Approximate Dynamic Programming with Gaussian Processes for Optimal Control of Continuous-Time Nonlinear Systems

Hirofumi Beppu [*] **Ichiro Maruta** [*] **Kenji Fujimoto** [*]

[*] *Department of Aeronautics and Astronautics, Graduate School of Engineering, Kyoto University, Kyoto, 615-8540, Japan (e-mail: beppu.hirofumi.36a@st.kyoto-u.ac.jp, maruta@kuaero.kyoto-u.ac.jp, k.fujimoto@ieee.org).*

Abstract: In this paper, a new algorithm for realization of approximate dynamic programming (ADP) with Gaussian processes (GPs) for continuous-time (CT) nonlinear input-affine systems is proposed to infinite horizon optimal control problems. The convergence for the ADP algorithm is proven based on the assumption of an exact approximation, where both the cost function and the control input converge to their optimal values, that is, the solution to the Hamilton-Jacobi-Bellman (HJB) equation. The approximation errors, however, are unavoidable in almost every case of applications. In order to tackle the problem, the proposed algorithm is derived with the proof of convergence, where the cost function and the control input, which are both approximated, converge to those of the ADP as the number of data points for GPs approaches infinity. A numerical simulation demonstrates the effectiveness of the proposed algorithm.

Keywords: Approximate dynamic programming, heuristic dynamic programming, value iteration, optimal control, Gaussian processes, nonparametric models.

1. INTRODUCTION

Approximate dynamic programming, or adaptive dynamic programming (ADP) (Werbos, 1992; Sutton and Barto, 1998; Murray et al., 2002) is a set of techniques to solve dynamic programming problems approximately and to achieve the cost function and optimal control policy by solving the Hamilton-Jacobi-Bellman (HJB) equation. The cost function or control policy are approximated to overcome the fundamental problem of classic dynamic programming, called the curse of dimensionality (Powell, 2007). ADP has two types of well established methods: value iterations and policy iterations. The control input sequences in policy iterations can stabilize the systems at every iteration although the initial control policy usually needs to be found among stabilizing inputs. On the other hand, in value iterations, the iteration can start with a simple value function and no need to find a stabilizing input. In this paper, the value iteration based ADP is addressed for nonlinear input-affine systems to obtain the solution to the HJB equation.

A number of policy iteration methods for both of discrete-time (DT) and continuous-time (CT) systems have been developed for the past few decades with proofs of the convergence (Lewis and Vrabie, 2009; Vrabie et al., 2009; Vamvoudakis and Lewis, 2010; Bian et al., 2014). The convergence of value iteration methods for DT systems has been proven in Al-Tamimi et al. (2008). In contrast to policy iteration methods, however, the value iteration methods have limited number of works for CT systems (Wu and Luo, 2012) although physical phenomena are written as CT differential equations. Since the discretization in time cannot avoid numerical errors which influence on the performance of controllers and theoretical properties, it is difficult to apply the results of DT systems to CT systems directly.

The approximation strategy for the control policy or value function is important as well. Parameterized models, e.g., neural networks, are widely used for the approximation (Vrabie and Lewis, 2009). This type of parametric methods can convert the general problems into optimizations for parameters which are easier to solve, but there still remains a problem of determining basis functions. It is hard to know in advance the class of functions to which value functions or control policies belong. Gaussian processes (GPs) (Rasmussen and Williams, 2005) are stochastic, data-driven, and nonparametric models developed in machine learning communities. In GPs, overfitting to the given data can be avoided based on a Bayesian inference framework. These characteristics of GPs are suitable to approximate unknown functions. Deisenroth et al. (2009) have applied GPs successfully to the classic dynamic programming in finite horizon problems for DT nonlinear systems although the convergence has not been proved yet.

In this paper, a new algorithm, Gaussian process heuristic dynamic programming (GPHDP), based on the value iteration based heuristic dynamic programming (HDP) (Werbos, 1992) combined with GPs is proposed to infinite horizon optimal control problems for CT nonlinear input-affine systems. Here, HDP is one of the varieties in ADP. Firstly, the HDP algorithm extended to CT systems is given with a proof of convergence which is based on the assumption of an exact approximation for the value function.

However, the approximation errors exist in almost every case where the guarantee of convergence breaks down. Next, in order to deal with the problem, GPHDP, the proposed algorithm, is derived by approximating the cost function with the mean function of GPs. Then, it is shown that both the cost function and the control input obtained in GPHDP converge to those of the HDP as the number of data points for GPs approaches infinity. Finally, the effectiveness of GPHDP is demonstrated by a numerical simulation.

2. NOTATION

The notations used in this paper are as follows.

- $\mathbb{S}^n_{+}$: the set of $n \times n$ positive semi-definite matrices.
- $\mathbb{S}^n_{++}$: the set of $n \times n$ positive definite matrices.
- $(a)_i$: the i-th component of $a \in \mathbb{R}^n$.
- $\|a\|$: the Euclidean norm of $a \in \mathbb{R}^n$.
- I_n: the $n \times n$ identity matrix.
- $(A)_{ij}$: the i-th row and j-th column component of $A \in \mathbb{R}^{n\times m}$.
- $\lfloor x \rfloor$: the floor function of $x \in \mathbb{R}$.
- $\mathrm{diag}(a)$: the $n \times n$ diagonal matrix whose diagonal components are the components of $a \in \mathbb{R}^n$.

3. PROBLEM SETTING

Consider a nonlinear dynamical system as

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t), \tag{1}$$

which is affine in inputs, where the state $x(t) \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$, and the control input $u(t) \in \mathbb{R}^m$. Assume that the state $x = 0$ is an equilibrium, i.e., $f(0) = 0$ and $u = 0$ when $x = 0$. For the sake of clarity, the solution of the equation (1) with an initial state $x_0 := x(0)$ and a control input u for $t \geq 0$ is defined as

$$x(t)^u_{x_0} := x_0 + \int_0^t f(x(\tau)) + g(x(\tau))u(\tau)\mathrm{d}\tau.$$

The system (1) is supposed to be stabilizable on a compact set $\mathcal{X} \subset \mathbb{R}^n$, i.e., there exists a control input u such that

$$\lim_{t\to\infty} x(t)^u_{x_0} = 0$$

for all initial state $x_0 \in \mathcal{X}$. The aim is to obtain a control input which minimizes the cost function

$$\begin{aligned} V(x_0) &= \int_0^\infty x(t)^u_{x_0}{}^{\mathrm{T}} Q x(t)^u_{x_0} + u(t)^{\mathrm{T}} R u(t)\mathrm{d}t \\ &=: \int_0^\infty L\left(x(t)^u_{x_0}, u(t)\right)\mathrm{d}t, \quad \forall x_0 \in \mathcal{X}, \end{aligned} \tag{2}$$

where $Q \in \mathbb{S}^n_{+}$ and $R \in \mathbb{S}^m_{++}$. The controller u in (2) is required to stabilize the system (1), to ensure the finiteness of the cost function (2), and to be continuous with $u = 0$ when $x = 0$. This class of controllers are defined to be admissible. Suppose that the cost function V in (2) is continuously differentiable. According to the dynamic programming, the right-hand side of (2) can be divided into two parts as

$$\begin{aligned} &\int_0^\infty L\left(x(t)^u_{x_0}, u(t)\right)\mathrm{d}t \\ &= \int_0^{\Delta t} L\left(x(t)^u_{x_0}, u(t)\right)\mathrm{d}t + \int_{\Delta t}^\infty L\left(x(t)^u_{x_0}, u(t)\right)\mathrm{d}t \\ &= \int_0^{\Delta t} L\left(x(t)^u_{x_0}, u(t)\right)\mathrm{d}t + V(x(\Delta t)^u_{x_0}). \end{aligned}$$

