Master's Thesis

Deep Reinforcement Learning for Self-Triggered Control

Guidance

Professor	Yoshito OHTA
Assistant Professor	Kenji KASHIMA

Ibuki TAKEUCHI

Department of Applied Mathematics and Physics

Graduate School of Informatics

Kyoto University

February 2021

Abstract

In this thesis, we construct the mathematical models of students who write the master's thesis and develop efficient algorithms for writing the master's thesis based on the models. We show that the proposed algorithms generate the thesis 65536 times more efficiently than writing by oneself.

Contents

1 Introduction

In many control applications, controllers are nowadays implemented using communication networks in which the control task has to share the communication resources with other tasks. Despite the fact that resources can be scarce, controllers are typically still implemented in a time-triggered fashion, in which control tasks are executed periodically. This design choice often leads to over-utilization of the available communication resources, and/or causes a limited lifetime of battery-powered devices, as it might not be necessary to execute the control task every period to guarantee the desired closed-loop performance.

Also in the area of "sparse control" (Gallieri & Maciejowski, 2012), in which it is desirable to limit the changes in certain actuator signals while still realizing specific control objectives, periodic execution of control tasks may not be optimal either. In both networked control systems with scarce communication resources and sparse control applications arises the fundamental problem of determining optimal sampling and communication strategies, where optimality needs to reflect both implementation cost (related to the number of communications and/or actuator changes) as well as control performance. It is expected that the solution to this problem results in control strategies that abandon the periodic time-triggered control paradigm.

Two approaches that abandon the periodic communication pattern are event-triggered control (ETC), see, e.g., Arzen (1999), Astrom and Bernhardsson (1999) and Donkers and Heemels (2012), Heemels, Sandee, and van den Bosch (2008), Heemels et al. (1999), Henningsson, Johannesson, and Cervin (2008), Lunze and Lehmann (2010), Tabuada (2007) and Wang and Lemmon (2009), and self-triggered control (STC), see, e.g., Almeida, Silvestre, and Pascoal (2010, 2011), Anta and Tabuada (2010), Donkers, Tabuada, and Heemels (2012), Mazo, Anta, and Tabuada (2010), Velasco, Fuertes, and Marti (2003) and Wang and Lemmon (2009). Although ETC is effective in the reduction of communication or actuator movements, it was originally proposed for different reasons, including the reduction of the use of computational resources and dealing with the event-based nature of the plants to be controlled. In ETC and STC, the control law consists of two elements being a feedback controller that computes the control input, and a triggering mechanism that determines when the control input has to be updated. The difference between ETC and STC is that in the former the triggering consists of verifying a specific condition continuously and when it becomes true, the control task is triggered, while in the latter at an update time the next update time is pre-computed.

At present ETC and STC form popular research areas. However, two important issues have only received marginal attention: (i) the co-design of both the feedback law and the triggering mechanism, and (ii) the provision of performance guarantees (by design). To elaborate on (i), note that current design methods for ETC and STC are mostly emulation-based approaches, by which we mean that the feedback controller is designed without considering the scarcity in the system ' s resources. The triggering mechanism is only designed in a subsequent phase, where the controller has already been fixed. Since the feedback controller is designed before the triggering mechanism, it is difficult, if not impossible, to obtain an optimal design of the combined feedback controller and triggering mechanism in the sense that the minimum number of control executions is achieved while guaranteeing closed-loop stability and a desired level of performance.

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 maximize 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 obtain a state feedback control law that outperforms the control performance of a naively designed simulation-based self-triggered control law.
- 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 $M = \{S, A, T, d_0, r, \gamma\}$. Here, S, A denotes state, action set, and $T(s^{'}|s,a)$ express transition probability. Also, $d_0, r(s,a), \gamma \in [0,1]$ are distribution of initial state, reward, discount factor respectively.

Now, the purpose of reinforcement learning is to find a policy such that

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

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

$$V^\pi(s) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|_{a_t=\pi(s_t)}, s_0 = s \tag{2}$$

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

Here, we define Q function, which is useful tool for analyzing reinforcement learning. Q function is given as

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

As showed in (4), Q function express the value when agent select action a freely and choose action according to the policy π from 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 in 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).

