2012 IEEE International Conference on Control Applications (CCA)
Part of 2012 IEEE Multi-Conference on Systems and Control
October 3-5, 2012. Dubrovnik, Croatia

Optimal Self-Triggering for Nonlinear Systems via Approximate Dynamic Programming

Domagoj Tolić, Rafael Fierro and Silvia Ferrari

Abstract— **In this paper we investigate optimal intermittent feedback for nonlinear control systems. Using the currently available measurements from a plant, we develop a methodology that outputs when to update the controller with new measurements such that a given cost function is minimized. Our cost function captures trade-offs between the performance and energy consumption of the control system. The optimization problem is formulated as a Dynamic Programming problem, and Approximate Dynamic Programming is employed to solve it. Instead of advocating a particular approximation architecture for Approximate Dynamic Programming, we formulate properties that successful approximation architectures satisfy. In addition, we consider problems with partially observable states, and propose Particle Filtering to deal with partially observable states and intermittent feedback. Finally, our approach is applied to a mobile robot trajectory tracking problem.**

I. Introduction

Recent years have witnessed an increasing interest in *event-triggered* implementations of control laws. Many works, such as [1], [2], [3], [4], [5] and [6], replace the traditional periodic paradigm, where the up-to-date information is transmitted and control laws are executed in a periodic fashion, with the event-triggered paradigm. In the event-triggered paradigm, one defines a desired performance and sampling (i.e., transmission of the up-to-date information) is triggered when an event, called a *triggering event*, representing the unwanted performance occurs. A variant of event-triggering, known as *self-triggering*, uses the current sampling instance to predict and preclude an occurrence of the triggering event (refer to [1], [2] and [6]). In order to simplify this presentation and improve readability, we refer to all these paradigms simply as *intermittent feedback*. Intermittent feedback is motivated by the rational use of expensive resources at disposition in an effort to decrease energy consumption, processing and sensing requirements.

At the moment, the research community is interested in extending intersampling intervals as much as possible without taking into account a deterioration in the performance due to intermittent feedback. In applications where energy consumption for using sensors, transmitting the obtained information, and executing control laws is relatively inexpensive compared to the slower convergence and excessive use of control power, extending intersampling intervals is not desirable. For instance, think of an airplane driven by an autopilot system designed to follow the shortest path between two points. Any deviation from the shortest path caused by intermittent feedback increases overall fuel consumption. This increase in fuel consumption is probably more costly than the cost of energy saved due to intermittent feedback. In this paper, we encode these energy consumption trade-offs in a cost function, and design an Approximate Dynamic Programming (ADP) approach that yields optimal intertransmission intervals with respect to the cost function.

The main contributions of this paper are threefold: a) formulation of the optimal self-triggering problem as a Dynamic Programming (DP) problem; b) employment of Particle Filters (PFs) fed by intermittent feedback to account for partially observable states; and c) formulation of properties that successful approximation architectures in ADP approaches satisfy. To the best of our knowledge, the problem of optimal intermittent feedback has yet to be addressed.

Similar problems to the problem considered herein are discussed in [7] and [8]. The authors in [7] balance control performance versus network cost by choosing the appropriate time delay-controller pair. The work in [8] investigates optimal control of hybrid systems based on ideas from dynamic and convex programming. While [7] associates costs with each of time delay-controller pairs, the work in [8] associates costs with switches between controllers. The optimization methods from [7] and [8] boil down to optimal control of switching systems (refer to [9] and [10]).

Motivated by [10], we adopt ADP (see [11] and [12]) as the strategy for tackling our problem. ADP is a set of methods for solving sequential decision-making problems under uncertainty by alleviating the computational burden of the infamous *curses of dimensionality* in DP [13]. In theory, DP solves a wide spectrum of optimization problems providing an optimal solution. In practice, straightforward implementations of DP algorithms are deemed computationally intractable for most of the applications. Therefore the need for efficient ADP methods. However, comprehensive analyses and performance guarantees of these approximate methods are still unresolved (except in very special settings), and present a critical area of research.

The rest of the paper is organized as follows. Section II presents the problem of optimal intermittent feedback and assumptions under which the problem is solved. The methodology brought together to solve the problem is presented in Section III. The proposed methodology is verified on a trajectory tracking controller in Section IV. Conclusions are drawn and future challenges are discussed in Section V.

This work was supported by NSF grants ECCS #1027775 and ECCS #1028506.

D. Tolić and R. Fierro are with MARHES Lab, Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131–0001, USA, {dtolic}@ece.unm.edu. S. Ferrari is with LISC, Department of Mechanical Engineering, Duke University, Durham, NC 27708-0005, USA.

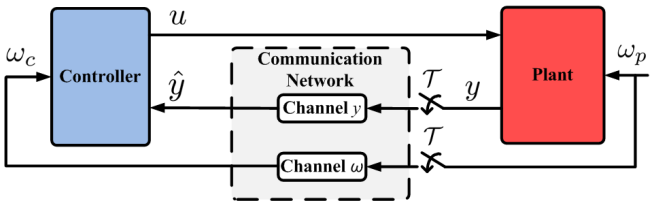


Fig. 1. Diagram of a plant and controller with discrete transmission instants and communication channels giving rise to *intermittent feedback*.

