Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

A Resource-Aware Approach to Self-Triggered Model Predictive Control

Stefan Wildhagen * **Colin N. Jones** ** **Frank Allgöwer** *

* *Institute for Systems Theory and Automatic Control, University of Stuttgart, Germany ({wildhagen,allgower}@ist.uni-stuttgart.de).*
** *Automatic Control Laboratory, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland (colin.jones@epfl.ch).*

Abstract: In this paper, we consider a self-triggered formulation of model predictive control. In this variant, the controller decides at the current sampling instant itself when the next sample should be taken and the optimization problem be solved anew. We incorporate a pointwise-in-time resource constraint into the optimization problem, whose exact form can be chosen by the user. Thereby, the proposed scheme is made resource-aware with respect to a universal resource, which may pertain in practice for instance to communication, computation, energy or financial resources. We show that by virtue of the pointwise-in-time constraints, also a transient and an asymptotic average constraint on the resource usage are guaranteed. Furthermore, we derive conditions on the resource under which the proposed scheme achieves recursive feasibility and convergence. Finally, we demonstrate our theoretical results in a numerical example.

Keywords: Event-based control, Model predictive and optimization-based control, Control under communication constraints, Control under computation constraints.

1. INTRODUCTION

Model predictive control (MPC) is an effective method for the control of nonlinear, constrained and multi-variable systems, and has received great attention in both theory (Mayne et al. (2000)) and practice (Qin and Badgwell (2003)). The main principle of MPC is receding horizon optimal control: an optimal control problem is solved over a finite horizon and the first part of the predicted input is applied to the process, before a new sample of the state is taken and the scheme is repeated. Traditionally, these samples are taken equidistantly in time although more recently, some approaches abandoned this concept in favor of aperiodic sampling. These works considered mostly MPC in the context of Networked Control Systems (NCS), where state measurements and control signals must be transmitted via communication networks with a limited amount of communication resources. In such an environment, to avoid network congestion, samples should be taken and new control inputs should be transmitted only when needed.

To determine when samples should be taken in aperiodic MPC for NCS, mainly event-triggered and self-triggered approaches are of interest. Event-triggered MPC (see for example Varutti et al. (2009); Eqtami et al. (2010); Li and Shi (2014)) uses a triggering condition, which is persistently monitored during runtime, in order to determine when the next sample should be taken. In contrast, in self-triggered MPC, the next sampling time is determined already at the current sampling time based on predictions of the system behavior. Some of the earliest approaches in this direction are presented in Bernardini and Bemporad (2012); Barradas-Berglind et al. (2012) and Henriksson et al. (2015). In the former two works, the sampling period is chosen as long as possible such that in closed loop, stability and a certain level of performance are still guaranteed. In the latter work on the other hand, the next sampling instant is determined based on an optimization problem, which jointly considers in its objective function the process cost and the communication load. In such an approach, as taken also in Zhan et al. (2017); Li et al. (2018), the controller may directly trade off between control performance and sampling interval.

In the latter works, the input is additionally held constant in between sampling instances. In effect, these self-triggered MPC schemes trigger transmissions of state measurements and new inputs over the communication network only when required by the process, thereby being resource-aware with respect to the network's communication resources. In practical applications, however, other types of resources are of interest as well: in battery-operated devices, energy is scarce; in cloud computing, there is a financial component to how much computational power is requested; on micro controllers, computing performance is considerably limited.

In this work, we propose a self-triggered MPC which determines the next sampling time via optimization similar to the above-mentioned schemes, although only the process cost is represented in its objective function. Hence, in contrast to these works (which add a cost term on the sampling interval), the proposed controller chooses the sampling interval that is most favorable for the process itself. To achieve resource awareness nonetheless, a *constraint* on the resource is added to the optimization problem. In particular, we consider a resource that is charged at a

⋆ Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 285825138; 390740016 and the Swiss National Science Foundation under the RISK project (grant 200021 175627).

constant rate and that, using a certain sampling interval, is decreased by a corresponding cost. Surplus resources are saved in a storage to be used at a later point in time. The resource constraint is then to enforce that the level of the resource does not fall below zero. Since we allow for arbitrary resource cost functions, the proposed self-triggered MPC is resource-aware with respect to a general resource. The concrete form of the resource cost function may be chosen by the user, such that energy-related, financial, computational or other types of resources can be incorporated into the controller.

A self-triggered MPC may adaptively change its sampling interval based on the current operating conditions of the process, and thereby react to unforeseen operating conditions such as a set point change or disturbances. Hence, it is in general not possible to bound the average sampling frequency over a certain time interval a priori (apart from trivial bounds by a lower bounded sampling interval). Our main contribution is to show that in the proposed formulation, closed-loop guarantees on the average resource usage are indeed achieved even in unforeseen operating conditions. Thereby, both transient (which hold over arbitrary, but finite time horizons) and asymptotic average constraints on the resource usage are enforced. Especially non-trivial transient bounds are novel in self-triggered MPC. Furthermore, we provide conditions on the process and the resource under which the proposed self-triggered MPC is recursively feasible and achieves convergence.

