Preprints of the
9th International Symposium on Advanced Control of Chemical Processes
The International Federation of Automatic Control
June 7-10, 2015, Whistler, British Columbia, Canada

Asynchronous Separable Self-triggered Model Predictive Control based on Relaxed Dynamic Programming

Liang Lu [*,**]

[*] *Department of Chemical and Biomolecular Engineering, Korea Advanced Institute of Science & Technology (KAIST), Daejeon, Korea.*
[**] *State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, China.*
(e-mail: liangup@gmail.com)

Abstract: This paper presents an asynchronous distributed self-triggered MPC controller design strategy for large-scale constrained linear systems. Based on the so-called relaxed dynamic programming (RDP) inequality, at a triggered time that corresponding to a subspace of the whole system, the synthesis procedure allows us to locally determine both the updated MPC control action and the next triggering time for this subsystem. The subsystems update their inputs asynchronously, while the resulting self-triggered MPC control law can still preserve stability and constraint satisfaction. The derived conditions can be adapted to robust distributed event-triggered MPC implementation with continuous monitoring of the systems, when the system model is uncertain.

1. INTRODUCTION

Model predictive control (MPC) is an advanced and promising methodology to optimally control and manage the energy consumption for constrained systems Maciejowski [2002], Mayne et al. [2000], Morari and Lee [1999]. To design a MPC controller, we use a given model of the system that you want to control to plan the future control moves to make the process behave optimally with some desired properties by solving a scalable optimization problem periodically in real time at each sampling time, which results in a unique sequence of optimal control inputs. Since the control law is updated at each sampling time repetitively, we call this type of control update as time-driven update. Over the last fifty years, MPC technology have been widely and successfully applied in many industries, including energy efficient building control, automotive powertrain systems Pekar et al. [2012], chemical plants Qin and Badgwell [2003], power grids Negenborn [2007] and water distribution systems Doan et al. [2002], Kozma [2014].

However, there are still some challenges for time-driven MPC, and I would like to address some of the challenges that are related to this paper. The first is large-scale distributed MPC and optimization. MPC depends on a holistic model that incorporates dynamics and constraints of the system, but in many situations, we can not obtain a holistic model for an extreme large-scale system, like power grids of a country. One solution for this kind of large-scale distributed system is that we break the overall system into a bunch of small subsystems that can be easily handled by scalable computation. So there is a growing interest in distributed MPC and its related optimization and implementation for large-scale heterogenous network systems Bertsekas and Tsitsiklis [1997], Giselsson and Rantzer [2013]. The second one is about the efficiency and fast implementation of the MPC algorithm. At each sampling instant, MPC control laws are computed by solving an optimization problem, and for large-scale systems, the online computation processes can be quiet time consuming, the MPC design procedures are also involved with sensor communications among these subsystems over each sampling instant Camponogara et al. [2002], which consume a lot of energy in practice, and if we can not get a solution for the next sampling time, the system may not be stabilized.

To improve the efficiency of the traditional time-driven MPC and motivated by the works on event-triggered and self-triggered control Anta and Tabuada [2010], Heemels et al. [2012], the paradigms of event-triggered and the self-triggered MPC has drawn increasing attention in recent years for economically updating the real-time MPC control law. In event-triggered MPC, a prescribed triggering condition is constantly checked with continuous measurements and if it is violated, the actuator is triggered for activation of the MPC update process, while in self-triggered control, at a triggered time, both the updated MPC control action and the next triggering time must be determined. Since both event- and self-triggered updates depend on some triggering rules, we call this type of control update as event-driven update. The key issue in choosing between the event-triggered and the self-triggered MPC implementations, is whether the system states can be continuously monitored .

Most of the existing event- and self-triggered control results are for continuous-time systems, and utilize the

⋆ This work has been partially supported by the BK21 Plus Program of Korea, the National Natural Science Foundation of China (Grant No. 61304091) and the Research Foundation for the Doctoral Program of Higher Education of China (Grant No. 20120042120035).

input-to-state stability (ISS) property to derive the error thresholds for triggering (see Tabuada [2007], Anta and Tabuada [2010], Heemels et al. [2012], Sandee [2006], and the references therein). Especially, my recent paper Lu and Lee [2015] proposed a self-triggered receding horizon mechanism for aperiodically implementing the receding horizon controllers based on so-called relaxed dynamic programming (RDP) inequality (Grüne [2009], Grüne and Rantzer [2008], Lincoln and Rantzer [2006]). A comprehensive review on event- and self-triggered MPC literatures can be found in Lu and Lee [2015].