II. Problem Statement and Assumptions

Consider a time-invariant nonlinear feedback control system consisting of a plant

$$\dot{x}_p = f_p(x_p, u, \omega_p), \quad y = g_p(x_p), \tag{1}$$

and a controller

$$\dot{x}_c = f_c(x_c, y, \omega_c), \quad u = g_c(x_c), \tag{2}$$

where $x_p \in \mathbb{R}^{n_p}$ and $x_c \in \mathbb{R}^{n_c}$ are the states, $y \in \mathbb{R}^{n_y}$ and $u \in \mathbb{R}^{n_u}$ are the outputs, and $\omega_p \in \mathbb{R}^{n_{\omega_p}}$ and $\omega_c \in \mathbb{R}^{n_{\omega_c}}$ are the external/exogenous inputs or disturbances of the plant and controller, respectively. Notice that y is the input of the controller, and u is the input of the plant. Let us denote the compound state of the closed-loop systems (1) and (2) by $x = (x_p, x_c)$ where $x \in \mathbb{R}^{n_x}$.

In order to account for the intermittent knowledge of y and ω_p by the controller, we model the links between the plant and controller as communication networks that cause intermittent exchange of information. More precisely, we introduce the output error vector e as follows:

$$e(t) := \hat{y}(t) - y(t) \tag{3}$$

where $\hat{y}$ is an estimate of y performed from the perspective of the controller, and the input error vector e_ω as follows:

$$e_\omega(t) := \hat{\omega}_p(t) - \omega_p(t) \tag{4}$$

where $\hat{\omega}_p$ is an estimate of ω_p from the perspective of the controller. For the sake of simplicity, we take $\hat{y}$ and $\hat{\omega}_p$ to be the most recently communicated values (or transmitted measurements) of the output and external input of the plant, i.e., we use the zero-order-hold strategy. Now we introduce $\mathcal{T} := \{t_i : i \in \mathbb{N}_0\}$ as the set of time instants when outputs and externals inputs of the plant are transmitted over communication networks. Finally, many control laws are designed such that $\omega_c = \hat{\omega}_p$. Examples are trajectory tracking controllers as in [4], [5] and [14]. An illustration of a control system indicating the communication channels that cause intermittent information is provided in Figure 1.

Next, we want to minimize the following cost function $V : \mathbb{R}^{n_x} \to \mathbb{R}$ that captures *performance vs. energy* trade-offs

$$V_{\tau_i}(x_0) = \mathop{\mathbb{E}}_{e_\omega}\left\{ \sum_{i=1}^{\infty} \gamma^i \Big[\underbrace{\int_{t_{i-1}}^{t_i} (x_p^T Q x_p + u^T R u)\mathrm{d}t}_{l(x_p,u,\tau_i)} + S \Big] \right\} \tag{5}$$

over all sampling policies τ_i and for all initial conditions $x_0 \in \mathbb{R}^{n_x}$. In addition, $\gamma \in (0,1)$ is a discount factor that makes the sum (5) finite provided that $l(x_p, u, \tau_i)$ is bounded over all $[t_{i-1}, t_i]$ where $t_i \in \mathcal{T}$ and

$$t_i = t_{i-1} + \tau_{i-1}. \tag{6}$$

For clarity, we use τ_i instead of $\tau(\hat{y}(t_i), \hat{\omega}_p(t_i))$, but one has to keep in mind that, in general, intersampling intervals τ_i's depend on the most recently transmitted information from the plant, i.e., on $\hat{y}(t_i)$ and $\hat{\omega}_p(t_i)$. In addition, Q and R are positive definite matrices, and a nonnegative constant S represents the cost incurred for sampling y and ω_p, transmitting $\hat{y}$ and $\hat{\omega}_p$, and updating the control signal u. In (5), the conditional expectation over a stochastic signal e_ω is denoted $\mathop{\mathbb{E}}_{e_\omega}$.

The main problem considered herein can now be stated:

Problem 1: For the system (1) and (2) with values of $\hat{\omega}_p$ and $\hat{y}$ received at t_i, $i \in \mathbb{N}_0$, find time intervals τ_i's until the next transmission instants such that (5) is minimized.

We solve the above problem under the following assumptions:

Assumption 1: $\hat{y}$ is corrupted by measurement noise.

Assumption 2: $\hat{\omega}_p$ is corrupted by measurement noise, and ω_p is arbitrary between two consecutive t_i's.

Due to these assumptions, we have to deal with partially observable states (see Subsection III-E for more details).

III. Methodology

This section presents the tools brought together to solve Problem 1 under Assumptions 1 and 2. Starting from the input-output-triggering that provides maximal stabilizing intersampling intervals $\tau_i^{\max}$'s, we find optimal τ_i^*'s for the cost function (5) resorting to ADP and PF.

