Efficient reinforcement learning for fluid flow control

Akira Kubo and Masaki Shimizu

Graduated School of Engineering Science, Osaka University, Toyonaka, 560-0043 Japan

(Dated: December 9, 2020)

Despite the low dimensionalities of dissipative viscous fluids, reinforcement learning (RL) requires many observables in fluid control problems. This is because the observables are assumed to follow a policy-independent Markov decision process in the RL framework. By including policy parameters as arguments of a value function, we construct a consistent algorithm with partial observables. Using typical examples of fluid control around a cylinder, we show that our algorithm is more stable and efficient than the existing RL algorithms, even under a small number of observations.

INTRODUCTION

Nowadays, data-driven modelings based on machine learning are more practical than traditional ones based on theories in various applications including fluid engineering. Machine learning is used to model, control, and optimize flows from the vast amount of data obtained from experiments and simulations. For a detailed review of machine-learning applications in fluid mechanics, refer to Brunton et al. (2020)\(^1\).

Reinforcement learning\(^2\) is one of machine learning techniques. Its algorithm finds the optimal control method that maximizes the total rewards in a given system. The learning algorithms adopt trial and error approach to determine the optimum control from the time-series of states, actions, and rewards. Recently, deep reinforcement learning\(^3\), the combination with deep learning, has achieved unprecedented performance in artificial intelligence. The first successful examples of applications include Go, Shogi, and other video games. These systems significantly outperform the best human players.\(^4\) Deep reinforcement learning has also been applied to autonomous driving.\(^5\)

Previous studies have investigated the applicability of reinforcement learning in the active control of fluids. Value-based reinforcement learning with Q-learning or SARSA has been applied in school movement optimization to reduce drag around cylinders\(^6,7\), and optimal glider control\(^8,9\). In recent years, deep reinforcement learning based on the actor-critic algorithm has been applied to flow control problems. Koizumi et al. (2018)\(^10\) used the deterministic actor-critic method to reduce drag around a cylinder optimally through blowing and suction control.\(^12\) Verma et al. (2018)\(^13\) combined deep long short-term memory with reinforcement learning to investigate the mechanism of fish school formation.\(^13\) Moreover, a series of research applies the deep reinforcement learning package Tensorforce to optimal control through blowing and suction.\(^14,17\)

Although fluid systems have infinite degrees of freedom, the number of observations is limited in actual control. In a dissipative system, such as a viscous fluid, the system's final state (attractor) requires only a finite number of observables to determine the state.\(^18\) Since these finite observables can describe the time evolution, their time-series has Markov property. Generally, reinforcement learning algorithms assume the time evolution as a Markov process that does not depend on the way of control (policy). However, this assumption fails when learning is performed with a small number of observable and many policy parameters such as in deep neural networks. In this paper, we describe a framework for consistent reinforcement learning that optimally controls fluid with a small number of observable, and adapt it to the typical problems of optimizing blowing/suction and propulsion direction around a cylinder.

ALGORITHM

Deterministic Actor-Critic Algorithm

In this section, the outline of the existing reinforcement learning algorithm is given only for the on-policy deterministic actor-critic method\(^19\) to simplify the analysis. To the best of our knowledge, all previous studies using deep reinforcement learning for fluid control are based on the actor-critic method. Here, we assume that our system evolves deterministically, and control is determined as a function of the state. These assumptions may be sufficient for fluid control. If exploration is to be considered, the algorithm can be changed to an off-policy type, according to the approach described by Silver et al. (2014)\(^20\).

We consider a discrete dynamical system given by the Map $F$:

$$s_{t+1} = F(s_t, a_t).$$

(1)

Here, $s_t$ is the state of the system at time $t$, and $a_t$ is the control to the system (action). In the deterministic policy framework the action is a function of a state, $a_t = \mu(s_t)$, which represents the policy. Generally, a policy is approximated by a parametric function such as $\mu(s) \approx \tilde{\mu}(s; \theta)$. Here, the parameters $\theta$ are called design variables of the objective function defined below. Hereafter, $\tilde{\cdot}$ represents a parametric function. Let the reward
be \( r_t = R(s_t, a_t) \). Then, we define the value function \( V \) and the objective function \( J \) as follows:

\[
V(s; \theta) = \sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t) |_{s_0 = s},
\]

\[
J(\theta) = \mathbb{E}(V|\theta) = \int_{s} \rho(s; \theta) V(s; \theta) ds.
\]