For distributed computation settings, asynchronous algorithms are of great importance and more realistic to implement than synchronous algorithms. The reason is that we can not do the computation for all the states simultaneously for arbitrarily large-scale systems. As we mentioned, for this case, we divide a whole system into many subsystems. Asynchronous distributed algorithms often involve many processors, and one for each subsystem, that exchange information with its neighbors. The information in one processor about other subsystems' computation maybe outdated and may not be synchronized in the updates. However, asynchronous algorithms have been reported to work surprisingly well in the field of dynamic programming Bertsekas [2013], Tsitsiklis [1994] and large-scale convex optimization Bertsekas and Tsitsiklis [1997], Liu et al. [2014].

In Giselsson and Rantzer [2010], the authors presented a suboptimal distributed MPC computation scheme based on relaxed dynamic programming. But the algorithm seems a bit limited since all subsystems must be synchronized, i.e. they all share the same sampling time. In this paper, I will explore the asynchronous algorithm for distributed MPC problem, and will extend my RDP-based self-triggered algorithm Lu and Lee [2015] for large-scale, highly distributed systems, which can maintain stability. To the author's knowledge, this is the first proposal to explore the asynchronous convergence property for distributed MPC problem.

The organization of the paper is as follows. In Section 2, we provide the definitions and the preliminary results that will be used in this paper and formulate the separable self-triggered MPC problem that will be dealt with in this paper. In Section 3, the relaxed dynamic programming approach is adopted to prolong the inter-triggering times and to reduce the number of MPC updates for each subsystem, and each subsystem update its control law asynchronously. The systematic asynchronous self-triggering scheme is provided. Section 4 presents an illustrative example and Section 5 concludes the paper.

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 $\mathbb{Z}_{[a,b)}$ denote the set $\{k \in \mathbb{Z} \mid a \leq k < b\}$.

2. PROBLEM SETUP

Consider the linear system

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

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, $y(t) \in \mathbb{R}^p$ and $g(t) \in \mathbb{R}^p$ are the state, input, (generalized) output and external signal at the time instant t, respectively. The sets $\mathbb{X}$ and $\mathbb{U}$ represent the state and input constraints. We assume that (A, B) is stabilizable, and (A, C) is observable.

In order to synthesize an optimal control law, we can formulate the problem as an infinite-horizon optimal control problem as follows.

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

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

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

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.

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

Because (A, C) is observable, i.e., we can reconstruct x_0 from y_k, $k = 0, 1, \cdots, N-1$, the optimization problem (4) is strictly convex and has a unique global minimizer. By adjusting the parameters in the matrices C, D and the vector g_k, we can easily adapt the MPC setup (4) for the regulation and the reference tracking MPC problems. Thus, we call the problem formulation (4) as the *generalized MPC setup*.

At each sampling time, solving the above optimization problem for a particular x_0, leads to a unique sequence of optimal control law 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 optimal (finite-horizon) value function is

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

The corresponding infinite horizon value is

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

The optimal control sequence $U_N^*(x(t))$ is turned into a feedback control strategy by applying the first control move to the system, i.e.,

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

2.1 Relaxed dynamic prgramming

In this paper, we will use the relaxed dynamic programming result in Lincoln and Rantzer [2006] to develop a synthesis strategy for self-triggered MPC.

The next proposition is a variant version of the main theorem stated in Lincoln and Rantzer [2006], Grüne and Rantzer [2008] for approximating the Bellman's optimality equation based on the finite-horizon value function $V^{(N)}(x(t))$ and its corresponding optimal control policy $\mu(x(t))$.

