2018 26th Mediterranean Conference on Control and Automation (MED)
Zadar, Croatia, June 19-22, 2018

Data-Driven Model-Free Model-Reference Nonlinear Virtual State-Feedback Control from Input-Output Data*

Mircea-Bogdan Radac, *Member, IEEE*, Radu-Emil Precup, *Senior Member, IEEE*,
Elena-Lorena Hedrea, and Ion-Cornel Mituletu

Abstract—**In this paper we show that it is possible to learn high performance model reference controllers using a model-free Q-learning approach, based on input-output (IO) samples collected from the controlled process. Under observability assumptions for the process, virtual extended state-space process models of different orders are built from the IO collected samples. We prove that state-space control of these virtual processes is equivalent to IO control of the initial process. For the virtual state-space process, high performance nonlinear Neural Network (NN) state-feedback controllers are learned based on the IO data collected from the initial process, to achieve output model-reference tracking control. Control learning is a two-step model-free process: an IO model-free controller first drives the process exploration of a wide operating range for IO samples collection that are then used to model-free learn the NN controllers. The approach is successfully validated on a highly nonlinear coupled aerodynamic system.**

I. Introduction

The interest in controller design based on input-output (IO) data has increased recently because of the more and more complicated processes whose precise models are expensive to determine and, even in a model-based control design, there is the possibility that the designed controller fails to be optimal due to infeasible analytical solutions for nonlinear and complex models and model uncertainties.

Literature survey acknowledges several so-called data-driven or data-based or model-free techniques that can improve control system (CS) performance by avoiding process models. Some emerge from classical control theory such as Virtual Reference Feedback Tuning (VRFT) [1], Iterative Feedback Tuning [2], Model Free Iterative Learning Control [3]–[5], Model Free (Adaptive) Control [6], [7], with representative applications [8]–[11]. Other techniques such as Reinforcement Learning (RL) [12], [13] have emerged both from machine learning/computer science and from classical control theory, where they are better known as Adaptive (Approximate, Neuro) Dynamic Programming [14]–[16]. Almost all above mentioned data-driven approaches share an optimal control framework.

*Research supported by the grant PCD-TC-2017.

M.-B. Radac, R.-E. Precup, R.-C Roman, E.-L. Hedrea and I.-C. Mituletu are with the Politehnica University of Timisoara, Department of Automation and Applied Informatics, Bd. V. Parvan 2, 300223 Timisoara, Romania (phone: +40 256403229, fax: +40 256403214; e-mail: mircea.radac@upt.ro, radu.precup@upt.ro, elena.constantin@student.upt.ro, ion.mituletu@student.upt.ro). I.-C. Mituletu is also with the Department of Electrical and Computer Engineering, "Eftimie Murgu" University of Resita, P-ta Traian Vuia 1-4, 320085 Resita, Romania.

The model-free RL techniques provide a very general stochastic nonlinear optimal control framework, and are best suited for state-space models on which the entire state information is available for measurement [13]–[14]. Whereas, the partial observability of the state leads to approximate control solutions only [17]. The main approach in this case is to use available historical IO data hoping to capture the entire state information at the current time, as a base for an optimal decision. IO data-driven linear time-invariant (LTI) observers are known [17], [18], while their nonlinear versions have only been used with model-free RL [19]–[22] but without a theoretical motivation.

The main contribution of this work exploits the nonlinear process observability for output reference model (ORM) control, by using IO samples collected from the process to build virtual state-space process models of different orders that extend the initial process model. Then, controlling the virtual fully state-observable process implies IO control for the underlying process. RL control of the virtual process is learned in a two-step model-free approach in this work. First, a model-free IO feedback controller guides the process in an exploratory regime in a wide operating range, for IO samples collection. Then virtual nonlinear state-feedback controllers for the virtual process are learned from the collected IO data by an iterative model-free approximate Value Iteration (IMF-AVI) implementation of batch-fitted Q-learning, as a representative RL technique. Among the RL techniques, model-free Q-learning [23], [24] (also known as Action-Dependent Heuristic Dynamic Programming) is popular owing to its simplicity and model-free character, and it has been implemented in many variants: online vs offline, adaptive or batch, for discrete/continuous states and actions, with/without function approximators, such as Neural Networks (NNs) [25]–[28].

Section II introduces the construction of the virtual fully-observable state-space process model after which the ORM control problem is formulated. Section III describes the IMF-AVI solution to the ORM problem whereas Section IV validates the proposed approach on a twin rotor aerodynamic system, with concluding remarks presented in Section V.

II. Model Reference Control for Unknown Nonlinear Processes

A. Process Observability

A discrete-time nonlinear unknown open-loop stable minimum-phase (MP) state-space deterministic strictly causal process is defined as

$$P:\begin{cases}\mathbf{x}_{k+1}=\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k),\\ \mathbf{y}_k=\mathbf{g}(\mathbf{x}_k),\end{cases} \tag{1}$$

where k indexes the discrete time, $\mathbf{x}_k=[x_{k,1}\dots x_{k,n}]^T\in\Omega_X\subset\Re^n$ is the n-dimensional state vector, $\mathbf{u}_k=[u_{k,1},\dots,u_{k,m_u}]^T\in\Omega_U\subset\Re^{m_u}$ is the control input signal, $\mathbf{y}_k=[y_{k,1},\dots,y_{k,p}]^T\in\Omega_Y\subset\Re^p$ is the measurable controlled output, $\mathbf{f}:\Omega_X\times\Omega_U\to\Omega_X$ is an *unknown* nonlinear system function continuously differentiable within its domain, $\mathbf{g}:\Omega_X\to\Omega_Y$ is an *unknown* nonlinear continuously differentiable output function. Initial conditions are not accounted for at this point. Let known Ω_U,Ω_Y and unknown Ω_X domains be compact convex. Equation (1) is a general un-restrictive form for most controlled processes. The following assumptions common to the data-driven formulation are:

A1: Process (1) is fully state observable and controllable.

