10.2514/6.2017-3566

AIAA AVIATION Forum
5-9 June 2017, Denver, Colorado
35th AIAA Applied Aerodynamics Conference

A continuous reinforcement learning strategy for closed-loop control in fluid dynamics

Charles Pivot*
Institut Pprime, CNRS, Université de Poitiers, ISAE-ENSMA, France

Lionel Mathelin
LIMSI, Université de Paris Saclay, France

Laurent Cordier
Institut Pprime, CNRS, Université de Poitiers, ISAE-ENSMA, France

Florimond Guéniat
Department of Mathematics, Florida State Univ., USA

Bernd R. Noack
LIMSI, Université de Paris Saclay, France

This article presents the application of a continuous Reinforcement Learning (RL) framework to the closed-loop control of the drag of a cylinder wake flow. RL is a pure-driven approach that can rely on scarce and streaming data to find an optimal control policy for a task in an unknown environment. An implicit model of the system at hand is learned from sensor measurements, allowing for gradients evaluation and leading to quick convergence to an efficient control policy. This data-driven method is well suited for experimental configurations, requiring no computations nor prior knowledge of the system, and enjoys intrinsic robustness. These properties are highly desirable in flow control where real-time constraints, limited embedded hardware computational power and severe noise prevent the use of model-based strategies. Specific issues of reinforcement learning in the context of closed-loop flow control are briefly discussed.

I. Introduction

While the design and capability of aircraft, and more generally of complex systems, have significantly improved over the years, closed-loop control can bring further improvement in terms of performance and robustness to non-modeled perturbations. In the context of flow control, closed-loop control however suffers from severe limitations preventing its use in many situations. As a paradigmatic example, a typical turbulent flow involves both a large range of spatial scales and exhibits a rich and fast dynamics. High frequency phenomena hence require a control command fast enough to adjust to the current state of the quickly evolving flow system.

While flow manipulation and open-loop control are common practice, much fewer successful closed-loop control efforts are reported in the literature. Further, many of them rely on unrealistic assumptions. If a model is employed as is common practice, being a high-fidelity Navier-Stokes model or a Reduced-Order Model (ROM), one often needs to observe the *whole* system for informing the model.[1,2] Hence, with this class of approaches, flow control is restricted to numerical simulations or experiments in a wind tunnel equipped with sophisticated visualization tools such as Particle Image Velocimetry.

This paper presents a *practical* strategy for closed-loop control of complex flows by alleviating the limitations of current methods. The present work relies on a change of paradigm: we want to derive a general nonlinear closed-loop flow control methodology suitable for *actual* configurations and as realistic as possible. No *a priori* model, not even a model structure, describing the dynamics of the system is required to be

*charles.pivot@univ-poitiers.fr

available. The approach proposed is *data-driven only*, with the sole information about the system given by scarce and spatially-constrained sensors, and relies on statistical learning methods.[3]

Among the challenges in machine learning methods, the question of dealing with poor observability and hidden information remains largely open. In the present work, we handle this issue by taking advantage of the intrinsic temporal relationship between measurements, as pioneered in.[4]

Reinforcement learning[5,6] is a suitable class of methods for the control of Markov processes, see for instance[7] for the control of 1-D and 2-D chaotic maps. The usual version that consists of considering a discretized version of the state and action spaces suffers from the so-called *curse of dimensionality* problem for large-scale dynamical systems. For this reason, we consider an alternative approach where the states and actions are continuous. This approach allows to exploit smoothness of the dynamics and the input-output map. Conceptually, a model is learned, allowing for gradients to be reliably estimated, hence significantly improving the convergence rate of learning the control policy. This is a significant difference with a *critic-only* approach such as Genetic Programming Control[2] where the policy is found as the solution of a minimization problem. The resulting control strategy, being data-driven only, does not require significant computational resources nor prior knowledge of the system.

