Anytime Control Algorithms for Embedded Real-Time Systems

Daniele Fontanelli* and Luca Greco*† and Antonio Bicchi* ⋆
daniele.fontanelli,bicchi@ing.unipi.it,greco@dsea.unipi.it

1 *Interdepartmental Research Center "E. Piaggio", University of Pisa
2 † University of Salerno

Abstract. In this paper we consider the problem of designing controllers for linear plants to be implemented in embedded platforms under stringent real-time constraints. These include preemptive scheduling schemes, under which the maximum execution time allowed for control software tasks is uncertain. We propose an "anytime control" design approach, consisting in a hierarchy of controllers for the same plant. Higher controllers in the hierarchy provide better closed-loop performance, while typically requiring a larger worst-case execution time. We provide a procedure for the design of controllers which, together with a conditioning process of the stochastic scheduling, provides better performance than prevailing worst case-based design, while guaranteeing almost sure stability of the resulting switching system.

1 Introduction

A general tendency can be observed in embedded systems towards implementation of a great variety of concurrent real-time tasks on the same platform, thus reducing the overall HW cost and development time. Among such tasks, those implementing control algorithms are usually highly time critical, and have traditionally imposed very conservative scheduling approaches, whereby execution time is allotted statically, which makes the overall architecture extremely rigid, hardly reconfigurable for additions or changes of components, and often underperforming.

Modern multitasking Real-Time Operating System (RTOS), running e.g. on embedded ECUs in the automotive domain, schedule their tasks dynamically, adapting to varying load conditions and Quality of Service requirements. Real-time preemptive algorithms, such as e.g. Rate Monotonic (RM) and Earliest Deadline First (EDF) [1–3] can suspend the execution of a task in the presence of requests by other, higher priority tasks. Guarantees of schedulability can be provided based on estimates of the Worst-Case Execution Time (WCET) of tasks.

⋆ This work was supported by the EC under contract IST 511368 (NoE) "HYCON - Hybrid Control: Taming Heterogeneity and Complexity of Networked Embedded Systems" and contract IST 045359 "PHRIENDS" - Physical Human-Robot Interaction: Dependability and Safety"

However, to make a given set of tasks schedulable and to limit the number of deadline misses, conservative assumptions are typically made which entail underexploitation of the computational platform and, ultimately, cost inefficiencies: for instance, a famous result of [1] shows that RM scheduling can meet all deadlines if the CPU utilization is not larger than 69.3%. On the other hand, when the computational power budget is given and fixed (which is often the case in industrial practice), then control algorithms may have to be drastically simplified to be computable within the allotted time. This clearly reflects in a degradation of the overall performance of the ensuing closed-loop system.

Substantial performance improvement would be gained if less conservative assumptions could be made on the CPU utilization. In particular, it is often the case that, for most of the CPU cycles, a time τ could be made available for a control task which is substantially longer than τ_{min}, although only the latter can be guaranteed in the worst case.

In this paper we propose a strategy to design control algorithms and to schedule their execution, so that the limits of current practice are overcome and better performance could be obtained with the same resources.

2 Anytime Control Algorithms

The key idea is to design controllers which can be implemented so that a useful result is guaranteed whenever the algorithm is run for at least τ_{min}; however, better results can be provided if longer times are allowed.

The idea is borrowed from so-called *anytime algorithms*, that have been proposed in real-time computation [4, 5]. The characteristic of anytime algorithms (or of *imprecise computation*, as they are sometimes referred to) is to always return an answer on demand; however, the longer they are allowed to compute, the better (e.g. more precise) an answer they will return. Thus, an anytime algorithm can be interrupted prematurely, still providing a valid result and improving the output accuracy as the available time increases. A periodic task is split in a *mandatory* part and one or more *optional* parts. The criticality of hard RT tasks is preserved ensuring only that the mandatory parts satisfy the time constraints. If all mandatory parts of a set of tasks are schedulable, *feasible mandatory constraint* is satisfied [4].

In digital filter design [6], this philosophy has been pursued by decomposing the full-order filter in a cascade of lower order filters whose execution is prioritized. Execution of code implementing the first block is always guaranteed within τ_{min}; code for blocks in the cascade is then executed sequentially, until a preemption event takes over. The latest computed block output is used as the anytime filter output. The overall performance of the filter was shown in [6] to be superior the the conservative solution of always using only the first filter block.

To adopt the anytime approach in the control domain, a classical monolithic control task should be replaced by a hierarchy of control tasks of increasing complexity, each providing a correspondingly increasing performance of the controlled system. For instance, the simplest control task in the hierarchy, which must be executable within τ_{min}, could be designed to guarantee only stability of the closed loop system, while whenever the scheduler provides "surplus" time, other more sophisticated control algorithms could be executed to obtain better "quality of control".