2.3 Algorithms adapted to the settings of state and action space

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.

Now, what about the case where the state space is continuous? Since the state s takes a continuous value, it cannot be stored in a table. Therefore, Minh et al.[2] 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 $\underset{a}{\operatorname{argmax}} Q^\pi(s, a)$.

Finally, 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.

2.4 Deterministic Policy Gradient Method

Silver et al.[3] 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.

Theorem 1 (Deterministic Policy Gradient Theorem)**.** *The gradient for evaluation function $\nabla_\theta J(\pi_\theta)$ is exist and calculated 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.

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

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

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) 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 measure π, respectively. The update algorithm of actor and critic is described below.

DDPG uses mini-batch learning. First, update idea of critic is shown. The purpose of critic is to approximate Q^π. Because Q function is able to be decomposed like (4),

$Q(s,a|\omega)$ should also be updated to satisfy this relation. For that, parameter ω is updated to the direction where it minimize Temporal Difference(TD) error:

$$\mathrm{TD} = Q(s,a|\omega) - \{r(s,a) + \gamma Q(s,\pi(s)|\omega)\} \tag{10}$$

Since it is difficult to optimize for whole (s,a) at once, DDPG uses the mean squared error for the mini-batch E as the loss function and reduces it.

$$Loss = \frac{1}{N}\sum_{s\in E}\mathrm{TD}^2 \tag{11}$$

Now, the above method of updating critic is supervised learning itself. Therefore, the data in the mini-batch must be i.i.d.. If the mini-batch E uses the data of the last N steps experienced by the agent, they are no longer independent. Hence, the agent stores the empirical data in a storage, called replay buffer, and randomly selects N data from it to make a mini-batch to increase the variance of it.

Next update law of actor are shown. Because 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, approximated policy gradient

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

is used. Furthurmore, an approximation to the expectation is made like following:

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

Therefore, the accuracy of the approximation of the policy gradient may be greatly degraded by the accuracy of critic approximation and the distribution of mini-batch.

2.5 Exploration in DDPG

Reinforcement learning uses empirical data collected by interaction with the environment to improve a policy. An agent needs to take a variety of actions in each state in order to know better actions. This is called "exploration". When the action space is discrete, a exploration method called "ε - greedy" is widely known.

In the case of continuous action space, it is common to collect data with the action determined by adding noise e to the output of the policy function during training, such as

$$a = \pi_\theta(s) + e. \tag{14}$$

(The noise-added policy is called the behavior policy in order to distinguish it from the learning policy.) Intuitively, the larger this noise is, the wider the exploration is expected to be. By increasing the distribution of behavioral data around $a = \pi_\theta(s)$, we can expect to improve the accuracy of approximating the gradient of the Q-function with respect to a.

Then, is it useful to enlarge the exploration noise in all cases? In DDPG, we stored the past empirical data in the replay buffer, and calculated the policy gradient using the data set selected with equal probability from them. Since the expected value of the policy gradient is taken for $s \sim \rho^{\pi_\theta}$, the state distribution of the replay buffer should be as close as possible to ρ^{π_θ}. Adding a large exploration noise may violate this property in some cases. Therefore, it may be disadvantageous to enlarge the exploration noise. This is considered to be a kind of problem called the "exploration and exploitation dilemma".

3 Problem Formulation

3.1 Self-Triggered Control

We consider control system like Fig 1.

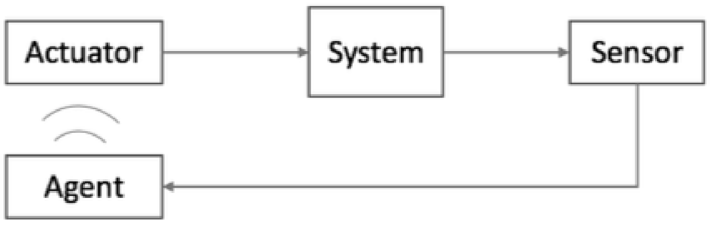

Figure 1: control system

Here, the control target is a continuous-time system as Equation (15)

$$\dot{s} = h(s, u) + \dot{w} \tag{15}$$

Now, we consider a feedback control for system (15). 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 seconds 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 $\tau(sec)$. The input u sent in the previous interaction is added until the time of the next interaction (ZOH control).

