IEICE TRANS. FUNDAMENTALS, VOL.E99–A, NO.2 FEBRUARY 2016

PAPER Special Section on Mathematical Systems Science and its Applications

Event-Triggered and Self-Triggered Control for Networked Control Systems Using Online Optimization

Koichi KOBAYASHI[†a)] ***and*** **Kunihiko HIRAISHI**[††]**,** ***Members***

SUMMARY Event-triggered and self-triggered control methods are an important control strategy in networked control systems. Event-triggered control is a method that the measured signal is sent to the controller (i.e., the control input is recomputed) only when a certain condition is satisfied. Self-triggered control is a method that the control input and the (non-uniform) sampling interval are computed simultaneously. In this paper, we propose new methods of event-triggered control and self-triggered control from the viewpoint of online optimization (i.e., model predictive control). In self-triggered control, the control input and the sampling interval are obtained by solving a pair of a quadratic programming (QP) problem and a mixed integer linear programming (MILP) problem. In event-triggered control, whether the control input is updated or not is determined by solving two QP problems. The effectiveness of the proposed methods is presented by numerical examples.

key words: *event-triggered control, self-triggered control, networked control systems, model predictive control*

1. Introduction

In recent years, much attention has been paid to analysis and synthesis of networked control systems (NCSs) [1], [4]. An NCS is a control system where components such as plants, sensors, controllers, and actuators are connected through communication networks. In each component, messages about the control input or the measured output are sent and received (see Fig. 1). In distributed control systems, subsystems are frequently connected via communication networks, and it is important to consider analysis and synthesis of such systems from the viewpoint of NCSs. In design of NCSs, there are several technical issues such as packet losses, transmission delays, communication constraints. However, it is difficult to consider these issues in a unified way, and it is suitable to discuss an individual issue. From this viewpoint, several results have been obtained so far (see e.g., [10]–[13]).

In this paper, we focus on the technical issue on sampling intervals. The sampling interval is chosen based on CPU processing time, communication bandwidth, and so on. Then, transmissions from the controller (the sensor) to the actuator (the controller) occur at each sampling time.

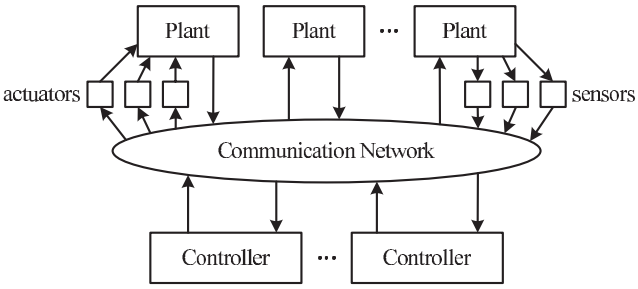

Fig. 1 Illustration of networked control systems. Actuators and sensors may be located in a distributed way.

However, in NCSs, it is desirable that sending and receiving messages should occur, only when there exists important information. In this sense, uniform sampling intervals are not necessarily suitable.

From this viewpoint, event-triggered control and self-triggered control have been proposed so far (see e.g., [2], [3], [7]–[9], [14], [15], [18], [19], [21]–[24]). In event-triggered control, the measured signal is sent to the controller only when a certain triggering condition on the measured signal is satisfied. In this method, it is important to consider how to design the triggering condition, i.e., the condition for sending the measured signal to the controller (equivalently, the condition for recomputing the control input using the measured signal). In self-triggered control, the next sampling time at which the control input is recomputed is computed. That is, both the sampling interval and the control input are computed simultaneously. In many existing works, first, the continuous-time controller is obtained, and after that, the sampling interval such that stability is preserved is computed.

To the best of our knowledge, few results using online optimization have been obtained so far. For example, in [22], [24], a design method based one-step finite horizon boundary has been proposed. In this method, the first sampling interval such that the optimal value of the cost function is improved, is computed under the constraint that other sampling periods are given as a constant. However, a nonlinear equation must be solved, and input constraints cannot be considered. In [14], [15], the authors have proposed an approximate solution method and an iterative solution method for self-triggered model predictive control (MPC). The MPC method in which the optimal control problem is solved at each time is one of online optimization methods. An approximate solution can be found fast, but the long

Manuscript received April 7, 2015.
Manuscript revised August 19, 2015.