A2: Process (1) is IO stable on known domain $\Omega_U\times\Omega_Y$.

A1 and *A2* are widely used in data-driven control, cannot be checked analytically for the unknown model (1) but can be inferred from historical and working knowledge with the process. Should such information not be available, the user can attempt process control under restraining safety operating conditions, that are usually dealt with at supervisory level control. IO stability (*A2*) is necessary if open-loop IO samples collection is intended for further IO control design and can be relaxed if a feedback stabilizing IO controller is used for IO data collection. Then IO process samples can be used to build data-driven observers as shown for linear systems in [17], [18] and used for nonlinear ones in [19]–[22], related to accessibility and flatness concepts [29]. A theoretical motivation is given:

Lemma 1. If $(\mathbf{f},\mathbf{g})$ in (1) is observable, then there exists a map $\mathbf{\Phi}$ and a positive integer r such that

$$\begin{aligned}&\mathbf{x}_k=\mathbf{\Phi}(\mathbf{Y}_{k,k-r},\mathbf{U}_{k-1,k-r}),\\ &\mathbf{Y}_{k,k-r}=[(\mathbf{y}_k)^T\dots(\mathbf{y}_{k-r})^T]^T,\mathbf{U}_{k-1,k-r}=[(\mathbf{u}_{k-1})^T\dots(\mathbf{u}_{k-r})^T]^T.\end{aligned} \tag{2}$$

Proof. Starting with $\mathbf{x}_k$ one iterates the following

$$\begin{aligned}&\mathbf{y}_k=\mathbf{g}(\mathbf{x}_k),\\ &\mathbf{y}_{k+1}=\mathbf{g}(\mathbf{x}_{k+1})=\mathbf{g}(\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k)):=\mathbf{gf}(\mathbf{x}_k,\mathbf{u}_k),\\ &\dots\\ &\mathbf{y}_{k+r}=\mathbf{gf}^r(\mathbf{x}_k,\mathbf{u}_k,\dots,\mathbf{u}_{k+r-1}),\end{aligned} \tag{3}$$

where $\mathbf{f}^t:=\mathbf{f}(\mathbf{f}(\dots\mathbf{f}(\mathbf{x})))$ for t times and $\mathbf{g}(\mathbf{f}(\mathbf{x})):=\mathbf{gf}(\mathbf{x})$. Then one can write, defining a map $\mathbf{A}$, that

$$\mathbf{Y}_{k+r,k}=\mathbf{A}(\mathbf{x}_k,\mathbf{U}_{k+r-1,k}), \tag{4}$$

where $\mathbf{Y}_{k+r,k}=[(\mathbf{y}_{k+r})^T,\dots,(\mathbf{y}_k)^T]^T,\mathbf{U}_{k+r-1,k}=[(\mathbf{u}_{k+r-1})^T,\dots,(\mathbf{u}_k)^T]^T$. Since $(\mathbf{f},\mathbf{g})$ is observable, there exists a positive integer r such that $\mathbf{A}(\mathbf{x},\mathbf{U})$ is invertible w.r.t. $\mathbf{x}$ [30] and

$$\mathbf{x}_k=\mathbf{A}^{-1}(\mathbf{Y}_{k+r,k},\mathbf{U}_{k+r-1,k}). \tag{5}$$

Note that $\mathbf{x}_k$ depends on future IO measurements up to time $k+r$. Replacing $k+r$ with t it follows that $\mathbf{x}_{t-r}=\mathbf{A}^{-1}(\mathbf{Y}_{t,t-r},\mathbf{U}_{t-1,t-r})$. Using (1), we also have:

$$\begin{aligned}&\mathbf{x}_{t-r+1}=\mathbf{f}(\mathbf{x}_{t-r},\mathbf{u}_{t-r}),\\ &\mathbf{x}_{t-r+2}=\mathbf{f}(\mathbf{x}_{t-r+1},\mathbf{u}_{t-r+1})=\mathbf{f}^2(\mathbf{x}_{t-r},\mathbf{u}_{t-r},\mathbf{u}_{t-r+1}),\\ &\dots\\ &\mathbf{x}_t=\mathbf{f}(\mathbf{x}_{t-1},\mathbf{u}_{t-1})=\mathbf{f}^r(\mathbf{x}_{t-r},\mathbf{u}_{t-r},\dots,\mathbf{u}_{t-1})\\ &\quad=\mathbf{f}^r(\mathbf{A}^{-1}(\mathbf{Y}_{t,t-r},\mathbf{U}_{t-1,t-r}),\mathbf{u}_{t-r},\dots,\mathbf{u}_{t-1}).\end{aligned} \tag{6}$$

Since all $\mathbf{u}_{t-r},\dots,\mathbf{u}_{t-1}\in\mathbf{U}_{t,t-r}$, last equation in (6) reads

$$\mathbf{x}_t=\mathbf{f}^r(\mathbf{A}^{-1}(\mathbf{Y}_{t,t-r},\mathbf{U}_{t-1,t-r}))=\mathbf{\Phi}(\mathbf{Y}_{t,t-r},\mathbf{U}_{t-1,t-r}). \tag{7}$$

which proves *Lemma 1*.

