Master's Thesis

Deep Reinforcement Learning for Self-Triggered Control

Guidance

Associate Professor Kenji KASHIMA

Ibuki TAKEUCHI

Department of Applied Mathematics and Physics

Graduate School of Informatics

Kyoto University

February 2021

Abstract

One of the control methods for continuous-time systems is the sampled-data control. This is a control method in which the system state is observed and new control inputs are communicated at periodic intervals. The disadvantage of the sampled-data control is that it requires communication at every interval even when the control performance can be maintained without updating the control inputs, which results in extra cost for communication.

In recent years, event-triggered control and self-triggered control have attracted much attention as control methods for efficient communication and control input design.

In this paper, self-triggered control is investigated. In the self-triggered control, unlike the sampled-data control and the event-triggered control, the periodic state observation is not performed. Instead, the controller itself decides the next trigger time and communicates the state observation and control input for the following time duration. For the self-triggered control, several model-based design methods have been proposed, but these methods do not explicitly consider the communication cost over a long time of control.

In this paper, we formulate an optimal self-triggered control problem where communication cost is explicitly included, which has not been considered in previous studies. To solve this problem, we consider a policy gradient method to the problem formulated in this paper. We also propose a reinforcement learning algorithm for approximate computation of the policy gradient.

Contents

1 Introduction

One of the control methods for continuous-time systems is the sampled-data control. This is a control method in which the system state is observed and new control inputs are communicated at periodic intervals. The disadvantage of the sampled-data control is that it requires communication at every interval even when the control performance can be maintained without updating the control inputs, which results in extra cost for communication.

In recent years, event-triggered control and self-triggered control have attracted much attention as control methods for efficient communication and control input design. First of all, event-triggered control is a control method that observes the system state at fixed time intervals as in the case of sampled-data control, and redesigns and communicates the control inputs only when the driving conditions are satisfied to achieve the desired control performance. Therefore, it can improve the efficiency in terms of communication cost compared with the sample value control. For event-triggered control, several model-based design methods introduced in [1] have been proposed, and model-free methods using reinforcement learning such as [2] have also been proposed.

Next, self-triggered control is described. In the self-triggered control, unlike the sampled-data control and the event-triggered control, the periodic state observation is not performed. Instead, the controller itself decides the next trigger time and communicates the state observation and control input after that time. For the self-triggered control, model-based design methods have been proposed in [3] and [4]. However, these methods do not explicitly consider the communication cost over a long time of control.

By the way, artificial intelligence is nowadays used in various situations, notably in automatic driving technology, and the development of research on the subject of artificial intelligence is remarkable. One of the concepts to realize artificial intelligence is reinforcement learning. Reinforcement learning is an algorithm that learns behaviors that optimize the long-term benefits by repeated trial and error. In addition, although not mathematically proven, reinforcement learning has been used to obtain meaningful results for nonlinear systems. In this paper, we investigate the usefulness of reinforcement learning as a method to realize the self-triggered control law.

The two main contributions of this research are

- To formulate the optimal self-triggered control problem for long-time costs explicitly considering communication cost, and to consider the policy gradient for it.
- To confirm the usefulness of reinforcement learning for self-triggered control not only for linear systems but also for non-linear systems.

2 Preliminaries

2.1 Background of Deep Reinforcement Learning

Consider a malkov decision process M given with tuple $M = \{S, A, \Phi, d_0, r, \gamma\}$. Here, S, A denotes state, action set, and $\Phi(s^{'}|s, a)$ express transition probability. Also, $d_0, r(s, a), \gamma \in [0, 1]$ are distribution of initial state, reward, discount factor respectively.

The purpose of reinforcement learning is to find a policy such that

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, J(\pi) \tag{1}$$

where evaluation function $J(\pi)$ and (state) value function $V^\pi(s)$ is given as following:

$$V^\pi(s) = \mathbb{E}_\Phi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|_{a_t=\pi(s_t)}, s_0 = s\right] \tag{2}$$

$$J(\pi) = \mathbb{E}_{s_0 \sim d_0}[V^\pi(s_0)] \tag{3}$$

The expectation $\mathbb{E}_\Phi$ takes over the transition probability.

Let us define Q function, which is useful tool for analyzing reinforcement learning. Q function is given as

$$\begin{aligned} Q^\pi(s, a) &= r(s, a) + \gamma \mathbb{E}_\Phi\left[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t)|_{a_t=\pi(s_t)}\right] \\ &= r(s, a) + \gamma \mathbb{E}_\Phi[V^\pi(s')]. \end{aligned} \tag{4}$$

As shown in (4), Q function express the value when the agent select action a freely and choose action according to the policy π from the next step. Thus, the Q-function is also known as the action value function.

2.2 Policy Iteration

There is an algorithm for achieving (1), called the policy iteration method. It consists of repeating the following two steps.

1. Policy Evaluation: Find (or approximate) action value function $Q^\pi(s, a)$.
2. Policy Improvement: Update policy as $\pi(s) = \underset{a}{\operatorname{argmax}}\, Q^\pi(s, a)$.

It is known that the optimal policy π^* can be obtained by repeating the above two steps (Policy Improvement Theorem[5]).

In the case that both the state space and the action space take discrete values, it is easy to obtain $\pi(s) = \underset{a}{\operatorname{argmax}}\, Q^\pi(s, a)$ by storing $Q^\pi(s, a)$ in a table. This is not true for the case where the state space is continuous. Since the state s takes a continuous value, it cannot be stored in a table. Therefore, DQN[7] took the approach of approximating

$Q^{\pi}(s,a)$ by parametrizing it using a neural network. Since the action space is discrete, it is still possible to obtain $\underset{a}{\operatorname{argmax}}\, Q^{\pi}(s,a)$.

In the case where both state and action space is continuous, the problem is that it is very expensive to obtain $\underset{a}{\operatorname{argmax}}\, Q^{\pi}(s,a)$. Thus, up to this point, the policy π has been determined by the Q-function, but this approach cannot be taken when both spaces are continuous. Therefore, the policy function is often parameterized as π_θ and the parameter θ is updated by gradient mothod.

2.3 Deterministic Policy Gradient Method

Silver et al.[8] finds the gradient for the evaluation function $J(\pi_\theta)$, even if the policy $\pi(s)$ is defined as deterministic policy. This gradient is known as deteministic policy gradient(DPG), and it is calculate as following theorem.

Proposition 1 (Deterministic Policy Gradient Theorem)**.** *The gradient for evaluation function $\nabla_\theta J(\pi_\theta)$ is given as,*

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^{\pi_\theta}}[\nabla_\theta \pi_\theta(s)\nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}] \tag{5}$$

where,

$$\rho^{\pi_\theta}(s) = \int_S \sum_{t=0}^{\infty} \gamma^t d_0(s_0) Pr(s_0 \to s, t, \pi_\theta)\, ds_0 \tag{6}$$

is discounted distribution. $Pr(s_0 \to s, t, \pi_\theta)$ denotes the probability of being in state s at time t when controlled from state s_0 with policy π.

We briefly describe the proof since the derivation of Theorem 1, the main result in this paper, utilizes the same argument.