[†]The author is with the Graduate School of Information Science and Technology, Hokkaido University, Sapporo-shi, 060-0814 Japan.
[††]The author is with the School of Information Science, Japan Advanced Institute of Science and Technology, Nomi-shi, 923-1292 Japan.
a) E-mail: k-kobaya@ssi.ist.hokudai.ac.jp

DOI: 10.1587/transfun.E99.A.468

computation time is generally required for deriving an exact solution. In [20], self-triggered predictive control for mixed logical dynamical systems has been studied.

In this paper, motivated by periodic event-triggered control [9], [21], we propose new methods of event-triggered control and self-triggered control from the viewpoint of online optimization (i.e., MPC). In periodic event-triggered control, the triggering condition is verified at each sampling time, where the sampling interval is given. In the proposed event-triggered control method, the triggering condition is given by two QP problems. By the triggering condition, we determine whether the control input at the next sampling time is updated or not. In the proposed self-triggered control method, we determine sampling times that update of the control input is skipped. In the proposed method, skip of the sampling time is determined by solving a pair of a QP problem and an MILP problem. The effectiveness of the proposed methods is presented by numerical examples. The proposed methods provide us a basic framework for event-triggered and self-triggered control methods.

Notation: Let $\mathcal{R}$ denote the set of real numbers. Let I_n, $0_{m\times n}$ denote the $n\times n$ identity matrix, the $m\times n$ zero matrix, respectively. For simplicity, we sometimes use the symbol 0 instead of $0_{m\times n}$, and the symbol I instead of I_n.

2. Problem Formulation

In this section, we formulate the self-triggered and event-triggered control problems. After a plant is explained as preliminaries, each problem is explained.

2.1 Preliminaries

Consider the following continuous-time linear system:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{1}$$

where $x(t) \in \mathcal{R}^n$ is the state, and $u(t) \in \mathcal{R}^m$ is the control input. For $u(t)$, the constraint $u_{\min} \le u(t) \le u_{\max}$ is imposed, where the vectors $u_{\min}, u_{\max} \in \mathcal{R}^m$ are a given constant vector. Let t_k, $k = 0, 1, \ldots$ denote the sampling time. The sampling interval is given by $h := t_{k+1} - t_k$, which is a non-negative constant. Assume that the control input is piecewise constant, that is, the control input is given by

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

Hereafter, we denote $x(t_k)$ and $u(t_k)$ as x_k and u_k, respectively. In addition, assume that a pair (A, B) is stabilizable.

Consider the following conventional finite-time sampled-data optimal control problem.

Problem 1: Suppose that for the system (1), the initial time t_0, the initial state x_0, and the prediction horizon N are given. Then, find a control input $u_0, u_1, \ldots, u_{N-1}$ minimizing the following cost function

$$J(x_0, u(t)) = \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} \left\{ x^T(t)Qx(t) + u^T(t)Ru(t) \right\} dt + x_N^T Q_f x_N, \tag{2}$$

where the weights Q and Q_f are positive semi-definite, and the weight R is positive definite.

Let J_0^* denote the optimal value of the cost function (2) in Problem 1.

Here, we explain one of the methods to choose Q_f. The weight Q_f may be given by a solution $P(h)$ of the following discrete-time algebraic Riccati equation

$$\begin{aligned}&\tilde{A}^T(h)P(h)\tilde{A}(h) - P(h) - (\tilde{A}^T(h)P(h)\tilde{B}(h) + \tilde{S}(h))\\ &\times(\tilde{B}^T(h)P(h)\tilde{B}(h) + \tilde{R}(h))^{-1}\\ &\times(\tilde{B}^T(h)P(h)\tilde{A}(h) + \tilde{S}^T(h)) + \tilde{Q}(h) = 0,\end{aligned}$$

where

$$\tilde{A}(h) := e^{Ah}, \quad \tilde{B}(h) := \int_0^h e^{A\tau} d\tau B,$$

and

$$\begin{bmatrix} \tilde{Q}(h) & \tilde{S}(h) \\ \tilde{S}^T(h) & \tilde{R}(h) \end{bmatrix} := \int_0^h e^{F^T t} \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} e^{Ft} dt,$$
$$F := \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}.$$

Then, $x_N^T Q_f x_N$, $Q_f = P(h)$ is the optimal value of the cost function $J = \int_{t_N}^{\infty} \left\{ x^T(t)Qx(t) + u^T(t)Ru(t) \right\} dt$ under no input constraint. Hence, we can approximately evaluate the behavior after t_N.

