Self-triggered MPC with Performance Guarantee using Relaxed Dynamic Programming ⋆

Liang Lu [a], Jan M. Maciejowski [b]

[a] *School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.*

[b] *Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, U.K.*

Abstract

This paper presents a self-triggered MPC controller design strategy for linear systems with state and input constraints. Based on the so-called relaxed dynamic programming inequality, the synthesis procedure determines both the updated MPC control action and the next triggering time. The resulting self-triggered MPC control law preserves stability and constraint satisfaction and also satisfies a certain specified performance requirement without requiring stabilizing terminal constraints. A robust self-triggered MPC scheme, based on tube-MPC idea, is also presented for linear systems with persistent bounded additive disturbances. Simulation examples illustrate the effectiveness of our proposed self-triggered MPC scheme.

Key words: Self-triggered Control, Model Predictive Control, Relaxed Dynamic Programming, Tube-based MPC.

1 Introduction

Model predictive control (MPC) is a powerful control technique for constrained systems [11,36,37,44]. In MPC, an open-loop optimal control problem is solved at each sampling time after a measurement update. Classical MPC can be termed "time-driven" as the control input profile is optimised repeatedly at a chosen fixed interval; at each sampling time a sequence of control values is computed, but usually only the first component is applied to the system, while the other components are discarded. Though it was originally developed to solve multivariable constrained control problems found in refineries and chemical plants [43], its application range has been significantly broadened to include the fields of automotive powertrains [41], power grids [39], and water distribution systems [16,32]. One of the potential bottlenecks of MPC, especially for large-scale systems, is its large computational requirement due to the need to solve an optimization problem on-line at each sampling time. In distributed MPC the computation is distributed over several controllers; in this case the communication required between these controllers at each sampling time can also be substantial, and can limit the amount of distribution that is possible [13]. Even in non-distributed MPC (and other centralised control schemes) the communication of many measurements and many actuator signals at each sampling time may pose a considerable burden.

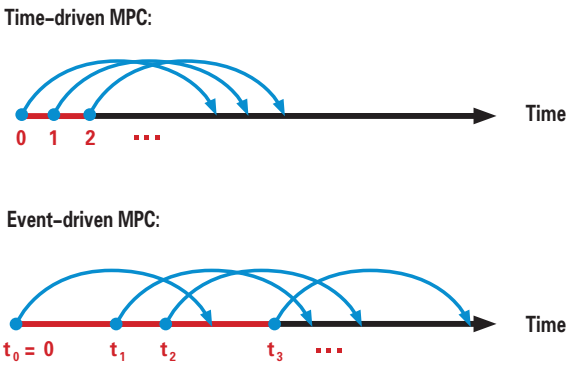

Figure 1. Time-driven MPC vs. Event-driven MPC.

To overcome the above-mentioned limitations of clas-

⋆ This paper was not presented at any IFAC meeting. This work was initially carried out when the first author was a Researcher at the Linköping University, Sweden and KAIST, South Korea. Corresponding author Liang Lu. Tel. +86-15524476228.

Email addresses: liangup@gmail.com (Liang Lu), jmm@eng.cam.ac.uk (Jan M. Maciejowski).

sical time-driven MPC, so-called "event-driven" MPC has been proposed and has gained a lot of attention in recent years. In event-driven MPC, instead of equally-spaced activation over time, the control input update is performed only when it is necessary or something significant happens (see Fig. 1), according to a triggering scheme designed together with the control computation. Event-driven MPC can take the form of "event-triggered" or "self-triggered". In event-triggered MPC, a prescribed triggering condition, based on incoming measurements, is constantly checked and if it is violated the MPC update process is triggered. On the other hand, in self-triggered MPC, at a triggering time both the updated MPC control actions and the next triggering time are determined. The key issue in choosing between the event-triggered and self-triggered MPC implementations is whether the system states can be continuously monitored [29]. If the system model has significant uncertainties or frequent disturbances, the event-triggered strategy may be more appropriate.

Most of the existing event- and self-triggered control results are for continuous-time systems, and utilize the input-to-state stability (ISS) property to derive appropriate error thresholds for the triggering (see [47,4,29,45], and the references therein). In [18], the authors extended the continuous-time event-triggered control strategy to discrete-time systems and applied it to compute the inter-sample time needed for an unconstrained MPC problem to guarantee a certain level of performance by maintaining measured errors within some bound set by the ISS conditions. Later in [19,20], this idea of checking the ISS property with respect to measured errors was extended to nonlinear discrete-time distributed systems by utilizing a stabilizing MPC setup with terminal cost and terminal constraints; the resulting event-triggered MPC scheme can achieve ultimate boundedness of the closed-loop system. More specifically, in [19], the triggering condition was implemented by viewing the interactions among the subsystems as bounded disturbances, while in [20], these neighbouring subsystems' interactions were considered explicitly in the local triggering conditions. All these results on the event-triggered strategy need regular monitoring of the states, and no performance guarantee is obtained. In the case that regular monitoring is not feasible or desirable, the self-triggered strategy is more appropriate.

The paper [30] presented a self-triggered unconstrained MPC strategy for multiple-loop networked control systems by analysing a tradeoff cost function that depends on the performance cost as well as the inter-sampling cost to schedule the sensor sampling, so as to guarantee collision-free transmissions. The paper [7] derived a self-triggered MPC strategy for discrete-time linear systems with terminal cost and terminal constraints in the MPC setup, which guarantees a certain sub-optimality criterion and constraint satisfaction. In that paper the control law remains constant during the non-triggering interval, which often results in consecutive triggering updates because keeping the control law constant can exacerbate model mismatch. More recently, the paper [25] derived a self-triggered MPC strategy with terminal cost and terminal constraints, with 'sparse' control signals. The idea is either to hold the control signal computed at the triggering time for as long as possible, or to zero the control actions after the triggering time, while still guaranteeing stability and some performance objective.

By contrast, in this paper we make full use of the (non-constant) 'tail' of the control sequence computed by MPC at the triggering times, and we maintain stability and performance requirements without terminal constraints or penalties. The key technical tool which enables us to ensure asymptotic stability without terminal constraints is the so-called relaxed dynamic programming inequality (RDP) [26,27,34]. Earlier pioneering works on the study of feasibility and stability for MPC without terminal constraints can be found in [42,46]. Similar ideas have been applied with time-varying control horizons [22], for shortening the prediction horizon [40] and for early termination of distributed MPC algorithms [23,21] with suboptimality and stability guarantees. However, to the authors' knowledge, this is the first paper which ensures feasibility, stability and guaranteed performance for MPC without terminal constraints, and exploits the relaxed dynamic programming idea, in the context of event-driven MPC.

Recently, a class of robust self-triggered MPC schemes was developed in combination with tube-based MPC [5,12]. (See [44, Chapter 3] for an account of tube-based MPC.) The same triggering condition was adopted in [15] for stochastic self-triggered MPC and in [35] for min-max robust self-triggered MPC formulation. However, the evaluation of the triggering condition in all of these requires the solution of a number of quadratic programs with complex time-varying constraint tightening at every triggering time, the number required being equal to the number of time steps between triggering times. The time-varying constraint tightening significantly slows down the computation of each QP problem. Thus, all of these methods are computationally very expensive at each triggering time, and incur the risk that the next triggering time should occur before its computation is complete. Furthermore, since these schemes apply open-loop control between triggering times, the uncertainty due to disturbances grows exponentially along the prediction horizon, requiring more conservative and more elaborate constraint tightening — typically reducing the domain of attraction as a result.

Inspired by the "assumed trajectory" idea in [17,9], in this paper we also obtain a robust self-triggered MPC scheme which can deal with bounded persistent additive disturbances, by extending the RDP-based self-triggered MPC scheme. We propose a novel closed-loop control policy in between triggering times; we take the measure-

ment of the system state at the latest triggering time and keep it constant as the "assumed trajectory" until the next triggering time. We apply "tube-like" state feedback between this assumed trajectory (as the real trajectory is not measured in between the triggering times) and the nominal trajectory at each sampling time, which does not involve any optimisation. This feedback reduces the rate of growth of uncertainty between triggering times, and the resulting deviation between the assumed trajectory and the real one can be bounded. In addition, in our proposed method only a single QP needs to be solved in order to evaluate the next triggering time, and we need less complex constraint tightening than occurs in [5,12,15]. Furthermore, the constraint tightening computation needs to be done only once, offline, rather than to enforce the time-varying constriant tightening online.

The organization of the paper is as follows. In Section 2, we provide some definitions and preliminary results that will be used in this paper and formulate the self-triggered MPC problem that will be dealt with in this paper. In Section 3, the RDP based approach is adopted to prolong the inter-triggering times to a maximum extent in order to minimize the number of MPC updates. A systematic self-triggering scheme is provided. In Section 4, we extend the RDP based approach to linear systems with bounded disturbances. Illustrative examples for both the deterministic and the robust cases are presented in Section 5, and some conclusions are given in Section 6.

Notation: Let $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{Z}$ and $\mathbb{Z}_+$ denote the set of real numbers, non-negative real numbers, integers, and non-negative integers, and let $\mathbb{Z}_{[a,b)}$ and $\mathbb{Z}_{[a,b]}$ denote the sets $\{k \in \mathbb{Z} \mid a \leq k < b\}$ and $\{k \in \mathbb{Z} \mid a \leq k \leq b\}$, respectively. Throughout this paper, t denotes sampling time, and k denotes the count of time-steps in prediction horizon. Given two sets $\mathcal{X}$, $\mathcal{Y} \subseteq \mathbb{R}^n$. The Minkowski set addition is defined by $\mathcal{X} \oplus \mathcal{Y} := \{x + y | x \in \mathcal{X}, y \in \mathcal{Y}\}$. The Pontryagin set difference is defined by $\mathcal{X} \ominus \mathcal{Y} := \{z | z \oplus \mathcal{Y} \subseteq \mathcal{X}\}$. The multiplication of a matrix and a set is defined by $X\mathcal{Y} = \{Xy | y \in \mathcal{Y}\}$.

2 Problem Setup

Consider the linear system

$$x(t+1) = Ax(t) + Bu(t), \quad x(0) = \bar{x}, \tag{1}$$
$$y(t) = Cx(t) + Du(t), \tag{2}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ and $y(t) \in \mathbb{R}^p$ are the state, input and (generalized) output at time instant t, respectively. The sets $\mathbb{X} \subseteq \mathbb{R}^n$ and $\mathbb{U} \subseteq \mathbb{R}^m$ represent the sets of state and input constraints containing the origin in their interiors. We assume that (A, B) is stabilizable, the pair (A, C) is observable and matrix D is full column rank. Note that y here contains the *controlled* variables, not the measured variables (which are in x), so that this full-rank assumption is not problematic.