The paper is organized as follows. Section II introduces notations and key techniques our approach relies on. The framework and basics of continuous reinforcement learning are presented in Section III. The resulting control strategy is illustrated in Section IV with the drag reduction of a two-dimensional cylinder wake flow. Concluding remarks close the paper in Section V.

II. Preliminaries

II.A. Notations

Let $\mathfrak{f}$ be the flow map of the underlying dynamical system, we consider in the following the continuous-time deterministic system,

$$\dot{\boldsymbol{X}}_t = \mathfrak{f}\left(\boldsymbol{X}_t, \boldsymbol{a}_t\right), \tag{1}$$

where $\boldsymbol{X}_t := \boldsymbol{X}\left(t\right) \in \mathcal{X}$ is the state of the system at time t, and $\boldsymbol{a}_t := \boldsymbol{a}\left(t\right) \in \mathcal{A} \subset \mathbb{R}^{n_a}$ is the action with n_a the number of actuators. Let $\boldsymbol{g} : \mathcal{X} \to \mathbb{R}^{n_g}$ be a sensor function, with n_g the number of sensors, the observed data at time t, $\boldsymbol{y}_t \in \mathbb{R}^{n_g}$ are defined as $\boldsymbol{y}_t := \boldsymbol{g}\left(\boldsymbol{X}_t\right)$.

II.B. Embedding

To simplify the description of the dynamics, a simple method of delays as described in[8] is used to reconstruct qualitative features of the phase space of the system from time series of measurements of a single observable $\boldsymbol{y}_t$. For that, sampling has to be fast enough to capture the small time scales of the dynamics of the observable. The embedding dimension n_e is defined as, at least, twice the correlation dimension of $\boldsymbol{y}_t$ that is estimated from time-series of the observable, using for instance the Grassberger-Proccacia algorithm.[9] Let Δt be the sampling rate of the measurement system, the data from the sensors are embedded in a reconstructed phase space Ω defined as:

$$\left(\boldsymbol{y}_t^T \; \boldsymbol{y}_{t-\Delta t}^T \cdots \boldsymbol{y}_{t-(n_e-1)\Delta t}^T\right) := \mathbf{Y}_t^T \in \Omega \subset \mathbb{R}^{n_e \times n_g}. \tag{2}$$

Under mild assumptions,[8] this embedding ensures there is a diffeomorphism between the phase space $\mathcal{X}$ and the reconstructed phase space Ω, so that $\mathbf{Y}_t$ is an observable of the system.

II.C. Delayed effect of the action

In many physical system of interest in flow control (for example the one described in Sec. IV), there is a delay τ_d between the introduction of the action and the measured effect by the sensors. To control these kinds of systems with a reinforcement learning algorithm, we need to extend[10] the reconstructed state $\mathbf{Y}_t$ with all the actions applied to the system between $t - \tau_d$ and t. Finally, the state of the system at time t, $\boldsymbol{x}_t$, is defined as

$$\left(\mathbf{Y}_t^T \; \boldsymbol{a}_t^T \; \boldsymbol{a}_{t-1}^T \cdots \; \boldsymbol{a}_{t-d}^T\right) := \boldsymbol{x}_t^T \in \mathcal{S} = \Omega \times \mathcal{A}^d, \tag{3}$$

where $d := \lceil \frac{\tau_d}{\Delta t_a} \rceil$ corresponds to the number of actions applied between $t - \tau_d$ and t, Δt_a being the time between two consecutive actions.