However, application of the anytime algorithm idea to control is much more challenging than it may superficially appear. The main conceptual roadblock is that, as opposed to most anytime computation and filtering algorithms, anytime controllers interact in feedback with dynamic systems, which fact entails issues such as

Hierarchical Design: the design of a set of controllers as progressive approximations towards a given target design does not typically provide the desired performance hierarchy. Indeed, performances of closed-loop systems are not trivially related to how close approximations are to the target, as it is e.g. in filter design;

Switched System Performance: unpredictable preemption events introduce stochastic switching among different closed-loop systems, which can subvert naïve expectations — e.g., switching between stabilizing controllers may well result in overall instability. More generally, closed-loop performance is strongly influenced by switching;

Practicality: implementation of both control and scheduling algorithms must be numerically accurate, yet very simple and non-invasive, not to contradict the very nature of the limited-resource, embedded control problem.

Composability: the computational structure of control algorithms should be inherited through the hierarchy levels, so that the computation of higher controllers in the hierarchy exploits results of computations executed for lower controllers. Although this property is not strictly required, it can greatly enhance effectiveness of anytime control.

3 Prior work

We will illustrate the relevance of the above issues with reference to prior work in the field. A first attempt to use the anytime control idea is reported in [7, 8], where standard system reduction methods (balanced truncation or modal decomposition) are used to decompose a target controller in simpler ones. Unfortunately, these methods do not provide any guarantee on closed-loop performance of the simplified controllers (not even stability, indeed). Even if ad-hoc choices are made that provide stabilizing controllers, stability under switching is not guaranteed by the method in [7, 8], unless a substantial dwelling time is assumed between switches, during which the constant use of the same controller is possible.

On the other hand, the substantial literature on *switching system* stability (see e.g. [9–12] and references therein) provides much inspiration and ideas for the problem at hand, but few results can be used directly. For instance, application of the important results of [13] would provide state-space realizations of different stabilizing controllers such that the overall closed-loop systems would remain stable under *any* switching law. Unfortunately, however, the method is thought for a different application, and assumes all controllers are designed by the internal-model approach, and have the same (rather heavy) computational complexity; most importantly, at each switching instant, a state-space transformation has to be applied, which is of comparable complexity as the controllers themselves. By the same practicality argument, algorithms for switched system stabilization (such as e.g. [14–16]) requiring the computation of complex functions of the state to ascertain which subsystem can be activated next time, are not applicable to anytime control.

In [17], the authors proposed a framework for the stability analysis of anytime control algorithms, based on a stochastic model for the scheduler. A set of controllers forming a hierarchy in complexity and performance was assumed to be given. Under these hypotheses, a switching policy capable of conditioning the stochastic properties of the scheduler was designed, such that overall stability (in the probabilistic sense of "almost sure" stability [18, 19]) of the resulting Markov Jump Linear System (MJLS) can be guaranteed. This paper complements [17] by providing a constructive design procedure for anytime controllers, and by illustrating the application of the complete methodology to two examples. The modelling framework and the main results of [17] are succinctly reported for the reader's convenience.

4 Scheduling Problem Formulation and Solution

Let $\Sigma \triangleq (A, B, C)$ be the given strictly proper linear, discrete time, invariant plant to be controlled, and let $\Gamma_i \triangleq (F_i, G_i, H_i, L_i)$, $i \in I \triangleq \{1, 2, \ldots, n\}$ be a family of feedback controllers for Σ. Assume that all controllers Γ_i stabilize Σ and are ordered by increasing computational time complexity, i.e. $WCET_i > WCET_j$ if $i > j$. Let the closed-loop systems thus obtained be $\Sigma_i \triangleq (\widehat{A}_i, \widehat{B}_i, \widehat{C}_i)$, where

$$\widehat{A}_i = \begin{bmatrix} A + BL_iC & BH_i \\ G_iC & F_i \end{bmatrix}; \widehat{B} = \begin{bmatrix} B_i \\ 0 \end{bmatrix}; \widehat{C}_i = \begin{bmatrix} C_i \; 0 \end{bmatrix}.$$

Problems related to jitter and delay are not considered in this work since they can be tackled in the design of the single controllers ([20]). Therefore, we assume that measurements are acquired and control inputs are released at every sampling instant tT_g, $t \in \mathbb{N}$, where T_g is a fixed sampling time. Let $\gamma_t \in [\tau_{min}, \tau_{max}]$, $\tau_{max} < T_g$, denote the time allotted to the control task during the t-th sampling interval. By hypothesis, $WCET_1 \leq \tau_{min}$ and $WCET_n \leq \tau_{max}$.

