REINFORCEMENT LEARNING STRATEGIES FOR CLOSED-LOOP CONTROL IN FLUID DYNAMICS

C. PIVOT[1], L. MATHELIN[2], L. CORDIER[1], F. GUÉNIAT[3], B. R. NOACK[2,4,5]

1. Institut Pprime, UPR 3346, CNRS, Université de Poitiers, ISAE-ENSMA, F-86961 Futuroscope Chasseneuil, France
2. LIMSI, UPR 3251, F-91405 Orsay, France
3. Center for Exascale Simulation of Plasma-Coupled Combustion, Univ. of Illinois at Urbana-Champaign, USA
4. Technische Universität Braunschweig, D-38108 Braunschweig, Germany
5. Technische Universität Berlin, D-10623 Berlin, Germany

Résumé :
This work discusses a closed-loop control strategy based on a pure-driven approach relying on scarce and streaming data. A continuous reinforcement learning algorithm is applied to the sensor measurements from which a Markov process model is derived, approximating the system dynamics. An implicit model of the system at hand is learned from the data, allowing for gradients evaluation and leading to quick convergence to an efficient control policy. This method is illustrated on the control of the drag of a cylinder flow.

Mots clés : *Contrôle d'écoulement ; Apprentissage automatique*

1 Introduction

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, [9, 1]. 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 discusses 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 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.

Reinforcement learning, [14, 3], is a suitable class of methods for the control of Markov processes, see for instance [8] 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 GPC, [1]. The resulting control strategy, being data-driven only, does not require significant computational resources nor prior knowledge of the system.

2 Preliminaries

In the following, we consider the continuous-time deterministic dynamical system

$$\dot{X}_t = \mathfrak{f}\left(X_t, \boldsymbol{a}_t\right), \qquad X \in \mathcal{X}, \quad \boldsymbol{a} \in \mathcal{A} \subset \mathbb{R}^{n_a}, \quad (1)$$

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

2.1 Embedding

Let Δt be the sampling rate of the measurement system. Sampling has to be fast enough to capture the small time scales of the dynamics of $\boldsymbol{y}_t$. The data from the sensors are embedded in a reconstructed phase space Ω :

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

The correlation dimension of $\boldsymbol{y}$ is estimated from the

time-series, for instance using the Grassberger-Proccacia algorithm, [4]. It allows to define the embedding dimension n_e as, at least, twice the correlation dimension. Under mild assumptions, this resulting embedding dimension ensures there is a diffeomorphism between the phase space $\mathcal{X}$ and the reconstructed phase space, [13], so that $\boldsymbol{y}$ is an observable on the system.

2.2 Delayed effect of the action

In some experiment (for example the one described in Sec. 4) there might be a delay τ_d between the action and its effect as measured by the sensors. To control such a system with a reinforcement learning algorithm, we need to extend the state $\boldsymbol{y}$ with each actions applied to the system between $t-\tau_d$ and t, [6] :

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

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

2.3 Identification of Local Linear Models

In a continuous framework, smoothness of the dynamics can be taken as an advantage to approximate functions. Many techniques for approximating the dynamics are available and we use a locally linear modelling, [15], of the different quantities involved in reinforcement learning. More precisely, a generic quantity $\boldsymbol{z}$ depending on generic parameters $\boldsymbol{u}$ will be locally linearly approximated from acquired knowledge $\left\{\boldsymbol{u}^{(i)}, \boldsymbol{z}^{(i)}\right\}_{i\in\mathcal{I}}$, where $\mathcal{I}$ is a set of past samples. Considering the K nearest neighbors $\mathcal{K}(\boldsymbol{u})$, in the sense of a given metric, $\left\{\boldsymbol{u}^{(i)}, \boldsymbol{z}^{(i)}\right\}_{i\in\mathcal{K}(\boldsymbol{u})}$, $\boldsymbol{z}(\boldsymbol{u})$ is approximated with a local least squares technique as

$$\boldsymbol{z}(\boldsymbol{u}) \approx \widehat{\boldsymbol{z}}(\boldsymbol{u}) = \beta \begin{pmatrix} \boldsymbol{u} \\ 1 \end{pmatrix}. \quad (4)$$