Proposition 1. Consider the system (1)-(2) and the feedback control law $\mu : \mathbb{X} \mapsto \mathbb{U}$ given as (5) that satisfies the following inequality

$$V^{(N)}(x(t)) \geq V^{(N)}(x(t+1)) + \alpha\|y(t)\|_2^2, \tag{6}$$

for a given scalar $\alpha \in (0, 1)$. Then,

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

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

The inequality (6) is refered to as the *relaxed dynamic programming (RDP) inequality.*

2.2 Separable Model Predictive Control

Introduce a non-overlapping partition of M subsystems $\mathcal{S}_i$, $i = 1, \cdots, M$. Let $x_i(t) \in \mathbb{R}^{n_i}$ denote the state of subsystem $\mathcal{S}_i$, i.e.

$$x(t) = [(x_1(t))^{\mathrm{T}}, (x_2(t))^{\mathrm{T}}, \ldots, (x_M(t))^{\mathrm{T}}]^{\mathrm{T}},$$

with $\sum_{i=1}^{M} n_i = n$. Each subsystem $\mathcal{S}_i$ has its own input vector $u_i(t) \in \mathbb{R}^{m_i}$ and output vector $y_i(t) \in \mathbb{R}^{p_i}$,

$$u(t) = [(u_1(t))^{\mathrm{T}}, (u_2(t))^{\mathrm{T}}, \ldots, (u_M(t))^{\mathrm{T}}]^{\mathrm{T}},$$
$$y(t) = [(y_1(t))^{\mathrm{T}}, (y_2(t))^{\mathrm{T}}, \ldots, (y_M(t))^{\mathrm{T}}]^{\mathrm{T}},$$

with $\sum_{i=1}^{M} m_i = m$ and $\sum_{i=1}^{M} p_i = p$. For the linear systems (1)-(2). Introduce a non-overlapping partition of M subsystems $\mathcal{S}_i$, $i = 1, \cdots, M$.

The dynamics for subsystem $\mathcal{S}_i$ can be expressed as

$$x_i(t+1) = A_{ii}x_i(t) + B_{ii}u_i(t) + \sum_{j \in \mathcal{N}_i} (A_{ij}x_j(t) + B_{ij}u_j(t)),$$
$$x_i(0) = \bar{x}_i,$$
$$y_i(t) = C_{ii}x_i(t) + D_{ii}u_i(t) + g_i(t).$$

Let $L_{ij} = \begin{bmatrix} A_{ij} & B_{ij} \\ C_{ij} & D_{ij} \end{bmatrix}$ denote the interconnection matrices. The set of neighbors of subsystem $\mathcal{S}_i$, which have direct influence on subsystem $\mathcal{S}_i$, is defined by the set

$$\mathcal{N}_i = \{j \in \{1, \ldots, M\} | L_{ij} \neq 0\}, \quad i = 1, \cdots, M.$$

The global constraint sets for the state and input are defined as

$$\mathbb{X} = \mathbb{X}_1 \times \cdots \times \mathbb{X}_M, \quad \mathbb{U} = \mathbb{U}_1 \times \cdots \times \mathbb{U}_M.$$

Let $J^{(N)}$ be partitioned as

$$J^{(N)} = (J_1^{(N)}, J_2^{(N)}, \ldots, J_M^{(N)}),$$

where $J_i^{(N)}$ is the cost function for the subsystem $\mathcal{S}_i$.

Consider the MPC setup for the subsystem $\mathcal{S}_i$

$$\min_{\mathbf{u}_i \triangleq [u_0^{i\,\mathrm{T}}, \ldots, u_{N-1}^{i\,\mathrm{T}}]^{\mathrm{T}}} J_i^{(N)}(x_i(t), x_j(\tau_{ij}(t)), u_j(\tau_{ij}(t)),$$
$$x_j(\tau_{ij}(t)+1), u_j(\tau_{ij}(t)+1), \ldots, x_j(\tau_{ij}(t)+N-1), \mathbf{u}_i) \tag{7}$$

$$\text{s.t. } x_{k+1}^i = A_{ii}x_k^i + B_{ii}u_k^i + \sum_{j \in \mathcal{N}_i} (A_{ij}x_k^j + B_{ij}u_k^j), \tag{7.1}$$
$$y_k^i = C_{ii}x_k^i + D_{ii}u_k^i + g_k^i, \tag{7.2}$$
$$x_0^i = x_i(t), \tag{7.3}$$
$$(x_0^j, u_0^j, x_1^j, u_1^j, \ldots, x_{N-1}^j, u_{N-1}^j) = (x_j(\tau_{ij}(t)), u_j(\tau_{ij}(t)),$$
$$x_j(\tau_{ij}(t)+1), u_j(\tau_{ij}(t)+1), \ldots, x_j(\tau_{ij}(t)+N-1),$$
$$u_j(\tau_{ij}(t)+N-1)), \tag{7.4}$$
$$x_k^i \in \mathbb{X}_i, \quad k = 0, 1, \ldots, N, \tag{7.5}$$
$$u_k^i \in \mathbb{U}_i, \quad k = 0, 1, \ldots, N-1, . \tag{7.6}$$

