

Acceleration of Gradient-based Path Integral Method for Efficient Optimal and Inverse Optimal Control

Masashi Okada¹ and Tadahiro Taniguchi^{1,2}

Abstract—This paper deals with a new accelerated path integral method, which iteratively searches optimal controls with a small number of iterations. This study is based on the recent observations that a path integral method for reinforcement learning can be interpreted as gradient descent. This observation also applies to an iterative path integral method for optimal control, which sets a convincing argument for utilizing various optimization methods for gradient descent, such as momentum-based acceleration, step-size adaptation and their combination. We introduce these types of methods to the path integral and demonstrate that momentum-based methods, like Nesterov Accelerated Gradient and Adam, can significantly improve the convergence rate to search for optimal controls in simulated control systems. We also demonstrate that the accelerated path integral could improve the performance on model predictive control for various vehicle navigation tasks. Finally, we represent this accelerated path integral method as a recurrent network, which is the accelerated version of the previously proposed path integral networks (PI-Net). We can train the accelerated PI-Net more efficiently for inverse optimal control with less RAM than the original PI-Net.

I. INTRODUCTION

In recent years, research on the path integral control framework, which originated from Kappen’s work [1], has significantly progressed. One of the most remarkable works in the area is Williams’s iterative path integral method [2], [3]. This method iteratively updates a control sequence to search for the optimal solution on the basis of importance sampling of trajectories. This approach solves the intrinsic problem of primitive path integral methods that require almost infinite samples for optimal solutions. Moreover, Williams derived different iterative methods [4], [5], which eliminate the affine dynamics constraints on the original path integral framework. This paper focuses on this type of iterative path integral methods.

Since path integral methods are derivative free, they possess several attractive features. First, as it is not necessary to approximate dynamics and cost models with linear and quadratic forms, non linear system dynamics and cost functions can be naturally employed. In [2], [5], highly non linear car dynamics have been employed to successfully control a miniature real vehicle that functions autonomously and with aggressive drifting. Dynamics can also be represented as trainable models, i.e., neural networks, thus allowing to solve model-based reinforcement learning tasks. Aggressive driving was also performed using the model-based reinforcement learning of neural dynamics [4].

Trainable cost models could be introduced to path integral methods; the problem would be how to train the parameterized cost models. This is known as inverse reinforcement learning or inverse optimal control problem. Okada proposed a solution introducing path integral networks (PI-Net) [6] which are recurrent networks to completely imitate the iterative path integral method. This network is differentiable with the parameters in the network and thus the network can be trained by back-propagation in the same way as done for standard neural networks. Inverse optimal control can be simply conducted by training PI-Net via imitation learning.

It has been claimed [3] that the convergence rate of the iterative method is fast enough to apply it to model predictive control (MPC [7]; a.k.a. receding horizon control), however, the convergence performance has not been clearly determined. On the other hand, the experiments of PI-Net in [6] required hundreds of iterations (or network recurrences) to obtain good training results, leading to massive RAM usage during back-propagation.

Based on the above, this paper discusses the convergence of iterative path integral methods, using the observations from recent work by Miyashita et al. [8]. In that report, it is shown that a path integral method for reinforcement learning (PI², Policy Improvement with Path Integral [9]) can be derived from a variant of a gradient descent method (i.e., mirror descent [10]), showing a connection between PI² and gradient descent. Accordingly, a connection between the iterative path integral method and gradient descent can be expected. In fact, as pointed out later in Sect. III, such connection can be derived using the same concepts as [8]. Considering the iterative path integral as gradient descent, this paper aims to accelerate the convergence of the iterative method utilizing optimization methods that were developed to accelerate the gradient descent.

The organization and contributions of this paper are summarized as follows. 1) Sect. III clarifies the relation between the iterative path integral method and gradient descent. It is shown that a new iterative path integral method is derived from the mirror descent search [8]. 2) In Sect. IV, we introduce optimization methods for gradient descent to accelerate the iterative method. We also discuss how the accelerated method is applied to MPC and PI-Net. 3) Finally, in Sect. V, we conduct simulated experiments of four dynamics systems. The experiment shows that the accelerated methods could dramatically accelerate the convergence compared to the original method. It is also shown that MPC and inverse optimal control benefit from the acceleration method.

¹ AI Solutions Center, Business Innovation Division, Panasonic Corporation okada.masashi001@jp.panasonic.com

² Ritsumeikan University, College of Information Science and Engineering taniguchi@em.ci.ritsumei.ac.jp

II. PRELIMINARIES

This section briefly reviews the formulation of stochastic optimal control problem, Williams's iterative path integral methods [2]–[5], and PI-Net [6].

A. Formulation of Stochastic Optimal Control

In this paper a discrete-time, continuous dynamics taking the following form is assumed:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \quad (1)$$

where $\mathbf{x}_t \in \mathbb{R}^n$ is the state of the system at time t , $\mathbf{u}_t \in \mathbb{R}^m$ is the control input at time t , and $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ denotes the state-transition function of the system. We assume that \mathbf{u}_t is a stochastic variable that takes the form:

$$\mathbf{u}_t = \boldsymbol{\mu}_t + \boldsymbol{\epsilon}_t, \quad (2)$$

where $\boldsymbol{\mu}_t \in \mathbb{R}^m$ is the deterministic variable and $\boldsymbol{\epsilon}_t \in \mathbb{R}^m$ is the stochastic variable which represents the system noise with zero mean, namely $\mathbb{E}[\mathbf{u}_t] = \boldsymbol{\mu}_t$. Given a finite time-horizon $t \in \{0, 1, 2, \dots, T-1\}$, state-action trajectory $\tau = \{\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \dots, \mathbf{u}_{T-1}, \mathbf{x}_T\}$ by (1) and trajectory cost $S(\tau) \in \mathbb{R}$, the objective of the stochastic optimal control problem is to find a control sequence $\boldsymbol{\mu}_{0:T-1}^* = (\boldsymbol{\mu}_0^*, \boldsymbol{\mu}_1^*, \dots, \boldsymbol{\mu}_{T-1}^*) \in \mathbb{R}^{m \times T}$ which minimize the objective function $J = \mathbb{E}[S(\tau)]$. The optimal problem is formulated as:

$$\boldsymbol{\mu}_{0:T-1}^* = \operatorname{argmin} J = \operatorname{argmin} \mathbb{E}[S(\tau)]. \quad (3)$$

In the following discussion, $\boldsymbol{\mu}_{0:T-1}$ is denoted as $\boldsymbol{\mu}$ for readability. We used same notation for $\mathbf{u} \equiv \mathbf{u}_{0:T-1} = (\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{T-1})$ and $\boldsymbol{\epsilon} \equiv \boldsymbol{\epsilon}_{0:T-1} = (\boldsymbol{\epsilon}_0, \boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_{T-1})$.