The remainder of the paper is organized as follows. In Section 2, the considered process and resource dynamics are introduced before the self-triggered MPC scheme is presented in Section 3. The main results on average resource usage, and on recursive feasibility and convergence are derived in Sections 4 and 5. A numerical example and an outlook are given in Sections 6 and 7, respectively.

Apart from standard notation, let $\mathbb{I}$ denote the set of integers and for $a, b \in \mathbb{I}$, let us define $\mathbb{I}_{[a,b]} \coloneqq \mathbb{I} \cap [a, b]$ and $\mathbb{I}_{\geq a} \coloneqq \mathbb{I} \cap [a, \infty)$.

2. PRELIMINARIES

2.1 System description

We consider a nonlinear continuous-time process

$$\dot{x}(t) = f(x(t), u(t)),\ x(0) \in \mathbb{X}_0, \tag{1}$$

with time $t \in [0, \infty)$, state $x(t) \in \mathbb{R}^n$, and input subject to the constraints $u(t) \in \mathbb{U} \subseteq \mathbb{R}^m$. We have for the vector field $f : \mathbb{R}^n \times \mathbb{U} \to \mathbb{R}^n$, and furthermore, there is a cost $\ell : \mathbb{R}^n \times \mathbb{U} \to [0, \infty)$ associated with the process. The following standing assumptions are borrowed from Findeisen (2006) and Faulwasser (2012), and will be assumed to hold true for the remainder of the paper without further mention.

Assumption 1. The input constraint set $\mathbb{U}$, the vector field f and the cost ℓ fulfill the following properties:

- $\mathbb{U}$ is compact and contains the origin.
- f is continuous and locally Lipschitz in x for any $u \in \mathbb{U}$, and fulfills $f(0, 0) = 0$.
- For any $x_0 \in \mathbb{X}_0$ and any input function $u(\cdot) \in \mathcal{PC}(\mathbb{U})$ (the set of piecewise continuous input functions with values in $\mathbb{U}$), (1) has an absolutely continuous solution.
- ℓ is continuous and positive definite.

2.2 Self-triggering mechanism and ZOH Input

The MPC is activated at discrete sampling instants $t_k, k \in \mathbb{I}_{\geq 0}$, which are not evenly spaced in time as in standard formulations. Instead, the MPC decides about the next sampling instant itself in a self-triggered fashion. The sampling interval or inter-sampling period $\Delta_k \coloneqq t_{k+1} - t_k \geq 0$ is hence a part of the optimization variables.

In between sampling times, the control law $\pi(x)$ is implemented in a zero-order hold fashion with constant input values during the sampling interval, i.e.,

$$u(t) = \pi(x(t_k)),\ t \in [t_k, t_{k+1}).$$

As a result, the considered control algorithm achieves a sparsely changing control input.

2.3 Resource constraints

In this work, we assume that there is a limited resource which determines when the controller may be activated and the control input may be changed. The level of the resource r is charged at a constant rate of $p \in [0, \infty)$ and using a certain sampling interval incurs a corresponding resource cost of $\mu(\Delta)$, $\mu : [0, \infty) \to [0, \infty)$. The resource level does not exceed a certain upper threshold $\overline{r} \in [0, \infty)$. Hence, the resource level at sampling times evolves according to

$$r_{k+1} = \min\{r_k + \Delta_k p - \mu(\Delta_k), \overline{r}\},\ r_0 \in [0, \overline{r}]. \tag{2}$$

A certain Δ_k is only feasible if the resource is not depleted using this sampling interval, i.e., the resource level is constrained by

$$r_k \geq 0,\ \forall k \in \mathbb{I}_{\geq 0}. \tag{3}$$

Such a form of resource dynamics is inspired by the token bucket specification, which is a common model in communication theory for the communication resources of a digital network (see Tanenbaum and Wetherall (2011)). Recently, this model was used in the context of NCS to determine when new inputs should be transmitted over a resource-constrained network (Linsenmayer and Allgöwer (2018); Wildhagen et al. (2019)). In the token bucket specification, the resource cost μ is assumed to be independent of the sampling interval. By the means of an arbitrary resource cost as used in this work, more general resource-constrained setups than NCS can be considered.

Importantly, if the resource evolving according to (2) fulfills constraint (3), then one can also guarantee constraints on the average resource usage. This is useful, e.g., if one wants to prescribe a certain average sampling frequency or energy consumption. We will detail this idea in Section 4.

3. SELF-TRIGGERED MPC SCHEME

In this section, we present the operation scheme of the proposed self-triggered MPC and discuss some of its properties. The MPC optimization problem introduced here provides no guarantees on recursive feasibility and convergence for the resulting closed-loop system. We will present an appropriate extension in order to provide theoretical guarantees on these matters in Section 5.

