49th IEEE Conference on Decision and Control
December 15-17, 2010
Hilton Atlanta Hotel, Atlanta, GA, USA

# On Anytime Control of Nonlinear Processes through Calculation of Control Sequences

Vijay Gupta  Daniel E. Quevedo

*Abstract*— **We present two related algorithms to calculate the control input when the processing resources available are time-varying. The basic idea is to calculate the control input for as many time steps into the future as allowed by the available processing resources at every time step. We analyze the stability of the resulting closed loop system using stochastic Lyapunov functions. For the LQG case, we provide explicit analytical performance and stability expressions. For more general cases, we indicate through numerical simulations that the increase in performance due to the proposed algorithm can be significant.**

## I. Introduction

Networked and embedded control have recently emerged as an important research focus. One challenging issue in such systems is that of time-varying and limited processing power. As more and more objects are equipped with microprocessors that are responsible for multiple functions such as control, communication, data fusion, system maintenance and so on, the implicit assumption traditionally made in control about the processor being able to execute the desired control algorithm at any time will break down. Similarly, if a remote controller controls many devices, multiple control tasks will compete for shared processor resources, leading to constrained availability of processing resources for any particular control loop. It is, thus, of interest to study control in the presence of limited and time-varying availability of processing power.

Owing to its importance, there are a growing number of works in this area. The impact of finite computational power has been looked at most closely for techniques such as receding horizon control (RHC). McGovern and Feron [16], [17] presented bounds on computational time for achieving stability for specific optimization algorithms, if the processor has constant, but limited, computational resources. Henriksson et al [9], [10] studied the effect of not updating the control input in continuous time systems for the duration of the computational delay for optimization algorithms based on active set methods. Also related are works on event-triggered and self-triggered control systems [24], [25], where a control input is calculated aperiodically, but on demand, depending on the process state. Finally, we would like to mention the related work in scheduling of control tasks ([2], [3] and the references therein) that looks at the problem of processor queue scheduling, when control calculation is merely one of the tasks in the queue.

One approach popular in real-time systems to tolerate the presence of time-varying processing resources is to develop *anytime* algorithms that provide *a* solution even with limited processing resources, and refine the solution as more resources become available. In control, however, there are very few methods available for developing anytime controllers. A notable work is that of Bhattacharya et al [1] who focused on linear processes and controllers, and presented a controller that updated a different number of states depending on the available computational time.Another important work is that of Greco et al [5], who proposed switching among an existing set of controllers that may require different execution times. In Gupta [6], [7], an anytime algorithm for systems with multiple inputs was presented, that was based on calculating the components of the control vector sequentially, and refining the process model as more processing time becomes available.

In this paper, we develop two related anytime control algorithms that provide better performance as more processing time is available. The basic idea is to utilize the extra processing time to compute tentative future control inputs. Although reminiscent of receding horizon control (in particular, works such as [23]), our algorithms do not solve a sequence of optimization problems of increasing size. Rather, we calculate the control values sequentially, reutilizing the already computed control inputs for the next computation. For general nonlinear processes, we analyze the stochastic stability. For the Linear Quadratic Regulator (LQR) case, we provide analytic stability and performance conditions.

The paper is organized as follows. We begin in Section II by formulating the problem, and stating the assumptions. In Section III, we present the proposed algorithms, and analyze the stochastic stability for general nonlinear processes in Section IV. The algorithm for the LQR case and the associated stability and performance results are provided in Section V. We numerically illustrate the improvement in performance using the proposed algorithm in Section VI.

## II. Problem Formulation

**Process Model:** We consider discrete-time nonlinear MIMO processes with state $x(k) \in \mathbf{R}^n$ and input $u(k) \in \mathbf{R}^p$, evolving as

$$x(k+1) = f(x(k), u(k)), \qquad k \geq 0 \tag{1}$$

Vijay Gupta is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, USA. ttvgupta2@nd.edu Research supported in part by the National Science Foundation under the grants NSF:0834771 and NSF:0846631.

