Systems & Control Letters 79 (2015) 59–67

Contents lists available at ScienceDirect

Systems & Control Letters

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

Resource-aware MPC for constrained nonlinear systems: A self-triggered control approach

T.M.P. Gommans *, W.P.M.H. Heemels

Eindhoven University of Technology, Department of Mechanical Engineering, Control Systems Technology Group, The Netherlands

ARTICLE INFO

Article history:
Received 16 May 2014
Received in revised form
27 January 2015
Accepted 5 March 2015
Available online 10 April 2015

Keywords:
Self-triggered control
Model predictive control
Sparse control
Networked control systems

ABSTRACT

In systems with resource constraints, such as actuation limitations or limited communication bandwidth, it is desired to obtain control signals that are either sparse or sporadically changing in time to reduce resource utilization. In this paper, we propose a resource-aware self-triggered MPC strategy for discrete-time nonlinear systems subject to state and input constraints that has three important features: Firstly, significant reductions in resource utilization can be realized without modifying the cost function by input regularization or explicitly penalizing resource usage. Secondly, the control laws and triggering mechanisms are synthesized so that a priori chosen performance levels (in terms of the original cost function) are guaranteed by design next to asymptotic stability and constraint satisfaction. Thirdly, we address the co-design problem of jointly designing the feedback law and the triggering condition. By means of numerical examples, we show the effectiveness of this novel strategy.

1. Introduction

In more and more control applications, it becomes essential to explicitly address resource constraints in the implementation and the design of the control law. Due to constraints on a system's actuation resources it may be desirable to have control signals that are either sparse or sporadically changing in time. The use of sparse or sporadically changing actuation signals can have several benefits, such as improved fuel efficiency or a larger lifetime of actuators subject to wear and tear. For instance, in [1] sparse thrust actuation signals are obtained to use fuel in the control of a spacecraft in an efficient manner, and in [2] sporadically changing actuation is used in the control of an autonomous underwater vehicle in order to decrease fuel consumption and increase the deployment time. Also in overactuated systems, it can be desirable to have a smart control allocation policy that does not require each actuator to be updated continuously in order to realize a desired level of performance. One specific example includes the usage of sparse actuation signals to control the roll of an overactuated ship [3]. Next to systems with constrained actuation resources, also in the area of networked control systems (NCSs) [4] with scarce communication resources there is an interest in control signals that are sparse or sporadically changing in time, depending on whether the control value is set to zero or held to the previous value, respectively, when no new information is received by the actuators, see, e.g., [5]. In case the communication in NCSs is wireless, a further resource constraint is often induced by battery-powered sensors. Indeed, the radio chip, used to communicate sensor data required to update the control law, is the primary source of energy consumption in the sensors [6]. Hence, reducing the number of times the control law needs to be executed in NCSs leads to lower network resource utilization and enhanced lifetimes of the battery-powered devices.

Given the above motivations, in this paper, we are particularly interested in resource-aware control strategies for discrete-time nonlinear systems that have scarce (actuation and/or communication) resources, and, moreover, are subject to state and input *constraints*. One of the most widely used techniques for control of constrained systems is model predictive control (MPC). In fact, several works already exist in the literature addressing the control of constrained systems with scarce resources. The most well known approach is based on modifying the MPC problem by appending the original MPC control cost with an ℓ_1 penalty on the input in order to obtain sparse input signals. Regularizing by the ℓ_1-norm is known to induce sparsity in the sense that, individual components of the input signal will be equal to zero, see, e.g., [3,7–9], in which the focus is on discrete-time *linear* systems. Also different types of sum-of-norms regularization can be used to obtain so-called group sparsity, meaning that at many

* Corresponding author.
E-mail address: t.m.p.gommans@tue.nl (T.M.P. Gommans).

http://dx.doi.org/10.1016/j.sysconle.2015.03.003
0167-6911/

time instances the entire input vector becomes zero, see, e.g., [10]. Although a regularization penalty on the input is effective in obtaining sparse solutions, due to the altered cost function no performance guarantees are given in terms of the original MPC cost function.

Considering the NCS literature, two main approaches oriented towards generating sporadically changing input signals (to save computation and communication resources) can be distinguished, namely, event-triggered control (ETC), see, e.g., [11–17], and self-triggered control (STC), see, e.g., [18–24], see also [25] for a recent overview. In ETC and STC, the control law consists of two elements being a feedback controller that computes the control input, and a triggering mechanism that determines when the control input has to be updated. The difference between ETC and STC is that in the former the triggering consists of verifying a specific condition continuously and when it becomes true, the control task is triggered, while in the latter at an update time the next update time is pre-computed. In the context of NCSs, another popular approach to reduce utilization of communication resources (without resorting to sporadically changing input signals) in networks with messages with a high payload is packet-based predictive control in which a sequence of inputs and/or states is sent to the actuators (instead of a single input value), see, e.g., [26–28]. In fact, there are several works available that combine ideas from ETC and packet-based predictive control, see, e.g., [29–31]. In [29] an event-triggered MPC setup for unconstrained systems was presented using tools from input-to-state stability (ISS). Extensions of this work were presented in [30] in which ETC schemes for constrained discrete-time systems were proposed. In particular, the optimal control sequence coming from the MPC optimization problem is applied in an open-loop fashion between the event times. In [31], another ETC scheme based on using the sequence of optimal controls obtained from the MPC optimization problem is presented, for discrete-time linear systems. In addition to sending the computed MPC input sequence to the actuators, in [31], also the predicated optimal state (or output) sequence is transmitted to the sensors. An event-triggering mechanism, positioned at the sensors, is chosen such that it detects if the true states deviate from the predicted states in the MPC problem, using absolute thresholds. Note that in [29–31] the control law and the triggering condition are designed independently, and also that no a priori performance guarantees on the (original) control cost are provided.

The ideas in [29–31] are appealing for NCS applications, but they still require the actuators to be updated in a periodic manner (at every time step), which is undesirable in control applications with constrained actuation resources. In addition the schemes in [29–31] would lead to bursts of communication at certain points in time, which one may like to avoid, especially in networks with messages with a low payload. To ameliorate this communication burden, [9] proposes a packet-based predictive control scheme that combines a classical quadratic MPC cost function with ℓ_1 input regularization and sends the obtained sparse input sequences to the actuators. A STC approach for unconstrained discrete-time linear systems that avoids bursts in communication and frequent updating of the actuators by sending only a single control value and keeping this constant until the next execution time is presented in [32]. The strategy in [32] solves a co-design problem of simultaneously designing the control law and the triggering condition and allows trading in control performance to obtain lower (communication and/or actuation) resource utilization. In particular, [32] follows an ℓ_1-like regularization to solve the co-design problem by augmenting the quadratic cost function related to control performance with a penalty related to sampling the system (and updating the control law). Due to the additional penalties in [9,32], no performance guarantees are given in terms of the original MPC cost function. In [33], a scheme with the same purpose as in [32] is considered, also for unconstrained discrete-time linear systems. Instead of sending a single control value, [33] employs also an optimization-based approach using rollout techniques for minimizing a desired quadratic cost function with an explicit constraint on the average sampling/actuation rate. However, note that [9,32,33] only apply to *unconstrained* discrete-time linear systems. All these works focus on discrete-time systems, but note that also in the context of continuous-time NMPC event-triggered implementations have been proposed in [34,35].

This paper provides an extension of our preliminary works [24,36]. In [24], we proposed a self-triggered control scheme with an a priori guaranteed level of performance, in terms of quadratic cost, for discrete-time linear *unconstrained* systems, which was extended in [36] to a self-triggered MPC scheme that is capable of handling state and input constraints for *linear* systems. In this paper, we propose a general framework for the self-triggered MPC strategy applying to discrete-time *nonlinear* systems subject to state and input constraints and a possibly *non-quadratic* cost function. Moreover, in [24,36], the focus is on obtaining sporadically changing input profiles, whereas this paper shows also how the framework can be extended to include the case where sparse input profiles are desired. The proposed self-triggered control law possesses three important features: (i) significant reductions in resource utilization are obtained, (ii) a priori closed-loop performance guarantees are provided (by design) in terms of the original cost function, next to asymptotic stability and constraint satisfaction, (iii) co-design of both the feedback law and triggering condition is achieved. To elaborate on these features, we emphasize that the proposed STC approach reduces resource utilization without modifying the cost function by input regularization or explicitly penalizing resource usage, thereby being essentially different from [3,7–10,32]. This feature enables that the proposed STC strategy will not only realize stabilization towards a desired equilibrium, but it also provides a priori guarantees on an infinite horizon cost directly related to the MPC cost (feature (ii)). More specifically, one can choose a priori a desired suboptimal level of control performance, and the controller design will automatically take this into account, while at the same time aiming for a reduction of the network and actuation resources (feature (i)). Regarding feature (iii), we note that most existing design methods for ETC and STC are emulation-based in the sense that the feedback controller is designed assuming a standard time-triggered implementation, and thus not taking the eventual event- or self-triggered implementation into account. The triggering mechanism in then designed in a subsequent phase (in which the controller is already fixed). Since the feedback controller is designed before the triggering mechanism, it is difficult, if not impossible, to obtain an optimal design of the combined feedback controller and the triggering mechanism in the sense that the minimum number of controller executions is achieved while guaranteeing stability and a certain level of closed-loop performance. In this paper, we provide a synthesis technique that determines the next execution time of the algorithm and the applied control value *simultaneously*, thereby selecting control values that are optimized for not being updated for a maximal number of steps. The presented self-triggered MPC framework offers the flexibility to configure it for limited actuator applications (inducing the control input to be sparse) or networked control applications (by, for instance, keeping the control input constant as much as possible) by selecting proper admissible control sequences in the MPC optimal control problem. This also further extends our results in the preliminary paper [36], where only the latter case was considered. Finally, compared to packet-based solutions, note that in our solution bursts in communications can

be avoided as it is the objective of the control strategy to send just a *single* control value and not a sequence of predicted future values to the actuators.