The main idea is to use a sampled-data version of the process (1) in the MPC optimization problem. Since the input is assumed to be constant in between two sampling instants, the state of the system after one sampling interval is determined only by the initial condition, the input and the inter-sampling period according to

$$x_{k+1} = \phi(x_k, u_k, \Delta_k) := \int_0^{\Delta_k} f(z(t), u_k)\mathrm{d}t,\ x_0 = x(0) \quad (4)$$

where

$$\dot{z}(t) = f(z(t), u_k),\ z(0) = x_k. \quad (5)$$

This sampled-data state naturally coincides with the state of the continuous-time process (1) at the sampling instances. Similarly, a sampled-data version of the cost can be defined

$$\lambda(x_k, u_k, \Delta_k) := \int_0^{\Delta_k} \ell(z(t), u_k)\mathrm{d}t. \quad (6)$$

The MPC optimization problem then minimizes the cost (6) along the predicted trajectories of the sampled-data process (4). Let $\boldsymbol{x}(t_k) := \{x_0(t_k), \dots, x_N(t_k)\}$ and $\boldsymbol{r}(t_k) := \{r_0(t_k), \dots, r_N(t_k)\}$ denote the predicted states and resource levels over the prediction horizon $N \in \mathbb{I}_{\geq 1}$. Furthermore, let $\boldsymbol{u}(t_k) := \{u_0(t_k), \dots, u_{N-1}(t_k)\}$ and $\boldsymbol{\Delta}(t_k) := \{\Delta_0(t_k), \dots, \Delta_{N-1}(t_k)\}$ denote the predicted inputs and sampling intervals. Then, at sampling time t_k, the controller solves the MPC optimization problem

$$\min_{\substack{\boldsymbol{x}(t_k),\boldsymbol{r}(t_k)\\ \boldsymbol{u}(t_k),\boldsymbol{\Delta}(t_k)}} \underbrace{\sum_{i=0}^{N-1} \lambda(x_i(t_k), u_i(t_k), \Delta_i(t_k))}_{=:V(\boldsymbol{x}(t_k),\boldsymbol{r}(t_k),\boldsymbol{u}(t_k),\boldsymbol{\Delta}(t_k))}$$

$$\text{s.t. } x_{i+1}(t_k) = \phi(x_i(t_k), u_i(t_k), \Delta_i(t_k)) \quad (7a)$$

$$r_{i+1}(t_k) = \min\{r_i(t_k) + p\Delta_i(t_k) - \mu(\Delta_i(t_k)), \overline{r}\} \quad (7b)$$

$$r_i(t_k) \geq 0,\ u_i(t_k) \in \mathbb{U} \quad (7c)$$

$$\underline{\Delta} \leq \Delta_i(t_k) \leq \overline{\Delta}, \quad \forall i \in \mathbb{I}_{[0,N-1]} \quad (7d)$$

$$x_0(t_k) = x(t_k),\ r_0(t_k) = r_k. \quad (7e)$$

In this problem, the independent optimization variables are the inputs $u_i(t_k)$ as well as the sampling intervals $\Delta_i(t_k)$. The predicted state and resource level trajectories must fulfill their respective dynamics and initial conditions as enforced in (7a), (7b) and (7e). Additionally, the predicted resource levels and inputs must comply with their respective constraints (7c). Lastly, a lower and upper bound $\underline{\Delta}, \overline{\Delta} \in [0, \infty)$ on the sampling intervals are imposed (7d).

Remark 1. Note that the state transition map ϕ does not have to be determined analytically in order to solve the MPC optimization problem. In practice, one may rely on numerical integration to determine ϕ approximately.

Remark 2. The lower and upper limits on the inter-sampling period are not necessary to guarantee recursive feasibility later. In contrast, they need to be carefully adjusted not to jeopardize this property, as we will see in Section 5. Then again, the upper limit guarantees a minimal attention devoted to the process and may facilitate finding a solution to the optimization problem in practice. The lower limit may be used to avoid Zeno behavior.

Remark 3. Since the MPC optimization problem relies entirely on a sampled-data formulation, state constraints on the continuously evolving process state cannot be incorporated. At sampling times, state constraints of the form $x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$ may be enforced according to

$$x_i(t_k) \in \mathbb{X}, \quad \forall i \in \mathbb{I}_{[0,N-1]}.$$

Although in theory, a guarantee on recursive feasibility would not be lost under the assumptions presented in Section 5, such a constraint provides no guarantee that the closed-loop state trajectory evolving in continuous time does not violate the state constraints.

Remark 4. In the MPC optimization problem, the sampling intervals may take values in a connected set. In contrast, previous works were formulated in discrete time, such that the sampling intervals took values in a discrete set giving rise to a mixed integer optimization problem.

In self-triggered MPC, the elapsed time within the prediction horizon is subject to the chosen sampling intervals instead of being predetermined. Previous formulations of self-triggered MPC have always considered one sampling instant as an independent optimization variable within the prediction horizon, i.e., $N = 1$ (cf. Henriksson et al. (2015); Zhan et al. (2017); Li et al. (2018)). The proposed approach generalizes this concept and considers an arbitrary but fixed number of sampling instants $N \geq 1$. It should be noted that using a general $N \geq 1$ was hinted at in (Henriksson et al., 2015, Remark 7).