Proof. First, we consider the gradient for $V^{\pi_\theta}(s)$.

$$\begin{aligned}
&\nabla_\theta V^{\pi_\theta}(s)\\
&= \nabla_\theta Q^{\pi_\theta}(s,\pi_\theta(s))\\
&= \nabla_\theta[r(s,\pi_\theta(s)) + \gamma \int_S Pr(s\to s',1,\pi_\theta)V^{\pi_\theta}(s')ds']\\
&= \nabla_\theta \pi_\theta(s)\nabla_a r(s,a)|_{a=\pi_\theta(s)}\\
&\quad + \gamma \int_S (\nabla_\theta \pi_\theta(s)\nabla_a Pr(s\to s',1,a)|_{a=\pi(s)} V^{\pi_\theta(s')}\\
&\qquad\qquad + Pr\left(s\to s',1,\pi_\theta)\nabla_\theta V^{\pi_\theta}(s')\right) ds'\\
&= \nabla_\theta \pi_\theta(s)\nabla_a[r(s,a) + \gamma \int_S Pr(s\to s',1,\pi_\theta)V^{\pi_\theta}(s')]_{a=\pi_{\theta(s)}} ds'\\
&\quad + \gamma \int_S Pr(s\to s',1,\pi_\theta)\nabla_\theta V^{\pi_\theta}(s')ds'\\
&= \nabla_\theta \pi_\theta(s)\nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)} + \gamma \int_S Pr(s\to s',1,\pi_\theta)\nabla_\theta V^{\pi_\theta}(s')ds'
\end{aligned} \tag{7}$$

By using this relation recursively, we have,

$$
\begin{aligned}
\nabla_\theta V^{\pi_\theta}(s) &= \sum_{i=0}^{\infty} \int_S \cdots \int_S Pr(s \to s', 1, \pi_\theta) Pr(s' \to s'', 1, \pi_\theta) \cdots \\
&\qquad \gamma^i \nabla_\theta \pi_\theta(s'^{\cdots\prime}) \nabla_a Q^{\pi_\theta}(s'^{\cdots\prime}, a)|_{a=\pi_\theta(s'^{\cdots\prime})} ds'^{\cdots\prime} \ldots ds' \\
&= \sum_{i=0}^{\infty} \gamma^i \int_S Pr(s \to s', i, \pi_\theta) \nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s, a)|_{a=\pi_\theta(s')} ds'. \qquad (8)
\end{aligned}
$$

Since $J(\pi) = \mathbb{E}_{s \sim d_0}[V^\pi(s)]$,

$$
\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \int_S d_0(s) V^{\pi_\theta}(s) ds \\
&= \int_S d_0(s) \nabla_\theta V^{\pi_\theta}(s) ds \\
&= \int_S \rho^{\pi_\theta} \nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s, a)|_{a=\pi_\theta(s)} ds \qquad (9)
\end{aligned}
$$

□

DDPG(Deep DPG)[9] is a deep reinforcement learning algorithm which utilize this policy gradient. It adopts an Actor-Critic structure, and learns a critic network $Q(s, a|\omega)$ which approximates Q^{π_θ}, and an actor network $\pi(s|\theta) = \pi_\theta$ which represents a policy π, respectively. The update algorithm of the actor and the critic is described below.

DDPG uses mini-batch learning. First, we describe how the critic is updated. The purpose of the critic is to approximate Q^π. Because Q function can be decomposed as (4), $Q(s, a|\omega)$ should also be updated to satisfy this relation. In view of this, parameter ω is updated toward the direction along which Temporal Difference(TD) error is minimized.

$$
Q(s, a|\omega) - \{r(s, a) + \gamma \mathbb{E}_{s'}[Q(s', \pi(s')|\omega)]\} \qquad (10)
$$

Since it is difficult to optimize for whole $(s, a) \in S \times A$ at once, DDPG optimize for the emperical data, stored in a strage called the replay buffer. To be more precise, DDPG updates the critic to minimize the MSE of the TD error for the mini-batch E created from the replay buffer. Therefore, the mini-batch E should be i.i.d.. Hence, mini-batch E is made by randomly sampling M data from the replay buffer to increase the variance of it. Then the critic is updated to minimize the loss function

$$
Loss = \frac{1}{M} \sum_{(s,a,s') \in E} \{Q(s, a|\omega) - (r(s, a) + \gamma Q(s', \pi(s')|\omega))\}^2. \qquad (11)
$$

Next, update law of the actor is described. Because the actor is the representation of policy function $\pi(s)$, policy gradient is used for its update. However, since correct Q-function as in equation (5) cannot be used in DDPG, policy gradient approximated using the critic instead of the true Q-function

$$
\mathbb{E}_{s \sim \rho^{\pi_\theta}}[\nabla_\theta \pi_\theta(s) \nabla_a Q(s, a|\omega)|_{a=\pi_\theta(s)}] \simeq \nabla_\theta J(\pi_\theta) \qquad (12)
$$

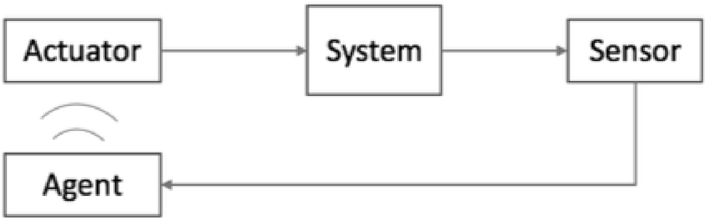

Figure 1: control system

is used. Furthurmore, the expectation is approximated as

$$\mathbb{E}_{s\sim\rho^{\pi}}[\nabla_{\theta}\pi_{\theta}(s)\nabla_{a}Q(s,a|\omega)|_{a=\pi_{\theta}(s)}] \simeq \frac{1}{M}\sum_{s\in E}[\nabla_{\theta}\pi_{\theta}(s)\nabla_{a}Q(s,a|\omega)|_{a=\pi_{\theta}(s)}]. \tag{13}$$

Therefore, the accuracy of the approximation of the policy gradient largely depends on the accuracy of critic approximation and the distribution of mini-batch E.

3 Problem Formulation

3.1 Self-Triggered Control

We consider the control system in Fig 1. Here, the system to be controled is a continuous-time system is given by

$$\dot{s} = h(s(t), u(t)) + \dot{w}. \tag{14}$$

where u and $\dot{w}$ denote control input and process noise.

In this paper, we call the observation of the state variable s and the sending of the input signal to the actuator as "interaction". In the self-triggered control, the agent does not make interaction continuously, but after a communication interval τ determined by the agent itself. In order to express it mathematically, we assume that the agent's control law π is a vector-valued function consisting of two elements, where the first element represents the input u sent to the actuator, and the second element represents the interval τ. Let t_i be the time of the i-th communication, and u_i be the input sent at t_i. The actuator continues to input u_i until the next communication time. That is, the control input $u(t)$ at time t is

$$u(t) = u_i, \forall t \in [t_i, t_{i+1}). \tag{15}$$

This control method is called Zero Order Hold (ZOH) mechanism.