Comment 1. The previous development introduces a new state $\mathbf{z}_k=[(\mathbf{Y}_{k,k-r})^T,(\mathbf{U}_{k-1,k-r})^T]^T\in R^{p(r+1)+m_ir}$ as an *alias* for $\mathbf{x}_k$ and also as a state vector for a virtual process model.

Comment 2. Observability of (1) is equivalent to an unique solution $\mathbf{x}_k$ of the $(r+1)\times p$ nonlinear equations (4).

Theorem 1. Accepting the definition of the state vector $\mathbf{z}_k$ given in *Comment 1*, the virtual state-space process model can be expressed, and it is fully observable.

Proof. We introduce the new state $\mathbf{z}_k\in R^{p(r+1)+m_ir}$ as

$$\begin{aligned}\mathbf{z}_k=[&(\mathbf{z}_{k,1})^T=(\mathbf{y}_k)^T,\dots,(\mathbf{z}_{k,r+1})^T=(\mathbf{y}_{k-r})^T,\\ &(\mathbf{z}_{k,r+2})^T=(\mathbf{u}_{k-1})^T,\dots,(\mathbf{z}_{k,2r+1})^T=(\mathbf{u}_{k-r})^T]^T.\end{aligned} \tag{8}$$

Then, the next state is

$$\begin{aligned}\mathbf{z}_{k+1}=[&(\mathbf{z}_{k+1,1})^T=(\mathbf{y}_{k+1})^T,\dots,(\mathbf{z}_{k+1,r+1})^T=(\mathbf{y}_{k-r+1})^T,\\ &(\mathbf{z}_{k+1,r+2})^T=(\mathbf{u}_k)^T,\dots,(\mathbf{z}_{k+1,2r+1})^T=(\mathbf{u}_{k-r+1})^T]^T.\end{aligned} \tag{9}$$

The virtual state-space process model becomes

$$\begin{aligned}&\mathbf{z}_{k+1,1}=\mathbf{y}_{k+1}=\mathbf{g}(\mathbf{x}_{k+1})=\mathbf{g}(\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k))\\ &\quad=\mathbf{g}(\mathbf{f}(\mathbf{\Phi}(\mathbf{Y}_{k,k-r},\mathbf{U}_{k-1,k-r}),\mathbf{u}_k))=\mathbf{g}(\mathbf{\Phi}(\mathbf{z}_k),\mathbf{u}_k),\\ &\mathbf{z}_{k+1,2}=\mathbf{y}_k=\mathbf{z}_{k,1},\\ &\dots\\ &\mathbf{z}_{k+1,r+1}=\mathbf{y}_{k-r+1}=\mathbf{z}_{k,r},\\ &\mathbf{z}_{k+1,r+2}=\mathbf{u}_k,\\ &\mathbf{z}_{k+1,r+3}=\mathbf{u}_{k-1}=\mathbf{z}_{k,r+2},\\ &\dots\\ &\mathbf{z}_{k+1,2r+1}=\mathbf{u}_{k-r+1}=\mathbf{z}_{k,2r}.\end{aligned} \tag{10}$$

Then the virtual state-space model (10) takes the form

$$\begin{aligned}&\mathbf{z}_{k+1}=\mathbf{F}(\mathbf{z}_k,\mathbf{u}_k),\\ &\mathbf{y}_k=\mathbf{z}_{k,1},\end{aligned} \tag{11}$$

that represents a fully observable deterministic process with partially-known dynamics, for which control can be attempted. This concludes the proof.

Comment 3. While the order n of (1) is generally

unknown, so it is r. One can try control for increasing values of r to observe the change in performance.

Comment 4. $\mathbf{A}^{-1}(\mathbf{Y}_{t,t-r}, \mathbf{U}_{t-1,t-r})$ may depend only on subsets of $\mathbf{Y}_{t,t-r}, \mathbf{U}_{t-1,t-r}$, namely $\mathbf{Y}_{t,t-n_y}, \mathbf{U}_{t-1,t-n_u}$, $n_y, n_u \le r$.

Comment 5. (11) is a partially known process, its first equation depends indirectly on the unknown initial process (1). Then, for RL control, model-free solutions are preferable such as, e.g., model-free approximate Value- or Policy-Iteration [13] that rely on collected input-state data (for (11) the state consists of IO data from the initial process). If, for nonlinear [17] processes, (11) is completely known, then model-based Value- or Policy-Iteration control should be used directly.

Comment 6. Since $\mathbf{y}_k$ is a both a subset of the state $\mathbf{z}_k$ and the output of the virtual process (11) and $\mathbf{u}_k$ is the control input of the virtual process (11), controlling the virtual process (11) implies that output control for the unknown initial process (1) is also achieved.

Comment 7. The proposed observability results account for a wide range of processes including time-delay ones. For positive integer nonzero delay d on the control input $\mathbf{u}_{k-d}$, additional states can extend the initial process model (1) as $\mathbf{x}^u_{k,1} = \mathbf{u}_{k-1}, \mathbf{x}^u_{k,2} = \mathbf{u}_{k-2}, ..., \mathbf{x}^u_{k,d} = \mathbf{u}_{k-d}$ and arrive at a state-space model without delays, in which the additional states are measurable as past input samples. A delay in the original (i.e., related to (1)) states $\mathbf{x}_{k-d}$, can be treated similarly. In the end, however, all the above additional states will eventually depend on the IO samples $\mathbf{z}_k$ according to the proposed observability results. For unknown delays and unknown number of original states, the number of past IO samples can be increased, hoping that the IO measurements will capture the process's original state vector entirely.