The state, resource level, input and inter-sampling period sequences solving the MPC optimization problem are denoted by the superscript *. The MPC then operates with the following scheme.

Algorithm 1. MPC Scheme

0) Set $t_0 = 0$ and $k = 0$.
1) At time t_k, measure $x(t_k)$ and solve the MPC optimization problem.
2) Apply $u(t) = u_0^*(t_k)$, $\forall t \in [t_k, t_k + \Delta_0^*(t_k))$ to (1) and calculate $r_{k+1} = \min\{r_k + p\Delta_0^*(t_k) - \mu(\Delta_0^*(t_k)), \overline{r}\}$.
3) Set $t_k \leftarrow t_k + \Delta_0^*(t_k)$, $k \leftarrow k + 1$ and go to 1).

Remark 5. As an extension, one could imagine a "multi-step" version of Algorithm 1, where the first $M \in \mathbb{I}_{[1,N]}$ parts of $\boldsymbol{u}^*(t_k)$ and $\boldsymbol{\Delta}^*(t_k)$ are applied in closed loop. Then the state would be sampled and the MPC optimization problem be solved again at $t_k + \sum_{i=0}^{M-1} \Delta_i^*(t_k)$. Although not further mentioned, all following results also hold under application of such a scheme.

4. AVERAGE CONSTRAINTS ON RESOURCE USAGE

Under application of Algorithm 1, apparently the closed-loop inter-sampling periods $\Delta_k := \Delta_0^*(t_k)$ are such that the resource constraint (3) is always fulfilled. With this observation, an upper limit on the average resource usage under application of the MPC can be established.

Lemma 1. Suppose that for a $k \in \mathbb{I}_{\geq 1}$, the MPC optimization problem is feasible at all sampling instances t_i, $i \in \mathbb{I}_{[0,k-1]}$. Then,

$$\frac{\sum_{i=0}^{k-1} \mu(\Delta_i)}{t_k} \leq \frac{r_0}{t_k} + p \quad (8)$$

holds for the closed loop under application of Algorithm 1.

Proof. From the resource dynamics (2), we obtain

$$r_k = \min\{r_{k-1} + \Delta_{k-1}p - \mu(\Delta_{k-1}), \overline{r}\}$$
$$\leq r_{k-1} + \Delta_{k-1}p - \mu(\Delta_{k-1}).$$

Repeating this argument and considering (3) gives

$$0 \leq r_k \leq r_0 + \textstyle\sum_{i=0}^{k-1} \Delta_i p - \mu(\Delta_i).$$

Because $\sum_{i=0}^{k-1} \Delta_i = t_k$, we obtain (8) by rearranging. □

Remark 6. Note that Lemma 1 also implies that

$$\frac{\sum_{i=j}^{k-1} \mu(\Delta_i)}{t_k - t_j} \leq \frac{r_j}{t_k - t_j} + p$$

for any $j \in \mathbb{I}_{[0,k-1]}$. In this respect, (8) can be seen as a transient average constraint, i.e., a constraint on a quantity averaged over any finite time period (Müller et al. (2014)).

With the following assumption, one can also bound the asymptotic average resource usage.

Assumption 2. Either $\underline{\Delta} = 0$ and $\mu(0) > 0$ or $\underline{\Delta} > 0$ holds.

Theorem 2. Consider Lemma 1 and in addition, suppose that Assumption 2 holds. Then, $\lim_{k\to\infty} t_k = \infty$ and

$$\lim_{k\to\infty} \frac{\sum_{i=0}^{k-1} \mu(\Delta_k)}{t_k} \leq p. \quad (9)$$

Proof. First, assume that $\underline{\Delta} = 0$ and $\mu(0) > 0$. Starting from a finite initial resource level r_0, the controller may pick $\Delta^*(t_k) = 0$ only at finitely many consecutive sampling instances. This is due to the fact that the resource depletes as it has no time to fill up again and $\mu(0) > 0$. After that, some $\Delta^*(t_k) > 0$ must be used. Repeating this argument gives $\lim_{k\to\infty} t_k = \infty$. Second, consider the case $\underline{\Delta} > 0$, for which establishing $\lim_{k\to\infty} t_k = \infty$ is trivial.

Since r_0 is finite, the claim follows readily from (8). □

Lemma 1 and Theorem 2 establish that, by enforcing the resource constraint (3) at each point in time, the proposed self-triggered MPC also achieves the transient average constraint (8) and the asymptotic average constraint (9) on the resource usage. Note that (8) and (9) hold for any scheme that guarantees the resource constraint (3) under the dynamics (2) in closed loop, independent of unforeseen operating conditions. MPC schemes which considered resource usage were formerly discussed in the classical setup in Gommans et al. (2017) and in the self-triggered setup in Henriksson et al. (2015); Zhan et al. (2017); Li et al. (2018). In the former work, a periodically sampled MPC was considered, optimizing the transmission schedule of new inputs over a resource-constrained communication network. An asymptotic average constraint on the transmission frequency $\lim_{k\to\infty} \frac{k}{t_k} \leq q$ was guaranteed, simply by imposing the same average resource usage over each sampling period. In the latter works, the trivial upper bound on the sampling rate $\frac{k}{t_k} \leq \frac{1}{\underline{\Delta}}$ (which holds for the proposed scheme as well) holds by design. In addition, these schemes achieve in the limit $\lim_{k\to\infty} \frac{k}{t_k} = \frac{1}{\Delta}$.