3.2 Previous Research for Self-Triggered Control

Several previous researches in self-triggered control make one step optimization of the next triggering time on each interaction. For example, [3] proposed the self-triggered control strategy for discrete time linear system

$$s_{i+1} = As_i + Bu_i + Dw_i$$

where w_i is a white noise which satisfies $\mathbb{E}[w_i] = 0$ and $\mathbb{E}[w_i w_i^\top] = I$ (I is a identity matrix). They attempted to design a self-triggered control law that reduces the control cost from given initial state s_0

$$J = \sum_{i=0}^{\infty} \mathbb{E}[\alpha^i (s_i^\top Q s_i + u_i^\top R u_i)|s_0]$$

while keeping the interaction interval as long as possible. Q and R are hyper parameter matrix. For the purpose, they proposed a method to adopt the maximum interval steps η such that the set $\mathcal{U}(s)$ of inputs u satisfying

$$\mathbb{E}[(\sum_{j=0}^{\eta-1} \alpha^j (s_j^\top Q s_j + u^\top R u)) + \alpha^\eta V(s')] \leq V(s)$$

where

$$V(s) = \beta_1 s^\top P s + \beta_2 \frac{\alpha}{1-\alpha} \mathrm{tr}(PDD^\top)$$

is not empty. Here, α, β_1, β_2 is a hyper parameters, and P is a unique positive definite solution of discrete time algebraic riccati equation

$$P = A^\mathsf{T} P A - A^\mathsf{T} P B \left(R + B^\mathsf{T} P B \right)^{-1} B^\mathsf{T} P A + Q.$$

In other words, they designed a self-triggered control law which has a cost function $V(s_0)$ as a performance guarantee.

Also, [4] proposed the self-triggered control strategy for continuous time control affine system

$$\dot{s} = f(s(t)) + g(s(t))u(t).$$

The control input u is defined as a solution of Quadratic Program, such as

$$\begin{aligned} \min_u \quad & u^\top u \\ \text{s.t.} \quad & \mathcal{L}_f V(s) + \mathcal{L}_g V(s) u + \varepsilon V(s) \leq 0 \end{aligned}$$

where $\mathcal{L}_f, \mathcal{L}_g$ denote Lie derivatives, ε is a hyperparameter and V is a Lyapnov function. Then, select the maximum interval time τ where the Lyapnov function V decreases at the next step. However, these approaches do not guarantee the optimality for the long-term control cost.

In this research, we formulate an optimal self-triggered control problem in which communication is explicitly included in the long-term control cost. This makes it possible to consider the problem of finding a control policy with a long-term cost, instead of a one-step optimization.

3.3 Optimal Self-Triggered Control

In self-triggered control, the agent needs to decide the input signal u and the interval τ at each step. Thus, the action a in reinforcement learning corresponds to $\begin{bmatrix} u & \tau \end{bmatrix}^\top$. (In this paper, we equate a with the tuple (u, τ).)

Now, in this research, the control law is given as a state feedback. Therefore, the policy function is given as follows:

$$\pi(s) = \begin{bmatrix} u(s) & \tau(s) \end{bmatrix}^\top \tag{16}$$

In order to converge to the origin state as quickly as possible with the minimum input energy while reducing the frequency of communication, the agent aims to find a policy π^* that minimizes the following expected discounted cost

$$J(\pi) = \mathbb{E}_{s\sim d_0}[V^\pi(s)] \tag{17}$$

where

$$V^\pi(s) = \int_0^\infty e^{-\alpha t}\mathbb{E}_w[s(t)^\top Qs(t) + u(t)^\top Ru(t) + \beta\delta(t)C(t)|s(0) = s]\mathrm{d}t \tag{18}$$

and $C(t)$ is a boolean function which denotes the agent interact at time t, $\delta(t)$ is the Dirac's delta function.

If we separate the definite integral for each interval, we have

$$\begin{aligned} V^\pi(s) &= \sum_{i=0}^\infty \int_{t_i}^{t_{i+1}} e^{-\alpha t}\mathbb{E}_w[s(t)^\top Qs(t) + u_i^\top Ru_i + \beta\delta(t)C(t)|s(0) = s]\mathrm{d}t \\ &= \sum_{i=0}^\infty e^{-\alpha t_i}\left(\int_0^{\tau_i} e^{-\alpha t}\mathbb{E}_w[s(t)^\top Qs(t) + u_i^\top Ru_i|s(0) = s_i]\mathrm{d}t + \beta\right) \\ &= \sum_{i=0}^\infty e^{-\alpha t_i} r(s_i, \pi(s_i)). \end{aligned} \tag{19}$$

Here, t_i is the time of the i-th communication and s_i is the state at that time. Also, let $[u_i, \tau_i] = \pi(s_i)$, and let be the reward function $r(s, u, \tau)$ of each step as

$$r(s, u, \tau) = \int_0^\tau e^{-\alpha t}\mathbb{E}_w[s(t)^\top Qs(t) + u^\top Ru|s(0) = s]\mathrm{d}t + \beta \tag{20}$$

Therefore, $V^\pi(s)$ satisfies the following Bellman equation.

$$V^\pi(s) = r(s, \pi(s)) + e^{-\alpha\tau(s)}\mathbb{E}_{s'}[V^\pi(s'(s, \pi(s)))] \tag{21}$$

In addition, the action value function Q^π is the discounted accumulation cost when the agent freely chooses an action in the first step and follows the policy π from the next step, which satisfies the following Bellman equation.

$$Q^\pi(s, u, \tau) = r(s, u, \tau) + e^{-\alpha\tau}\mathbb{E}_{s'}[Q^\pi(s'(s, u, \tau), \pi(s'(s, u, \tau)))] \tag{22}$$

4 Reinforcement Learning for Self-Triggered Control

In this section, we consider the application of reinforcement learning to find the optimal self-trigger policy π^*. Simply thinking, we can formulate it as a reinforcement learning problem by taking the interaction as one step. Furthermore, DDPG may also be applied by approximating the Q-function, which satisfies the equation (22) using a critic network, to obtain the policy gradient. In this section, we discuss the validity of this approach.

4.1 Deterministic Policy Gradient for Self-Triggered Control

Since the discount factor in Equation (19) depends on τ at each step, it differs from the general reinforcement learning problem. In this subsection, we discuss how the DPG is affected by this difference.