B. Output reference model control problem definition

Let the discrete-time known open-loop stable minimum-phase state-space deterministic strictly causal ORM be

$$ORM: \begin{cases} \mathbf{x}^m_{k+1} = \mathbf{f}^m(\mathbf{x}^m_k, \mathbf{r}_k), \\ \mathbf{y}^m_k = \mathbf{g}^m(\mathbf{x}^m_k), \end{cases} \quad (12)$$

where $\mathbf{x}^m_k = [x^m_{k,1}, ..., x^m_{k,n_m}]^T \in \Omega_{X_m} \subset \Re^{n_m}$ is the state vector of the ORM, $\mathbf{r}_k = [r_{k,1}, ..., r_{k,p}]^T \in \Omega_{R_m} \subset \Re^p$ is the reference input signal, $\mathbf{y}^m_k = [y^m_{k,1}, ..., y^m_{k,p}]^T \in \Omega_{Y_m} \subset \Re^p$ is the ORM's output, $\mathbf{f}^m : \Omega_{X_m} \times \Omega_{R_m} \to \Omega_{X_m}$, $\mathbf{g}^m : \Omega_{X_m} \to \Omega_{Y_m}$ are *known* nonlinear mappings. Initial conditions are zero unless stated otherwise. Note that $\mathbf{r}_k, \mathbf{y}_k, \mathbf{y}^m_k$ have size p for square feedback CSs. If the ORM (12) is LTI, it is always possible to express the ORM as an IO LTI transfer matrix $\mathbf{y}^m_k = \mathbf{M}(z)\mathbf{r}_k$, where $\mathbf{M}(z)$ is commonly an asymptotically stable unit gain rational transfer matrix and $\mathbf{r}_k$ is the reference input that drives both the feedback CS and the ORM. We introduce an extended process comprising of the virtual process (11) coupled with the reference model (12). For this, we consider the reference input $\mathbf{r}_k$ as a set of measurable exogenous signals that evolve according to $\mathbf{r}_{k+1} = \mathbf{h}^m(\mathbf{r}_k)$, with unknown $\mathbf{h}^m : R^m \to R^m$ but measurable $\mathbf{r}_k$. Then the virtual extended process that consists of (11) and (12) is:

$$\mathbf{x}^E_{k+1} = \begin{pmatrix} \mathbf{z}_{k+1} \\ \mathbf{x}^m_{k+1} \\ \mathbf{r}_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{F}(\mathbf{z}_k, \mathbf{u}_k) \\ \mathbf{f}^m(\mathbf{x}^m_k, \mathbf{r}_k) \\ \mathbf{h}^m(\mathbf{r}_k) \end{pmatrix} = \mathbf{E}(\mathbf{x}^E_k, \mathbf{u}_k), \mathbf{x}^E_k \in \Omega_{X^E}. \quad (13)$$

The ORM tracking control problem is formulated in an optimal control framework. Let the infinite horizon cost function (c.f.) to be minimized starting with $\mathbf{x}_0$ be [31]

$$J^\infty_{MR}(\mathbf{x}^E_0, \boldsymbol{\theta}) = \sum_{k=0}^{\infty} \gamma^k \left\| \mathbf{y}^m_k(\mathbf{x}^E_k) - \mathbf{y}_k(\mathbf{x}^E_k, \boldsymbol{\theta}) \right\|_2^2 = \sum_{k=0}^{\infty} \gamma^k \left\| \boldsymbol{\varepsilon}_k(\mathbf{x}^E_k, \boldsymbol{\theta}) \right\|_2^2. \quad (14)$$

In (14), the discount factor $0 < \gamma \le 1$ sets the controller's horizon, $\gamma < 1$ is usually used in practice to guarantee learning convergence to optimal control. $\mathcal{U}_{MR} = \left\| \mathbf{y}^m_k(\mathbf{x}^E_k) - \mathbf{y}_k(\mathbf{x}^E_k) \right\|_2^2$ is the stage cost where measurable $\mathbf{y}_k$ depends via unknown $\mathbf{g}()$ on $\mathbf{x}_k$ and $\mathcal{U}_{MR}$ penalizes the deviation of $\mathbf{y}_k$ from the ORM's output $\mathbf{y}^m_k$. $\boldsymbol{\varepsilon}_k$ is the ORM tracking error and depends on $\mathbf{x}^E_k$, $\boldsymbol{\theta} \in \Re^{n_\theta}$ parameterizes a nonlinear state-feedback admissible controller [13] defined as $\mathbf{u}_k \overset{def}{=} C(\mathbf{x}^E_k, \boldsymbol{\theta})$, which used in (14) shows that all CS's trajectories depend on $\boldsymbol{\theta}$. Any stabilizing controller sequence (or controller) rendering a finite c.f. is called *admissible*. A finite J^∞_{MR} holds if $\boldsymbol{\varepsilon}_k$ is a square-summable sequence, ensured by an asymptotically stabilizing controller if $\gamma = 1$ or by a stabilizing controller if $\gamma < 1$. $J^\infty_{MR}(\boldsymbol{\theta})$ in (14) is the value function of using the controller $C(\boldsymbol{\theta})$. Let the optimal controller $\mathbf{u}^*_k = C^*(\mathbf{x}^E_k, \boldsymbol{\theta}^*)$ that minimizes (14) be

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J^\infty_{MR}(\mathbf{x}^E_0, \boldsymbol{\theta}). \quad (15)$$

Tracking a nonlinear ORM can also be used, however, tracking a linear one renders highly desirable indirect feedback linearization of the CS, where a linear CS's behavior generalizes well in wide operating ranges [4]. Then the ORM tracking control problem of this work should make $\mathcal{U}_{MR} \approx 0$ when $\mathbf{r}_k$ drives both the CS and the ORM.

