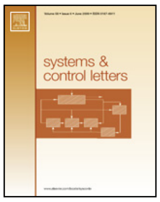
Systems & Control Letters 62 (2013) 262–268

Contents lists available at SciVerse ScienceDirect

Systems & Control Letters

journal homepage: www.elsevier.com/locate/sysconle



On oscillation free controller changes

Milton Cunguara [a,*], Tomas Silva [a], Paulo Pedreiras [b]

[a] DETI/IEETA/University of Aveiro, Aveiro, Portugal
[b] DETI/IT/University of Aveiro, Aveiro, Portugal

ARTICLE INFO

Article history:
Received 12 June 2012
Received in revised form
26 October 2012
Accepted 3 December 2012
Available online 18 January 2013

Keywords:
Adaptive control systems
Digital control
Linear control
Distributed control

ABSTRACT

Control systems are typically subject to strong, and often conflicting, constraints. One of the techniques that have been extensively investigated consists in using several controllers, designed to have different resource needs, and consequently, to exhibit different performance levels. These controllers are switched dynamically, to obtain the desired quality of control while using the lowest possible amount of resources.

However, the research made so far has been essentially focused on the rules for triggering the controller switching, neglecting the full extent that such changes have in the system. In particular, switching controllers often causes output oscillations that may negate the potential performance gains. In this paper, firstly, the cause for oscillations in the presence of period changes is investigated. Then, it is presented a solution, based on a change of basis matrix. The experimental evaluation shows that important performance gains are achieved in key control performance indicators such as overshoot, settling time and error.

1. Introduction and motivation

The extraordinary development of microprocessors made digital control the dominant technique in the design of control systems. When compared with conventional analog controllers, digital controllers are usually less expensive, present higher flexibility and adaptability, are more scalable and can perform several more functions concurrently not all necessarily related to control. This contrasts clearly with analog compensators, which are usually developed specifically for each individual system and can hardly be reused.

Designing systems according to the worst-case requirements of each process may lead to expensive and inefficient designs. Thus, to save system resources, a number of approaches were put forward in the literature, as discussed in Section 2. Regarding specifically control systems, many of these approaches rely on controller changes, taking advantage of the monotonically increasing relationship between the control performance and resource utilization. Therefore, less resource demanding, albeit less efficient, controllers are used whenever the system error is small, while more resource demanding and efficient controllers are selected when the system experiences a large error.

In the controller change methodologies hitherto proposed in the literature, controller transitions are almost always followed by an output oscillation. At best, these oscillations degrade the quality of control. However, it may easily happen that such oscillations in turn cause another controller change, which triggers itself another oscillation and so on and so forth. One example of this effect is provided in Section 4. Consequently, systems with multiple controllers tend to oscillate between two or more controllers, even when the system conditions are stable, thus degrading the quality of the control and wasting resources. This paper intends to provide system designers with the necessary means to predict and avoid such oscillations.

The remaining of this paper is organized as follows: Section 2 presents a survey of related work, focusing in co-design and period adaptation. Section 3 debates the types of controller adaptations and introduces, both formally and informally, the problems that arise when a period switch is performed. Afterwards this section presents a formal solution for performing oscillation-free controller changes. Section 4 presents the evaluation of the proposed solution. Finally, Section 5 presents the conclusions.

2. Related work

The study of switching controllers began decades ago, hence enjoying a vast body of literature. However, the study of oscillation free transitions remained a fringe topic between the control and scheduling communities. This fact rendered its associated literature not so vast. This section provides an overview of the scientific contributions that are close and relevant to the work presented in this paper.

In [1] it was proposed the use of several controllers, each tuned to a given period. The scheduler would then choose a controller

* Corresponding author.
E-mail addresses: milton.cunguara@ua.pt (M. Cunguara), tos@ua.pt (T. Silva), pbrp@ua.pt (P. Pedreiras).

0167-6911/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysconle.2012.12.001

of a given period according to the error. This idea was somehow complemented in [2], with a study of optimal transition error levels, though the former study was centered around distributed systems whereas the latter was focused on centrally scheduled systems. Further work in this area includes the investigation of when a number of pre-calculated controllers outperform the dynamical chosen ones, in terms of computational load, allowed periods, response time and related metrics.

Several studies, for example [3] and more prominently [4], discuss whether the controller changes should be based on instantaneous or on filtered/accumulated measures of error. [3] favors instantaneous measures, based in a number of experiments that show that the advantages of filtering do not compensate the additional computational burden. [4] presents similar experiments, although with different systems, and concludes that filtering was worthwhile. This apparent contradiction certainly deserves a deeper study, which however, is out of the scope of this paper.

In [5] and more recently in [6] the authors present a rather different approach. Co-design is (re)introduced with a new set of goals and paradigms. Each controlled process has a given weight and the goal of the scheduler is to assign the periods in a way that minimizes the sum of the weights times the squared errors. More formally, let $f(\tau_j) = \sum_{k \in P} w_k e_k^2$ be the total error, given that it was chosen the schedule τ_j, and the kth process has error e_k and weight w_k. Then the chosen schedule is:

$$\tau = \arg\min \left(f(\tau_j) \right). \tag{1}$$

[7] proposes minimizing the quality of control. Nevertheless, not only the mechanism is somewhat similar to its predecessors, as it may also present itself more difficult to minimize.