Daniel Quevedo is with the School of Electrical Engineering & Computer Science, The University of Newcastle, NSW 2308, Australia. dquevedo@ieee.org Research supported by Australian Research Council's Discovery Projects funding scheme (project number DP0988601).

where $f(0,0) = 0$ and the initial state $x(0) = x_0$ is arbitrarily distributed. The state and control input may additionally have to satisfy constraints of the form $h(x(k), u(k)) \in \mathbf{S}$ for some set $\mathbf{S}$. For pedagogical ease, we assume full state feedback at the controller.

We assume the existence of a Lyapunov function $V(x(k)) \geq |x(k)|$ for the process (1) such that if sufficient computational resources were available, a *baseline* control law exists that yields the control input $\kappa(x(k))$ that satisfies $h(x(k), \kappa(x(k))) \in \mathbf{S}$ and ensures that

$$V(x(k+1)) < \epsilon V(x(k)),$$

where $x(k+1) = f(x(k), \kappa(x(k)))$ and $0 < \epsilon < 1$ is given. We assume $V(0) = 0$. For future notational ease we define

$$V_k \triangleq V(x(k)).$$

*Remark 2.1:* In the absence of model inaccuracies and disturbances, off-line state predictions would also have zero error. Presence of such inaccuracies, however, will necessitate feedback. The algorithms that we propose will use feedback whenever permitted by the available computational resources. As a first step, in this work, we concentrate on the *nominal* model described by (1) to analyze the stability of the closed loop system. We will analyze the impact of disturbances in future work.

**Processing Time Availability:** In the classical formulation, it is assumed that the processing resources available to the controller are sufficiently large so that the controller can generate the control input by essentially discounting any processing resource constraint. However, as discussed earlier, in networked and embedded systems, the computation resources available at every time step for calculating the control input may vary. For simplicity, and without loss of generality, we map the availability of processing resources at time step $k$ to availability of execution time that is available for the control calculation at time $k$. We make the following assumptions:

- The execution time $\tau(k)$ available at any time $k$ is an independent and identically distributed sequence, with a well-defined probability mass function. While stochastic models for either the availability of the execution time, or the time requirement for execution of a task, are less common than deterministic models, we note that this framework also has a long history [15], [27]. One reason to consider this framework is that if some tasks have stochastic execution time requirements, the availability of the processor for other tasks can be modeled by a probabilistic function. In any case, similar ideas as developed in this paper can be applied to deterministic models as well.
- The controller does not have a priori knowledge of the value of $\tau(k)$. This is a realistic assumption in shared systems where the controller task can be preempted by other computational tasks.

**Problem Description:** The first concern for a control system design is stability. Since the execution time availability is time-varying in a stochastic manner, the control input is random, and thus the system (1) evolves stochastically. Various stability notions for stochastic systems have been studied in the literature (e.g., [12], [13]). We are interested in the following definitions. About the equilibrium point $x = 0$, the system (1) is

- *stochastically stable* if

$$\mathbb{E}\left[\sum_{k=0}^{\infty} x^T(k)x(k)|x(0)\right] < \infty,$$

  where the expectation is taken with respect to the stochastic sequence $\{\tau(k)\}$.
- *mean square stable* if

$$\lim_{k\to\infty} \mathbb{E}\left[x^T(k)x(k)|x(0)\right] < \infty,$$

  where the expectation is taken with respect to the stochastic sequence $\{\tau(k)\}$.

In this paper, we propose and analyze two anytime control algorithms that utilize any extra execution time to enhance stability and performance of the closed loop system.

## III. Algorithm Descriptions

The algorithms are based on the following basic idea. The execution time required to calculate the control input is an increasing function of the number of control inputs that are calculated. Thus, as the processor is provided with more execution time, the controller can calculate a longer sequence of future control inputs that guarantee a decrease in Lyapunov function. These future control inputs can then be stored and used even if the execution time availability precludes any calculation at the future time steps.