A. Input-Output-Triggering via the Small-Gain Theorem

Building on the small-gain theorem, we develop *input-output-triggering* in [15] (the work from [15] is partially published in [16]). In other words, based on the currently available but outdated measurements of the outputs and external inputs of a plant, a simple expression for when to obtain new up-to-date measurements and execute the control law is provided. The details of [15] are out of scope of this paper and are not needed in order to follow the rest of the paper. In fact, with slight modifications, our ADP approach is applicable to any self-triggered sampling policy (e.g., [1], [2], and [6]). Essentially, self-triggered sampling policies output maximal allowable intersampling intervals $\tau_i^{\max}$'s that provably yield stable closed-loop system (1) and (2). Starting from these $\tau_i^{\max}$'s, the work presented herein finds

$$\tau_i^* \in [0, \tau_i^{\max}], \quad i \in \mathbb{N}_0 \tag{7}$$

that minimize (5). Because we know the upper bounds $\tau_i^{\max}$'s of stabilizing sampling policies, τ_i^*'s obtained in this paper provably stabilize the plant. A comprehensive treatment of the problem whether ADP solutions of optimal problems yield stability can be found in [17].

B. Dynamic Programming

Notice that the cost function (5) has the standard DP form. Let us now introduce a state transition function f that maps $x(t_{i-1})$, $u(t_{i-1})$ and $\hat{\omega}_p(t_{i-1})$ to $x(t_i)$ given some e_ω over $[t_{i-1}, t_i]$, i.e.,

$$x(t_i) = f(x(t_{i-1}), u(t_{i-1}), \tau_{i-1}, \hat{\omega}_p(t_{i-1}), e_\omega). \quad (8)$$

Due to intermittent feedback and presence of nonlinearities in the plant and controller, the state transition function over τ_i's, in general, cannot be given in a closed form with, for example, a difference equation [18]. This is a typical impediment one faces when analyzing nonlinear systems under intermittent feedback. Therefore, in general, the state transition function (8) needs to be simulated using (1), (2), $\hat{y}(t_{i-1})$, $\hat{\omega}_p(t_{i-1})$ and e_ω over time horizon τ_{i-1}.

Next, let us assume that e_ω is a stationary stochastic process. Consequently, since we consider the infinite horizon problem (5) and a time-invariant control system (1) and (2), τ_i is not a function of t_i. Hence, we simply write τ instead of τ_i in the rest of the paper. Solving the DP problem of minimizing (5) backwards through time is combinatorially impossible since the state space x in (5) is uncountable. Therefore, we write the stochastic control problem of minimizing (5) over τ in its equivalent form known as the Bellman equation

$$V^*(z) = \inf_{\tau \in [0, \tau^{\max}]} \Big(l(z, u, \tau) + \\ + \gamma \mathop{\mathbb{E}}_{e_\omega} \{ V^*(f(z, u, \tau, \hat{\omega}_p, e_\omega)) \} \Big) \quad (9)$$

where $V^*(z)$ is called the optimal value function (or optimal cost-to-go function), and represents the cost incurred by an optimal policy τ^* when the initial condition in (5) is z. It is well known that V^* is the unique fixed point of (9). Therefore, the problem of minimizing (5) boils down to finding V^* in (9).

For notational convenience, we introduce the Bellman operator $\mathcal{M}$ as

$$\mathcal{M}g = (\mathcal{M}g)(z) = \inf_{\tau \in [0, \tau^{\max}]} \Big(l(z, u, \tau) + \\ + \gamma \mathop{\mathbb{E}}_{e_\omega} \{ g(f(z, u, \tau, \hat{\omega}_p, e_\omega)) \} \Big) \quad (10)$$

for any $g : \mathbb{R}^{n_x} \to \mathbb{R}$. Since $\gamma \in (0, 1)$, it can be shown that $\mathcal{M}$ is a contraction, i.e.,

$$\|\mathcal{M}u - \mathcal{M}v\|_s \leq \gamma \|u - v\|_s \quad (11)$$

where $\|v\|_s = \sup_{z \in \mathbb{R}^{n_x}} v(z)$. The set $\mathcal{B}$ of all bounded, real valued functions with the norm $\|\cdot\|_s$ is a Banach space. Therefore, for each initial $V^0 \in \mathcal{B}$, the sequence of value functions $V^{n+1} = \mathcal{M}V^n = \mathcal{M}^{n+1}V^0$ converges to V^*.

Two remarks are in order. First, it can be shown that the problem of finding an optimal τ^* for each state in (9) is non-convex. However, since τ is confined to a rather small compact set $[0, \tau^{\max}]$, we utilize gradient search methods with constraints from different initial points in order to obtain τ^*. Second, the conditional expectation $\mathop{\mathbb{E}}_{e_\omega}$ in (9) can be obtained in a closed form only for special cases. Otherwise, it can be calculated numerically by replacing the integral with a sum using a quadrature approximation. In Section IV, we use the Simpson formula [19].

Lastly, due to the "curses of dimensionality", solving (9) for $V^*(z)$ or iterating an initial V^0 is deemed intractable for most of the problems of interest; hence, we employ ADP in the next subsection where our goal is to find an approximation $\hat{V}^*$ of V^*.