The cost function V^* which satisfies the HJB equation

$$V^*(x_0) = \min_u \left(\int_0^{\Delta t} L\left(x(t)^u_{x_0}, u(t)\right)\mathrm{d}t + V^*(x(\Delta t)^u_{x_0}) \right) \tag{3}$$

is time invariant. This fact is known as Bellman's principle of optimality. If Δt is sufficiently small, it follows from (3) that

$$\begin{aligned} 0 &= \min_u \lim_{\Delta t\to 0} \frac{1}{\Delta t} \left\{ \int_0^{\Delta t} L\left(x(t)^u_{x_0}, u(t)\right)\mathrm{d}t \right. \\ &\quad \left. + \left(V^*(x(\Delta t)^u_{x_0}) - V^*(x_0)\right) \right\} \\ &= \min_u \frac{\mathrm{d}}{\mathrm{d}t'} \left(\int_0^{t'} L\left(x(t)^u_{x_0}, u(t)\right)\mathrm{d}t + V^*(x(t')^u_{x_0}) \right)\Bigg|_{t'=0} \\ &= \min_u \left(L(x_0, u(0)) + \frac{\partial V^*}{\partial x}(x_0)(f(x_0) + g(x_0)u(0)) \right). \end{aligned} \tag{4}$$

The first-order necessary condition of the optimal control input u^* is derived by taking the partial derivative of the right-hand side of (4) with respect to u as

$$2u^{\mathrm{T}} R + \frac{\partial V^*(x)}{\partial x} g(x) = 0.$$

Thus the feedback control

$$u^*(x) = -\frac{1}{2} R^{-1} g(x)^{\mathrm{T}} \frac{\partial V^*(x)}{\partial x}^{\mathrm{T}} \tag{5}$$

is obtained. Substituting (5) in (4), the HJB equation turns into the nonlinear partial differential equation

$$\begin{aligned} &\frac{\partial V^*(x)}{\partial x} f(x) - \frac{1}{4} \frac{\partial V^*(x)}{\partial x} g(x) R^{-1} g(x)^{\mathrm{T}} \frac{\partial V^*(x)}{\partial x}^{\mathrm{T}} + x^{\mathrm{T}} Q x \\ &= 0. \end{aligned} \tag{6}$$

In the case of linear systems, (6) becomes the Riccati equation, which is solved efficiently. For general nonlinear systems, however, it is difficult to solve (6) accurately.

The main objective of the proposed method is to obtain the solution $V^*(x)$ in (6). In the following discussion, Δt is assumed to be sufficiently small.

4. HDP ALGORITHM FOR CT SYSTEMS

In this section, the HDP algorithm is extended to deal with nonlinear input-affine CT systems and to solve the HJB equation (3) with a proof of convergence.

4.1 Outline of HDP Algorithm

At the first step of HDP algorithm, the initial value is set

$$V_0 : \mathbb{R}^n \to 0. \tag{7}$$

By using (7), u_0 is obtained as

$$u_0(x) = \arg\min_u \left(\int_0^{\Delta t} L\left(x(t)^u_{x_0}, u(t)\right) \mathrm{d}t + V_0(x(\Delta t)^u_{x_0}) \right) \\ = -\frac{1}{2} R^{-1} g(x)^{\mathrm{T}} \frac{\partial V_0(x)}{\partial x}^{\mathrm{T}}. \tag{8}$$

Then the cost function is updated with u_0 in (8) as

$$V_1(x_0) = \min_u \left(\int_0^{\Delta t} L\left(x(t)^u_{x_0}, u(t)\right) \mathrm{d}t + V_0(x(\Delta t)^u_{x_0}) \right) \\ = \int_0^{\Delta t} L\left(x(t)^{u_0}_{x_0}, u_0(x(t)^{u_0}_{x_0})\right) \mathrm{d}t + V_0(x(\Delta t)^{u_0}_{x_0}). \tag{9}$$

The computation of (8) and (9) is repeated until the cost function converges. The summary of this procedure is given in Algorithm 1.

Algorithm 1. HDP Algorithm for CT Systems
Step 1:
Set $V_0 : \mathbb{R}^n \to 0$ and $i \leftarrow 0$.
Step 2:
Update the control input $u_i(x)$ as

$$u_i(x) = -\frac{1}{2} R^{-1} g(x)^{\mathrm{T}} \frac{\partial V_i(x)}{\partial x}^{\mathrm{T}}. \tag{10}$$

Step 3:
Update the cost function $V_{i+1}(x_0)$ as

$$V_{i+1}(x_0) = \int_0^{\Delta t} L\left(x(t)^{u_i}_{x_0}, u_i(x(t)^{u_i}_{x_0})\right) \mathrm{d}t + V_i(x(\Delta t)^{u_i}_{x_0}). \tag{11}$$

Step 4:
If V_{i+1} converges, then stop and returns the cost function V_{i+1}, otherwise set $i \leftarrow i+1$, go back to *Step 2*, and continue.

4.2 Proof of Convergence for HDP Algorithm

This section gives a proof of convergence of the HDP algorithm based on the result of Al-Tamimi et al. (2008) for DT systems. Here the system of interests is modified to CT ones.

Lemma 1. Let $u_i(x)$ and $V_{i+1}(x_0)$ be the control input and cost function defined by (10) and (11), respectively, and $a_1, a_2, \ldots$ be a sequence of any arbitrary control inputs. Define a sequence of value functions $V^a_{i+1}(x_0)$ for $i = 0, 1, \ldots$ as

$$V^a_{i+1}(x_0) := \int_0^{\Delta t} L\left(x(t)^{a_i}_{x_0}, a_i(t)\right) \mathrm{d}t + V^a_i(x(\Delta t)^{a_i}_{x_0}).$$

Then for all i, $V_i(x_0) \le V^a_i(x_0)$ holds if $V_0(x_0) = V^a_0(x_0) = 0$.