3.2 Optimal Self-Triggered Control

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 $J(\pi) = \mathbb{E}_{s \sim d_0}[V^\pi(s)]$, for the initial state $s \sim d_0$.

$$V^\pi(s) = \int_0^\infty e^{-\alpha t}\mathbb{E}_w[s(t)^\top Qs(t) + u(t)^\top Ru(t) + \beta C(t)|s(0) = s]\mathrm{d}t \tag{16}$$

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

$$\begin{aligned} V^\pi(s) &= \sum_{i=0}^\infty \int_{t_i}^{t_{i+1}} e^{-\alpha t}\mathbb{E}_w[s(t)^\top Qs(t) + u_i^\top Ru_i + \beta C(t)|s(0) = s]\mathrm{d}t \\ &= \sum_{i=0}^\infty e^{-\alpha t_i}(\int_0^{\tau_i} \mathbb{E}_w[s(t)^\top Qs(t) + u_i^\top Ru_i|s(0) = s_i]\mathrm{d}t + \beta) \\ &= \sum_{i=0}^\infty e^{-\alpha t_i} r(s_i, \pi(s_i)). \end{aligned} \tag{17}$$

Here, t_i is the time of the ith communication and s_i is the state at that time. Also, let $[u_i, \tau_i] = \pi(s_i)$. Therefore, $V^\pi(s)$ satisfies the following Bellman equation.

$$V^\pi(s) = r(s, \pi_\theta(s)) + e^{-\alpha\tau}\mathbb{E}_{s'}[V^\pi(s'(s, \pi_\theta(s)))] \tag{18}$$

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

$$Q^\pi(s, u, \tau) = r(s, u, \tau) + e^{-\alpha\tau}\mathbb{E}_{s'}[Q^\pi(s'(s, u, \tau), \pi(s'(s, u, \tau)))] \tag{19}$$

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 consider the reinforcement learning problem by taking the communication as one step. Furthermore, DDPG may also be applied by approximating the Q-function, which satisfies the equation (19) 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 (17) depends on τ at each step, it differs from the general reinforcement learning problem. In this subsection, we discuss how the DDPG is affected by this difference.

Actually, due to the property of Q-functions such as (19), DPG cannot be computed as in (5). Since

$$\begin{aligned}
\nabla_\theta V^{\pi_\theta}(s) &= \nabla_\theta V^{\pi_\theta}(s, \pi_\theta(s)) \\
&= \nabla_\theta [r(s, \pi_\theta(s)) + e^{-\alpha\tau(s)} \mathbb{E}_{s'}[V^{\pi_\theta}(s')] \\
&= \nabla_\theta \pi_\theta(s) \nabla_a r(s,a)|_{a=\pi_\theta(s)} \\
&\quad + e^{-\alpha\tau(s)} \int_S \{\nabla_\theta \pi_\theta(s) \nabla_a Pr(s \to s', 1, a)|_{a=\pi(s)} V^{\pi_\theta(s')} \\
&\qquad\qquad + Pr(s \to s', 1, \pi_\theta) \nabla_\theta V^{\pi_\theta}(s')\} ds' \\
&\quad + \int_S \nabla_\theta e^{-\alpha\tau(s)} Pr(s \to s', 1, \pi_\theta) V^{\pi_\theta}(s') ds',
\end{aligned} \tag{20}$$