B. Iterative Path Integral Methods

The iterative path integral methods [2]–[5] are optimization methods for the stochastic optimal control problem. These methods assume that the system noise $\boldsymbol{\epsilon}_t$ is zero-mean Gaussian $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \Sigma)$ with a covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$, and suppose a trajectory cost function $S(\tau)$ as the sum of arbitrary state-cost and quadratic control-cost over time time-horizon:

$$\begin{aligned} S(\tau) &= S_{\mathbf{x}}(\tau) + S_{\mathbf{u}}(\tau), \\ S_{\mathbf{x}}(\tau) &= \phi(\mathbf{x}_T) + \sum_{t=0}^{T-1} q(\mathbf{x}_t), \\ S_{\mathbf{u}}(\tau) &= \sum_{t=0}^{T-1} \mathbf{u}_t^T R \mathbf{u}_t, \end{aligned} \quad (4)$$

where $\phi: \mathbb{R}^n \rightarrow \mathbb{R}$ and $q: \mathbb{R}^n \rightarrow \mathbb{R}$ are respectively terminal- and running-cost, and $R \in \mathbb{R}^{m \times m}$ is a weight matrix for the quadratic control cost.

These methods solve the optimal problem by iteratively updating $\boldsymbol{\mu}$ with the equation:

$$\begin{aligned} \boldsymbol{\mu}^{(j)} &= \boldsymbol{\mu}^{(j-1)} + \Delta \boldsymbol{\mu}^{(j-1)}, \\ \Delta \boldsymbol{\mu}^{(j-1)} &= \frac{\mathbb{E}_{p^{(j-1)}} \left[e^{-\tilde{S}(\tau)/\lambda} \boldsymbol{\epsilon} \right]}{\mathbb{E}_{p^{(j-1)}} \left[e^{-\tilde{S}(\tau)/\lambda} \right]}, \end{aligned} \quad (5)$$

where j is the update iteration index, $p^{(j-1)}$ is the probability density function of the stochastic variable \mathbf{u} , $\lambda \in \mathbb{R}^+$ is a hyper-parameter called the inverse temperature, and $\tilde{S}(\tau)$ is the modified trajectory cost function. For example, Williams et al. [2] define $\tilde{S}(\tau)$ as:

$$\tilde{S}(\tau) = S_{\mathbf{x}}(\tau) + \sum_{i=0}^{T-1} \boldsymbol{\mu}_i^T R \boldsymbol{\epsilon}_i. \quad (6)$$

Although different concepts are adopted to derive the methods in [3] and [4], [5]¹, they are practically equivalent and theoretically related. Specifically the method in [4] and [5] can exactly recover the method in [3] if dynamics is constrained to be affine in control. Eq. (5) can be implemented on digital computers by approximating the expectation value with the Monte Carlo integral as shown in Alg. 1, where K is the number of trajectories to be simulated for the approximation.

Algorithm 1 Iterative Path Integral Method [2]–[5]

input $\mathbf{x}_0, \boldsymbol{\mu}^{(j-1)}$: Input state & Control sequence
output $\boldsymbol{\mu}^{(j)}$: Improved control sequence
1: Sample $\{\boldsymbol{\epsilon}^{(0)}, \boldsymbol{\epsilon}^{(1)}, \dots, \boldsymbol{\epsilon}^{(K-1)}\}$
2: **for** $k \leftarrow 0$ **to** $K-1$ **do**
3: $\mathbf{x}_0^{(k)} \leftarrow \mathbf{x}_0$
4: $\tau^{(k)} \leftarrow \emptyset$
5: **for** $t \leftarrow 0$ **to** $T-1$ **do**
6: $\tau^{(k)} \leftarrow \tau^{(k)} \cup \{\mathbf{x}_t^{(k)}, \mathbf{u}_t^{(k)}\}$
7: $\mathbf{u}_t^{(k)} \leftarrow \boldsymbol{\mu}_t^{(j-1)} + \boldsymbol{\epsilon}_t^{(k)}$
8: $\mathbf{x}_{t+1}^{(k)} \leftarrow f(\mathbf{x}_t^{(k)}, \mathbf{u}_t^{(k)})$
9: **end for**
10: $\tau^{(k)} \leftarrow \tau^{(k)} \cup \{\mathbf{x}_T^{(k)}\}$
11: **end for**
12: $\Delta \boldsymbol{\mu}^{(j-1)} \leftarrow \sum_{k=0}^{K-1} e^{-\tilde{S}(\tau^{(k)})/\lambda} \boldsymbol{\epsilon}^{(k)} / \sum_{k=0}^{K-1} e^{-\tilde{S}(\tau^{(k)})/\lambda}$
13: $\boldsymbol{\mu}^{(j)} \leftarrow \boldsymbol{\mu}^{(j-1)} + \Delta \boldsymbol{\mu}^{(j-1)}$

Let U be the iteration numbers ($j \in \{0, 1, \dots, U-1\}$), the computational complexity of the iterative methods is $O(U \times T \times K)$. Since we can independently simulate K trajectories ($\ell 2$ – $\ell 11$ in Alg. 1), GPU parallelization is feasible and would significantly reduce the effect of K , achieving real-time MPC [3].