The remainder of the paper is organized as follows. After indicating the notational conventions used in this paper, Section 2 the problem formulation is presented where a stabilizing MPC setup is described, which forms the basis for developing the self-triggered MPC strategy. The proposed self-triggered approach is discussed in Section 3, while Section 4 contains the implementation considerations. The effectiveness of the proposed approach is demonstrated by means of numerical examples in Section 5. Finally, Section 6 presents the conclusions.

1.1. Nomenclature

Let $\mathbb{R}$ and $\mathbb{N}$ denote the set of real numbers and the set of non-negative integers (including zero), respectively. For $s, t \in \mathbb{N}$, the notation $\mathbb{N}_{\geqslant s}$, $\mathbb{N}_{[s,t)}$, $\mathbb{N}_{(s,t)}$ and $\mathbb{N}_{[s,t]}$ is used to denote the sets $\{r \in \mathbb{N} \mid r \geqslant s\}$, $\{r \in \mathbb{N} \mid s \leqslant r < t\}$, $\{r \in \mathbb{N} \mid s < r < t\}$ and $\{r \in \mathbb{N} \mid s \leqslant r \leqslant t\}$, respectively. Sequences of vectors are indicated by bold letters, e.g., $\mathbf{u} = (u_0, u_1, \ldots, u_M)$ with $u_i \in \mathbb{R}^{n_u}$, $i \in \{0, 1, \ldots, M\}$, where $M \in \mathbb{N} \cup \{\infty\}$ will be clear from the context. A function $\alpha : \mathbb{R}_{\geqslant 0} \to \mathbb{R}_{\geqslant 0}$ is a $\mathcal{K}$-function if it is continuous, strictly increasing and $\alpha(0) = 0$. The empty set is denoted by $\emptyset$.

2. Problem formulation

In this paper we consider a discrete-time nonlinear system of the form

$$x_{t+1} = g(x_t, u_t), \tag{1}$$

where $x_t \in \mathbb{R}^{n_x}$ and $u_t \in \mathbb{R}^{n_u}$ are the state and the input, respectively, at time $t \in \mathbb{N}$. We assume that $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is continuous and that $g(0, 0) = 0$, indicating that 0 is an equilibrium for (1) under zero input. The system (1) is subject to input and state constraints given by

$$u_t \in \mathbb{U} \quad \text{and} \quad x_t \in \mathbb{X}, \quad t \in \mathbb{N}, \tag{2}$$

where $\mathbb{X} \subseteq \mathbb{R}^{n_x}$ and $\mathbb{U} \subseteq \mathbb{R}^{n_u}$ are convex and compact sets containing the origin in their interiors. For $N \in \mathbb{N}$, $x_k(x, \mathbf{u})$ denotes the solution to (1) at time $k \in \mathbb{N}_{[0,N]}$ initialized at $x_0 = x$ and with control input sequence given by $\mathbf{u} = (u_0, u_1, \ldots, u_{N-1})$.

2.1. Standard stabilizing MPC setup

For the system (1), we now consider the MPC setup given by the following optimization problem. For a fixed prediction horizon $N \in \mathbb{N}_{\geqslant 1}$, given state $x_t = x \in \mathbb{X}$ at time $t \in \mathbb{N}$,

$$\min J_N(x, \mathbf{u}) := F(x_N(x, \mathbf{u})) + \sum_{k=0}^{N-1} L(x_k(x, \mathbf{u}), u_k) \tag{3a}$$

$$\text{w.r.t. } \mathbf{u} \in \mathcal{U}_N(x) := \Big\{ \mathbf{u} \in \mathbb{U}^N \;\Big|\; \forall k \in \{1, 2, \ldots, N-1\}, \; x_k(x, \mathbf{u}) \in \mathbb{X} \text{ and } x_N(x, \mathbf{u}) \in \mathbb{X}_T \Big\}. \tag{3b}$$

Here, $\mathbb{X}_T \subseteq \mathbb{X}$ is the terminal set, which is assumed to be closed and to contain the origin in its interior. The stage cost is given by $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}_{\geqslant 0}$, where $L(0, 0) = 0$, and the terminal cost is given by $F : \mathbb{R}^{n_x} \to \mathbb{R}_{\geqslant 0}$, where $F(0) = 0$. We assume that L and F are continuous functions.

A state $x \in \mathbb{X}$ is called feasible for the optimization problem (3) if there exists at least one admissible input sequence, i.e., $\mathcal{U}_N(x) \neq \emptyset$. The set of feasible states is denoted by $\mathbb{X}_f$, i.e., $\mathbb{X}_f = \{x \in \mathbb{X} \mid \mathcal{U}_N(x) \neq \emptyset\}$. As N is finite, the given conditions ensure the existence of a minimum for the optimization problem (3). For $x \in \mathbb{X}_f$, $V(x)$ denotes the corresponding minimum value for the optimization problem (3). Hence, $V : \mathbb{X}_f \to \mathbb{R}_{\geqslant 0}$ is the MPC value function given by

$$V(x) := \min_{\mathbf{u} \in \mathcal{U}_N(x)} J_N(x, \mathbf{u}). \tag{4}$$

For any state $x \in \mathbb{X}_f$, there exists a unique optimal sequence of control updates denoted by $\mathbf{u}^*(x) = (u_0^*(x), \ldots, u_{N-1}^*(x))$. Hence,

$$J_N(x, \mathbf{u}^*(x)) = \min_{\mathbf{u} \in \mathcal{U}_N(x)} J_N(x, \mathbf{u}). \tag{5}$$

The resulting MPC law $u^{mpc} : \mathbb{X}_f \to \mathbb{U}$ is now defined as

$$u^{mpc}(x) := u_0^*(x), \tag{6}$$

which is implemented in a receding horizon fashion as

$$u_t = u^{mpc}(x_t), \quad t \in \mathbb{N}. \tag{7}$$

To guarantee recursive feasibility and closed-loop stability in the above setup we will use the terminal set and cost method, see, e.g., [37], for which we require the following assumptions throughout the remainder of the paper.

Assumption 1. There exists $K : \mathbb{X}_T \to \mathbb{R}^{n_u}$ such that for $x \in \mathbb{X}_T$

$$F(g(x, K(x))) - F(x) \leqslant -L(x, K(x)), \tag{8a}$$

$$g(x, K(x)) \in \mathbb{X}_T, \tag{8b}$$

$$K(x) \in \mathbb{U}. \tag{8c}$$

Moreover, there exist $\alpha_1, \alpha_2 \in \mathcal{K}$ such that

$$L(x, u) \geqslant \alpha_1(\|x\|), \quad \text{for all } x \in \mathbb{X}_f, u \in \mathbb{U}, \tag{8d}$$

$$F(x) \leqslant \alpha_2(\|x\|), \quad \text{for all } x \in \mathbb{X}_T. \tag{8e}$$

Under these standing assumptions, the optimal MPC law as in (7) results in a closed-loop system given by (1) and (7), which

(i) is asymptotically stable in $\mathbb{X}_f$, in the sense that
- [well-posedness]: for each $x_0 \in \mathbb{X}_f$ the corresponding trajectory x_t to (1) and (7) exists for all $t \in \mathbb{N}$;
- [Lyapunov stability]: for any $\varepsilon > 0$ there is a $\delta > 0$ such that if $x_0 \in \mathbb{X}_f$ and $\|x_0\| \leqslant \delta$ then the corresponding trajectory x_t to (1) and (7) satisfies $\|x_t\| \leqslant \varepsilon$;
- [attractivity]: for any $x_0 \in \mathbb{X}_f$ the corresponding trajectory x_t to (1) and (7) satisfies $\lim_{t\to\infty} x_t = 0$.

(ii) satisfies input and state constraints, i.e., (2).

(iii) leads to performance guarantees of the form

$$\sum_{t=0}^{\infty} L(x_t, u_t) \leqslant V(x_0). \tag{9}$$

See [37] and the references therein, for the proofs and further details on the terminal set and cost method.