where $J_i^{(N)} \triangleq \sum_{k=0}^{N-1} \|y_k^i\|_2^2$, and τ_{ij} in (7.4) denotes the time when processor i receives the information from its neighbors, and $t - \tau_{ij}(t)$ are communication "delays" with $\tau_{ij}(t) \leq t$. This means the values of neighbors' information (x_k^j, u_k^j), $k = 0, 1, \ldots, N-1$, we use to do the MPC update for subsystem i may not be up-to-date, and different processors are used for different subsystems to calculate the local MPC law separately. Denote the MPC trajectory obtained by (7) as

$$\xi_i = (x_0^i, u_0^i, \ldots, x_{N-1}^i, u_{N-1}^i), \text{ and}$$
$$\xi = (\xi_1, \xi_2, \ldots, \xi_M).$$

One local processor i may have out-of-date information about the computations of other processors. Thus, processor i's knowledge of a vector $\xi^i(t)$ may contain out of date information of $\xi(t)$, where

$$\xi^i(t) = [\xi_1(\tau_{i1}(t)), \xi_2(\tau_{i2}(t)), \ldots, \xi_M(\tau_{iM}(t))],$$

and $\xi_i(\tau_{ii}(t)) = 0$.

At each sampling time for subsystem $\mathcal{S}_i$ denoted as t^i, solving the optimization problem (7) leads to a sequence of control law from time t^i to time $t^i + N - 1$, given by

$$U_N^i(x_i(t), \xi^i(t)) = [u_0^{i\,\mathrm{T}}(x_i(t), \xi^i(t)), u_1^{i\,\mathrm{T}}(x_i(t), \xi^i(t)), \ldots,$$
$$u_{N-1}^{i\,\mathrm{T}}(x_i(t), \xi^i(t))]^{\mathrm{T}}.$$

For the subsystem $\mathcal{S}_i$, denote

$$V_i^{(N)}(x_i(t), \xi^i(t)) \triangleq J_i^{(N)}(x_i(t), \xi^i(t), U_N^i(x_i(t), \xi^i(t))), \tag{8}$$

and we use separate processor i to compute the local MPC control law and update the value $V_i^{(N)}(x_i(t), \xi^i(t))$. Hence, we call this kind of MPC update schemes as "**separable MPC**". $V_i^{(N)}(x_i(t_0), \xi^i(t_0))$ and $\xi^i(t_0)$ can be computed as a warm start for the result in this paper from any existed distributed optimization algorithms for MPC, for instance, ADMM algorithm for separable MPC (Lu [2014]).

Let $\mathbb{I}_\infty$ denote the admissible finite solution set of the infinite horizon LQ problem. As stated in paper Primbs and Nevistić [2000], $\mathbb{I}$ can be an arbitrary compact subset of $\mathbb{I}_\infty$ which is forward invariant for horizon length N under the receding horizon policy. Denote

$$\mathbb{I} = \mathbb{I}_1 \times \mathbb{I}_2 \times \cdots \times \mathbb{I}_M.$$

Assumption 1. For each subsystem $\mathcal{S}_i$, we assume the infinite horizon cost $V_i^{(\infty)}(x_i(t_0), \xi^i(t_0))$ is finite and bounded by a positive scalar $\bar{V}_i$ for $x(t_0) \in \mathbb{I}_i$, that is, $\mathbb{I}_i$ is assumed to be control invariant, for $i = 1, \ldots, M$.

2.3 Asynchronous Self-triggered MPC problem

Define the sets of triggering times of subsystem $\mathcal{S}_i$ as $\mathcal{T}_i := \{t_l^i \mid l \in \mathbb{Z}_+\}$, which satisfy $t_{l+1}^i > t_l^i$ for all $l \in \mathbb{Z}_+$. In between the interval $[t_l, t_{l+1})$,

$$u_i(t) := u^i_{(t-t_l^i)}(x_i(t_l^i), \xi^i(t_l^i)), \quad t \in \mathbb{Z}_{[t_l^i, t_{l+1}^i)}. \tag{9}$$

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

In order to achieve convergence in our asynchronous self-triggered algorithm, we make the following assumptions.