we have

$$\begin{aligned}
&\nabla_\theta V^{\pi_\theta}(s) \\
&= \sum_{i=0}^{\infty} \int_S \cdots \int_S Pr(s_0 \to s_1, 1, \pi_\theta) Pr(s_{i-1} \to s_i, 1, \pi_\theta) \\
&\qquad e^{-\alpha \sum_{k=0}^{i-1} \tau(s_k)} \nabla_\theta \pi_\theta(s_i) \nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s_i)} ds_i ds_{i-1} \dots ds_1 \\
&\quad + \sum_{i=1}^{\infty} \int_S \cdots \int_S Pr(s_0 \to s_1, 1, \pi_\theta) Pr(s_{i-1} \to s_i, 1, \pi_\theta) \\
&\qquad e^{-\alpha \sum_{k=1}^{i-1} \tau(s_{k-1})} \nabla_\theta e^{-\alpha\tau(s_{i-1})} V^{\pi_\theta}(s_i) ds_i ds_{i-1} \dots ds_1.
\end{aligned} \tag{21}$$

Thus, we can conclude that DPG for self-triggered control can be written as

$$\begin{aligned}
&\nabla_\theta J(\pi_\theta) \\
&= \mathbb{E}_{s_0 \sim d_0}[\nabla_\theta V^{\pi_\theta}(s_0)] \\
&= \sum_{i=0}^{\infty} \int_S \cdots \int_S d_0(s_0) Pr(s_0 \to s_1, 1, \pi_\theta) \cdots Pr(s_{i-1} \to s_i, 1, \pi_\theta) \\
&\qquad e^{-\alpha \sum_{k=0}^{i-1} \tau(s_k)} \nabla_\theta \pi_\theta(s_i) \nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s_i)} ds_i ds_{i-1} \dots ds_0 \\
&\quad + \sum_{i=1}^{\infty} \int_S \cdots \int_S d_0(s_0) Pr(s_0 \to s_1, 1, \pi_\theta) \cdots Pr(s_{i-1} \to s_i, 1, \pi_\theta) \\
&\qquad e^{-\alpha \sum_{k=1}^{i-1} \tau(s_{k-1})} \nabla_\theta e^{-\alpha\tau(s_{i-1})} V^{\pi_\theta}(s_i) ds_i ds_{i-1} \dots ds_0.
\end{aligned} \tag{22}$$

Here, we give the ideal algorithm for reinforcement learning for self-triggered control.

Algorithm 1 Ideal algorithm for Self-Triggered Control RL

```
Write me.
```

following two issues are raised.

- As in the case of fixed discount rates, it is not possible to summarize the multiple integrals
- A term about $\nabla_\theta e^{-\alpha\tau(s_{i-1})}$ appears

In the next section, we propose a method to address both of these issues.

4.2 Proposed Method

First, in order to summarize the multiple integrals, we want to approximate $e^{-\alpha\sum_{j=0}^{i-1}\tau(s_k)}$ independent of s_k. Thus, we propose to fix the discount rate as

$$\gamma_\theta = \mathbb{E}_{s\sim E}[e^{-\alpha\tau(s)}] \tag{23}$$

where E denotes the replay buffer. With this approximation, the multiple integrals can be summarized as

$$\begin{aligned}\sum_{i=0}^{\infty}\int_S\cdots\int_S & Pr(s_0\to s_1,1,\pi_\theta)Pr(s_{i-1}\to s_i,1,\pi_\theta)\\ & e^{-\alpha\sum_{j=0}^{i-1}\tau(s_k)}\nabla_\theta\pi_\theta(s_i)\nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s_i)}ds_i ds_{i-1}\dots ds_0\\ \simeq \sum_{i=0}^{\infty}\int_S & Pr(s\to s',i,\pi_\theta)\gamma_\theta^i\nabla_\theta\pi_\theta(s')\nabla_a Q^{\pi_\theta}(s',a')|_{a'=\pi_\theta(s')}ds'\end{aligned} \tag{24}$$

like in theorem 1. Also, with γ_θ, the term about $\nabla_\theta e^{-\alpha\tau(s_{i-1})}$ can be approximated as

$$\begin{aligned}\sum_{i=1}^{\infty}\int_S\cdots\int_S & Pr(s_0\to s_1,1,\pi_\theta)Pr(s_{i-1}\to s_i,1,\pi_\theta)\\ & e^{-\alpha\sum_{j=1}^{i-1}\tau(s_{k-1})}\nabla_\theta e^{-\alpha\tau(s_{i-1})}V^{\pi_\theta}(s_i)ds_i ds_{i-1}\dots ds_1\\ \simeq \frac{1}{\gamma_\theta}\sum_{i=1}^{\infty}\int_S & Pr(s\to s',i,\pi_\theta)\gamma_\theta^{i-1}\nabla_\theta\gamma_\theta V^{\pi_\theta}(s')ds'.\end{aligned} \tag{25}$$

