 7, now focusing on the *FBS+algorithm* and *FBS* lines, we can evaluate the accuracy of the optimization procedure. The *FBS* line represents the evaluation of the framework when the optimization procedure is not solved using our approximation algorithm, but using an exact solution (obtained using the Matlab nonlinear optimization command *fmincon*).

Although the best behavior is obtained for the exact solution to the optimization problem, our solution is quite

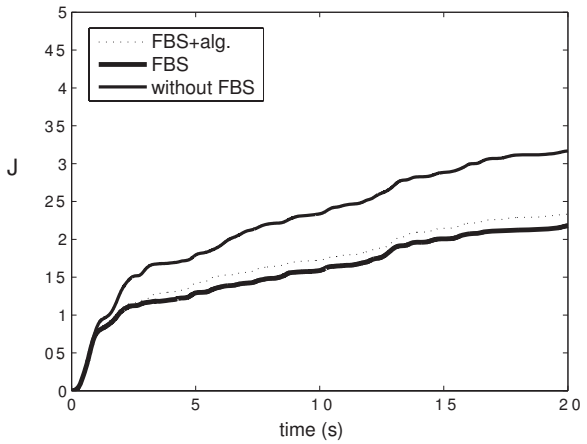

Figure 7. Evaluation

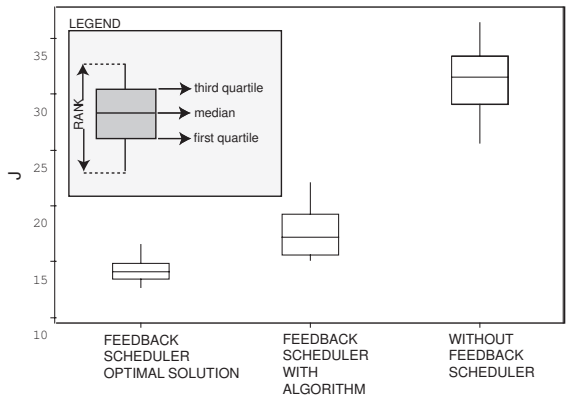

Figure 8. Statistical results

close to the optimum, and the execution time is much lower. It has to be pointed out that the difference between the *FBS+algorithm* and *FBS* curves will be smaller or larger depending on the number of sampling periods used in the construction of the table used in the optimization procedure. That is, a smaller difference will require one to have a better approximation (larger number of periods), at the expense of more overhead.

In terms of complexity, in the simulation setup, the exact solution has been obtained using a function of cost $O(n^3)$. Recall that our search is $O(n)$. In terms of simulation time on a Linux system on a standard laptop, for a run of 20 s, the simulation that uses the exact solution takes about 30 seconds while the simulation with our procedure is almost instantaneous. In terms of clock operation time, the overhead penalty for the exact solution is about 200 to 1.

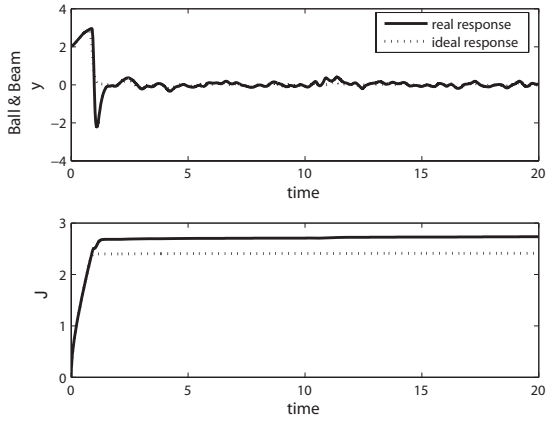

Figure 9. Closed-loop system with state observer: response and evaluation

5.4. Statistical Results

As the validity of the optimization procedure can not be based on a single simulation, we ran a set of experiments that are summarized in Figure 8. We have assigned a random disturbance pattern to the set of plants, and for each run we have changed the pattern seed. Figure 8 shows in a statistical fashion the accumulated cost of the plant responses for each simulation run. The box-plots indicate that the median of the set of values for the suboptimal algorithm compared to the optimal is not very important. However, the improvement of the suboptimal with respect to the policy without the feedback scheduler is still significant.

5.5. Evaluation Including State Observers

Our framework permits the application of arbitrary linear control laws. Looking at state feedback techniques in practical applications, one commonly encountered difficulty is that some of the state variables are not accessible for direct measurement. It then becomes necessary to reconstruct the full state vector using a state observer in order to be able to compute the feedback control signal. The feedback scheduler can then base its decisions on the estimated state vectors $\hat{x}_i$ instead of on the actual state vectors x_i.