Assumption 2. $\lim_{t\to\infty} \tau_{ij}(t) = \infty$ for all $i, j = 1, \ldots, M$.

Assumption 3. Every processor update infinitely often, i.e. the set of times $t \in \mathcal{T}_i$ that processor i updates $V_i^{(N)}(x_i(t), \xi^i(t))$ is infinite, for $i = 1, \ldots, M$.

Set an extended state $z_i(t) = (x_i(t), \xi^i(t))$, and define the set

$$\Psi_i = \{z_i \in \mathbb{I}_i \times \mathbb{I} \times \mathbb{U} \mid V_i^{(\infty)}(z_i) \le \bar{V}_i\}.$$

For all $z_i \in \Psi_i$, we can express the asynchronous update law for processor i as

$$V_i^{(N)}(z_i(t)) = \begin{cases} J_i^{(N)}(z_i(t), U_N^i(z_i(t)), & t \in \mathcal{T}_i, \\ V_i^{(N)}(z_i(t-1)), & t \notin \mathcal{T}_i \cup \{0\}. \end{cases} \tag{10}$$

Let $z = (z_1, z_2, \ldots, z_M)$ be the collection of the information that possesses by each processor i for $i = 1, \ldots, M$, and $\Psi = \Psi_1 \times \Psi_2 \times \cdots \times \Psi_M$, $V^{(N)}(z) = (V_1^{(N)}(z_1), V_2^{(N)}(z_2), \ldots, V_M^{(N)}(z_M))$, and denote by $R(\Psi)$ for the set of these real-valued functions defined on Ψ.

The following Proposition is an amended theorem of [Proposition 2.6.1 in the book Bertsekas [2013]], which is essential for proving convergence of the asynchronous separable self-triggered MPC algorithm in this paper.

Proposition 2. (Asynchronous Convergence Theorem) For the MPC setup (7), let Assumptions 1,2 and 3 hold, and assume that there's a sequence of nonempty subsets $\{S(h)\} \subset R(\Psi)$ with $S(h+1) \subset S(h)$ for all h, and with the following properties:

(1) Synchronous Convergence Condition: We have

$$V^{(N)}(z(t_{h+1})) \in S(h+1), \quad \forall V^{(N)}(z(t_h)) \in S(h), h \in \mathbb{Z}.$$

(2) Box Condition: For all h, $S(h)$ is a Cartesian product of the form

$$S(h) = S_1(h) \times S_2(h) \times \ldots \times S_M(h),$$

where $S_i(h)$ is a set of real-valued functions for subsystem $\mathcal{S}_i$, $i = 1, 2, \cdots, M$.

Then for every $V^{(N)}(z(0)) \in S(0)$, the sequence $\{V^{(N)}(z(t))\}$ generated by the asynchronous algorithm (10) converges pointwise to a fixed point of $V^{(N)}$.

Proof. We show by induction that for each $h \in \mathbb{Z}$, there is a time t_h such that:

(a) $V^{(N)}(z(t)) \in S(h)$ for all $t \ge t_h$.

(b) For all subsystems $\mathcal{S}_i$'s and $t \in \mathcal{T}_i$ with $t \ge t_h$, we have

$$(V_1^{(N)}(z_1(\tau_{i1}(t))), V_2^{(N)}(z_2(\tau_{i2}(t))), \ldots, V_i^{(N)}(z_i(\tau_{ii}(t))), \ldots, V_M^{(N)}(z_M(\tau_{iM}(t)))) \in S(h), \quad \tau_{ii}(t) = t.$$