2.1 Infinite-horizon Optimal Control

In order to synthesize an optimal control law, the following infinite-horizon optimal control problem is formulated:

$$\begin{aligned} \min_{\mathbf{u} \triangleq [u^{\mathrm{T}}(0), \cdots, u^{\mathrm{T}}(\infty)]^{\mathrm{T}}} & J^{(\infty)}(\bar{x}, \mathbf{u}) \triangleq \sum_{t=0}^{\infty} \|y(t)\|_2^2, \quad (3) \\ \text{s.t. } & x(t) \in \mathbb{X}, \\ & u(t) \in \mathbb{U}, \\ & x(0) = \bar{x}, \\ & x(t+1) = Ax(t) + Bu(t), \\ & y(t) = Cx(t) + Du(t), \end{aligned}$$

where $\|y(t)\|_2^2$ is the quadratic cost for $t = 0, 1, 2, \cdots, \infty$. $J^{(\infty)}(\bar{x}, \mathbf{u})$ is the infinite horizon cost function for some starting state $\bar{x}$ and control law $\mathbf{u}$ from time 0 to ∞. Let the optimal control law be denoted by $\mathbf{u}^*$ and the corresponding optimal infinite horizon cost function by

$$V^{(\infty)}(\bar{x}) \triangleq J^{(\infty)}(\bar{x}, \mathbf{u}^*).$$

$V^{(\infty)}(\bar{x})$ is the so called "value function". To compute the value function, one has to solve the Bellman's optimality equation, which is in general very difficult due to the computational barrier known as the "curse of dimensionality".

2.2 Finite-horizon MPC Optimization

In MPC, we take a finite horizon $N \in \mathbb{Z}_+$, instead of infinity horizon and solve the following optimization problem repetitively at each sampling time.

$$\begin{aligned} \min_{\mathbf{u} \triangleq [u_0^{\mathrm{T}}, \cdots, u_{N-1}^{\mathrm{T}}]^{\mathrm{T}}} & J^{(N)}(x(t), \mathbf{u}) \triangleq \sum_{k=0}^{N-1} \|y_k\|_2^2, \quad (4) \\ \text{s.t. } & x_k \in \mathbb{X}, \quad k = 1, \ldots, N, \\ & u_k \in \mathbb{U}, \quad k = 0, 1, \ldots, N-1, \\ & x_0 = x(t), \\ & x_{k+1} = Ax_k + Bu_k, \quad k = 0, 1, \ldots, N-1, \\ & y_k = Cx_k + Du_k, \quad k = 0, 1, \ldots, N-1. \end{aligned}$$

The optimization problem (4) is strictly convex and has a unique minimum.

At each sampling time, solving the above optimization problem for a particular x_0 leads to a unique sequence of optimal control moves from

time t to time $t+N-1$, given by $U_N^*(x(t)) = [u_0^{*\mathrm{T}}(x(t)), u_1^{*\mathrm{T}}(x(t)), \ldots, u_{N-1}^{*\mathrm{T}}(x(t))]^{\mathrm{T}}$.

The (finite-horizon) value function is

$$V^{(N)}(x(t)) \triangleq J^{(N)}(x(t), U_N^*(x(t))),$$

as compared to the corresponding infinite horizon value function denoted as

$$V^{(\infty)}(x(t)) \triangleq J^{(\infty)}(x(t), U_\infty^*(x(t))).$$

The open-loop optimal control solution is turned into a feedback control strategy by applying the first control move of the optimal control sequence $U_N^*(x(t))$ to the system and solving the same problem again with a feedback update of the state at next sample time. As the reference is time varying, the MPC control law will be represented by

$$u(t) = \mu(x(t)) := u_0^*(x(t)). \tag{5}$$

Consequently, the closed-loop system becomes

$$x(t+1) = Ax(t) + B\mu(x(t)), \tag{6}$$

$$y_\mu(t) = Cx(t) + D\mu(x(t)). \tag{7}$$

2.3 Recursive Feasibility

As the system has constraints, this subsection is devoted to characterize the region from which the receding horizon optimal control problem is feasible, and if the horizon length N is chosen long enough the evolution trajectories will always remain in this feasible region under the MPC control law (5).

First, we introduce some definitions and notations to formalize recursive feasibility.

Definition 1 *A control sequence* $\mathbf{u} = (u(0), u(1), \ldots, u(N-1))$ *is said to be* admissible *for* $x(0) \in \mathbb{X}$, *if* $Ax(t) + Bu(t) \in \mathbb{X}$ *for* $(x(t), u(t)) \in \mathbb{X} \times \mathbb{U}$ *holds for all* $t \in \{0, 1, \ldots, N-1\}$. *The set of all admissible control sequences of length* N *is denoted by* $\mathcal{U}^N(x(0))$.

The feasible region for horizon N is defined as

$$\mathbb{I}_N := \{x \in \mathbb{X} : \mathcal{U}^N(x) \neq \emptyset\}.$$

The region $\mathbb{I}_\infty$ is called *viability kernel* [10]. It characterizes the set of the infinite horizon feasible initial conditions of system (1) subject to input and state constraints. From the definition, we can immediately get the following properties:

1) $\mathbb{I}_0 \supseteq \mathbb{I}_1 \supseteq \cdots \supseteq \mathbb{I}_\infty$,

2) $\mathbb{I}_\infty := \cap_{N \in \mathbb{Z}_+} \mathbb{I}_N$.

The sequence of feasible sets $\mathbb{I}_N$'s becomes *stationary*, if there exists $N_0 \in \mathbb{Z}_+$, such that $\mathbb{I}_N = \mathbb{I}_{N_0}$ holds for all $N \geq N_0$.

Definition 2 *A set* $\mathcal{P} \subseteq \mathbb{R}^n$ *is called a* controlled positively invariant (CPI) set *or a* viable set *for system (1), if* $\mathcal{P} \subseteq \mathbb{X}$ *and for all* $x \in \mathcal{P}$, *there is a* $u \in \mathbb{U}$, *such that* $Ax + Bu \in \mathcal{P}$ *holds.*

$\mathbb{I}_\infty$ is also called *maximal positively invariant (PI) set*, which includes all the possible CPI set $\mathcal{P}$, i.e. $\mathcal{P} \subseteq \mathbb{I}_\infty$.

Definition 3 *A set* $\mathcal{P}$ *is called* RH N-invariant *or* recursively feasible with respect to a horizon $N \in \mathbb{Z}_+$ *if* $\mathcal{P}$ *is a CPI set for the closed-loop system (6) under the MPC controller (5) with a receding horizon (RH)* N, *i.e.*

$$x(0) \in \mathcal{P} \Rightarrow x(t) \in \mathcal{P}, \quad \forall t \in \mathbb{Z}_+.$$

The following proposition is from [10], which shows for sufficiently large horizon N, MPC controller will generate recursive feasibility on the whole viability kernel $\mathbb{I}_\infty$. This property is inferred from *stationarity* of the feasible sets $\mathbb{I}_N$'s [28,31].

Proposition 1 *If* $V^{(\infty)}(x(t)) < c$ *holds for some* $c \in \mathbb{R}_+$ *and all* $x(t) \in \mathbb{I}_\infty$, *the feasible sets* $\mathbb{I}_N$*'s become stationary for some* $N_0 \in \mathbb{Z}_+$, *i.e.,* $\mathbb{I}_{N_0} = \mathbb{I}_{N_0+1} = \mathbb{I}_{N_0+2} = \cdots = \mathbb{I}_\infty$.

In order to characterize the feasible region from which the constrained infinite horizon optimal control problem has a finite solution, we make the following assumption.

Assumption 1 *Throughout this paper, we assume the optimal infinite horizon cost* $V^{(\infty)}(x(0))$ *is finite for all* $x(0) \in \mathbb{I}_\infty$, *i.e.,* $\sup V^{(\infty)}(\mathbb{I}_\infty)$ *is finite.*

Given a horizon length N and a positive scalar ν, in order to determine a RH N-invariant set, we define the sub-level S_ν^N of finite horizon value function $V^N(x)$

$$S_\nu^N = \{x \in \mathbb{X} : V^N(x) \leq \nu\}.$$

In our setting, the inequality $V^N(x) \leq V^{(\infty)}(x)$ holds. Hence, S_ν^N contains S_ν^∞.

2.4 Relaxed Dynamic Programming

In this paper, we will use the relaxed dynamic programming result in [34] to develop a triggering condition for

self-triggered MPC to ensure stability and to obtain a performance guarantee in terms of the infinite horizon quadratic cost.

The next proposition is a variant of the main proposition stated in [34,27] for approximating the Bellman's optimality equation based on the finite-horizon value function $V^{(N)}(x(t))$ defined in Section 2.2 and its corresponding optimal control policy $\mu(x(t))$.

Proposition 2 *Let $\nu = V^N(x(0))$. Consider the system (1)-(2) and the feedback control law $\mu : \mathbb{X} \mapsto \mathbb{U}$ given as (5) that satisfies the following inequality*

$$V^{(N)}(x(t)) \geq V^{(N)}(x(t+1)) + \alpha \|y_\mu(t)\|_2^2, \qquad (8)$$

for a given scalar $\alpha \in (0, 1]$ and all $x(t) \in S_\nu^N$. Then,

$$\alpha \sum_{t=0}^{\infty} \|y_\mu(t)\|_2^2 \leq V^{(\infty)}(x(0)), \qquad (9)$$

where $x(t+1)$ and $y_\mu(t)$ are obtained by applying $\mu(x(t))$ to the closed-loop system, i.e., $x(t+1) = Ax(t) + B\mu(x(t))$ and $y_\mu(t) = Cx(t) + D\mu(x(t))$.