Using the CAN protocol, [4] discusses the advantages of period switching at the network level, thus approaching a co-design perspective. In [8] and references therein, a discussion regarding feedback scheduling is presented. Feedback scheduling can be loosely defined as a scheduling methodology in which the task's current characteristics (e.g. period, worst-case execution time) are defined based on current operating conditions.

The problem of scheduling such processes has also made significant advances, one of the most relevant ones being the elastic task model [9]. Drawing from an analogy of strained springs, in which the sum of the displacements is equal to the total displacement and the displacement of each string is inversely proportional to its elastic coefficient, in this method the task's utilization may be adjusted to have a given utilization level. Each task has an individual weight that controls its relative level of adaptivity. This model is general enough to be ported to the control scenario without major modifications. Nonetheless, it was further generalized in [10] for encompassing other types of minimization. It must be pointed out that in all such studies it is assumed that there is a mechanism for correctly/optimally choose the task's weights. However, to the best of our knowledge, there are no studies addressing this issue.

In [11] non-periodic controllers are presented. The controllers are executed and then, based on the current state and error, the next activation instant is selected. However, this method has a few drawbacks and limitations. One of the most important ones is that it implies a centralized architecture, since the controller must have a new sample at a moment that, in a distributed system, is not available. Additionally, even though a number of simulations were presented, it is not clear that there is no periodic controller that achieves the same quality of control. Furthermore, its erratic activation pattern, which was further discussed in [12], renders its associated schedulers far more complex.

3. Controller adaptation through period switch

Controller changes are performed whenever the current controller is not the most suitable one to deal with the situation at hand. Different controller characteristics can be modified. To systematize the presentation, the type of controller adaptations are categorized into three non-simultaneously exclusive classes: period change, order change and complexity (type) change. Any controller change should fit in at least one of these classes.

A *period change* is the most basic type of change and possibly the most common, as verified by the authors in the literature survey and references presented therein. This is probably because this type of change leads to systems in which the schedulability is relatively easy to analyze using common schedulability tests. In this type of controller change, there is a change in the sample and actuation period of the controller, accompanied by a change in the controller's parameters. However, usually nothing is done regarding the system state at the commutation instants which, as will be seen later may cause oscillations. A typical example is the change from a Proportional-Integral-Derivative (PID) controller at a relatively high sample rate, during instants of high system dynamics, to a relatively low sample rate during periods of low system dynamics.

An *order change* is typical of systems with singularities in regions of the s-plane (z-plane) that at certain times have a low or negligible effect in the output. This situation happens, for example, in a system with a poorly canceled zero-pole pair. Another example is an industrial machinery with two functional modes, one of them including a high frequency, low response time output. In principle, when such machinery is in a *slower* mode it could use a lower order model that does not take into account the higher frequency singularities.

A *complexity change* in the controller is also possible. An example would be the changing between a PID and a state space controller with an observer and a full-state feedback controller. This is by far the option with fewer references in the literature.

It is important to stress that these categories are not disjoint. A single controller change may belong to more than one category. For example, a change from a PID at 17 ms to a state-space (Kalman filter plus full-state feedback) at 11 ms is both a complexity and a period change.

As mentioned above, period change is arguably the controller change class most commonly reported in the literature, thus this article will focus in this approach.

3.1. Period switch

During its normal operation, PID or state-space controllers save a number of past inputs or a linear combination of them. Upon a controller change the output will almost certainly oscillate because after that change the state still *points* to the output in the instants induced by the previous period, as opposed to the instants induced by the current period.

To illustrate the problem, consider a controller with a period T_i and N state variables. At a given instant t there is a period change and the new period is T_j. In the former case the state includes estimates of $[y(t)\ y(t - T_i) \ldots y(t - (N - 1)T_i)]$. However, when the controller transition occurs, the state does not automatically change to point to $[y(t)\ y(t - T_i) \ldots y(t - (N - 1)T_i)]$ (see Fig. 1). Since the controller is now tuned for the latter state, it will produce sub-optimal control values until all old state values are replaced by new ones, spaced apart by T_j. It is elucidative to note that a controller that saves the values of the last N positions will oscillate whenever the controller changes, whereas a controller that saves both the positions and its $N - 1$ successive derivatives does not

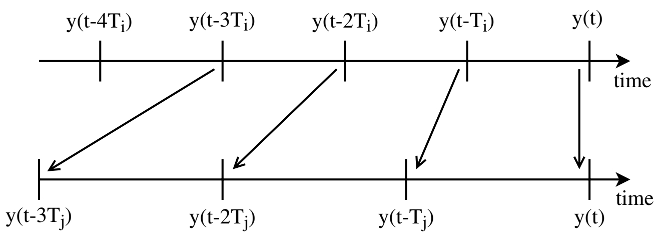


Fig. 1. State in classical controller change.

Table 1
Notation.

Γ	Feedback controller
k	Discrete-time variable
t	Continuous-time variable
T	Period of a function
r	Reference signal
u	Input variable
w	Input noise
v	Sampling noise
x	State space variable
y	Output variable
$\boldsymbol{A}$	State transition matrix
$\boldsymbol{B}$	Input matrix
$\boldsymbol{C}$	Output matrix
n	System order
$\boldsymbol{I}_n$	Identity matrix of size n
λ	Set of eigen values of a matrix
m_i	Multiplicity of the ith eigen value of a matrix
$\boldsymbol{P}$	Change of basis matrix
[vec]	Matrix vectorization operator

oscillate upon a controller change, since at the switching instant the state already points to the right place.