II.D. Identification of Local Linear Models

In the continuous framework of reinforcement learning, we would like that experience acquired on a limited subset of the state space be usefully generalized to produce a good approximation of quantities of interest over a much larger subset. This problem of generalization[11] can be cured by relying on *function approximations*. Many methods for approximating functions are available. Here, to approximate the different quantities involved in reinforcement learning, we use a local linear regression (LLR).[12] For a quantity $\boldsymbol{z}$ depending on generic parameters $\boldsymbol{\theta}$, the principle is to store in a matrix called the memory, an ensemble $\mathcal{I}$ of samples $\left\{\boldsymbol{\theta}^{(i)}, \boldsymbol{z}^{(i)}\right\}_{i\in\mathcal{I}}$ previously determined. When a query $\boldsymbol{\vartheta}$ is made, LLR uses a limited number of samples of the memory to give a prediction $\widehat{\boldsymbol{z}}\left(\boldsymbol{\vartheta}\right)$ of the true output $\boldsymbol{z}\left(\boldsymbol{\vartheta}\right)$. With a local least squares technique, the approximation is given by

$$\boldsymbol{z}\left(\boldsymbol{\vartheta}\right) \approx \widehat{\boldsymbol{z}}\left(\boldsymbol{\vartheta}\right) = \beta \begin{pmatrix} \boldsymbol{\vartheta} \\ 1 \end{pmatrix}, \tag{4}$$

where only the K nearest neighbors $\mathcal{K}\left(\boldsymbol{\vartheta}\right)$ of $\boldsymbol{\vartheta}$ is used for the determination of β. This coefficient can be determined through a pseudo inverse as $\beta = Z\,\Theta^{+}$ where $\Theta := \begin{pmatrix} \left\{\boldsymbol{\theta}^{(i)}\right\}_{i\in\mathcal{K}(\boldsymbol{\vartheta})} \\ \mathbf{1}_K \end{pmatrix} \in \mathbb{R}^{(n_{\boldsymbol{\theta}}+1)\times K}$ and $Z := \left(\left\{\boldsymbol{z}^{(i)}\right\}_{i\in\mathcal{K}(\boldsymbol{\vartheta})}\right) \in \mathbb{R}^{n_{\boldsymbol{z}}\times K}$ collect the relevant inputs and outputs from the database. $\beta \in \mathbb{R}^{n_{\boldsymbol{z}}\times(n_{\boldsymbol{\theta}}+1)}$ is composed of a bias term and an approximation of the gradient of $\boldsymbol{z}$ around $\boldsymbol{\vartheta}$. Efficient strategies are used to manage and update the collection of samples.[13]

III. Continuous reinforcement learning

Classical reinforcement learning approaches are based on the assumption that we have a Markov Decision Process (MDP) consisting of the set of states $\mathcal{X}$, set of actions $\mathcal{A}$, the reward function $r : \mathcal{X}\times\mathcal{A} \to \mathbb{R}$ and $\gamma \in [0,1]$ a discount factor.

The goal of reinforcement learning is to discover an optimal *policy* $\pi : \mathcal{X} \to \mathcal{A}$ that maps states to actions so as to maximize the *Value function*, V^{π}, defined as the sum of the future expected rewards when starting at state $\boldsymbol{X}_t$:

$$V^{\pi}\left(\boldsymbol{X}_t\right) := \int_t^{+\infty} e^{-\frac{s-t}{\tau}} r(\boldsymbol{X}_s, \boldsymbol{a}_s)\,\mathrm{d}s, \qquad \boldsymbol{a}_s := \pi\left(\boldsymbol{X}_s\right). \tag{5}$$

τ is the time constant for discounting future rewards. An important feature of this infinite-horizon formulation is that the value function and the policy do not depend explicitly on time, which is convenient in estimating them using function approximators. With some manipulation, (5) may be reformulated as the Bellman equation associated to the optimal policy:[14]

$$V^{\pi}\left(\boldsymbol{X}_t\right) = \max_{\boldsymbol{a}_t[t,+\infty[} \left[r(\boldsymbol{X}_t, \boldsymbol{a}_t) + \gamma V^{\pi}(\boldsymbol{X}_{t+1})\right], \tag{6}$$

where $\gamma = e^{-\frac{\Delta t}{\tau}}$ with Δt the sampling time. To determine an optimal policy, the rewards associated with an action when in a given state are first learned. Then, this information is used to derive a control policy to drive the system along transitions and actions associated with the largest rewards.