Here, \( \gamma \) is a discount rate and satisfying \( 0 \leq \gamma < 1 \), and \( \rho \) represents the probability density in the state space. Optimum control determines the optimum design parameters \( \theta^* \) that maximize the objective function as

\[
\theta^* = \arg \max_{\theta} J(\theta),
\]

\[
\mu^*(s) = \tilde{\mu}(s; \theta^*).
\]

When the time-series of the state is a stationary process such as an attractor in a dissipative system, \( \theta^* \) is independent of \( \gamma \). This is because \( \gamma \) changes \( J \) only by a constant multiple. In the actor-critic method, the objective function \( J \) is optimized via the action-value function \( Q \) as

\[
Q(s, a; \theta) = \sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t) |_{s_0 = s, a_0 = a}.
\]

As for the policy, the action-value function is approximated by the parametric function of \( \omega \) such that
\[
Q(s, a; \theta) \simeq \tilde{Q}(s, a; \theta, \omega).
\]

At the end of every episode, the critic updates \( \tilde{Q} \), following the SARSA method, as

\[
\delta_t = r_{t+1} + \gamma \tilde{Q}(s_{t+1}, a_{t+1}; \theta_n, \omega_n) - \tilde{Q}(s_t, a_t; \theta_n, \omega_n)
\]

\[
\omega_{n+1} = \omega_n + \alpha_n \sum_{t \in T_n} \delta_t \nabla_\omega \tilde{Q}(s_t, a_t; \theta_n, \omega_n)
\]

Here, the subscript \( n \) represents the episode number, and \( T_n = \{t_{n1}, t_{n2}, \cdots, t_{nN}\} \) represents the set of \( N \) samples at episode \( n \). On the other hand, the actor follows the deterministic policy gradient theorem \[10\] given by

\[
\nabla_\theta J(\theta) = \int_{s} \rho(s; \theta) \nabla_\theta \tilde{\mu}(s; \theta) \nabla_a \tilde{Q}(s, a; \theta) |_{a = \tilde{\mu}(s; \theta)} ds,
\]

and updates the design variables by the stochastic gradient as

\[
\theta_{n+1} = \theta_n + \alpha_n \sum_{t \in T_n} \nabla_\theta \tilde{\mu}(s_t; \theta_n) \nabla_a \tilde{Q}(s_t, a_t; \theta_n).
\]

Here, \( \alpha \) is the learning rate. The actor-critic method determines the optimal policy by repeatedly updating the action value function \( Q \) and the policy \( \tilde{\mu} \) according to equations \( \text{[8]} \) and \( \text{[10]} \) respectively.

**Algorithm for Fluid Systems**

In a dissipative system, such as a fluid, the solution trajectory of the final state of the system is often in a much lower dimensional space (attractor) than that of the system. Assuming that the dimension of the attractor is \( D_n \), the state can be specified in most cases with \( 2[D_n] + 1 \) observables. \([18]\) For example, we can determine the state of a microorganism that swims in water with periodic motion using only three variables. Let \( o \) be observables sufficient to determine the state of the attractor. The transition map is defined by the equation \( \text{[11]} \) only when all the variables \( s \) are observed. The equation \( \text{[11]} \) is not directly dependent on \( \theta \). Contrarily, the policy \( \theta \) independence of the evolution of a system is a necessary condition of the policy gradient theorem. \([19, 21]\) Thus, even if the attractor is low-dimensional, many observations are required to update a policy appropriately when using typical reinforcement learning algorithms for fluid control problems. In the following discussion, we construct the consistent algorithm with partial observables.

The value function can be determined with the minimum number of arguments by specifying \( o \) and \( \theta \). Therefore, there exists \( \tilde{V} \) such that

\[
V(s; \theta) = \tilde{V}(o; \theta).
\]

We approximate \( \tilde{V} \) as a parametric function for \( v \) and \( w \) as,

\[
\hat{V}(o; \theta) \simeq \tilde{V}(o; \theta; w) = w^T \phi(o, \theta).
\]

Here, \( \phi \) is a nonlinear function of \( o \) and \( \theta \), which are constructed from the time-series data. We assume that the policy is a linear function of the observable \( o \):

\[
\mu(o; \theta) = \theta^T o.
\]

Arbitrary control becomes possible by increasing the number of observables. Otherwise, the use of nonlinear basis functions enables complex control.