2.2. Resource-aware MPC with guaranteed performance

Clearly for all $x_0 \in \mathbb{X}_f$, the implementation of (7) requires the computation of the control value $u^{mpc}(x_t)$ and uses communication and/or actuation resources at *each* time instant $t \in \mathbb{N}$. This is undesirable in systems with limited resources, where it may be preferable to use input profiles that reduce the required resource utilization. As already indicated in the introduction, such resource-aware input profiles are typically sparse in case actuation resources are limited or sporadically changing in case communication resources are limited.

The main focus of this paper is to synthesize control laws that take scarceness of the available (actuation and/or communication)

resources into account, but still guarantee certain performance requirements in terms of the infinite horizon cost as in (9), next to closed-loop asymptotic stability in $\mathbb{X}_f$ as in (i) and constraint satisfaction as in (ii). In particular, regarding the former, we aim at obtaining resource-aware input signals while still satisfying the performance guarantee

$$\sum_{t=0}^{\infty} L(x_t, u_t) \leqslant \beta V(x_0), \tag{10}$$

where a performance degradation index $\beta \geqslant 1$ is introduced to balance the degree of sub-optimality and the resource utilization.

Remark 1. Although we focus on the MPC setup discussed in this section, the proposed self-triggered strategy can be based on any asymptotically stabilizing MPC setup that is recursively feasible such that for each $x \in \mathbb{X}_f$ there exists $u \in \mathbb{U}$ such that $V(g(x,u)) - V(x) \leqslant -L(x,u)$ and $g(x,u) \in \mathbb{X}_f$.

3. Self-triggered strategy

In this section we present a so-called self-triggered approach to solve the problem formulated in Section 2.2, focusing on the case of sparsity. In Section 3.2, we present variants of the self-triggered approach based on other input profiles relevant for certain type of applications (e.g., requiring sporadically changing input profiles) and in Section 3.3 we derive various important theoretical properties.

3.1. Approach and algorithm for self-triggered MPC

To introduce the approach, consider the set of execution times $\{t_l \mid l \in \mathbb{N}\} \subseteq \mathbb{N}$ of the control algorithm that satisfies $t_{l+1} > t_l$, for all $l \in \mathbb{N}$, and $t_0 = 0$, and the corresponding control signal

$$u_t = \begin{cases} \bar{u}_l, & \text{if } t = t_l, \\ 0, & \text{if } t \in \mathbb{N}_{[t_l+1, t_{l+1})}, \end{cases} \tag{11}$$

where $\bar{u}_l \in \mathbb{U}$, $l \in \mathbb{N}$. At an execution time t_l the goal is to choose both the next control value $\bar{u}_l$ and the next execution time $t_{l+1} > t_l$. In fact, t_{l+1} will be selected as large as possible while still leading to the guarantee (10). Note that there is no actuation (and also no computation and communication) on times $t_l+1, t_l+2, \ldots, t_{l+1}-1$. Instrumental in guaranteeing (10) will be the dissipation-like inequality

$$\sum_{t=t_l}^{t_{l+1}-1} L(x_t, u_t) \leqslant \beta \left[V(x_{t_l}) - V(x_{t_{l+1}}) \right] \tag{12}$$

$$\text{subject to } \begin{cases} u_t = \begin{cases} \bar{u}_l \in \mathbb{U}, & \text{if } t = t_l, \\ 0, & \text{if } t \in \mathbb{N}_{[t_l+1, t_{l+1})}, \end{cases} \\ x_t \in \mathbb{X}, \quad t \in \{t_l+1, t_l+2, \ldots, t_{l+1}-1\}, \\ x_{t_{l+1}} \in \mathbb{X}_f, \end{cases}$$

as we will see. In particular, at execution time t_l with state x_{t_l} the aim will now be to maximize t_{l+1} such that there exists a $\bar{u}_l$ for which the conditions in (12) are feasible. To formalize this setup, the notation

$$\bar{\mathcal{U}}_M(x) := \Big\{ \mathbf{u} \in \mathbb{U}^M \mid \forall i \in \mathbb{N}_{[1,M-1]}, \ \big(\Pi_i \mathbf{u} = 0 \text{ and } x_i(x, \mathbf{u}) \in \mathbb{X} \big), \text{ and } x_M(x, \mathbf{u}) \in \mathbb{X}_f \Big\} \tag{13}$$

is introduced, in which the projection operators $\Pi_i : (\mathbb{R}^{n_u})^M \to \mathbb{R}^{n_u}$, $i \in \mathbb{N}_{[0,M-1]}$, are given by $\Pi_i \mathbf{u} = u_i$ for $\mathbf{u} = (u_0, u_1, \ldots, u_{M-1})$ with $u_i \in \mathbb{R}^{n_u}$, $i \in \mathbb{N}_{[0,M-1]}$. Hence, the constraints in (13) specify, amongst others, that $\mathbf{u} \in \bar{\mathcal{U}}_M(x)$ is of the form $(\bar{u}, 0, \ldots, 0)$ for some $\bar{u} \in \mathbb{U}$. An example of $\mathbf{u} \in \bar{\mathcal{U}}_M(x)$ is shown in Fig. 1(a).

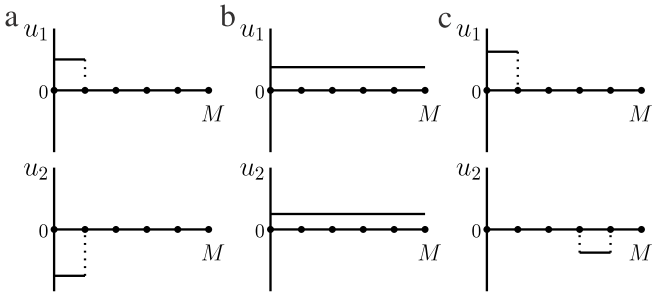

Fig. 1. Examples of: (a) $\mathbf{u} \in \bar{\mathcal{U}}_M(x)$, (b) $\mathbf{u} \in \bar{\mathcal{U}}_M(x)$, (c) $\mathbf{u} \in \hat{\mathcal{U}}_M(x)$.

Using the notation in (13) and letting $M_l := t_{l+1} - t_l$ and $x_{t_l} = x$, inequality (12) can be rewritten as

$$\sum_{k=0}^{M_l-1} L(x_k(x, \mathbf{u}), \Pi_k \mathbf{u}) \leqslant \beta \left[V(x) - V(x_{M_l}(x, \mathbf{u})) \right] \tag{14}$$

$$\text{subject to } \mathbf{u} \in \bar{\mathcal{U}}_{M_l}(x),$$

where we used the time-invariance of both the system dynamics (1) and the constraints (2) to apply a time shift. Hence, at time t_l with state $x_{t_l} = x$ we aim at maximizing $M_l \in \mathbb{N}$, say $M_l^* \in \mathbb{N}$, such that (14) is still feasible for some $\mathbf{u} \in \bar{\mathcal{U}}_{M_l^*}(x)$. Naturally, $t_{l+1} = t_l + M_l^*$ and u_t in (11) is then taken as $u_t = \Pi_{t-t_l} \mathbf{u}$, $t \in \mathbb{N}_{[t_l, t_{l+1})}$. To express this formally, we define for $x \in \mathbb{X}_f$ and fixed $M_l = M$,

$$\mathcal{U}_M^{st}(x) := \Big\{ \mathbf{u} \in \bar{\mathcal{U}}_M(x) \;\Big|\; (16) \text{ holds} \Big\}, \tag{15}$$

where

$$\sum_{k=0}^{M-1} L(x_k(x, \mathbf{u}), \Pi_k \mathbf{u}) \leqslant \beta \left[V(x) - V(x_M(x, \mathbf{u})) \right]. \tag{16}$$

Based on the above considerations, the following self-triggered MPC algorithm is proposed, where we use an integer $\bar{M} \in \mathbb{N}_{\geqslant 1}$ as a predefined upperbound on the inter-execution times. The upperbound $\bar{M}$ allows to limit the maximal time allowed between two consecutive execution times. As a consequence, it limits the number of optimization problems that are to be solved in evaluating Algorithm 1, on which we elaborate further in Section 4. In this way, the computational resources required for evaluating Algorithm 1 can be appropriately adapted, e.g., based on the available hardware.

Algorithm 1. At execution time $t_l \in \mathbb{N}$, $l \in \mathbb{N}$, with corresponding state $x_{t_l} = x$, the next update time t_{l+1} and the corresponding control signal for $t \in \mathbb{N}_{[t_l, t_{l+1})}$ are given by

$$t_{l+1} = t_l + \mathcal{M}^{st}(x), \tag{17a}$$

$$u_t = \Pi_{t-t_l} \mathbf{u}, \ t \in \mathbb{N}_{[t_l, t_{l+1})} \quad \text{for some } \mathbf{u} \in \mathcal{U}^{st}(x), \tag{17b}$$

where

$$\mathcal{M}^{st}(x) := \max\{M \in \{1, 2, \ldots, \bar{M}\} \mid \mathcal{U}_M^{st}(x) \neq \emptyset\} \tag{18}$$

and

$$\mathcal{U}^{st}(x) := \mathcal{U}^{st}_{\mathcal{M}^{st}(x)}(x). \tag{19}$$

Here, $\mathcal{M}^{st} : \mathbb{R}^{n_x} \to \{1, 2, \ldots, \bar{M}\}$ and $\mathcal{U}^{st} : \mathbb{R}^{n_x} \to \mathbb{U}^{\mathcal{M}^{st}(x)}$ denote the time between two executions and the set of possible control sequences, respectively. Note that only $\Pi_0 \mathbf{u}$ with $\mathbf{u} \in \mathcal{U}^{st}(x)$ has

to be transmitted to the actuators, as afterwards only zero inputs are implemented until a new control value is received.