The coefficients β can be determined through a pseudo inverse as $\beta = Z\,U^+$ where $U := \begin{pmatrix} \left\{\boldsymbol{u}^{(i)}\right\}_{i\in\mathcal{K}(\boldsymbol{u})} \\ \mathbf{1}_K \end{pmatrix} \in \mathbb{R}^{(n_{\boldsymbol{u}}+1)\times K}$ and $Z := \left(\left\{\boldsymbol{z}^{(i)}\right\}_{i\in\mathcal{K}(\boldsymbol{u})}\right) \in \mathbb{R}^{n_{\boldsymbol{z}}\times K}$ collect the relevant inputs and outputs from the database. $\beta \in \mathbb{R}^{n_{\boldsymbol{z}}\times(n_{\boldsymbol{u}}+1)}$ is composed of a bias term and an approximation of the gradient of $\boldsymbol{z}$ around $\boldsymbol{u}$.

Efficient strategies are used to manage and update the collection of samples, [5].

3 Continuous reinforcement learning

We work in the framework of Markov Decision Processes (MDP), which are 5-tuples $\{\mathcal{X}, \mathcal{A}, \mathfrak{f}, r, \gamma\}$, with $\mathcal{X}$, $\mathcal{A}$ and $\mathfrak{f}$ already defined, $r : \mathcal{X}\times\mathcal{A} \to \mathbb{R}$ a reward function and $\gamma \in [0,1]$ a discount factor.

The goal of the control strategy is to identify the optimal *policy* $\pi : \mathcal{X} \to \mathcal{A}$, which describes the best action to apply when in a given state so as to maximize the *Value function*, V^π, defined as the sum of the future expected rewards when starting at state X_t :

$$V^\pi(X_t) := \int_t^{+\infty} e^{-\frac{s-t}{\tau}} r(X_s, \boldsymbol{a}_s)\,\mathrm{d}s, \qquad \boldsymbol{a}_s := \pi(X_s), \quad (5)$$

where τ is the time constant for discounting future rewards. Equation (5) may be reformulated as the Bellman equation associated to the optimal policy, [2] :

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

where $\gamma = e^{-\frac{\Delta t}{\tau}}$ with Δt the sampling time.

To derive a control strategy, one needs to determine a policy which would give the best control action given the current state of the system. First, the rewards associated with an action when in a given state are 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, [14, 10, 7]. 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), which can also be obtained by differentiating the Value (5) *i*.e.

$$\delta_{t+1} := r(X_{t+1}, a_t) + \gamma V^\pi(X_{t+1}) - V^\pi(X_t). \quad (7)$$

We now introduce a model-based continuous method for minimizing the temporal difference. In our case, the state 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 modeled with the use of local linear models, [5]. We note $\widehat{V^\pi}(\boldsymbol{x})$ the approximation of the value function, see Sec. 2.3, $\widehat{V^\pi}(\boldsymbol{x}) = \beta^V \begin{pmatrix} \boldsymbol{x} \\ \mathbf{1} \end{pmatrix}$. The flow 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 $\widehat{\boldsymbol{x}_{t+1}} = \widehat{\phi}(\boldsymbol{x}_t, \boldsymbol{a}_t) \approx$

$\beta^\phi \begin{pmatrix} \boldsymbol{x}_t \\ \boldsymbol{a}_t \\ \mathbf{1} \end{pmatrix}$, and $\pi(\boldsymbol{x}) \approx \widehat{\pi}(\boldsymbol{x}) = \beta^\pi \begin{pmatrix} \boldsymbol{x} \\ \mathbf{1} \end{pmatrix}$.