The approach proposed in this paper generalizes the incorporation of average resource constraints into resource-aware MPC in two directions: First, the proposed approach provides a higher flexibility in the scheduling of sampling instances than previous approaches. In view of (8), an excess of resource usage over *any* finite time horizon (as long as (3) is fulfilled) can be compensated for in finite time with a later phase of little resource usage. In Gommans et al. (2017), the resource usage can only be redistributed within the considered sampling period, while in the self-triggered setups, an excess of resource usage is only compensated for in infinite time. A non-trivial transient average constraint as given by (8) is even novel in self-triggered MPC. Second, since an arbitrary resource cost μ is considered, more general constraints can be enforced by virtue of (8) and (9) than only bounding the sampling rate $\frac{k}{t_k}$. Here, the physical interpretation of the resource constraints depends on the cost function, which may be given by the setup or can be chosen by the user. Two examples for the interpretation of the average resource constraints (8) and (9) in a technical context are given below.

Example 1. Consider that updates of state measurements and inputs need to be transmitted between sensors and the controller, respectively the controller and actuators via a communication network fulfilling the token bucket specification. In this case the resource cost is $\mu(\Delta) = c$, $c \in [0, \infty)$, such that an upper bound on the average sampling frequency is enforced: $\frac{k}{t_k} \leq \frac{r_0}{t_k c} + \frac{p}{c}$. Due to the fact that the input changes only at sampling times and the self-triggering setup, the same bound holds for the average transmission frequency of control inputs and state measurements over the communication network.

Example 2. Consider that there is an energy resource, where the resource cost μ is related to the energy that is consumed by a processing unit to solve the MPC optimization problem within a certain sampling interval. Typically in processing units, the energy μ required for computation is inversely proportional to the computation time Δ. Since the MPC may decide on its computation time in the self-triggered setup, one may place a constraint on the average power consumption $\frac{\sum \mu}{t}$ of the control: $\lim_{k\to\infty} \frac{\sum \mu(\Delta_k)}{t_k} \leq p$. Constraining the asymptotic power consumption is especially meaningful in battery-operated devices, e.g., drones, where the control unit may account for a substantial part of the overall energy usage.

Remark 7. Similar concepts of ensuring average constraints by pointwise-in-time constraints are also known in the literature on economic MPC (Angeli et al. (2012); Müller et al. (2014)). These approaches use a classical periodically activated MPC to control a discrete-time system.

5. ENSURING RECURSIVE FEASIBILITY AND CONVERGENCE

A well-established way to theoretically guarantee recursive feasibility and stability in MPC is to add a terminal equality constraint to the MPC optimization problem. Although simple, this approach is in general restrictive and leads to a smaller feasible set compared to more advanced methods, e.g., the terminal set method. Nonetheless, we use this formulation here in a first step to provide valuable insights into the requirements that the resource must fulfill in order to guarantee recursive feasibility and convergence.

For the terminal equality constraint setup, the MPC optimization problem is denoted by $\mathcal{P}(x(t_k), r_k)$ and reads

$$\begin{aligned} V^*(x(t_k), r_k) := \min_{\substack{\boldsymbol{x}(t_k), \boldsymbol{r}(t_k)\\ \boldsymbol{u}(t_k), \boldsymbol{\Delta}(t_k)}} \; & V(\boldsymbol{x}(t_k), \boldsymbol{r}(t_k), \boldsymbol{u}(t_k), \boldsymbol{\Delta}(t_k)) \\ \text{s.t. } & (7a) - (7e) \\ & x_N(t_k) = 0, \; r_N(t_k) \geq 0. \end{aligned} \quad (10)$$

Note that constraints on the terminal process state and resource level (10) were added, as compared to the MPC optimization problem defined in Section 3. We denote by $\mathcal{XR}$ the set of all process states and resource levels such that $\mathcal{P}(x, r)$ is feasible.

Next, we analyze the analytic properties of the proposed control scheme. To this end, let us first define $\mathbb{D}$ as the set of sampling intervals which fulfill

$$\mathbb{D} := \{\Delta \in [0, \infty) | p\Delta - \mu(\Delta) \geq 0\}. \tag{11}$$

Assumption 3. The set $\mathbb{D}$ is nonempty.

In other words, there exists at least one sampling interval such that the resource fills up to a level where sampling and changing the input is possible.