3.2. Variants of self-triggered strategy

The STC strategy proposed in the previous section, produces solutions with input profiles that have so-called group sparsity property [38] in the sense that $u_t = 0,\ t = \mathbb{N}_{[t_l+1,t_{l+1})}$, see Fig. 1(a). In different applications it can be preferable to have other profiles of the input than enforced by $\mathbf{u} \in \bar{\mathcal{U}}_M(x)$ in (15). Our framework is general in the sense that it offers the flexibility to adopt such different input profiles. To make this more clear, consider NCS-type of applications, in which it is sometimes desirable to avoid frequent (continuous) changes of the control input. Hence, it is desirable to keep the inputs constant between two execution times. The approach presented in Section 3 can include this easily by replacing $\bar{\mathcal{U}}_M(x)$ in (15) by

$$\tilde{\mathcal{U}}_M(x) := \Big\{\mathbf{u} \in \mathbb{U}^M \mid \exists \bar{u} \in \mathbb{U},\ \forall i \in \mathbb{N}_{[0,M-1]},\ \Pi_i\mathbf{u} = \bar{u},\ \forall i \in \mathbb{N}_{[1,M-1]},\ x_i(x,\mathbf{u}) \in \mathbb{X} \text{ and } x_M(x,\mathbf{u}) \in \mathbb{X}_f\Big\}. \quad (20)$$

The constraints in (20) specify, amongst others, that $\mathbf{u} \in \tilde{\mathcal{U}}_M(x)$ is of the form $(\bar{u}, \bar{u}, \ldots, \bar{u})$, for some $\bar{u} \in \mathbb{U}$, as depicted in Fig. 1(b). Note that, for fixed $M \in \mathbb{N}$, the strategy based on $\mathbf{u} \in \tilde{\mathcal{U}}_M(x)$ requires the same amount of data to be transmitted as $\mathbf{u} \in \bar{\mathcal{U}}_M(x)$. The only difference is that in the former the actuators perform a "to hold" action, while in the latter a "to zero" strategy [5] is followed.

Both the strategies based on $\bar{\mathcal{U}}_M(x)$ and $\tilde{\mathcal{U}}_M(x)$ have n_u free control variables, but all elements of the input vector are updated simultaneously. A strategy leading to sparse solutions (instead of group sparse solutions), i.e., allowing each actuator to actuate independently but sporadically, can be obtained by replacing $\bar{\mathcal{U}}_M(x)$ in (15) by

$$\hat{\mathcal{U}}_M(x) = \Big\{\mathbf{u} \in \mathbb{U}^M \mid \forall j \in \mathbb{N}_{[1,n_u]},\ \mathrm{nz}(\mathbf{u},j) \leqslant 1 \text{ and } \forall i \in \mathbb{N}_{[1,M-1]},\ x_i(x,\mathbf{u}) \in \mathbb{X} \text{ and } x_M(x,\mathbf{u}) \in \mathbb{X}_f\Big\}, \quad (21)$$

in which $\mathrm{nz}(\mathbf{u}, j)$ denotes the number of non-zero elements in the set $\{u_0^{(j)}, u_1^{(j)}, \ldots, u_{M-1}^{(j)}\}$, where $\mathbf{u} = (u_0, u_1, \ldots, u_{M-1})$ and $u_i = (u_i^{(1)}, u_i^{(2)}, \ldots, u_i^{(n_u)}) \in \mathbb{R}^{n_u}, i = 0, 1, \ldots, M-1$. This strategy allows for sparse profiles of the form depicted in Fig. 1(c), which are relevant in, for instance, control of overactuated systems, see, e.g., [3]. Note that due to the additional degree of freedom in the input update times (compared to group sparse profiles), we now consider $(M+1)^{n_u}$ possible discrete choices of input update patterns. However, for a fixed $M \in \mathbb{N}$, one could solve them in an ordered manner trying first the options with a minimal number of actuators that require a non-zero value. Moreover, for fixed $M \in \mathbb{N}$, it requires more data to be communicated, as both the control values and the timing information of when the control values are to be implemented should be sent to the actuators.

Although the discussed input profiles address (actuation and/or communication) resource constraints in *all* elements of the input, the framework can also address problems where resource constraints only apply to certain elements of the input (e.g., the case where only one element u_j is transmitted over a wireless link with limited bandwidth, or the case where only the actuator corresponding to input u_j is subject to wear and tear).

For ease of exposition, in the next section we focus on the algorithm discussed in Section 3.1, but the resulting derivations apply *mutatis mutandis* to the cases described in this section as well, see Remark 2 at the end of Section 3.3.

3.3. Theoretical properties

The self-triggered MPC law given by Algorithm 1 has the following important properties. Recall that the conditions regarding the terminal set and terminal cost (Assumption 1) are standing assumptions.

Theorem 1 (*Recursive Feasibility*)**.** *Given $\beta \geqslant 1$, the control law as in Algorithm 1 is well posed, i.e., for all $x \in \mathbb{X}_f$, $\mathcal{M}^{st}(x) \in \mathbb{N}_{\geqslant 1}$ and $\mathcal{U}^{st}(x) \neq \emptyset$.*

Proof. The proof is based on showing that for each $x \in \mathbb{X}_f$ there exists $u_0 \in \mathbb{U}$ such that

$$\beta\left[V(g(x,u_0)) - V(x)\right] \leqslant -L(x,u_0) \quad (22)$$

and $g(x,u_0) \in \mathbb{X}_f$. Observe that (22) is equivalent to the condition (16) for $M = 1$. In fact, we are going to show that (22) holds for $u_0 = u^{mpc}(x)$. We will employ that $u^{mpc}(x)$ was constructed based on (3) using the conditions (8a)–(8c) related to the terminal set and cost method. We take $\mathbf{u}^*(x) := (u_0^*(x),\ u_1^*(x),\ \ldots,\ u_{N-1}^*(x)) \in \mathcal{U}_N(x)$ and thus $u_0^*(x) = u^{mpc}(x)$, see (5) and (7). Moreover, we take $\tilde{\mathbf{u}} := \big(u_1^*(x),\ \ldots,\ u_{N-1}^*(x),\ K(x_N(x,\mathbf{u}^*(x)))\big)$ and note that as $\mathbf{u}^*(x) \in \mathcal{U}_N(x)$ we have that $x_N(x,\mathbf{u}^*(x)) \in \mathbb{X}_T$, now we can invoke (8b) and (8c) to see that $g\,(x_N(x,\mathbf{u}^*(x)), K\,(x_N(x,\mathbf{u}^*(x)))) \in \mathbb{X}_T$ and $K(x_N(x,\mathbf{u}^*(x))) \in \mathbb{U}$. Hence, we conclude that $\tilde{\mathbf{u}} \in \mathcal{U}_N(g(x,u^{mpc}(x)))$. By following standard arguments, see, e.g., [37], we have that

$$\begin{aligned}
&V\Big(g\big(x,u^{mpc}(x)\big)\Big) - V(x)\\
&\leqslant F\Big(x_N\Big(g\big(x,u^{mpc}(x)\big),\tilde{\mathbf{u}}\Big)\Big)\\
&\quad + \sum_{k=0}^{N-1} L\Big(x_k\Big(g\big(x,u^{mpc}(x)\big),\tilde{\mathbf{u}}\Big),\Pi_k\tilde{\mathbf{u}}\Big)\\
&\quad - F\big(x_N(x,\mathbf{u}^*(x))\big) - \sum_{k=0}^{N-1} L\big(x_k(x,\mathbf{u}^*(x)),\Pi_k\mathbf{u}^*(x)\big),\\
&= F\Big(g\Big(x_N(x,\mathbf{u}^*(x)),K\big(x_N(x,\mathbf{u}^*(x))\big)\Big)\Big) - F\big(x_N(x,\mathbf{u}^*(x))\big)\\
&\quad + L\Big(x_N\big(x,\mathbf{u}^*(x)\big),K\big(x_N(x,\mathbf{u}^*(x))\big)\Big) - L\big(x,u^{mpc}(x)\big). \quad (23)
\end{aligned}$$

As $x_N(x,\mathbf{u}^*) \in \mathbb{X}_T$, we can invoke (8a) to obtain from (23)

$$V\left(g\left(x,u^{mpc}(x)\right)\right) - V(x) \leqslant -L(x,u^{mpc}(x)). \quad (24)$$