Actually, due to the property of Q-functions such as (22), DPG cannot be computed as in (5). Since

$$\begin{aligned}
\nabla_\theta V^{\pi_\theta}(s) &= \nabla_\theta Q^{\pi_\theta}(s, \pi_\theta(s)) \\
&= \nabla_\theta [r(s, \pi_\theta(s)) + e^{-\alpha \tau_\theta(s)} \mathbb{E}_{s'}[V^{\pi_\theta}(s')] \\
&= \nabla_\theta \pi_\theta(s) \nabla_a r(s, a)|_{a=\pi_\theta(s)} \\
&\quad + e^{-\alpha\tau_\theta(s)} \int_S \{\nabla_\theta \pi_\theta(s) \nabla_a Pr(s \to s', 1, a)|_{a=\pi(s)} V^{\pi_\theta}(s') \\
&\qquad\qquad + Pr(s \to s', 1, \pi_\theta) \nabla_\theta V^{\pi_\theta}(s')\} ds' \\
&\quad + \int_S \nabla_\theta e^{-\alpha\tau_\theta(s)} Pr(s \to s', 1, \pi_\theta) V^{\pi_\theta}(s') ds',
\end{aligned} \tag{23}$$

we have

$$\begin{aligned}
&\nabla_\theta V^{\pi_\theta}(s) \\
&= \sum_{i=0}^{\infty} \int_S \cdots \int_S Pr(s_0 \to s_1, 1, \pi_\theta) \cdots Pr(s_{i-1} \to s_i, 1, \pi_\theta) \\
&\qquad e^{-\alpha t_i} \nabla_\theta \pi_\theta(s_i) \nabla_a Q^{\pi_\theta}(s, a)|_{a=\pi_\theta(s_i)} ds_i ds_{i-1} \ldots ds_1 \\
&\quad + \sum_{i=1}^{\infty} \int_S \cdots \int_S Pr(s_0 \to s_1, 1, \pi_\theta) \cdots Pr(s_{i-1} \to s_i, 1, \pi_\theta) \\
&\qquad e^{-\alpha t_{i-1}} \nabla_\theta e^{-\alpha\tau_\theta(s_{i-1})} V^{\pi_\theta}(s_i) ds_i ds_{i-1} \ldots ds_1 \\
&= \sum_{i=0}^{\infty} \int_S \cdots \int_S Pr(s_0 \to s_1, 1, \pi_\theta) \cdots Pr(s_{i-1} \to s_i, 1, \pi_\theta) \\
&\qquad e^{-\alpha t_i} \nabla_\theta \pi_\theta(s_i) \nabla_a Q^{\pi_\theta}(s, a)|_{a=\pi_\theta(s_i)} ds_i ds_{i-1} \ldots ds_1 \\
&\quad + \sum_{i=0}^{\infty} \int_S \cdots \int_S Pr(s_0 \to s_1, 1, \pi_\theta) \cdots Pr(s_i \to s_{i+1}, 1, \pi_\theta) \\
&\qquad e^{-\alpha t_i} \nabla_\theta e^{-\alpha\tau_\theta(s_i)} V^{\pi_\theta}(s_{i+1}) ds_{i+1} ds_i \ldots ds_1,
\end{aligned} \tag{24}$$

where

$$t_i = \begin{cases} 0 & (i=0) \\ \sum_{k=0}^{i-1} \tau_\theta(s_{k-1}) & (otherwise) \end{cases}. \tag{25}$$

Now, since $J(\pi_\theta) = \mathbb{E}_{s_0 \sim d_0}[V^{\pi_\theta}(s_0)]$, we have deterministic policy gradient theorem for self-triggered control.

Theorem 1 (Deterministic Policy Gradient Theorem for Self-Triggered Control)**.** *The gradient for evaluation function* (17)*,* (18) *is given by*

$$\begin{aligned}
&\nabla_\theta J(\pi_\theta) \\
&= \mathbb{E}_{s_0 \sim d_0}[\nabla_\theta V^{\pi_\theta}(s_0)] \\
&= \sum_{i=0}^{\infty} \int_S \cdots \int_S d_0(s_0) Pr(s_0 \to s_1, 1, \pi_\theta) \cdots Pr(s_i \to s_{i+1}, 1, \pi_\theta) \\
&\quad e^{-\alpha t_i}\{\nabla_\theta \pi_\theta(s_i) \nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s_i)} + \nabla_\theta e^{-\alpha \tau_\theta(s_i)} V^{\pi_\theta}(s_{i+1})\} ds_{i+1} ds_i \ldots ds_0.
\end{aligned} \tag{26}$$

4.2 Value function for Self-Triggered Control

In DPG for reinforcement learning problems with a fixed discount factor, it is sufficient that the gradient of the critic $Q(s, a|\omega)$ with respect to a can correctly approximate that of the Q-function. However, in the case of reinforcement learning for self-triggered control considered in this paper, the value of $Q(s, \pi(s)|\omega)$ itself must also be correctly approximated. Therefore, we need to pay attention to the TD learning of critic.

In this section, we discuss whether the critic learned using TD learning can approximate the value of $Q^\pi(s, a)$. First of all, since the Bellman equation for the Q-function in self-triggered control should satisfy (22), the critic should set the TD error

$$TD = Q(s, u, \tau|\omega) - \{r(s, u, \tau) + e^{-\alpha\tau}\mathbb{E}_{s'}[Q(s'(s, u, \tau), \pi(s'(s, u, \tau))|\omega)]\} \tag{27}$$

to be zero for all (s, u, τ). In this section, we discuss the algorithm for learning such a critic.

First, we discuss how to create a dataset D for training. From the equation (26), the approximation of $Q(s, \pi(s)|\omega)$ and $\nabla_a Q(s, a|\omega)|_{a=\pi(s)}$ is necessary for the calculation of the gradient direction. If the distribution of the state s in the training dataset is biased, the accuracy of the function approximation for the state s which is not in the distribution will be low. Also, for the calculation of $\nabla_a Q(s, a|\omega)|_{a=\pi(s)}$, it is necessary that the action a is distributed around the trajectory of $\pi(s)$. Then, on the state space S, we sample M states s with equal probability, and for each state s we choose

$$[u, \tau] = \pi(s) + e \tag{28}$$

and input u for the time τ. (where e is a exploration noise). The resulting reward r and the next state s' are observed, and the data tuple (s, u, τ, r, s') is stored in the dataset D. Note that, considering the stochasticity of action selection and system noise, we collect data from each state s for R times.

The critic is trained by repeatedly creating a mini-batch E by sampling m data from the dataset D, and updating the critic parameter ω toward decreasing the MSE of the TD error for the mini-batch E, using the gradient

$$g = \frac{\partial}{\partial \omega}\left[\frac{1}{m}\sum_{(s,u,\tau,r,s')\in E}\big(Q(s,u,\tau|\omega) - \{r + e^{-\alpha\tau}Q(s',\pi(s')|\omega)\}\big)^2\right]. \tag{29}$$

Algorithm 1 shows the learning algorithm for the critic described in this section.

Algorithm 1 TD Learning for Critic Network

Sample M states s with equal probability from state space S.
for $r = 0$ to R **do**
  For all sampled s, choose $[u, \tau] = \pi(s) + e$.
  Execute action u for τ second to the environment.
  Receive r and observe next state s'.
  Store (s, u, τ, r, s') to data set D.
end for
for epoch $= 0$ to N **do**
  Select m data pairs (s, u, τ, r, s') from D and make a mini-batch E.
  Calculate gradient $g = \frac{\partial}{\partial\omega}\frac{1}{m}\sum_E \big(Q(s,u,\tau|\omega) - \{r + e^{-\alpha\tau}Q(s',\pi(s')|\omega)\}\big)^2$.
  Update ω with gradient g.
end for

In the last part of this section, we compare $Q^\pi(s, \pi(s))$ for a self-triggered control law π with the critic $Q(s, \pi(s)|\omega)$ which approximates $Q^\pi(s, \pi(s))$ using the algorithm 1. Both $Q^\pi(s, \pi(s))$ and $Q(s, \pi(s)|\omega)$ are functions of state s. The state s is assumed to be two-dimensional, and the comparison between them is shown in Figure 2.