Assumption 4. The minimum and maximum sampling intervals are such that there exists a $\Delta \in \mathbb{D}$ which fulfills

$$\underline{\Delta} \leq \Delta \leq \overline{\Delta}.$$

Theorem 3. Suppose that $(x(0), r_0) \in \mathcal{XR}$ and that Assumptions 3 and 4 hold. Then $\mathcal{P}(x(t_k), r_k)$ under application of Algorithm 1 is feasible for all $k \in \mathbb{I}_{\geq 0}$.

Proof. Assume that $\mathcal{P}$ was feasible at time t_k. Denote $\tilde{\Delta} \in \mathbb{D}$ a sampling interval that fulfills Assumption 4. Then at the next sampling instant $t_{k+1} = t_k + \Delta_0^*(t_k)$, consider the candidate input and inter-sampling period sequences

$$\begin{aligned} \tilde{\boldsymbol{u}}(t_{k+1}) &= \{u_1^*(t_k), \ldots, u_{N-1}^*(t_k), 0\} \\ \tilde{\boldsymbol{\Delta}}(t_{k+1}) &= \{\Delta_1^*(t_k), \ldots, \Delta_{N-1}^*(t_k), \tilde{\Delta}\}, \end{aligned} \tag{12}$$

which result in the state and resource level sequences

$$\begin{aligned} \tilde{\boldsymbol{x}}(t_{k+1}) &= \{x_1^*(t_k), \ldots, x_N^*(t_k), \phi(x_N^*(t_k), 0, \tilde{\Delta})\} \\ \tilde{\boldsymbol{r}}(t_{k+1}) &= \{r_1^*(t_k), \ldots, r_N^*(t_k), \\ &\qquad \min\{r_N^*(t_k) + p\tilde{\Delta} - \mu(\tilde{\Delta}), \overline{r}\}\}. \end{aligned} \tag{13}$$

Due to $x_N^*(t_k) = 0$ and $f(0,0) = 0$, we have $\phi(x_N^*(t_k), 0, \tilde{\Delta}) = x_N^*(t_k) = 0$. Furthermore, $r_N^*(t_k) + p\tilde{\Delta} - \mu(\tilde{\Delta}) \geq r_N^*(t_k) \geq 0$ according to (11), such that (10) is fulfilled by these sequences. The remaining constraints are also fulfilled since $0 \in \mathbb{U}$ and $\underline{\Delta} \leq \tilde{\Delta} \leq \overline{\Delta}$. □

Before we comment on convergence, we first state the following intermediate result on the sampled-data cost.

Lemma 4. It holds that $\lambda(0, 0, \Delta) = 0$ for all $\Delta \in [0, \infty)$.

Proof. With the zero initial condition, zero input and $f(0, 0) = 0$, we have from (5) that $z(t) = 0$ for all $t \in [0, \Delta]$. From the definition of the sampled-data cost (6) and positive definiteness of ℓ, we obtain the statement. □

Theorem 5. Suppose that $(x(0), r_0) \in \mathcal{XR}$ and that Assumptions 2, 3 and 4 hold. Then, the closed-loop process state under application of Algorithm 1 with the optimization problem $\mathcal{P}$ converges to 0, i.e., $x(t) \to 0$ for $t \to \infty$.

Proof. Consider again the feasible sequences (12) and (13). Consider the difference of value functions at t_{k+1} and t_k

$$\begin{aligned} &V^*(x(t_{k+1}), r_{k+1}) - V^*(x(t_k), r_k) \\ &\leq V(\tilde{\boldsymbol{x}}(t_{k+1}), \tilde{\boldsymbol{r}}(t_{k+1}), \tilde{\boldsymbol{u}}(t_{k+1}), \tilde{\boldsymbol{\Delta}}(t_{k+1})) - V^*(x(t_k), r_k) \\ &= \sum_{i=1}^{N-1} \lambda(x_i^*(t_k), u_i^*(t_k), \Delta_i^*(t_k)) + \underbrace{\lambda(0, 0, \tilde{\Delta})}_{=0 \text{ (Lemma 4)}} \\ &- \lambda(x(t_k), u_0^*(t_k), \Delta_0^*(t_k)) - \sum_{i=1}^{N-1} \lambda(x_i^*(t_k), u_i^*(t_k), \Delta_i^*(t_k)) \\ &= -\lambda(x(t_k), u_0^*(t_k), \Delta_0^*(t_k)). \end{aligned} \tag{14}$$

Using an induction on (14) yields

$$\lim_{k\to\infty} V^*(x(t_k), r_k) \leq V^*(x(0), r_0) - \sum_{k=0}^{\infty} \lambda(x(t_k), u_0^*(t_k), \Delta_0^*(t_k)).$$

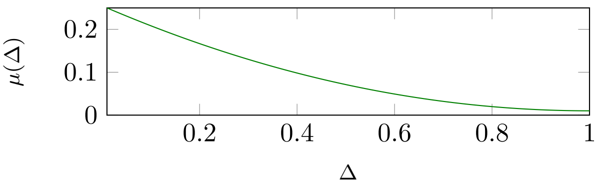

Figure 1. Energy cost function μ in the numerical example.