Since $\beta \geqslant 1$, we see that (24) implies (22) for $u_0 = u^{mpc}(x)$. Clearly, $u^{mpc}(x) \in \mathbb{U}$ and since we already established $\tilde{\mathbf{u}} \in \mathcal{U}_N(g(x,u^{mpc}(x)))$, we also have $g(x,u^{mpc}(x)) \in \mathbb{X}_f$. Therefore, $u^{mpc}(x) \in \tilde{\mathcal{U}}_1(x)$ and due to (24) also $u^{mpc}(x) \in \mathcal{U}_1^{st}(x)$. Hence, $\mathcal{U}^{st}(x) \neq \emptyset$ and $\mathcal{M}^{st}(x) \in \mathbb{N}_{\geqslant 1}$ for all $x \in \mathbb{X}_f$. □

Theorem 2 (*Sub-Optimality and Constraint Satisfaction*)**.** *Let $\beta \geqslant 1$. The control law as in Algorithm 1 satisfies the performance guarantee given in (10) as well as the constraints given in (2) for the closed-loop system (1) with (17) for all $x_0 \in \mathbb{X}_f$.*

Proof. Condition (12) with the control law in (17), yields for $l \in \mathbb{N}$

$$\sum_{t=t_l}^{t_{l+1}-1} L(x_t,u_t) \leqslant \beta\left[V(x_{t_l}) - V(x_{t_{l+1}})\right] \quad (25)$$

and by the definition in (15),

$$\begin{aligned}
&u_t \in \mathbb{U} \quad \text{for } t \in \mathbb{N}_{[t_l,t_{l+1})},\\
&x_t \in \mathbb{X} \quad \text{for } t \in \mathbb{N}_{(t_l,t_{l+1})} \quad \text{and} \quad x_{t_{l+1}} \in \mathbb{X}_f \subseteq \mathbb{X}.
\end{aligned} \quad (26)$$

Summing (25) for $l = 0$ to $L \in \mathbb{N}$ and using $t_0 = 0$ lead to

$$\begin{aligned}
l = 0; \quad & \sum_{t=t_0}^{t_1-1} L(x_t, u_t) \quad \leq \quad \beta\Big[V(x_{t_0}) - V(x_{t_1})\Big] \\
l = 1; \quad & \sum_{t=t_1}^{t_2-1} L(x_t, u_t) \quad \leq \quad \beta\Big[V(x_{t_1}) - V(x_{t_2})\Big] \\
\vdots \quad & \vdots \qquad\qquad\qquad \vdots \\
l = L; \quad & \sum_{t=t_L}^{t_{L+1}-1} L(x_t, u_t) \quad \leq \quad \beta\Big[V(x_{t_L}) - V(x_{t_{L+1}})\Big] \\
+ \quad & \sum_{t=0}^{t_{L+1}-1} L(x_t, u_t) \quad \leq \quad \beta\Big[V(x_0) - V(x_{t_{L+1}})\Big] \leq \beta V(x_0),
\end{aligned}$$

where in the latter inequality we used that V takes only nonnegative values. Let $L \to \infty$, the desired performance guarantee (10) is obtained. Since we have (26) for all $l \in \mathbb{N}$, also the input and state constraints as in (2) are satisfied. □

Theorem 3 (*Stability*). *Let $\beta \geqslant 1$. The control law given by Algorithm 1 renders the closed-loop system given by (1) and (17) asymptotically stable for initial conditions in $\mathbb{X}_f$.*

Proof. The proof is based on showing that V is a Lyapunov function for the closed-loop system (1) and (17). We will first show that $V(x) \leqslant \alpha_2(\|x\|)$ for all $x \in \mathbb{X}_T$, with α_2 as in (8e). Let $\tilde{\mathbf{x}} := (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_N)$ denote the state sequence generated by the auxiliary system $\tilde{x}_{t+1} = g(\tilde{x}_t, K(\tilde{x}_t))$ starting from $\tilde{x}_0 \in \mathbb{X}_T$ and let $\tilde{\mathbf{u}} = (K(\tilde{x}_0), K(\tilde{x}_1), \ldots, K(\tilde{x}_{N-1}))$ be the corresponding input sequence. Note that as $\tilde{x}_0 \in \mathbb{X}_T$, from (8b) and (8c) we have that $\tilde{\mathbf{x}} \in \mathbb{X}_T^N$ and $\tilde{\mathbf{u}} \in \mathbb{U}^N$. Now, we invoke (8a) to obtain

$$\begin{aligned}
&F(\tilde{x}_1) - F(\tilde{x}_0) + L(\tilde{x}_0, K(\tilde{x}_0)) \leqslant 0, \\
&F(\tilde{x}_2) - F(\tilde{x}_1) + L(\tilde{x}_1, K(\tilde{x}_1)) \leqslant 0, \\
&\vdots \\
&F(\tilde{x}_N) - F(\tilde{x}_{N-1}) + L(\tilde{x}_{N-1}, K(\tilde{x}_{N-1})) \leqslant 0,
\end{aligned}$$

and observe that by summing these inequalities we get $-F(\tilde{x}_0) + F(\tilde{x}_N) + \sum_{k=0}^{N-1} L(\tilde{x}_i, K(\tilde{x}_i)) \leqslant 0$, implying that $J_N(\tilde{x}_0, \tilde{\mathbf{u}}) \leqslant F(\tilde{x}_0)$ for any $\tilde{x}_0 \in \mathbb{X}_T$. Using this fact and (8e), for $\tilde{x}_0 = x \in \mathbb{X}_T$ we have that

$$V(x) \leqslant J_N(x, \tilde{\mathbf{u}}) \leqslant F(x) \leqslant \alpha_2(\|x\|). \tag{27}$$

From (10) and (8d), we have that for all $x \in \mathbb{X}_f$

$$\sum_{t=0}^{\infty} \alpha_1(\|x_t\|) \leqslant \sum_{t=0}^{\infty} L(x_t, u_t) \leqslant \beta V(x), \tag{28}$$

where x_t is the solution to the closed-loop system (1) and (17) starting from $x_0 = x \in \mathbb{X}_f$, and u_t as in (17b). This immediately implies that we have the attractivity property in $\mathbb{X}_f$, i.e., for all $x_0 \in \mathbb{X}_f$ we have $\lim_{t\to\infty} \|x_t\| = 0$. To establish Lyapunov stability we fix $\delta > 0$ such that $\mathcal{B}_\delta := \{x \in \mathbb{R}^{n_x} \mid \|x\| \leqslant \delta\} \subseteq \mathbb{X}_T$ and let $\varepsilon > 0$ be such that $\beta\alpha_2(\delta) \leqslant \alpha_1(\varepsilon)$. Then for $x_0 \in \mathcal{B}_\delta$, from (27) and (28) we see that

$$\sum_{t=0}^{\infty} \alpha_1(\|x_t\|) \leqslant \beta V(x_0) \leqslant \beta\alpha_2(\|x_0\|) \leqslant \beta\alpha_2(\delta) \leqslant \alpha_1(\varepsilon), \tag{29}$$

hence, $\alpha_1(\|x_t\|) \leqslant \alpha_1(\varepsilon)$ for all $t \in \mathbb{N}$, i.e., $x_t \in \mathcal{B}_\varepsilon$ for all $t \in \mathbb{N}$. This completes the proof. □

Hence, Theorems 1–3 establish the properties required in the problem formulation given in Section 2. By maximizing at every execution time t_l the next value t_{l+1}, also significant sparsity properties, and communication reductions (following the variants in Section 3.2), will be realized (see the examples in Section 5).

Remark 2. Regarding the theoretical properties for the variants resulting in infrequently changing input profiles and sparse input profiles, described by $\tilde{\mathcal{U}}_M(x)$ in (20) and $\hat{\mathcal{U}}_M(x)$ in (21), respectively, note that for $x \in \mathbb{X}_f$ and $M = 1$ we have that $\bar{\mathcal{U}}_1(x) = \tilde{\mathcal{U}}_1(x) = \hat{\mathcal{U}}_1(x)$. The proof of Theorem 1 is based on showing that for $x \in \mathbb{X}_f$ a feasible solution exists for $M = 1$, hence, Theorem 1 applies directly to the variants $\tilde{\mathcal{U}}_M(x)$ and $\hat{\mathcal{U}}_M(x)$ as well.

Note that by using $\mathbf{u} \in \tilde{\mathcal{U}}_M(x)$ or $\mathbf{u} \in \hat{\mathcal{U}}_M(x)$ in (15), we see that (26) holds. Moreover, as (15) implies that (16) is satisfied, the performance guarantee (10) can be obtained analogously to the proof of Theorem 2.

Both the variants based on $\mathbf{u} \in \tilde{\mathcal{U}}_M(x)$ and $\mathbf{u} \in \hat{\mathcal{U}}_M(x)$ satisfy the performance guarantee (10), and, moreover, are based on the terminal set and cost method, subject to the conditions given in Assumption 1. As a consequence, the stability proof for these variants can be obtained analogously to the proof of Theorem 3, by using the properties related to the terminal set and terminal cost method, i.e., (8a)–(8e), and the performance guarantee (10).