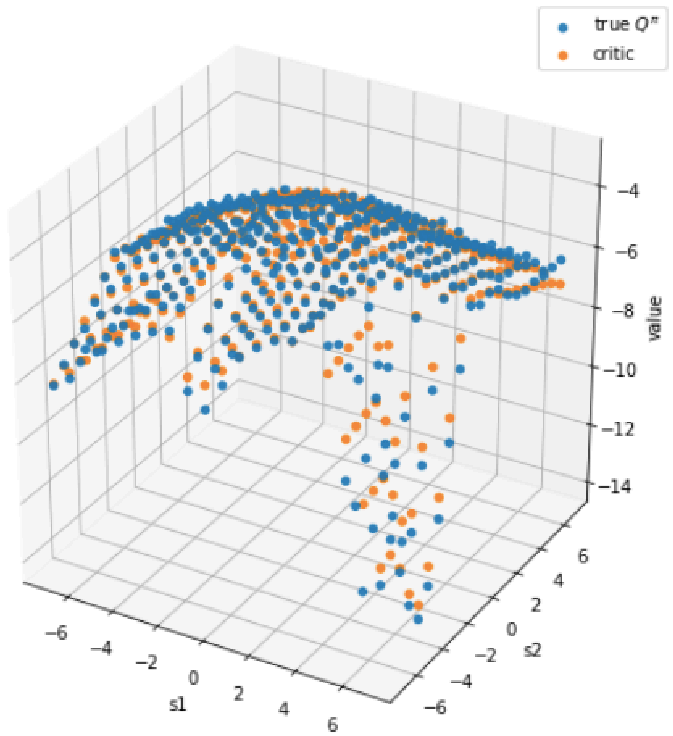

Figure 2: Approximation of $Q^\pi(s, \pi(s))$

In Figure 2, the blue points indicate the true Q^π and the orange points indicate the critics. The true Q^π is obtained by simulation. From Figure 2, we can see that the critic learned by Algorithm 1 is a good approximation of Q^π.

4.3 Naive implementation

We start by considering an naive implementation for computing the exact policy gradient (26) without considering the computational complexity. Assume that the critic is learned with Alrogithm 1. Equation (26) is the expectation of

$$\sum_{i=0}^{\infty} e^{-\alpha t_i}\{\nabla_\theta \pi_\theta(s_i)\nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s_i)} + \nabla_\theta e^{-\alpha\tau_\theta(s_i)} Q(s_{i+1}, \pi_\theta(s_{i+1})|\omega)\} \tag{30}$$

with respect to the stochasticity of the initial state distribution and the system noise. We consider the method of approximate calculations of (26) on the computer. For an initial state distribution d_0, we generate M initial states s_0, then P times controls are performed from each s_0 for time T, and

$$\sum_{i\in T_i} e^{-\alpha t_i}\{\nabla_\theta \pi_\theta(s_i)\nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s_i)} + \nabla_\theta e^{-\alpha\tau_\theta(s_i)} Q(s_{i+1}, \pi_\theta(s_{i+1})|\omega)\} \tag{31}$$

is calculated using all data pairs (s_i, s_{i+1}, t_i) experienced on each control. Here, let T_i be the set $\{i \mid t_i \le T\}$. If we take P, T and M to be infinitely large, and if $Q(s, a|\omega)$ is a good approximation of $Q^{\pi_\theta}(s, a)$, The average of the computed results of (31) for all control paths is considered to be a good approximation of (26). In algorighm 2, the reinforcement learning method with ideal calculation of policy gradient at each step.

Algorithm 2 Naive Implementation of Self-Triggered Control RL

Initialize actor $\pi_\theta(s)$ and critic $Q(s, u, \tau|\omega)$.
Learn the critic $Q(s, u, \tau|\omega)$ with algorithm 1.
for *epoch* $= 0$ to N **do**
  for $m = 0$ to M **do**
    Initialize $s_0 \sim d_0$.
    for episode $= 0$ to P **do**
      Initialize episode memory E.
      while $t \le T$ **do**
        Select $[u, \tau] = \pi_\theta(s)$.
        Execute action u for τ second to the environment.
        Receive r and observe next state s'.
        Store tuple (s, s', t) to the episode memory E.
      end while
      Calculate (31) with episode memory E.
    end for
    Take the average of (31) over P paths, and let it be $V^{\pi_\theta}(s_0)$.
  end for
end for
Take the average of $V^{\pi_\theta}(s_0)$ over the generated s_0 and let it be policy gradient g.
Update the actor with approximated policy gradient g.

4.4 Practical Implementation

As mentioned before, the above algorithm does not take into account the problem of computational complexity. We consider an efficient method to approximate the policy gradient inspiered by DDPG. The most important point is the state distribution of the mini-batch which takes the sample mean to approximate equation (26).

During the training, the agent decide

$$[u_i, \tau_i] = \pi_\theta(s_i) + e_i \tag{32}$$

and input u_i for time τ on each steps (e_i is a exploration noise). We observe the reward r_i and the next state s_{i+1}, and store the data tuple $(s_i, u_i, \tau_i, r_i, s_{i+1}, t_i)$ in the replay buffer. It differs from DDPG in that the time t_i from the start of control is stored in the replay buffer. In order to bring the diversity of data in the replay buffer, we assume that after every T seconds of control, the initial state s_0 is generated and the control is replayed again. Assuming that the actor is updated only gradually, the replay buffer stores the experience gained by policies similar to the current policy. Thus, if we create a mini-batch E by sampling the experience with probability $e^{-\alpha t_i}$, we can expect that the sample mean

$$\frac{1}{M}\sum_{(s,s')\in E}\{\nabla_\theta \pi_\theta(s)\nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s)} + \nabla_\theta e^{-\alpha\tau_\theta(s)}Q(s,\pi_\theta(s)|\omega)\} \tag{33}$$

for the mini-batch E will approximate (26) well. This is because the distribution of the mini-batch E is discounted for $e^{-\alpha t}$.

On the other hands, the critic is updated by reducing the MSE of TD error for mini-batch E. However, it is known that if the the target value

$$r + e^{-\alpha\tau}Q(s',\pi_\theta(s')|\omega) \tag{34}$$

is calculated with the actor $\pi_\theta(s)$ and the critic $Q(s,a|\omega)$, the learning proved to be unstable in many environments[2][9]. So, as in DQN and DDPG, we use target networks to stabilize the training. Target networks $\pi_{\theta'}(s), Q(s,a|\omega')$ are created as copies of $\pi_\theta(s), Q(s,a|\omega)$, and used only for the calculation of the target value. That is, the critic is updated using the gradient

$$g = \frac{\partial}{\partial\omega}\left[\frac{1}{m}\sum_{(s,u,\tau,r,s')\in E}\left(Q(s,u,\tau|\omega) - \{r + e^{-\alpha\tau}Q(s',\pi_{\theta'}(s')|\omega')\}\right)^2\right]. \tag{35}$$

After the update of θ and ω, the parameters of target networks are updated as

$$\begin{aligned}\theta' &\leftarrow (1-\xi)\theta' + \xi\theta\\ \omega' &\leftarrow (1-\xi)\omega' + \xi\omega\end{aligned} \tag{36}$$

with a hyper parameter $\xi \ll 1$.