The temporal difference can then be used for updating the value database, [5] :

$$\boldsymbol{e} \leftarrow \lambda\, \boldsymbol{e}, \tag{8}$$

$$e_i = 1, \qquad i \in \mathcal{K}(\boldsymbol{x}_t), \tag{9}$$

$$\widehat{V^\pi} \leftarrow \widehat{V^\pi} + \alpha_{t,c}\, \delta_t\, \boldsymbol{e}, \tag{10}$$

where $\boldsymbol{e} \in [0,1]^{|\mathcal{I}|}$ is the eligibility trace, [12]. Eligibility trace is a way to account for states (or states-actions) pairs visited several time steps in the past. The importance of these states decays with time with a factor $\lambda \in [0,1]$. $\alpha_{t,c} \in]0,1[$ is a learning rate.

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

$$\Delta\boldsymbol{a} \propto \alpha_{t,a}\, \nabla_{\boldsymbol{a}} \widehat{V^\pi}, \tag{11}$$

with $\alpha_{t,a} \in]0,1[$ a learning rate. The gradient in (11) cannot be directly evaluated but, assuming the Bellman equation (6), is satisfied, one can write :

$$\nabla_{\boldsymbol{a}} \widehat{V^\pi} = \nabla_{\boldsymbol{a}} r(X_t, \boldsymbol{a}_t) + \gamma \nabla_{\boldsymbol{a}} \widehat{V^\pi}(\widehat{\boldsymbol{x}_{t+1}}). \tag{12}$$

The first term of the right hand side is analytically known and, using the chain rule, the second term can be decomposed as, [5],

$$\nabla_{\boldsymbol{a}} \widehat{V^\pi}(\widehat{\boldsymbol{x}_{t+1}}) = \left(\nabla_{\boldsymbol{x}} \widehat{V^\pi}(\widehat{\boldsymbol{x}_{t+1}})\right)^{\mathsf{T}} \nabla_{\boldsymbol{a}} \widehat{\phi}(\boldsymbol{x}, a) = \beta^V_{\boldsymbol{x}}\, \beta^\phi_{\boldsymbol{a}}, \tag{13}$$

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

$$\boldsymbol{a}^{(i)} \leftarrow \boldsymbol{a}^{(i)} + \Delta\boldsymbol{a}, \qquad i \in \mathcal{K}(\boldsymbol{x}_t). \tag{14}$$

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

$$\boldsymbol{a} \leftarrow \boldsymbol{a} + \mathcal{N}(\mathbf{0}, \Sigma), \tag{15}$$

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

4 Proof-of-concept : Two-dimensional cylinder wake flow

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 $\boldsymbol{y}$ is constructed from the time series $\{y\,(t - n\,\Delta t)\}_{n=0}^{n_e-1}$ of an array of 3 pressure sensors located half a radius downstream the cylinder, sampled every $\Delta t = 5$ time units (t.u.). The embedding dimension is determined to be $n_e = 2$. We use the finite element code `Feel++`, [11], 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.

The control is achieved via rotation of the cylinder at a rotational speed $\boldsymbol{a} \equiv a\,(t)$. The actuation is updated every 2 time units. The cost function to minimize is the drag F_D induced by the cylinder penalized with the intensity of the command.

$$r(X_t, a) = -\rho\,|F_D|^2 - |a|^2. \tag{16}$$

The drag F_D is a function of the state 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 1000 t.u. (which corresponds to about 400 periods of vortex shedding) where the exploration is added every 10 time steps, up to $t = 750$ t.u., with an amplitude of 0.5 rad.(t.u.)$^{-1}$. The maximum rotation speed of the cylinder is set to 5 rad.(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 present approach as well as the NULL command, *i.e*, no control, $a \equiv 0$. 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.

Références

[1] S. Brunton and B. Noack. Closed-loop turbulence control : Progress and challenges. *App. Mech. Rev.*, 67(5) :050801, 2015.

[2] K. Doya. Reinforcement learning in continuous time and space. *Neural Computation*, 12 :219–245, 2000.

	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

TABLE 1 – Parameters of the Model Learning Actor-Critic algorithm.