C. Approximate Dynamic Programming

Among a number of methods in ADP, we choose the Value Iteration (VI) method for its simplicity and a wide spectrum of applications. Notice that $\mathcal{B}$ is an infinite dimensional vector space, meaning that it takes infinitely many parameters to describe V^*. Therefore, one introduces an approximate value function $\hat{V}^i$ of V^i where $i \in \mathbb{N}_0$. Approximate value functions $\hat{V}^i$, $i \in \mathbb{N}_0$, can be represented in finite parameter approximation architectures such as neural networks (NNs). Note that it is not possible to obtain true value functions V^i's but only their approximations; hence, we write $\hat{V}^i$ instead of V^i. Basically, VI performs

$$\hat{V}^{i+1} = \mathcal{M}\hat{V}^i, \quad i \in \mathbb{N}_0 \quad (12)$$

until

$$\|\hat{V}^{i+1} - \hat{V}^i\|_s < \epsilon \quad (13)$$

where $\epsilon > 0$.

In order to calculate $\hat{V}^{i+1}$ in (12), we need to apply (10) over all $z \in \mathbb{R}^{n_x}$. Obviously, this is computationally impossible since $\mathbb{R}^{n_x}$ contains uncountably many points. Therefore, many ADP approaches focus on a compact subset $\mathcal{C}_x \subset \mathbb{R}^{n_x}$, choose a finite set of points $\mathcal{X} \subset \mathcal{C}_x$, and calculate $\hat{V}^{i+1}$ only for the points in $\mathcal{X}$. Afterwards, the values of $\hat{V}^{i+1}$ for $\mathcal{C}_x \setminus \mathcal{X}$ are obtained via some kind of interpolation/generalization.

D. Approximation Architecture

The problem of choosing an approximation architecture that fits $\hat{V}^{i+1}$ to $\hat{V}^{i+1}(\mathcal{X})$ and, at the same time, is able to interpolate/generalize for $\hat{V}^{i+1}(\mathcal{C}_x \setminus \mathcal{X})$ appears to be crucial in order for ADP to converge. It is considered that ADP in not converging when either the stopping criterion (13) is never reached (refer to [20] and [21]) or $\hat{V}^*$ is not an accurate approximation of V^* [22]. The latter criterion is concerned with suboptimality of the obtained solution. In this paper, we focus on the former deferring suboptimality analyses for future work.

The key property that has to be preserved by an approximation architecture is the contraction property (11) (refer to [20], [21] and [23]). In [20], the author classifies function approximators as expansion or contraction approximators. *Expansion approximators*, such as linear regressors and NNs, exaggerate changes on $\mathcal{C}_x \setminus \mathcal{X}$. *Contraction approximators* (or local averagers), such as k-nearest-neighbor, linear interpolation, grid methods and other state aggregations methods, conservatively respond to changes in $\mathcal{X}$. Therefore, on the one

hand, a VI that includes a contraction approximator always converges, in the sense of (13), to the fixed point determined by the approximator, say $\hat{V}^*_{ca}$. However, not much can be said about the value $\|V^* - \hat{V}^*_{ca}\|_s$ (see [20] and [21]). On the other hand, a VI that includes an expansion approximator might diverge [21]. However, NNs are still a widely used approximation architecture due to their notable successes (for example, [24], [25], and [26]), adaptive architectures [27], performance guarantees under certain assumptions [23], and inventions of novel NN architectures. These novel NN architectures are also called nonparametric approximation [23] and they adapt to the training data. Examples are kernel-based NNs (refer to [28] and [29]) and recurrent NNs (refer to [30] and [29]).

A goal of this paper is not to advocate certain architectures. Instead, based on our experience and the references above, we define properties that successful approximation architectures possess (e.g., contraction approximators and kernel-based NNs). Based on the specifics of the problem (dimensionality of the problem, availability and density of data, available processing power, memory requirements, etc.), one should choose a suitable architecture.

Desired Properties: Assume that $V^*(x)$ is a smooth function on $\mathcal{C}_x$, and choose a smooth function approximator. At the i^{th} step, where $i \in \mathbb{N}_0$, randomly pick any $x' \in \mathcal{C}_x$, calculate $(\mathcal{M}\hat{V}^i)(x')$, and fit $\hat{V}^i(x')$ to $(\mathcal{M}\hat{V}^i)(x')$ obtaining $\hat{V}^{i+1}$. We are seeking an approximation architecture that satisfies the following properties

(i) $\hat{V}^{i+1}(x') = (\mathcal{M}\hat{V}^i)(x')$;

(ii) $\text{supp}(\hat{V}^{i+1} - \hat{V}^i) = \mathcal{C}_i$, where $\text{supp}(f) = \{x : f(x) \neq 0\}$ is the support of a function f, and $\mathcal{C}_i \subset \mathcal{C}_x$ is a convex and compact neighborhood of x; and

(iii) for any $c \in \partial\mathcal{C}_i$, where $\partial\mathcal{C}_i$ denotes the boundary of $\mathcal{C}_i$, the following holds

$$\hat{V}^{i+1}[S] \subseteq [\hat{V}^{i+1}(c), \hat{V}^{i+1}(x')], \tag{14}$$

where $\hat{V}^{i+1}[S]$ is the image of the segment $\mathcal{S}$ connecting x' and c,

in order to have $\|V^{i+1} - V^i\|_s \to 0$ as $i \to \infty$.

