# On a Control Lyapunov Function based Anytime Algorithm for Control of Nonlinear Processes [⋆]

**Vijay Gupta** [*] **Daniel E. Quevedo** [**]

[*] *Department of Electrical Engineering, University of Notre Dame, Notre Dame, USA.* vgupta2@nd.edu
[**] *School of Electrical Engineering & Computer Science, The University of Newcastle, NSW 2308, Australia.* dquevedo@ieee.org

**Abstract:** We present an algorithm to calculate control inputs when available processing resources are time-varying. The basic idea is to calculate the control input to decrease a Lyapunov function value as compared to as many time steps in the past as allowed by the system to be controlled and the available processing resources. We analyze the stability of the resulting closed loop system using stochastic Lyapunov functions and indicate, through numerical simulations, that the performance gains obtained can be significant.

*Keywords:* Anytime control, Cyberphysical systems, Stochastic Lyapunov analysis

## 1. INTRODUCTION

A lot of attention has recently been focused on networked and embedded control (see, e.g., the special issue Antsaklis and Baillieul (2007) and the references therein). One key issue which makes networked control system design challenging is the presence of non-ideal communication links. Another aspect, which has been studied less, but plays an important role especially in embedded systems is that of time-varying and limited processing power. As more and more objects are equipped with micro-processors 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 (1998, 1999) presented bounds on computational time for achieving stability for specific optimization algorithms, if the processor has constant, but limited, computational resources. Henriksson and Akesson (2004); Henriksson et al. (2002) 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 (Tabuada (2007); Wang and Lemmon (2009)) as also online sampling (Cervin et al. (2010)), 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 (Caccamo et al. (2002); Cervin et al. (2002); Seto et al. (1996)) 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 (see, e.g., Horvitz (1990); Huang and Cheng (1995); Millan-Lopez et al. (1994); Yoshomito et al. (1992); Zilberstein (1996) for applications in some representative areas). In control, however, there are very few methods available for developing anytime controllers. A notable work is that of Bhattacharya and Balas (2004) who focused on linear processes and controllers, and presented a controller that updated a different number of states depending on the available computational time. However, the available computational time was required to be known to the controller a priori. Another important work is that of Greco et al. (2007), who proposed switching among an existing set of controllers that may require different execution times. In Gupta (2009, 2010), 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 an anytime control algorithm that provides better performance as more processing time is available. The basic idea is to utilize the extra processing

[⋆] The research was supported in part by the National Science Foundation under the grants NSF:0834771 and NSF:0846631 for the first author and under Australian Research Council's Discovery Projects funding scheme (project number DP0988601) for the second author.

time to refine the control input to decrease the Lyapunov function as compared to the value at as many time steps in the past as possible. Thus, the effect of not being able to compute an input at a previous time step can be mitigated. For general nonlinear processes, we analyze stochastic stability of the closed loop system and indicate through numerical simulations that the increase in performance through the proposed algorithm can be significant.

Although similar in some respects to our work in Gupta and Quevedo (2010), this work differs in many important ways. The algorithm in Gupta and Quevedo (2010) is based on utilizing the extra processing time to calculate possible control inputs in the future, similar to a control trajectory in RHC. The effect of the varying processing time appeared as the number of time steps into the future that the control inputs were calculated for. These inputs were stored and used at those time steps where the processing availability was so low that no control input could be calculated. In the present work, however, only one control input is calculated at every time step. The control input is assigned the value 0 to begin with, and the input is refined as more processing time becomes available to ensure that the Lyapunov function value when that input is used decreases as compared to the value further into the past. Thus, the increase in the Lyapunov function value due to the non-calculation of a control input at any time step can be mitigated in the future. Even though the analysis techniques based on stochastic stability are similar in the two works, the fundamental algorithms are, thus, quite different.