Define an event set $L_\tau \triangleq \{\tau_1, \ldots, \tau_n\}$, and a map

$$\begin{aligned} \mathcal{T} : [\tau_{min}, \tau_{max}] &\rightarrow L_\tau \\ \gamma_t &\mapsto \tau(t) \end{aligned}$$

where

$$\tau(t) = \begin{cases} \tau_1, \text{ if } \gamma_t \in [\tau_{min}, WCET_2) \\ \tau_2, \text{ if } \gamma_t \in [WCET_2, WCET_3) \\ \vdots \quad \text{if} \vdots \\ \tau_n, \text{ if } \gamma_t \in [WCET_n, \tau_{max}] \end{cases}$$

Assume a stochastic description of the scheduling process to be given by

$$\Pr\{\tau(t) = \tau_i\} = \bar{\pi}_{\tau_i},\ 0 < \bar{\pi}_{\tau_i} < 1,\ \sum_{i \in I} \bar{\pi}_{\tau_i} = 1,$$

where $\bar{\pi}_{\tau_i}$ denotes the probability associated to the event that the time slot γ_t is such that all controllers $\varGamma_j, j \leq i$, but no controller $\varGamma_k, k > i$, can be executed. The distribution $\bar{\pi}_\tau = [\bar{\pi}_{\tau_1}, \bar{\pi}_{\tau_2}, \cdots, \bar{\pi}_{\tau_n}]^T$ can be regarded simply as an i.i.d. process, or, in a slightly more complex but general way, as the invariant probability distribution of a finite state discrete-time homogeneous irreducible aperiodic Markov chain given by

$$\pi(t+1) = P^T \pi(t),\ \pi(0) = \pi_0.$$

where $P = (p_{ij})_{n \times n}$ is the transition probability matrix and p_{ij} is the transition probability from state i to state j of the Markov chain (e.g. from controller $\varGamma_i$ to $\varGamma_j$).

Under these hypotheses, the switching process generates a discrete-time Markov Jump Linear System (MJLS)

$$x_{t+1} = \widehat{A}_{\tau_t} x_t \tag{1}$$

Definition 1. *[19] The MJLS (1) is said almost surely stable (AS-stable) if there exists $\mu > 0$ such that, for any $x_0 \in \mathbb{R}^N$ and any initial distribution π_0, the following condition holds*

$$\Pr\left\{\limsup_{t \to \infty} \frac{1}{t} \ln \|x_t\| \leq -\mu\right\} = 1.$$

Let $\|\cdot\|$ be a matrix norm induced by some vector norm. The following sufficient condition for AS-stability was proved in [18]:

Theorem 1 (1–step average contractivity). *[18] If*

$$\xi_1 = \prod_{i \in I} \|\widehat{A}_{\tau_i}\|^{\overline{\pi}_{\tau_i}} < 1 \tag{2}$$

then the MJLS (1) is AS-stable.

4.1 Stochastic Schedule Conditioning

We define a *switching policy* to be a map $s : \mathbb{N} \to I$, $t \mapsto s(t)$, which determines an upper bound to the index i of the controller to be executed at time t, i.e. $i \leq s(t)$. In other terms, at time tT_g, the system starts computing the controller algorithm until it can provide the output of $\Gamma_{s(t)}$, unless a preemption event occurs forcing it to provide only $\Gamma_{\tau(t)}$, i.e. the highest controller computed before preemption. Application of a switching policy s to a set of feedback systems Σ_i, $i \in I$ under a scheduler τ generates a switching linear system (Σ_i, τ, s) which, under suitable hypotheses, is also a MJLS. The stochastic characterization of the chain τ is assumed to be a-priori known. Furthermore, in a real application (e.g. automotive domain) different working conditions lead to different stochastic descriptions, thus different Markov chains for the scheduler can be considered.

As an example, the most conservative policy is to set $s(t) \equiv 1$, i.e. forcing always the execution of the simplest controller Γ_1, regardless of the probable availability of more computational time. By assumption, this (non-switching) policy guarantees stability of the resulting closed loop system.

On the opposite, a "greedy" strategy would set $s(t) \equiv n$, which leads to providing $\Gamma_{\tau(t)}$ for all t. Although this policy attempts at maximizing the utilization of the most performing controller, it is well known that switching arbitrarily among asymptotically stable systems Σ_i may easily result in an unstable behavior [21].

A sufficient condition for the greedy switching policy to provide an AS-stable system is provided by Theorem 1. This condition however is rarely satisfied. Indeed, the fact that each matrix $\widehat{A}_{\tau_i}$ is Schur guarantees the existence of a specific norm $\|\cdot\|_{w_i}$ such that $\|\widehat{A}_{\tau_i}\|_{w_i} < 1$, but no single norm $\|\cdot\|_w$ exist in general such that $\|\widehat{A}_{\tau_i}\|_w < 1 \ \forall \tau_i$[3]. The AS stability condition of theorem 1 would require that, for a chosen norm, for all controllers with $\|\widehat{A}_{\tau_i}\|_w > 1$ $\bar{\pi}_{\tau_i}$ is sufficiently small, i.e. they are scheduled by the OS sufficiently rarely.