Proof. Since the right-hand side of (11) is minimized by $u_i(x)$ and $V_0(x_0) = V^a_0(x_0) = 0$, for $i = 1$, it follows that

$$V_1(x_0) = \int_0^{\Delta t} L\left(x(t)^{u_0}_{x_0}, u_0(x(t)^{u_0}_{x_0})\right) \mathrm{d}t + V_0(x(\Delta t)^{u_0}_{x_0}) \\ = \int_0^{\Delta t} L\left(x(t)^{u_0}_{x_0}, u_0(t)\right) \mathrm{d}t \\ \le \int_0^{\Delta t} L\left(x(t)^{a_0}_{x_0}, a_0(t)\right) \mathrm{d}t = V^a_1(x_0).$$

Suppose $V_i(x_0) \le V^a_i(x_0)$ holds for $i \ge 1$. Then by (10), it follows that

$$V_{i+1}(x_0) = \int_0^{\Delta t} L\left(x(t)^{u_i}_{x_0}, u_i(x(t)^{u_i}_{x_0})\right) \mathrm{d}t + V_i(x(\Delta t)^{u_i}_{x_0}) \\ \le \int_0^{\Delta t} L\left(x(t)^{a_i}_{x_0}, a_i(t)\right) \mathrm{d}t + V_i(x(\Delta t)^{a_i}_{x_0}) \\ \le \int_0^{\Delta t} L\left(x(t)^{a_i}_{x_0}, a_i(t)\right) \mathrm{d}t + V^a_i(x(\Delta t)^{a_i}_{x_0}) \\ = V^a_{i+1}(x_0).$$

By mathematical induction, it is shown that $V_i(x_0) \le V^a_i(x_0)$ holds for all i. $\square$

Lemma 2. If the system (1) is controllable, there exits an upper bound $U(x_0)$ such that $V_i(x_0) \le U(x_0)$ holds for $i = 0, 1, \ldots$, where $V_{i+1}(x_0)$ is defined as (11) with $V_0(x_0) = 0$.

Proof. Let s be any stabilizing and admissible control input. Define a sequence of value functions $V^s_{i+1}(x_0)$ for $i = 0, 1, \ldots$ as

$$V^s_{i+1}(x_0) = \int_0^{\Delta t} L\left(x(t)^s_{x_0}, s(t)\right) \mathrm{d}t + V^s_i(x(\Delta t)^s_{x_0}), \tag{12}$$

which satisfies $V^s_0(x_0) = V_0(x_0) = 0$. Subtracting $V^s_i(x_0)$ from (12) yields

$$V^s_{i+1}(x_0) - V^s_i(x_0) \\ = V^s_i(x(\Delta t)^s_{x_0}) - V^s_{i-1}(x(\Delta t)^s_{x_0}) \\ = V^s_{i-1}(x(2\Delta t)^s_{x(\Delta t)}) - V^s_{i-2}(x(2\Delta t)^s_{x(\Delta t)}) \\ \vdots \\ = V^s_1(x(i\Delta t)^s_{x((i-1)\Delta t)}) - V^s_0(x(i\Delta t)^s_{x((i-1)\Delta t)}).$$

Then it follows from $V^s_0(x_0) = 0$ that

$$V^s_{i+1}(x_0) \\ = V^s_1(x(i\Delta t)^s_{x((i-1)\Delta t)}) + V^s_i(x_0) \\ = V^s_1(x(i\Delta t)^s_{x((i-1)\Delta t)}) + V^s_1(x((i-1)\Delta t)^s_{x((i-2)\Delta t)}) \\ + V^s_{i-1}(x_0) \\ \vdots \\ = V^s_1(x(i\Delta t)^s_{x((i-1)\Delta t)}) + \cdots + V^s_1(x_0) \\ = \sum_{n^i=0}^{i} \int_{n^i\Delta t}^{(n^i+1)\Delta t} L\left(x(t)^s_{x(n_i\Delta t)}, s(t)\right) \mathrm{d}t \\ \le \sum_{n^i=0}^{\infty} \int_{n^i\Delta t}^{(n^i+1)\Delta t} L\left(x(t)^s_{x(n_i\Delta t)}, s(t)\right) \mathrm{d}t \\ = \int_0^{\infty} L\left(x(t)^s_{x_0}, s(t)\right) \mathrm{d}t.$$

Since the admissibility of s guarantees the finiteness of cost functions, it follows that

$$V^s_{i+1}(x_0) \le \int_0^{\infty} L\left(x(t)^s_{x_0}, s(t)\right) \mathrm{d}t = U(x_0).$$

In Lemma 1, set $a_i = s$ for all i, then $V_i^a(x_0) = V_i^s(x_0)$. This leads to $V_i(x_0) \leq V_i^s(x_0) \leq U(x_0)$ for all i by using Lemma 1, which completes the proof. $\square$

Lemma 3. If the condition in Lemma 2 holds and the HJB equation (3) has a unique solution, then there exists a minimum upper bound $V^*(x_0) \leq U(x_0)$ such that $V^*(x_0)$ solves (6) with $V_i(x_0) \leq V^*(x_0) \leq U(x_0)$ for $i = 0, 1, \ldots$.

Proof. Let s be any stabilizing and admissible control input defined in the proof of Lemma 2. Then it follows that

$$\begin{aligned} V^*(x_0) &= \int_0^\infty L\left(x(t)_{x_0}^{u^*}, u^*(x(t)_{x_0}^{u^*})\right) \mathrm{d}t \\ &\leq \int_0^\infty L\left(x(t)_{x_0}^{s}, s(t)\right) \mathrm{d}t = U(x_0). \end{aligned}$$

Therefore, it follows from Lemma 2 that

$$V_i(x_0) \leq V_i^{s=u^*}(x_0) \leq V^*(x_0) \leq U(x_0),$$

and Lemma 3 is proved. $\square$

The convergence of Algorithm 1 is shown in the following.

Theorem 4. Let $u_i(x)$ and $V_{i+1}(x_0)$ be defined in (10) and (11), respectively. It follows that $V_i(x_0) \leq V_{i+1}(x_0)$ for all i if $V_0(x_0) = 0$. When $i \to \infty$, V_i and u_i converge to V^*, the solution to the HJB equation (6), and u^*, respectively.

Proof. Let a_i and V_{i+1}^a be defined in Lemma 1. If $V_0(x_0) = V_0^a(x_0) = 0$, then from Lemma 1, $V_i(x_0) \leq V_i^a(x_0)$ holds for all i. Because the control input a_i can be chosen arbitrarily, set $a_i = u_{i+1}$ and

$$\begin{aligned} V_{i+1}^a(x_0) = &\int_0^{\Delta t} L\left(x(t)_{x_0}^{u_{i+1}}, u_{i+1}(x(t)_{x_0}^{u_{i+1}})\right) \mathrm{d}t \\ &+ V_i^a(x(\Delta t)_{x_0}^{u_{i+1}}). \end{aligned} \tag{13}$$

Now it will be shown that $V_i^a(x_0) \leq V_{i+1}(x_0)$ for all i if $V_0(x_0) = V_0^a(x_0) = 0$. When $V_0(x_0) = V_0^a(x_0) = 0$, it follows from (11) that

$$V_1(x_0) - V_0^a(x_0) = \int_0^{\Delta t} L\left(x(t)_{x_0}^{u_i}, u_i(x(t)_{x_0}^{u_i})\right) \mathrm{d}t \geq 0.$$

Next suppose that $V_{i-1}^a(x_0) \leq V_i(x_0)$ for $i \geq 1$. Then it follows from (11) and (13) that

$$V_{i+1}(x_0) - V_i^a(x_0) = V_i(x(\Delta t)_{x_0}^{u_i}) - V_{i-1}^a(x(\Delta t)_{x_0}^{u_i}) \geq 0.$$

By mathematical induction, it is shown that $V_i(x_0) \geq V_{i+1}^a(x_0)$ for all i. Combining this with the result of Lemma 1, it follows that $V_i(x_0) \leq V_{i+1}(x_0)$ for all i.