Reinforcement Learning is a suitable class of methods for the control of Markov processes when the distribution of transition probabilities and values are difficult to evaluate.[5,15,16] In the following, the algorithm we consider is based on the sequential minimization of the temporal difference defined at time t as the difference between the two sides of the Bellman equation (6). This quantity, also called inconsistency, can be obtained by differentiating the Value function (5) *i.e.*

$$\delta_t := r(\boldsymbol{X}_t, \boldsymbol{a}_t) + \gamma V^{\pi}(\boldsymbol{X}_{t+1}) - V^{\pi}(\boldsymbol{X}_t). \tag{7}$$

We now introduce a model-based continuous method for minimizing the temporal difference. In our case, the state $\boldsymbol{X}_t$ is unknown and, in the following, we use $\boldsymbol{x}_t$ instead, which only involves accessible information on the system. The value function V^{π} is approximated with the use of local linear models.[13] We note $\widehat{V^{\pi}}(\boldsymbol{x})$ the approximation of the value function, see Sec. II.D, $\widehat{V^{\pi}}(\boldsymbol{x}) = \beta^V \begin{pmatrix} \boldsymbol{x} \\ \mathbf{1} \end{pmatrix}$. The flow map ϕ of the system, which describes the evolution of $\boldsymbol{x}_t$, and the policy π are also approximated with the use of local linear models. We have $\widehat{\boldsymbol{x}_{t+1}} = \widehat{\phi}(\boldsymbol{x}_t, \boldsymbol{a}_t) = \beta^{\phi} \begin{pmatrix} \boldsymbol{x}_t \\ \boldsymbol{a}_t \\ \mathbf{1} \end{pmatrix}$, and $\pi(\boldsymbol{x}_t) \approx \widehat{\pi}(\boldsymbol{x}_t) = \beta^{\pi} \begin{pmatrix} \boldsymbol{x}_t \\ \mathbf{1} \end{pmatrix}$.

The temporal difference equation (7) can then be used for updating the estimated value function[13] following

$$\widehat{V^{\pi}} \leftarrow \widehat{V^{\pi}} + \alpha_{t,c}\, \delta_t\, \boldsymbol{e}, \tag{8}$$

where $\boldsymbol{e} \in [0,1]^{|\mathcal{I}|}$ is the eligibility trace,[11] and $\alpha_{t,c} \in]0,1[$ is a learning rate. The concept of eligibility trace represents a temporary record of the occurrence of an event, such as the visiting of a state or the taking of an action. Essentially, the trace $\boldsymbol{e}$ marks the memory parameters associated with the event as eligible for undergoing learning changes. The importance of these events decays with time with a factor $\lambda \in [0,1]$. The eligibility trace is then updated as

$$\boldsymbol{e} \leftarrow \lambda\, \boldsymbol{e}, \tag{9}$$

where $e_i = 1$ for $i \in \mathcal{K}(\boldsymbol{x}_t)$. The policy is also updated. At each time t, the actions $\left\{\boldsymbol{a}^{(i)}\right\}_{i \in \mathcal{K}(\boldsymbol{x}_t)}$, used for the approximation of the policy (see (4)) are modified. Finding the optimal policy requires to maximize the value function. Consequently, at any time t, the optimal corrections $\Delta \boldsymbol{a}$ have to follow the gradient of the value function with respect to the chosen action $\boldsymbol{a}$ *i.e.*

$$\Delta \boldsymbol{a} \propto \alpha_{t,a}\, \nabla_{\boldsymbol{a}} \widehat{V^{\pi}}, \tag{10}$$

with $\alpha_{t,a} \in]0,1[$ a learning rate. The gradient in (10) cannot be directly evaluated, but assuming that the Bellman equation (6) is satisfied, one can write:

$$\nabla_{\boldsymbol{a}} \widehat{V^{\pi}} = \nabla_{\boldsymbol{a}} r(\boldsymbol{X}_t, \boldsymbol{a}_t) + \gamma \nabla_{\boldsymbol{a}} \widehat{V^{\pi}}(\widehat{\boldsymbol{x}_{t+1}}). \tag{11}$$