A switching policy that suitably conditions the scheduler to provide AS-stability was studied in [17], which is illustrated below. Introduce a homogeneous irreducible aperiodic Markov chain σ with the same number n of states as the scheduler chain τ. The states are labelled as σ_i, with the meaning that if the associated process form $\sigma(t)$ is equal to σ_i, then $s(t) = i$, i.e. in the next sampling interval tT_g at most the i–th controller is computed (this will actually happen if no preemption occurs). We will refer to σ as the conditioning Markov chain. The synthesis of such a conditioning Markov chain can be formulated as the

[3] When this happens, such norm is a common Lyapunov function and the system remains stable for all switching sequences. It is well known that this is rarely the case.

	Comp. Compl.	Num. Reliab.
Generic	$N\,(N+2)$	–
Companion	$2N$	bad
Jordan	$2N$ to $3N$	good

Table 1. Computational complexity (considered as the number of multiplications except by 0 or 1) and numerical reliability of different state-space realizations of a strictly proper transfer function $G(z)$ with N poles. The generic case assumes no particular structure in the systems matrices.

following Linear Programming problem:

$$\begin{array}{ll} \multicolumn{2}{c}{\text{Find a vector } \overline{\pi}_\sigma = \left[\overline{\pi}_{\sigma_1} \cdots \overline{\pi}_{\sigma_n}\right]^T \text{ such that}} \\ 1) & \sum_{i=1}^{n} c_i \overline{\pi}_{\sigma_i} < 0 \\ 2) & 0 < \overline{\pi}_{\sigma_i} < 1 \\ 3) & \sum_{i=1}^{n} \overline{\pi}_{\sigma_i} = 1, \end{array} \tag{3}$$

where

$$c_i = \sum_{h=1}^{n} \overline{\pi}_{\tau_h} \ln\left(\left\|\widehat{A}_{\min(\tau_h,\sigma_i)}\right\|\right).$$

Should this problem not have a feasible solution, multi–step switching policies can be considered, whereby the conditioning Markov chain suggests the sequence of controllers to be executed in the next m steps (see [17] for details). This way, some control patterns, i.e. substrings of symbols in I, are preferentially used with respect to others.

5 Design of a Control Algorithm Hierarchy

In this section we address the problem of designing an ordered set of control algorithms providing increasing closed-loop performance. A top-down design approach to this problem would start with the design of a complex, high-performance controller Γ_n (by e.g. a H_∞ technique); progressively simpler controllers Γ_i, $n-1 \geq i \geq 1$ may then be obtained by e.g. model reduction techniques. As already remarked, however, this approach does not systematically guarantee closed-loop performance under switching. Moreover, most model reduction techniques require state-space realizations with full dynamic matrices F_i, which makes them impractical in real-time embedded applications.

Indeed, practicality requirements imply careful consideration of algorithmic implementations of control laws [22]. Table 1 reports a comparison among three different state space representations for SISO systems.

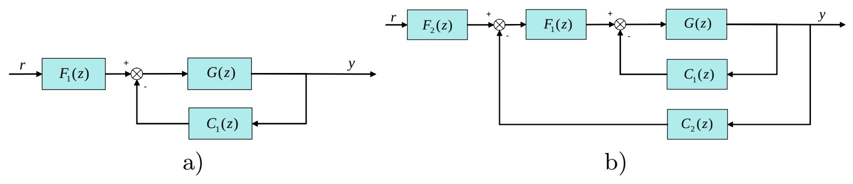


Fig. 1. Two stages of a classical cascade design procedure

We propose here a simple, bottom-up design technique which is suitable for addressing the main requirements of anytime control algorithms. The method is based on classical cascade design. Consider the two design stages illustrated in fig. 1, in which controllers are designed to ensure increasing performance by any classical synthesis technique. The scheme in fig. 1 cannot be implemented as a composable anytime control, because after computation of the a) scheme, the input to the $F_1(z)$ block needs to be recomputed completely if the b) scheme is to be applied. However, by simple block manipulations, the scheme in fig. 2 can be obtained, where we set

$$\hat{C}_2(z) = F_1(z)C_2(z).$$

The scheme in fig. 2 is suitable for anytime implementation. Indeed the se-

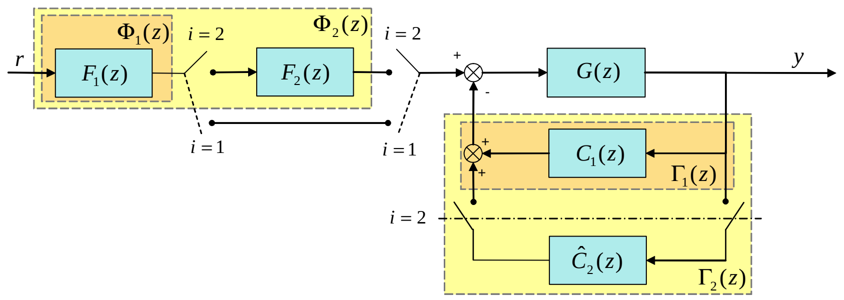


Fig. 2. A switched control scheme suitable for anytime control implementation. The scheme is equivalent to fig. 1-a when the switches are in the $i = 1$ position, and to fig. 1-b for $i = 2$.