From Lemma 3, it follows that $V_\infty(x_0) \leq V^*(x_0)$. In order to see whether $V_\infty(x_0)$ is $V^*(x_0)$, (11) can be rewritten for $i \to \infty$ as

$$\begin{aligned} &V_\infty(x(\Delta t)_{x_0}^{u_\infty}) - V_\infty(x_0) \\ &= -\int_0^{\Delta t} L\left(x(t)_{x_0}^{u_\infty}, u_\infty(x(t)_{x_0}^{u_\infty})\right) \mathrm{d}t \leq 0. \end{aligned}$$

Thus $V_\infty(x_0)$ is a Lyapunov function for u_∞ which is stabilizable and admissible. This means that $V_\infty(x_0)$ is a candidate for the solution to the HJB equation (6). Since by Lemma 2, $V_i(x_0) \leq V_\infty(x_0)$ holds for all i, it also follows from Lemma 2 that $V_\infty(x_0) = U(x_0)$. Moreover, from Lemma 3 it follows that $V^*(x_0) \leq V_\infty(x_0) = U(x_0)$. This suggests that $V^*(x_0) \leq V_\infty(x_0) \leq V^*(x_0)$. Therefore, it follows that $\lim_{i\to\infty} V_i(x_0) = V^*(x_0)$ and $\lim_{i\to\infty} u_i = u^*$, which completes the proof. $\square$

5. GPHDP ALGORITHM

In order to implement Algorithm 1 in Section 4.1 practically, V_i can be approximated with some basis functions whose coefficients are updated instead. Since, however, it is unknown which classes of functions V_i belong to, it is difficult to determine the basis functions in advance. In addition, the convergence of the algorithm may break down because the proof of convergence in Section 4.2 is derived under the assumption that each step of the algorithm can be computed exactly. In this section, a new algorithm, GPHDP, is proposed based on the HDP algorithm for CT systems incorporated with GPs to overcome the problems mentioned above.

GPs are a tool for estimating functions generating data as probability distributions. There are two main reasons to apply GPs to implement the HDP algorithm. One of them is that since GPs are nonparametric, if data are given properly, and the kernel functions and hyperparameters are set appropriately, they can output any smooth continuous functions. Another is that GPs can interpolate with uncertainties due to lack of data. This can help with avoiding overconfidence of the evaluations of estimated functions using discrete points.

5.1 Outline of GPHDP Algorithm

In the following, suppose that the given data set $\{X, \hat{\boldsymbol{V}}_i\}$ is composed of the state vectors x_d for $d = 1, \ldots, D$ and the corresponding measurement

$$\hat{V}_{id} = V_i(x_d) + e, \quad e \sim \mathcal{N}(0, \sigma_e^2),$$

where X and $\hat{\boldsymbol{V}}_i$ are defined as

$$\begin{aligned} X &:= [x_1, \ldots, x_D], \\ \hat{\boldsymbol{V}}_i &:= [\hat{V}_{i1}, \ldots, \hat{V}_{iD}]^{\mathrm{T}}. \end{aligned}$$

The aim here is to estimate the latent function V_i based on the data set. In a framework of Bayesian inference, the estimate of V_i is computed probabilistically by

$$p(V_i|X, \hat{\boldsymbol{V}}_i) = \frac{p(\hat{\boldsymbol{V}}_i|V_i, X)p(V_i)}{p(\hat{\boldsymbol{V}}_i|X)},$$

where $p(\hat{\boldsymbol{V}}_i|V_i, X)$, $p(V_i)$, $p(\hat{\boldsymbol{V}}_i|X)$ are a likelihood function, a prior distribution, and a marginal likelihood function, respectively. In estimating the latent function with GPs, it is possible to put a prior distribution directly on the function space without explicit parametrization. This prior assumes a smoothness of the function V_i. GPs are specified fully with the mean function m and the covariance function or kernel function k. Thus GPs are regarded as probability distributions on functions. When the latent function V_i follows a GP, the notation

$$V_i \sim \mathcal{GP}(m, k)$$

is used. For simplicity, set $m \equiv 0$ in the following without loss of generality. In GP modeling based on the data set $\{X, \hat{\boldsymbol{V}}_i\}$, the probability distribution of V_i corresponding to any state x is expressed with the mean function and covariance function as

$$\begin{aligned} m_i(x) &= \mathbb{E}_{V_i}[V_i] = k(x, X)(K_{XX} + \sigma_e^2 I_n)^{-1}\hat{\boldsymbol{V}}_i, \qquad (14) \\ k_i(x, x) &= \mathrm{var}_{V_i}[V_i] \\ &= k(x, x) - k(x, X)(K_{XX} + \sigma_e^2 I_n)^{-1}k(x, X)^{\mathrm{T}}, \end{aligned}$$

respectively, where $k(x, X)$ and K_{XX} are defined as

$$k(x, X) = [k(x, x_1), \dots, k(x, x_D)],$$
$$(K_{XX})_{jj'} = k(x_j, x_{j'}), \quad j, j' = 1, 2, \dots, D$$

respectively. For the covariance function, a Gaussian kernel

$$k(x_j, x_{j'}) = \exp\left(-\frac{1}{2h_p^2}(x_j - x_{j'})^{\mathrm{T}}(x_j - x_{j'})\right) \tag{15}$$

is one of the common choices where h_p is a hyperparameter (or a set of hyperparameters) which can be determined by given data. In GPHDP algorithm, the mean function m_i is used to approximate V_i in (10) and (11). The outline of the algorithm is given in Algorithm 2 where ϵ is a constant for the convergence judgment.

Algorithm 2. GPHDP Algorithm
Step 1:
Set $\hat{V}_{0d} \leftarrow 0$ for all $d \in \{0, 1, \dots, D\}$ and $i \leftarrow 0$.
Step 2:
Update the control input $\hat{u}_i(x)$ as

$$\hat{u}_i(x) = -\frac{1}{2}R^{-1}g(x)^{\mathrm{T}}\frac{\partial m_i(x)}{\partial x}^{\mathrm{T}}, \tag{16}$$

where $m_i(x)$ is computed by using (14) based on the value of $\hat{\boldsymbol{V}}_i$.
Step 3:
For all d, update the data of the cost function $\hat{V}_{(i+1)d}$

$$\hat{V}_{(i+1)d} = \int_0^{\Delta t} L\left(x(t)_{x_d}^{\hat{u}_i}, \hat{u}_i(x(t)_{x_d}^{\hat{u}_i})\right)\mathrm{d}t + m_i(x(\Delta t)_{x_d}^{\hat{u}_i}), \tag{17}$$

where $m_i(x(\Delta t)_{x_d}^{\hat{u}_i})$ is computed by using (14) based on the value of $\hat{\boldsymbol{V}}_i$. If $\hat{V}_{(i+1)d} - \hat{V}_{id} < 0$, then set $\hat{V}_{(i+1)d} \leftarrow \hat{V}_{id}$.
Step 4:
If it follows that

$$\|\hat{\boldsymbol{V}}_{i+1} - \hat{\boldsymbol{V}}_i\| < \epsilon,$$

then stop and returns the mean function m_{i+1}, otherwise set $i \leftarrow i + 1$, go back to *Step 2*, and continue.