The induction hypothesis is true for $h = 0$, since $V^{(N)}(z(t_0)) \in S(0)$. Assuming it is true for a given h, we will show that there exists a time t_{h+1} with the required properties. For each $i = 1, 2, \ldots, M$, let t^i be the first element of $\mathcal{T}_i$ such that $t^i \ge t_h$. Applying the Synchronous Convergence Condition, i.e. set $V_j^{(N)}(z_j(\tau_{ij}(t^i))) = J_j^{(N)}(z_j(\tau_{ij}(t^i))), U_N^j(z_j(\tau_{ij}(t^i)))$ for $j = 1, 2, \ldots, M$, then we have

$$(V_1^{(N)}(z_1(\tau_{i1}(t^i))), V_2^{(N)}(z_2(\tau_{i2}(t^i))), \ldots, V_i^{(N)}(z_i(t^i)), \ldots, V_M^{(N)}(z_M(\tau_{iM}(t^i)))) \in S(h),$$

implying (in view of the Box Condition) that

$$V_i^{(N)}(z_i(t^i)) \in S_i(h+1).$$

Similarly, for every $t \in \mathcal{T}_i$, $t \ge t^i$, we have $V_i^{(N)}(z_i(t+1)) \in S_i(h+1)$. Between elements of $\mathcal{T}_i$, $V_i^{(N)}(z_i(t))$ does not change. Thus,

$$V_i^{(N)}(z_i(t)) \in S_i(h+1), \quad \forall t \ge t^i.$$

Let ${t_h}' = \max_i\{t^i\}$. Then, using the Box Condition we have

$$V^{(N)}(z(t)) \in S(h+1), \quad \forall t \ge {t_h}'.$$

Finally, by Assumption , we have $\tau_{ij}(t) \to \infty$ as $t \to \infty$, $t \in \mathcal{T}_i$, we can choose a time $t_{h+1} \ge {t_h}'$ that is sufficiently large so that $\tau_{ij}(t) \ge {t_h}'$ for all i, j and $t \in \mathcal{T}_i$ with $t \ge t_{h+1}$. We then have, $V_j^{(N)}(z_i(t)) \in S_j(h+1)$, for all $t \in \mathcal{T}_i$ with $t \ge t_{h+1}$ and all $j = 1, \ldots, M$, which (by the Box Condition) implies that

$$(V_1^{(N)}(z_1(\tau_{i1}(t))), V_2^{(N)}(z_2(\tau_{i2}(t))), \ldots, V_i^{(N)}(z_i(t)), \ldots, V_M^{(N)}(z_M(\tau_{iM}(t)))) \in S(h+1).$$

The induction is complete. □

3. RDP-BASED APPROACH

When the event triggered at time instant t_l^i, we have to decide both the control law $u_0^i(z_i(t_l^i))$ and the next triggering time t_{l+1}^i such that $t_{l+1}^i (< t_l^i + N)$ is as large as possible while still guaranteeing the relaxed dynamic programming inequality condition.

Next, we present the main theorem of our asynchronous distributed self-triggered scheme, which can be verified locally for each subsystem.

Theorem 1. Let Assumptions 1-3 hold. If an upper bound $\bar{V}_i^{(N)}(z_i(t))$ can be found for $t \in \{t_l^i \mid l \in \mathbb{Z}_+\}$ such that

$$\bar{V}_i^{(N)}(z_i(t)) \ge V_i^{(N)}(z_i(t)) \tag{11}$$

and

$$V_i^{(N)}(z_i(t_l^i)) - \bar{V}_i^{(N)}(z_i(t_{l+1}^i)) \geq e_i(t_l^i) + \alpha \sum_{t=t_l^i}^{t_{l+1}^i - 1} \|y_i(t)\|_2^2, \quad (12)$$

satisfied for all $i = 1, \ldots, M$ and a given scalar $\alpha \in (0, 1)$, where the sequence $\{e(t_l^i)\}$ is

$$e_i(t_l^i) = e_i(t_{l-1}^i) + \alpha \sum_{t=t_{l-1}^i}^{t_l^i - 1} \|y_i(t)\|_2^2 + \bar{V}_i^{(N)}(z_i(t_l^i)) - \bar{V}_i^{(N)}(z_i(t_{l-1}^i))$$

for all $l \geq 2$ and

$$e_i(t_1^i) = \alpha \sum_{t=t_0^i}^{t_1^i - 1} \|y_i(t)\|_2^2 + \bar{V}_i^{(N)}(z_i(t_1^i)) - V_i^{(N)}(z_i(t_0^i)),$$