ries of $F_1(z)$ and $F_2(z)$ is in open-loop (hence equivalent to an anytime filter), while the parallel connection in the feedback loop is simply obtained by summing the new result by $\hat{C}_2(z)$ to the previous one by $C_1(z)$. Using Jordan form realizations of the blocks provides good numerical accuracy as well as low computational complexity. The cascade design method can be applied iteratively to provide a complete hierarchy of controllers, satisfactorily addressing the issues of hierarchical design, practicality, and composability.

6 Tracking Control and Bumpless Transfer

The schedule conditioning technique of section 4.1 is able to address the switched system performance issue satisfactorily when a regulation problem is considered. However, in reference tracking tasks, the performance can be severely impaired by switching between different controllers. This section is devoted to analyze this problem and propose a simple technique to assist in making smooth transitions for the system switching between controllers.

Consider the problem of tracking a constant or slowly-varying set-point r. With reference to the design scheme in fig. 2, the problem can be solved by scaling the reference input r by the steady state gain of each controller Γ_i, $1 \leq i \leq n$. Let x_s denote the state of the controlled plant, and x_{c_i} the state of the i–th controller component $C_i(z)$, and let $\bar{x}_s$, $\bar{x}_{c_i}$ denote the corresponding equilibrium values reached when the reference r is applied. Suppose now that, at some instant in time tT_g, the i–th level controller is active and the system components are at the equilibrium state $\bar{x}_s, \bar{x}_{c_1}, \cdots, \bar{x}_{c_i}$; and that, at time $(t+1)T_g$, the execution of the j–th level controller is imposed by a preemption event or a conditioned schedule. If $j \leq i$, it can be easily verified that the active part of the system state remains at an equilibrium $\bar{x}_s, \bar{x}_{c_1}, \cdots, \bar{x}_{c_j}$. If instead $j > i$ (low-to-high level switching), the state is perturbed from the equilibrium. Indeed, the activation of a higher level controller abruptly introduces the dynamics of the re-activated (sleeping) states.

Sleeping states can be managed such that they are kept constant during their idle period, or they can be zeroed instantaneously or progressively with a simple and computationally inexpensive dynamics. No matter on how they are managed, their re-activation can produce a jump in the state value. Notice that keeping constant their values results in no jumps only if also the reference does not change. These jumps produce an undesirable behavior from a performance point of view. The use of a bumpless-like technique is advisable to cope with this issue. According to this approach, the idle states are re-set to suitable values before re-activation to avoid jumps and to leave the system at its equilibrium, if such was the case before the switching occurs. The sleeping state initialization value is computed as

$$x_{\text{sleeping}_j} = u_{c_j}(I - A_{c_j})^{-1} B_{c_j} = u_{c_j} W$$

where u_{c_j} is the current input to the j–th part of the controller. Notice that the switching logic introduces negligible overhead, since the vector W can be computed off–line.

7 Examples

The control of the two mechanical systems depicted in fig. 3 will be used to illustrate the application of the proposed technique. We report in table 2 the sampled-time linearized dynamics of the two systems (continuous-time models are readily available in the literature, see e.g. [23, 24]).

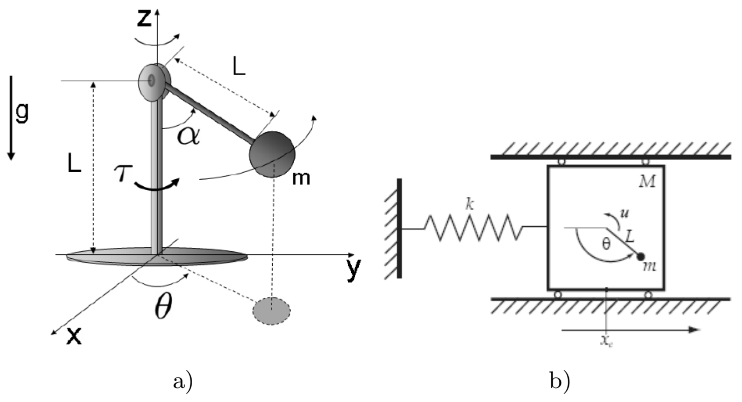


Fig. 3. Mechanical systems adopted for the anytime controller simulations: a Furuta pendulum with zero offset ([23] - a) and a Translational Oscillator/Rotational Actuator (TORA) system ([24] - b)

In both examples, the first controller $C_1(z)$ is designed to ensure the stability requirement. The controllers $C_2(z)$ and $C_3(z)$ for the Furuta pendulum example are obtained applying twice in cascade an LQG design technique. For the TORA example, the controllers are instead carefully designed by hand to achieve performance enhancement with minimal complexity increase, i.e more *practical* in the sense of section 2). Prefilters $F_i(z)$ (depicted in fig. 2) are constants used to adapt the steady-state gain and ensure static requirements. The scheduler process is modeled simply by its steady-state probability distribution, which is assumed equal in the two examples and given by $\bar{\pi}_\tau = [1/20, 5/20, 14/20]$.