To overcome this problem at the switching instants the state must be updated, in order to point to the corresponding values in the new time base. One possible way to carry out this conversion is to move forward/backward in time, using the system's continuous time equations. Obviously, some previous input values would have to be stored in order to allow such computations. Despite conceptually simple, this approach implies a relatively high processing overhead. This is a serious problem, since most digital control systems are normally subject to stringent resource constraints. This observation led to the development of the methodology presented in Section 3.3, which requires much simpler computations at runtime.

3.2. Problem formulation

This paper addresses feedback control systems, possibly distributed, as depicted in Fig. 2. It is assumed that the control system incorporates more than one controller designed to minimize an arbitrary cost function (e.g. energy, CPU utilization, network utilization), and that it is used a state-space representation. Each controller Γ_i has an associated sampling rate (T_i), to which corresponds a state-space representation SYS_i. Table 1 presents the notation used in the paper.

Let $\Gamma_i \equiv \{\{SYS_i, T_i\}, i \in \{1, 2, \dots n\}\}$ be a set of feedback controllers for SYS. It is assumed that all controllers Γ_i are properly designed and can stabilize system SYS. Furthermore, any controller Γ_i may be used during the system lifetime, although at any time instant only one controller is active. The problem of selecting the best controller, selecting the optimum switching instants or design the controllers are out of the scope of this paper. Methods described in the literature (e.g. [1,2]) can be used to this end.

Without loss of generality, consider that at an arbitrary time instant t it is requested a change of controller from Γ_i to Γ_j. The associated state-space representations, SYS_i and SYS_j, are respectively,

$$x_i(k+1) = \boldsymbol{A}_i x_i(k) + \boldsymbol{B}_i u(k) \quad (2a)$$

$$y(k) = \boldsymbol{C}_i x_i(k) \quad (2b)$$

and

$$x_j(k+1) = \boldsymbol{A}_j x_j(k) + \boldsymbol{B}_j u(k) \quad (3a)$$

$$y(k) = \boldsymbol{C}_j x_j(k) \quad (3b)$$

The problem thus consists in devising a mechanism to switch from SYS_i to SYS_j without oscillation, i.e. finding the value x_j in the space of SYS_j that corresponds to the state x_i in SYS_i at any arbitrary time instant.

As an additional requirement the online part of such mechanism must be lightweight, both in terms of processing and memory, since the state conversions must be performed during runtime upon controller changes in resource-constrained hardware.

3.3. Proposed solution

The state conversion between different time bases requires a few steps. Firstly, an auxiliary representation is defined. The application of the similarity transformation of such auxiliary representation, first on the transition matrix and then on the input/output matrix, leads to a Sylvester and to a simpler linear equation respectively. The resulting Sylvester equation is homogeneous, thus classic solving methods are inadequate, leading to the use of the Kronecker product. From this process it is extracted an ensemble of matrices that verify the transition matrix condition. Furthermore, any non-singular linear combination of such matrices is also a solution. Finally, it is searched for a linear combination of the matrix ensemble that also verifies the input/output matrix similarity equation. The remaining of this section presents, in detail, each one of these steps.

To carry out the state conversion, lets start by introducing an auxiliary representation SYS_{ja}, which is sampled at rate T_j but its continuous time version has a state equal to the state of SYS_i, i.e let SYS_{ja} be

$$x_{ja}(k+1) = \boldsymbol{A}_{ja} x_{ja}(k) + \boldsymbol{B}_{ja} u(k) \quad (4a)$$

$$y(k) = \boldsymbol{C}_i x_{ja}(k). \quad (4b)$$

The introduction of SYS_{ja} turns the original problem into a simpler one, i.e. a change of basis problem. Since, by design, $x_i = x_{ja}$ and both SYS_{ja} and SYS_j are sampled at the same rate, there should be a matrix $\boldsymbol{P}$ such that $x_j = \boldsymbol{P} x_{ja}$. The existence and uniqueness of such matrix is a standard result [13].

To find $\boldsymbol{P}$, it can be used the Sylvester equation[1]: $\boldsymbol{A_{ja}} = \boldsymbol{P}^{-1}\boldsymbol{A}_j\boldsymbol{P}$, or

$$\boldsymbol{P}\boldsymbol{A}_{ja} - \boldsymbol{A}_j\boldsymbol{P} = 0. \quad (5)$$

Classical efficient approaches, e.g. [14,15], fail to give satisfactory solutions to this particular problem, since they always provide the trivial solution ($\boldsymbol{P} = 0$). It is noteworthy that this equation also appears in robust pole placement techniques, as attested by [16] and references therein.[2] However, the mechanisms used to solve it in that context are not easily portable to this problem. Therefore, we opted by using the Kronecker product approach. In this approach, Eq. (5) is transformed into:

$$\left(\boldsymbol{A}_{ja}^T \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{A}_j\right) vec(\boldsymbol{P}) = 0 \quad (6)$$

[1] $\boldsymbol{AX} + \boldsymbol{XB} = \boldsymbol{C}$ in which $\boldsymbol{A} = -\boldsymbol{A_j}$, $\boldsymbol{B} = \boldsymbol{A_{ja}}$ and $\boldsymbol{C} = \boldsymbol{0}$.
[2] This mechanism is used in Matlab® function place.