Using these approximations, the DPG for optimal self-triggered control can be regarded as

$$\nabla_\theta J(\pi_\theta) \simeq \mathbb{E}_{s\sim\hat{\rho}^{\pi_\theta}}[\nabla_\theta\pi_\theta(s)\nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)} + \frac{1}{\gamma_\theta}V^{\pi_\theta}(s)\nabla_\theta\gamma_\theta] - \mathbb{E}_{s\sim d_0}[\frac{1}{\gamma_\theta}V^{\pi_\theta}(s)\nabla_\theta\gamma_\theta] \tag{26}$$

where,

$$\hat{\rho}^{\pi_\theta}(s) = \int_S\sum_{i=0}^{\infty}\gamma_\theta^i d_0(s)Pr(s_0\to s,i,\pi_\theta)ds_0 \tag{27}$$

The DDPG algorithm using proposed method is shown below.

Algorithm 2 DDPG for Self-Triggered Control

```
  Write me.
```

4.3 Model Settings

In the 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 $\begin{bmatrix} u & \tau \end{bmatrix}^\top$.

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

$$\pi(s) = \begin{bmatrix} u(s) & \tau(s) \end{bmatrix}^\top \tag{28}$$

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 2.

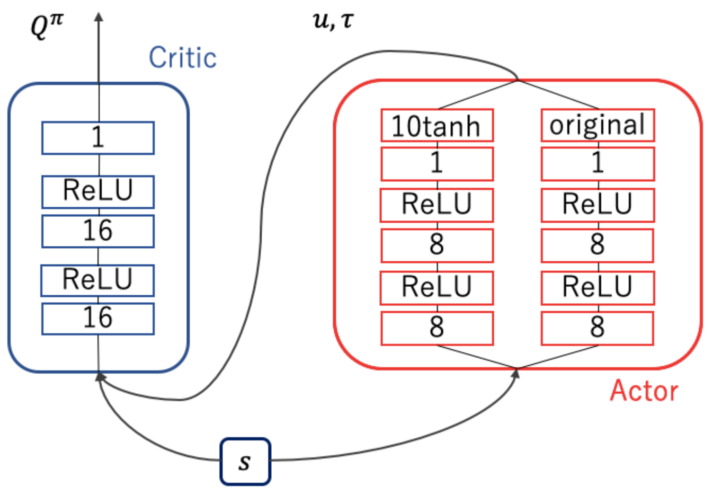

Figure 2: Agent Model

The activation function ”original” shown in Fig 2 is defined as $0.99 \times \text{sigmoid} + 0.01$ to meet upper and lower limits of interval described in the next section.

5 Consideration

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 $0.01(s) \sim 1.0(s)$.

5.1 Linear Case

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

$$\dot{s} = As + Bu = \begin{bmatrix} -1 & 4 \\ 2 & -3 \end{bmatrix} s + \begin{bmatrix} 2 \\ 4 \end{bmatrix} u. \tag{29}$$

Here, the input signal u is limited to $-10 \sim 10$.

5.1.1 Initial Policy

For learning efficiency, we use π^l_{init} such that

$$\pi^l_{\text{init}} = \begin{bmatrix} -Ks & 0.01 \end{bmatrix} \tag{30}$$

where K is a feedback gain calculated by Linear Quadratic Regulator. This policy stabilize the system.

5.1.2 Learned Policy

Fig 3 shows a control path with learned policyπ^l_{RL}, stating from $s_0 = [3., 3.]$.