2.2 Self-Triggered Optimal Control Problem

Using J_0^*, consider the following self-triggered optimal control problem.

Problem 2: Suppose that for the system (1), J_0^* and the non-negative constant $N_s \le N$ are given. Then, find a control input $u_0, u_1, \ldots, u_{N-1}$ maximizing $l \in \{0, 1, \ldots, N_s\}$ satisfying the following conditions

$$u_0 = u_1 = \cdots = u_l, \quad l \le N_s, \tag{3}$$
$$J(x_0, u(t)) \le \gamma J_0^*, \tag{4}$$

where $\gamma \ge 1$ is a given constant.

In Problem 2, the time interval $[t_0, t_{l+1})$ in which the same control input is applied is maximized under the constraint (4). Hence, control performance can be adjusted by suitably giving γ. In this problem, we focus on only the time interval $[t_0, t_{l+1})$, but by applying the following procedure to generate the control input, we can realize self-triggered control.

Procedure of Self-triggered Optimal Control:

Step 1: Set $t_0 = 0$, and give the initial state x_0.

Step 2: In the controller, solve Problem 1 and Problem 2.

Step 3: Until control starts, send the message about u_0 to

the plant.

Step 4: Apply $u(t) = u_0$, $t \in [t_0, t_{l+1})$.

Step 5: In the controller, calculate the predicted state $\hat{x}_{l+1}$.

Step 6: In the controller, solve Problem 1 and Problem 2 by using $\hat{x}_{l+1}$ as x_0.

Step 7: Send the message about u_0 to the plant.

Step 8: Wait until time t_{l+1}.

Step 9: Update $t_0 := t_{l+1}$, measure x_0, send the message about x_0 to the controller, and return to Step 4.

Remark 1: In the method proposed in [20], non-uniform sampling intervals in the time interval $[0, N]$ are calculated. In Problem 2, to reduce the computational cost, we focus on only the first sampling interval.

2.3 Event-Triggered Optimal Control Problem

Next, we consider the event-triggered optimal control problem studied here. In this paper, we suppose the following situation.

(i) The measured state is sent to the controller at time t_k.

(ii) Whether the control input is sent to the plant or not is determined by solving a certain problem.

The above (i) is not imposed in the conventional event-triggered control method (see also Remark 2). However, in several cases, the state at each time is useful for after-control analysis. Hence, we focus on sending of the control input.

Consider the following problem.

Problem 3: Suppose that for the system (1), J_0^* and the past control input u_{-1} ($u(t) = u_{-1}$, $t < 0$) are given. Then, find a control input $u_0, u_1, \ldots, u_{N-1}$ maximizing $p \in \{0, 1\}$ satisfying the following conditions

$$u_{-1} = u_{p-1}, \tag{5}$$

$$J(x_0, u(t)) \le \gamma J_0^*, \tag{6}$$

where $\gamma \ge 1$ is a given constant.

In this problem, if p is obtained as $p = 1$, then $u_{-1} = u_0$ holds, that is, the control input is not changed. If p is obtained as $p = 0$, then u_{-1} and u_0 may be different. By applying the following procedure to generate the control input, we can realize event-triggered control.

Procedure of Event-triggered Optimal Control:

Step 1: Set $t_0 = 0$, and give the initial state x_0.

Step 2: In the controller, solve Problem 1.

Step 3: Until control starts, send the message about u_0 to the plant.

Step 4: Apply $u(t) = u_0$, $t \in [t_0, t_1)$.

Step 5: In the controller, calculate the predicted state $\hat{x}_1$.

Step 6: In the controller, solve Problem 1 and Problem 3 by using $\hat{x}_1$ as x_0.

Step 7: If $u_0 = u_{-1}$ holds, do not send any massage. Otherwise, send the message about u_0 to the plant.

Step 8: Wait until time t_1.

Step 9: Update $t_0 := t_1$, measure x_0, send the message about x_0 to the controller, and return to Step 4.

Remark 2: In the conventional event-triggered control method, the triggering condition, that is, the condition for sending the measured state to the controller is given by a simple function. Hence, the triggering condition is implemented in sensors, i.e., simple devices. In the proposed method, the triggering condition is given by two optimization problems (Problem 1 and Problem 3). Its implementation in sensors will be difficult in the current step. Simplification of the triggering condition is important, and will be focused on as one of the future efforts.