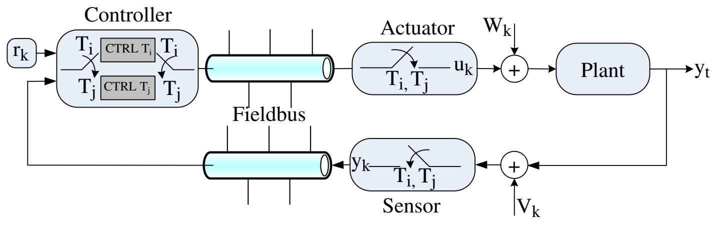


Fig. 2. Control architecture.

where $vec(\boldsymbol{P})$ is a vectorized version of the matrix $\boldsymbol{P}$ and $\boldsymbol{I}_n$ is the $(n \times n)$ identity matrix.

Eq. (6) is an eigen problem, associated with the eigenvalue 0. The solution of such eigen problem is a series of vectors that when passed back into the initial matrix space (the inverse of vec), gives rise to a series of eigen matrices to which any non-singular linear combination is also a solution of Eq. (5). This stems from a known result of linear algebra, which states that any linear combination of eigenvectors associated with the same eigenvalue is also an eigenvector associated with the eigenvalue in question. This principle is combined to the fact that any eigenvector associated with the eigenvalue 0 is a solution of Eq. (5).

Nevertheless, it is noteworthy that this problem may have more than one solution. It is well known that the eigenvalues of $(\boldsymbol{A} \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{B})$ (with $\boldsymbol{A}$, $\boldsymbol{B}$ square matrices of size n) are $\lambda_i(\boldsymbol{A}) - \lambda_j(\boldsymbol{B})$, where $\lambda_i(\boldsymbol{X})$ is the ith eigenvalue of matrix $\boldsymbol{X}$. In this particular case $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same eigenvalues, hence each eigenvalue of $\boldsymbol{A}$ produces m_i^2 zero eigenvalues in the matrix $\boldsymbol{P}$, where m_i is the multiplicity of the ith eigenvalue. Thus, $\boldsymbol{P}$ has an eigenvalue of 0 with multiplicity $\sum_{i=1}^{q} m_i^2 \geq n$, where q is the number of distinct eigenvalues of $\boldsymbol{A}$. Therefore, the extra solutions appear whenever the transition matrix ($\boldsymbol{A}_{ja}$, $\boldsymbol{A}_j$ or $\boldsymbol{A}_j$ depending on the representation) has repeated eigenvalues.

The eigen matrices presented above relate to the spectral decomposition of the matrix in question in the sense that they expand the matrix of eigenvectors (upon proper multiplication) in a sum-like fashion. Thus, in this framework, the extra eigen matrices are induced by the rank of the eigen space spawned by eigenvalues with high multiplicity.

Due to the similarity transformation, matrix P must verify

$$\boldsymbol{P}\boldsymbol{B}_{ja} = \boldsymbol{B}_j. \tag{7}$$

Eqs. (5) (or (6)) and (7) define matrix P. More precisely, the first equation defines a rank three (3-dimensional) tensor, that is conveniently written as a number of eigen matrices and the product of such matrix by matrix B_{ja} can be written as:

$$\boldsymbol{P}\boldsymbol{B}_{ja} = \left(\sum_i \omega_i \boldsymbol{P}_i\right) \boldsymbol{B}_{ja} \tag{8}$$

where $\boldsymbol{P}_i$ are the eigen matrices and ω_i are unknown coefficients. Define now, $s_i \equiv \text{vec}(\boldsymbol{P}_i\boldsymbol{B}_{ja})$. Then (7) and (8) can be rewritten as

$$vec(\boldsymbol{B}_j) = [s_1\, s_2 \ldots s_z] \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_z \end{bmatrix} \tag{9}$$

or simply, $\text{vec}(\boldsymbol{B}_j) = \boldsymbol{S}\Omega$, were $\boldsymbol{S}$ and Ω are implicitly defined. The previous equation provides the values of ω that give the actual solution, i.e. solution of both Eqs. (5) and (7). Furthermore, $\boldsymbol{P} = \sum_i \omega_i \boldsymbol{P}_i$.

Algorithm 1 provides a summarized description of the procedure that determines the change of basis matrix. The operation *inv_vec* used in the algorithm is the inverse of *vec* operator that transforms matrices into row vectors. $\boldsymbol{S}^{-1}$ may not exist, as discussed above. Since this happens only if the system of equations that define $\boldsymbol{P}$ is over-determined, a proper pseudo-inverse may be used instead. The case of multiple inputs (or outputs if matrix $\boldsymbol{C}$ is used) is handled seamlessly by the algorithm, if in fact the pair $(\{\boldsymbol{A}_{ja}, \boldsymbol{B}_{ja}\}, \{\boldsymbol{A}_j, \boldsymbol{B}_j\})$ are different representations of the same system (at the same rate), in which case it is guaranteed that $vec(\boldsymbol{B}_j)$ is on the space spanned by $\boldsymbol{S}$, nonetheless this situation must also be dealt with using the pseudo-inverse of $\boldsymbol{S}$.

It follows that if P switches from SYS_i to SYS_j, then $\boldsymbol{P}^{-1}$ switches from SYS_j to SYS_i.

$$x_j = \boldsymbol{P}x_i \tag{10a}$$

$$x_i = \boldsymbol{P}^{-1}x_j. \tag{10b}$$

It is important to stress that a vector of ones may be an eigenvector of $\boldsymbol{P}$ associated to the eigenvalue 1. For example, the steady-state of a variable-phase representation is in the space spanned by such vector. Whenever this happens there is no need to perform these operations since $\boldsymbol{P}\text{x} = \text{x}$. This can be generalized to other situations, which is the prime motive why the systems considered in, for example, [17] did not oscillate, even though no compensation was made.