In our proposed algorithm, the critic and the actor work as follows. The critic updates \( \tilde{V} \) every \( N_s \) steps of a numerical simulation. The unit of this updating is called episode. The weight \( w_n \) of \( \tilde{V} \) at the end of \( n \) episodes is calculated as follows:

\[
\delta_t = \sum_{k=0}^{N_s-1} \gamma^k r_{t+k} + \gamma^{N_s} \tilde{V}(o_{t+N_s}; \theta_{t+N_s}) - \tilde{V}(o_t, \theta_t),
\]

\[
w_n = \arg \min w \sum_{t \in T_n} \delta_t^2.
\]

Here, \( \delta_t \) represents the \( N_s \) step TD error. \( T_n = \{ (n-M)N_c, (n-M)N_c + 1, \cdots, nN_c - 1 \} \) is the set of steps
in the last $M$ episodes. The basis functions $\phi$ are constructed from the time series $\{(o_t, \theta_t) | t \in T_n\}$. We use RBF sampler \cite{21} of the package scikit-learn \cite{22} for the basis functions. Then, the actor calculates the stochastic gradient $g_n$ at the $n$th episode by the linear regression algorithm:

$$g_n, b_n = \arg \min_{g,b} \sum_{t \in T_n} (g^T \theta_t + b - \tilde{V}(o_t, \theta_t))^2.$$  

(16)

In both minimization problems, given by equations (15) and (16), the singular value decomposition is used to avoid singularities of linear equations. Using the gradient, $g_n$, the policy is updated at each time step in episode $n$ as follows:

$$\theta_{t+1} = \theta_t + \alpha g_n$$  

(17)

Here, $\alpha$ is the learning rate. In the limit $\alpha \to 0$, the orbit $o_t$ changes in a quasi-static manner, and $\tilde{V}$ is defined on the attractor.

**BENCHMARKS**

**Drag reduction through Blowing and Suction**

We apply the proposed algorithm to the active flow control problem \cite{14} to validate its effectiveness. The goal of this problem is to minimize the magnitude of the drag acting on a cylinder by controlling the mass flow rate of two synthetic jets symmetrically located on the top and bottom of a cylinder immersed in a two-dimensional incompressible fluid. The flow configuration and optimization problem have been adopted from Rabault et al. (2019) \cite{14}. We use their open source code and replace only the reinforcement learning part of that code. The parameters of reinforcement learning are as follows: $N_s = 200$, $N_o = 4000$, $M = 20$, and $\alpha = 10^{-2} - n/500$.

Figure 1 shows the learning history, where $N_o$ denotes the number of observables (pressure probes in this problem). From this figure, it can be seen that the drag coefficient $C_D$ decreases more stably when learning using the proposed algorithm compared to that in the previous studies. This is mainly because the optimization process in the proposed algorithm is deterministic, and the trajectory remains close to an attractor for each policy parameter $\theta$. The time variation of $C_D$ obtained according to the optimal policy for each case is shown in Fig 2. In statistical stationary state, after a sufficiently long term from the initial condition, our linear policy with $N_o = 11$ has achieved mean drag reduction of 0.2%, compared to a previous study \cite{14} adopting a large-scale neural network policy function with $N_o = 151$.

**Optimization of propulsion direction**

For the second demonstration, we consider a cylinder driven by a constant-magnitude external force $F$ in a 2D incompressible flow field (Fig. 3). The Reynolds number Re based on the external force is set to 100, and thus, a vortex street is observed in the wake of the cylinder, leading to suppression of the cylinder movement. The cylinder is a rigid body whose mass density is two times larger than that of the fluid, and its rotational motion is ignored for simplicity. The goal here is to maximize the time-averaged velocity $\langle V_x \rangle$ of the cylinder by controlling the direction $\beta$ of the external force. The observables are the $y$-component of velocity $V_y$ of the cylinder, lift $L$ and time delayed values of these two quantities: $o_t = (V_y(t), L(t))$. The dashed lines represent the time average from 250 to 500 for each number of observables $N_o$. The inset shows the zoomed trajectories of the time range $400 \leq t \leq 500$. 

![FIG. 1. Learning history of the blowing/suction problem. $\langle C_D \rangle$ is the time-averaged drag coefficient $C_D$ within one episode.](image1.png)

![FIG. 2. Time variation of the drag coefficient $C_D$. The dashed lines represent the time average from 250 to 500 for each number of obsevables $N_o$. The inset shows the zoomed trajectories of the time range $400 \leq t \leq 500$.](image2.png)
The reward is defined as $r_t = V_{x,t}$. The parameters of reinforcement learning are $N_s = 200, N_e = 4000, M = 20,$ and $\alpha = 10^{-4} - n/500$. The immersed boundary method is used for object representation. The time step $dt$ is set at 0.01.