3. Solution Method

3.1 Solution Method for Problem 1

According to the conventional result on sampled-data control theory, Problem 1 can be equivalently rewritten as the following optimal control problem of discrete-time linear systems.

Problem 4: Find a control input sequence $u_0, u_1, \ldots, u_{N-1}$ minimizing the following cost function

$$J(x_0, u(t)) = \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} \tilde{Q}(h) & \tilde{S}(h) \\ \tilde{S}^T(h) & \tilde{R}(h) \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + x_N^T Q_f x_N \tag{7}$$

subject to

$$x_{k+1} = \tilde{A}(h)x_k + \tilde{B}(h)u_k,$$

$$u_{\min} \le u_k \le u_{\max}.$$

Next, consider reducing Problem 4 to a quadratic programming (QP) problem. Define

$$\bar{x} := \begin{bmatrix} x_0^T & x_1^T & \cdots & x_N^T \end{bmatrix}^T,$$

$$\bar{u} := \begin{bmatrix} u_0^T & u_1^T & \cdots & u_{N-1}^T \end{bmatrix}^T.$$

Then, we can obtain

$$\bar{x} = \bar{A}x_0 + \bar{B}\bar{u},$$

where

$$\bar{A} = \begin{bmatrix} I \\ \tilde{A}(h) \\ \tilde{A}^2(h) \\ \vdots \\ \tilde{A}^N(h) \end{bmatrix},$$

$$\bar{B} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \tilde{B}(h) & 0 & \cdots & \vdots \\ \tilde{A}(h)\tilde{B}(h) & \tilde{B}(h) & \cdots & \vdots \\ \tilde{A}^2(h)\tilde{B}(h) & \tilde{A}(h)\tilde{B}(h) & \cdots & \vdots \\ \vdots & \vdots & \cdots & 0 \\ \tilde{A}^{N-1}(h)\tilde{B}(h) & \tilde{A}^{N-2}(h)\tilde{B}(h) & \cdots & \tilde{B}(h) \end{bmatrix}.$$

In addition, we define

$$\bar{Q} := \text{block-diag}(\tilde{Q}(h), \tilde{Q}(h), \ldots, \tilde{Q}(h), Q_f),$$
$$\bar{S} := \begin{bmatrix} \text{block-diag}(\tilde{S}(h), \tilde{S}(h), \ldots, \tilde{S}(h)) \\ 0_{n\times(N-1)m} \end{bmatrix},$$
$$\bar{R} := \text{block-diag}(\tilde{R}(h), \tilde{R}(h), \ldots, \tilde{R}(h)).$$

Then, the cost function (7) can be rewritten as follows:

$$\begin{aligned} J &= \bar{x}^T\bar{Q}\bar{x} + 2\bar{x}^T\bar{S}\bar{u} + \bar{u}^T\bar{R}\bar{u} \\ &= \bar{u}^T L_2\bar{u} + L_1\bar{u} + L_0, \end{aligned}$$

where

$$\begin{aligned} L_2 &= \bar{R} + \bar{B}^T\bar{S} + \bar{S}^T\bar{B} + \bar{B}^T\bar{Q}\bar{B}, \\ L_1 &= 2x_0^T\bar{A}^T(\bar{S} + \bar{Q}\bar{B}), \\ L_0 &= x_0^T\bar{A}^T\bar{Q}\bar{A}x_0. \end{aligned}$$

Finally, $\bar{u}_{\min} := [\, u_{\min}^T \; u_{\min}^T \; \cdots \; u_{\min}^T \,]^T$ and $\bar{u}_{\max} := [\, u_{\max}^T \; u_{\max}^T \; \cdots \; u_{\max}^T \,]^T$ are also defined.

Under the above preparation, Problem 4 is equivalent to the following QP problem:

Problem 5: Find $\bar{u}$ minimizing the cost function $\bar{u}^T L_2\bar{u} + L_1\bar{u} + L_0$ subject to $\bar{u}_{\min} \le \bar{u} \le \bar{u}_{\max}$.

This QP problem can be solved by using a suitable solver. It may be solved by using a multi-parametric optimization approach [6], [16]. In this case, J_0^* and the optimal control input are given by a piecewise affine function with respect to x_0, and an online computational cost is reduced.