The paper is organized as follows: We begin in Section 2 by formulating the problem, and stating the assumptions. In Section 3, we present the proposed algorithm, and analyze the stochastic stability for general nonlinear processes in Section 4. We numerically illustrate the improvement in performance using the proposed algorithm in Section 5.

## 2. 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

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

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 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 system. We will analyze the impact of disturbances and model inaccuracies 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:

(1) 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 (e.g., Liu et al. (2005); Zhou et al. (1999)). 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.
(2) 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., Ji et al. (1991); Kushner (1971)). We are interested in the following definitions: About the equilibrium point $x = 0$, the system (1) is

- *stochastically stable*, if the conditional expectation
$$\mathbb{E}\left[\sum_{k=0}^{\infty} x^T(k)x(k)|x(0)\right] < \infty,$$
where the expectation is taken with respect to the random process $\{\tau(k)\}$.
- *mean square stable*, if the conditional expectation
$$\lim_{k\to\infty} \mathbb{E}\left[x^T(k)x(k)|x(0)\right] < \infty,$$
where the expectation is taken with respect to $\{\tau(k)\}$.

Even with a controller that implements the baseline control law $\kappa(x(k))$ that stabilizes the process if sufficient execution time is available, sporadic control input calculation due to time-varying execution time availability may lead to performance degradation, or even stability loss.

Two streams of work have been proposed to minimize the performance degradation:

(1) The first approach (e.g., Tabuada (2007); Wang and Lemmon (2009)) is to utilize event triggered or time triggered approaches to schedule the calculation of control inputs at the minimal frequency required to maintain stability / performance guarantees. The available execution time at the times when the control input is not calculated may be 'stored' for future use. Alternatively, this approach can be viewed as assuming that the scheduling algorithm at the processor provides sufficient amount of time to calculate the control input, if the input is calculated at a sufficiently low frequency.
(2) The second approach (e.g., Greco et al. (2007); Gupta (2009)) is to design anytime control algorithms. Anytime algorithms are algorithms that progressively refine the solution as more time becomes available. Thus, depending on the execution time available, a different 'quality' of control input is generated. As more execution time becomes available, the control input is refined leading to better performance.

In this paper, we are interested in the second approach. We will propose and analyze an anytime control algorithm that utilizes any extra execution time available to enhance stability and performance of the closed loop system.

*Assumption 2.* As part of the problem formulation, it is also important to specify the value of the control input, which is used if the controller is unable to calculate a new control input $u(k)$ at time $k$. Several choices can be made, including applying zero control, the control input $u(k-1)$, and so on. As Schenato (2009) has pointed out, different choices may be optimal for different plant parameter values. For sake of concreteness, we shall assume that if the new control input cannot be calculated, zero control input ($u(k) = 0$) is applied. Our approach can be easily generalized for other choices of the control inputs, e.g., if the previous control input $u(k-1)$ is held.

## 3. ALGORITHM DESCRIPTION

The algorithm is based on the following basic idea:
The control input aims at decreasing a Lyapunov function of the closed loop system. At some instances, the execution time may be insufficient to calculate such a control input and the Lyapunov function value may increase. The algorithm aims at countering the effect of such increases at the time steps where enough time is available so that a control value can be calculated that decreases the Lyapunov function value as compared to $n$ time steps ago, where $n$ increases as more execution time becomes available. Thus, while *a* control input is available at any level of execution time, the quality of the control input, as measured by the decrease in the Lyapunov function value it achieves, increases as more execution time becomes available.

More formally, we define at every time $k$, a number $N_k$ such that the control $u_k$ is calculated to ensure that $V_{k+1} < \epsilon V_{k+1-N_k}$ (while satisfying the other constraints).