With $x(t)$ and $u(t)$, the closed-loop state and input trajectories resulting from an application of Algorithm 1 with the optimization problem $\mathcal{P}$, we obtain further, using the definition of the sampled-data cost (6),

$$\begin{aligned} \lim_{k\to\infty} V^*(x(t_k), r_k) &\leq V^*(x(0), r_0) - \sum_{k=0}^{\infty} \int_{t_k}^{t_{k+1}} \ell(x(\tau), u(\tau))\mathrm{d}\tau \\ &= V^*(x(0), r_0) - \lim_{k\to\infty} \int_0^{t_{k+1}} \ell(x(\tau), u(\tau))\mathrm{d}\tau. \end{aligned} \tag{15}$$

Recall that from Theorem 2, it holds that $\lim_{k\to\infty} t_k = \infty$. Since $V^*(x(\infty), r_\infty) \geq 0$ due to the positive definite ℓ, and since $V^*(x(0), r_0)$ is finite, we have from (15) that

$$\int_0^\infty \ell(x(\tau), u(\tau))\mathrm{d}\tau$$

is also finite. We have that ℓ is continuous and positive definite, and $x(\cdot)$ is absolutely continuous. Further, $x(\cdot)$ and $\dot{x}(\cdot)$ are bounded from Assumption 1. Using Barbalat's Lemma (Khalil (2002)), we conclude that $x(t) \to 0$ as $t \to \infty$. □

6. NUMERICAL EXAMPLE

For numerical simulation, we consider a double integrator

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \ x(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

with the input constraint $u(t) \in [-2, 2]$. The MPC optimization problem $\mathcal{P}$ with terminal equality constraint and $N = 20$ is used here. The process cost penalizes quadratically the state and input

$$\ell(x, u) = x^\top \mathrm{diag}(100, 100)x + u^2. \tag{16}$$

The considered resource pertains to Example 2 in Section 4, where the resource cost μ is related to the energy expended using a certain sampling interval Δ. The minimum and maximum inter-sampling periods allowed by the processing unit are $\underline{\Delta} = 0.01\,\mathrm{s}$ and $\overline{\Delta} = 1\,\mathrm{s}$. Within these bounds, the energy cost is described in this example by

$$\mu(\Delta) = 0.2449(\Delta - \underline{\Delta})^2 - 0.4848(\Delta - \underline{\Delta}) + 0.25.$$

A plot of this function can be found in Figure 1, where it can be seen that lowering the sampling time leads to a quadratically increasing energy cost. The resource allows a maximum level of $\overline{r} = 0.5$ and is charged at a rate of $p = 0.5$. With these parameters, $p\Delta - \mu(\Delta) = 0$ is fulfilled for $\Delta \approx 0.176$, such that Assumptions 2, 3 and 4 hold. The energy resource is full at the initial time, i.e., $r_0 = 0.5$.

In Figure 2, one can see the resulting process state and input trajectories. The states and the input converge to the origin. After initial feasibility, recursive feasibility and convergence are guaranteed by Theorems 3 and 5.

Figure 3 shows the sampling intervals chosen by the control scheme and the resulting resource costs and resource levels

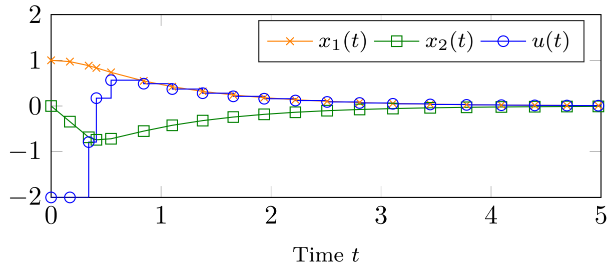

Figure 2. Process state and input trajectories.

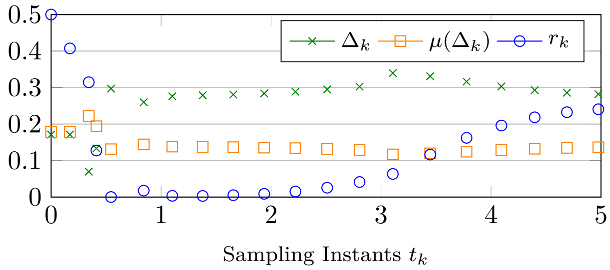

Figure 3. Chosen sampling intervals and resulting costs.