and $e_i(t_1^i) = 0$. Then

$$\lim_{t \to \infty} y_i(t) = 0, 1 \leq i \leq M.$$

Proof. First, we prove the synchronous convergence condition of the Proposition 2 by letting the triggering time be synchronized, that is, if one subsystem is triggered, we do MPC for all subsystems simultaneously. They all share the same sampling time $t_l = \min t_l^i$ and we replace t_l^i with t_l in (11) and (12). Since $e_i(t_l) < 0$, summation of the conditions (11) and (12) over $i = 1, \ldots, M$ implies the relaxed dynamic programming inequality

$$V^{(N)}(z(t_l)) \geq V^{(N)}(z(t_{l+1})) + \alpha \sum_{t=t_l}^{t_{l+1}-1} \|y(t)\|_2^2, \quad (13)$$

which means for $t \in \{t_l \mid l \in \mathbb{Z}_+\}$, $V^{(N)}(z(t))$ converges, as t increases. Thus, we prove the synchronous condition.

We observe from condition (11), the sup-norm spheres of the Cartesian product of the upper bound $\|\bar{V}_1^{(N)}\|_\infty \times \|\bar{V}_2^{(N)}\|_\infty \times \cdots, \times \|\bar{V}_M^{(N)}\|_\infty$ satisfies the Box Condition in the Proposition 2, and from Assumption 1 and (12), we can conclude that this box converges to the optimal value 0 as we proceeding the MPC updates. Hence, if Assumptions 1-3 and the local conditions (11) and (12) are satisfied, we can conclude that $V^{(N)}(z(t)) \to 0$ as $t \to \infty$ by the asynchronous update law (10). Thus, $\lim_{t\to\infty} y_i(t) = 0, 1 \leq i \leq M$. This completes the proof. □

Distributed Triggering rule:

At an distributed MPC triggering time $t_l^i \in \mathbb{Z}_+$ with $l \in \mathbb{Z}_+$, we solve the distributed MPC problem (7), and the next DMPC triggering time t_{l+1}^i can be calculated by

$$t_{l+1}^i = t_l^i + \mathcal{M}_{t_l^i}(x_i(t_l^i)),$$
$$u_i(t) = u_{(t-t_l^i)}^i(x_i(t_l^i)), \quad t \in \mathbb{Z}_{[t_l^i, t_{l+1}^i)},$$

where

$$\mathcal{M}_{t_l^i}(x_i(t_l^i)) \triangleq \sup\{N_{t_l^i} \in [1, \ldots, N-1]\}$$
$$\text{s.t.} V_i^{(N)}(x_i(t_l^i)) - \bar{V}_i^{(N)}(x_i(t_l^i + N_{t_l^i})) \geq e_i(t_l^i) + \alpha \left(\sum_{t=t_l^i}^{t_l^i + N_{t_l^i} - 1} \|y_i(t)\|_2^2 \right).$$

In order to calculate an upper bound $\bar{V}_i^{(N)}(x_i(t_l^i + N_{t_l^i}))$, we will apply a "shifted" input sequence $\bar{U}_{N^i}^i(x_i(t_l^i + N_{t_l^i}), \xi^i(t_l^i + N_{t_l^i})) = [\, {u_{N_{t_l^i}}^i}^{\mathrm{T}}(x_i(t_l^i + N_{t_l^i}), \xi^i(t_l^i + N_{t_l^i})), \ldots, {u_{N^i-1}^i}^{\mathrm{T}}(x_i(t_l^i + N_{t_l^i}), \xi^i(t_l^i + N_{t_l^i})), 0_{m_i \times N_{t_l^i}} \,]^{\mathrm{T}}$.

Hence, the upper bound

$$\bar{V}_i^{(N^i)}(x^i(t_l^i + N_{t_l^i})) \triangleq J_i^{(N^i)}(x^i(t_l^i + N_{t_l^i}), \xi^i(t_l^i + N_{t_l^i}), \bar{U}_N^i(x^i(t_l^i + N_{t_l^i}), \xi^i(t_l^i + N_{t_l^i}))).$$

4. ILLUSTRATIVE EXAMPLE

Let's consider a model that is similar to the numerical example in Giselsson and Rantzer [2013] which is randomly generated. The dynamical system consists three subsystems and each subsystem has five states and one input. As the problem is a regulation problem, the output of the ith subsystem 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 state and input constraints are set within the range $[-2, 2]$. The control horizon is chosen to $N = 6$ with $\alpha = 0.8$. If the system's states reach a small neighborhood of the origin, the simulation is terminated. The simulation results are presented in Fig. 1. The circle dots trajectories represent the simulation results of synchronous self-triggered MPC based on RDP in Lu and Lee [2015], and the star dots trajectories represent the simulation results for asynchronous self-triggered MPC. The MPC triggering times are recorded in Fig. 2.