More formally, we define at every time $k$, a buffer $b_k(i), i = \{1, 2, \cdots\}$ as a stack whose elements are valid control inputs. The algorithm proceeds as follows:

*Algorithm $\mathcal{A}_1$:*

1) *Initialization:* Set $b_0(1) = 0$, $k = 0$.
2) *Control update:* Do
   a) Set $j = 0$.
   b) If processor time available, calculate control input $u(k+j)$, such that it ensures $V(k+j+1) < \epsilon V(k+j)$ if elements $b_k(i+1) = u(k+i)$ for $i = 0, \cdots, j-1$ are used. If sufficient time not available, break.
   c) If $j = 0$, set all elements of buffer $b_k(.) = 0$.
   d) Set $b_k(j+1) = u(k+j)$ calculated above.
   e) Set $j = j+1$ and repeat Step 2b.
3) *Time update:* Do
   a) Set $b_{k+1}(j) = b_k(j+1)$ for all $j \geq 1$.
   b) Apply control $u(k) = b_k(1)$, update state $x(k+1)$ and the Lyapunov function $V(k+1)$.
   c) Set $k = k+1$ and go to Step 2.

The buffer provides the control input at the present time steps and suggested inputs at future time steps. At the time steps when more processor time is available, a longer trajectory of the control inputs is calculated and stored. Algorithm $\mathcal{A}_1$ assumes that as soon as the processor calculates a control input at time $k$, it throws away the remaining elements in

the buffer (see Step 2c). In the second algorithm, namely $\mathcal{A}_2$, Step 2c is eliminated. Thus, buffer elements may stem from calculations carried out at different time instants.

*Remark 3.1:* In the presentation above, there is no bound assumed on the buffer size. If present, such bounds can be readily imposed. The analysis presented in Section IV can also be extended to this case in a straight-forward manner.

*Remark 3.2:* Although the algorithm does not *refine* the control input $u(k)$ if more processing time is available at time step $k$, it is nonetheless anytime in the sense that it utilizes any extra processing time to provide a buffer against performance loss at future instances where sufficient processing time may not be available.

*Remark 3.3:* The algorithm is based on calculating a control input trajectory that guarantees a decrease in the Lyapunov function at future time steps. Alternatively, one could also conceive an algorithm wherein a reduction of the Lyapunov function as compared to *past* time steps is sought.

*Remark 3.4:* The algorithm does not require knowledge of the probability distribution of the processor time availability. Such descriptions may even be time varying for the algorithm to be implemented. However, in the analysis of the algorithm, we will require the execution time $\tau(k)$ to be independent and identically distributed with a given probability mass function.

*Remark 3.5:* The basic idea of the algorithm, i.e., calculation of future control inputs, is reminiscent of the philosophy behind receding horizon control. However, there are some differences between the two methods. For one, neither of the algorithms $\mathcal{A}_1$ or $\mathcal{A}_2$ calculate a control input to optimize a given cost function. Rather, the control input is calculated to guarantee a decrease in a specified Lyapunov function. Moreover, the number of time steps for which the control input is calculated (corresponding to the horizon of the receding horizon control method) is time-varying and externally imposed by the available processor time. As has been recognized, the effect of increasing the horizon in RHC may be counter-intuitive, e.g., performance and stability may be adversely affected as the horizon is increased [20], [18]. However, the proposed algorithms do not display such effects even though the 'horizon' is dynamically altered.

## IV. Stability Analysis

We begin by identifying conditions under which the baseline algorithm and the proposed algorithm stabilize the system. We begin with the baseline algorithm. Under the baseline algorithm, the control $u(k) = \kappa(x(k))$ is calculated if sufficient processor time is available at step $k$; otherwise no control is calculated. If no control is calculated, then a zero control input (u(k) = 0) is applied, c.f. [22]. Thus, if we denote the probability that the controller is unable to calculate any control input with a probability $p_0$, then the process evolution is similar to a networked control system in which the controller is able to communicate with the actuator with a probability $1 - p_0$ at any time step. Stability conditions for such a system have been derived both for linear systems [8], [11] and nonlinear systems [21]. Our analysis for the baseline control law follows these works closely.

We will make the following assumption throughout the sequel.