3.2 Solution Method for Problem 2

Next, consider solving Problem 2. Here, continuous variables $v_1, v_2, \ldots, v_{N_s}$ and binary variables $\delta_1, \delta_2, \ldots, \delta_{N_s}$ are introduced as follows:

$$\begin{cases} u_1 = \delta_1 u_0 + (1-\delta_1)v_1, \\ u_2 = \delta_2 u_0 + (1-\delta_2)v_2, \\ \vdots \\ u_{N_s} = \delta_{N_s} u_0 + (1-\delta_{N_s})v_{N_s}, \end{cases} \tag{8}$$

and

$$\begin{cases} \delta_2 \le \delta_1, \\ \delta_3 \le \delta_2, \\ \vdots \\ \delta_{N_s} \le \delta_{N_s-1}. \end{cases} \tag{9}$$

For example, if $\delta_1 = 1$, then $u_1 = u_0$ and $\delta_2 \le 1$ hold. The equality $u_1 = u_0$ implies that the control input is not changed at time t_1. The inequality $\delta_2 \le 1$ implies that δ_2 is a free binary variable. If $\delta_1 = 0$, then $\delta_2 \le 0$ holds, that is, $\delta_2 = \delta_3 = \cdots = \delta_{N_s} = 0$ holds from (9).

Then, Problem 2 is equivalent to the following problem.

Problem 6: Find continuous variables $\bar{u}, v_0, v_1, \ldots, v_{N_s}$ and binary variables $\delta_1, \delta_2, \ldots, \delta_{N_s}$ maximizing the cost function $\delta_1 + \delta_2 + \cdots + \delta_{N_s}$ subject to $\bar{u}_{\min} \le \bar{u} \le \bar{u}_{\max}$, (8), (9), and

$$\bar{u}^T L_2\bar{u} + L_1\bar{u} + L_0 \le \gamma J_0^*. \tag{10}$$

In this problem, a product of a binary variable and a continuous variable such as $\delta_1 v_0$ is included. Such products can be linearized by using the following lemma [5].

Lemma 1: Consider $x \in \mathcal{X} \subseteq \mathcal{R}^n$, $\delta \in \{0, 1\}$, and $g : \mathcal{R}^n \to \mathcal{R}^m$. Then, $z = \delta g(x)$ is equivalent to the following linear inequalities:

$$\underline{g}\delta \le z \le \overline{g}\delta,$$
$$g(x) - \overline{g}(1-\delta) \le z \le g(x) - \underline{g}(1-\delta),$$

where $\underline{g} = \min_{x\in\mathcal{X}} g(x)$ and $\overline{g} = \max_{x\in\mathcal{X}} g(x)$.

Using this lemma, Problem 6 can be equivalently rewritten as a mixed integer linear programming (MILP) problem with a quadratic constraint. The quadratic constraint is given by (10). This problem can be solved by using a suitable solver. We remark here that the optimal solution for Problem 6 can be obtained by solving QP problems at most N_s times.

3.3 Solution Method for Problem 3

In a similar way to the solution method for Problem 6, Problem 3 can be equivalently rewritten as an MILP problem with a quadratic constraint. Here, as the other method, we explain a simple method.

In the case of $p = 0$, solving Problem 3 is equivalent to solving Problem 1 i.e., Problem 5. Consider solving Problem 1 with the constraint $u_0 = u_{-1}$. Let J_1^* denote the optimal value of the cost function in this problem. If $J_1^* \le \gamma J_0^*$ is satisfied, then the optimal value of p is given by $p = 1$, and the optimal control input is given by the control input corresponding to J_1^*. Otherwise the optimal value of p is given by $p = 0$, and the optimal control input is given by the control input corresponding to J_0^*. Thus, the optimal solution for Problem 3 can be derived by solving two QP problems.

Remark 3: In the proposed solution methods, an MILP problem or multiple QP problems (more precisely, convex QP problems) must be solved. An MILP problem is in general NP-hard. Although a convex QP problem is solved in polynomial time, multiple QP problems must be solved. Hence, in both Problem 2 and Problem 3, the sampling interval h must be carefully chosen considering the trade-off between control performance and computation time. In Problem 2, the parameter N_s must be also chosen carefully. However, there is a possibility that the proposed approach cannot

be applied to fast dynamical systems. Such weakness is the common weakness of online optimization (i.e., model predictive control).