Remark 1: Let us consider two value functions $\hat{u}^i$ and $\hat{v}^i$ in the i^{th} step, and apply $\mathcal{M}$ at a randomly chosen x'_i. Due to (11), we have $\|(\mathcal{M}\hat{u}^i)(x'_i) - (\mathcal{M}\hat{v}^i)(x'_i)\| \leq \gamma\|\hat{u}^i(x'_i) - \hat{v}^i(x'_i)\|$. From property (i), we conclude that $\|\hat{u}^{i+1}(x'_i) - \hat{v}^{i+1}(x'_i)\| \leq \gamma\|\hat{u}^i(x'_i) - \hat{v}^i(x'_i)\|$. Since the approximator is smooth, we know that there exists a neighborhood $\mathcal{C}'_i \subseteq \mathcal{C}_i$ of x'_i such that $\sup_{x\in\mathcal{C}'_i}\|\hat{u}^{i+1}(x) - \hat{v}^{i+1}(x)\| \leq \sup_{x\in\mathcal{C}'_i}\|\hat{u}^i(x) - \hat{v}^i(x)\|$. This means that the nonexpansion property required in [20] is obtained locally around x'_i. The nonexpansion property from [20] is (11) when γ is replaced with 1. Finally, property (iii) eliminates counterexamples in which the Lebesgue measures of $\mathcal{C}'_i$, $i \in \mathbb{N}_0$, tend to zero. Consequently, generalization of the approximation architecture is ensured.

Remark 2: Property (i) is the accuracy requirement in order to preserve (11). Property (ii) is the "local property" found in [20], [21] and [27]. This local property is built in the activation functions of the kernel-based NNs. Property (iii) is used to ensure that $\mathcal{C}'_i$'s are not merely x'_i's. In addition, property (iii) curbs expansiveness on $\mathcal{C}_i \setminus \mathcal{C}'_i$.

Remark 3: Notice that *Desired Properties* imply online learning of NNs [29]. The motivation behind this choice lies in the fact that it is straightforward to check properties (i), (ii) and (iii) in online learning. Moreover, since we randomly pick points $x'_i \in \mathcal{C}_x$ in each step, we do not have to specify $\mathcal{X}$. By choosing random x'_i's, we also avoid the problem of *exploration vs. exploitation* [13]. On the other side, when using batch learning, properties (i), (ii) and (iii) cannot be guaranteed since NNs are expansion approximators. In fact, not until we switched to online learning in the example from Section IV, convergence was obtained. An extension of *Desired Properties* for batch learning and the problem of choosing $\mathcal{X}$ are left for the future work.

Remark 4: As the stopping criterion we use the following: when $\|\hat{V}^{i+1} - \hat{V}^i\|_s < \epsilon$ for $N \in \mathbb{N}$ consecutive steps, the value iteration method has converged.

E. Partially Observable States

Notice that the approximate value function $\hat{V}(x)$ is a function of state x. Up to this point we did not take into account that x is not available due to Assumptions 1 and 2. In other words, we are solving the DP problem (5) with partially observable states. More details about strategies for solving DP problems with partially observable states are found in Chapter 5 of [11].

Let us assume that the controller can access its state x_c. Consequently, the controller can calculate u at any given time. However, the controller does not have access to the state of the plant x_p but merely to $\hat{\omega}_p$ and $\hat{y}$. We circumvent this problem by introducing a particle filter that provides estimates $\hat{x}_p$ of the actual state x_p.

More precisely, we model the closed-loop system (1) and (2) as

$$\begin{aligned} x_p(t_i) &= f_p^d(x_p(t_{i-1}), u(t_{i-1}), \tau_{i-1}, \hat{\omega}_p(t_{i-1}), e_\omega), \\ \hat{y}(t_i) &= g(x_p(t_i), \nu), \end{aligned} \tag{15}$$

where f_p^d represents a discrete transition function of the plant obtained in similar fashion as (8), and statistics of the process noise e_ω and measurement noise ν are known and time invariant. Based on (15), we build a particle filter that extracts $\hat{x}_p$ from $\hat{\omega}_p$ and $\hat{y}$ and feeds the controller. Details of our particle filter implementation under intermittent feedback can be found in [31] and [32].

We deal with partially observable states by first obtaining $\hat{V}^*(x)$ for the case of perfect state information. Then, we employ particle filtering and iterate $\hat{V}^*(x)$ using (12) to obtain the approximation of $\hat{V}^*(\hat{x})$ for the case of partially observable states. Basically, since $\hat{V}^*(x)$ is a close estimate of $\hat{V}^*(\hat{x})$ when $\hat{x}$ is a close estimate of x, we exploit $\hat{V}^*(x)$ to fine tune $\hat{V}^*(\hat{x})$.