3.4. Complexity of the methodology

The methodology proposed in this paper comprises two complementary steps. Firstly, once the controllers are designed, $\boldsymbol{P}$ is computed using Algorithm 1. This step is carried out off-line, during system design, in non-constrained hardware.

Algorithm 1 complexity is dominated by the complexity of the SVD algorithm, which is used to compute the null space of $(\boldsymbol{A}_{ja}^T \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{A}_j)$. This step can be done using the eigen decomposition instead. However, due to its inherent numerical instability and to the fact that these computations are carried out off-line, this option was not adopted. SVD has a complexity $\bigcirc(m^3)$, where m is the number of rows (columns) of the matrix, which in this case is square. Thus, as the system order is n, the overall complexity becomes $\bigcirc(n^6)$. The situation can be ameliorated e.g. with the use of QR-decomposition (with column pivoting due to the singularity of the matrix). The algorithm's execution time was measured in an Acer Desktop PC, featuring an Intel Core 2 Quad Q6600 CPU at 2.40 GHz. For all systems reported in Section 4, the execution time was found to be below the clock resolution (2^{-8} s).

Algorithm 1 Algorithm for computing the change of basis matrix

```
$K \leftarrow \left(\boldsymbol{A}_{ja}^T \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{A}_j\right)$
$\boldsymbol{ns} \leftarrow \text{nullspace}(K)$
**for all** Columns of $\boldsymbol{ns}$ **do**
    $\boldsymbol{S}_i \leftarrow vec\left(inv\_vec(\boldsymbol{ns}_i)\boldsymbol{B}_{ja}\right)$
**end for**
$\Omega \leftarrow \boldsymbol{S}^{-1} vec(\boldsymbol{B}_j)$
$\boldsymbol{P} \leftarrow \boldsymbol{ns} \times \Omega$
```

The actual change of basis, triggered by controller changes, is performed online using Eqs. (10a) and (10b). This operation implies only a simple matrix product involving a square matrix and a column vector of corresponding size. Hence, the change of basis computation is lightweight, both in terms of processing power and memory, being implementable even in resource-constrained hardware.

3.5. Orthogonality with respect to control and scheduling

In both control systems and real-time communities, there are many different methodologies addressing a myriad of specific problems. Frequently, such methodologies are incompatible with others, thus restricting its practical applicability.

The controller adaptation methodology proposed in this article is agnostic with respect to control, scheduling and system architecture aspects. For example, moving from a centralized to a distributed architecture poses no complications as far as the controller correctly receives sensor data and is allowed to send actuation data—the computations remain exactly the same. Regarding the scheduling discipline, the only requirement is that one additional task, which is the adaptation algorithm, must be executed within one period upon a controller change. No assumptions are made about the scheduling policy, priority scheme or any other scheduling-related aspects. Finally, the controller-change method also does not pose any constraints on the nature or dynamics of the system, being applicable to any system for which a set of stable controllers can be designed.

4. Evaluation of controller adaptation through period switch

This section presents the evaluation results obtained from three different systems. The systems are represented using the *variable phase* (the state transition matrix is a *companion* matrix and the input matrix has one single 1 and $n-1$ zeros). This choice was made due to its popularity in control practice, which results from the fact that it can be readily obtained from the physical characteristics of the systems.

The simulations were made using a square wave as the reference signal. This wave changed between ± 1 with a duty cycle of 50% and a period of 4 s. The controllers used the standard regulators theory to drive the system according to the reference signal. Pole-placement was used to place the poles regularly spaced in the interval [0.85 0.9].

Unless stated otherwise, all simulations last for 10 s. The controller is changed periodically with a period of 600 ms. The sub-parts of the control system (i.e. sensor, controller, actuator) communicate through a network, as depicted in Fig. 2, with a delay of 2 ms and a jitter with a Poisson distribution with mean arrival time also of 2 ms. These values of delays and jitter were chosen to reflect values usually found in fieldbuses without isochronous transfer support.

The proposed method was tested in two systems. The first system was sampled at 10 and 15 ms. At 10 ms it has the following state space representation:

$$x(k+1) = \begin{bmatrix} 0 & 1 \\ -0.82 & 1.8 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k).$$

At 15 ms the state space equations are:

$$x(k+1) = \begin{bmatrix} 0 & 1 \\ -0.6109 & 1.5694 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$

$$y = \begin{bmatrix} 0.3009 & 1.7783 \end{bmatrix} x(k).$$

Table 2
Metrics comparison.

	SYS1 prop	SYS1 classic	SYS2 prop	SYS2 classic
Rise time (ms)	247	256	525	585
Overshoot (ms)	0	13.63	0	20.9
Settling time (5%)	439 ms	Undefined	887 ms	Undefined
ISE	1584	1633	4051	4448
Corrected *ISE*	1.74×10^{-6}	14.19	2.93×10^{-6}	3.59

Fig. 3(a) depicts the system behavior when the oscillation control technique proposed in this paper is used, while Fig. 3(b) depicts the results obtained in identical conditions, with the exception that the oscillation control technique is absent. By simple observation it can be concluded that the proposed approach behaves as expected, exhibiting no oscillations, whereas in the classical approach controller commutations result in output oscillations.