4. Implementation considerations

In the implementation of the control law in Algorithm 1, the evaluation of (16) plays a crucial role. Note that the term $V(x_M(x, \mathbf{u}))$ in the right-hand side of (16) is a priori unknown due to the fact that it depends on $\mathbf{u}$, which is yet to be constructed. We rearrange terms in (16) to observe that each $\mathbf{u} \in \mathcal{U}_M^{st}(x)$ is characterized by

$$\beta V(x_M(x, \mathbf{u})) + \sum_{k=0}^{M-1} L(x_k(x, \mathbf{u}), \Pi_k \mathbf{u}) \leqslant \beta V(x), \tag{30a}$$

$$\text{and} \quad \mathbf{u} \in \bar{\mathcal{U}}_M(x). \tag{30b}$$

To implement Algorithm 1, the objective is now for a given $x \in \mathbb{X}_f$ to find the largest $M \in \mathbb{N}$ for which (30) is feasible. For fixed $M \in \mathbb{N}$, feasibility of (30) for some $\mathbf{u} \in \bar{\mathcal{U}}_M(x)$ is equivalent to the minimum of the left-hand side of (30a) subject to $\mathbf{u} \in \bar{\mathcal{U}}_M(x)$ being smaller than the right-hand side of (30a). More formally, $\mathcal{U}_M^{st}(x) \neq \emptyset$ if and only if

$$\min_{\mathbf{u} \in \bar{\mathcal{U}}_M(x)} \beta V(x_M(x, \mathbf{u})) + \sum_{k=0}^{M-1} L(x_k(x, \mathbf{u}), \Pi_k \mathbf{u}) \leqslant \beta V(x). \tag{31}$$

As we do not have an explicit formula for $V(x_M(x, \mathbf{u}))$, it is unclear how to verify (31). To resolve this issue, we replace $V(x_M(x, \mathbf{u}))$ in (31) by the corresponding optimization problem (4), which effectively results in an extension of the prediction horizon. To perform this replacement, a similar notation to the one in (13) is introduced for the horizon $M + N$, leading to

$$\begin{aligned}
\bar{\mathcal{U}}_{M,N}(x) := \Big\{ & \mathbf{u} \in \mathbb{U}^{M+N} \mid \forall i \in \mathbb{N}_{[1,M-1]}, \ \Pi_i \mathbf{u} = 0, \\
& \forall j \in \mathbb{N}_{[1,M+N-1]}, \ x_j(x, \mathbf{u}) \in \mathbb{X} \\
& \text{and } x_{M+N}(x, \mathbf{u}) \in \mathbb{X}_T \Big\}.
\end{aligned} \tag{32}$$

Fig. 2 shows an example of $\mathbf{u} \in \bar{\mathcal{U}}_{M,N}(x)$.

Replacing $V(x_M(x, \mathbf{u}))$ in (31) by the corresponding optimization problem (4) and incorporating the notation in (32), the implementation of Algorithm 1 can be realized as follows.

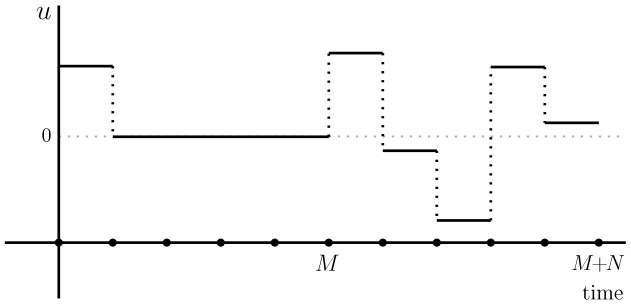

Fig. 2. Example of $\mathbf{u} \in \bar{\mathcal{U}}_{M,N}(x)$.

Algorithm 2. At execution time t_l and state $x_{t_l} = x$, find the largest $M \in \{1, 2, \ldots, \bar{M}\}$, denoted by M^*, such that

$$\min_{\mathbf{u}\in\bar{\mathcal{U}}_{M,N}(x)} \beta F(x_{N+M}(x,\mathbf{u})) + \beta \sum_{k=M}^{M+N-1} L(x_k(x,\mathbf{u}), \Pi_k\mathbf{u}) + \sum_{k=0}^{M-1} L(x_k(x,\mathbf{u}), \Pi_k\mathbf{u}) \leqslant \beta V(x), \tag{33}$$

holds and let $\mathbf{u}^* \in \bar{\mathcal{U}}_{M^*,N}(x)$ be a corresponding minimizer. The next execution time $t_{l+1} = t_l + M^*$ and $\bar{u}_l = \Pi_0\mathbf{u}^*$ in (11).

Note that in Algorithm 2, it has to be verified if the obtained minimizer in the left-hand side is smaller than the right-hand side of the inequality (33), i.e., smaller than $\beta V(x)$, where $V(x)$ obtained by solving the MPC problem in (4). Hence, the algorithm is based on solving similar nonlinear optimization problems as the original MPC setup (only over a somewhat larger horizon), but with (almost) the same number of free variables.

Comparing the standard MPC setup in Section 2.1, which requires using computation, actuation and/or communication resources at each time instant $t \in \mathbb{N}$, with the proposed self-triggered MPC scheme, we see that for the latter scheme the resources are only needed at the execution times t_l, $l \in \mathbb{N}$, while on times $\{t_l + 1, t_l + 2, \ldots, t_{l+1} - 1\}$ no computation, actuation and/or communication resources are required. However, not that at the execution times more computations are needed, i.e., more optimal control problems have to be solved, although the average computational load might be roughly the same as the optimal MPC setup that has to solve optimal control problems at each time instant. In this sense, (more bursty) computations are "traded" for (less) resource utilization (cf. [39]).

In particular, to obtain $M^* = \mathcal{M}^{st}(x)$ for the fixed upperbound $\bar{M} \in \mathbb{N}_{\geqslant 1}$, it is necessary to solve $\bar{M} + 1$ optimization problems, corresponding to MPC problems with prediction horizons $N, N + 1, \ldots, N + \bar{M}$, respectively (having $N + 1$ vectors of free control variables). Hence, at time t_l and state $x_{t_l} = x$, one has to solve $\bar{M} + 1$ optimization problems, and then the next $\mathcal{M}^{st}(x) - 1$ steps, no computation (and no actuation and communication) resources are needed. This indicates that when $\mathcal{M}^{st}(x)$ is large and close to $\bar{M}$, the average number of optimization problems be solved is almost the same as for the standard MPC setup given in Section 2.1. However, the computations are more "bursty" now (only at t_l, $l \in \mathbb{N}$), but this gives the advantage of obtaining sparse or sporadically changing control signals with constraint satisfaction and guaranteed performance in terms of the *original* MPC cost.

Remark 3. An alternative implementation scheme that preserves the properties in Theorems 1–3 is obtained by replacing $\mathcal{M}^{st}(x)$ in (18) by

$$\mathcal{M}^{st}_{incr}(x) = \left\{M \in \mathbb{N}_{[1,\bar{M}]} \mid \mathcal{U}^{st}_m(x) \neq \emptyset,\ m \in \mathbb{N}_{[1,M]}\right\}$$

and $\mathcal{U}^{st}(x)$ in (19) by

$$\mathcal{U}^{st}_{incr}(x) = \mathcal{U}^{st}_{\mathcal{M}^{st}_{incr}(x)}(x).$$

This scheme incrementally increases M until (33) ceases to hold, and does not search for possibly larger values beyond $\mathcal{M}^{st}_{incr}(x)$ for which (33) can still be guaranteed. This alternative implementation requires solving $\mathcal{M}^{st}_{incr}(x) + 1$ optimization problems when in state x, while in the next $\mathcal{M}^{st}_{incr}(x)$ no computation, communication and actuation is needed. This indicates that for the incremental scheme that when $\mathcal{M}^{st}_{incr}(x)$ is typically large, the average number of optimization problems to be solved is almost the same as the optimal MPC setup in Section 2.1. We will illustrate this quantitatively in Section 5.

Remark 4. In order to deal with the computational burden of standard MPC problems, it is common practice to reduce the degrees of freedom of the optimization problem by restricting the input to be constant over several time-steps. This policy is referred to as "move blocking", see, e.g., [40–42]. Move blocking is usually applied towards the end of the optimization horizon. In contrast to move blocking, the proposed self-triggered strategy fixes the input at the *beginning* of the optimization horizon (see Fig. 2), while giving full flexibility towards the end of the horizon. This is an interesting distinction.

5. Numerical examples

In order to illustrate the effectiveness of the self-triggered MPC scheme, two numerical examples are presented. Both examples consider nonlinear systems subject to state constraints. In the first example, we require $\mathbf{u} \in \bar{\mathcal{U}}_M(x)$ (group sparsity), whereas in the second example we consider a system with two inputs and consider the variant based on $\mathbf{u} \in \hat{\mathcal{U}}_M(x)$ (sparsity). In both cases we consider a quadratic cost function (3a) of the form

$$J_N(x,\mathbf{u}) := x_N^\top(x,\mathbf{u})Px_N(x,\mathbf{u}) + \sum_{k=0}^{N-1}\left(x_k^\top(x,\mathbf{u})Qx_k(x,\mathbf{u}) + u_k^\top Ru_k\right),$$

where $Q \succ 0$, $R \succ 0$ and $P \succ 0$ are weighting matrices.