The algorithm proceeds as follows:

*Algorithm* $\mathcal{A}_1$ At every time $k$, do

(1) Set $N_k = 1$. Also set $\bar{u}_k = 0$.
(2) Check if
$$V_{k+1} < \epsilon V_{k+1-N_k},$$
where $0 < \epsilon < 1$ is given, using $u_k = \bar{u}_k$. If so, go to Step 3, else calculate $u_k$ to a value that ensures $V_{k+1} < \epsilon V_{k+1-N_k}$, and set $\bar{u}_k$ to this value.
(3) If more execution time is available, set $N_k = N_k + 1$ and go to Step 2, else set $k = k + 1$ and go to Step 1.

Thus, at the time steps when more processor time is available, a control input that decreases the Lyapunov function compared to more time steps in the past is calculated. Thus, the effect of not being able to calculate a control input at an intermediate step can be counteracted.

*Remark 3.* In the presentation above, there is no bound assumed on $N_k$. If such bounds exists (e.g., due to memory constraints), they can be readily imposed. The analysis presented in Section 4 can also be extended to this case in a straight-forward manner.

*Remark 4.* The algorithm is *anytime* in the sense that a control input value is available at any execution time availability. As more execution time becomes available, the quality of the input is refined. However, note that each of the steps in the algorithm is assumed to be an atomic operation. If any of these steps is interrupted, the last calculated value of $\bar{u}_k$ is used as the control input.

*Remark 5.* The algorithm calculates a control input that guarantees a decrease of the Lyapunov function as compared to past time steps. Alternatively, one could also conceive an algorithm in which a control input trajectory of varying length that reduces the Lyapunov function at *future* time steps (Gupta and Quevedo (2010)).

*Remark 6.* The algorithm does not require knowledge of the probability distribution of the processor time availability. Such descriptions are even allowed to be time varying. However, in the analysis of the algorithm, see Section 4, we will require the execution time $\tau(k)$ to be independent and identically distributed with a given probability mass function.

*Remark 7.* The algorithm implicitly assumes that a control input exists that reduces the Lyapnuov function value as compared to its value $N_k$ time steps ago. In other words, the algorithm assumes that the Lyapunov function along with the system dynamics equation provides a control Lyapunov function. For $N_k = 1$, the control Lyapunov function is $V(f(x(k), u(k)))$. If the control inputs applied from time $j = k - N_k + 1$ to $j = k - 1$ be denoted by $\bar{u}(j)$, then the control Lyapunov function is $V(f(f(f(\cdots f(x(k - N_k + 1), \bar{u}(k - N_k + 1)) \cdots, \bar{u}(k-1)), u(k)))$. Thus, the algorithm requires the existence of a *common control* Lyapunov function going back $N_k$ time steps. This requirement may also impose an upper bound on the possible values of $N_k$.

## 4. STABILITY ANALYSIS

We will next identify 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. In keeping with Assumption 2, if no new control input can be calculated, a zero control input ($u(k) = 0$) is applied. (The arguments can be easily generalized for other choices of the control inputs, e.g., holding the previous control input $u(k-1)$.) Thus, if $p_0$ denotes the probability that the controller is unable to calculate any control input, then the process evolution is similar to that of 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 systems have been derived both for linear systems (Gupta and Martins (2010); Ishii (2009)) and nonlinear systems (Quevedo and Nešić (2010)). Our subsequent analysis follows these works closely.

In the sequel, we make the following assumption:

*Assumption 8.* There exists $1 \leq \alpha < 1/p_0$ such that

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

*Remark 9.* 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 Gupta and Martins (2010).

*Theorem 10.* Consider the baseline control policy $\kappa(x(k))$, the above problem formulation, and suppose that Assumption 8 holds. The process is stable (in both stochastic and mean square sense) if

$$p_0\alpha + (1-p_0)\epsilon < 1. \qquad (3)$$

**Proof.** First we note that $x(k)$ is a Markov process. If we denote the event that a control input is calculated at time $k$ by $\Delta_k = 1$ and the event that the input is not calculated by $\Delta_k = 0$, we can calculate

$$\begin{aligned} E[V_{k+1} - V_k | V_k] &= p_0 E[V_{k+1} - V_k | V_k, \Delta_k = 0] \\ &\quad + (1-p_0) E[V_{k+1} - V_k | V_k, \Delta_k = 1] \\ &\leq p_0(\alpha - 1)V_k + (1-p_0)(\epsilon - 1)V_k \\ &= (p_0\alpha + (1-p_0)\epsilon - 1)V_k. \end{aligned}$$

Thus, if (3) holds, (Kushner, 1971, Chapter 8.4.2, Theorem 2) implies mean square and stochastic stability.