The first term of the right hand side is analytically known. Using the chain rule, the second term can be decomposed as[13]

$$\nabla_{\boldsymbol{a}} \widehat{V^{\pi}}(\widehat{\boldsymbol{x}_{t+1}}) = \left(\nabla_{\boldsymbol{x}} \widehat{V^{\pi}}(\widehat{\boldsymbol{x}_{t+1}})\right)^{\mathsf{T}} \nabla_{\boldsymbol{a}} \widehat{\phi}(\boldsymbol{x}, a) = \beta^{V}_{\boldsymbol{x}}\, \beta^{\phi}_{\boldsymbol{a}}, \tag{12}$$

where $\beta^{V}_{\boldsymbol{x}}$ correspond to the part of β^{V} associated with the state $\boldsymbol{x}_t$, and $\beta^{\phi}_{\boldsymbol{a}}$ correspond to the part of β^{ϕ} associated with the action $\boldsymbol{a}$. The actions are then updated following :

$$\boldsymbol{a}^{(i)} \leftarrow \boldsymbol{a}^{(i)} + \Delta \boldsymbol{a} \quad \text{with} \quad i \in \mathcal{K}(\boldsymbol{x}_t). \tag{13}$$

To improve the performance of the algorithm, exploration is added every p time steps by randomly perturbing the command law :

$$\boldsymbol{a} \leftarrow \boldsymbol{a} + \mathcal{N}(\mathbf{0}, \Sigma), \tag{14}$$

where $\mathcal{N}(\mathbf{0}, \Sigma)$ denotes a centered Gaussian noise with covariance Σ, here chosen as $\Sigma = \sigma^2\, \mathrm{I}_{n_a}$.

IV. Proof-of-concept: Two-dimensional cylinder wake flow

Table 1. Parameters of the Model Learning Actor-Critic algorithm of.[13]

	Actor	Critic	Model
Learning rate	$\alpha_{t,a} = 0.002$	$\alpha_{t,c} = 0.005$	-
Memory size	$\lvert\mathcal{I}_a\rvert = 10000$	$\lvert\mathcal{I}_c\rvert = 10000$	$\lvert\mathcal{I}_m\rvert = 10000$
Neighbours	$K_a = 80$	$K_c = 80$	$K_m = 80$

Other parameters	
γ	0.93
λ	0.7
ρ	0.1

To illustrate the methodology discussed above, we consider a 2-D laminar flow around a circular cylinder. The Reynolds number of the flow is $Re = 200$ based on the cylinder diameter and the upstream flow velocity. The observable $\boldsymbol{y}_t$ is constructed from the time series $\{\boldsymbol{y}\,(t - n\,\Delta t)\}_{n=0}^{n_e - 1}$ of an array of three pressure sensors

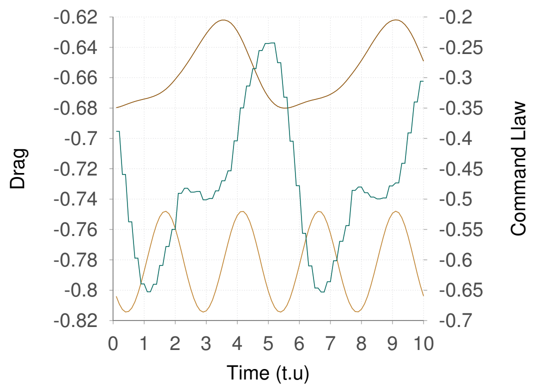

Figure 1. Time evolution of the drag force F_D for the cylinder wake flow at $Re = 200$. Pale brown corresponds to the drag associated with the uncontrolled flow. Dark brown is the drag obtained by reinforcement learning at convergence. Blue corresponds to the control law determined by reinforcement learning.

located half a radius downstream the cylinder, sampled every $\Delta t = 5$ time units. The embedding dimension is determined to be $n_e = 2$. We use the finite element code `Feel++`[17] to solve the Navier-Stokes equations on an unstructured grid. Free-stream velocity boundary conditions are imposed at the top and bottom of the numerical domain. The time step of the simulation is $dt = 0.1$ time unit (t.u.).