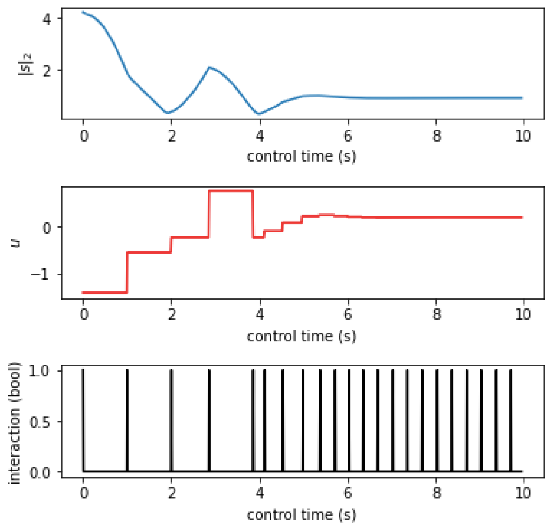

Figure 3: A control path with learned policy π^l_{RL}

Figure 3 shows, from top to bottom, the L2 - norm of state s, the control input u, and the boolean representing interaction. From this figure, we can see that the communication interval is determined flexibly at each interaction.

Now, we confirm whether the learned policy π^l_{RL} enlarge the evaluation function $J(\pi)$ compared from the initital policy π^l_{init}. Since $J(\pi) = \mathbb{E}_{s_0}[V^\pi(s_0)]$, we generate 500 initial states s_0 according to the initial state distribution d_0, and average the results of $V^{\pi}(s_0)$. We conduct this approximation of expectation 1000 times. From the result

$$J(\pi^l_{\mathrm{init}}) =, \ J(\pi^l_{\mathrm{RL}}) =, \tag{31}$$

we can confirm $J(\pi^l_{\mathrm{init}}) < J(\pi^l_{\mathrm{RL}})$. It can be concluded that the policy has been improved.

5.1.3 Comparison with Naiive Model Based Control

In this section, we compare the control performance with that of a naively designed model-based self-triggered control law π^l_{MB}. The control law π^l_{MB} is defined as the solution of

$$\pi^l_{\mathrm{MB}}(s) = \operatorname*{argmin}_{u,\tau} \left\{ \int_0^{\tau} s(t)^{\top} s(t)\mathrm{d}t + u^2 - \lambda\tau + V_{\mathrm{LQR}}(s'(s,u,\tau)) \right\} \tag{32}$$

where $s(0) = s$. The control performance of π^l_{MB} is

$$\textit{write me} \tag{33}$$

From the result, we can conclude that the control law obtained by our method outperforms the control law designed by a naive model-based design.

5.2 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

$$\frac{\mathrm{d}}{\mathrm{dt}} \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ \frac{3g}{2l}\sin\theta + \frac{3}{ml^2}a \end{pmatrix}. \tag{34}$$

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 $-10 \sim 10$.

5.2.1 Initial Policy

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

$$\pi_{\mathrm{init}} = \begin{bmatrix} -Ks & 0.2 \end{bmatrix} \tag{35}$$

where K is a feedback gain calculated by Linear Quadratic Regulator. This policy stabilize the system.

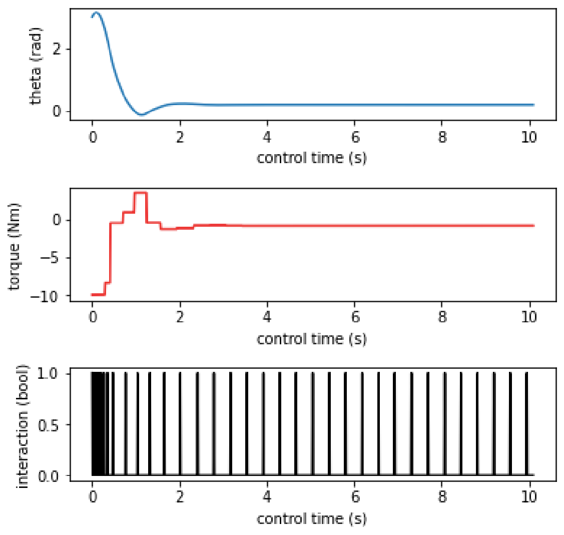

Figure 4: A control path with learned policy π_{RL}

5.2.2 Learned Policy

Fig 4 shows a control path with learned policy, stating from $s_0 = [3., 3.]$.
Figure 4 shows, from top to bottom, the angle of the pendulum θ, the control input u, and the boolean representing interaction. Similarly to the linear case, Figure 4 shows that the agent decides the communication interval flexibly.