For the proposed algorithm $\mathcal{A}_1$, the stability analysis is more subtle, bearing similarities to that of randomly sampled systems (Kushner and Tobias (1969); Xie and Xie (2009)). 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 $\mathcal{K}$. Furthermore, 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 highest value of $N_k$ for which the controller calculates a control input is $j$ 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 $k$ 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 11.* Consider the algorithm $\mathcal{A}_1$, the above problem formulation, and suppose that Assumption 8 holds. Then

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

where

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

**Proof.** Since the processor time availability is i.i.d., $\Delta_i$ is distributed geometrically as

$$\text{Prob}(\Delta_i = j) = (1-p_0)p_0^{j-1}, \qquad j \in \{1,2,3,\cdots\}.$$

To calculate $E[V_{k_i+1}|x(k_i)]$, we use the total probability formula twice. First, we condition on the maximum value $N$ of $N_{k_i}$ for which the controller calculates a control input at time $k_i$. Thus,

$$\begin{aligned} E[V_{k_i+1}|x(k_i)] &= E[E[V_{k_i+1}|x(k_i), N]] \\ &= \sum_{m=1}^{\infty} p_m E[V_{k_i+1}|x(k_i), N = m]. \qquad (4) \end{aligned}$$

The quantity $E[V_{k_i+1}|x(k_i), N = m]$ can be calculated by conditioning further on $\Delta_{i-1}$. This gives

$$\begin{aligned} E[V_{k_i+1}|x(k_i), N = m] &= E[E[V_{k_i+1}|x(k_i), N = m, \Delta_{i-1}]] \\ &= \sum_{j=1}^{\infty} (1-p_0)p_0^{j-1} E[V_{k_i+1}|x(k_i), N = m, \Delta_{i-1} = j]. \end{aligned}$$

Using Assumption 8 and Step 2 of algorithm $\mathcal{A}_1$, we obtain

$$\begin{aligned} E[V_{k_i+1}|x(k_i), N = m, \Delta_{i-1} = j] \\ = \begin{cases} \epsilon V_{k_{i-1}+1} & j \leq m \\ \alpha^{j-m}\epsilon V_{k_{i-1}+1} & j > m. \end{cases} \end{aligned}$$

Thus,

$$\begin{aligned} E[V_{k_i+1}|x(k_i), N = m] &= (1-p_0)\Big( \sum_{j=1}^{m} p_0^{j-1}\epsilon \\ &\quad + \sum_{j=m+1}^{\infty} p_0^{j-1}\alpha^{j-m}\epsilon \Big) V_{k_{i-1}+1} = \Omega_m V_{k_{i-1}+1}. \end{aligned}$$

Substitution into (4) yields the result. □

Having established Lemma 11, we can analyze the stability conditions when algorithm $\mathcal{A}_1$ is used.

*Theorem 12.* Consider the algorithm $\mathcal{A}_1$ and suppose that Assumption 8 holds. If

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

where the terms $\Omega_m$ have been defined in Lemma 11, then the closed loop system is stochastically stable and mean square stable.