The control is achieved via rotation of the cylinder at a rotational speed denoted by $a\,(t)$. The actuation is updated every two time units. $\mathcal{J}$, the cost function to minimize is the sum of the drag F_D induced by the cylinder and a penalization term corresponding to the intensity of the command, *i.e.*

$$\mathcal{J}(\boldsymbol{X}_t, a) = -r(\boldsymbol{X}_t, a) = -\rho\, F_D^2 - a^2. \tag{15}$$

The drag F_D is a function of the state $\boldsymbol{X}_t$. The penalty $\rho > 0$ is chosen such that the resulting command remains within the operating range of the control.

The total time of simulation is 1500 t.u. which corresponds to about 400 periods of vortex shedding. The exploration is added every 10 time steps, up to $t = 750$ t.u., with an amplitude of 0.5 rad.$(\text{t.u.})^{-1}$. The maximum rotation speed of the cylinder is set to 5 rad.$(\text{t.u.})^{-1}$. The other parameters of the algorithm are presented in Table 1.

The drag coefficient of the cylinder is plotted in Fig. 1 for the optimal policy determined by reinforcement learning and also for the uncontrolled flow ($a = 0$). The identified control is seen to perform well. The drag is reduced by 17 % and the control command oscillates at the period of the wake, half the frequency of the drag, as expected.

To illustrate the convergence of the algorithm, the average of the Value function over a sliding time horizon of 10 t.u. is plotted in Fig. 2. This quantity gives an indication of the expectation of the Value over the invariant measure set. On this figure, two regimes can be observed. The first one, for $t < 700$ t.u., corresponds to a *t*ransient regime, where the algorithm explores the phase space. The second regime, for $t > 700$ t.u., shows that the sum of the value increases. For $t > 1400$ t.u, the algorithm has converged and the sum of the value reaches a plateau.

V. Concluding remarks

In this work, we present an experiment-oriented control strategy which does not require any prior knowledge of the physical system to be controlled nor significant computational resources. This strategy allows the learning of a control policy from scarce and point-wise sensors, hence very limited information on the system at hand. From the sensors' streaming data, an implicit model of the state dynamics is built.

The resulting method is compliant with actual configurations where instrumentation is limited and spatially-constrained. Owing to the construction and the use of low-dimensional models, the algorithm runs in real-time and allows closed-loop control.

The method is illustrated by the reduction of the drag of the two-dimensional flow around a circular cylinder. Measurements are provided by an array of pressure probes and actuation is achieved by rotating the cylinder. 17% of drag reduction is obtained. Current efforts are devoted to control a more involved

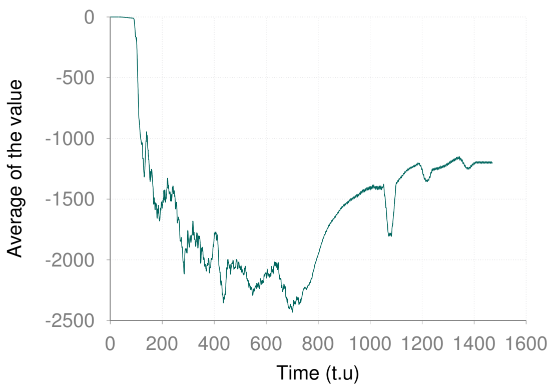

Figure 2. Average of the Value function over a sliding time horizon of 10 t.u..

configuration made of a fluid-structure interaction system in the turbulent regime.

More generally, the method presented in this work is readily applicable to physical systems with a causal link between the actuators and the cost functional as evaluated from the sensors. It does not rely on a prior model and instead learns directly from observing the system under stimulation by the actuators, hence being suitable for practical configurations.