Remark 5. At *Step 3* in Algorithm 2, $\hat{V}_{(i+1)d}$ is replaced by $\hat{V}_{id}$ if $\hat{V}_{(i+1)d} - \hat{V}_{id} < 0$ to avoid the contradiction in Theorem 4. The reason why this case happens occasionally is that m_i is influenced by stochastic factors.

5.2 Proof of Convergence for GPHDP Algorithm

In this section, the proof of convergence for the GPHDP algorithm is given. That is, the convergence in terms of the number of data points for (16) and (17), i.e., $\lim_{D\to\infty}\hat{V}_{id} = V_i(x_d)$ and $\lim_{D\to\infty}\hat{u}_i(x) = u_i(x)$, is proven.

For simplicity, a hyperrectangle $\mathcal{X}$ is considered here and let

$$(\underline{x})_n := \min_{x\in\mathcal{X}}(x)_n, \quad (\bar{x})_n := \max_{x\in\mathcal{X}}(x)_n.$$

Now X is supposed to be arranged on the n-dimensional grid points, that is,

$$\left\{\left[(\underline{x})_1 + \frac{(\bar{x})_1 - (\underline{x})_1}{N-1}c_1, (\underline{x})_2 + \frac{(\bar{x})_2 - (\underline{x})_2}{N-1}c_2, \dots, (\underline{x})_n + \frac{(\bar{x})_n - (\underline{x})_n}{N-1}c_n\right]\middle| c_1, c_2, \dots, c_n \in \{0, 1, \dots, N-1\}\right\}.$$

Thus the total number of data points becomes $D = N^n$. For the discussion below, the following assumption is made.

Assumption 6. Given a data set $\{X, \hat{\boldsymbol{V}}_i\}$, there exist a kernel function k and a set of hyperparameters h_p such that

$$m_i(x_d) = k(x_d, X)(K_{XX} + \sigma_e^2 I_n)^{-1}\hat{\boldsymbol{V}}_i = \hat{V}_{id} \tag{18}$$

holds for all i and d.

Remark 7. In GPs, output functions can be expressed by infinite-dimensional feature vectors with some specific kernels, e.g., a Gaussian kernel in (15), which return any smooth continuous functions (Rasmussen and Williams, 2005). Assumption 6 is based on this characteristic. In fact, if $\sigma_e^2 = 0$, then (18) holds strictly.

Firstly, the lemma which is necessary to prove the convergence of $\hat{u}(x(t))$ is given in the following.

Lemma 8. Given a data set $\{X, \boldsymbol{V}_i\}$ for the system (1). If Assumption 6 holds, then for all $\varepsilon > 0$, there exists an $\bar{N} > 0$ such that for all $N > \bar{N}$, $x \in \mathcal{X}$, and i, it follows that

$$\|\hat{u}_i(x) - u_i(x)\| < \varepsilon.$$

Here $\boldsymbol{V}_i$ is defined as

$$\boldsymbol{V}_i := [V_{i1}, \dots, V_{iD}]^{\mathrm{T}}$$
$$:= [V_i(x_1), \dots, V_i(x_D)]^{\mathrm{T}}.$$

Proof. Choose x_d from X such that

$$(x_d)_\nu = \left\lfloor\frac{(x)_\nu}{(\Delta(N))_\nu}\right\rfloor(\Delta(N))_\nu, \quad \nu \in \{1, 2, \dots, n\}, \tag{19}$$

where

$$(\Delta(N))_\nu := \frac{(\underline{x})_\nu - (\bar{x})_\nu}{N-1} > 0.$$

Suppose that x_d is reselected depending on N. Since $\lim_{N\to\infty}(\Delta(N))_\nu = 0$ and $\lim_{N\to\infty} x_d = x$, it follows that

$$\lim_{N\to\infty}\frac{V_i(x_d + (\Delta(N))_\nu b_\nu) - V_i(x_d)}{(\Delta(N))_\nu}$$
$$= \lim_{(\Delta(N))_\nu\to 0}\frac{V_i(x_d + (\Delta(N))_\nu b_\nu) - V_i(x_d)}{(\Delta(N))_\nu} = \frac{\partial V_i(x)}{\partial(x)_\nu},$$

where b_ν denotes a standard basis defined as

$$b_\nu := [0, \dots, 0, \underbrace{1}_{\nu\text{-th}}, 0, \dots, 0].$$

Thus it follows that

$$\lim_{N\to\infty}\left\|\left[\frac{V_i(x_d + (\Delta(N))_1 b_1) - V_i(x_d)}{(\Delta(N))_1}, \dots, \frac{V_i(x_d + (\Delta(N))_n b_n) - V_i(x_d)}{(\Delta(N))_n}\right]^{\mathrm{T}} - \frac{\partial V_i(x)}{\partial x}^{\mathrm{T}}\right\| = 0.$$

The derivative of the mean function can be written as

$$\frac{\partial m_i}{\partial(x)_\nu}(x) = \frac{V_i(x_d + (\Delta(N))_\nu b_\nu) - V_i(x_d)}{(\Delta(N))_\nu} + (E_i(x))_\nu,$$

where $E_i(x)$ denotes an error. Since from Assumption 6, the data set is interpolated exactly, $\lim_{N\to\infty}|(E_i(x))_\nu| =$

0. Therefore, for all $\varepsilon > 0$, there exist δ_1, δ_2, and $\bar{N} > 0$ such that for all $N > \bar{N}$, it follows that

$$\left\| \left[\frac{V_i(x_d + (\Delta(N))_1 b_1) - V_i(x_d)}{(\Delta(N))_1}, \ldots, \right. \right. \\ \left. \left. \frac{V_i(x_d + (\Delta(N))_n b_n) - V_i(x_d)}{(\Delta(N))_n} \right]^{\mathrm{T}} - \frac{\partial V_i(x)}{\partial x}^{\mathrm{T}} \right\| < \delta_1,$$

$$\|E_i(x)\| < \delta_2$$

and

$$\|\hat{u}_i(x) - u_i(x)\| \\ = \left\| -\frac{1}{2} R^{-1} g(x(t))^{\mathrm{T}} \left(\frac{\partial m_i(x)}{\partial x}^{\mathrm{T}} - \frac{\partial V_i(x)}{\partial x}^{\mathrm{T}} \right) \right\| \\ \le \left\| -\frac{1}{2} R^{-1} g(x)^{\mathrm{T}} \right\| \cdot \left\| \left[\frac{V_i(x_d + (\Delta(N))_1 b_1) - V_i(x_d)}{(\Delta(N))_1}, \ldots, \right. \right. \\ \left. \left. \frac{V_i(x_d + (\Delta(N))_n b_n) - V_i(x_d)}{(\Delta(N))_n} \right]^{\mathrm{T}} - \frac{\partial V_i(x)}{\partial x}^{\mathrm{T}} + E_i(x) \right\| \\ \le \left\| -\frac{1}{2} R^{-1} g(x)^{\mathrm{T}} \right\| \left\{ \left\| \left[\frac{V_i(x_d + (\Delta(N))_1 b_1) - V_i(x_d)}{(\Delta(N))_1}, \ldots, \right. \right. \right. \\ \left. \left. \frac{V_i(x_d + (\Delta(N))_n b_n) - V_i(x_d)}{(\Delta(N))_n} \right]^{\mathrm{T}} - \frac{\partial V_i(x)}{\partial x}^{\mathrm{T}} \right\| \\ \left. + \|E_i(x)\| \right\} < \left\| -\frac{1}{2} R^{-1} g(x)^{\mathrm{T}} \right\| (\delta_1 + \delta_2) < \varepsilon.$$