**Proof.** From Lemma 11, if $\Omega < 1$, then $V_{k_i+1}$ is a stochastic Lyapunov function for the closed loop function at the time instants $\{k_i\}$. Moreover, since $\{x(k_i)\}$ is a Markov chain, (Kushner, 1971, Chapter 8.4.2, Theorem 2) implies exponential stability at $k_i \in \mathcal{K}$:

$$E[V_{k_i+1}|x(k_0)] \leq \Omega^i V_{k_0+1}, \qquad i = \{1,2,\cdots\}. \qquad (5)$$

For the time instants $k \notin \mathcal{K}$, i.e., at those time steps when no control input is calculated, we can proceed as follows:

First note that, as in the proof of Lemma 11,

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

$$E[\sum_{l=k_i+1}^{k_{i+1}} V_{l+1}|x(k_i)] =$$

$$(1-p_0)\sum_{j=1}^{\infty} p_0^{j-1} E[\sum_{l=k_i+1}^{k_{i+1}} V_{l+1}|x(k_i), \Delta_i = j].$$

Since $\epsilon < 1 < \alpha$, we bound

$$E[\sum_{l=k_{i+1}}^{k_{i+1}} V_{l+1}|x(k_i), \Delta_i = j] \leq \sum_{l=1}^{j-1} \alpha^l E[V_{k_i+1}|x(k_i)]$$

$$= \frac{\alpha(\alpha^j - 1)}{\alpha - 1} E[V_{k_i+1}|x(k_i)].$$

Thus,

$$E[\sum_{l=k_i+1}^{k_{i+1}} V_{l+1}|x(k_i)]$$

$$\leq \sum_{j=1}^{\infty} \frac{\alpha(1-p_0)p_0^{j-1}(\alpha^{j-1} - 1)}{\alpha - 1} E[V_{k_i+1}|x(k_i)]$$

$$= \beta E[V_{k_i+!}|x(k_i)],$$

where $\beta \triangleq \alpha p_0/(1 - p_0\alpha)$. Using (5), we then obtain

$$E[\sum_{l=k_0+1}^{k_{m+1}} V_{l+1}|x(k_0)] \leq \beta E[\sum_{i=0}^{m} \Omega^i V_{k_0+1}|x(k_0)].$$

Since $0 < \Omega < 1$, the right hand side converges as $m \to \infty$, and $E[\sum_{l=k_0+1}^{\infty} V_{l+1}|x(k_0)] < \infty$.

On the other hand, by assumption, we have that $V(x(k)) \geq |x(k)|$. This implies stochastic stability (and thus mean square stability) at all time instants.

## 5. NUMERICAL EXAMPLES

We next illustrate that the performance gains by using the proposed algorithm can be significant. We first consider the following process, taken from Nešić et al. (1999):

$$x(k+1) = x(k) + 0.2(x^3(k) + u(k)). \qquad (6)$$

The associated baseline control law is given by

$$\kappa(x(k)) = -x^3(k) - x(k)$$

with Lyapunov function $V_k = x^2(k)$.

To evaluate performance, we consider the quadratic cost

$$J = E\left[\sum_{k=0}^{\infty} \left(10x^2(k) + u^2(k)\right)\right], \qquad (7)$$

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 the proposed algorithm $\mathcal{A}_1$, as compared to the baseline algorithm. A Monte Carlo simulation with a total of 500 simulations, each of them lasting 100 time steps, was used to generate the data. The figure shows that a significant improvement in performance can be achieved by using the proposed algorithm.

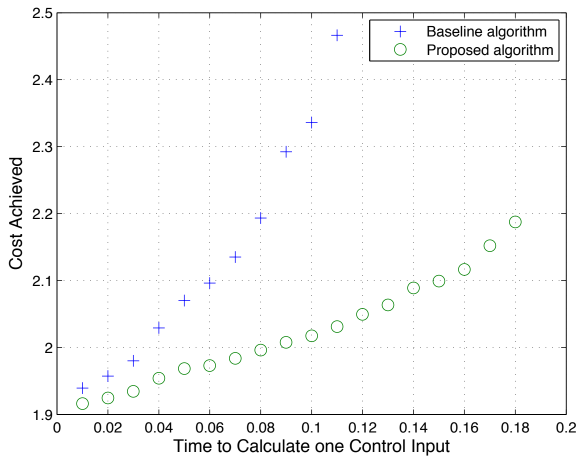

Fig. 1. Cost achieved as a function of execution time required to calculate one control input for (6).