C. Path Integral Networks

Okada et al. [6] regarded the iterative executions of Alg. 1 as a *double-looped* recurrent network, termed PI-Net². The dynamics f and/or cost models $S(\tau)$ in the network can be represented as trainable models such as neural networks. Since the network is fully differentiable, we can optimize the internal model parameters by end-to-end training of PI-Net by back-propagation. This property of the network can be utilized for inverse optimal control to learn cost models latent

¹ Ref. [3] is based on the linearization of the Hamilton-Jacob Bellman equation and application of Feynman-Kac lemma, whereas [4], [5] adopt information theoretic approach using KL divergence and free energy.

² Please see Fig. 2 for an overview of the recurrent architecture of PI-Net. Although the figure shows the architecture of accelerated PI-Net introduced in Sect. IV, the difference is slight and removing the flows of $\Delta \boldsymbol{\mu}$ (also introduced later) makes it completely equivalent to the original.

in experts' demonstrations. Inverse optimal control with PI-Net is simply achieved by imitation learning, in which PI-Net is supervisedly trained so that the network output mimics the experts' demonstrations.

Let B be the mini-batch size for PI-Net training. Then the back-propagation requires a RAM size of $O(B \times U \times T \times K)$, which immediately grows, forcing to use massive of RAM. For instance in [6], hundreds of giga-bytes RAM were used even though they focused on rather simple tasks. Therefore, schemes to reduce the parameters while achieving good training results are necessary.

III. CONNECTION OF ITERATIVE PATH INTEGRAL TO GRADIENT DESCENT

This section shows the connection between iterative path integral methods and gradient descent. Relations between the iterative methods are also discussed.

A. Iterative Path Integral Method from Mirror Descent

Miyashita et al. employed the mirror descent [8] to policy search of parameterized model so as to maximize the expected cumulative reward, deriving PI². Since the derivation approach is very general, it is easily applied to optimal control search by regarding control sequence μ as policy parameters. The following brief derivation results in a different iterative path integral method. Details of the derivation concept can be found in [8], as the present procedure is based on that works.

We consider to optimize $p^{(j-1)}$ in order to minimize the objective function J by iteratively updating $p^{(j-1)}$ using the mirror descent:

$$p^{(j)} = \underset{p}{\operatorname{argmin}} \langle g, p \rangle + \frac{1}{\alpha} D_{\text{KL}}(p, p^{(j-1)}) + \beta \left(1 - \int dP \right), \quad (7)$$

where g is the gradient of the objective J with respect to p , $\langle \cdot, \cdot \rangle$ is the cross-product operator and α is a hyper-parameter corresponding to the step-size. β is the Lagrange multiplier for the constraint $\int dP = 1$ where P is the cumulative distribution of p . $D_{\text{KL}}(p, p^{(j-1)})$ is the KL divergence between two probability density function defined as:

$$D_{\text{KL}}(p, p^{(j-1)}) = \int \log \frac{p}{p^{(j-1)}} dP. \quad (8)$$

The gradient g can be calculated from:

$$g = \frac{\partial J}{\partial p} = \frac{1}{\partial p} \int p S(\tau) d\mathbf{u} = S(\tau) d\mathbf{u}, \quad (9)$$

then the inner-product term becomes:

$$\langle g, p \rangle = \int S(\tau) p d\mathbf{u} = \int S(\tau) dP. \quad (10)$$

Using the above relations, we can represent the argument of argmin in (7) as:

$$\int \left(S(\tau) + \frac{1}{\alpha} \log \frac{p}{p^{(j-1)}} - \beta \right) dP + \beta. \quad (11)$$

Organizing the above equation yields:

$$\frac{1}{\alpha} D_{\text{KL}} \left(p, p^{(j-1)} e^{\alpha(\beta - S(\tau))} \right) + \beta. \quad (12)$$

By minimizing this equation, the update law with respect to p is derived as:

$$p^{(j)} = p^{(j-1)} e^{\alpha(\beta - S(\tau))}. \quad (13)$$

The Lagrange multiplier β can be removed using the constraint $\int dP^{(j)} = 1$, then:

$$p^{(j)} = \frac{p^{(j-1)} e^{-\alpha S(\tau)}}{\int e^{-\alpha S(\tau)} dP^{(j-1)}} = \frac{p^{(j-1)} e^{-\alpha S(\tau)}}{\mathbb{E}_{p^{(j-1)}} [e^{-\alpha S(\tau)}]}. \quad (14)$$

Next, we derive the update law for μ from (14) with the relation $\int \mathbf{u} dP^{(j)} = \mu^{(j)}$. By multiplying both sides of (14) by $\mathbf{u} d\mathbf{u}$ and integrating both sides:

$$\mu^{(j)} = \frac{\int e^{-\alpha S(\tau)} \mathbf{u} dP^{(j-1)}}{\mathbb{E}_{p^{(j-1)}} [e^{-\alpha S(\tau)}]}. \quad (15)$$

Substituting (2) in (15) leads to:

$$\mu^{(j)} = \mu^{(j-1)} + \frac{\int e^{-\alpha S(\tau)} \epsilon dP^{(j-1)}}{\mathbb{E}_{p^{(j-1)}} [e^{-\alpha S(\tau)}]}, \quad (16)$$

$$\therefore \mu^{(j)} = \mu^{(j-1)} + \frac{\mathbb{E}_{p^{(j-1)}} [e^{-S(\tau)/\lambda} \epsilon]}{\mathbb{E}_{p^{(j-1)}} [e^{-S(\tau)/\lambda}]}. \quad (17)$$

At the final step, we replaced α with $1/\lambda$ to emphasize the similarity between the original methods and the newly derived method. If we replace $S(\tau)$ with $\tilde{S}(\tau)$, the derived and original methods are completely equivalent.

The convergence rate of the mirror descent is theoretically proved to be $O(1/j)$ [11] and this would be valid in the the derived method. Considering the similarity of methods, we also assume that the original method has the same convergence.