In general, the utilization of observers implies a slight plant response degradation compared to a simple state feedback. This fact is illustrated in Figure 9. The figure shows the response and the accumulated cost of the ball-and-beam plant when a deadbeat state observer is used. The top subfigure shows the ideal response (no observer, dashed line) and the response when the observer is used (solid line). The bottom sub-figure evaluates the responses.

As it can be seen in Figure 9, the main difference occurs when there is a perturbation, such as at time 1 s. In this case, the observer takes some time to estimate the cor-

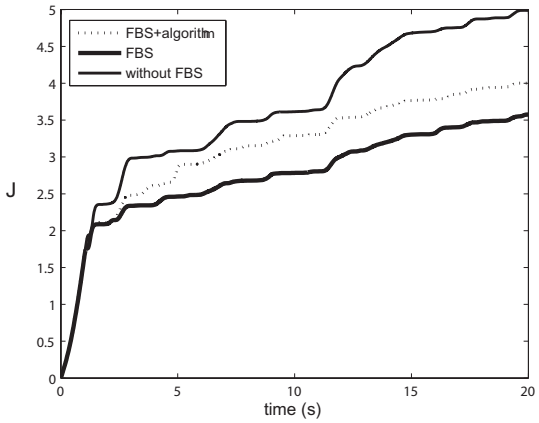

Figure 10. Evaluation with observers

rect states, and therefore, to mimic the ideal response. In addition, noise also causes mismatches between the ideal response and the response achieved using estimates, which are small compared to the ones introduced by perturbations.

Figure 10 shows the evaluation of our feedback scheduling framework for observer-based state feedback controllers. To do so, we have used the same simulation setup presented in Section 5 but using deadbeat state observers for each controller. The figure should be interpreted in the same way as Figure 7, where no observers were used. First, Figure 10 shows that the framework still provides improvement in overall control performance in spite of the introduction of observers. Second, comparing Figures 7 and 10, it can be seen that the inclusion of observers causes larger accumulated errors for all of the three evaluated strategies.

6. Conclusion

The paper has presented a practical solution to the problem of distributing limited computing resources to a set of linear control tasks. Cost functions incorporating the current states of the plants are used to capture the actual performance of each control loop in the optimization. Rather than solving the full non-linear optimization on-line, the cost functions are parameterized and evaluated only for a limited set of sampling periods. The on-line algorithm can then quickly search the table for a near-optimal solution. Simulated examples involving three different plants and controllers have been performed to show the validity of the proposed feedback scheduling framework.

Acknowledgments

This work has been supported by EU/FP6/ARTIST2, SSF/FLEXCON, and the Swedish Research Council.

References

[1] C. L. Liu and J. W. Layland. “Scheduling algorithms for multiprogramming in a hard-real-time environment.” *Journal of the ACM*, 20:1, pp. 40–61, 1973.

[2] K. J. Åström and B. Wittenmark. *Computer-Controlled Systems*, 3rd Ed. Prentice Hall, 1997.

[3] D. Henriksson, A. Cervin. “Optimal On-line Sampling Period Assignment for Real-Time Control Tasks Based on Plant State Information.” In *Proc. 44th IEEE Conference on Decision and Control and European Control Conference ECC,* 2005.

[4] P. Martí, C. Lin, S. A. Brandt, M. Velasco, J. M. Fuertes. “Optimal State Feedback Based Resource Allocation for Resource-Constrained Control Tasks.” In *Proc. 23rd IEEE Real-Time Systems Symposium,* 2004.

[5] A. Cervin, J. Eker. “Control-Scheduling Codesign of Real-Time Systems: The Control Server Approach.” *Journal of Embedded Computing* 1:2, pp. 209–224, 2005.

[6] 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,* Washington, DC, pp. 13-21, 1996.

[7] E. Bini and M. Di Natale. “Optimal Task Rate Selection in Fixed Priority Systems.” In *Proc. 26th IEEE Real-Time Systems Symposium,* Miami, FL, 2005.

[8] L. Palopoli, C. Pinello, A. Sangiovanni Vincentelli, L. Elghaoui, A. Bicchi. “Synthesis of robust control systems under resource constraints.” In *Proc. Hybrid Systems: Computation and Control,* Stanford, CA, March 2002.

[9] D. Henriksson, A. Cervin and K.-E. rzn. “TrueTime: Simulation of Control Loops Under Shared Computer Resources.” In *Proc. 15th IFAC World Congress on Automatic Control*, Barcelona, Spain, 2002.

[10] W. J. Rugh. *Linear System Theory*. Second Edition. Prentice Hall, 1996.

[11] M. Dogruel and U. Ozguner. “Stability of a set of matrices: A control theoretic approach.”. In *Proc. 34th IEEE Conference of Decision and Control,* 1995.