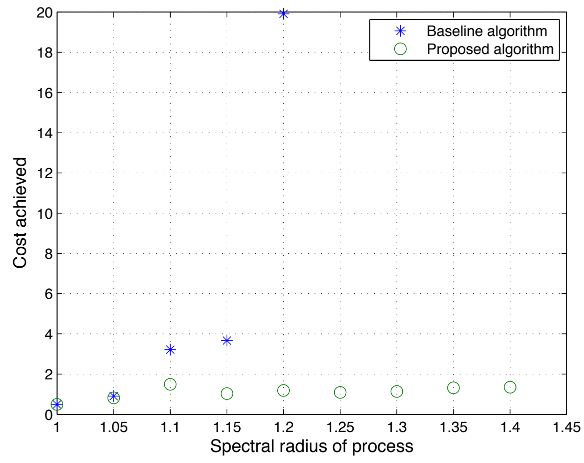

Fig. 2. Cost (7) as a function of the spectral radius of (8).

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

$$x(k+1) = \alpha x(k) + u(k), \qquad (8)$$

with the same cost as in (7) as the scalar $\alpha$ is varied. The execution time availability is the same as in the previous example. The time required for calculation of one control input is assumed to be 0.2. We note that even the stability region can be seen to improve with the proposed algorithm

## 6. CONCLUSIONS AND FUTURE DIRECTIONS

We proposed an anytime control algorithm based on calculating the control input to decrease the Lyapunov function value as compared to as many time steps in the past as allowed by the available processing resources at every time step. Thus, the effect of not being able to calculate the control input at some time steps can be mitigated. We analyzed the stability of the resulting closed loop system using stochastic Lyapunov functions. Simple numerical examples illustrated the performance gain with the proposed algorithm.

This is but a first step towards a more complete theory of anytime control algorithms. We have not yet obtained analytic expressions for performance for general non-linear systems, or considered the effect of model imperfections. Finally, a joint design of the anytime algorithm and a processor scheduler can also be considered.

## REFERENCES

Antsaklis, P. and Baillieul, J. (2007). Special issue on networked control systems. *Proceedings of the IEEE*, 95(1), 9–28.

Bhattacharya, R. and Balas, G.J. (2004). Anytime control algorithms: Model reduction approach. *AIAA Journal of Guidance, Control and Dynamics*, 27(5), 767–776.

Caccamo, M., Buttazzo, G., and Sha, L. (2002). Handling execution overruns in hard real-time control systems. *IEEE Transactions on Computers*, 51(7), 835–849.

Cervin, A., Eker, J., Bernhardsson, B., and Arzen, K.E. (2002). Feedback-feedforward scheduling of control tasks. *Real-Time Systems*, 23(1-2), 25–53.

Cervin, A., Velasco, M., Mart, P., and Camacho, A. (2010). Optimal on-line sampling period assignment: Theory and experiments. *IEEE Transactions on Control Systems Technology*. To appear.

Greco, L., Fontanelli, D., and Bicchi, A. (2007). Almost sure stability of anytime controllers via stochastic scheduling. In *Proceedings of the IEEE Int. Conf. on Decision and Control*, 5640–5645.

Gupta, V. (2009). On an anytime algorithm for control. In *Proceedings of the IEEE Int. Conf. on Decision and Control*.

Gupta, V. (2010). On a control algorithm for time-varying processor availability. In *Proceedings of the Hybrid Systems, Control and Computation Conference (HSCC)*.

Gupta, V. and Martins, N.C. (2010). On stability in the presence of analog erasure channels between controller and actuator. *IEEE Transactions on Automatic Control*, 55(1), 17–179.