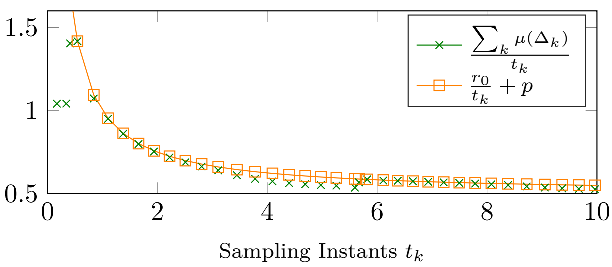

Figure 4. Average resource cost and theoretical limit.

at the sampling instants. To compensate the initial condition, one can see that the controller favors shorter intervals over a couple of sampling instants, at the expense of higher resource costs. By this measure, a high performance with respect to the process cost (16) can be achieved. However, it is only possible to use these very short inter-sampling periods until the stored resources are used up. After approximately 0.5 s, longer intervals must be used to comply with the resource constraint (3), although the plant is still far from the set point. After 2.5 s, the plant has almost converged and longer sampling intervals are used afterwards, such that the resource level rises again.

A set point change from $[0, 0]^\top$ to $[1, 0]^\top$ at 5 s was additionally implemented to demonstrate that the bound given by (8) holds whenever the MPC optimization problem is recursively feasible, independent of set point changes or other unforeseen operating conditions. In Figure 4, the cumulated average resource cost from an application of the MPC is depicted, together with its theoretical limit as given by Lemma 1. We observe that especially right after the initial time and the set point change, the theoretical bound is very tight. After the plant has converged, the difference becomes larger. This complies very much with the discussion in Section 4: because less resources were spend at other times (when the plant is close to the set point), it is possible to spend more when desired (right after a set point change). In other words, using less resources at times enables a short-time excess of resource usage, all while obeying transient constraints on the average usage.

7. OUTLOOK

Future work could investigate a way to incorporate state constraints in closed loop despite the sampled-data formulation of the optimization problem. In a similar direction, also a way to handle disturbances on the process could be studied. Furthermore, one might imagine that the resource cost is also state-dependent, which would allow the user to incorporate more advanced resource constraints.

REFERENCES

Angeli, D., Amrit, R., and Rawlings, J.B. (2012). On average performance and stability of economic model predictive control. *IEEE Trans. Automatic Control*, 57(7), 1615–1626.

Barradas-Berglind, J., Gommans, T., and Heemels, W. (2012). Self-triggered MPC for constrained linear systems and quadratic costs. In *Proc. Conf. on Nonlinear Model Predictive Control*.

Bernardini, D. and Bemporad, A. (2012). Energy-aware robust model predictive control based on noisy wireless sensors. *Automatica*, 48(1), 36 – 44.

Eqtami, A., Dimarogonas, D.V., and Kyriakopoulos, K.J. (2010). Event-triggered control for discrete-time systems. In *Proc. American Control Conf.*, 4719–4724.

Faulwasser, T. (2012). *Optimization-based solutions to constrained trajectory tracking and path-following problems*. Shaker, Aachen, Germany.

Findeisen, R. (2006). *Nonlinear Model Predictive Control: A Sampled-Data Feedback Perspective*. VDI Verlag, Düsseldorf, Germany.

Gommans, T., Theunisse, T.A.F., Antunes, D., and Heemels, M. (2017). Resource-aware MPC for constrained linear systems: Two rollout approaches. *Journal of Process Control*, 51, 68–83.

Henriksson, E., Quevedo, D.E., Peters, E.G.W., Sandberg, H., and Johansson, K.H. (2015). Multiple-loop self-triggered model predictive control for network scheduling and control. *IEEE Trans. Control Systems Technology*, 23(6), 2167–2181.

Khalil, H.K. (2002). *Nonlinear Systems*. Prentice Hall, New Jersey, 3 edition.

Li, H., Yan, W., and Shi, Y. (2018). Triggering and control codesign in self-triggered model predictive control of constrained systems: With guaranteed performance. *IEEE Trans. Automatic Control*, 63(11), 4008–4015.

Li, H. and Shi, Y. (2014). Event-triggered robust model predictive control of continuous-time nonlinear systems. *Automatica*, 50(5), 1507 – 1513.

Linsenmayer, S. and Allgöwer, F. (2018). Performance oriented triggering mechanisms with guaranteed traffic characterization for linear discrete-time systems. In *Proc. European Control Conf.*, 1474–1479.

Mayne, D., Rawlings, J., Rao, C., and Scokaert, P. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6), 789 – 814.

Müller, M.A., Angeli, D., and Allgöwer, F. (2014). Transient average constraints in economic model predictive control. *Automatica*, 50(11), 2943 – 2950.

Qin, J. and Badgwell, T. (2003). A survey of industrial model predictive control technology. *Control engineering practice*, 11, 733–764.

Tanenbaum, A.S. and Wetherall, D.J. (2011). *Computer Networks*. Pearson, 5th edition.

Varutti, P., Kern, B., Faulwasser, T., and Findeisen, R. (2009). Event-based model predictive control for networked control systems. In *Proc. Conf. on Decision and Control*, 567–572.

Wildhagen, S., Müller, M.A., and Allgöwer, F. (2019). Predictive control over a dynamical token bucket network. *IEEE Control Systems Letters*, 3(4), 859–864.

Zhan, J., Li, X., and Jiang, Z. (2017). Self-triggered robust output feedback model predictive control of constrained linear systems. In *American Control Conf.*, 3066–3071.