Proof. Let the feedback control law $\mu(x(t))$ be computed as (5). Summing (8) over time $t = 0, \ldots, T-1$ yields

$$\alpha \sum_{t=0}^{T-1} \|y_\mu(t)\|_2^2 \leq V^{(N)}(x(0)) - V^{(N)}(x(T)).$$

Since $V^{(N)}(x(T)) \geq 0$, as $T \to \infty$,

$$\alpha \sum_{t=0}^{\infty} \|y_\mu(t)\|_2^2 \leq V^{(N)}(x(0)).$$

As there is no terminal cost in the cost function and as $V^{(N)}(x(0)) : S_\nu^N \to \mathbb{R}_+$ is increasing as N increases, we have $V^{(N)}(x(0)) \leq V^{(\infty)}(x(0))$. Thus,

$$\alpha \sum_{t=0}^{\infty} \|y_\mu(t)\|_2^2 \leq V^{(\infty)}(x(0)).$$

This completes the proof. □

Remark 1 *The inequality (8) is the so-called relaxed dynamic programming (RDP) inequality. By Bellman's equation, the inequality in (8) becomes equality with $\alpha = 1$ (optimal performance). In particular, inequality (8) implies that $V^{(N)}(x)$ is a Lyapunov function on S_ν^N for $\alpha > 0$, which gives recursive feasibility and uniform asymptotic stability on S_ν^N, i.e. S_ν^N is RH N-invariant and contractive under the MPC feedback controller $\mu(x(t))$.*

Remark 2 *Inequality (9) gives a rigorous bound $\alpha \in (0, 1]$ on the sub-optimality of the closed-loop system with respect to infinite-horizon performance. As $N \to \infty$, $V^{(N)}(x(t)) \to V^{(\infty)}(x(t))$ for all $x(t) \in \mathbb{I}_\infty$, which follows from the stationarity of the feasible region $\mathbb{I}_N$. If $\sup V^{(\infty)}(\mathbb{I}_\infty)$ is finite, then S_ν^∞ contains $\mathbb{I}_\infty$ for $\nu \geq \sup V^{(\infty)}(\mathbb{I}_\infty)$, hence S_ν^N also contains $\mathbb{I}_\infty$, which implies that $\mathbb{I}_\infty$ is contained in the stabilizable set. So with Assumption 1, we can expect (8) to hold for all N sufficiently large on the whole $\mathbb{I}_\infty$.*

Throughout this paper, we can reasonably make the following assumption.

Assumption 2 *Given any initial conditions in $\mathcal{I} \subseteq \mathbb{I}_\infty$, the control horizon $N \geq N^*$ is known for a prescribed $\alpha \in (0, 1]$ to satisfy the RDP inequality (8).*

Results on quantitative estimates of N^* and computation of α in the RDP inequality for stabilizing horizons N for various systems can be found in the references [1,2,3,6,48].

While under reasonable condition on the system (1), we can ensure that N^* exists, it may be difficult to compute it a priori. Therefore, our triggering mechanism will detect if N was chosen too small, such that it can be adapted, cf. Remark 7.

2.5 Self-triggered MPC: Sparse and Non-sparse

The main difference between time-driven and event-driven MPC is that, in event-driven MPC, the control updates are performed upon significant occurrences instead of at uniformly spaced sampling times. The following aperiodic self-triggered receding horizon mechanism is suggested with the purpose of alleviating the computational and measurement efforts associated with control computation and state monitoring.

At a sampling time t, we perform the MPC prediction process, and get a sequence of optimal control moves $u_0^*(x(t)), u_1^*(x(t)), \ldots, u_{N-1}^*(x(t))$. In this paper, instead of only implementing the first control move and throwing away the rest, we want to use up the obtained input sequence as much as possible (as illustrated in Fig. 2 (a)). For that, we must determine how long we can follow the calculated open-loop input trajectory while stability can be maintained, constraints satisfied, and certain prescribed performance levels guaranteed.

Define the triggering times $\{t_l \mid l \in \mathbb{Z}_+\}$, which satisfy $t_{l+1} > t_l$ for all $l \in \mathbb{Z}_+$ and $t_{l+1} - t_l < N$. In between the interval $[t_l, t_{l+1})$,

$$u(t) = \tilde{\mu}(t, x(t_l)) := u_{(t-t_l)}^*(x(t_l)), \quad t \in \mathbb{Z}_{[t_l, t_{l+1})}. \qquad (10)$$

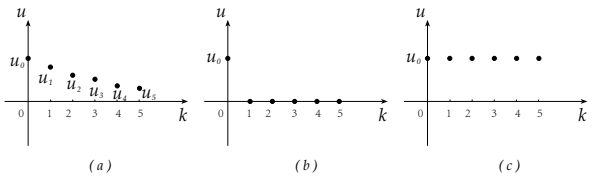

Figure 2. Self-triggering strategies (Non-sparse: (a), sparse: (b) & (c)).

Remark 3 *As $u_0^*(x(t_l)), u_1^*(x(t_l)), \ldots, u_{N-1}^*(x(t_l))$ is a feasible open-loop input trajectory from time t_l to $t_l + N-1$ and we follow this trajectory until the next triggering time t_{l+1} in self-triggered MPC, the input constraint $u(t) \in \mathbb{U}$ for $t \in \mathbb{Z}_{[t_l, t_{l+1})}$ is automatically satisfied.*

The self-triggered MPC implementation scheme in (10) is to follow the open-loop MPC input trajectory until the next triggering time. We recommend this implementation, because it can incorporate more information from the MPC control sequence we compute. However, in some applications, in order to shape sparsity in the control actuators, the synthesis strategies either keep the control signal constant between triggering times or implement only the first control move with no actuation until the next triggering time as plotted in Fig. 2 (b) and Fig. 2 (c). In these situations, we replace the implementation scheme (10) with one of the following two sparse implementations:

(i) To hold:

$$u(t) = \bar{\mu}(t, x(t_l)) := u_0^*(x(t_l)), \quad t \in \mathbb{Z}_{[t_l, t_{l+1})}, \tag{11}$$

(ii) To zero:

$$u(t) = \mathring{\mu}(t, x(t_l)) := \begin{cases} u_0^*(x(t_l)), & t = t_l, \\ 0, & t \in \mathbb{Z}_{[t_l+1, t_{l+1})}, \end{cases} \tag{12}$$

where $u_0^*(x(t_l))$ is obtained from solving the MPC problem (4) by setting $x_0 = x(t_l)$.

Remark 4 *Note that in these sparse implementations, in general, only one control move information in the computed MPC control sequence is used which can introduce more model mismatch, more trade off in the performance and consecutive triggerings are expected. We will investigate these issues in Section 5.*

The problem we want to solve in this paper is formally stated as follows:

Problem 1: At a time instant t_l, compute an optimal receding horizon control sequence according to (4) as $U_N^*(x(t_l))$, and given a performance index $0 < \alpha < 1$, decide the next triggering time t_{l+1} such that t_{l+1} is as large as possible while the closed-loop system of (1) and (10) has the following design properties:

(i) The input and state constraints are satisfied for all $x \in \mathcal{I}$.
(ii) The system (1) and (10) is (uniformly) asymptotically stable in $\mathcal{I}$.
(iii) The system (1), (2) and (10) satisfies the performance requirement

$$\sum_{t=0}^{\infty} \|y_{\tilde{\mu}}(t)\|_2^2 \leq \alpha^{-1} V^{(\infty)}(x(0)) \tag{13}$$

for all closed-loop trajectories of (1) and (10) with $x(0) = \bar{x} \in \mathcal{I}$.

As the control strategy (10) can be replaced by (11) or (12) for the closed-loop trajectories, we use $y_{cl}(t)$, instead of $y_{\tilde{\mu}}(t)$ in the rest of this paper.

3 RDP-Based Approach

In this section, we will revise the relaxed dynamic programming inequality in Proposition 2 so that it can be used within the self-triggered MPC scheme.

At a triggered time instant t_l, self-triggered MPC has to decide both the control law and the next triggering time t_{l+1} such that $t_{l+1} (< t_l + N)$ is as large as possible while satisfying the properties of (i)-(iii) that are stated in Problem 1. Calculation of the next triggering time will be based on the RDP inequality for finite-horizon value function $V^{(N)}(x(t_l))$.

In the self-triggered MPC setting, multiple control moves calculated at time t_l may be implemented before the next control calculation at time t_{l+1} is performed. So we amend the RDP inequality as follows:

$$V^{(N)}(x(t_l)) \geq V^{(N)}(x(t_{l+1})) + \alpha \sum_{t=t_l}^{t_{l+1}-1} \|y_{cl}(t)\|_2^2, \tag{14}$$

where $\sum_{t=t_l}^{t_{l+1}-1} \|y_{cl}(t)\|_2^2$ represents the sum of the running costs at times $t_l, t_l+1, \ldots, t_{l+1}-1$ with the control $u(t)$ applied as in (10). As the value of $V^{(N)}(x(t_{l+1}))$ for the next triggering time is not available at t_l, we will use an upper bound for it. Furthermore, we will also introduce an extra slack variable which collects the slacks due to the introduction of these upper bounds and the slacks inherent from the RDP inequalities before t_l, in order to reduce the conservativeness associated with the using of upper bound. The main theorem of this paper is stated as follows, which proves after all the above mentioned modifications on the RDP inequality, a certain bound of performance, thus asymptotic stability are

still guaranteed. Note that if we set α close to 1, then we obtain a time-driven optimal MPC trajectory, i.e. we compute the optimal MPC controller at each sampling time. However, in self-triggered MPC, we want to relax the performance requirements in order to get the desired properties, like less MPC updates and sparse control implementations, etc.. So we will set $\alpha \in (0,1)$ in the following, which means that we use suboptimal MPC.

Theorem 1 *Let $\nu = V^N(x(0))$. If an upper bound $\bar{V}^{(N)}(x(t))$ can be found for $t \in \{t_l \mid l \in \mathbb{Z}_+\}$ such that*

$$\bar{V}^{(N)}(x(t)) \geq V^{(N)}(x(t)) \tag{15}$$

and

$$V^{(N)}(x(t_l)) - \bar{V}^{(N)}(x(t_{l+1})) \geq e(t_l) + \alpha \sum_{t=t_l}^{t_{l+1}-1} \|y_{cl}(t)\|_2^2, \tag{16}$$

are satisfied for a given scalar $\alpha \in (0,1)$ and all $x(t) \in S_\nu^N$, where the sequence $\{e(t_l)\}$ is

$$e(t_l) = e(t_{l-1}) + \alpha \sum_{t=t_{l-1}}^{t_l-1} \|y_{cl}(t)\|_2^2 + \bar{V}^{(N)}(x(t_l)) - \bar{V}^{(N)}(x(t_{l-1})) \tag{17}$$

for all $l \geq 2$ and

$$e(t_1) = \alpha \sum_{t=t_0}^{t_1-1} \|y_{cl}(t)\|_2^2 + \bar{V}^{(N)}(x(t_1)) - V^{(N)}(x(t_0)),$$

and $e(t_0) = 0$, then,

$$\alpha \sum_{t=t_0}^{\infty} \|y_{cl}(t)\|_2^2 \leq V^{(\infty)}(x(t_0)). \tag{18}$$

Furthermore, if $V^{\infty}(x(t_0)) \leq \bar{V} < \infty$, then

$$\lim_{t \to \infty} y_{\tilde{\mu}}(t) = 0, \tag{19}$$

i.e., $\lim_{t\to\infty} x(t) = 0$.