*Assumption 4.1:* There exists $1 \leq \alpha < \frac{1}{p_0}$ such that

$$V(f(x,0)) \leq \alpha V(x), \qquad \forall \text{ valid state values } x. \tag{2}$$

*Remark 4.1:* This assumption bounds the rate of increase of the Lyapunov function when no control input is calculated and applied. As an instance, for a scalar linear process with parameter $a$, the assumption implies $pa^2 < 1$, which has been shown to be necessary and sufficient for stabilizability in [8].

*Theorem 4.1:* Consider the problem formulation above when the baseline algorithm is used and Assumption 4.1 holds. The process is stable (in both stochastic and mean square sense) if

$$p_0\alpha + (1 - p_0)\epsilon < 1. \tag{3}$$

*Proof:* Omitted for space constraints. ■

For the proposed algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$, the stability analysis is more subtle and is similar to the study of randomly sampled systems [14], [26]. We begin with the stability analysis of algorithm $\mathcal{A}_1$.

Define the time steps at which a control input is calculated by the sequence $\{k_i\}$ for $i \in \{0, 1, 2, \cdots\}$, or equivalently the set $\mathcal{K}$. Also denote the time between two successive instances of calculation of the control input by $\Delta_i$, thus $\Delta_i = k_{i+1} - k_i$. For ease of exposition, we will assume $k_0 = 0$[1]. Denote the probability that the controller calculates $j$ control inputs by $p_j$ for $j \geq 0$. Since the processor time availability is i.i.d., the probabilities $p_j$ are independent of the specific time at which the inputs are calculated. Thus, as before, $p_0$ denotes the probability of no control being calculated. We begin with the following lemma.

*Lemma 4.1:* Consider the above problem formulation with the algorithm $\mathcal{A}_1$ being used. Then

$$E[V_{k_{i+1}} | x(k_i)] \leq \sum_{m=1}^{N} p_m \Omega_m V_{k_i},$$

where

$$\Omega_m = (1 - p_0)\left(\epsilon \frac{1 - (p_0\epsilon)^m}{1 - p_0\epsilon} + \alpha \frac{(p_0\epsilon)^m}{1 - p_0\alpha}\right) \geq 0.$$

*Proof:* The main idea of the proof is to condition $V_{k_{i+1}}$ on both $\Delta_i$ and the number of control packets calculated at time $k_i$. The details are omitted for space restrictions. ■

Using this result, we can analyze the stability conditions when algorithm $\mathcal{A}_1$ is used.

*Theorem 4.2:* Consider the above problem formulation with the algorithm $\mathcal{A}_1$ being used. If

$$\Omega \triangleq \sum_{m=1}^{\infty} p_m \Omega_m < 1,$$

[1]The more general case when $k_0 > 0$ can be treated similarly and without much technical difficulty.

where the terms $\Omega_m$ have been defined in Lemma 4.1, then the closed loop system is stochastically stable and mean square stable.

*Proof:* The basic idea of the proof is to note that if $\Omega < 1$, then $V_{k_i}$ is a stochastic Lyapunov function for the closed loop function at the time instants $\{k_i\}$. For the intermediate times between $k_i$ and $k_{i+1}$, we bound $E[V_k]$ by conditioning on the number of packet calculated at time $k_i$. Since $V(x(k)) \geq |x(k)|$ by assumption, this implies stochastic stability (and thus mean square stability) at all time instants. ■

The analysis of algorithm $\mathcal{A}_2$ is more involved. The primary reason for this is the fact that the number of control inputs in the buffer at time $k$, $N_B(k)$, may be more than the number of control inputs calculated at time $k$, denoted by $N_u(k)$. In fact, the evolution of the buffer length is given by

$$N_B(k) = \max\{N_B(k-1) - 1, N_u(k)\} \tag{4}$$

with the initial condition $N_B(-1) = 0$. Thus, the buffer length $N_B(k)$ is not an i.i.d. process and the analysis done for algorithm $\mathcal{A}_1$ does not extend directly.

The crucial observation for this case is that if $N_u(k) \geq N_B(k-1) - 1$, then the buffer is overwritten by the newly calculated control inputs, and $N_B(k)$ at those instants is indeed an i.i.d. process. Using this fact, we can state the following result analogous to Lemma 4.1.