Algorithm 3 shows the efficient algorithm utilizing these idea. We refer to Algorithm 3 as the proposed method.

5 Numerical Evaluation

In this section, we study the effectiveness of the reinforcement learning approach to the optimal self-triggered control problem. We conduct numerical experiments and review the results for the cases of linear and nonlinear control systems, respectively. In both cases, the communication interval is allowed to be

$$0.01 \le \tau \le 10.0 \tag{37}$$

Algorithm 3 Practical Implementation of Self-Triggered Control RL

Initialize the actor $\pi_\theta(s)$ and the critic $Q(s, u, \tau|\omega)$.
Make target networks $\pi_{\theta'}(s)$ and $Q(s, u, \tau|\omega')$ by cloning the actor and the critic respectively.
for episode $= 0$ to M **do**
  Initialize $s_0 \in d_0$.
  Set $i = 0, t_i = 0$.
  while $t_i \leq T$ **do**
    Select $[u_i, \tau_i] = \pi_\theta(s_i) + e_i$.
    Execute action u_i for τ_i second to the environment.
    Receive r_i and observe s_{i+1}.
    Store $(s_i, u_i, \tau_i, r_i, s_{i+1}, t_i)$ to the replay buffer.
    Make mini-batch E considering probability $e^{-\alpha t_i}$.
    Update the critic ω to decrease

$$L = \sum_{(s,u,\tau)\in E} Q(s, u, \tau|\omega) - \{r(s, u, \tau) + e^{-\alpha\tau} Q(s', \pi_{\theta'}(s')|\omega')\}.$$

    Calculate approximated policy gradient using (33).
    Update the actor with approximated policy gradient.
    Update target networks as (36).
  end while
end for

5.1 Evaluation Criteria

In this section, we use the valuation function $J(\pi)$ as a criterion to evaluate the policy π. $J(\pi)$ is the expectation of the value function $V^{\pi}(s)$ with respect to the initial state distribution s_0. In this paper, we assume that the initial state distribution d_0 is a uniform distribution on the initial state space S in both linear and nonlinear cases. Then, the space S is discretized into a grid, and the value function $V^{\pi}(s)$ for each state s on the grid is calculated by simulation and averaged to approximate the valuation function $J(\pi)$. In order to take into account the effect of system noise, $V^{\pi}(s)$ is the average of the long-time costs of several simulations for each state s.

5.2 Linear System

First, we adopt reinforcement learning to self-triggered control for linear system. The system to be controlled is

$$\dot{s} = As + Bu + D\dot{w} = \begin{bmatrix} -1 & 4 \\ 2 & -3 \end{bmatrix} s + \begin{bmatrix} 2 \\ 4 \end{bmatrix} u + \begin{bmatrix} 0.6 \\ 0.3 \end{bmatrix} \dot{w} \tag{38}$$

where $\dot{w}$ is wiener process noise. Here, the input signal u is limited to $-10 \sim 10$. Let the initial state space be $S = \{s = [s_0, s_1]^\top \in \mathbb{R}^2 | s_0 \in [-7, 7], s_1 \in [-7, 7]\}$.

5.2.1 Initial Policy

For the comparison with the control performance with that of a naively designed model-based self-triggered control law, we use $\pi_{\mathrm{MB}}(s)$ such that

$$\pi_{\mathrm{MB}}(s) = \underset{u,\tau}{\operatorname{argmin}} \left\{ u^2 - \lambda\tau + {s'_e}^\top P s'_e + \mathrm{Tr}(P\Sigma) \right\} \tag{39}$$

where $s'_e = \mathbb{E}_w[s'(s, u, \tau)]$, $\Sigma = Var_w[s'(s, u, \tau)]$ are expectation and variance of next state respectively, and P is a unique solution of continuous time algebraic riccati equation

$$A^\top P + PA - PBR^{-1}B^\top P + Q = \mathbf{0} \tag{40}$$

where, Q and R are hyper parameter matrix. The last two terms of objective function of (39) denotes the optimal cost when the continuous control is performed with LQR controller from the next state s'_e. By incorporating this cost into the objective, we can find u and τ such that the transition to the state which requires high control cost in the next step is avoided.

In order to use π_{MB} as an initial policy for reinforcement learning, we represent π_{MB} as a neural network approximated by supervised learning of $\pi_{\mathrm{MB}}(s)$ for M randomly generated states s in the state space S. The evaluation value of supervised π_{MB} for several λ are shown in Table 1. We refer to the best policy in Table 1 ($\lambda = 1.0$) as $\hat{\pi}_{\mathrm{MB}}$ and use it as the initial policy for proposed method. Figure 3 shows the control path with initial policy $\hat{\pi}_{\mathrm{MB}}$ stating from $s_0 = [3., 3.]$.

λ	$J(\pi)$
0.01	22.14
0.1	80.02
0.5	19.61
1.0	11.32
5.0	12.23
10	15.72
50	32.21
100	37.38

Table 1: Evaluation value for several λ

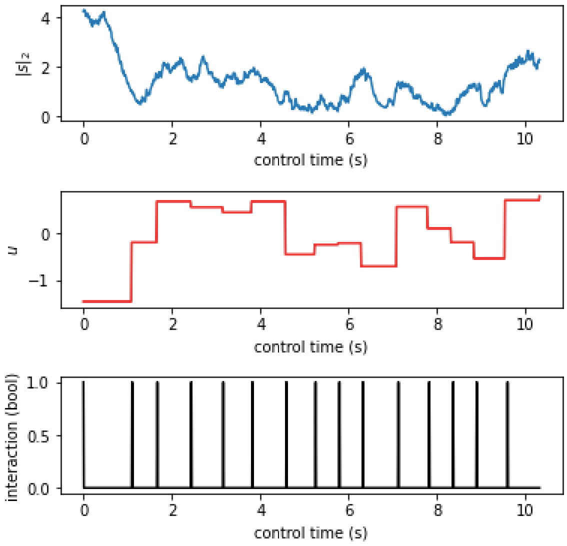

Figure 3: A control path with learned policy $\hat{\pi}_{\text{MB}}$

In Figure 3, the norm of state s, the control signal u and the boolean which denotes whether agent interact with environment at time t second are shown from top to bottom.

5.2.2 Result of Proposed Method

First, we consider the results of reinforcement learning using the proposed method 1 with $\hat{\pi}_{\text{MB}}$ as the initial policy. Figure 4 shows the path controlled by the policy π^{L}_{prop} from the initial state $s_0 = \begin{bmatrix} 3 & 3 \end{bmatrix}$.