Proof. At time $t_T \in \{t_l \mid l \in \mathbb{Z}_+\}$, we proceed to derive $e(t_T)$ by induction from (17) as

$$\begin{aligned} e(t_T) &= e(t_{T-1}) + \alpha \sum_{t=t_{T-1}}^{t_T-1} \|y_{cl}(t)\|_2^2 + \bar{V}^{(N)}(x(t_T)) \\ &\quad - \bar{V}^{(N)}(x(t_{T-1})) = \cdots = e(t_1) + \alpha \sum_{t=t_1}^{t_T-1} \|y_{cl}(t)\|_2^2 \\ &\quad + \bar{V}^{(N)}(x(t_T)) - \bar{V}^{(N)}(x(t_1)) \\ &= \alpha \sum_{t=t_0}^{t_T-1} \|y_{cl}(t)\|_2^2 + \bar{V}^{(N)}(x(t_T)) - V^{(N)}(x(t_0)), \end{aligned}$$

which implies

$$\alpha \sum_{t=t_0}^{t_T-1} \|y_{cl}(t)\|_2^2 = e(t_T) - \bar{V}^{(N)}(x(t_T)) + V^{(N)}(x(t_0)). \tag{20}$$

From (16) and because $\alpha \sum_{t=t_l}^{t_{l+1}-1} \|y_{cl}(t)\|_2^2 \geq 0$, we have

$$e(t_T) \leq V^{(N)}(x(t_T)) - \bar{V}^{(N)}(x(t_{T+1})).$$

Insert this into (20) gives

$$\begin{aligned} \alpha \sum_{t=t_0}^{t_T-1} \|y_{cl}(t)\|_2^2 &\leq V^{(N)}(x(t_0)) - \bar{V}^{(N)}(x(t_T)) \\ &\quad + V^{(N)}(x(t_T)) - \bar{V}^{(N)}(x(t_{T+1})) \end{aligned}$$

Because

$$V^{(N)}(x(t_T)) - \bar{V}^{(N)}(x(t_T)) \leq 0,$$

we get

$$\begin{aligned} \alpha \sum_{t=t_0}^{t_T-1} \|y_{cl}(t)\|_2^2 &\leq V^{(N)}(x(t_0)) - \bar{V}^{(N)}(x(t_{T+1})) \\ &\leq V^{(N)}(x(t_0)) - V^{(N)}(x(t_{T+1})) \\ &\leq V^{(N)}(x(t_0)) \leq V^{(\infty)}(x(t_0)). \end{aligned} \tag{21}$$

As $t_T \to \infty$, we get (18). Furthermore, given the boundedness of $V^{(\infty)}(x(t_0))$, and the positive definiteness of the term $\|y_{cl}(t)\|_2^2$, we get immediately $\lim_{t\to\infty} y_{cl}(t) = 0$, i.e. $\lim_{t\to\infty} x(t) = 0$.

This completes the proof. □

Remark 5 *The slack variable $e(t_l)$ in the above theorem is composed of two parts: The first part comes from the slack of the RDP inequality (16) in the previous triggering time, which is*

$$e(t_{l-1}) + \alpha \sum_{t=t_{l-1}}^{t_l-1} \|y_{cl}(t)\|_2^2 + \bar{V}^{(N)}(x(t_l)) - V^{(N)}(x(t_{l-1})).$$

The second is due to the conservativeness in the overbound estimate of the value function in the RDP inequality (16) two triggered steps back:

$$V^{(N)}(x(t_{l-1})) - \bar{V}^{(N)}(x(t_{l-1})).$$

Remark 6 *From (15) and (16), the two parts in Remark 5 are all "≤ 0", thus we have $e(t_l) \leq 0$, which reduces the conservativeness associated with using the upper bound in the inequality (16). Due to the introduction of the slack term $e(t_T)$, the self-triggering mechanism will accept a longer period in between triggering times t_l and t_{l+1} even if the predictive decay of $V^{(N)}(x(t_l)) - \bar{V}^{(N)}(x(t_{l+1}))$ is not sufficiently large provided the Lyapunov function $V^{(N)}(x(t))$ already accumulated enough decay in previous time steps. At the same time, from (21), we always get sufficient decrease of $V^{(N)}(x(t_{T+1}))$ with respect to $V^{(N)}(x(t_0))$, i.e. $x(t_{T+1}) \in S_\nu^N$.*

3.1 RDP-based Triggering Scheme

At an MPC update time $t_l \in \mathbb{Z}_+$ with $l \in \mathbb{Z}_+$, we compute the MPC control update according to (4), and implement the control strategy selected from (10), (11) or (12). The next MPC update time t_{l+1} can be calculated by

$$t_{l+1} = t_l + \mathcal{N}_{t_l}(x(t_l)), \tag{22}$$

where the inter-triggering interval $\mathcal{N}_{t_l}(x(t_l))$ is given by

$$\mathcal{N}_{t_l}(x(t_l)) \triangleq \max\{N_{t_l} \in \mathbb{Z}_{[1,N-1]}\} \tag{23}$$

$$\text{s.t.} \quad V^{(N)}(x(t_l)) - \bar{V}^{(N)}(x(t_l + N_{t_l})) \geq e(t_l) + \alpha \left(\sum_{t=t_l}^{t_l+N_{t_l}-1} \|y_{cl}(t)\|_2^2 \right). \tag{24}$$

In order to calculate the upper bound $\bar{V}^{(N)}(x(t_l+N_{t_l}))$, we apply a "shifted" input sequence $\bar{U}_N(x(t_l + N_{t_l})) = [\, {u^*_{N_{t_l}}}^{\mathrm{T}}(x(t_l)), \ldots, {u^*_{N-1}}^{\mathrm{T}}(x(t_l)), 0_{m\times N_{t_l}}\,]^{\mathrm{T}}$. The reason for adding zeros in the tails of the "shifted" input sequence is that there's no available control sequence beyond ${u^*_{N-1}}^{\mathrm{T}}(x(t_l))$. It is quite realistic to understand when no control sequence is available, we do nothing. Actually, this zero input signal, also gives a sign that we need to trigger the system to update the MPC control sequence in order to guarantee stability and performance.

Hence, the upper bound

$$\bar{V}^{(N)}(x(t_l + N_{t_l})) \triangleq J^{(N)}(x(t_l + N_{t_l}), \bar{U}_N(x(t))). \tag{25}$$

From Remark 6, we can conclude that if the RDP checking condition (24) is satisfied, $x(t_l + N_{t_l}) \in S_\nu^N$. Thus, state constraints are always satisfied by our RDP-based self-triggered MPC scheme and S_ν^N is a RH N-invariant set of our self-triggering MPC scheme.

Note that the solution of problem (23) is very cheap. We can simply increment N_{t_l} and check whether the RDP condition (24) is satisfied or not by forward simulation.

3.2 A Posteriori Analysis

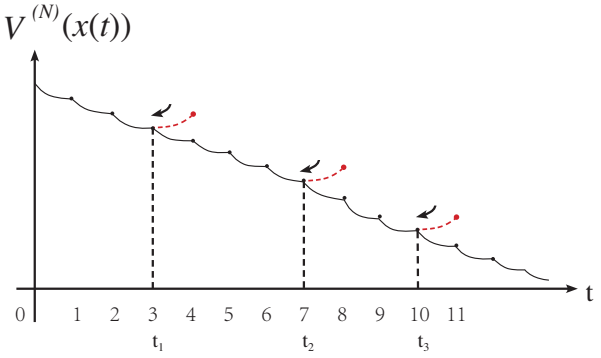

Figure 3. Illustration of RDP-based Triggering Scheme.

The RDP-based self-triggered scheme proposed in Section 3.1 guarantees the satisfaction of the RDP inequality (16). Thus if the inequality (24) is satisfied by implementing the "shifted" input sequence $\bar{U}_N(x(t_l + N_{t_l}))$, the system is invariant during the time interval $[t_l, t_l + \mathcal{N}_{t_l}(x(t_l))]$, and if the inequality (24) is not satisfied at $t_l + \mathcal{N}_{t_l}(x(t_l)) + 1$, we set $t_{l+1} = t_l + \mathcal{N}_{t_l}(x(t_l))$ and solve the MPC optimization problem for $t = t_{l+1}$. The idea of the scheme is illustrated in Fig. 3.

In the worst case, when $\mathcal{N}_{t_l}(x(t_l)) = 1$, we set $t_{l+1} = t_l + 1$ and from Assumption 2, we know that if we update MPC at $t_{l+1} = t_l + 1$, the optimal value for t_{l+1} will satisfy (8), thus (14), and from Remark 6, satisfy (16). In summary, Assumption 2 and our RDP self-triggering scheme in Section 3.1 guarantee that (16) is always satisfied.

Remark 7 *(On adjustment of the horizon length N in real time) If N was chosen too small to satisfy Assumption 2, the triggering mechanism will realize this, because optimization problem (23) will not have a feasible solution. So we can increase N in order to make problem (23) feasible for the given α.*

An illustration is shown in Section 5 (Fig. 8). But in this paper we assume that N is chosen large enough and remains fixed. A full treatment of the case when N is allowed to vary deserves separate attention in another paper.

4 Extension to Linear Systems with Bounded Disturbances

In this section, we consider an extension of the RDP-based design to the case of linear uncertain systems defined by

$$x_R(t+1) = Ax_R(t) + Bu_R(t) + w(t), \quad (26)$$
$$y_R(t) = Cx_R(t) + Du_R(t), \quad (27)$$

where $x_R(t) \in \mathbb{R}^n$ is the state of the real system (which can be measured) and $w \in \mathbb{W}$ is an unmeasurable but bounded disturbance. We call (1) the 'nominal dynamics' of system (26).

Our approach is based on the tube-based MPC approach, as proposed in [44], since optimizing over all possible feedback policies is not tractable in general. In the standard tube-based MPC, instead of solving problem (4), we solve the following optimization problem with tightened constraints

$$\begin{aligned} \min_{\mathbf{u}\triangleq[u_0^{\mathrm{T}},\cdots,u_{N-1}^{\mathrm{T}}]^{\mathrm{T}}} \; & J^{(N)}(x(t),\mathbf{u}) \triangleq \sum_{k=0}^{N-1} \|y_k\|_2^2, \quad (28)\\ \text{s.t. } & x_k \in \mathbb{X} \ominus \mathcal{E}, \quad k = 1,\dots,N,\\ & u_k \in \mathbb{U} \ominus K\mathcal{E}, \quad k = 0,1,\dots,N-1,\\ & x_0 = x(t),\\ & x_{k+1} = Ax_k + Bu_k, \quad k = 0,1,\dots,N-1,\\ & y_k = Cx_k + Du_k, \quad k = 0,1,\dots,N-1. \end{aligned}$$

where $\mathcal{E} \subseteq \mathbb{X}$ is a bounded set that will be specified.