For the Furuta pendulum example in fig. 3-a, solving the LP problem 3 leads to a steady state conditioning probability distribution $\bar{\pi}_\sigma = [0.017, 0.98, 0.003]$. The resulting conditioned distribution $\bar{\pi}_d = [0.058, 0.94, 0.002]$ thus satisfies the 1-step average contractivity condition (2), hence AS-stability is guaranteed.

In fig. 4-a, the *Root Mean Squares* (RMS) of the regulation error for different controllers is shown, corresponding to perturbed initial conditions $x_0 = [0, \pi/10, 0]^T$. Plots labeled Controller 1, 2, and 3, corresponding to results obtained without switching, are reported for reference. Notice the performance increase obtained by more complex controllers. Fig. 4-b shows a sample realization of the stochastic process used to model the OS scheduler. The RMS obtained by the greedy switching policy applied to this schedule shows instability. Notice that the axis labels on the right apply to this plot in fig. **??**-a. On the same figure **??**-a, the plot labeled "Markov" shows the RMS error obtained by the stochastically conditioned scheduler. Sample realizations of the conditioning and conditioned stochastic schedules used in simulations are reported in fig. 4-b.

The example shows how the proposed stochastic switching policy ensures the AS-stability of the closed loop system (which is not guaranteed by the greedy policy), while it obtains a definite performance increase (of the order of 50%) with respect to the conservative scheduler (corresponding to using only Con-

Furuta Pendulum System $G(z) = \frac{1.2\,10^{-3}(z+3.7)(z+0.3)}{(z-1)(z^2-1.7z+1)}$	
$C_1(z) = \frac{31.6(z^2-1.8z+1.1)}{(z-0.1)(z-0.5)}$	$F_1(z) = 21.28$
$C_2(z) = \frac{117.7(z-1.3)(z-4.7\,10^{-3})(z^2-1.4z+0.7)}{(z-0.5)(z+0.2)(z-0.1)(z^2+0.4z+0.9)}$	$F_2(z) = 0.54$
$C_3(z) = \frac{1370.9(z-0.4)(z-0.6)(z-0.2)(z+0.2)(z-4.7\,10^{-3})(z^2-0.7z+0.2)(z^2-0.4z+0.3)}{(z-0.5)^2(z+0.3)^2(z-0.1)^2(z^2+0.6z+0.8)(z^2+3.4z+4.7)}$	$F_3(z) = 2.73$

Tora System $G(z) = \frac{0.27266(z+1)(z^2-1.967z+1)}{(z-1)^2(z^2-1.964z+1)}$	
$C_1(z) = \frac{2.0895(z-0.75)}{(z+0.3761)}$	$F_1(z) = 0.38$
$C_2(z) = \frac{0.8(z-0.4)}{(z+0.6)}$	$F_2(z) = 1.79$
$C_3(z) = \frac{0.73(z^2-0.76z+0.2228)}{(z+0.3)^2}$	$F_3(z) = 1.29$

Table 2. Sampled-time transfer functions for systems in fig. 3 and hierarchical controllers used in simulations.

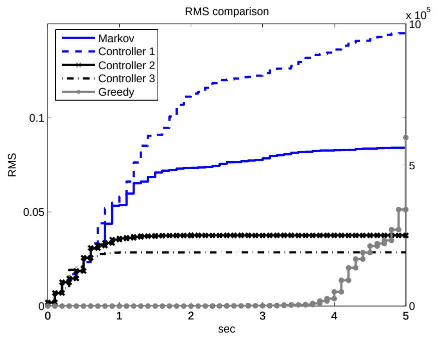
troller 1, see fig. 4-a).

Fig. 5 reports similar plots to illustrate application of the proposed methodology to the TORA example (fig. 3-b), with controllers chosen according to table 2. No solution to the 1-step average contractivity condition could be found in this case. A four-steps lifted version of the problem admits a solution, according to which the conditioning sequence of controllers is a concatenation of the 3^4 possible combinations of length 4 of the three controllers. The steady state conditioning probability distribution $\bar{\pi}_\sigma \in (0,1)^{3^4}$ is not reported here; it is however worth noticing that the particular controller sequence $\Gamma_2 - \Gamma_2 - \Gamma_2 - \Gamma_3$ is by far the most likely, being used in the $89,204\%$ of cases (except for preemptions). In figure 5-b, the scheduled and the conditioned controllers are depicted: the prevalence of the preferred pattern is apparent.