Now let's compare $J(\pi_{\mathrm{RL}})$ and $J(\pi_{\mathrm{init}})$. As in linear case, we approximate $J(\pi) = \mathbb{E}_{s_0}[V^{\pi}(s_0)]$ in the same way. The result is

$$J(\pi_{\mathrm{init}}) = 4.23 \pm 0.083,\ \ J(\pi_{\mathrm{RL}}) = 37.12 \pm 0.071. \tag{36}$$

Since $J(\pi_{\mathrm{init}}) < J(\pi_{\mathrm{RL}})$, we can conclude that the policy has been improved in the non-linear case as well.

5.3 Optimality

Is the policy π_{RL} described in the last section an optimal? If it is an optimal solution, the policy gradient must be $\mathbf{0}$. Therefore from equation (5), a condition

$$\mathbb{E}_{s\sim\rho^{\pi_{\mathrm{RL}}}}[\nabla_{\theta}\pi_{\mathrm{RL}}(s)\nabla_a Q^{\pi_{\mathrm{RL}}}(s,a)|_{a=\pi_{\mathrm{RL}}(s)}] = \mathbf{0} \tag{37}$$

should be met.

In order to check whether the policy π_{RL} satisfies the condition (37), we want to know $Q^{\pi_{\mathrm{RL}}}(s,a)$. Then, we obtain the approximation of value function $V^{\pi_{\mathrm{RL}}}(s) = \sum_{i=0}^{N} \gamma^i r(s_i, \pi_{\mathrm{RL}}(s_i))$ with N step simulation:

$$V_{\mathrm{ap}}^{\pi_{\mathrm{RL}}}(s) = \sum_{i=0}^{N} \gamma^i r(s_i, \pi_{\mathrm{RL}}(s_i)). \tag{38}$$

Next, the approximation of $Q^{\pi_{\mathrm{RL}}}$ is calculated with $V_{\mathrm{ap}}^{\pi_{\mathrm{RL}}}$ as following:

$$Q_{\mathrm{apd}}^{\pi_{\mathrm{RL}}}(s,a) = r(s,a) + \gamma V_{\mathrm{ap}}^{\pi_{\mathrm{RL}}}.(s') \tag{39}$$

Since $\gamma < 1$, $Q_{\mathrm{apd}}^{\pi_{\mathrm{RL}}}(s,a)$ is a good approximation if the number of simulation steps N is sufficiently large. However, to compute the policy gradient, we need to compute $\nabla_a Q^{\pi_{\mathrm{RL}}}(s,a)$, while $Q_{\mathrm{apd}}^{\pi_{\mathrm{RL}}}$ can only be computed the value at (s,a). Then, we approximate the function $Q_{\mathrm{apd}}^{\pi_{\mathrm{RL}}}(s,a)$ by supervised learning with $Q_{\mathrm{apd}}^{\pi\{textrmRL}}(s,a)$ as the teacher data. This function is defined as $Q_{\mathrm{ap}}^{\pi_{\mathrm{RL}}}$.

Now, when we calculate the policy gradient using $Q_{\mathrm{ap}}^{\pi_{\mathrm{RL}}}$, we obtain

$$\mathbb{E}_{s\sim\rho^{\pi_{\mathrm{RL}}}}[\nabla_\theta \pi_{\mathrm{RL}}(s)\nabla_a Q_{\mathrm{ap}}^{\pi_{\mathrm{RL}}}(s,a)|_{a=\pi_{\mathrm{RL}}(s)}] \neq \mathbf{0}, \tag{40}$$

which means that the policy π_{RL} is not an optimal policy. We will discuss the reason for this.