A 'tube-based' feedback policy is applied to the real system, of the form

$$u_R(t) = K(x_R(t) - x(t)) + \mu(x(t)) \quad (29)$$

where $x(t)$ is the nominal state and $K \in \mathbb{R}^{m\times n}$ is such that $A_K := A + BK$ is strongly stable, i.e. all the eigenvalues of A_K are contained in the interior of the unit disc. If we define $\delta := x_R - x$, the deviation between the actual state x_R and the nominal state x, then the dynamic of this deviation state is

$$\delta(t+1) = A_K\delta(t) + w(t). \quad (30)$$

Definition 4 *A set $\mathcal{E} \subseteq \mathbb{R}^n$ is called an* robustly positively invariant (RPI) set *for the discrete-time system $x(t+1) = \Phi(x(t), w(t))$ with disturbance set $\mathbb{W}$, if $\mathcal{E} \subseteq \mathbb{X}$ and for all $x \in \mathcal{E}$ and all $w \in \mathbb{W}$, it holds that $\Phi(x, w) \in \mathcal{E}$ holds.*

Since A_K is stable and w is bounded, there exists a RPI set $\mathcal{E}$ for the system (30) satisfying [38]

$$A_K\mathcal{E} \oplus \mathbb{W} \subseteq \mathcal{E}. \quad (31)$$

However, the policy (29) needs real-time measurements of x_R, which we only have at triggering times t_l, $l \in \mathbb{Z}_+$. We therefore modify the scheme outlined above, and propose our robust tube-based self-triggered MPC in the next subsection.

4.1 RDP-based Robust Triggering Scheme

In the robust self-triggered MPC setting, instead of repeatedly computing the MPC update (28) at each sampling time, in this subsection we aim to adopt the same RDP condition for the nominal dynamics as in Section 3.1 to determine the time to update the MPC solution. As discussed in Remark 4, the sparse implementation will introduce more model mismatch (cf. Fig. 9 to Fig. 11). In this section, we utilize the non-sparse implementation strategy for the nominal dynamics.

At a triggering time $t_l \in \mathbb{Z}_+$ with $l \in \mathbb{Z}_+$, we solve the MPC optimization problem (28) and get a sequence of nominal control inputs $U_N^*(x(t_l)) = [{u_0^*}^{\mathrm{T}}(x(t_l)), {u_1^*}^{\mathrm{T}}(x(t_l)), \dots, {u_{N-1}^*}^{\mathrm{T}}(x(t_l))]^{\mathrm{T}}$ for the nominal dynamics (1)-(2) and apply control according to ((10)) before the next sampling instant. The shift sequence of this solution will be used to compute the next triggering time t_{l+1} by

$$t_{l+1} = t_l + \mathcal{N}_{t_l}(x(t_l)), \quad (32)$$

where

$$\begin{aligned} \mathcal{N}_{t_l}(x(t_l)) \triangleq & \max\{N_{t_l} \in \mathbb{Z}_{[1,N-1]}\} \quad (33)\\ \text{s.t. } & (i)\ V^{(N)}(x(t_l)) - \bar{V}^{(N)}(x(t_l+N_{t_l}))\\ & \geq e(t_l) + \alpha\left(\sum_{t=t_l}^{t_l+N_{t_l}-1} \|y_{\tilde{\mu}}(t)\|_2^2\right), \quad (34)\\ & (ii)\ x_R(t_l) - x(t_l+N_{t_l}) \in \Delta\mathcal{E}, \quad (35) \end{aligned}$$

where $\Delta\mathcal{E}$ is a bounded set containing 0. An algorithm for finding suitable $\mathcal{E}$ and $\Delta\mathcal{E}$ is presented as Algorithm 1 (cf. Section 4.3). Compare the condition (33) with (23) the only difference is that we add a new condition (ii) in (35). The reason for adding this condition will be explained in the next subsection.

4.2 Robust Control Policy

For $t_l < t < t_{l+1}$, we do not have measurements of x_R. So we redefine the control policy as follows

$$\begin{aligned} u_R(t) &= K(x_R(t_l) - x(t)) + \tilde{\mu}(t, x(t_l))\\ &:= K(x_R(t_l) - x(t)) + u_{t-t_l}^*(x(t_l)), \quad t \in \mathbb{Z}_{[t_l, t_{l+\mathcal{N}_{t_l}})} \end{aligned} \quad (36)$$

The differences between control policies (29) and (36) are: First, in the correction term, we replace $x_R(t)$ with $x_R(t_l)$. Second, we replace $\mu(x(t))$ by $\tilde{\mu}(t, x(t_l))$.

At time instant t_l, we define the updates of x_R, x and δ before t_{l+1} as shown at the top of the next page. At a time instant $t_l + M$, $M \in [1, \mathcal{N}_{t_l}(x(t_l))]$, we have

$$\delta(t_l + M) = A_K \delta(t_l + M - 1) + d(t_l + M - 1), \quad (37)$$

where $d(t_l + M - 1) = BK(x_R(t_l) - x_R(t_l + M - 1)) + w(t_l + M - 1)$. As A_K is strongly stable and $w(t_l + M - 1) \in \mathbb{W}$ is bounded, if we can guarantee $BK(x_R(t_l) - x_R(t_l + M - 1)) \oplus \mathbb{W} \in \mathcal{F}$, where $0 \in \mathcal{F}$ is a bounded set, i.e. $d \in \mathcal{F}$, then we can compute an invariant set $\mathcal{E}$ for the system (37) under policy (36) satisfying

$$A_K \mathcal{E} \oplus \mathcal{F} \subseteq \mathcal{E}. \quad (38)$$

Given $\mathcal{E}$, a candidate RPI set for δ, suppose that we can find a set $\Delta\mathcal{E}$ such that

$$x_R(t_l) - x(t_l + M - 1) \in \Delta\mathcal{E} \quad (39)$$

and that

$$A_K \mathcal{E} \oplus BK(\Delta\mathcal{E} \oplus \mathcal{E}) \oplus \mathbb{W} \subseteq \mathcal{E}, \quad (40)$$

where $BK(\Delta\mathcal{E} \oplus \mathcal{E}) \oplus \mathbb{W} \subseteq \mathcal{F}$.

Then we have that

$$x_R(t_l) - x_R(t_l + M - 1) = [x_R(t_l) - x(t_l + M - 1)] + [x(t_l + M - 1) - x_R(t_l + M - 1)] \in \Delta\mathcal{E} \oplus \mathcal{E} \quad (41)$$

and hence, from (37), if $\delta(t_l + M - 1) \in \mathcal{E}$, then

$$\delta(t_l + M) \in A_K \mathcal{E} \oplus BK(\Delta\mathcal{E} \oplus \mathcal{E}) \oplus \mathbb{W} \subseteq \mathcal{E} \quad (42)$$

so that $\mathcal{E}$ is indeed an RPI set for (37). Note that $0 \in \mathcal{E}$.

So we need to check condition (24) and (39) together for $N_{t_l} \in \mathbb{Z}_{[1,N-1]}$. If these hold then conditions (i) and (ii) in (33) hold.

Theorem 2 *Consider the closed-loop system (26)-(27) and (36) under RDP-based triggering scheme as in (32) with the MPC control update at the triggering time according to (28). Let $\nu = V^N(x(0))$. Then the trajectory $x_R(t)$ starting from any initial condition $x_R(0) \in S_\nu^N$, converges asymptotically to the set $\mathcal{E}$.*

Proof. Note that the system dynamics in the optimization problems (4) and (28) are the same. From (40), we have

$$x_R(t_l + k) \in x(t_l + k) \oplus \mathcal{E} \quad (43)$$

for $k = 1, \ldots, \mathcal{N}_{t_l}(x(t_l))$ and $l \in \mathbb{Z}_+$, then we can conclude $x_R(t_l + k) \in S_\nu^N \subset \mathbb{X}$ if $x(t_l) \in S_\nu^N \ominus \mathcal{E}$ by (28). From (32), $x_R(t_{l+1}) = x_R(t_l + \mathcal{N}_{t_l}(x(t_l)))$, thus $x_R(t_{l+1}) \in x(t_{l+1}) \oplus \mathcal{E} \in S_\nu^N \subset \mathbb{X}$. This proves the invariance at the sampling instants $t \in [t_l, t_{l+1}]$ in between the triggering times. By induction, if $x(0) = x(t_0) \in S_\nu^N \ominus \mathcal{E}$,

$$x_R(t) \in x(t) \oplus \mathcal{E} \in S_\nu^N, \quad (44)$$

for all $t \in \bigcup_{l \in \mathbb{Z}_+} [t_l, t_{l+1}]$. As the RDP condition (34) (which is the same as (16)) is satisfied, from Theorem 1, we have $\lim_{t \to \infty} x(t) = 0$ for all $x(t) \in S_\nu^N \ominus \mathcal{E}$, i.e. $S_\nu^N \ominus \mathcal{E}$ is forward invariant for $x(t)$. Furthermore, from (44), $x_R(t)$ converges asymptotically to the set $\mathcal{E}$. □

4.3 Computation of sets

In order to compute the RPI set, we first introduce the following definitions from [11] and we rewrite them a bit according to [33].

Definition 5 *[11, Definition 11.12] For the discrete-time system $x(t+1) = \Phi(x(t), w(t))$, we denote the* one-step controllable set *to the set $\mathcal{E}$ with disturbance set $\mathbb{W}$ as*

$$\mathrm{Pre}(\mathcal{E}) = \{x \mid \Phi(x(t), w(t)) \in \mathcal{E}, \forall w \in \mathbb{W}\}. \quad (45)$$

Definition 6 *[11, Definition 11.14] For the discrete-time system $x(t+1) = \Phi(x(t), w(t))$, we denote the* one-step reachable set *from the set $\mathcal{E}$ with disturbance set $\mathbb{W}$ as*

$$\mathrm{Reach}(\mathcal{E}) = \{\Phi(x(t), w(t)) \mid \forall x \in \mathcal{E}, \forall w \in \mathbb{W}\}. \quad (46)$$

With the above definitions, for system (37), one-step controllable set to $\mathcal{E}$ can be computed by

$$\mathrm{Pre}(\mathcal{E}) = ((\mathcal{E} \ominus \mathbb{W}) \oplus (-BK(\Delta\mathcal{E} \oplus \mathcal{E}))) \circ A_K. \quad (47)$$

Here, $\mathcal{P} \circ A$ with a set $\mathcal{P}$ and a matrix A denotes the inverse mapping of $\mathcal{P}$ under linear map A, i.e. $\mathcal{P} \circ A = \{x \mid Ax \in \mathcal{P}\}$ [33].