This completes the proof of Lemma 8. □

Next define $\tilde{V}_{(i+1)d}$ as

$$\tilde{V}_{(i+1)d} := \int_0^{\Delta t} L\left(x(t)_{x_d}^{u_i}, u_i(x(t)_{x_d}^{u_i})\right) \mathrm{d}t + m_i(x(\Delta t)_{x_d}^{u_i}). \tag{20}$$

Then the following lemma about the relationship between $\hat{V}_{(i+1)d}$ and $\tilde{V}_{(i+1)d}$ is derived.

Lemma 9. If Lemma 8 holds, then for all $\varepsilon > 0$, there exists an $\bar{N} > 0$ such that for all $N > \bar{N}$, $x_d \in X$, d, and i, it follows that

$$|\hat{V}_{(i+1)d} - \tilde{V}_{(i+1)d}| < \varepsilon.$$

Proof. By subtracting (20) from (17), it follows that

$$\hat{V}_{(i+1)d} - \tilde{V}_{(i+1)d} \\ = \int_0^{\Delta t} L\left(x(t)_{x_d}^{\hat{u}_i}, \hat{u}_i(x(t)_{x_d}^{\hat{u}_i})\right) - L\left(x(t)_{x_d}^{u_i}, u_i(x(t)_{x_d}^{u_i})\right) \mathrm{d}t \\ + m_i(x(\Delta t)_{x_d}^{\hat{u}_i}) - m_i(x(\Delta t)_{x_d}^{u_i}).$$

Since it follows that $\lim_{N\to\infty} \hat{u}_i = u_i$ from Lemma 8,

$$\lim_{N\to\infty} \left(L\left(x(t)_{x_d}^{\hat{u}_i}, \hat{u}_i(x(t)_{x_d}^{\hat{u}_i})\right) - L\left(x(t)_{x_d}^{u_i}, u_i(x(t)_{x_d}^{u_i})\right) \right) = 0$$

holds from the continuity of L, and

$$\lim_{N\to\infty} (m_i(x(\Delta t)_{x_d}^{\hat{u}_i}) - m_i(x(\Delta t)_{x_d}^{u_i})) = 0$$

also holds from the continuity of m_i. Therefore, for all $\varepsilon > 0$, there exist δ_3, δ_4, and $\bar{N} > 0$ such that for all $N > \bar{N}$, it follows that

$$\left| L\left(x(t)_{x_d}^{\hat{u}_i}, \hat{u}_i(x(t)_{x_d}^{\hat{u}_i})\right) - L\left(x(t)_{x_d}^{u_i}, u_i(x(t)_{x_d}^{u_i})\right) \right| < \delta_3, \\ \left| m_i(x(\Delta t)_{x_d}^{\hat{u}_i}) - m_i(x(\Delta t)_{x_d}^{u_i}) \right| < \delta_4,$$

and

$$|\hat{V}_{(i+1)d} - \tilde{V}_{(i+1)d}| \\ = \left| \int_0^{\Delta t} L\left(x(t)_{x_d}^{\hat{u}_i}, \hat{u}_i(x(t)_{x_d}^{\hat{u}_i})\right) - L\left(x(t)_{x_d}^{u_i}, u_i(x(t)_{x_d}^{u_i})\right) \mathrm{d}t \right. \\ \left. + m_i(x(\Delta t)_{x_d}^{\hat{u}_i}) - m_i(x(\Delta t)_{x_d}^{u_i}) \right| \\ \le \left| \int_0^{\Delta t} L\left(x(t)_{x_d}^{\hat{u}_i}, \hat{u}_i(x(t)_{x_d}^{\hat{u}_i})\right) - L\left(x(t)_{x_d}^{u_i}, u_i(x(t)_{x_d}^{u_i})\right) \mathrm{d}t \right| \\ + \left| m_i(x(\Delta t)_{x_d}^{\hat{u}_i}) - m_i(x(\Delta t)_{x_d}^{u_i}) \right| \\ \le \int_0^{\Delta t} \left| L\left(x(t)_{x_d}^{\hat{u}_i}, \hat{u}_i(x(t)_{x_d}^{\hat{u}_i})\right) - L\left(x(t)_{x_d}^{u_i}, u_i(x(t)_{x_d}^{u_i})\right) \right| \mathrm{d}t \\ + \left| m_i(x(\Delta t)_{x_d}^{\hat{u}_i}) - m_i(x(\Delta t)_{x_d}^{u_i}) \right| \\ < \int_0^{\Delta t} \delta_3 \mathrm{d}t + \delta_4 = \Delta t \delta_3 + \delta_4 < \varepsilon.$$

Thus Lemma 9 is proven. □

By using the two lemmas proven above, the proof of convergence for GPHDP algorithm is derived as follows.

Theorem 10. Given $\hat{\boldsymbol{V}}_0 = 0$ for the system (1). Then under Assumption 6, for all $\varepsilon > 0$, there exists an $\bar{N} > 0$ such that for all $N > \bar{N}$, $x \in \mathcal{X}$, $x_d \in X$, d, and i, it follows that

$$|\hat{u}_i(x) - u_i(x)| < \varepsilon, \\ |\hat{V}_{(i+1)d} - V_{i+1}(x_d)| < \varepsilon.$$

Proof. Since $\hat{\boldsymbol{V}}_0 = 0$, from (14), it follows that

$$m_0(x) = 0 = V_0(x). \tag{21}$$

Substituting (21) into (16) yields

$$\hat{u}_0(x) = -\frac{1}{2} R^{-1} g(x)^{\mathrm{T}} \frac{\partial m_0(x)}{\partial x}^{\mathrm{T}} = 0 = u_0(x).$$

Then it follows from (17) that

$$\hat{V}_{1d} = \int_0^{\Delta t} L\left(x(t)_{x_d}^{\hat{u}_0}, \hat{u}_0(x(t)_{x_d}^{\hat{u}_0})\right) \mathrm{d}t + m_0(x(\Delta t)_{x_d}^{\hat{u}_0}) \\ = \int_0^{\Delta t} L\left(x(t)_{x_d}^{u_0}, u_0(x(t)_{x_d}^{u_0})\right) \mathrm{d}t + V_0(x(\Delta t)_{x_d}^{u_0}) \\ = V_1(x_d). \tag{22}$$