Under classical control rules, the process time delay and non-minimum-phase (N-MP) character should be accounted in $\mathbf{M}(z)$. However, the NMP zeroes make $\mathbf{M}(z)$ non-invertible in addition to requiring their knowledge via identification, affecting the subsequent VRFT IO control design, motivating the MP assumption on the process.

Finally, since $\mathbf{x}^E_k$ contains $\mathbf{z}_k$ which itself contains past values of $\mathbf{u}_k$, it means that the controller $C(\mathbf{x}^E_k, \boldsymbol{\theta})$ is in fact equivalent to a dynamic feedback controller for (1).

III. SOLUTION TO THE ORM TRACKING PROBLEM

For unknown virtual extended process dynamics (13), minimization of (14) can be tackled using an iterative model-free approximate Value Iteration (IMF-AVI). A c.f. that extends $J^{\infty}_{MR}(\mathbf{x}^E_k)$ called the Q-function (or action-value function) is first defined for each state-action pair. Let the Q-function of acting as $\mathbf{u}_k$ in state $\mathbf{x}^E_k$ and then following the control (policy) $\mathbf{u}_k = C(\mathbf{x}^E_k)$ be

$$\Theta^C(\mathbf{x}^E_k, \mathbf{u}_k) = \mathcal{U}(\mathbf{x}^E_k, \mathbf{u}_k) + \gamma \Theta^C(\mathbf{x}^E_{k+1}, C(\mathbf{x}^E_{k+1})). \tag{17}$$

The optimal Q-function $\Theta^*(\mathbf{x}^E_k, \mathbf{u}^*_k)$ obeys Bellman's optimality equation

$$\Theta^*(\mathbf{x}^E_k, \mathbf{u}^*_k) = \min_{\mathbf{u}_k} \left(\mathcal{U}(\mathbf{x}^E_k, \mathbf{u}_k) + \gamma \Theta^*(\mathbf{x}^E_{k+1}, \mathbf{u}^*_{k+1}) \right), \tag{18}$$

where the optimal controller and optimal Q-functions are

$$\begin{aligned} \mathbf{u}^*_k &= C^*(\mathbf{x}^E_k) = \arg\min_C \Theta^C(\mathbf{x}^E_k, \mathbf{u}_k), \\ \Theta^*(\mathbf{x}^E_k, \mathbf{u}^*_k) &= \min_{\mathbf{u}_k} \Theta^*(\mathbf{x}^E_k, \mathbf{u}_k). \end{aligned} \tag{19}$$

Then, for $J^{\infty,*}_{MR}(\mathbf{x}^E_k) = \min_{\mathbf{u}} J^{\infty}_{MR}(\mathbf{x}^E_k, \mathbf{u})$ it follows that $J^{\infty,*}_{MR}(\mathbf{x}^E_k) = \Theta^*(\mathbf{x}^E_k, \mathbf{u}^*_k)$.

The optimal Q-function and optimal controller can be found with IMF-AVI, using, e.g., NNs as function approximators. NNs usage imply Q-function estimate and controller estimate parameterizations. The optimal Q-function and optimal controller estimates can be updated from the input-state transition samples collected from the process in various ways: in online/offline mode, batch or sample-by-sample update [24]–[28]. The proposed IMF-AVI entails a batch-wise approach.

IMF-AVI uses a dataset of collected input-state transition samples $\boldsymbol{\delta} = \{(\mathbf{x}^E_0, \mathbf{u}_0), (\mathbf{x}^E_1, \mathbf{u}_1), \ldots\}$ from the process (13). These transition samples from D can be collected using different strategies: randomly or controlled (using an already pre-stabilizing controller), or combinations of both (as it will be used later in this paper). The only requirement is that the pairs $(\mathbf{x}^E_k, \mathbf{u}_k)$ explore well the space $\Omega_{X^E} \times \Omega_U$. Then IMF-AVI starts with initial approximations of the Q-function (or critic) estimate $\Theta(\mathbf{x}^E_k, \mathbf{u}_k) : \Omega_{X^E} \times \Omega_U \to \Re$ and of the controller (*actor*, *policy*) estimate $\mathbf{C}(\mathbf{x}^E_k) : \Omega_{X^E} \to \Omega_U$. These approximations are here randomly initialized NNs, let them be denoted $\Theta_0(\mathbf{x}^E_k, \mathbf{u}_k)$ and $\mathbf{C}_0(\mathbf{x}^E_k)$. To converge to the optimal controller $\mathbf{C}^*(\mathbf{x}^E_k)$ estimate and optimal Q-function $\Theta^*(\mathbf{x}^E_k, \mathbf{u}_k)$ estimate, starting with $i = 0$, the steps of IMF-AVI at the current iteration i are:

ST1. The training data for the Q-function estimate consists of the input data $\{(\mathbf{x}^E_k, \mathbf{u}_k)\}$ and the target output data $\{\mathcal{U}_{MR}(\mathbf{x}^E_k, \mathbf{u}_k) + \gamma \Theta_i(\mathbf{x}^E_{k+1}, C_i(\mathbf{x}^E_{k+1}))\}$, for all successive state transitions $(\mathbf{x}^E_k, \mathbf{u}_k), (\mathbf{x}^E_{k+1}, \mathbf{u}_{k+1})$ recorded in $\boldsymbol{\delta}$. Here $J^C(\mathbf{x}^E_{k+1}) = \Theta_i(\mathbf{x}^E_{k+1}, C_i(\mathbf{x}^E_{k+1}))$ is the value function of using control $\mathbf{C}_i(\mathbf{x}^E_k)$. Then $\Theta_{i+1}(\mathbf{x}^E_k, \mathbf{u}_k)$ results after batch-training the NN over the input and target data, in a supervised learning framework.