4. Numerical Examples

4.1 Self-Triggered Optimal Control

First, we present a numerical example on self-triggered control. Consider the following unstable system:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0.1 & -1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t).$$

The input constraint is given by $u(t) \in [-10, +10]$. Parameters in Problem 1 and Problem 2 are given as follows: $t_0 = 0$, $x_0 = [\ 10\ \ 10\ ]^T$, $h = 0.5$, $N = 10$, $N_s = 2$, $Q = 100I_2$, and $R = 1$. Then, $Q_f = P(h)$ can be derived as

$$P(h) = \begin{bmatrix} 117.79 & 17.67 \\ 17.67 & 17.55 \end{bmatrix}.$$

We present the computational result in the case of $\gamma = 1.1$. Figure 2 shows the obtained state trajectory, and Fig. 3 shows the control input trajectory. From these figures, we see that the sampling interval is non-uniform and the state converges to the origin. The control input and l in Problem 2 at each time are derived as follows:

$$\begin{aligned}
u(t) &= -10.00, \quad t \in [0, 1.5), \quad l = 2,\\
u(t) &= -7.03, \quad t \in [1.5, 3.0), \quad l = 2,\\
u(t) &= 3.10, \quad t \in [3.0, 4.0), \quad l = 1,\\
u(t) &= -1.04, \quad t \in [4.0, 5.0), \quad l = 1,\\
u(t) &= 0.18, \quad t \in [5.0, 6.5), \quad l = 2,\\
u(t) &= -0.18, \quad t \in [6.5, 7.0), \quad l = 0,\\
u(t) &= 0.01, \quad t \in [7.0, 8.5), \quad l = 2.
\end{aligned}$$

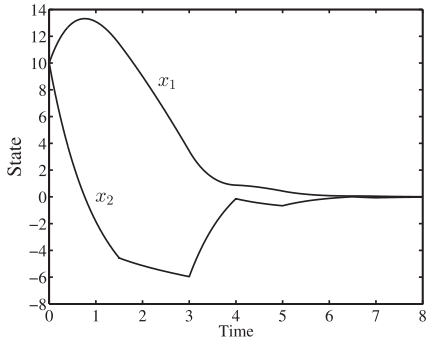

Fig. 2 State trajectory.

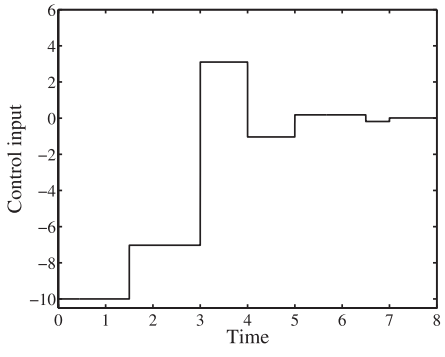

Fig. 3 Control input.

In the cases of $\gamma = 1.09$ and $\gamma = 1.11$, the sampling intervals were the same as those in the case of $\gamma = 1.1$. In the case of $\gamma = 1.0$, the sampling intervals were 0.5 sec except for the second sampling interval that is equal to 1.5 sec. In the case of $\gamma = 2.0$, the first five sampling intervals were obtained by 1.5 sec. In the case of $\gamma = 3.0$, all sampling intervals in the transient response were obtained by 1.5 sec. Thus, depending on γ, i.e., required control performance, an appropriate sampling interval can be obtained.

We comment the computation time for solving the pair of Problem 1 and Problem 2. In the above simulations, this pair was solved 30 times, where we used Gurobi Optimizer 5.6.2 on the computer with the Intel Core i7-4770K processor and the 32 GB memory. Then, the worst computation time was 0.30 sec, and the mean computation time was 0.24 sec. Since h is given by $h = 0.5$, we see that the pair of Problem 1 and Problem 2 was solved in online. Since the closed-loop system is stable in this example, we consider that the suitable h was chosen.

4.2 Event-Triggered Optimal Control

Next, we present a numerical example on event-triggered control. The plant and parameters in Problem 3 are the same as those in the previous subsection.

We present the computational result in the case of $\gamma = 1.1$. Figure 4 shows the obtained state trajectory, and Fig. 5 shows the control input trajectory. From these figures, we see that update of the control input is non-uniform and the state converges to the origin. The control input at each time is derived as follows:

$$\begin{aligned}
u(t) &= -10.00, \quad t \in [0, 2.5),\\
u(t) &= 4.95, \quad t \in [2.5, 3.0),\\
u(t) &= -1.15, \quad t \in [3.0, 4.0),\\
u(t) &= 1.29, \quad t \in [4.0, 4.5),\\
u(t) &= -0.28, \quad t \in [4.5, 5.5),\\
u(t) &= 0.33, \quad t \in [5.5, 6.0),\\
u(t) &= -0.07, \quad t \in [6.0, 6.5),\\
u(t) &= 0.01, \quad t \in [6.5, 7.5),\\
u(t) &= -0.01, \quad t \in [7.5, 8.0).
\end{aligned}$$