Figure 3 shows the learning history in the cases of $N_o = 2, 4$ and 8. As the number of observables increases, the objective function increases rapidly. The performance with $N_o = 8$ is improved by approximately 1%, compared to the case when external forces are applied from a constant angle. In this control problem, the flow becomes unstable when the cylinder velocity is relatively high. Thus, the learning rate $\alpha$ needs to be small for successful and stable learning.

**CONCLUSION**

Existing reinforcement learning algorithms for fluid control are inefficient under a small number of observables even if the flow is laminar. By incorporating the low-dimensional nature of the dissipative system into the learning algorithm, we resolved this problem and presented a framework for reinforcement learning that stably optimizes the policy for a small number of observables. As demonstrated in previous studies, the orbit of flow hardly reaches the attractor during the learning process in the probabilistic policy framework. This leads to inefficient optimization of the attractor. Moreover, the deterministic policy described in this paper enables stable optimization on an attractor, and it is not necessary to learn the processes repeatedly.

Although the proposed algorithm is simple, the value function can be approximated using a neural network as in deep reinforcement learning, instead of the generalized linear model of form (11). A more complicated regression model may be applied instead of the linear regression model of (15). In addition, although on-policy-type learning was used in this study, the proposed algorithm can be easily modified to the off-policy type to emphasize exploration.

**ACKNOWLEDGEMENTS**

This work was supported by JSPS KAKENHI Grant Number JP17K14588. Simulations were performed on the “Plasma Simulator” (NEC SX-Aurora TSUBASA) of NIFS with support from the NIFS Collaboration Research program (NIFS19KNSS124).

*shimizu@me.es.osaka-u.ac.jp*

[1] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, Annual Review of Fluid Mechanics 52, 477 (2020).
[2] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction (MIT press, 2018).
[3] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, arXiv preprint arXiv:1811.12560 (2018).
[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al., arXiv preprint arXiv:1712.01815 (2017).
[5] J. Janai, F. Giney, A. Behl, A. Geiger, et al., Foundations and Trends® in Computer Graphics and Vision 12, 1 (2020).
[6] M. Gazzola, B. Hejazialhosseini, and P. Koumoutsakos, SIAM Journal on Scientific Computing 36, B622 (2014).
[7] M. Gazzola, A. A. Tchieu, D. Alexeev, A. de Brauer, and P. Koumoutsakos, Journal of Fluid Mechanics 789, 726 (2016).
[8] G. Novati, S. Verma, D. Alexeev, D. Rossinelli, W. M. Van Rees, and P. Koumoutsakos, Bioinspiration & biomimetics 12, 036001 (2017).
[9] F. Guérit, L. Mathelin, and M. Y. Hussaini, Theoretical and Computational Fluid Dynamics 30, 497 (2016).
[10] G. Reddy, A. Celani, T. J. Sejnowski, and M. Vergassola, Proceedings of the National Academy of Sciences 113, E4877 (2016).
[11] G. Reddy, J. Wong-Ng, A. Celani, T. J. Sejnowski, and M. Vergassola, Nature 562, 236 (2018).
[12] H. Koizumi, S. Tsutsumi, and E. Shima, in 2018 Flow Control Conference (2018) p. 3691.
[13] S. Verma, G. Novati, and P. Koumoutsakos, Proceedings of the National Academy of Sciences 115, 5849 (2018).
[14] J. Rabault, M. Kuchta, A. Jensen, U. Réglade, and N. Cerardi, Journal of Fluid Mechanics 865, 281 (2019).
[15] J. Rabault and A. Kuhnle, arXiv preprint arXiv:1906.10382 (2019).
[16] H. Tang, J. Rabault, A. Kuhnle, Y. Wang, and T. Wang, Physics of Fluids 32, 053605 (2020).
[17] R. Paris, S. Beneddine, and J. Dandois, arXiv preprint arXiv:2006.11005 (2020).
[18] R. Temam, Infinite-dimensional dynamical systems in mechanics and physics, Vol. 68 (Springer Science & Business Media, 2012).
[19] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller (2014).
[20] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, Advances in neural information processing systems 12, 1057 (1999).
[21] A. Rahimi and B. Recht, in Advances in neural information processing systems (2009) pp. 1313–1320.
[22] F. Pedregosa et al., the Journal of machine Learning research 12, 2825 (2011).
[23] M. Uhlmann, Journal of Computational Physics 209, 448 (2005).