*Lemma 4.2:* Consider the above problem formulation with the algorithm $\mathcal{A}_2$ being used. Then

$$E[V_{k'_{i+1}} | x(k'_i)] \leq \sum_{m=1}^{N} p_m \Omega'_m V_{k'_i},$$

where

$$\Omega'_m = \Big( \sum_{j=1}^{m} q_{j,m} \epsilon^j + \sum_{j=m+1}^{\infty} q_{j,m} \alpha^{j-m} \epsilon^m \Big),$$

and

$$q_{j,N_u(k'_i)} = \begin{cases} 1 - \sum_{l=0}^{N_u(k'_i)-1} p_l \\ \qquad j = 1 \\ (1-p_0) \prod_{k=N_u(k'_i)-2}^{0} \sum_{l=0}^{k} p_l \\ \qquad 1 < j \leq N_u(k'_i) \\ (1-p_0) p_0^{i-N_u(k'_i)} \prod_{k=N_u(k'_i)-2}^{0} \sum_{l=0}^{k} p_l \\ \qquad j > N_u(k'_i). \end{cases} \tag{5}$$

Given this lemma, the stability analysis follows along the lines of Theorem 4.2. We state the result without proof.

*Theorem 4.3:* Consider the above problem formulation with the algorithm $\mathcal{A}_2$ being used. If

$$\Omega' \triangleq \sum_{m=1}^{\infty} p_m \Omega'_m < 1,$$

where the terms $\Omega'_m$ have been defined in Lemma 4.2, then the closed loop system is stochastically stable and mean square stable.

## V. Application to LQR control

If we restrict our attention to unconstrained linear processes, the algorithm can be analyzed in more detail. Thus, consider the special case of the process evolving as

$$x(k+1) = Ax(k) + Bu(k),$$

with no constraint of the form $h(x(k), u(k)) \in \mathbf{S}$. We assume that the pair $(A, B)$ is controllable. It is known that for linear systems, one possible Lyapunov function candidate is $V_k = x^T(k) P x(k)$ for a positive definite matrix $P$. However, we discuss an alternate formulation of the algorithm for this special case that is based on the same idea as earlier; however, the algorithm in this case explicitly utilizes knowledge about the probability mass function of the execution time $\tau(k)$. This will allow us to analytically characterize the performance of the closed loop system.

We assume that the performance criterion is given by the quadratic cost

$$J = E\left[ \sum_{k=0}^{\infty} \left( x^T(k) Q x(k) + u^T(k) R u(k) \right) \right],$$

with positive definite matrices $Q$ and $R$ and where the expectation is taken with respect to the stochastic execution time availability. If there were no processor limitations, the optimal control is the LQR control law $u(k) = Kx(k)$, where $K = (B^T P B + R)^{-1} B^T P A$, and $P$ is the positive definite solution of the Riccati equation

$$P = A^T P A + Q - A^T P B (B^T P B + R)^{-1} B^T P A.$$

If there are processor constraints, then the controller is able to calculate the control input only at certain time steps. The baseline algorithm for this case is to calculate $u(k)$ at any time with probability $1 - p_0$ and to apply control 0 with a probability $p_0$. We consider a smarter version of the baseline algorithm where the probability of the calculation of a control input at any time step is taken into account explicitly. In that case, the system is a Markovian jump linear system and the optimal control is given by $u(k) = K_b x(k)$, where $K_b = (B^T P_b B + R)^{-1} B^T P_b A$, and $P_b$ is the positive definite solution of the Riccati-like equation

$$P_b = A^T P_b A + Q - (1-p_0) A^T P_b B (B^T P_b B + R)^{-1} B^T P_b A.$$

Further, the achieved cost is given by $x^T(0) P_b x(0)$.

The anytime algorithm $\mathcal{A}_1$ can be modified for this special case as follows. Suppose, for simplicity, that there is a maximum number $N_{\max}$ of control inputs that can be calculated at any time step even if the processor were fully available. As before, denote by $p_m$ the probability that $m$ control inputs were calculated at any time, $m = 1, \cdots, N_{\max}$. Then the system can be described by a Markov chain that has $N_{\max}+1$ states. The $i$-th state corresponds to the case when the buffer has $i-1$ control inputs available. Denote the fact that the Markov state at time $k$ is $i$ by $r_k = i$. Note that this may be the case if either at time $k$, enough time was available to calculate $i$ inputs, or if $r_{k-1} = i+1$ and no control was calculated at time $k$.