ST2. The training data for the controller learning has the input data $\{\mathbf{x}^E_k\}$ and the target output data is $\{\mathbf{u}^* = \min_{\mathbf{u} \in \Lambda} \Theta_{i+1}(\mathbf{x}^E_k, \mathbf{u})\}$, meaning that the Q-function estimate $\Theta_{i+1}(\mathbf{x}^E_k, \mathbf{u})$ minimization is attempted. Here $\Lambda \subset \Omega_U$ is a finite set of control actions that uniformly cover Ω_U and can be obtained e.g. as grid partition points of Ω_U. A supervised batch training produces the improved controller $\mathbf{C}_{i+1}$.

ST3. Test $\mathbf{C}_{i+1}$ on a standard test scenario. If performance is acceptable then stop. Else, set $i = i + 1$ and go to *ST1*.

Step *ST1* corresponds to one back-up evaluation step of the current iteration controller. $\mathbf{C}_i$ used in step *ST1* assumes that the current iteration controller minimizes Θ_i, so that no explicit minimization $\min_{\mathbf{u} \in U} \Theta_i(\mathbf{x}^E_{k+1}, \mathbf{u})$ over a continuous action space is needed.

Concerning the acceptable performance test in *ST3*, after each iteration, the resulting controller $\mathbf{C}_{i+1}$ is tested against the initial controller used for data collection. If the initial controller is outperformed, the control performance improvement is achieved.

Note that in the step *ST2*, the minimization of $\Theta_{i+1}(\mathbf{x}^E_k, \mathbf{u})$ leads to discrete targets $\mathbf{u}^* = \min_{\mathbf{u} \in \Lambda} \Theta_{i+1}(\mathbf{x}^E_k, \mathbf{u})$ owing it to the grid search and the trained controller NN estimate output will in fact interpolate between these values. Thus, the resulting controller will output continuous actions. In this respect, the IMF-AVI implementation presented here differs from the approach [27], where the controller NN was trained by cascading the Q-function estimate NN and the controller NN, keeping the weights of the Q-function NN constant at the current iteration and setting the targets of the cascaded NN to zero {0} [31]. For IMF-AVI to converge, it does not need the explicit controller improvement step *ST2* if the target used in *ST1* is $\{\mathcal{U}_{MR}(\mathbf{x}^E_k, \mathbf{u}_k) + \gamma \min_{\mathbf{u} \in \Lambda} \Theta_i(\mathbf{x}^E_{k+1}, \mathbf{u})\}$, but then it is difficult to verify when has IMF-AVI converged to a better controller.

Following *Comment 7*, for the ORM tracking problem, the knowledge on process time delays is important since it establishes the relative degree of the process which has to be taken into account in the ORM, along with the MP character of the process necessary for ORM construction. Except for knowledge of the domain $\Omega_U \times \Omega_Y$, IMF-AVI does not use the process dynamics, justifying its model-free label.

IV. VALIDATION CASE STUDY

The case study aims validation of the ORM tracking problem on a twin rotor system angular position control [32] belonging to aerodynamic systems' class including quadrotors [33] . The azimuth (horizontal) motion behaves

as an integrator while the pitch (vertical) positioning is affected differently by the gravity for the up and down motions. Coupling between the two channels is present. A simplified deterministic continuous-time state-space model of this process is given as two coupled state-space sub-systems:

$$\begin{cases} \dot{\omega}_h = \left(sat\,(U_h) - M_h(\omega_h)\right)/2.7\cdot 10^{-5}, \\ \dot{K}_h = \left(0.216\,F_h(\omega_h)\cos\alpha_v - 0.058\,\Omega_h + 0.0178\,sat\,(U_v)\cos\alpha_v\right), \\ \Omega_h = K_h/\left(0.0238\cdot\cos^2\alpha_v + 3\cdot 10^{-3}\right), \\ \dot{\alpha}_h = \Omega_h, \end{cases} \tag{20}$$

$$\begin{cases} \dot{\omega}_v = \left(sat\,(U_v) - M_v(\omega_v)\right)/1.63\cdot 10^{-4}, \\ \dot{\Omega}_v = \dfrac{1}{0.03}\begin{pmatrix} 0.2\,F_v(\omega_v) - 0.0127\,\Omega_v - 0.0935\,\sin\alpha_v - \\ 9.28\cdot 10^{-6}\,\Omega_v\left|\omega_v\right| + 4.17\cdot 10^{-3}\,sat\,(U_h) - 0.05\cos\alpha_v \\ -\,0.021\,\Omega_h^2\sin\alpha_v\cos\alpha_v - 0.093\,\sin\alpha_v + 0.05 \end{pmatrix}, \\ \dot{\alpha}_v = \Omega_v, \end{cases}$$

where $sat\,(\,)$ is the saturation function on $[-1,1]$, $U_h = u_1$ is the azimuth motion control input, $U_v = u_2$ is the vertical motion control input, $\alpha_h\,(rad) = y_1 \in [-\pi,\pi]$ is the azimuth angle output, $\alpha_v\,(rad) = y_2 \in [-\pi/2,\pi/2]$ is the pitch angle output, other states being described in [32]. The nonlinear static characteristics obtained by polynomial fitting from experimental data are for $\omega_v,\omega_h \in (-4000\,;4000)$:

$$\begin{aligned} &M_v(\omega_v) = 9.05\cdot 10^{-12}\omega_v^3 + 2.76\cdot 10^{-10}\omega_v^2 + 1.25\cdot 10^{-4}\omega_v^1 + 1.66\cdot 10^{-4}, \\ &F_v(\omega_v) = -1.8\cdot 10^{-18}\omega_v^5 - 7.8\cdot 10^{-16}\omega_v^4 + 4.1\cdot 10^{-11}\omega_v^3 + \\ &2.7\cdot 10^{-8}\omega_v^2 + 3.5\cdot 10^{-5}\omega_v^1 - 0.014, \\ &M_h(\omega_h) = 5.95\cdot 10^{-13}\omega_h^3 - 5.05\cdot 10^{-10}\omega_h^2 + 1.02\cdot 10^{-4}\omega_h^1 + 1.61\cdot 10^{-3}, \\ &F_h(\omega_h) = -2.56\cdot 10^{-20}\omega_h^5 + 4.09\cdot 10^{-17}\omega_h^4 + 3.16\cdot 10^{-12}\omega_h^3 \\ &-\,7.34\cdot 10^{-9}\omega_h^2 + +2.12\cdot 10^{-5}\omega_h^1 + 9.13\cdot 10^{-3}. \end{aligned} \tag{21}$$

A zero-order hold on the inputs and a sampler on the outputs of (20) lead to an equivalent MP discrete-time model of relative degree 1 suitable for IO data collection. The process's dynamics are *not used* for learning the control.

A. Initial linear MIMO controller with model-free VRFT

An initial model-free MIMO multivariable IO controller is first found using model-free VRFT technique, as described in [22]. The ORM is of the form $\mathbf{M}(z) = \text{diag}\,(M_1(z), M_2(z))$ where $M_1(z), M_2(z)$ are the discretized versions of $M_1(s) = M_2(s) = 1/(3s+1)$ for a sampling period of $T_s = 0.1\,\text{sec}$. The VRFT prefilter is selected as $\mathbf{L}(z) = \mathbf{M}(z)$. A pseudo-random binary signal (PRBS) of amplitude $[-0.1;0.1]$ is used on both inputs to open-loop excite the pitch and azimuth dynamics simultaneously, as shown in Fig. 1.

An un-decoupling linear diagonal controller with the parameters computed by linear least-squares as in VRFT is

$$\mathbf{C}(z;\boldsymbol{\theta}) = \begin{bmatrix} P_{11}(z)/(1-z^{-1}) & 0 \\ 0 & P_{22}(z)/(1-z^{-1}) \end{bmatrix}, \tag{22}$$

$$P_{11}(z) = 2.9341 - 5.8689\,z^{-1} + 3.9303\,z^{-2} - 0.9173\,z^{-3} - 0.0777\,z^{-4},$$

$$P_{22}(z) = 0.6228 - 1.1540\,z^{-1} + 0.5467\,z^{-2}.$$

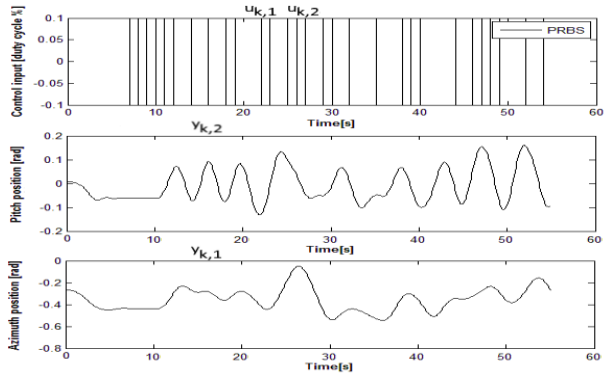

Fig. 1. Open-loop IO data from the process for VRFT controller tuning.

Note that VRFT controller tuning aims to indirectly minimize the undiscounted ($\gamma = 1$) J_{MR}^{∞} from (14). The same goal is pursued by subsequent NN state-feedback controller tuning for the virtual extended process.

B. Learning NN controllers with IMF-AVI

The next goal is to achieve ORM tracking by matching the closed-loop CS with the same reference model $\mathbf{M}(z) = \text{diag}\,(M_1(z), M_2(z))$. The diagonal $\mathbf{M}(z)$ asks for decoupling control to be ensured. The previous MIMO linear controller is used in closed-loop to collect significantly more IO data $\{\mathbf{u}_k, \mathbf{y}_k\}$ for 2000 s, as shown in Fig. 2 only for the first 400 s. The reference inputs $r_{k,1} \in [-3;3], r_{k,2} \in [-1.4;1.4]$ model sequences of steps of 3 s and 7 s, respectively, having uniform random amplitudes. The $C_{11}(z), C_{22}(z)$'s outputs are each perturbed every 5th sample with a uniform random number in $[-1;1]$ to ensure proper exploration, in order to visit many input-states-outputs combinations. The saturated versions of $sat\,(u_{k,1}), sat\,(u_{k,2}) \in [-1;1]$ are collected along $y_{k,1}, y_{k,2}, r_{k,1}, r_{k,2}$. The reference inputs drive the ORM

$$\begin{cases} x_{k+1,1}^m = 0.9672\,x_{k,1}^m + 0.03278\,r_{k,1}, \\ x_{k+1,2}^m = 0.9672\,x_{k,2}^m + 0.03278\,r_{k,2} \\ \mathbf{y}_k^m = [\,y_{k,1}^m \;\; y_{k,2}^m\,]^T = [\,x_{k,1}^m \;\; x_{k,2}^m\,]^T. \end{cases} \tag{23}$$