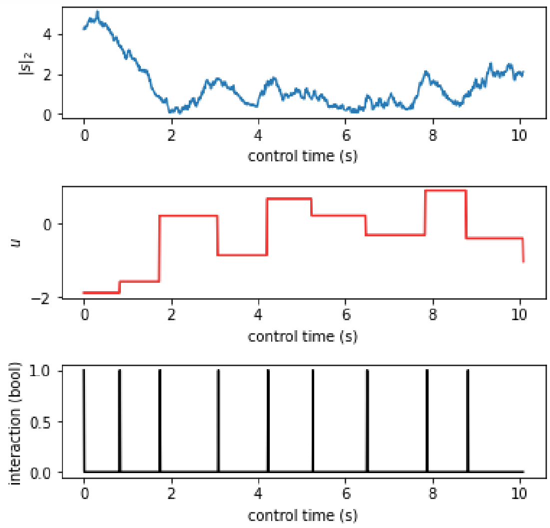

Figure 4: A control path with learned policy π^L_{prop}

The evaluation value of this policy π^L_{prop} is

$$J(\pi^L_{\mathrm{prop}}) \simeq 6.82 \tag{41}$$

Thus, we can see the improvement of the policy from $\hat{\pi}_{\mathrm{MB}}$.

The history of the value of the evaluation function $J(\pi_\theta)$ as the policy parameter θ is updated is shown in Figure 5.

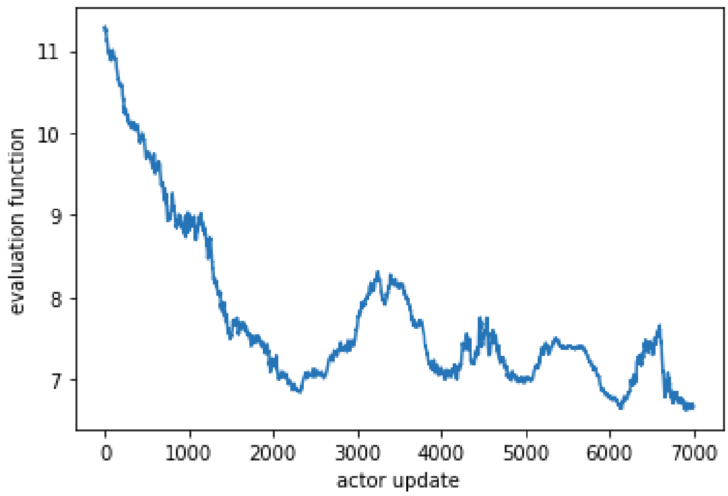

Figure 5: Policy improvement on linear case

Figure 5 shows an example of successful learning. However, since the calculation of the policy gradient depends on the approximation accuracy of the critic according to the equation (33), we often observed a sharp deterioration of the policy using the proposed method. Therefore, the learning accuracy of critic is a future work.

5.3 Non-linear Case

In this subsection, we investigate whether the self-triggered control law can be learned by reinforcement learning even when the control target is extended to non-linear systems, especially control affine systems. We consider an inverted pendulum, whose state-space representation is

$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\dot{\theta} \\ \frac{3g}{2l}\sin\theta + \frac{3}{ml^2}u\end{bmatrix} + \begin{bmatrix}1 \\ 1\end{bmatrix}\dot{w}. \tag{42}$$

where $\dot{w}$ is wiener process noise. Therefore, for an inverted pendulum, the state variable s is considered to be $\begin{pmatrix}\theta & \dot{\theta}\end{pmatrix}^\top$.

As in the linear case, the input signal u is limited to $-10 \sim 10$. And let the initial state space be $S = \{[\theta, \dot{\theta}]^\top \in \mathbb{R}^2 | \theta \in [-\pi, \pi], \dot{\theta} \in [-2\pi, 2\pi]\}$.

5.3.1 Initial Policy

The initial policy π_{init} used in this case is

$$\pi_{\text{init}}(s) = \begin{bmatrix}-Ks & 0.2\end{bmatrix} \tag{43}$$

where K is a feedback gain calculated by Linear Quadratic Regulator for linearized system around $s = \mathbf{0}$. Figure 6 shows the control path with initial policy π_{init} stating from $s_0 = [3., 3.]$.

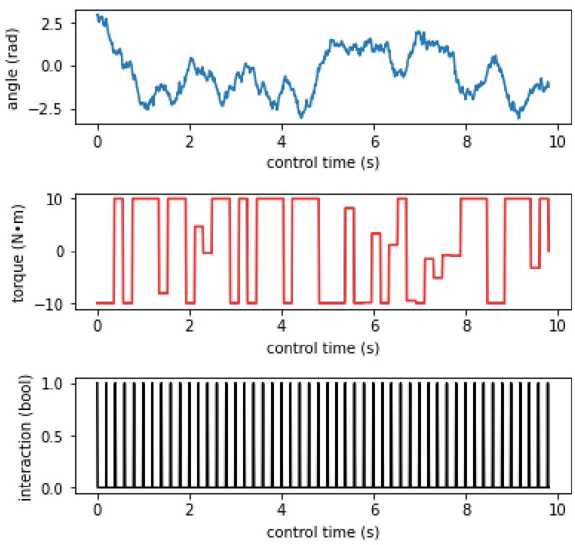

Figure 6: A control path with initial policy π_{init}

The evaluation value of π_{init} is

$$J(\pi_{\text{init}}) \simeq 62.49 \tag{44}$$

In Figure 6, the angle of pendulum θ rad, the torque u N·m and the boolean which denotes whether agent interact with environment at time t second are shown from top to bottom.

5.3.2 Result of Proposed Method

First, we show the results of reinforcement learning by the proposed method 1. Figure 7 shows the control path by the obtained policy π^N_{prop} stating from $s_0 = [3., 3.]$.

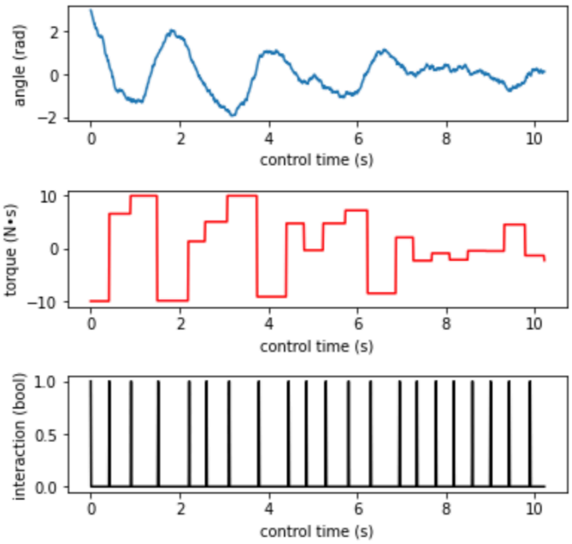

Figure 7: A control path with learned policy π^N_{prop}

The evaluation value of π^N_{prop} is

$$J(\pi^N_{\text{prop}}) \simeq 30.57 \tag{45}$$

Thus, we can confirm the improvement of the policy.

The change of the value of the evaluation function $J(\pi_\theta)$ as the policy parameter θ is updated is shown in Figure 8.