B. Differences Between the Iterative Path Integral Methods

As summarized in Appx. I, the original path integral method [4] is derived based on iterative importance sampling. It is interesting to note that different concepts (i.e., importance sampling and mirror descent) result in similar iterative methods. However, it may be said that gradient descent is a more adequate method for optimal solution search. In addition, the fact that the iterative method is gradient-based allows us to utilize a variety of optimization methods developed for gradient descent. Furthermore, the derivation with mirror descent did not use several assumptions that are necessary for the original methods. This is summarized in Table I, which also shows that the newly derived method is a generalization of the original ones. Since we mainly focused on the convergence performance, the generalized property will not be examined in this paper. The experiments in Sect. V consider the same assumptions as in references [4], [5].

IV. EMPLOYING OPTIMIZATION METHODS FOR GRADIENT DESCENT

This section introduces a variety of optimization methods for gradient descent in the iterative path integral method. We also describe the application of one method to MPC and PI-Net.

TABLE I
ASSUMPTIONS ON ITERATIVE PATH INTEGRAL METHODS

	[2], [3]	[4], [5]	Derived
Dynamics	Control Affine	Arbitrary	Arbitrary
Traj. Cost	Eq. (4)	Eq. (4)	Arbitrary
Sys. Noise	Gaussian	Gaussian	Arbitrary

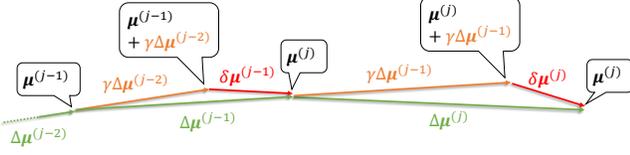


Fig. 1. Nesterov Accelerated Gradient: accumulation of past momenta accelerate the convergence.

1) *Nesterov Accelerated Gradient (NAG)* [12]: NAG utilizes the past update (or momentum); in the present work $\Delta\mu^{(j-2)} = \mu^{(j-1)} - \mu^{(j-2)}$. In this method: (1) a current solution is *drifted* by the momentum and then (2) the drifted solution is *slided* by the gradient. The iterative path integral method can simply adopt this concept. Let us consider a probability density function $p_m^{(j-1)}$ in which the mean value is drifted from $\mu^{(j-1)}$ by the momentum $\Delta\mu^{(j-2)}$:

$$\mathbb{E}_{p_m^{(j-1)}}[\mathbf{u}] = \mu^{(j-1)} + \gamma\Delta\mu^{(j-2)}, \quad (18)$$

where $\gamma(<1)$ is a decay parameter. By replacing $p^{(j-1)}$ in Eq. (14) to $p_m^{(j-1)}$, we get an update law considering the momentum:

$$\begin{aligned} \mu^{(j)} &= \mu^{(j-1)} + \Delta\mu^{(j-1)}, \\ \Delta\mu^{(j-1)} &= \gamma\Delta\mu^{(j-2)} + \delta\mu^{(j-1)}, \\ \delta\mu^{(j-1)} &= \frac{\mathbb{E}_{p_m^{(j-1)}}[e^{-S(\tau)/\lambda}\epsilon]}{\mathbb{E}_{p_m^{(j-1)}}[e^{-S(\tau)/\lambda}]}. \end{aligned} \quad (19)$$

After a few modifications of Alg. 1, the implementation of (19) is achieved, as shown in Alg. 2 where modified or added lines are highlighted. Fig. 1 shows an intuitive illustration of this approach.

2) *AdaGrad* [13]: AdaGrad adapts step-sizes considering the accumulation of past gradients. To apply this concept to our case, we modify the top equation of (19):

$$\mu^{(j)} = \mu^{(j-1)} + \eta^{(j-1)} \circ \Delta\mu^{(j-1)}, \quad (20)$$

where $\eta^{(j-1)} \in \mathbb{R}^{m \times T}$ is the newly introduced step size vector, and “ \circ ” indicates the element-wise product. Starting with $\eta^{(-1)} = \mathbf{1}$, $\eta^{(j-1)}$ is adapted by the update law of AdaGrad.

3) *Adam* [14]: Adam is the combination of the momentum-based acceleration and step-size adaptation. Our implementation uses the same equations and hyper-parameters proposed in the reference.

The above introductions are rather heuristic and we will not discuss their effect on the convergence from a theoretical perspective. Contrary, another accelerated mirror descent method that assures the rate of $O(1/j^2)$ is theoretically

derived [8], [11]. However, in this study, we did not employ said method because some equations in it cannot be represented as a closed-form. Therefore, additional iterative algorithms [15] must be introduced into the iterative path integral method to solve the equations, making it difficult to apply this method to real-time MPC and PI-Net.

Algorithm 2 NAG-Accelerated Path Integral Method

input $\mathbf{x}_0, \mu^{(j-1)}$: Input state & Control sequence
 $\Delta\mu^{(j-2)}$: Momentum
output $\mu^{(j)}$: Improved control sequence
 $\Delta\mu^{(j-1)}$: Momentum

- 1: Sample $\{\epsilon^{(0)}, \epsilon^{(1)}, \dots, \epsilon^{(K-1)}\}$
- 2: **for** $k \leftarrow 0$ **to** $K-1$ **do**
- 3: $\mathbf{x}_0^{(k)} \leftarrow \mathbf{x}_0$
- 4: $\tau^{(k)} \leftarrow \emptyset$
- 5: **for** $t \leftarrow 0$ **to** $T-1$ **do**
- 6: $\tau^{(k)} \leftarrow \tau^{(k)} \cup \{\mathbf{x}_{t+1}^{(k)}, \mathbf{u}_t^{(k)}\}$
- 7: $\mathbf{u}_t^{(k)} \leftarrow \mu_t^{(j-1)} + \gamma\Delta\mu_t^{(j-2)} + \epsilon_t^{(k)}$
- 8: $\mathbf{x}_{t+1}^{(k)} \leftarrow f(\mathbf{x}_t^{(k)}, \mathbf{u}_t^{(k)})$
- 9: **end for**
- 10: $\tau^{(k)} \leftarrow \tau^{(k)} \cap \{\mathbf{x}_T^{(k)}\}$
- 11: **end for**
- 12: $\delta\mu^{(j-1)} \leftarrow \frac{\sum_{k=0}^{K-1} e^{-S(\tau^{(k)})/\lambda} \epsilon^{(k)}}{\sum_{k=0}^{K-1} e^{-S(\tau^{(k)})/\lambda}}$
- 13: $\Delta\mu^{(j-1)} \leftarrow \gamma\Delta\mu^{(j-2)} + \delta\mu^{(j-1)}$
- 14: $\mu^{(j)} \leftarrow \mu^{(j-1)} + \Delta\mu^{(j-1)}$

A. Application to Model Predictive Control

This section and the next focus on the NAG accelerated path integral method and exemplify its applications to MPC and PI-Net.