IV. Case Study - Trajectory Tracking

In this section, we apply the optimal self-triggered sampling policy to the *trajectory tracking* controller presented in

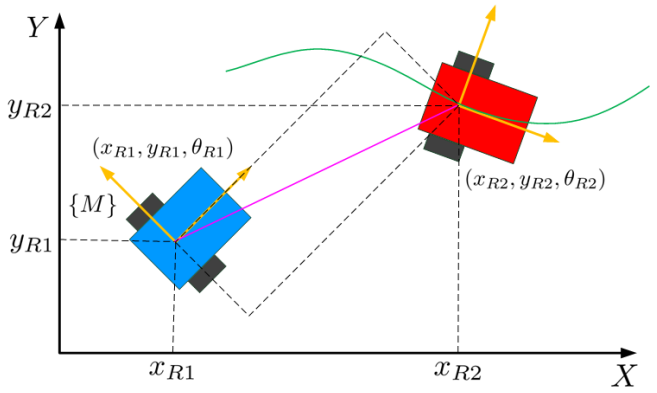


Fig. 2. Illustration of the trajectory tracking problem considered in this paper.

[14]. In [14], a velocity-controlled unicycle robot R_1 given by

$$\dot{x}_{R1} = v_{R1}\cos\theta_{R1},\ \dot{y}_{R1} = v_{R1}\sin\theta_{R1},\ \dot{\theta}_{R1} = \omega_{R1} \quad (16)$$

tracks a trajectory generated by a virtual velocity-controlled unicycle robot R_2 with states $(x_{R2}, y_{R2}, \theta_{R2})$, and linear and angular velocities v_{R2} and ω_{R2}, respectively. See Figure 2 for an illustration. The tracking error x_p in the coordinate frame $\{M\}$ of robot R_1 becomes

$$x_p = \begin{bmatrix} \cos\theta_{R1} & \sin\theta_{R1} & 0 \\ -\sin\theta_{R1} & \cos\theta_{R1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{R2} - x_{R1} \\ y_{R2} - y_{R1} \\ \theta_{R2} - \theta_{R1} \end{bmatrix}. \quad (17)$$

Applying the following control law

$$\begin{aligned} v_{R1} &= v_{R2}\cos x_3 + k_1 x_1, \\ \omega_{R1} &= \omega_{R2} + k_2 v_{R2}\frac{\sin x_3}{x_3}x_2 + k_3 x_3 \end{aligned} \quad (18)$$

where k_1, k_2 and k_3 are positive control gains, Proposition 3.1 in [14] shows that the control law (18) makes the origin $x_p = [0\ 0\ 0]^T$ globally asymptotically stable provided that $v_{R2}(t), \omega_{R2}(t)$ and their derivatives are bounded for all times t and $\lim_{t\to\infty} v_{R2}(t) \neq 0$ or $\lim_{t\to\infty} \omega_{R2}(t) \neq 0$.

Since the controller (18) is not a dynamic controller, we have that $x = x_p$. Next, for the sake of simplicity, we use the following measurement model: $\hat{y}(t_i) = x(t_i) + \nu$. The external input is $\omega_p = [v_{R2}\ \ \omega_{R2}]^T$. When emulating noise in (15), we use $e_\omega \in U([-0.3, 0.3] \times [-0.3, 0.3])$ and $\nu \in U([-0.15, 0.15] \times [-0.15, 0.15] \times [-0.15, 0.15])$ where $U(\mathcal{S})$ denotes the uniform distribution over a compact set $\mathcal{S}$.

The following coefficients were used in the cost function (5): $Q = 0.1I_3$, $R = 0.1I_2$, $S = 15$ and $\gamma = 0.96$ where I_n is the $n \times n$ identity matrix and $n \in \mathbb{N}$. A remark is in order regarding the choice of Q, R and S. On the one hand, as we decrease S and keep Q and R fixed, the obtained sampling policy τ approaches zero. On the other hand, as S becomes greater, τ approaches τ^{max}. The above choice of Q, R and S yields $\tau \in [0.6\tau^{max}, 0.9\tau^{max}]$.

As the approximation architecture we choose a Multilayer Perceptron (MLP) with 100 hidden neurons. In addition, we confine x to the set $\mathcal{C}_x = [-100, 100]^2 \times [-30\pi, 30\pi]$.