In the following, it is shown that for all i,

$$\lim_{N\to\infty} \hat{u}_i(x) = u_i(x), \quad \lim_{N\to\infty} \hat{V}_{(i+1)d} = V_{i+1}(x_d),$$

by mathematical induction. Firstly, it is proved that

$$\lim_{N\to\infty} \hat{u}_1(x) = u_1(x), \quad \lim_{N\to\infty} \hat{V}_{2d} = V_2(x_d), \\ \lim_{N\to\infty} \hat{V}_{2d} = V_2(x_d)$$

hold. By (22) and Lemma 8, it follows that

$$\lim_{N\to\infty} \hat{u}_1(x) = u_1(x)$$

for all $x \in \mathcal{X}$. Since now Lemma 8 holds, from Lemma 9,

$$\lim_{N\to\infty} \hat{V}_{2d} = \tilde{V}_{2d} \tag{23}$$

holds. Next choose $x_{d'}$ from X such that $(x_{d'})_n$ is defined in (19) by replacing x with $x(\Delta t)_{x_d}^{u_1}$. Denote $m_1(x(\Delta t)_{x_d}^{u_1})$ with an error term $E'_i(x_{d'})$ as

$$m_1(x(\Delta t)_{x_d}^{u_1}) = m_1(x_{d'}) + E'_1(x_{d'}) \\ = \hat{V}_{1d'} + E'_1(x_{d'}) \\ = V_1(x_{d'}) + E'_1(x_{d'}). \tag{24}$$

Since $\lim_{N\to\infty} x_{d'} = x(\Delta t)^{u_1}_{x_d}$ and $\lim_{N\to\infty} E_1'(x_{d'}) = 0$ from Assumption 6, it follows from (24) that

$$\lim_{N\to\infty} m_1(x(\Delta t)^{u_1}_{x_d}) = V_1(x(\Delta t)^{u_1}_{x_d}).$$

Therefore, it follows that

$$\begin{aligned}&\lim_{N\to\infty} |\tilde{V}_{2d} - V_2(x_d)|\\ &= \lim_{N\to\infty} \left| \int_0^{\Delta t} L\left(x(t)^{u_1}_{x_d}, u_1(x(t)^{u_1}_{x_d})\right) \mathrm{d}t + m_1(x(\Delta t)^{u_1}_{x_d}) \right.\\ &\left. - \int_0^{\Delta t} L\left(x(t)^{u_1}_{x_d}, u_1(x(t)^{u_1}_{x_d})\right) \mathrm{d}t - V_1(x(\Delta t)^{u_1}_{x_d}) \right|\\ &= \lim_{N\to\infty} |m_1(x(\Delta t)^{u_1}_{x_d}) - V_1(x(\Delta t)^{u_1}_{x_d})| = 0,\end{aligned}$$

that is,

$$\lim_{N\to\infty} \tilde{V}_{2d} = V_2(x_d). \tag{25}$$

Moreover, from (23) and (25), it is shown that

$$\lim_{N\to\infty} \hat{V}_{2d} = V_2(x_d). \tag{26}$$

Next assume that for $i \geq 1$,

$$\lim_{N\to\infty} \hat{u}_i(x) = u_i(x), \quad \lim_{N\to\infty} \tilde{V}_{(i+1)d} = V_{i+1}(x_d),$$
$$\lim_{N\to\infty} \hat{V}_{(i+1)d} = V_{i+1}(x_d)$$

hold. Then because the data set $\{X, \boldsymbol{V}_{i+1}\}$ is given, from Lemma 8, it follows that

$$\lim_{N\to\infty} \hat{u}_{i+1}(x) = u_{i+1}(x).$$

By combining this with Lemma 9, it follows that

$$\lim_{N\to\infty} \hat{V}_{(i+2)d} = \tilde{V}_{(i+2)d}.$$

Taking the same procedure as in deriving (26), it is shown that

$$\lim_{N\to\infty} \tilde{V}_{(i+2)d} = V_{i+2}(x_d), \quad \lim_{N\to\infty} \hat{V}_{(i+2)d} = V_{i+2}(x_d).$$

By induction, it is proved that for all i,

$$\lim_{N\to\infty} \hat{u}_i(x) = u_i(x), \quad \lim_{N\to\infty} \hat{V}_{(i+1)d} = V_{i+1}(x_d),$$

and this completes the proof of Theorem 10. $\square$

6. EVALUATION OF GPHDP

In order to see the effectiveness of GPHDP, the algorithm is applied to swinging up a pendulum, which is known as a nonlinear and challenging optimal control problem.

6.1 Example

Consider a CT pendulum system with $x = [\theta, \dot{\theta}]^{\mathrm{T}}$ as

$$\dot{x} = f(x) + g(x)u = \begin{pmatrix} \dot{\theta} \\ \dfrac{mg_r l}{ml^2} \sin\theta \end{pmatrix} + \begin{pmatrix} 0 \\ \dfrac{1}{ml^2} \end{pmatrix} u, \tag{27}$$

where θ (rad) and $\dot{\theta}$ (rad/s) are angle and angular velocity of the pendulum, respectively, and $m = 1$ kg is the mass of pendulum, $g_r = 9.8\,\mathrm{m/s^2}$ is the gravitational acceleration, and $l = 1$ m is the length of pendulum. The goal is to swing up the pendulum and balance it in the inverted position at $[\theta, \dot{\theta}]^{\mathrm{T}} = 0$ with an arbitrary initial state. The cost function is chosen as

$$V(x_0) = \int_0^{\infty} x(t)^u_{x_0}{}^{\mathrm{T}} Q x(t)^u_{x_0} + u(x(t)^u_{x_0})^{\mathrm{T}} R u(x(t)^u_{x_0}) \mathrm{d}t, \tag{28}$$

where $Q = \mathrm{diag}([1, 0.01]^{\mathrm{T}})$ and $R = 1$.

For the parameters of GPHDP, set $\Delta t = 0.1$ s, X to be on $D = 31 \times 31 = 961$ uniform grids in a region of $[-2\pi, 2\pi] \times [-10 \times 10]$, k to be a Gaussian kernel in (15), $h_p = 1$, $\sigma_e^2 = 0$ at the origin, otherwise $\sigma_e^2 = 1 \times 10^{-6}$, and $\epsilon = D \times 10^{-5}$.

6.2 Results

The iteration of Algorithm 2 stopped at the $i + 1 = 100$th step. The mean function $m_{100}(x)$ in (14) corresponding to the cost function obtained by GPHDP is shown in Fig. 1, where the blue circles denote the positions of data set $\{X, \hat{\boldsymbol{V}}_{100}\}$. Figure 2 shows the two-dimensional mapping of the mean value.

From Fig. 3, it is observed that the norm $\|\hat{\boldsymbol{V}}_{i+1} - \hat{\boldsymbol{V}}_i\|$ converges well by the end of the iteration as $\|\hat{\boldsymbol{V}}_{100} - \hat{\boldsymbol{V}}_{99}\| = 8.1 \times 10^{-3}$.