Alg. 3 shows the procedure of MPC using the accelerated path integral. In $\ell 7-8$, not only the control sequence $\mu_{1:T-1}$, but also the momentum $\Delta\mu_{1:T-1}$ are retained for the next step to *warm start* the optimization. In our implementation, $\ell 9$ is performed as $\mu_{T-1} \leftarrow \mu_{T-2}$ and $\Delta\mu_{T-1} \leftarrow \Delta\mu_{T-2}$.

Algorithm 3 MPC with NAG-Accelerated Path Integral

input $\mu, \Delta\mu$: Initial control sequence & Momentum

- 1: **while** True **do**
- 2: Observe current state \mathbf{x}_t
- 3: **for** $j \leftarrow 0$ **to** U **do**
- 4: $(\mu, \Delta\mu) \leftarrow$ Execute Alg. 2 inputting $(\mathbf{x}_t, \mu, \Delta\mu)$
- 5: **end for**
- 6: Send μ_0 to actuators
- 7: $\mu_{0:T-2} \leftarrow \mu_{1:T-1}$
- 8: $\Delta\mu_{0:T-2} \leftarrow \Delta\mu_{1:T-1}$
- 9: Initialize $\mu_{T-1}, \Delta\mu_{T-1}$
- 10: **end while**

B. Application to PI-Net and Inverse Optimal Control

Fig. 2 illustrates the network representation of Alg. 2. The operations of this network are essentially the same than in the original PI-Net described in reference [6], except for the flows of the momentum $\Delta\mu$, which are newly added. Thus we omit a detailed explanation of how this network operates. This network is certainly differentiable and trained

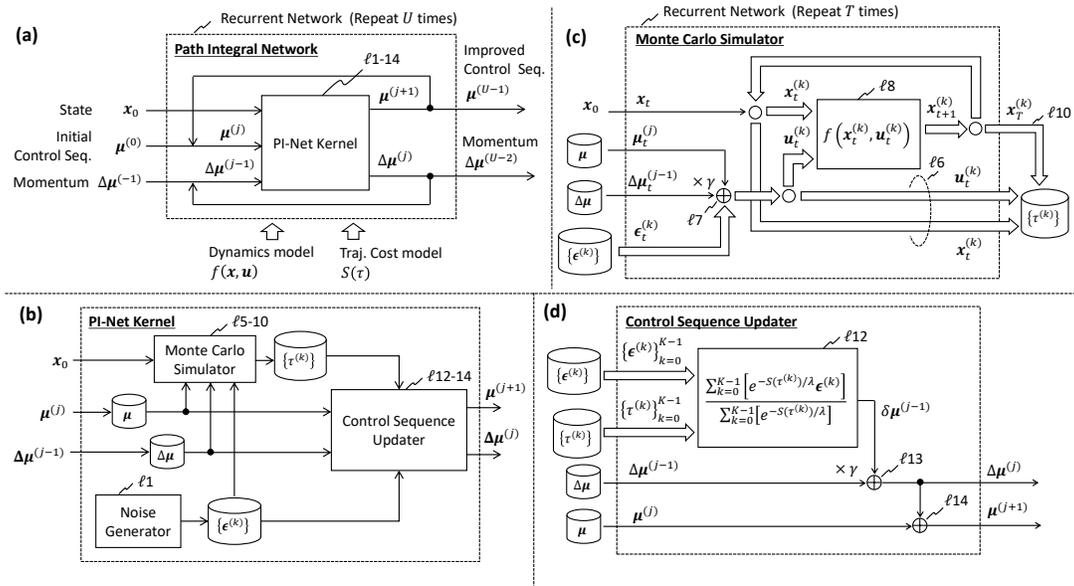


Fig. 2. Architecture of the accelerated PI-Net. ‘ ℓ ’ labels indicate the corresponding line numbers in Alg. 2. Block arrows in (c,d) indicate multiple signal flow with respect to K trajectories.

by standard back-propagation. We can hence utilize the same training schemes than those of the original PI-Net to carry out the inverse optimal control.

V. EXPERIMENTS

This section presents several experiments that we conducted under simulated settings to examine the performance of the accelerated path integral method on convergence, MPC, and inverse optimal control.

A. Convergence Rate Comparison

The objective of this experiment was to evaluate the convergence performance of the iterative path integral combined with the gradient methods from Sect. IV. We also compared the path integral method with following methods: 1) Differential Dynamic Programming (DDP) [16] as a reference to demonstrate the validity of our method and its implementation, and 2) the original iterative path integral method [5] as a baseline to examine how adapting gradient methods could improve the performance.

We focused on four non linear simulated dynamics systems: inverted pendulum [17], and miniature vehicle systems (hovercraft [18], quadrotor [19] and car [20]). The aims of the control tasks are as follows: to swing the pendulum up (inverted pendulum), to navigate the vehicles towards target positions (hovercraft and quadrotor) and to navigate the vehicle to go around an oval track at a desired speed. Fig. 3 illustrates the systems and their control tasks. The cost functions that encode the above tasks are summarized in Appx. II. Parameters of the path integral method are commonly set as $\lambda = 0.01$ and $K = 1000$. Initial control sequence and momentum were initialized as $\mu^{(0)} = \mathbf{0}$ and $\Delta\mu^{(-1)} = \mathbf{0}$. The decay parameter of NAG was set as $\gamma = 0.8$.

With these conditions, we observed the convergence performance via optimization of the cost functions. Fig. 4 summarizes the results. In this figure, NAG-accelerated path integral achieved faster convergence than the baseline for all control tasks. In addition, the converged cost values are lower than the reference cost by DDP for the three navigation tasks. Although Adam resulted in unstable convergence in the quadrotor task, it achieved much faster convergence than NAG for other tasks particularly for the car navigation task. In this experiment, AdaGrad showed no improvement from the baseline.