The computation of sets $\mathcal{E}$ and $\Delta\mathcal{E}$ is presented in Algorithm 1.

Remark 8 *We notice that algorithms in [8, Algorithm 26.2] and [9, Algorithm 1] are wrong in that they compute the one-step reachable set (ReachSet) which, for system (37), is*

$$\mathrm{Reach}(\mathcal{E}) := \mathcal{E}^+ = A_K \mathcal{E} \oplus BK(\Delta\mathcal{E} \oplus \mathcal{E}) \oplus \mathbb{W},$$

instead of one-step controllable set (PreSet) in Step 3, the convergence of our Algorithm 1 is achieved by iterations $\mathcal{E}_{\text{next}} = \mathrm{Pre}(\mathcal{E}) \cap \mathcal{E}$.

$$
\begin{aligned}
x_R(t_l+1) &:= Ax_R(t_l) + BK(x_R(t_l) - x(t_l)) + Bu_0^*(x(t_l)) + w(t_l),\\
x(t_l+1) &:= Ax(t_l) + Bu_0^*(x(t_l)),\\
\delta(t_l+1) &= A_K\delta(t_l) + w(t_l),\\
x_R(t_l+2) &:= Ax_R(t_l+1) + BK(x_R(t_l) - x(t_l+1)) + Bu_1^*(x(t_l)) + w(t_l+1),\\
x(t_l+2) &:= Ax(t_l+1) + Bu_1^*(x(t_l)),\\
\delta(t_l+2) &= A_K\delta(t_l+1) + BK(x_R(t_l) - x_R(t_l+1)) + w(t_l+1),\\
&\vdots\\
x_R(t_l+M) &:= Ax_R(t_l+M-1) + BK(x_R(t_l) - x(t_l+M-1)) + Bu_{M-1}^*(x(t_l)) + w(t_l+M-1),\\
x(t_l+M) &:= Ax(t_l+M-1) + Bu_{M-1}^*(x(t_l)),\\
\delta(t_l+M) &= A_K\delta(t_l+M-1) + BK(x_R(t_l) - x_R(t_l+M-1)) + w(t_l+M-1).
\end{aligned}
$$

Algorithm 1 Computation of sets

1: Choose an initial (rough) hyper-cube $\Delta\mathcal{E}$ and a constant $\gamma \in (0,1)$.
2: Initialize, $\mathcal{E} = BK\Delta\mathcal{E} \oplus \mathbb{W}$.
3: Compute $\mathrm{Pre}(\mathcal{E})$ as in (47).
4: Set $\mathcal{E}_{\text{next}} = \mathrm{Pre}(\mathcal{E}) \cap \mathcal{E}$.
5: IF $\mathcal{E}_{\text{next}} == \mathcal{E}$, then GOTO Step 6). Otherwise, set $\mathcal{E} := \mathcal{E}_{\text{next}}$ and GOTO Step 4).
6: IF $\mathcal{E} \subset \mathbb{X}$, then STOP. Otherwise set $\Delta\mathcal{E} = \gamma\Delta\mathcal{E}$, and GOTO Step 2).

Remark 9 *(On initialization of $\Delta\mathcal{E}$) $\Delta\mathcal{E}$ in Step 1 of Algorithm 1 should be chosen such that there exists a set $\mathcal{E}$ which satisfies the invariance condition (40). From (39), $\Delta\mathcal{E}$ can be viewed as a bounded "tube" between $x_R(t_l)$ and the predicted nominal state trajectory $x(t_l+M-1)$. If $\Delta\mathcal{E}$ is chosen too large, the prediction error is large, thus it is difficult to satisfy (40). So $\Delta\mathcal{E}$ should be chosen to be sufficiently small to make the algorithm converge and get an invariant set $\mathcal{E}$. One reasonable option for $\Delta\mathcal{E}$ is to choose it as a hyper-cube whose magnitude is the same as the size of the set $\mathbb{W}$, since w determines the difference between the real state trajectory x_R and the nominal state trajectory x.*

5 Illustrative Example

Consider system (1) with three subsystems and each subsystem has five states and one input [24].

$$
x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{15} \end{bmatrix}, \quad u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \quad A = \begin{bmatrix} A_1 & 0 & 0 \\ 0 & A_2 & 0 \\ 0 & 0 & A_3 \end{bmatrix},
$$

$$
B = \begin{bmatrix} B_{11} & 0 & B_{13} \\ B_{21} & B_{22} & B_{23} \\ 0 & B_{32} & B_{33} \end{bmatrix},
$$

where

$$
A_1 = \begin{bmatrix} 0.2647 & 0.2367 & 0.0015 & 0.1930 & 0.0565 \\ 0.0540 & 0.4256 & 0.1543 & 0.2090 & 0.0790 \\ 0.1006 & 0.2433 & 0.4554 & 0.1450 & 0.1478 \\ 0.0844 & 0.1107 & 0.0589 & 0.4532 & 0.2098 \\ 0.0581 & 0.2377 & 0.2358 & 0.0832 & 0.3263 \end{bmatrix},
$$

$$
A_2 = \begin{bmatrix} 0.4552 & 0.1087 & 0.0984 & 0.0699 & 0.1265 \\ 0.0266 & 0.3975 & 0.1085 & 0.1573 & 0.1319 \\ 0.2077 & 0.1772 & 0.4946 & 0.1488 & 0.0228 \\ 0.2127 & 0.1877 & 0.1449 & 0.4964 & 0.2287 \\ 0.0896 & 0.1916 & 0.2148 & 0.0217 & 0.4764 \end{bmatrix},
$$

$$
A_3 = \begin{bmatrix} 0.4547 & 0.0949 & 0.1016 & 0.1821 & 0.1000 \\ 0.2497 & 0.4990 & 0.1598 & 0.1224 & 0.2509 \\ 0.0402 & 0.2458 & 0.5019 & 0.1616 & 0.1017 \\ 0.0599 & 0.1627 & 0.1415 & 0.4773 & 0.1666 \\ 0.1775 & 0.2175 & 0.2360 & 0.0502 & 0.4807 \end{bmatrix},
$$

and

$$B_{11} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.4026 \end{bmatrix}, \quad B_{13} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.8272 \end{bmatrix},$$

$$B_{21} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.4758 \end{bmatrix}, \; B_{22} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.4390 \end{bmatrix}, \; B_{23} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.6198 \end{bmatrix},$$

$$B_{32} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.3039 \end{bmatrix}, \quad B_{33} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.9956 \end{bmatrix}.$$

As the problem is a regulation problem, the regulated output is chosen to be $y_k = \begin{pmatrix} I_{15} \\ 0 \end{pmatrix} x_k + \begin{pmatrix} 0 \\ I_3 \end{pmatrix} u_k$. Both the state and input constraints are set to require that their trajectories lie within the range $[-2, 2]$. The control horizon is chosen as $N = 7$ and the performance degradation parameter $\alpha = 0.9$. If the system's state reaches a small neighborhood of the origin, the simulation is terminated. First, we implement the non-sparse implementation strategy. The simulation results are presented in Fig. 4. The circle trajectories represent the simulation results for the classical time-driven MPC scheme, whereas the star trajectories represent the simulation results for the self-triggered MPC scheme proposed in this paper. All the computations and simulations are performed in MATLAB version R2016b on an Intel 2.8GHz Centrino laptop. The total computationtime for the classical MPC scheme is 35.8625 seconds, while the computation time for the self-triggered MPC scheme that presented in this paper is only 7.3075 seconds.

From Fig. 4, we can see that, if we do not add the self-triggered mechanism to the MPC algorithm, it takes 43 MPC updates for the system's state trajectory to evolve to the origin as shown in the 3rd subplot of Fig. 4. If we embed the proposed triggering scheme in the MPC algorithm, it only needs 9 MPC updates to achieve the convergence to the origin as shown in the 4th subplot of Fig. 4. Hence, the proposed strategy can significantly reduce the MPC update times to achieve regulation. The triggering instants are recorded in Fig. 5.

Effects of $e(t_l)$:

In order to show the effects of the slack $e(t_l)$ in reducing the conservatism in (16). We remove the $e(t_l)$ term in (24) of our triggering scheme. The simulation results are shown in Fig. 6 for $\alpha = 0.8, 0.7$, and 0.6. For $\alpha = 0.8$, if there's no $e(t_l)$ term presents in (24), we need 43 updates. For $\alpha = 0.7$, we need 38 updates. For $\alpha = 0.6$, we need 11 updates. However, if the $e(t_l)$ term presents in (24), from Fig. 5, we only need 9 updates for $\alpha = 0.9$. These simulations verify that the slack $e(t_l)$ has significant role in reducing the conservatism as remarked in Remark 6.

Online Adjustment of Horizon N (Remark 7):

First, in Fig. 7, we show that when $N = 2$ is too small, the trajectories do not converge to the origin as required by the control objective. Then we introduce the online adjustment of horizon N mechanism as in Remark 7. Fig. 8 shows the horizon increases from $N = 3$ to 8 and the stability and overall performance are guaranteed.

Sparse Implementation:

For the same settings as described above, we next implement the sparse implementation scheme and the simulation result is presented in Fig. 9. As discussed in the paper, keep the control signal constant between triggering time can introduce more model mismatch. If we restrict the performance requirement to $\alpha = 0.9$, it is almost like the result of time-driven MPC. So next we lower the performance bound α to 0.8. Now we can achieve the sparse character as shown in Fig. 10, and the computation time is 15.5897 seconds. It needs 15 updates as illustrated in Fig. 11 while for non-sparse implementation in Fig. 5, there are only 9 triggering instants. Hence, the non-sparse strategy can significantly reduce the MPC update times to achieve regulation.