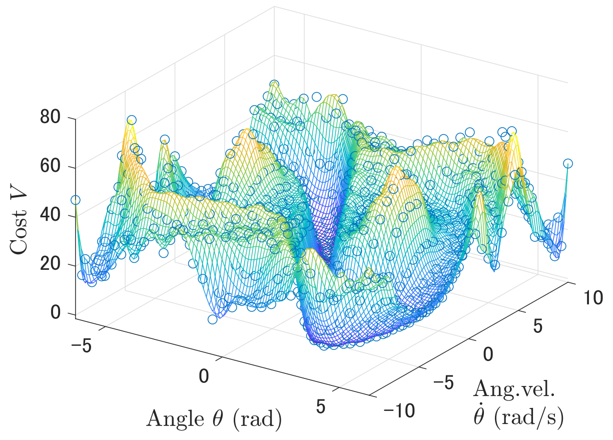

Fig. 1. Mean value of the cost function in GPHDP.

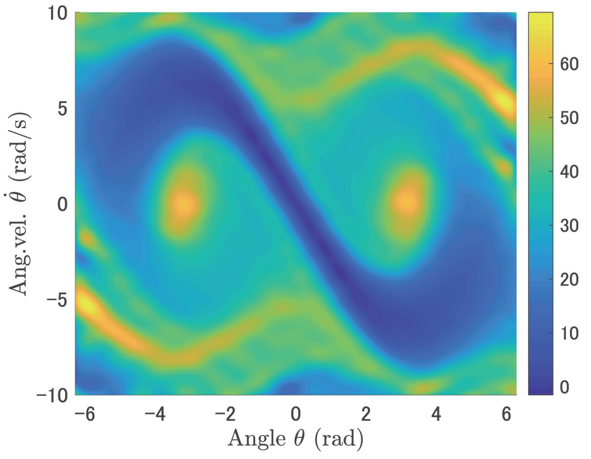

Fig. 2. The 2D mapping of the mean value in GPHDP.

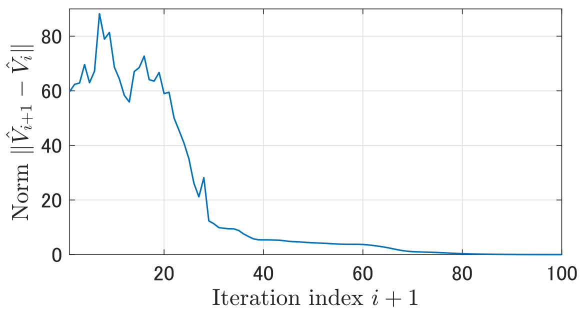

Fig. 3. The iteration history of the norm $\|\hat{\boldsymbol{V}}_{i+1} - \hat{\boldsymbol{V}}_i\|$.

The results of numerical simulation with the initial state $[\theta_0, \dot{\theta}_0]^{\mathrm{T}} = [\pi, 0]^{\mathrm{T}}$ where the position of pendulum is the bottom are given in Figs. 4 and 5, where for comparison, the results of the linear quadratic regulator (LQR) for the pendulum system (27) are shown together. As seen in these figures, the controller obtained by GPHDP swing up the pendulum to the top position and stabilize it with a smaller actuator input although the LQR directly lift up the pendulum with a bigger input. The finite horizon costs in (28) for 10 s by the controller of GPHDP and LQR are 58.42 and 253.36, respectively. This result shows that GPHDP could achieve the better performance in terms of the cost (28).

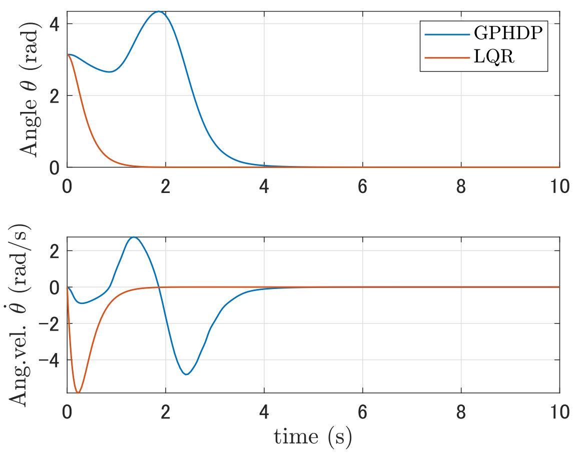

Fig. 4. The time history of the states of pendulum.

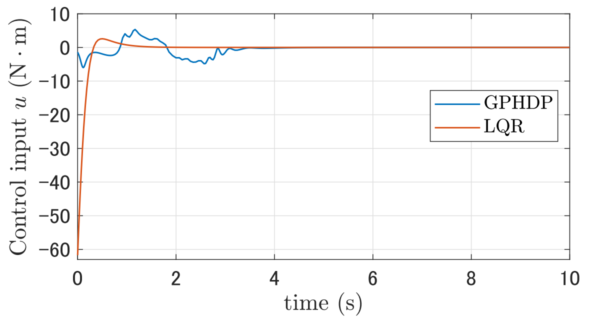

Fig. 5. The time history of the control input.

7. CONCLUSION

This paper has proposed a new algorithm, GPHDP, for realization of the HDP algorithm for CT nonlinear input-affine systems whose convergence is guaranteed. The convergence of the cost function and the controller obtained by GPHDP in the sense of the number of data points has been also proven under the reasonable assumption based on the GPs property. The effectiveness of the proposed method is demonstrated for the inverted pendulum.

ACKNOWLEDGEMENTS

This work was supported by JSPS KAKENHI Grant Number JP19J23306.

REFERENCES

Al-Tamimi, A., Lewis, F.L., and Abu-Khalaf, M. (2008). Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4), 943–949.

Bian, T., Jiang, Y., and Jiang, Z.P. (2014). Adaptive dynamic programming and optimal control of nonlinear nonaffine systems. *Automatica*, 50(10), 2624–2632.

Deisenroth, M.P., Rasmussen, C.E., and Peters, J. (2009). Gaussian process dynamic programming. *Neurocomputing*, 72(7), 1508–1524.

Lewis, F.L. and Vrabie, D. (2009). Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3), 32–50.

Murray, J.J., Cox, C.J., Lendaris, G.G., and Saeks, R. (2002). Adaptive dynamic programming. *Trans. Sys. Man Cyber Part C*, 32(2), 140–153.

Powell, W.B. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics)*. Wiley-Interscience.

Rasmussen, C.E. and Williams, C.K.I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Sutton, R.S. and Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Vamvoudakis, K.G. and Lewis, F.L. (2010). Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, 46(5), 878–888.

Vrabie, D., Pastravanu, O., Abu-Khalaf, M., and Lewis, F. (2009). Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica*, 45(2), 477–484.

Vrabie, D. and Lewis, F. (2009). Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Networks*, 22(3), 237–246.

Werbos, P. (1992). Approximate dynamic programming for realtime control and neural modelling. *Handbook of intelligent control: neural, fuzzy and adaptive approaches*, 493–525.

Wu, H.N. and Luo, B. (2012). Heuristic dynamic programming algorithm for optimal control design of linear continuous-time hyperbolic pde systems. *Industrial & Engineering Chemistry Research*, 51(27), 9310–9319.