B. Model Predictive Control

We conducted MPC simulations of the three vehicle systems with Alg. 3 and a baseline method (i.e., MPC with the original path integral [5]). The cost functions mentioned above were also used in this section. In the simulations of the hovercraft and quadrotor, a target position was changed when tasks were completed (i.e., vehicle reached to the target). Path integral parameters were set as $(U, K, \gamma) = (10, 100, 0.8)$ for the hovercraft and quadrotor navigation tasks and $(U, K, \gamma) = (25, 200, 0.3)$ for the car navigation task. The results of these simulations are summarized in Fig. 5 and Table II in which our accelerated method successfully improved control performances especially for the hovercraft task.

C. Inverse Optimal Control

Inverse optimal control was conducted with both the original PI-Net [6] and the accelerated one. We focused on the pendulum swing-up task described above. We considered DDP with the given dynamics and cost models as an expert and generated demonstrations in the same manner as in reference [6]. From the demonstrations, we prepared two dataset \mathcal{D}_{train} and \mathcal{D}_{test} , each of which was used for training

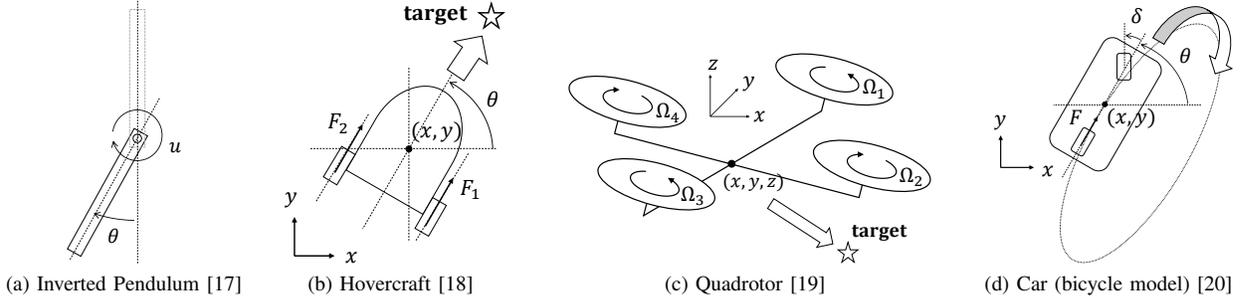


Fig. 3. Sketches of the systems and their control tasks. States of each system comprises positions, orientations and their time-derivatives. The following are also regarded as states: current thruster outputs $F_{1,2}$ (hovercraft), current rotor frequencies $\Omega = \Omega_{1,2,3,4}$ (quadrotor), and current steering angle δ and rear-tire force F_r (car). Control inputs are torque u (inverted-pendulum) and the desired values of $F_{1,2}$, Ω , (δ, F_r) .

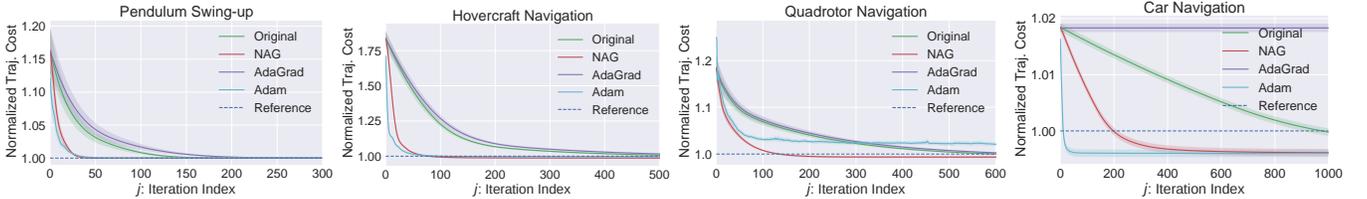


Fig. 4. Results of the experiment on convergence performance from Sect. V-A. For each setting, we conducted 100 optimizations with different state inputs and the results represent the average. In order to verify the optimization results, we also show reference cost values obtained from DDP. The cost values were normalized (references = 1)

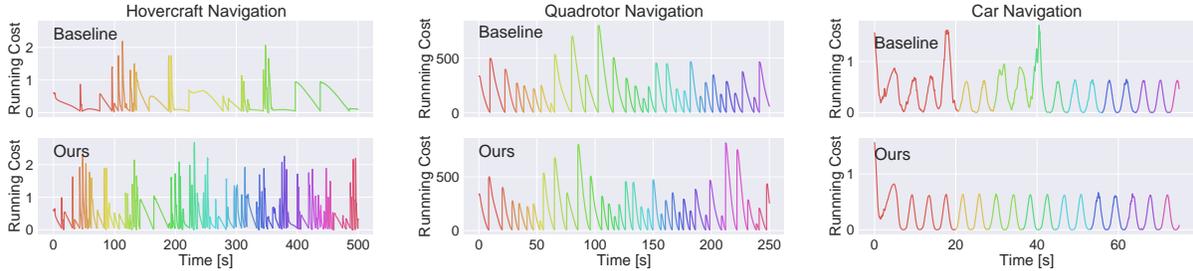


Fig. 5. Time transition of running cost $q(\mathbf{x})$ during the simulations of vehicle navigations. For clarity purposes, these figures focus only on the first seconds of the entire simulations. Changes of color indicate task completions; i.e., faster color change means faster task completions.

TABLE II
SUMMARY OF MPC SIMULATION RESULTS.

	Hovercraft		Quadrotor		Car	
	Baseline [5]	Ours	Baseline [5]	Ours	Baseline [5]	Ours
Number of Task Completions (# Reached-Targets or # Laps)	68	220	87	108	8	8
Mean Time to Task Completion [s]	17.93	5.700	8.713	7.004	9.094	8.784
Mean Accumulated-cost to Task Completion	237.5	91.92	5.386×10^4	3.849×10^4	132.8	91.86

or test. The dataset takes the form $(\mathbf{x}_t^*, \mu_t^*) \in \mathcal{D}$ where \mathbf{x}_t^* is a state input and μ_t^* is the corresponding control by the expert.