5.1. Self-triggered strategy for $\mathbf{u} \in \bar{\mathcal{U}}_M(x)$

We consider an open-loop unstable discrete-time nonlinear system given by

$$g(x_t, u_t) = \begin{cases} x^{(1)}_{t+1} = x^{(1)}_t + x^{(2)}_t + \frac{1}{4}(x^{(1)}_t)^2, \\ x^{(2)}_{t+1} = x^{(2)}_t + \frac{1}{4}(x^{(2)}_t)^2 + u_t, \end{cases} \tag{34}$$

where $x_t = [x^{(1)}_t\ x^{(2)}_t]^\top$, and the weighting matrices of the running cost are chosen as

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad R = 1. \tag{35}$$

Additionally, state constraints $-1 \leqslant x^{(i)}_t \leqslant 1$, $i = 1, 2$, $t \in \mathbb{N}$, are imposed on the system and the prediction horizon is fixed to $N = 20$. The upperbound on the inter-transmission times is set to $\bar{M} = 50$. The terminal cost and terminal set satisfying Assumption 1 are computed based on linearization of the system and upper bounding the linearization error, and are given by $F(x) = x^\top Px$, $x \in \mathbb{R}^{n_x}$, where $P = \begin{bmatrix} 5.8942 & 4.7384 \\ 4.7384 & 9.2263 \end{bmatrix}$, $K = -[0.4221\ \ 1.2439]$ and $\mathbb{X}_T = \{x \in \mathbb{X} \mid x^\top Px \leqslant \alpha\}$, with $\alpha = 0.2967$. We consider 20 initial

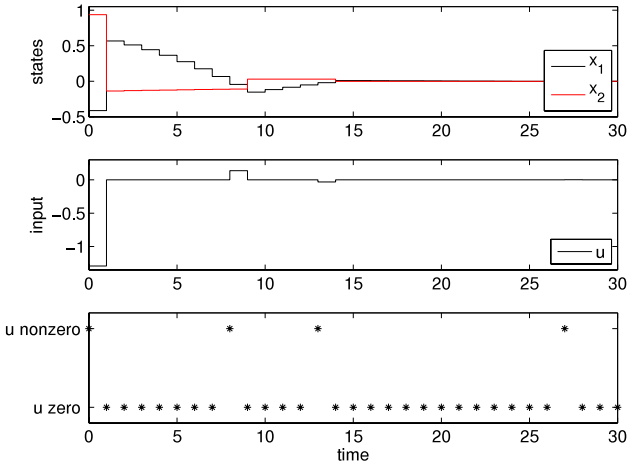

Fig. 3. Resulting states and inputs versus time, for $\beta = 1.1$.

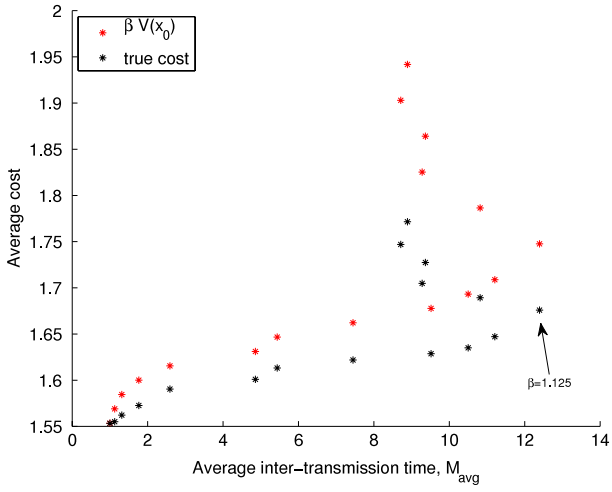

Fig. 4. Cost versus the (average) inter-transmission time, for $\beta \in \{1,\ 1.010,\ \ldots,\ 1.100,\ 1.125,\ \ldots,\ 1.250\}$.

conditions selected randomly within $\mathbb{X}_f$. Fig. 3 shows the time response for one initial condition $x_0 = [-0.4121\ \ 0.9363]^\top$ with $\beta = 1.1$. We observe that even for a slight decrease in performance (i.e., an increase of at most 10% of the guaranteed upperbound on the infinite horizon cost, see (10)), we obtain input profiles with significant sparsity.

Fig. 4 shows the cost versus the (average) inter-transmission time for $\beta \in \{1,\ 1.010,\ \ldots,\ 1.100,\ 1.125,\ \ldots,\ 1.250\}$ averaged over 20 initial conditions. The true infinite horizon cost (cf. (10)) is obtained by evaluating $\sum_{k=0}^{H} x_k^\top Q x_k + u_k^\top R u_k$ over (sufficiently large) simulation horizons H. We see that by increasing β beyond $\beta > 1.125$ the cost increases, but there is no further reduction in resource utilization. However, with at most a 12.5% increase on the infinite horizon cost in (10) (i.e., $\beta = 1.125$), we obtain an average inter-transmission time M_{avg} of 12.4. Hence, for this nonlinear example, we considerably reduce the system's resource utilization with only a slight decrease in performance compared to the traditional MPC (with an average inter-transmission time of 1). This shows the algorithm's capabilities of providing significant reductions in resource utilization, even at a slight decrease in performance.

5.2. Self-triggered strategy for $\mathbf{u} \in \hat{\mathcal{U}}_M(x)$

We modify system (34) by adding a second input to illustrate the effectiveness of the sparsity variant based on $\mathbf{u} \in \hat{\mathcal{U}}_M(x)$ (see

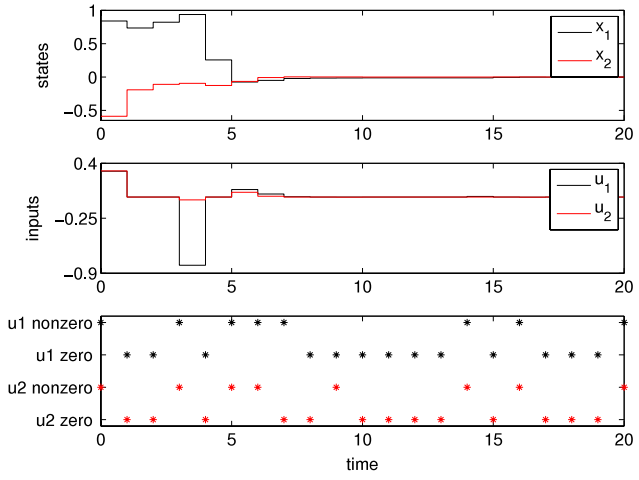

Fig. 5. Resulting states, inputs and indication of element-wise input sparsity versus time, for $\beta = 1.05$.

(21)). Consider the system

$$g(x_t, u_t) = \begin{cases} x_{t+1}^{(1)} = x_t^{(1)} + x_t^{(2)} + \frac{1}{4}(x_t^{(1)})^2 + u_t^{(1)}, \\ x_{t+1}^{(2)} = x_t^{(2)} + \frac{1}{4}(x_t^{(2)})^2 + u_t^{(2)}, \end{cases} \tag{36}$$

where $x_t = [x_t^{(1)}\ \ x_t^{(2)}]^\top$, $u_t = [u_t^{(1)}\ \ u_t^{(2)}]^\top$, and the weighting matrices of the running cost are chosen as

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{37}$$

Additionally, state constraints $-1 \leqslant x_t^{(i)} \leqslant 1,\ i = 1, 2,\ t \in \mathbb{N}$, are imposed on the system and the prediction horizon is fixed to $N = 10$. Due to the combinatorial nature of the required optimization problem, the upperbound on the inter-transmission times is set to $\bar{M} = 8$. The terminal cost and terminal set satisfying Assumption 1 are computed based on linearization of the system and upper bounding the linearization error, and are given by $F(x) = x^\top P x$, $x \in \mathbb{R}^{n_x}$, where $P = \begin{bmatrix} 3.1878 & 1.3464 \\ 1.3464 & 4.8938 \end{bmatrix}$, $K = -\begin{bmatrix} 0.5939 & 0.6732 \\ 0.0793 & 0.7737 \end{bmatrix}$ and $\mathbb{X}_T = \{x \in \mathbb{X} \mid x^\top P x \leqslant \alpha\}$, with $\alpha = 1.9955$. Fig. 5 shows the time response for initial condition $x_0 = [0.8404\ \ -0.5880]^\top$ with $\beta = 1.05$. As we require $\mathbf{u} \in \hat{\mathcal{U}}_M(x)$, this allows each actuator to update independently but sporadically. From Fig. 5, we observe that the algorithm exploits this additional degree of freedom (compared to group sparse input profiles). Moreover, we again observe that, for a slight decrease in performance (i.e., an increase of at most 5% of the guaranteed upperbound on the infinite horizon cost, see (10)), we obtain input profiles with significant sparsity (i.e., 66% of the input values are zero).