Gupta, V. and Quevedo, D.E. (2010). On anytime control of nonlinear processes through calculation of control sequences. In *IEEE Conference on Decision and Control*. Submitted.

Henriksson, D. and Akesson, J. (2004). Flexible implementation of model predictive control using sub-optimal solutions. Technical Report TFRT-7610-SE, Department of Automatic Control, Lund University.

Henriksson, D., Cervin, A., Akesson, J., and Arzen, K.E. (2002). On dynamic real-time scheduling of model predictive controllers. In *Proceedings of the 41st IEEE Conference on Decision and Control*.

Horvitz, E.J. (1990). *Computation and Action under Bounded Resources*. Ph.D. thesis, Department of Computer Science and Medicine, Stanford University.

Huang, X. and Cheng, A.M.K. (1995). Applying imprecise algorithms to real-time image and video transmission. In *Proceedings of the International Conference on Parallel and Distributed Systems*, 96–101.

Ishii, H. (2009). Limitations in remote stabilization over unreliable channels without acknowledgements. *Automatica*, 45, 2278–2285.

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

Kushner, H.J. (1971). *Introduction to Stochastic Control*. Holt, Rinehart and Winston Inc.

Kushner, H.J. and Tobias, L. (1969). On the stability of randomly sampled systems. *IEEE Transactions on Automatic Control*, AC-14(4), 319324.

Liu, D., Hu, X., Lemmon, M., and Ling, Q. (2005). Scheduling tasks with markov-chain constraints. In *Proceedings of the 17th Euromicro Conference on Real-time Systems*.

McGovern, L.K. and Feron, E. (1998). Requirements and hard computational bounds for real-time optimization in safety critical control systems. In *Proceedings of the IEEE Conference on Decision and Control*.

McGovern, L.K. and Feron, E. (1999). Closed-loop stability of systems driven by real-time dynamic optimization algorithms. In *Proceedings of the IEEE Conference on Decision and Control*.

Millan-Lopez, V., Feng, W., and Liu, J.W.S. (1994). Using the imprecise computation technique for congestion control on a real-time traffic switching element. In *Proc. of the International Conference on Parallel and Distributed Systems*, 202–208.

Nešić, D., Teel, A.R., and Kokotović, P.V. (1999). Sufficient conditions for stabilization of sampled-data nonlinear systems via discrete-time approximations. *Systems and Control Letters*, 38(4-5), 259–270.

Quevedo, D.E. and Nešić, D. (2010). On stochastic stability of packetized predictive control of non-linear systems over erasure channels. In *Proc. IFAC Symposium on Nonlinear Control Systems (NOLCOS 2010)*.

Schenato, L. (2009). To hold or to zero control inputs with lossy links? *IEEE Transactions on Automatic Control*, 54(5), 1093–1099.

Seto, D., Lehoczky, J., Sha, L., and Shin, K. (1996). On task schedulability in real-time control system. In *Proc. IEEE Real-Time Systems Symp.*

Tabuada, P. (2007). Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9), 1680–1685.

Wang, X. and Lemmon, M.D. (2009). Self-triggered feedback control systems with finite-gain l2 stability. *IEEE Transactions on Automatic Control*, 45(3).

Xie, L. and Xie, L. (2009). Stability analysis of networked sampled-data linear systems with markovian packet losses. *IEEE Transactions on Automatic Control*, 54(6), 13751381.

Yoshomito, H., Arita, D., and Taniguchi, R. (1992). Real-time communication for distributed vision processing based on imprecise computation model. In *Proceedings of International Parallel and Distributed Processing Symposium*, 128–133.

Zhou, T., Hu, X., and Sha, E.M. (1999). A probabilistic performance metric for real-time system design. In *Proc. of the 7th International Workshop on Hardware-Software Codesign (CODES) (ACM/IEEE)*, 90–94.

Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *Artificial Intelligence Magazine*, 17(3), 73–83.