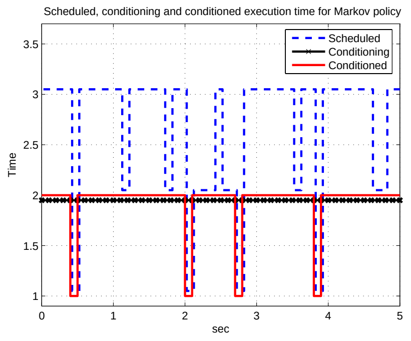
Results of simulations of the different controllers and scheduling policies are reported in fig. 5-a, for a regulation problem from perturbed initial conditions $X_0 = [0, 0, 0.1, 0]^T$.

The RMS performance plots in fig. 5-a show that the greedy policy (in this particular case) does not lead to divergence. However, it is quite remarkable that our proposed policy performs better than both the greedy and the conservative policies (indeed, slightly better than even using always Controller 2, which is not a feasible choice).

Finally, results of application of the proposed technique for a tracking control problem for the TORA example are reported in fig. 6. The reference to be tracked is a piecewise constant signal of amplitude $\pi/4$, period 10 seconds and pulse width of 30%. The comparison of RMS performance shows that direct application of the conditioned switching policy performs poorly in the tracking case. This is due to the re-activation issues pointed out earlier. Using the simple



a)



b)

Fig. 4. Regulation results for the Furuta pendulum example: a) RMS error of the closed loop system with different control schedulings b) allowable execution times provided by the OS scheduler, and the conditioning and conditioned processes.

bumpless switching technique proposed in Section 6, a significant performance improvement is achieved, as shown in both figures 6-a,b. The RMS performance of the bumpless conditioned switching policy is better than both the greedy and the conservative approaches.

8 Conclusions

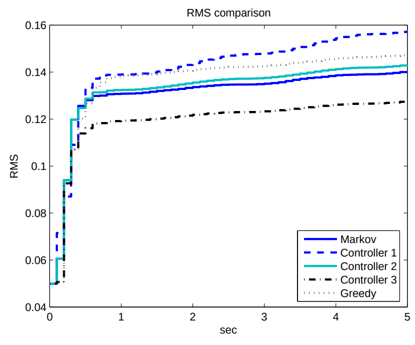
We have shown that underexploitation of CPU time caused by conservative control scheduling policies can be effectively reduced, and control performance can be significatively enhanced, by adopting a schedule conditioning algorithm that uses a stochastic model of a preemptive RTOS scheduler.

We also discussed ideas for the design of controllers which, together with a conditioning process of the stochastic scheduling, provide better performance than prevailing worst case-based design, while guaranteeing almost sure stability of the resulting switching system. A practical and effective technique for bumpless switching has been introduced, to reduce the negative effects of switching in tracking problems.

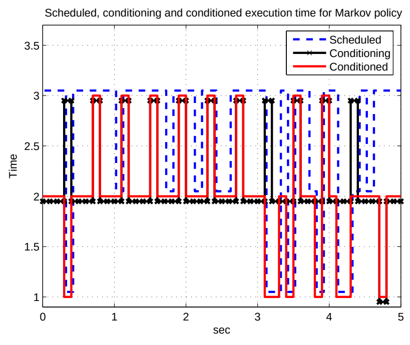
Much work remains to be done on a systematic design procedure for arriving at a hierarchical, composable, practical design of controllers for anytime implementation, and on numerical apects involved in the solution of the (multi-step) average contractivity equation.

References

1. Liu, C.L., Layland, J.W.: Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the Association for Computing Machinery **20**(1) (1973)



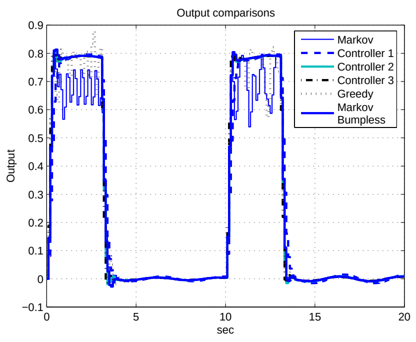
a)



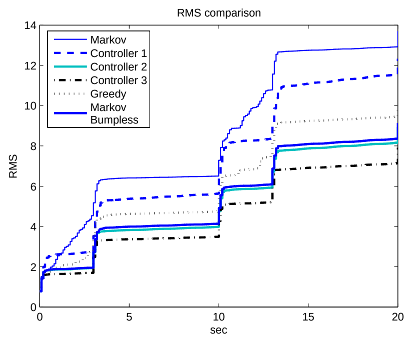
b)

Fig. 5. Regulation results for the TORA example: a) RMS error of the closed loop system with different control schedulings b) allowable execution times provided by the OS scheduler, and the conditioning and conditioned processes.