While these preliminary results are encouraging, statistical learning methods face a number of issues when applied to the closed-loop control of complex fluid flows. Among these, the system is likely not to be stationary, in contrast with the ergodicity assumption these methods rely on. Further, delays between actions and sensor measurements are not constant over time while the input-output map may exhibit discontinuities when the flow topology undergoes a bifurcation. An additional issue is that the training time for statistical learning-based control is often too large in many situations. These issues should be addressed before the general concept of statistical learning for flow control can be applied in practical situations. This is the subject of intense ongoing efforts.

References

[1]Mathelin, L., Pastur, L., and Le Maître, O., “A Compressed-Sensing Approach for Closed-Loop Optimal Control of Nonlinear Systems,” *Theoret. Comput. Fluid Dyn.*, Vol. 26, No. 1-4, 2012, pp. 319–337.

[2]Brunton, S. and Noack, B. R., “Closed-Loop Turbulence Control: Progress and Challenges,” *App. Mech. Rev.*, Vol. 67, No. 5, 2015, pp. 050801.

[3]Murphy, K. P., *Machine Learning: A Probabilistic Perspective*, MIT Press, Cambridge, 2012.

[4]Guéniat, F., Mathelin, L., and Hussaini, M., “A statistical learning strategy for closed-loop control of fluid flows,” *Theoret. Comput. Fluid Dyn.*, 2016, pp. 1–14.

[5]Watkins, C. and Dayan, P., “Q-Learning,” *Mach. Learn.*, Vol. 8, No. 3-4, 1992, pp. 279–292.

[6]Gosavi, A., “Target-Sensitive Control of Markov and Semi-Markov Processes,” *Int. J. Control Autom.*, Vol. 9, No. 5, 2011, pp. 941–951.

[7]Lin, C. and Jou, C., “Controlling Chaos by GA-Based Reinforcement Learning Neural Network,” *IEEE T. Neural Network*, Vol. 10, No. 4, 1999, pp. 846–859.

[8]Takens, F., Rand, D., and Young, L., “Dynamical Systems and Turbulence,” *Lect. Notes Math.*, Vol. 898, No. 9, 1981, pp. 366.

[9]Grassberger, P. and Procaccia, I., “Measuring the Strangeness of Strange Attractors,” *Physica D*, Vol. 9, 1983, pp. 189–208.

[10]Katsikopoulos, K. V. and Engelbrecht, S. E., “Markov decision processes with delays and asynchronous cost collection,” *IEEE Transactions on Automatic Control*, Vol. 48, No. 4, April 2003, pp. 568–574.

[11]Sutton, R. S. and Barto, A. G., *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, A Bradford Book, 1998.

[12]Wilson, D. and Martinez, T., “Reduction techniques for instance-based learning algorithms,” *Mach. Learn.*, Vol. 38, No. 3, 2000, pp. 257–286.

[13]Grondman, I., Vandraager, M., Busoniu, M., Babuska, R., and Schuitema, E., “Efficient model learning methods for actor-critic control,” *Sys. Man Cyber.*, Vol. 42, No. 3, 2012, pp. 591–602.

[14]Doya, K., “Reinforcement Learning In Continuous Time and Space,” *Neural Computation*, Vol. 12, 2000, pp. 219–245.

[15]Powell, W., *Approximate Dynamic Programming: Solving the curses of dimensionality*, Vol. 703, John Wiley & Sons, 2007.

[16]Lewis, F. and Vrabie, D., “Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control,” *Circuits Syst. Mag., IEEE*, Vol. 9, No. 3, 2009, pp. 32–50.

[17]Prud’Homme, C., Chabannes, V., Doyeux, V., Ismail, M., Samake, A., and Pena, G., “Feel++: A Computational Framework for Galerkin Methods and Advanced Numerical Methods,” *ESAIM: Proceedings*, Vol. 38, Dec. 2012, pp. 429–455.