Going forward in time, the Markov chain transition matrix $\Phi_{for}$ can be easily calculated in terms of $p_i$. Further, we can define another Markov chain that corresponds to moving backward in time, i.e., the elements of the transition matrix $\Phi_{back}$ are given by

$$\Phi_{back}(i,j) = \text{Prob}(r_{k-1} = j | r_k = i).$$

The matrix $\Phi_{back}$ can be calculated from $\Phi_{for}$.

Thus the system evolves as a Markovian jump linear system with the Markov chain transition matrix $\Phi_{back}$. The optimal control is thus obtained as follows (see, e.g., [4]). Denote the steady state probability of being in the state $i$ of the Markov chain by $\pi_i$. Consider the coupled Riccati equations

$$\pi_j P_j = \sum_{i=1}^{N_{\max}+1} \pi_i \Phi_{back}(i,j) f_i(P_i), \tag{6}$$

where

$$f_i(X) = \begin{cases} A^T X A + Q \\ \qquad i = 1 \\ A^T X A + Q - A^T X B (B^T X B + R)^{-1} B^T X A \\ \qquad \text{otherwise} \end{cases}$$

Let $\{P_i\}$ be the positive definite solutions of the above equations (6). Then, at any time $k$, if $m$ control inputs are calculated, then the control input corresponding to time $k+l$, $l = 0, \cdots, m-1$ is calculated as $u_{k+l}(k) = K_{l+1} x(k)$, where $K_{l+1} = (B^T P_{l+1} B + R)^{-1} B^T P_{l+1} A$. Further, the cost function evaluates to $\sum_{i=1}^{N_{\max}+1} p_{i-1} x^T(0) P_i x(0)$. Some numerical examples of the performance improvement with the proposed algorithm are provided later.

## VI. Numerical Examples

In this section, we illustrate that even for simple systems, the performance improvement by using the algorithm can be significant. We first consider a process from [19] in the form (1) with the update equation

$$x(k+1) = x(k) + 0.01(x^3(k) + u(k)),$$

with the associated baseline control law $u(k) = -x^3(k) - x(k)$. It can be verified that the baseline control law stabilizes the system with the function $V(k) = x^2(k)$ as one possible Lyapunov function with associated parameter $\epsilon = 10^{-4}$. We consider the quadratic cost $J = E\left[\sum_{k=0}^{\infty} \left(x^2(k) + u^2(k)\right)\right]$, where the expectation is with respect to the availability of execution time as described below.

We assume that the execution time available is uniformly distributed in the interval $[0, 1]$. The execution time can also be viewed as the fraction of the maximum possible processor time that is available at any time step. Figure 1 shows the percentage improvement in cost achieved as a function of the time taken to calculate one control input for both algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$, as compared to the baseline algorithm through a Monte Carlo simulation of the system. A total of 1000 simulations, each of them lasting 1000 time steps, were used to generate the data. Notice that the buffer length is automatically upper bounded by the maximum execution time available. The figure shows that a significant improvement in performance can be achieved by using the proposed algorithms. The improvement in performance persists even if the controller does not have perfect knowledge of process dynamics. For the same controller as above, we consider that the true update equation is

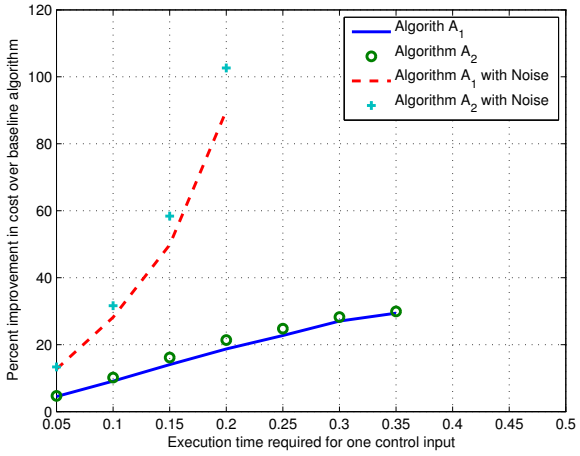


Fig. 1. Cost achieved as a function of execution time required to calculate one control input