2. Liu, J.W.S.: Real–Time Systems. Prentice Hall Inc., Upper Saddle River, NJ (2000)
3. Buttazzo, G.: Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. Kluwer Academic Publishers, Boston (1997)
4. Liu, J.W.S., Shih, W.K., Lin, K.J., Bettati, R., Chung, J.Y.: Imprecise computation. Proceedings of the IEEE **82**(1) (January 1994) 83–93
5. Liu, J.W.S., Lin, K.J., Shih, W.K., Yu, A.C.S., Chung, J.Y., Zhao, W.: Algorithms for scheduling imprecise computations. Computer **24**(5) (1991) 58–68
6. Perrin, N., Ferri, B.: Digital filters with adaptive length for real–time applications. In Le Royal Meridien, K.E., ed.: Proc. IEEE Real-Time and Embedded Technology and Applications Symposium, Toronto, Canada (May 2004)
7. Bhattacharya, R., Balas, G.J.: Implementation of control algorithms in an environment of dynamically scheduled CPU time using balanced truncation. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, Monterey, CA (August 2002)
8. Bhattacharya, R., Balas, G.J.: Anytime control algorithm: Model reduction approach. Journal of Guidance, Control, and Dynamics **27**(5) (2004) 767–776
9. Branicky, M.S.: Stability of hybrid systems: State of the art. In: Proc. 36th IEEE Conf. On Decision and Control, San Diego, California, USA (December 1997) 120–125
10. DeCarlo, R.A., Branicky, M.S., Pettersson, S., Lennartson, B.: Perspectives and results on the stability and stabilizability of hybrid systems. IEEE Proceedings **88**(7) (2000) 1069–1082
11. Liberzon, D., Morse, A.S.: Basic problems in stability and design of switched systems. IEEE Contr. Syst. Mag. **19**(5) (1999) 59–70
12. Ye, H., Michel, A.N., Hou, L.: Stability theory for hybrid dynamical systems. IEEE Trans. Automat. Contr. **43**(4) (1998) 461–474
13. Hespanha, J.P., Morse, A.S.: Switching between stabilizing controllers. Automatica **38**(11) (2002) 1905–1917



a)



b)

Fig. 6. Tracking results for the Tora system: a) output signals; b) RMS errors.

14. Wicks, M.A., Peleties, P., DeCarlo, R.: Construction of piecewise Lyapunov functions for stabilizing switched systems. In: Proc. 33rd IEEE Conf. On Decision and Control, Lake Buena Vista, FL (December 1994) 3492–3497
15. Wicks, M., DeCarlo, R.: Solution of coupled Lyapunov equations for the stabilization of multimodal linear systems. In: Proc. American Control Conf., Albuquerque, NM (June 1997) 1709–1713
16. Pettersson, S., Lennartson, B.: Stabilization of hybrid systems using a min-projection strategy. In: Proc. American Control Conf., Arlington, Virginia (June 2001) 223–228
17. Greco, L., Fontanelli, D., Bicchi, A.: Almost sure stability of anytime controllers via stochastic scheduling. In: Proc. IEEE Int. Conf. on Decision and Control, New Orleans, LO (December 2007)
18. Fang, Y., Loparo, K., Feng, X.: Almost sure and δ-moment stability of jump linear systems. Int. J. Control **59**(5) (1994) 1281–1307
19. Bolzern, P., Colaneri, P., Nicolao, G.D.: On almost sure stability of discrete-time Markov jump linear systems. In: Proc. 43rd IEEE Conf. On Decision and Control. Volume 3. (2004) 3204–3208
20. Cervin, A., Lincoln, B., Eker, J., Årzén, K.E., Buttazzo, G.: The jitter margin and its application in the design of real-time control systems. In: Proc. 10th Int. Conf. on Real-Time and Embedded Computing Systems and Applications, Gothenburg, Sweden (August 2004)
21. Liberzon, D., Hespanha, J.P., Morse, A.S.: Stability of switched systems: A Lie-algebraic condition. Systems & Control Letters **37**(3) (1999) 117–122
22. Åström, K., Wittenmark, B.: Computer Controlled Systems. Prentice Hall Inc. (November 1996)
23. Furuta, K., Yamakita, M., Kobayashi, S.: Swing-up control of inverted pendulum using pseudo-state feedback. Proceedings of the Institution of Mechanical Engineers. Pt.I. Journal of Systems and Control Engineering **206**(I4) (1992) 263–269
24. Bupp, R., Bernstein, D., Coppola, V.: A benchmark problem for nonlinear control design: Problem statement, experimental testbed and passive, nonlinear compensation. In: Proc. Amer. Contr. Conf. (1995) 4363–4367