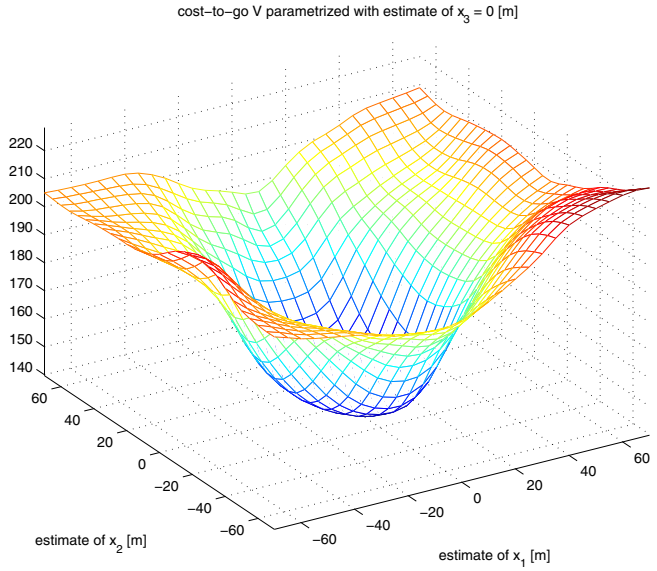


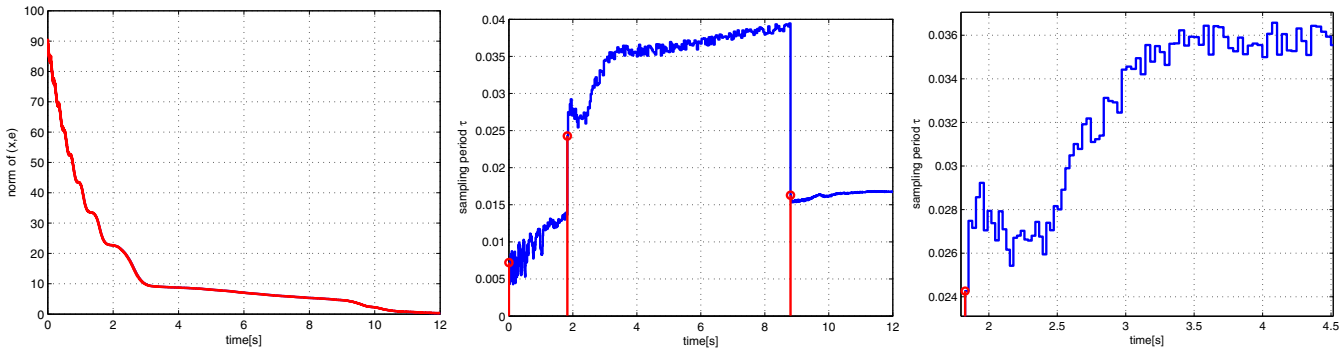
Fig. 3. Approximation $\hat{V}^*(\hat{x})$ of the optimal value function $V^*(x)$ for $\omega_p = (1, 1)$ depicted as a function of $\hat{x}_1 \in [-70, 70]$ and $\hat{x}_2 \in [-70, 70]$ when $\hat{x}_3 = 0$.

Not until we used that many hidden neurons, properties (i), (ii) and (iii) were satisfied on $\mathcal{C}_x$. Even though activation functions in MLPs are not locally responsive, we were able to satisfy (i), (ii) and (iii). We presume the reason is low dimensionality of the considered tracking problem. For high dimensional problems, the kernel-based NNs appear to be more suitable. In the stopping criterion from Remark 4 we choose $\epsilon = 1$ and $N = 10$, and obtain $\hat{V}^*(x)$ in about 300 to 400 steps depending on the initial $\hat{V}^0(x)$ and ω_p. Afterwards, we obtain $\hat{V}^*(\hat{x})$ from $\hat{V}^*(x)$ using (12) and $\hat{x}$ fed from the particle filter. With $\epsilon = 1$ and $N = 10$, it takes about 50 simulations for $\hat{V}^*(\hat{x})$ to converge starting from $\hat{V}^*(x)$. The obtained approximation $\hat{V}^*(\hat{x})$ of $V^*(x)$ for $\omega_p = (1, 1)$ is illustrated in Figure 3.

In the simulation included in this paper, we choose $k_1 = 1.5$, $k_2 = 1.2$ and $k_3 = 1.1$. Figure 4 is obtained for the trajectory generated with $\omega_p(t) = (1, 1)_{[0,1.83)} + (0.6, 0.15)_{[1.83,8.8)} + (2, 2)_{[8.8,12]}$ where $t_\mathcal{S}$ is the indicator function on a set $\mathcal{S}$, i.e., $t_\mathcal{S} = t$ when $t \in \mathcal{S}$ and zero otherwise.

V. Conclusions

This paper investigates the problem of optimal input-output-triggering for nonlinear systems. We replace the traditional periodic paradigm, where up-to-date information is transmitted and control laws are executed in a periodic fashion, with optimal intermittent feedback. In other words, we develop a methodology that, based on the currently available but outdated measurements of the outputs and external inputs of a plant, provides time instants when to obtain new up-to-date measurements and execute the control law such that a given cost function is minimized. The optimization problem is formulated as a DP problem, and ADP is employed to solve it. In addition, because the investigated problems contain partially observable states, our methodology includes



(a) (b) (c)

Fig. 4. Illustration of the optimal input-output-triggering: (a) Norm of (x, e); (b) Values of sampling period τ_i between two consecutive transmissions. Red stems indicate time instants when changes in ω_p happen; and, (c) A detail from Figure 4(b).

Particle Filtering under intermittent feedback. Furthermore, instead of advocating one approximation architecture over another in ADP, we formulate properties that successful approximation architectures satisfy. Finally, our approach is successfully applied to a trajectory tracking controller for velocity-controlled unicycles.

In the future, the main goal is to further investigate the properties of successful approximation architectures. In addition, we plan to estimate how suboptimal the methodology developed in this paper is.

REFERENCES

[1] A. Anta and P. Tabuada, "To sample or not to sample: Self-triggered control for nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2030 – 2042, September 2010.

[2] M. Lemmon, *Event-triggered Feedback in Control, Estimation, and Optimization*, ser. Lecture Notes in Control and Information Sciences, A. Bemporad, M. Heemels, and M. Johansson, Eds. Springer Verlag, 2010, vol. 405.