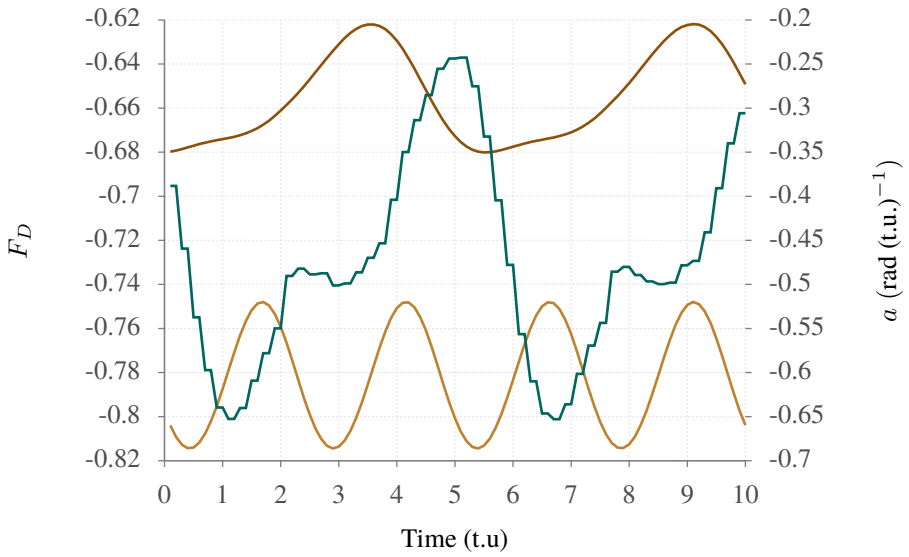

FIGURE 1 – Drag force F_D of the cylinder at $Re = 200$ function of time. In pale brown, the drag associated with the NULL command. In dark brown, the drag after convergence of the algorithm. In green, the command law evaluated by our algorithm.

[3] A. Gosavi. Target-sensitive control of Markov and semi-Markov processes. *Int. J. Control Autom.*, 9(5) :941–951, 2011.

[4] P. Grassberger and I. Procaccia. Measuring the strangeness of strange attractors. *Physica D*, 9 :189–208, 1983.

[5] I. Grondman, M. Vandraager, M. Busoniu, R. Babuska, and E. Schuitema. Efficient model learning methods for actor-critic control. *Sys. Man Cyber.*, 42(3) :591–602, 2012.

[6] K. V. Katsikopoulos and S. E. Engelbrecht. Markov decision processes with delays and asynchronous cost collection. *IEEE Transactions on Automatic Control*, 48(4) :568–574, April 2003.

[7] F. Lewis and D. Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *Circuits Syst. Mag., IEEE*, 9(3) :32–50, 2009.

[8] C.T. Lin and C.P. Jou. Controlling chaos by ga-based reinforcement learning neural network. *IEEE T. Neural Networ.*, 10(4) :846–859, 1999.

[9] L. Mathelin, L. Pastur, and O. Le Maître. A compressed-sensing approach for closed-loop optimal control of nonlinear systems. *Theo. Comp. Fluid Dyn.*, 26(1-4) :319–337, 2012.

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

[11] C. Prud'Homme, V. Chabannes, V. Doyeux, M. Ismail, A. Samake, and G. Pena. Feel++ : A Computational Framework for Galerkin Methods and Advanced Numerical Methods. *ESAIM : Proceedings*, 38 :429–455, December 2012.

[12] R.S. Sutton and A.G. Barto. *Reinforcement learning : An introduction*, volume 116. Cambridge Univ Press, 1998.

[13] F. Takens, D.A. Rand, and L.S. Young. Dynamical systems and turbulence. *Lect. Notes Math.*, 898(9) :366, 1981.

[14] C. Watkins and P. Dayan. Q-learning. *Mach. Learn.*, 8(3-4) :279–292, 1992.

[15] D. Wilson and T. Martinez. Reduction techniques for instance-based learning algorithms. *Mach. Learn.*, 38(3) :257–286, 2000.