The second system commutes between sampling periods 16 and 20 ms, having the following representations:

$$x(k+1) = \begin{bmatrix} 0 & 1.0000 & 0 \\ 0 & 0 & 1.0000 \\ 0.3150 & -1.4300 & 2.1000 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k)$$

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} x(k)$$

and

$$x(k+1) = \begin{bmatrix} 0 & 1.0000 & 0 \\ 0 & 0 & 1.0000 \\ 0.2360 & -1.1990 & 1.9373 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k)$$

$$y = \begin{bmatrix} -0.0172 & 0.3045 & 1.4277 \end{bmatrix} x(k).$$

Fig. 4 shows the simulation results of this system. Once again, the classical approach caused the system to oscillate, whereas with the approach proposed in this paper the output does not oscillate.

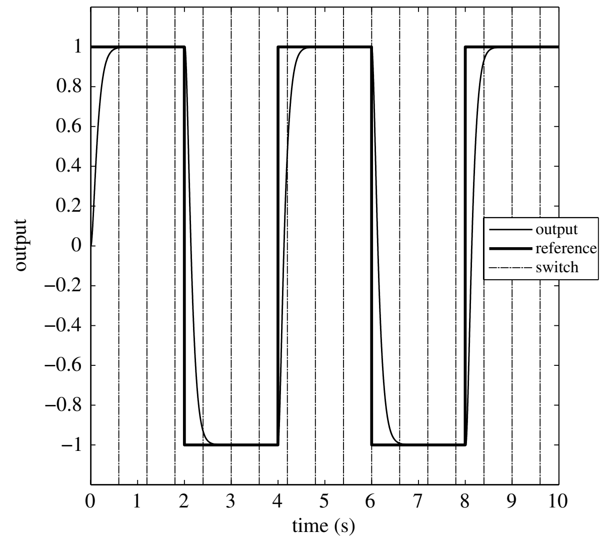
A series of measurements were made to quantify the improvements shown in Figs. 3 and 4. They are summarized in Table 2. The first row of this table shows the rise-time of the different systems. The proposed and the classical methods have rather similar values because during the rise time window there weren't any controller transitions.

The second row of Table 2 shows the overshoot values. The proposed mechanism does not exhibit overshoot because it does not oscillate despite the controller changes, a property that classical approaches lack. A similar situation is seen in the settling time, since classical approaches under controller change never reach a state that can be considered steady (5% error band).

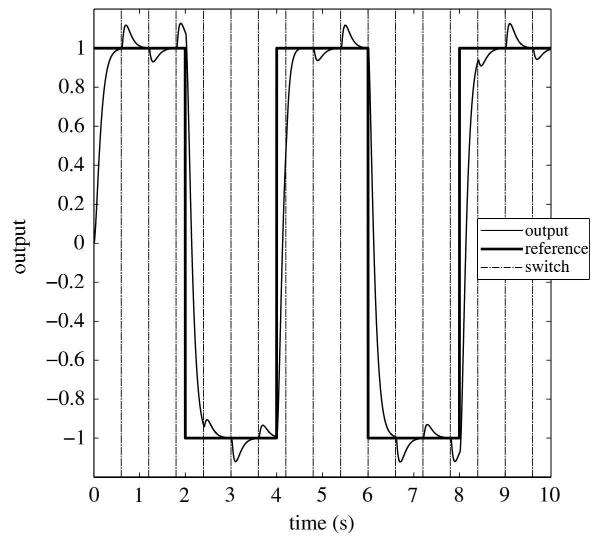
$$ISE = \int_0^T (y(t) - r(t))^2 \, dt. \tag{11}$$

The Integral Squared Error (*ISE*) presented in the last but one row of Table 2 is computed according to Eq. (11), in which T is the duration of the experiment, $y(t)$ is the output signal and $r(t)$ is the reference signal. The improvement in the *ISE* does not seem to be substantive, mostly due to the fact that the controllers are changed with a relatively low frequency. Hence, the *ISE* is dominated by the time that the systems spend tracking the input signal. Note that the impact of controller-change induced oscillations is higher on the slower system, since the associated recovery time is higher. This effect appears distinguishably when comparing the *ISE* of the first (faster dynamics) and second (slower dynamics) systems. The faster system experiences an *ISE* reduction of 3.1%, while the slower system has an *ISE* reduction of 9.8%.

To highlight the controller changes' impact on the *ISE*, it was also computed the *corrected ISE*, shown in the last row of Table 2.

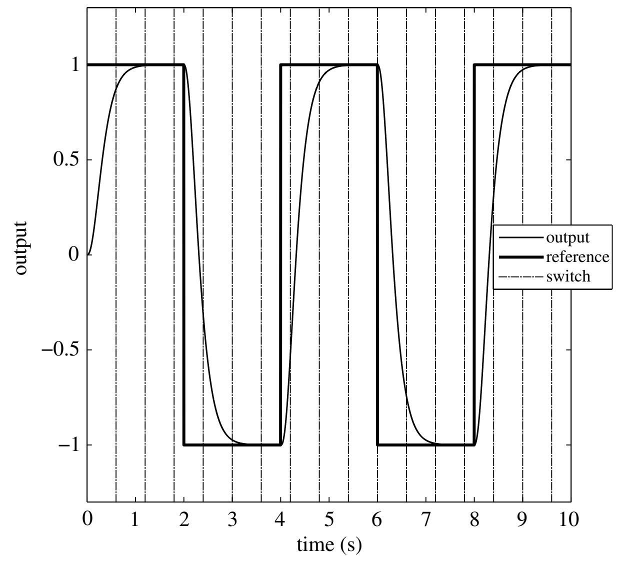


(a) Proposed.