[3] H. Yu and P. Antsaklis, "Event-triggered real-time scheduling for stabilization of passive and output feedback passive systems," in *Proceedings of the American Control Conference*, San Francisco, CA, June-July 2011, pp. 1674 – 1679.

[4] P. Tallapragada and N. Chopra, "On event triggered trajectory tracking for control affine nonlinear systems," in *Proceedings of the IEEE Conference on Decision and Control*, December 2011, pp. 5377–5382.

[5] D. Tolić and R. Fierro, "Stability of feedback linearization under intermittent information: A target-pursuit case," in *American Control Conf.*, San Francisco, CA, 2011, pp. 3184 – 3190.

[6] C. Nowzari and J. Cortés, "Self-triggered coordination of robotic networks for optimal deployment," in *American Control Conf.*, San Francisco, CA, June-July 2011, pp. 1039–1044.

[7] S. Hirche, C. Chen, and M. Buss, "Performance oriented control over networks switching controllers and switched time delay," *Asian Journal of Control*, vol. 10, no. 1, pp. 24–33, 2008.

[8] S. Hedlund and A. Rantzer, "Optimal control of hybrid systems," in *Conference on Decision and Control*, 1999, pp. 3972–3977.

[9] S. C. Bengea and R. A. DeCarlo, "Optimal control of switching systems," *Automatica*, vol. 41, no. 1, pp. 11–27, 2005.

[10] X. Xuping and P. Antsaklis, "Optimal control of switched systems based on parameterization of the switching instants," *IEEE Trans. on Automatic Control*, vol. 49, no. 1, pp. 2–16, January 2004.

[11] D. P. Bertsekas, *Dynamic Programming and optimal control, Vol. I*, 3rd ed. Belmont, Massachusetts: Athena Scientific, 2005.

[12] ——, *Dynamic Programming and optimal control, Vol. II*, 3rd ed. Belmont, Massachusetts: Athena Scientific, 2007.

[13] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, ser. Wiley Series in Probability and Statistics. Hoboken, NJ: John Wiley and Sons, Inc., 2007.

[14] C. C. de Wit, H. Khennouf, C. Samson, and O. Sordalen, "Nonlinear control design for mobile robots," *Recent Trends in Mobile Robots*, pp. 121–156, 1993.

[15] D. Tolić, R. Fierro, and R. G. Sanfelice, "Input-output-triggering in nonlinear systems: A small gain theorem approach," in preparation.

[16] D. Tolić, R. G. Sanfelice, and R. Fierro, "Self-triggering in nonlinear systems: A small gain theorem approach," in *20th Mediterranean Conference on Control and Automation*, July 2012, to appear.

[17] W. Zhang, "Controller synthesis for switched systems using approximate dynamic programming," Ph.D. dissertation, Purdue University, Indiana, December 2009.

[18] D. Nešić, A. Teel, and P. Kokotović, "Sufficient conditions for stabilization of sampled-data nonlinear systems via discrete-time approximations," *Sys. and Cont. Letters*, vol. 38, no. 4-5, pp. 259–270, 1999.

[19] P. J. Davis and P. Rabinowitz, *Methods of numerical integration*, 2nd ed. Mineola, NY: Dover Publications, Inc., 2007.

[20] G. J. Gordon, *Stable function approximation in dynamic programming*, School of Computer Science, Carnegie Mellon University, 1995, technical report.

[21] J. M. Lee, N. S. Kaisare, and J. H. Lee, "Choice of approximator and design of penalty function for an approximate dynamic programming based control approach," *Journal of Process Control*, vol. 16, no. 2, pp. 135 – 156, February 2006.

[22] B. O'Donoghue, Y. Wang, and S. Boyd, "Min-max approximate dynamic programming," in *IEEE Multi-Conference on Systems and Control*, Denver, CO, September 2011, pp. 424–431.

[23] L. Busoniu, R. Babuska, B. D. Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*, ser. Automation and Control Engineering Series. CRC Press, 2010.

[24] A. Samuels, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[25] G. Tesauro, "Neurogammon: a neural network backgammon program," in *IJNN Proceedings III*, 1990, pp. 33–39.

[26] L. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3-4, pp. 293–322, 1992.

[27] P. Vamplew and R. Ollington, "Global versus local constructive function approximation for on-line reinforcement learning," in *Proceedings of the 18th Australian Joint conference on Advances in Artificial Intelligence*, ser. AI'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 113–122.

[28] B. M. Bethke, "Kernel-based approximate dynamic programming using bellman residual elimination," Ph.D. dissertation, MIT, Massachusetts, February 2010.

[29] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Prentice Hall, November 2008.

[30] I. Szita, "Rewarding excursions: Extending reinforcement learning to complex domains," Ph.D. dissertation, Eötvös Loránd University, Budapest, Hungary, March 2007.

[31] D. Tolić and R. Fierro, "Adaptive sampling for tracking in pursuit-evasion games," in *IEEE Multi-Conference on Systems and Control*, Denver, CO, September 2011, pp. 179 –184.

[32] ——, *A Comparison of a Curve Fitting Tracking Filter and Conventional Filters under Intermittent Information*, Department of Electrical and Computer Engineering, University of New Mexico, October 2010, technical report. [Online]. Available: http://hdl.handle.net/1928/11424