Tube-based Robust Implemetation:

The final simulation is for the robust case presented in Section 4. We only implement the non-sparse implementation strategy. The disturbance is added to system (26) as $w(t) = \phi w(t-1) + \xi(t)$, where $\xi(t)$ is generated from normal distribution with standard deviation $\sigma = 0.1 \times I_{15\times 1}$ and $\phi = 0.7 \times I_{15\times 1}$ is the autocorrelation factor. $\mathbb{W} = \{w | \|w\|_\infty \leq 0.05\}$. The control horizon is chosen as $N = 7$ and the performance degradation parameter $\alpha = 0.9$. We initialize $\Delta\mathcal{E}$ as a hyper-cube whose lower bound and upper bound are the same as the set $\mathbb{W}$ and choose $\gamma = 0.7$ for Algorithm 1. The simulation result is shown in Fig. 12 and Fig. 13. The computation time is 9.3282 seconds and it takes 10 updates in 50 sampling instants, which shows the effectiveness of our proposed policy (36) for robust self-triggered MPC.

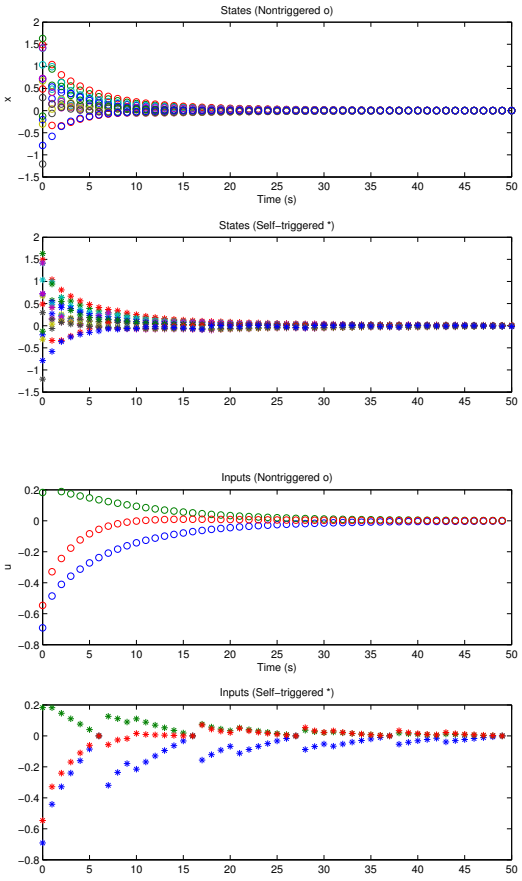

Figure 4. State and input trajectories with $\alpha = 0.9$. Circles for the classical MPC and stars for the self-triggered MPC.

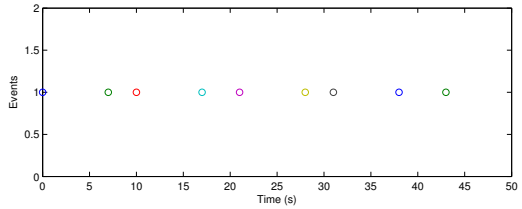

Figure 5. Event triggering instants. The triggering instants are marked with the circles with the value 1.

6 Conclusions

This paper proposed a self-triggered MPC synthesis procedure for (disturbed) linear systems subject to state and input constraints based on the relaxed dynamic programming inequality. The interval between control input updates is maximized such that the overall closed-loop system maintains asymptotic stability, satisfies given constraints, and meets a certain prescribed performance level. The illustrative examples showed that the number of control updates in the self-triggered MPC can be significantly reduced compared to the classic time-driven MPC while the system's state trajectory still reaches

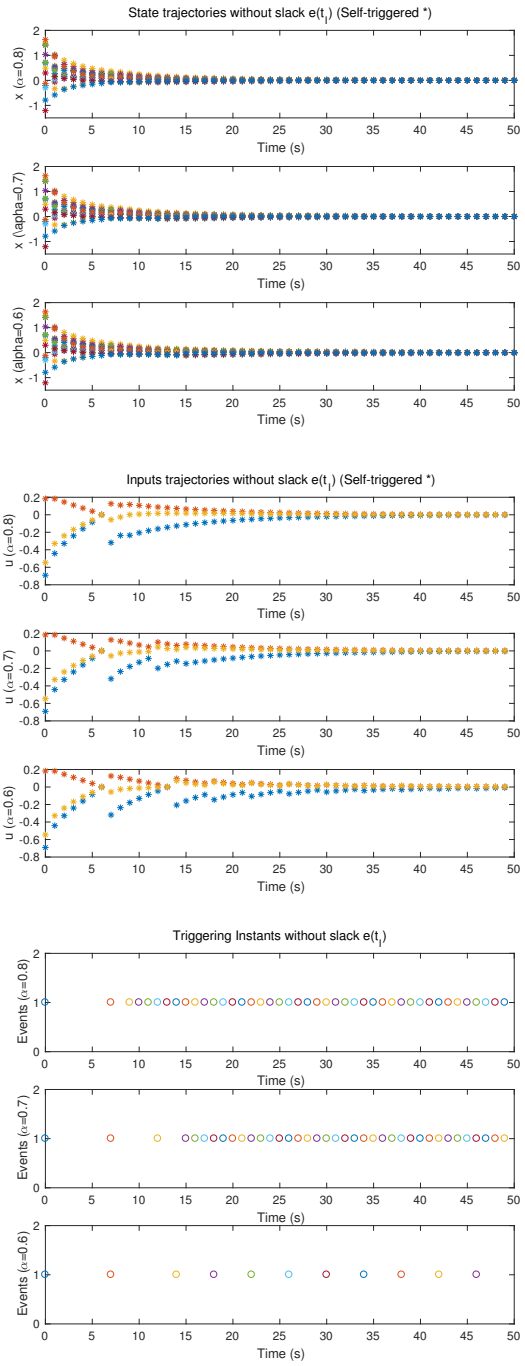

Figure 6. Simulation results without the slack $e(t_l)$ in (24) of our triggering scheme.

a small neighborhood of the equilibrium safely. This can help significantly in the setting of distributed MPC where the communication of control updates among subsystems can put heavy loads in the network traffic and act as a bottleneck for implementation. In this sense, the reduced frequency of updates and the knowledge of next update time for each unit can be valuable in the

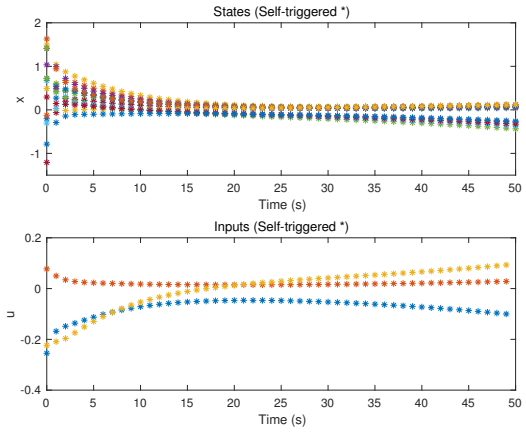

Figure 7. Simulation results with horizon N=2.

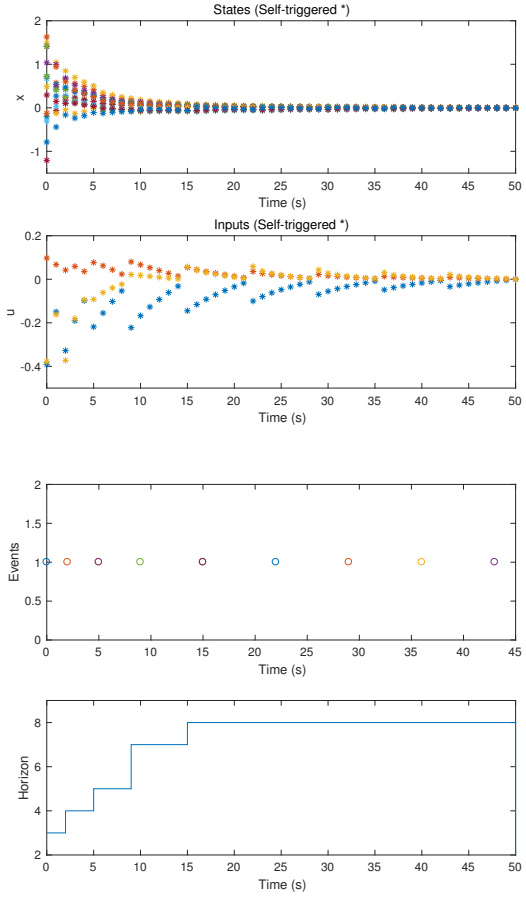

Figure 8. Simulation results with more flexible self-triggered MPC with adjusting horizon mechanism.

scheduling of network communication. Extensions of the idea to tracking problems and to distributed MPC are currently being explored.

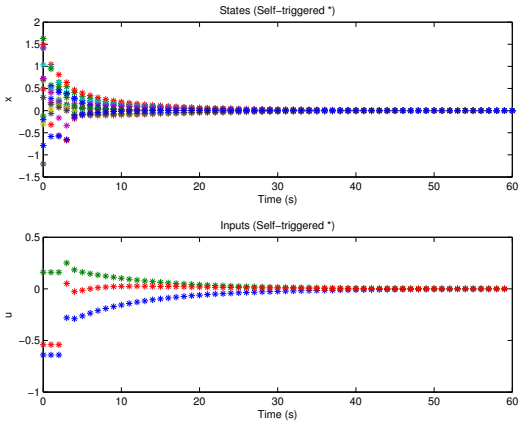

Figure 9. State and input trajectories of sparse implementation with $\alpha = 0.9$.

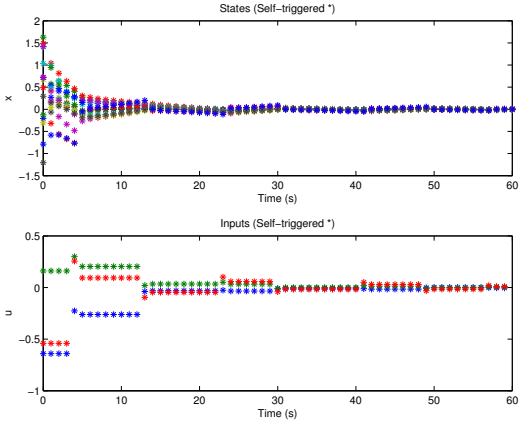

Figure 10. State and input trajectories of sparse implementation with $\alpha = 0.8$.

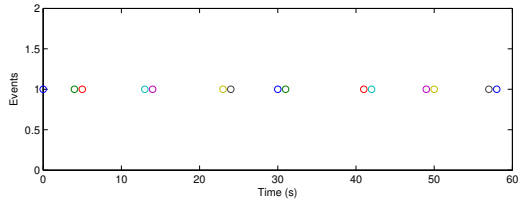

Figure 11. Event triggering instants of sparse implementation with $\alpha = 0.8$. The triggering instants are marked with the circles with the value 1.