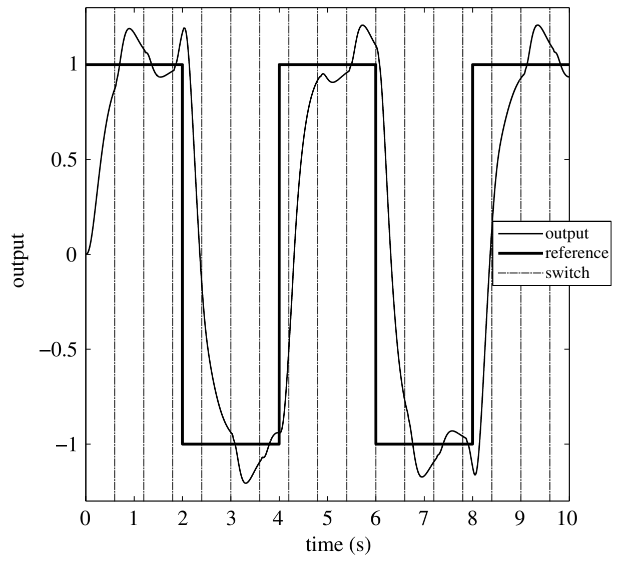


(b) Classical.

Fig. 3. Response of first system to a square-wave.



(a) Proposed.



(b) Classical.

Fig. 4. Response of second system to a square-wave.

This modified *ISE* value is computed only for points in which both the reference and output signals are stable, hence removing the impact of the reference signal changes, and thus the tracking error. With this metric the performance difference becomes much more evident. The corrected *ISE* is essentially null for both systems, when the oscillation control technique is applied. Note that the absolute value of the corrected *ISE* is lower for system two because its output is stable during a shorter amount of time, thereby reducing the time interval during which the corrected *ISE* is computed.

The final simulation presents an example of a system in which a controller change causes an oscillation, that in turn causes another controller change and so on and so forth. The system changes between a sample rate of 10 and 15 ms, according to the output error. There are two threshold values for the estimate of the output error, to induce hysteresis. The higher threshold is set to 0.80 and the lower threshold is set to 0.10. The system remains at the higher rate as long as the lower threshold is not crossed. Similarly, it remains at the lower sampling rate as long as the higher threshold is not crossed. At the highest sampler rate, i.e. 10 ms, the system is represented by:

$$x(k+1) = \begin{bmatrix} 0 & 1 \\ -0.6400 & 1.5217 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k).$$

At the sampling rate of 15 ms it has the representation:

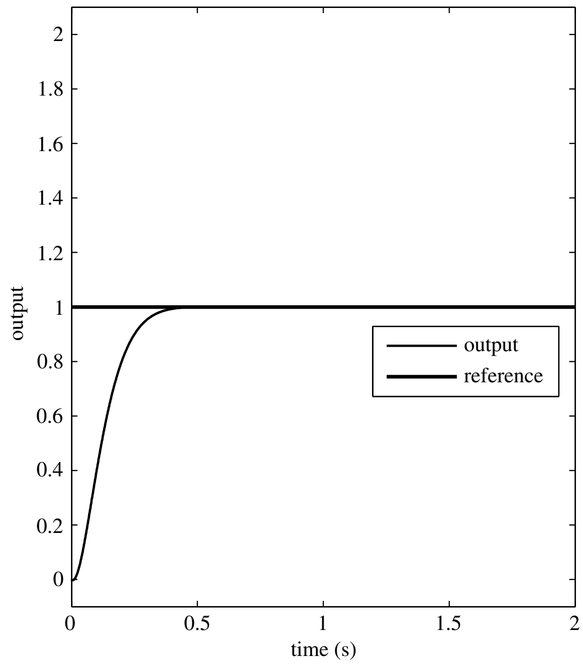
$$x(k+1) = \begin{bmatrix} 0 & 1 \\ -0.5120 & 1.2751 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$

$$y = \begin{bmatrix} 1.5950 & 0.4073 \end{bmatrix} x(k).$$

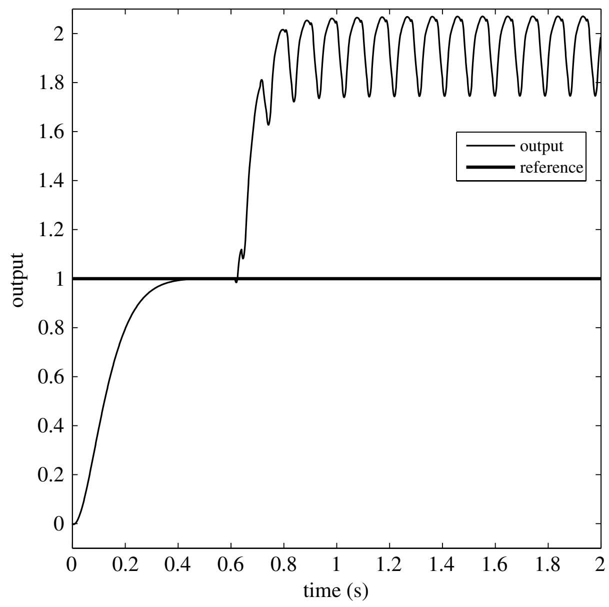
Fig. 5 depicts the behavior of such system. The *continuous* controller commutation pattern is evident. It is also evident that the approach presented in this paper completely avoids it.

5. Conclusion

Switching dynamically among different controllers, in order to always have the active controller that uses the lowest possible



(a) Proposed.



(b) Classical.