In other words, the control input was updated 8 times until time 8. In the case of $\gamma = 1.09$, the control input was updated 8 times until time 8. In the cases of $\gamma = 1.11$ and $\gamma = 2.0$, the control input was updated 7 times until time 8. In the cases of $\gamma = 3.0$ and $\gamma = 4.0$, the control input was

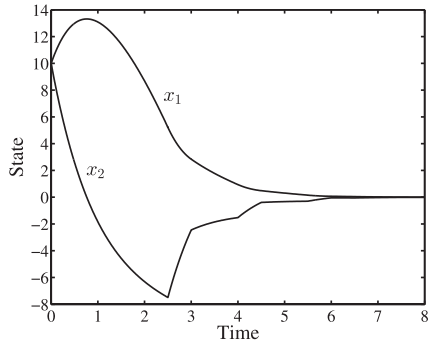

Fig. 4 State trajectory.

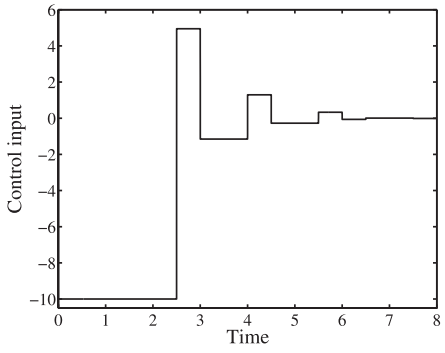

Fig. 5 Control input.

updated 5 times until time 8. In the case of $\gamma = 1.0$, the control input was updated 14 times until time 8 (i.e., the control input was almost always updated at each time). Thus, in this example, there is a trend that updating times decrease for larger γ.

We comment about the computation time for solving the pair of Problem 1 and Problem 3. In the above simulations, this pair was solved 40 times. Then, the worst computation time was 0.11 sec, and the mean computation time was 0.08 sec. Since h is given by $h = 0.5$, we see that the pair of Problem 1 and Problem 3 was solved in online. In also the case of event-triggered optimal control, the closed-loop system is stable, and we consider that the suitable h was chosen.

5. Conclusion

In this paper, we discussed the self-triggered control method and the event-triggered control method from the view point of online optimization (i.e., model predictive control). In the proposed self-triggered control method, the control input and the sampling interval are optimized by solving the QP problem and the MILP problem. In the proposed event-triggered control method, the condition for updating the control input is given by two QP problems, and the control input and the timing that the control input is sent to the plant are optimized. The proposed methods provide us a basic framework for event-triggered and self-triggered optimal control methods.

As was pointed in Remark 2, simplification of the triggering condition in the proposed event-triggered control method is one of the future efforts. Then, it is important to derive an approximate condition in which solving the optimization problem is not needed. Theoretical analysis of the relation between the sampling time interval and the parameter γ is also important. In addition, stability analysis of the closed-loop system is also one of the future efforts. Then, the method proposed in [17] will be useful.

This research was partly supported by Grant-in-Aid for Scientific Research (C) 26420412.