Acknowledgements

The authors would like to thank the kind comments and valuable suggestions for this paper from Professor Lars Grüne at the University of Bayreuth, and the anonymous reviewers for their careful reading of the manuscript and comments.

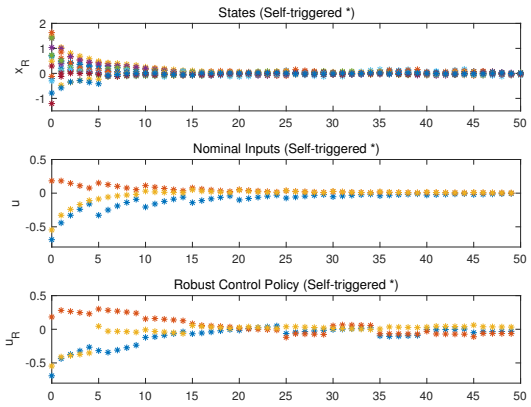

Figure 12. State and input trajectories of robust self-triggered MPC case.

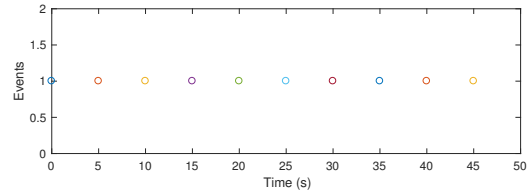

Figure 13. Event triggering instants of robust self-triggered MPC case.

References

[1] N. Altmüller and L. Grüne, "Distributed and boundary model predictive control for the heat equation," *GAMM-Mitt. 35*, no. 2, pp. 131–145, 2012.

[2] N. Altmüller, L. Grüne and K. Worthmann, "A comparative stability analysis of Neumann and Dirichlet boundary MPC for the heat equation," in *Proceedings of the 1st IFAC Workshop on Control of Systems Modeled by Partial Differential Equations*, Paris, France, 2013.

[3] N. Altmüller and L. Grüne, "Receding horizon optimal control for the wave equation," in *Proceedings of the 49th IEEE Conference on Decision and Control*, Atlanta, Georgia, 2010.

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

[5] E. Aydiner, F. D. Brunner, W. P. M. H. Heemels and F. Allgöwer, "Robust Self-Triggered Model Predictive Control for Constrained Discrete-Time LTI Systems based on Homothetic Tubes," in *Proceedings of the European control conference*, Linz, Austria, 2015.

[6] B. Azmi and K. Kunisch, "On the stabilizability of the Burgers equation by receding horizon control," *SIAM J. Control Optim.*, vol. 54, no. 3, pp. 1378–1405, 2016.

[7] J. D. J. Barrads Berglind, T. M. P. Gommans, and W. P. M. H. Heemels, "Self-trigggered MPC for constrained linear systems and quadratic costs," in *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference*, Noordwijkerhout, The Netherlands, 2012.

[8] G. Betti, M. Farina and R. Scattolini, "Distributed MPC: A noncooperative approach based on robustness concepts," in *Distributed MPC Made Easy*, Springer, 2013.

[9] G. Betti, M. Farina and R. Scattolini, "Realization issues, tuning, and testing of a distributed predictive control algorithm," *Journal of Process Control*, vol. 24, pp. 424–434, 2014.

[10] A. Boccia, L. Grüne and K. Worthmann, "Stability and feasibility of state constrained MPC without stabilizing terminal constraints," *Systems & Control Letters*, vol. 72, pp. 14–21, 2014.

[11] F. Borrelli, A. Bemporad and M. Morari , *Predictive control for linear and hybrid systems*. Cambridge University Press, New York, 2017.

[12] F. D. Brunner, M. Heemels and F. Allgöwer, "Robust self-triggered MPC for constrained linear systems: A tube-based approach," *Automatica*, vol. 72, pp. 73–83, 2016.

[13] E. Camponogara, D. Jia, B.H. Krogh and S. Talukdar, "Distributed model predictive control," *IEEE Control Systems Magazine*, vol. 22, issue. 1, pp. 44–52, 2002.

[14] L. Chisci, J. A. Rossiter and G. Zappa, "Systems with persistent disturbances: predictive control with restricted constraints," *Automatica*, vol. 37, pp. 1019–1028, 2001.

[15] L. Dai, Y. Gao, L. Xie, K. H. Johansson and Y. Xia, "Stochastic self-triggered model predictive control for linear systems with probabilistic constraints," *Automatica*, vol. 92, pp. 9–17, 2018.

[16] M. D. Doan, P. Giselsson, T. Keviczky, B. De Schutter, and A. Rantzer, "A distributed accelerated gradient algorithm for distributed model predictive control of a hydro power valley," *Control Engineering Practice*, vol. 21, issue. 11, pp. 1594–1605, 2013.

[17] W. B. Dunbar and R. M. Murray, "Distributed receding horizon control of multi-vehicle formation stabilization," *Automatica*, vol. 42, no. 4, pp. 549–558, 2006.

[18] A. Eqtami, D. V. Dimarogonas and K. J. Kyriakopoulos, "Event-triggered control for discrete-time systems," in *Proceedings of the 2010 American Control Conference*, Baltimore, USA, 2010.

[19] A. Eqtami, D. V. Dimarogonas and K. J. Kyriakopoulos, "Event-triggered strategies for decentralized model predictive controllers," in *Proceedings of IFAC World Congress*, Milano, Italy, 2011.

[20] A. Eqtami, D. V. Dimarogonas and K. J. Kyriakopoulos, "Event-based model predictive control for the cooperation of distributed agents," in *Proceedings of the 2012 American Control Conference*, Montréal, Canada, 2012.

[21] F. Farokhi, I. Shames and K. H. Johansson, "Distributed MPC via dual decomposition and alternating direction method of multipliers," in J. M. Maestre and R. R. Negenborn, Eds., *Distributed MPC Made Easy*, Springer-Verlag, 2012.

[22] P. Giselsson, "Adaptive nonlinear model predictive control with suboptimality and stability guarantees," in *Proceedings of the 49th IEEE Conference on Decision and Control (CDC)* , Atlanta, USA, 2010.

[23] P. Giselsson and A. Rantzer, "Distributed model predictive control with suboptimality and stability guarantees," in *Proceedings of the 49th IEEE Conference on Decision and Control (CDC)* , Atlanta, USA, 2010.

[24] P. Giselsson, *Gradient-Based Distributed Model Predictive Control*. PhD thesis, Department of Automatic Control, Lund University, Sweden, November 2012 (Supplement A).

[25] T.M.P. Gommans and W.P.M.H. Heemels, "Resource-aware MPC for Constrained Nonlinear Systems: A Self-Triggered Control Approach," *Systems & Control Letters*, vol. 79, pp. 59–67, 2015.

[26] L. Grüne, "Analysis and design of unconstrained nonlinear MPC schemes for finite and infinite dimensional systems," *SIAM Journal on Control and Optimizaiton*, vol. 48, pp. 1206–1228, 2009.

[27] L. Grüne and A. Rantzer, "On the infinite horizon performance of receding horizon controllers," *IEEE Transactions on Automatic Control*, vol. 53, no. 9, pp. 2100–2111, 2008.

[28] L. Grüne , "NMPC without terminal constraints," in *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control*, Noordwijkerhout, The Netherlands, 2012.

[29] W. P. M. H. Heemels, K. H. Johansson and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *Proceedings of the IEEE Conference on Decision and Control (CDC)* , Maui, USA, 2012.

[30] E. Henriksson, D. E. Quevedo, H. Sandberg and K. H. Johansson, "Self-trigggered model predictive control for network scheduling and control," in *Proceedings of the 8th IFAC Symposium on Advanced Control of Chemical Processes*, Furama Riverfront, Singapore, 2012.

[31] E. C. Kerrigan, *Robust constraint satisfaction: invariant sets and predictive control*. PhD thesis, University of Cambridge, U. K., 2000.

[32] A. Kozma, *Distributed optimization methods for large scale optimal control*. PhD thesis, Faculty of Engineering Science, KU Leuven, Belgium, 2014.

[33] M. Kvasnica, B. Takács, J. Holaza and D. Ingole, "Reachability analysis and control synthesis for uncertain linear systems in MPT," *IFAC-PapersOnLine*, vol. 48, no. 14, pp. 302–307, 2015.

[34] B. Lincoln and A. Rantzer, "Relaxing dynamic programming," *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1249–1260, 2006.

[35] C. Liu, H. Li, J. Gao and D. Xu, "Robust self-triggered minmax model predictive control for discrete-time nonlinear systems," *Automatica*, vol. 89, pp. 333–339, 2018.

[36] J. M. Maciejowski, *Predictive control: with constraints*. Prentice-Hall, Harlow, UK, 2002.

[37] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[38] D. Q. Mayne, M. M. Seron and S. V. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, pp. 219–224, 2005.

[39] R. R. Negenborn, *Multi-agent model predictive control with applications to power networks*. PhD thesis, Delft University of Technology, Delft, The Netherlands, 2007.

[40] J. Pannek and K. Worthmann, "Stability and performance guarantees for model predictive control algorithms without terminal constraints," *ZAMM 94*, no. 6, pp. 317–330, 2014.

[41] J. Pekar, P. Garimella, D. Germann and G.E. Stewart, "Experimental Results for Sensor Selection and Multivariable Controller Design for a Heavy-Duty Diesel Engine," in *Proceedings of the E-COSM2012 Conference*, Rueil-Malmaison, France, 2012.

[42] J. A. Primbs and V. Nevistić, "Feasibility and stability of constrained finite receding horizon control," *Automatica*, vol. 36, pp. 965–971, 2000.

[43] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, pp. 733–764, 2003.

[44] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison, USA, pp. 220–242, 2009 (Chapter 3 Robust Model Predictive Control).

[45] J. H. Sandee, *Event-driven control in theory and practice: trade-offs in software and control performance*. PhD Dissertatie, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2006.

[46] J. S. Shamma and D. Xiong, "Linear nonquadratic optimal control," *IEEE Transactions on Automatic Control*, vol. 42, no. 6, pp. 875–879, 1997.

[47] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680–1685, 2007.

[48] K. Worthmann, M. W. Mehrez, M. Zanon, M., G. K. Mann, R. G. Gosine and M. Diehl, "Model predictive control of nonholonomic mobile robots without stabilizing constraints and costs," *IEEE Trans. Control Syst. Techn.*, vol. 24, no. 4, pp. 1394–1406, 2016.