Fig. 5. Example of system locked in oscillations.

amount of resources, while meeting a minimum quality of control, has been a field of intense research. However, in existing approaches controller changes are often followed by output oscillations, which may degrade the quality of control and waste resources.

In this paper, we investigated the origin of such output oscillations, which are due to the state variables not being in agreement before and after the controller commutations. To solve this problem, a novel mechanism which finds a change of basis matrix that turns a state variable under a given representation into another one was developed. During runtime this mechanism only requires a simple matrix product thus being implementable in resource-constrained hardware, typically found in control systems applications.

Simulations of diverse distributed systems, subject to network contention, were carried out. The obtained results are in perfect accordance with the expectations. The same systems subject to the same stimulus experienced oscillations when the classical approaches were employed, while such oscillations did not occur when the adaptation method presented in this paper was used. Quantitatively, the absence of such oscillations results in the nullification of the overshoot and *ISE* due to controller changes.

References

[1] A. Antunes, P. Pedreiras, A. Mota, Adapting the sampling period of a real-time adaptive distributed controller to the bus load, in: 10th IEEE Conference on Emerging Technologies and Factory Automation, ETFA, vol. 1, 2005, pp. 1084–1087 http://dx.doi.org/10.1109/ETFA.2005.1612648.

[2] A. Cervin, M. Velasco, P. Martí, A. Camacho, Optimal online sampling period assignment: theory and experiments, in: IEEE Transactions on Control Systems Technology, vol. 19, 2010, pp. 1–9.

[3] R. Castane, P. Marti, M. Velasco, A. Cervin, 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, 2006, pp. 171–182, ISBN 0-7695-2619-5, http://dx.doi.org/10.1109/ECRTS.2006.24.

[4] A. Anta, P. Tabuada, On the benefits of relaxing the periodicity assumption for networked control systems over CAN, in: Proceedings of the 2009 30th IEEE Real-Time Systems Symposium, in: RTSS 09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 3–12. http://dx.doi.org/10.1109/RTSS.2009.39.

[5] A. Cervin, J. Eker, Feedback scheduling of control tasks, Decision and Control, 2000. Proceedings of the 39th IEEE Conference on, vol. 5, 2000, pp. 4871–4876, ISSN 0191-2216, http://dx.doi.org/10.1109/CDC.2001.914702.

[6] A. Cervin, J. Eker, B. Bernhardsson, K.-E. Årzén, Feedback feedforward scheduling of control tasks, Real-Time Syst. (ISSN: 0922-6443) 23 (2002) 25–53. http://dx.doi.org/10.1023/A:1015394302429.

[7] G. Buttazzo, M. Velasco, P. Marti, Quality-of-Control Management in Overloaded Real-Time Systems, IEEE Trans. Comput. (ISSN: 0018-9340) 56 (2007) 253–266. http://dx.doi.org/10.1109/TC.2007.34.

[8] C. Lozoya, P. Martí, M. Velasco, J.M. Fuertes, control performance evaluation of selected methods of feedback scheduling of real-time control tasks, in: 17th IFAC World Congress, 2008.

[9] G. Buttazzo, G. Lipari, M. Caccamo, L. Abeni, Elastic scheduling for flexible workload management, IEEE Trans. Comput. (ISSN: 0018-9340) 51 (3) (2002) 289–302. http://dx.doi.org/10.1109/12.990127.

[10] T. Chantem, X.S. Hu, M. Lemmon, Generalized elastic scheduling, in: Real-Time Systems Symposium, 2006. RTSS '06. 27th IEEE International, 2006, pp. 236–245, ISSN 1052-8725, http://dx.doi.org/10.1109/RTSS.2006.24.

[11] M. Velasco, P. Martí, J.M. Fuertes, C. Lozoya, S.A. Brandt, Experimental evaluation of slack management in real-time control systems: Coordinated vs. self-triggered approach, Syst. Archit. (ISSN: 1383-7621) 56 (2010) 63–74. http://dx.doi.org/10.1016/j.sysarc.2009.11.005.

[12] P. Martí, M. Velasco, E. Bini, The optimal boundary and regulator design problem for event-driven controllers, in: Proceedings of the 12th International Conference on Hybrid Systems: Computation and Control, HSCC '09, 2009, pp. 441–444, ISBN 978-3-642-00601-2, http://dx.doi.org/10.1007/978-3-642-00602-9_31.

[13] A.P. de Melo, Teoria dos Sistemas de Controlo Lineares, second ed., Universidade de Aveiro, 2010.

[14] R.H. Bartels, G.W. Stewart, Solution of the matrix equation $AX + XB = C[F4]$, Commun. ACM 15 (9) (1972) 820–826.

[15] G. Golub, S. Nash, C. Van Loan, A Hessenberg–Schur method for the problem $AX + XB = C$, IEEE Trans. Autom. Control (ISSN: 0018-9286) 24 (6) (1979) 909–913. http://dx.doi.org/10.1109/TAC.1979.1102170.

[16] A. Varga, Robust pole assignment techniques via state feedback, in: Decision and Control, 2000. Proceedings of the 39th IEEE Conference on, vol. 5, 2000, pp. 4655–4660, ISSN 0191-2216, http://dx.doi.org/10.1109/CDC.2001.914662.

[17] M. Velasco, P. Marti, E. Bini, Control-driven tasks: modeling and analysis, in: Real-Time Systems Symposium, 2008, 2008, pp. 280–290, ISSN 1052-8725, http://dx.doi.org/10.1109/RTSS.2008.29.