$$x(k+1) = x(k) + 0.01(x^3(k) + u(k)) + w(k),$$

where $w(k)$ is white noise uniformly distributed in the interval $[0, 0.01]$. The cost now considered is

$$J = E\left[\lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K} \left(x^2(k) + u^2(k)\right)\right].$$

As shown in Figure 1, the proposed algorithms are reasonably robust to such perturbations.

As the system becomes more unstable, the proposed algorithms can be expected to achieve better performance compared to the baseline algorithm. Figure 2 illustrates this intuitive effect for the linear process

$$x(k+1) = \alpha x(k) + u(k),$$

with the cost

$$J = E\left[\sum_{k=0}^{\infty} \left(2x^2(k) + 2u^2(k)\right)\right],$$

as the scalar $\alpha$ is varied. The execution time availability is same as above, and the time required for calculation of one control input is assumed to be 0.3. The percentage improvement is plotted for algorithm $\mathcal{A}_1$, as compared to the 'smart' baseline algorithm considered in Section V. The plots are for the analytical expressions that were obtained for this case.

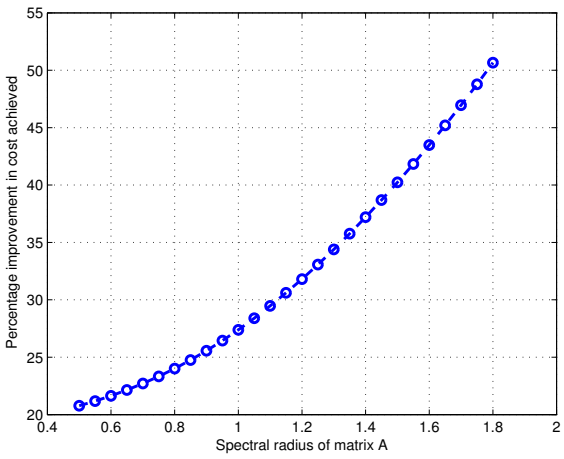


Fig. 2. Performance improvement using the proposed algorithm as a function of the spectral radius of the process

## VII. Conclusions

We proposed two related anytime control algorithms. The algorithms were based on computing a series of control inputs as more time becomes available, that guarantee a decrease in the Lyapunov function value at future time steps. Thus, even if processor does not provide sufficient resources at some time steps, the controller provides a control input. For non-linear systems, we provided analytic conditions for stochastic stability. For unconstrained linear systems, we provided analytic expressions for stability and performance gain with the algorithm. Simple numerical examples illustrated the performance gain with the proposed algorithm.

## References


[1] R. Bhattacharya and G. J. Balas, "Anytime Control Algorithms: Model Reduction Approach," AIAA Journal of Guidance, Control and Dynamics, 27(5), September-October 2004.

[2] M. Caccamo, G. Buttazzo and L. Sha, "Handling Execution Overruns in Hard Real-time Control Systems," IEEE Transactions on Computers, 51(7), July 2002.

[3] A. Cervin, J. Eker, B. Bernhardsson and K-E. Arzen, "FeedbackFeedforward Scheduling of Control Tasks", Real-Time Systems, 23(1-2), 25-53, 2002.

[4] O. L. V. Costa, M. D. Fragoso, and R. P. Marques, "Discrete-Time Markov Jump Linear Systems," Springer, Series: Probability and its Applications, 2005.

[5] L. Greco, D. Fontanelli and A. Bicchi, "Almost Sure Stability of Anytime Controllers via Stochastic Scheduling", IEEE Int. Conf. on Decision and Control, 5640-5645, December 2007.

[6] V. Gupta, "On an Anytime Algorithm for Control," IEEE Int. Conf. on Decision and Control, December 2009.