References

[1] C.T. Abdallah and H.G. Tanner, "Complex networked control systems: Introduction to the special section," IEEE Control Syst. Mag., vol.27, no.4, pp.30–32, 2007.
[2] A. Anta and P. Tabuada, "Self-triggered stabilization of homogeneous control systems," 2008 American Control Conference, pp.4129–4134, 2008.
[3] A. Anta and P. Tabuada, "To sample or not to sample: Self-triggered control for nonlinear systems," IEEE Trans. Autom. Control, vol.55, no.9, pp.2030–2042, 2010.
[4] P. Antsaklis and J. Baillieul, "Special issue on technology of networked control systems," Proc. IEEE, vol.95, no.1, pp.5–8, 2007.
[5] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," Automatica, vol.35, no.3, pp.407–427, 1999.
[6] F. Borrelli, Constrained Optimal Control of Linear and Hybrid Systems, Lecture Notes in Control and Information Sciences, vol.290, Springer, 2003.
[7] A. Camacho, P. Martí, M. Velasco, C. Lozoya, R. Villá, J.M. Fuertes, and E. Griful, "Self-triggered networked control systems: An experimental case study," 2010 IEEE International Conference on Industrial Technology, pp.123–128, 2010.
[8] W.P.M.H. Heemels, K.H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," Proc. 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), pp.3270–3285, 2012.
[9] W.P.M.H. Heemels, M.C.F. Donkers, and A.R. Teel, "Periodic event-triggered control for linear systems," IEEE Trans. Autom. Control, vol.58, no.4, pp.847–861, 2013.
[10] L.-S. Hu, T. Bai, P. Shi, and Z. Wu, "Sampled-data control of networked linear control systems," Automatica, vol.43, no.5, pp.903–911, 2007.
[11] H. Ishii, "Stabilization under shared communication with message losses and its limitations," Proc. 45th IEEE Conference on Decision and Control, pp.4974–4979, 2006.
[12] H. Ishii, "H^∞ control with limited communication and message losses," Syst. Control. Lett., vol.57, no.4, pp.322–331, 2008.
[13] K. Kobayashi and K. Hiraishi, "Mixed-integer-programming-based approach to optimal design of networked control systems," SICE Journal of Control, Measurement, and System Integration, vol.4, no.3, pp.243–248, 2011.
[14] K. Kobayashi and K. Hiraishi, "Self-triggered model predictive control with delay compensation for networked control systems," IEICE Trans. Fundamentals, vol.E96-A, no.5, pp.861–868, 2013.
[15] K. Kobayashi and K. Hiraishi, "Self-triggered model predictive control using optimization with prediction horizon one," Math. Probl. Eng., vol.2013, pp.1–9, 2013.
[16] M. Kvasnica, Real-Time Model Predictive Control via Multi-Parametric Programming: Theory and Tools, VDM Verlag, 2009.
[17] M. Lazar, "Flexible control Lyapunov functions," Proc. 2009 American Control Conference, pp.102–107, 2009.
[18] D. Lehmann, E. Henriksson, and K.H. Johansson, "Event-triggered

model predictive control of discrete-time linear systems subject to disturbances," Proc. 2013 European Control Conf., pp.1156–1161, 2013.

[19] M. Mazo, Jr. and P. Tabuada, "On event-triggered and self-triggered control over sensor/actuator networks," Proc. 47th IEEE Conference on Decision and Control, pp.435–440, 2008.

[20] S. Nakao and T. Ushio, "Self-triggered predictive control with time-dependent activation costs of mixed logical dynamical systems," IEICE Trans. Fundamentals, vol.E97-A, no.2, pp.476–483, 2014.

[21] R. Postoyan, A. Anta, W.P.M.H. Heemels, P. Tabuada, and D. Nesic, "Periodic event-triggered control for nonlinear systems," Proc. 52nd IEEE Conference on Decision and Control, pp.7397–7402, 2013.

[22] M. Velasco, P. Martí, J. Yépez, F.J. Ruiz, J.M. Fuertes, and E. Bini, "Qualitative analysis of a one-step finite-horizon boundary for event-driven controllers," Proc. IEEE Conference on Decision and Control and European Control Conference, pp.1662–1667, 2011.

[23] X. Wang and M.D. Lemmon, "Self-triggered feedback control systems with finite-gain $\mathcal{L}_2$ stability," IEEE Trans. Autom. Control, vol.54, no.3, pp.452–467, 2009.

[24] J. Yépez, M. Velasco, P. Martí, E.X. Martín, and J.M. Fuertes, "One-step finite horizon boundary with varying control gain for event-driven Networked Control Systems," Proc. 37th Annual Conference of the IEEE Industrial Electronics Society, IECON 2011, pp.2606–2611, 2011.

Koichi Kobayashi received the B.E. degree in 1998 and the M.E. degree in 2000 from Hosei University, and the D.E. degree in 2007 from Tokyo Institute of Technology. He is currently an Associate Professor at the Graduate School of Information Science and Technology, Hokkaido University. His research interests include control of discrete event and hybrid systems. He is a member of the IEEE, IEEJ, ISCIE, and SICE.

Kunihiko Hiraishi received from the Tokyo Institute of Technology the B.E. degree in 1983, the M.E. degree in 1985, and the D.E. degree in 1990. He is currently a Professor at the School of Information Science, Japan Advanced Institute of Science and Technology. His research interests include discrete event systems and formal verification. He is a member of the IEEE, SICE, and IPSJ.