5.4 Approximation Accuracy of Critic

Since policy π_{RL} makes

$$\mathbb{E}_{s\sim\rho^{\pi_{\mathrm{RL}}}}[\nabla_\theta \pi_{\mathrm{RL}}(s)\nabla_a Q(s,a|\omega)|_{a=\pi_{\mathrm{RL}}(s)}] \simeq \mathbf{0} \tag{41}$$

for critic network $Q(s,a|\omega)$, it is a solution to make the approximated gradient be 0. Therefore, the reason why we could not learn the optimal policy is considered to be the low approximation accuracy of critic. In this experiment, the exploration noise is given as small noise. This leads to a bias in the distribution of the action a in the replay buffer. For example, the distribution of actions experienced in the neighborhood of $s = \begin{bmatrix} 0 & 0 \end{bmatrix}$ is shown in Figure 5.

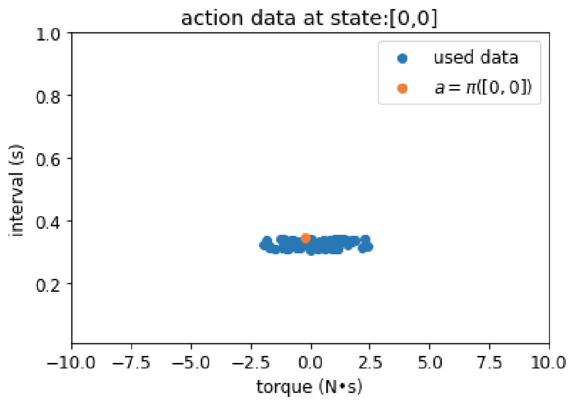

Figure 5: Action distribution in replay buffer around $s = [0\ 0]$.

Since the approximation accuracy of $\nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s)}$ is crucial for the calculation of the policy gradient, the problem is the lack of data around $a = \pi_\theta(s)$ which is represented by the orange dot in Figure 5.

5.5 Ingenuity

We see that small scale exploration noise leads low accuracy of critic's approximation. However, as we confirmed in section 2.5, large scale exploration noise cause problem of experience distribution. Here, by using the information of the control target, we consider to devise a exploration method.

The ideal distribution of replay buffer meets following 2 conditions:

1. The distribution of state s is similar to discounted distribution $\rho^{\pi}(s)$
2. The variance of action a is large for each state s.

Therefore, we propose a method to control the noise scale inversely proportional to the gradient of the state s' of the next step with respect to the change of input a.

$$e \sim \frac{1}{c\|g\| + 1}\mathcal{N}(0, 1) \tag{42}$$

where $g = \frac{ds'}{da}$, c is a hyper parameter.

However, in the case of a non-linear system, it is difficult to compute the gradient of the next state s', so we will compute the gradient for a linearization of the system around the origin state.

As a result of reinforcement learning for inverted pendulum using this exploration method, the obtained policy π_{MBRL} achieve

$$J(\pi_{\mathrm{MBRL}}) = 68.10 \pm 0.018. \tag{43}$$

From equation (36) and (43), it is shown that learning had conducted better than that with small exploration noise.

6 Conclusion

In this paper, a method for obtaining the state feedback control law is proposed. In particular, it is confirmed by numerical examples that the control law obtained by our method outperforms the control law designed by a naive model-based way.

Furthermore, we also tackle the self-triggered control for nonlinear systems, which has not been considered in previous studies, and successfully obtain useful control laws.

Acknowledgments

The author would like to express his sincere gratitude to Professor Yoshito Ohta and Assistant Professor Kenji Kashima for their helpful advices.

References

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

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

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

[4] 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.

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

[6] 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.

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

[8] 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.

A Appendix

This is an appendix. This is a citation [?].

Table 1: This is a table.

	A	B
C	70	80
D	100	0

Master's Thesis

Deep Reinforcement Learning for Self-Triggered Control

Guidance

Professor	Yoshito OHTA
Assistant 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

In this thesis, we construct the mathematical models of students who write the master's thesis and develop efficient algorithms for writing the master's thesis based on the models. We show that the proposed algorithms generate the thesis 65536 times more efficiently than writing by oneself.