6. Conclusions

In this paper we proposed a self-triggered control strategy for discrete-time nonlinear systems subject to state and input constraints. The proposed strategy has three important features: (i) significant reductions in resource utilization are obtained, (ii) a priori closed-loop performance guarantees are provided (by design), (iii) co-design of both the feedback law and triggering condition is achieved. Regarding the performance guarantees (feature (ii)), the control laws and triggering mechanisms were designed jointly such that an a priori chosen (sub-optimal) level of performance in terms of the original infinite horizon cost function is guaranteed, next to asymptotic stability and constraint

satisfaction. Interestingly, the presented framework is flexible in the sense that it can be configured to generate both sparse and sporadically changing input profiles thereby having the ability to serve various application domains in which different resources are scarce or expensive, e.g., communication bandwidth in networked control systems, battery power in wireless control, or fuel in space or underwater vehicles. The effectiveness of the approach was illustrated by means of numerical examples, showing a significant reduction in the usage of the system's resources, without trading much of the guaranteed achievable performance. As such, the proposed approach provides a viable control strategy to balance the usage of the system's resources and control performance.

Acknowledgements

Tom Gommans and Maurice Heemels are supported by the Dutch Science Foundation (STW) and the Dutch Organization for Scientific Research (NWO) under the VICI grant "Wireless controls systems: A new frontier in automation" (No. 11382).

The authors would like to thank Thomas Theunisse for his contributions in the implementation of the presented algorithms.

References

[1] E.N. Hartley, M. Gallieri, J.M. Maciejowski, Terminal spacecraft rendeveuz and capture with ℓasso model predictive control, Internat. J. Control 86 (2013) 2104–2113.
[2] M. Chyba, S. Grammatico, V.T. Huynh, J. Marriott, B. Piccoli, R.N. Smith, Reducing actuator switchings for motion control of autonomous underwater vehicles, in: American Control Conference, 2013, pp. 1406–1411.
[3] M. Gallieri, J.M. Maciejowski, ℓasso MPC: smart regulation of over-actuated systems, in: American Control Conference, 2012, pp. 1217–1222.
[4] J.P. Hespanha, P. Naghshtabrizi, Y. Xu, A survey of recent results in networked control systems, Proc. IEEE 95 (2007) 138–162.
[5] L. Schenato, To zero or to hold control inputs with lossy links? IEEE Trans. Automat. Control 54 (2009) 1093–1099.
[6] V. Raghunathan, C. Schurgers, S. Park, M.B. Srivastava, Energy-aware wireless microsensor networks, IEEE Signal Process. Mag. 19 (2002) 40–50.
[7] M. Gallieri, J.M. Maciejowski, Stabilising terminal cost and terminal controller for ℓasso-MPC: enhanced optimality and region of attraction, in: European Control Conference, 2013, pp. 524–529.
[8] M. Annergren, A. Hansson, B. Wahlberg, An ADMM algorithm for solving ℓ_1 regularized MPC, in: Conference on Decision and Control, 2012, pp. 4486–4491.
[9] M. Nagahara, D.E. Quevedo, Sparse representations for packetized predictive networked control, in: IFAC World Congress, 2011, pp. 84–89.
[10] H. Ohlsson, F. Gustafsson, L. Ljung, S. Boyd, Trajectory generation using sum-of-norms regularization, in: Conference on Decision and Control, 2010, pp. 540–545.
[11] K.J. Åström, B.M. Bernhardsson, Comparison of periodic and event based sampling for first order stochastic systems, in: IFAC World Congress, 1999, pp. 301–306.
[12] K.-E. Arzén, A simple event-based PID controller, in: IFAC World Congress, Vol. 18, 1999, pp. 423–428.
[13] M.C.F. Donkers, W.P.M.H. Heemels, Output-based event-triggered control with guaranteed $\mathcal{L}_\infty$ gain and improved and decentralized event-triggering, IEEE Trans. Automat. Control 57 (2012) 1362–1376.
[14] W.P.M.H. Heemels, J.H. Sandee, P.P.J. van den Bosch, Analysis of event-driven controllers for linear systems, Internat. J. Control 81 (2008) 571–590.
[15] J. Lunze, D. Lehmann, A state-feedback approach to event-based control, Automatica 46 (2010) 211–215.
[16] P. Tabuada, Event-triggered real-time scheduling of stabilizing control tasks, IEEE Trans. Automat. Control 52 (2007) 1680–1685.
[17] J. Yepez, M. Velasco, P. Marti, E.X. Martin, J.M. Fuertes, One-step finite horizon boundary with varying control gain for event-driven networked control systems, in: IEEE Industrial Electronics Society, 2011, pp. 2606–2611.
[18] M. Mazo Jr., A. Anta, P. Tabuada, An ISS self-triggered implementation of linear controllers, Automatica 46 (2010) 1310–1314.
[19] M. Velasco, P. Marti, J. Yepez, F.J. Ruiz, J.M. Fuertes, E. Bini, Qualitative analysis of a one-step finite-horizon boundary for event-driven controllers, in: IEEE Joint Conference Decision and Control and European Control Conference, 2011, pp. 1662–1667.
[20] X. Wang, M. Lemmon, Self-triggered feedback control systems with finite-gain $\mathcal{L}_2$ stability, IEEE Trans. Automat. Control 45 (2009) 452–467.
[21] J. Almeida, C. Silvestre, A.M. Pascoal, Self-triggered output feedback control of linear plants, in: American Control Conference, 2011, pp. 2831–2836.
[22] A. Anta, P. Tabuada, To sample or not to sample: self-triggered control for nonlinear systems, IEEE Trans. Automat. Control 55 (2010) 2030–2042.
[23] M.C.F. Donkers, P. Tabuada, W.P.M.H. Heemels, Minimum attention control for linear systems: a linear programming approach, Discrete Event Dyn. Syst. 24 (2014) 199–218.
[24] T.M.P. Gommans, D. Antunes, M.C.F. Donkers, P. Tabuada, W.P.M.H. Heemels, Self-triggered linear quadratic control, Automatica 50 (2014) 1279–1287.
[25] W.P.M.H. Heemels, K.H. Johansson, P. Tabuada, An introduction to event-triggered and self-triggered control, in: Conference on Decision and Control, 2012, pp. 3270–3285.
[26] A. Bemporad, Predictive control of teleoperated constrained systems with unbounded communication delays, in: Conference on Decision and Control, Vol. 2, 1998, pp. 2133–2138.
[27] W. Hu, G. Liu, D. Rees, Event-driven networked predictive control, IEEE Trans. Ind. Electron. 54 (2007) 1603–1613.
[28] L. Greco, A. Chaillet, A. Bicchi, Exploiting packet size in uncertain nonlinear networked control systems, Automatica 48 (2012) 2801–2811.
[29] A. Eqtami, V. Dimarogonas, K.J. Kyriakopoulos, Event-triggered control for discrete-time systems, in: American Control Conference, 2010, pp. 4719–4724.
[30] A. Eqtami, D.V. Dimarogonas, K.J. Kyriakopoulos, Event-triggered strategies for decentralized model predictive controllers, in: IFAC World Congress, 2011, pp. 10068–10073.
[31] D. Bernardini, A. Bemporad, Energy-aware robust model predictive control based on noisy wireless sensors, Automatica 48 (2012) 36–44.
[32] E. Henriksson, D.E. Quevedo, H. Sandberg, K.H. Johansson, Self-triggered model predictive control for network scheduling and control, in: IFAC International Symposium on Advanced Control of Chemical Processes, 2012, pp. 432–438.
[33] D. Antunes, W. Heemels, Rollout event-triggered control: beyond periodic control performance, IEEE Trans. Automat. Control 59 (2014) 3296–3311.
[34] P. Varutti, B. Kern, T. Faulwasser, R. Findeisen, Event-based model predictive control for networked control systems, in: Conference on Decision and Control, 2009, pp. 567–572.
[35] P. Varutti, R. Findeisen, Event-based NMPC for networked control systems over UDP-like communication channels, in: American Control Conference, 2011, pp. 3166–3171.
[36] J.D.J. Barradas Berglind, T.M.P. Gommans, W.P.M.H. Heemels, Self-triggered MPC for constrained linear systems and quadratic costs, in: IFAC Conference on Nonlinear Model Predictive Control, 2012, pp. 342–348.
[37] D.Q. Mayne, J.B. Rawlings, C.B. Rao, P.O.M. Stockaert, Constrained model predictive control: stability and optimality, Automatica (2000) 789–814.
[38] J. Huang, T. Zhang, The benefit of group sparsity, Ann. Statist. 38 (2010) 1978–2004.
[39] J.K. Yook, D.M. Tilbury, N.R. Soparkar, Trading computation for bandwidth: reducing communication in distributed control systems using state estimators, IEEE Trans. Control Syst. Technol. 10 (2002) 503–518.
[40] J.M. Maciejowski, Predictive Control with Constraints, Prentice Hall, 2000.
[41] S.J. Qin, T.A. Badgwell, An overview of industrial model predictive control technology, Chemical Process Control—V 93 (1997) 232–256.
[42] P. Tøndel, T.A. Johansen, Complexity reduction in explicit model predictive control, in: IFAC World Congress, 2002.