The dynamics in PI-Net was same as with the expert and we represent the cost model as a neural network, which had a single hidden layer with 12 hidden nodes and arctangent activation functions. Then, PI-Net was trained by optimizing the mean squared errors (MSE) between the PI-Net output $\mu_0^{(U-1)}$ and the expert control μ_t^* so that the neural cost approximates the true cost. Common parameters were set as $K = 100$, $U \in \{50, 100, 200\}$. The decay parameter γ was set to 0.8.

Fig. 6 shows the convergence of the MSEs during training, where the small number of iterations (i.e. $U = \{50, 100\}$) impeded sufficient convergence for the baseline. In contrast, accelerated PI-Net results show efficient convergences for all Us . This result contributes to significantly reduce the computational complexity for back-propagation as summarized in Table III.

Table. IV summarizes the training and test errors between the reference and the output of trained PI-Net with different Us . The accelerated PI-Net leads to smaller errors when $U \in \{50, 100\}$, however, the baseline with no acceleration shows better results for $U = 200$. We suppose this results

TABLE III
RAM USAGE AND COMPUTATIONAL TIME FOR PI-NET TRAINING

U	50	100	200
RAM [GB]	32.5	49.9	81.3
Time per Epoch [s]	335	653	1318

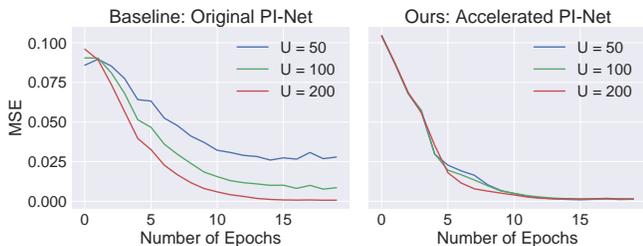


Fig. 6. Convergence of MSE w.r.t. \mathcal{D}_{train} during PI-Net training.

from the side effect of the momentum-based acceleration; the accumulated past momenta interfered with *fine* optimization. We believe that carefully scheduling the decay parameter γ over U iterations can alleviate this. Fig. 7 illustrates the learned cost model by the accelerated PI-Net ($U = 200$).

VI. CONCLUSIONS

In this work, we developed a gradient-based and accelerated iterative path integral optimal control method. This work is greatly inspired by the mirror descent search from Miyashita et al. [8], which is such a powerful and general framework that one can directly apply it to the stochastic optimal control problem, thus deriving a gradient-based iterative path integral method. Given the relation between the path integral and gradient descent, we could introduce optimization methods used for gradient descent into the path integral. The simulated experiments showed that momentum-based methods (i.e., NAG and Adam) could significantly accelerate the convergence on optimal control search. We also applied the NAG-accelerated path integral method to MPC and PI-Net, demonstrating the improvement of control performance and the efficiency on PI-Net training for inverse optimal control.

The results of this work suggest several directions for future research. First, although the path integral method derived in this paper is less constrained than the original methods, we did not analyze this in detail. This generalized property allows us to consider non-Gaussian settings and to design more expressive trajectory cost functions $S(\tau)$. We will further examine the effectiveness in order to broaden the applicability in practical systems.

Moreover, the causes of Adam’s instability and AdaGrad’s poor performance are not fully understood. A suggestion from this paper is to utilize NAG because it is simple to apply and showed reliable acceleration for all the cases in our test. However, further examination and understanding of these optimization methods will yield faster convergence. In addition, the state-of-the-art optimization methods developed in the deep learning field, such as [21], can be good solutions

TABLE IV
MSE B/W TRAINED PI-NET OUTPUTS AND THE EXPERT DEMONSTRATIONS.

	U	Baseline [6]	Ours
\mathcal{D}_{train}	50	2.736×10^{-2}	1.234×10^{-3}
	100	8.775×10^{-3}	1.670×10^{-3}
	200	6.625×10^{-4}	1.720×10^{-3}
\mathcal{D}_{test}	50	1.788×10^{-2}	1.613×10^{-3}
	100	4.523×10^{-3}	1.302×10^{-3}
	200	4.630×10^{-4}	1.340×10^{-3}

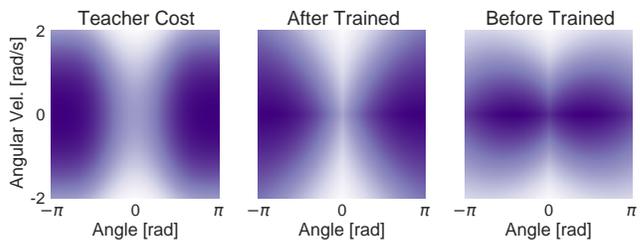


Fig. 7. Visualized cost map from PI-Net training.

for further improvement.

Finally, constructing real-hardware demonstrations of MPC and inverse optimal control, utilizing our accelerated method, are on-going work.

REFERENCES

- [1] H. J. Kappen, “Linear theory for control of nonlinear stochastic systems,” *Phys. Rev. Lett.*, vol. 95, no. 20, 2005.
- [2] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, “Aggressive driving with model predictive path integral control,” in *ICRA*, 2016.
- [3] G. Williams, A. Aldrich, and E. A. Theodorou, “Model predictive path integral control: From theory to parallel computation,” *J. Guid. Control Dyn.*, 2017.
- [4] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. Rehg, B. Boots, and E. Theodorou, “Information theoretic MPC for model-based reinforcement learning,” in *ICRA*, 2017.
- [5] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, “Information theoretic model predictive control: Theory and applications to autonomous driving,” *arXiv:1707.02342*, 2017.
- [6] M. Okada, L. Rigazio, and T. Aoshima, “Path integral networks: End-to-end differentiable optimal control,” *arXiv:1706.09597*, 2017.
- [7] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.
- [8] M. Miyashita, S. Yano, and T. Kondo, “Mirror descent search and acceleration,” *arXiv:1709.02535*, 2017.
- [9] E. Theodorou, J. Buchli, and S. Schaal, “A generalized path integral control approach to reinforcement learning,” *JMLR*, vol. 11, 2010.
- [10] S. Bubeck et al., “Convex optimization: Algorithms and complexity,” *Foundations and Trends® in Machine Learning*, 2015.
- [11] W. Krichene, A. Bayen, and P. L. Bartlett, “Accelerated mirror descent in continuous and discrete time,” in *NIPS*, 2015.
- [12] Y. Nesterov, “A method of solving a convex programming problem with convergence rate $O(1/k^2)$,” *Soviet Mathematics Doklady*, 1983.
- [13] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *JMLR*, 2011.
- [14] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [15] W. Krichene, S. Krichene, and A. Bayen, “Efficient bregman projections onto the simplex,” in *CDC*, 2015.