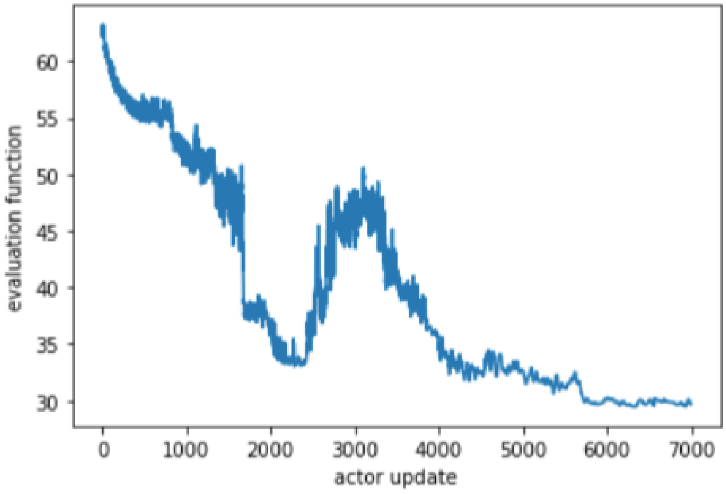

Figure 8: Policy improvement on non-linear case

6 Conclusion

In this paper, we formulated an optimal self-triggered control problem where the communication cost was explicitly included, which had not been considered in previous studies. Then, we considered a reinforcement learning approach to the problem.

First, from the configuration of the evaluation function, we confirmed that the deterministic policy gradient theorem for general reinforcement learning was not directly applicable, and then we gave a policy gradient theorem that was compatible with the formulated optimal self-triggered control problem.

In this paper, we also proposed a reinforcement learning algorithm for approximate computation of the policy gradient. As a result of the implementation, for the linear system, we can improve the policy in the sense of the formulated evaluation function for the control law designedheuristically based on the model. We also succeeded in improving the policy for self-triggered control of nonlinear systems, which was not solved in the previous study.

However, the computational complexity and the way of saving the empirical data are important issues to be solved in the future, because they greatly affect the results of the calculation of the policy gradient.

Acknowledgments

The author would like to express his sincere gratitude to Professor Yoshito Ohta, Associate Professor Kenji Kashima and Assistant Professor Kentaro Ohki for their helpful advices. He would also like to thank his colleagues in the control systems theory field laboratory for discussing the issue with him.

References

[1] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada. “ An introduction to event-triggered and self-triggered control. ” *In Proc. of the 51st IEEE International Conference on Decision and Control*, 2012.

[2] D. Baumann, J. J. Zhu, G. Martius, and S. Trimpe. “Deep Reinforcement Learning for Event-Triggered Control. ” *In Proc. of the 57th IEEE International Conference on Decision and Control*, 2018.

[3] T. Gommans, D. Antunes, T. Donkers, P. Tabuada, and M. Heemels. “Self-triggered linear quadratic control. ” *Automatica*, vol. 50, no. 4, pp. 1279-1287, 2014.

[4] G. Yang, C. Belta, and R. Tron. “ Self-triggered Control for Safety Critical Systems Using Control Barrier Functions.” *In American Control Conference (ACC) Philadelphia, USA*, 2019.

[5] R. S. Sutton and A. G. Barto. “ Reinforcement Learning: An introduction” MIT Press. 1998.

[6] C. J. Watkins, and P. Dayan. Q-learning. *Machine Learning*, vol. 8, no. 3-4, pp. 279-292, 1992.

[7] V. Minh, K. Kavukcouglu, D. Silver. et al.. “ Human-level control through deep reinforcement learning.” *Nature 518*, pp.529-533, 2015.

[8] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, et al.. “ Deterministic Policy Gradient Algorithms.” *ICML Beijing, China.*, 2014, Beijing.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N.Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2015.

[10] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska. “ A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradient.” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291-1307, 2012.

[11] T. Degris, M. White and R. Sutton. “ Off-Policy Actor-Critic.” *ICML Edinburgh, United Kingdom*, 2012.

[12] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. ”Policy gradient methods for reinforcement learning with function approximation.” *In Advances in Neural Information Processing Systems*, 2000.

[13] D. P. Kingma and J. Ba. “ Adam: A Method for Stochastic Optimization.” *arXiv preprint arXiv: 1412.6980*, 2014.

A Appendix

A.1 Model Settings

We use DDPG as reinforcement learning algorithm. As described in section 2, actor and critic is expressed as neural networks respectively. Experiments have shown that learning diverges when using a general network, so here we use the special network. The architecture of 2 networks is in Fig 9.

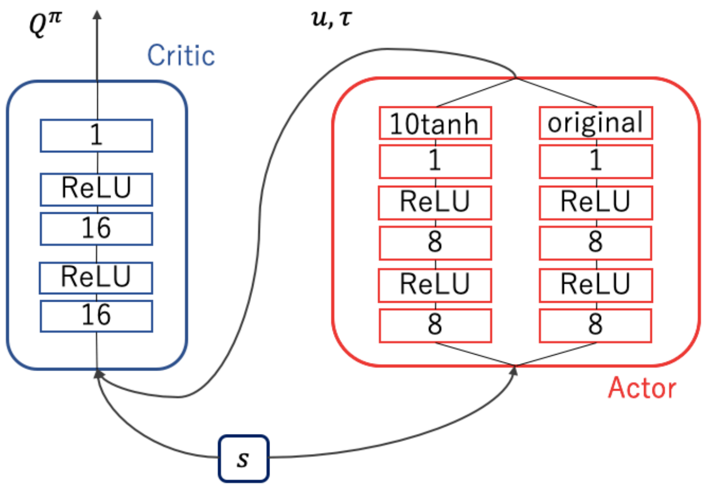

Figure 9: Agent Model

The activation function "original" shown in Fig 9 is defined as $0.99 \times \text{sigmoid} + 0.01$ to meet upper and lower limits of interval described in the next section.

Master's Thesis

Deep Reinforcement Learning for Self-Triggered Control

Guidance

Associate Professor Kenji KASHIMA

Ibuki TAKEUCHI

Department of Applied Mathematics and Physics

Graduate School of Informatics

Kyoto University

February 2021

Deep Reinforcement Learning for Self-Triggered Control

Ibuki TAKEUCHI

February 2021

Deep Reinforcement Learning for Self-Triggered Control

Ibuki TAKEUCHI

Abstract

One of the control methods for continuous-time systems is the sampled-data control. This is a control method in which the system state is observed and new control inputs are communicated at periodic intervals. The disadvantage of the sampled-data control is that it requires communication at every interval even when the control performance can be maintained without updating the control inputs, which results in extra cost for communication.

In recent years, event-triggered control and self-triggered control have attracted much attention as control methods for efficient communication and control input design.

In this paper, self-triggered control is investigated. In the self-triggered control, unlike the sampled-data control and the event-triggered control, the periodic state observation is not performed. Instead, the controller itself decides the next trigger time and communicates the state observation and control input for the following time duration. For the self-triggered control, several model-based design methods have been proposed, but these methods do not explicitly consider the communication cost over a long time of control.

In this paper, we formulate an optimal self-triggered control problem where communication cost is explicitly included, which has not been considered in previous studies. To solve this problem, we consider a policy gradient method to the problem formulated in this paper. We also propose a reinforcement learning algorithm for approximate computation of the policy gradient.