5. CONCLUSIONS

This paper proposed an efficient MPC implementation scheme for large-scale distributed systems with a relaxed dynamic programming based self-triggered MPC synthesis procedure. All the subsystems update their control policy asynchronously, and the intervals between each subsystem's local MPC updates are maximized. With the proposed self-triggered MPC synthesis procedure, the overall closed-loop systems can still achieve asymptotic stability and constraint satisfaction.

REFERENCES

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

D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Athena Scientific, 1997.

D. P. Bertsekas. *Abstract Dynamic Programming.* Athena Scientific, 2013.

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.

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.

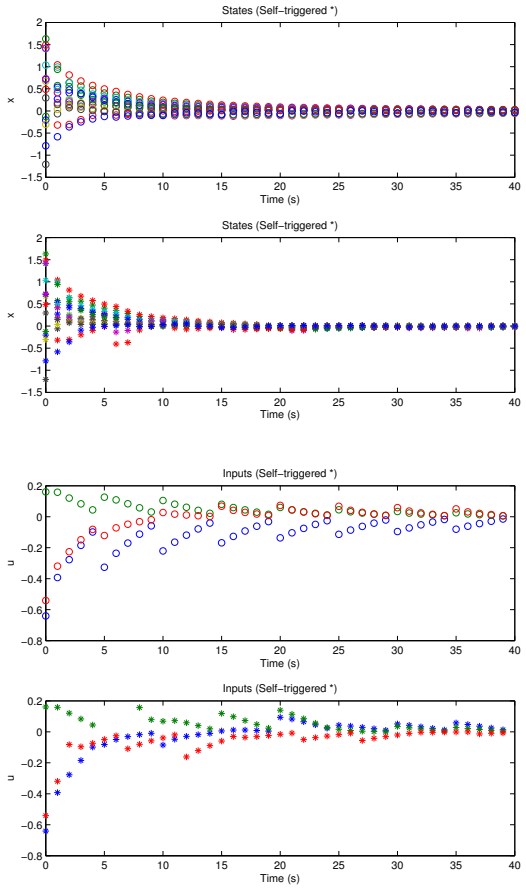

Fig. 1. State and input trajectories for $N = 6$. Circle dots for synchronous self-triggered MPC and star dots for asynchronous self-triggered MPC.

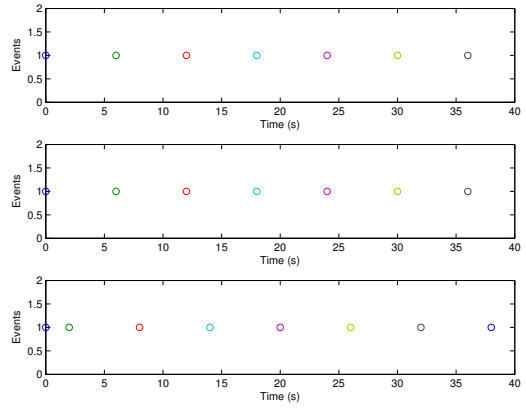

Fig. 2. Event triggering instants. The triggering instants are marked with the circles with the value 1.

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.

P. Giselsson and A. Rantzer. On feasibility, stability and performance in distributed model predictive control. *IEEE Transactions on Automatic Control*, , vol. 59, issue 4, pp. 1031–1036, 2013.

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.

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.

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.

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

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

J. Liu, S. J. Wright, C. Ré, V. Bittorf and S. Sridhar. An Asynchronous Parallel Stochastic Coordinate Descent Algorithm. *arXiv preprint*, version 3, Nov 2014.

L. Lu. Separable constrained model predictive control via alternating direction method of multipliers. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, 2014.

L. Lu and J. H. Lee. Synthesis of self-triggered receding horizon controllers: A relaxed dynamic programming approach. *submitted to Chinese Control Conference 2015.*

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

M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers and Chemical Engineering*, vol. 23, pp. 667–682, 1999.

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.

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

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.

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

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

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.

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

J. N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, vol. 16, pp. 185–202, 1994.