- [16] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *ACC*, 2005.
- [17] H. O. Wang, K. Tanaka, and M. F. Griffin, "An approach to fuzzy control of nonlinear systems: Stability and design issues," *IEEE Trans. Fuzzy Syst.*, vol. 4, no. 1, 1996.
- [18] H. Seguchi and T. Ohtsuka, "Nonlinear receding horizon control of an underactuated hovercraft," *Int. J. Robust Nonlin*, 2003.
- [19] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro UAV testbed," *IEEE Robot. Autom. Mag.*, vol. 17, no. 3, 2010.
- [20] J. Gonzales, F. Zhang, K. Li, and F. Borrelli, "Autonomous drifting with onboard sensors," in *International Symposium on Advanced Vehicle Control*, 2016.
- [21] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *NIPS*, 2016.

APPENDIX I DERIVATION OF THE ORIGINAL ITERATIVE PATH INTEGRAL METHOD

Let p^* be the optimal probability density with μ^* . We also define p_0 as the density of uncontrolled sequence $\mu = \mathbf{0}$. In [4], the relation of these distributions are proved to be:

$$p^* = \frac{p_0 e^{-S_{\mathbf{x}}(\tau)/\lambda}}{\mathbb{E}_{p_0} [e^{-S_{\mathbf{x}}(\tau)/\lambda}]}. \quad (21)$$

Applying the similar approach with (15), (16) results in the closed-form optimal solution: $\mu^* = \mathbb{E}_{p_0} [e^{-S_{\mathbf{x}}(\tau)/\lambda} \epsilon] / \mathbb{E}_{p_0} [e^{-S_{\mathbf{x}}(\tau)/\lambda}]$. However this solution requires sampling of trajectories from the zero-mean distribution p_0 . Since this kind of sampling is inefficient requiring almost infinite samples, Ref. [4] alternatively employs iterative importance sampling from $p^{(j-1)}$:

$$\begin{aligned} p^{(j)} &= \frac{p^{(j-1)}/p^{(j-1)}}{p^{(j-1)}/p^{(j-1)}} \cdot \frac{p_0 e^{-S_{\mathbf{x}}(\tau)/\lambda}}{\mathbb{E}_{p_0} [e^{-S_{\mathbf{x}}(\tau)/\lambda}]} \\ &= \frac{p^{(j-1)} \cdot [(p_0/p^{(j-1)}) \cdot e^{-S_{\mathbf{x}}(\tau)/\lambda}]}{\mathbb{E}_{p^{(j-1)}} [(p_0/p^{(j-1)}) \cdot e^{-S_{\mathbf{x}}(\tau)/\lambda}]}. \end{aligned} \quad (22)$$

Considering that p is Gaussian, we can represent the likelihood ratio $p_0/p^{(j-1)}$ as exponential form. Then integrating (22) yields the original iterative method (5), (6). We note that $R = \lambda \Sigma^{-1}/2$ is supposed in the derivation.

APPENDIX II COST FUNCTIONS

This section summarizes the cost functions used in the experiment. For all tasks, terminal cost was defined as same with running cost: $\phi(\mathbf{x}) = q(\mathbf{x})$.

a) Pendulum Swing-up:

$$\begin{aligned} q(\mathbf{x}) &= (1 + \cos \theta)^2 + \dot{\theta}^2, \\ R &= 5, \end{aligned} \quad (23)$$

where θ is the angle of the pendulum.

b) Hovercraft Navigation:

$$\begin{aligned} q(\mathbf{x}) &= h(d, w_d) + h(v, w_v) + h(\cos \theta_d - 1, w_\theta) \\ &\quad + w_F \cdot (F_1^2 + F_2^2), \\ R &= O, \end{aligned} \quad (24)$$

where,

$$h(x, w) = \sqrt{x^2 + w^2} - w, \quad (25)$$

d and θ_d are the distance and angular differences between current state and target state respectively, v is the velocity, and $(w_d, w_v, w_\theta, w_F) = (10^{-6}, 10^{-2}, 1, 0.2)$.

c) Quadrotor Navigation:

$$\begin{aligned} q(\mathbf{x}) &= h(\Delta x, w_x) + h(\Delta y, w_y) + h(\Delta z, w_z) \\ &\quad + w_v \cdot v^2 + w_q \cdot \|\Delta \mathbf{q}\| + w_\omega \cdot \|\boldsymbol{\omega}\| + w_\Omega \cdot \|\boldsymbol{\Omega}\|, \\ R &= O, \end{aligned} \quad (26)$$

where $\Delta x, \Delta y, \Delta z$ are differences between current positions and target positions, v is the velocity, $\Delta \mathbf{q}$ is the difference between current orientation and desired orientation, $\boldsymbol{\omega}$ is the angular velocity and $(w_{x,y,z}, w_v, w_q, w_\omega, w_\Omega) = (50, 10, 200, 10^{-3}, 10^{-6})$.

d) Car Navigation:

$$\begin{aligned} q(\mathbf{x}) &= 100 \left\{ \left(\frac{x}{2} \right)^2 + y^2 - 1 \right\}^2 + (v_x - 1.25)^2, \\ R &= O, \end{aligned} \quad (27)$$

where v_x is the forward velocity of the car.

ACKNOWLEDGMENT

The authors would like to thank Shiro Yano and Megumi Miyashita for insightful discussion.