[7] V. Gupta, "On a Control Algorithm for Time-varying Processor Availability," Hybrid Systems, Control and Computation Conference (HSCC), April 2010.

[8] V. Gupta and N. C. Martins, "On Stability in the Presence of Analog Erasure Channels between Controller and Actuator," IEEE Transactions on Automatic Control, 55(1):175-179, Jan 2010.

[9] D. Henriksson and J. Akesson, "Flexible Implementation of Model Predictive Control using Sub-optimal Solutions," Internal Report No. TFRT-7610-SE, Department of Automatic Control, Lund University, April 2004.

[10] D. Henriksson, A. Cervin, J. Akesson and K. E. Arzen, "On Dynamic Real-Time Scheduling of Model Predictive Controllers," In Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas, NV, December 2002.

[11] H. Ishii, "Limitations in remote stabilization over unreliable channels without acknowledgements," Automatica, 45: 2278-2285, 2009.

[12] Y. Ji, H. J. Chizeck, X. Feng, and K. A. Loparo, "Stability and control of discrete-time jump linear systems," Control Theory Advanced Technology, 7(2): 247-270, 1991.

[13] H. J. Kushner, "Introduction to Stochastic Control," Holt, Rinehart and Winston Inc., New York N.Y.

[14] H. J. Kushner and L. Tobias, "On the stability of randomly sampled systems, IEEE Trans. Automat. Contr., AC-14(4):319324, Aug. 1969.

[15] D. Liu, X. Hu, M.D. Lemmon, and Q. Ling, "Scheduling Tasks with Markov-Chain Constraints," 17th Euromicro Conference on Real-time Systems, July 2005.

[16] L. K. McGovern and E. Feron, "Requirements and Hard Computational Bounds for Real-time Optimization in Safety Critical Control Systems," IEEE Conference on Decision and Control (CDC 98), 1998.

[17] L. K. McGovern and E. Feron, "Closed-loop Stability of Systems Driven by Real-Time Dynamic Optimization Algorithms," IEEE Conference on Decision and Control (CDC 99), 1999.

[18] R. M. Murray, J. Hauser, A. Jadbabaie, M. B. Milam, N. Petit, W. B. Dunbar and R. Franz, "Online control customization via optimization-based control," chapter in "Software-Enabled Control, Information technology for dynamical systems" (eds) T. Samad , G. Balas, 149-174, Wiley-Interscience, 2003.

[19] D. Nesic, A. R. Teel and P. V. Kokotovic, "Sufficient conditions for stabilization of sampled-data nonlinear systems via discrete-time approximations," Sys. Contr. Lett., 38(4-5):259-270, 1999.

[20] V. Nevistic and J. A. Primbs, "Finite Receding Horizon Linear Quadratic Control: A Unifying Theory for Stability and Performance Analysis," California Institute of Technology, Pasadena, CDS Technical Memo CIT-CDS 97-001, January 1997.

[21] D. E. Quevedo and D. Nesic, "On Stochastic Stability of Packetized Predictive Control of Non-linear Systems over Erasure Channels," IFAC Symposium on Non-Linear Control (NOLCOS), 2010, Accepted.

[22] L.Schenato, "To hold or to zero control inputs with lossy links?", IEEE Transactions on Automatic Control, 54(5):1093-, 2009.

[23] P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings, "Suboptimal Model Predictive Control (Feasibility Implies Stability)," IEEE Transactions on Automatic Control, 44(3):648-654, 1999.

[24] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," IEEE Transactions on Automatic Control, 52(9), 1680-1685, September 2007.

[25] X. Wang and M. D. Lemmon, "Self-triggered Feedback Control Systems with Finite-Gain L2 Stability," IEEE Transactions on Automatic Control, 45(3):452-,2009.

[26] L. Xie and L. Xie, "Stability analysis of networked sampled-data linear systems with Markovian packet losses, IEEE Trans. Automat. Contr., 54(6):13751381, June 2009.

[27] T. Zhou, X. Hu and E.H-M. Sha, "A probabilistic performance metric for real-time system design," 7th International Workshop on Hardware-Software Codesign (CODES) (ACM/IEEE), pp. 90-94, May 1999.