Then the states of the ORM (also outputs of the ORM) are also collected along the previous variables. ORM's states are known whereas for the process (20) only two states (also outputs) are measured. The intention here is to build the state of the extended state-space process (13) according to *Theorem 1*. Let this extended state be built from process IO dataset $\{\mathbf{u}_k, \mathbf{y}_k\}$ and from input-state data of the ORM be:

$$\mathbf{x}_k^E = [\underbrace{y_{k,1}\;y_{k,2}\;y_{k-1,1}\;y_{k-1,2}\;y_{k-2,1}\;y_{k-2,2}\;u_{k-1,1}\;u_{k-1,2}}_{(\mathbf{z}_k)^T}\;\underbrace{x_{k,1}^m\;x_{k,2}^m}_{(\mathbf{x}_k^m)^T}\;\underbrace{r_{k,1}\;r_{k,2}}_{(\mathbf{r}_k)^T}]^T \tag{24}$$

Note the introduction of the alias state $\mathbf{z}_k$ follows *Comments* 3 and 4. Different number of past IO samples can

be tried. Essentially, $\mathbf{x}_k^E$ built from IO data will generate the dataset $\boldsymbol{\delta} = \{(\mathbf{x}_0^E, \mathbf{u}_0), (\mathbf{x}_1^E, \mathbf{u}_1), ...\}$ used for IMF-AVI.

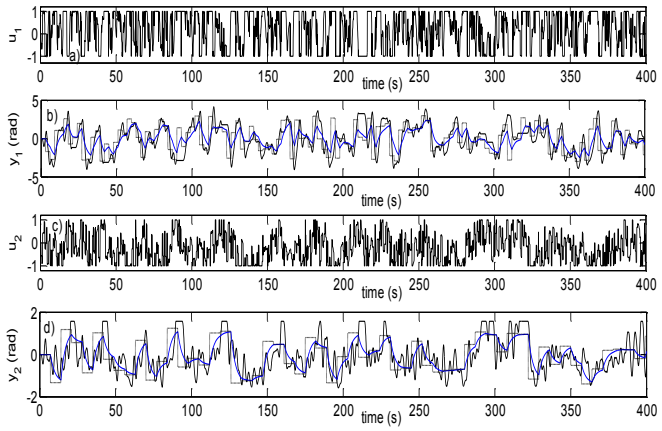

Fig. 2. IO data collection with the linear SISO controllers: a) $u_{k,1}$; b) $y_{k,1}$ (black), $y_{k,1}^m$ (blue), $r_{k,1}$ (black dotted); c) $u_{k,2}$; d) $y_{k,2}$ (black), $y_{k,2}^m$ (blue), $r_{k,2}$ (black dotted).

The controller NN (C-NN) estimate is a 12–6–2 (12 inputs because $\mathbf{x}_k^E \in \Re^{12}$, 6 neurons in the hidden layer and 2 outputs corresponding to $u_{k,1}, u_{k,2}$) with tanh activation function in the hidden layer and linear output activation. The Q-function NN (Q-NN) estimate is 14–30–1 with the same parameters as C-NN. Initial weights of both NNs are uniform random numbers with zero-mean and variance 0.3. Both NNs are to be trained using Levenberg-Marquardt for maximum 100 epochs. The available dataset is randomly divided into training (80%), validation (10%) and test data (10%). Early stopping during training is enforced after 10 increases of the training c.f. mean sum of squared errors (MSSE)) evaluated on the validation data. These training settings are going to be used on all subsequent scenarios.

When training NNs, other data preprocessing steps are also important such as data normalization (mean removal and amplitude scaling). As $\mathbf{x}_k^E$ is built, one observes that all the amplitudes of the elements of $\mathbf{x}_k^E$ are dictated by the amplitude of $\mathbf{y}_k$ and $\mathbf{u}_k$ (since their past values are used in $\mathbf{z}_k$) while $\mathbf{x}_k^m, \mathbf{r}_k$ have the magnitude of $\mathbf{y}_k$ because data was collected in closed-loop. Since the amplitudes of $\mathbf{y}_k$ and $\mathbf{u}_k$ are similar and centered on zero, no data normalization will be used. If in practice $\mathbf{y}_k$ and $\mathbf{u}_k$ have significantly different ranges, data normalization improves the NN training.

A discount $\gamma = 0.95$ will be used and the discrete set of control actions $\Lambda \subset \Omega_U$ used to minimize the Q-NN estimate in Step *S2* of the IMF-AVI is the Cartesian product of two identical sets of control actions, each containing 11 equally spaced values in $[-1;1]$ (i.e. $\{-1;-0.8;...,0.8,1\}$). Each iteration of the IMF-AVI produces a C-NN that is tested on the standard test scenario shown in Fig. 3 by measuring a finite-time normalized c.f. $J = 1/N \cdot J_{MR}^N$ for $N = 2000$ samples over 200 s. The IMF-AVI is iterated 200 times and all the stabilizing controllers that are better than the VRFT MIMO controller running on the standard test scenario descried in Fig. 3 (in terms of smaller *J*) are stored. The best C-NN is found at iteration 93 and renders $J = 0.0221$ ten times smaller than $J = 2.2713$ obtained with the VRFT MIMO controller running on the standard test scenario. Note that the state-feedback C-NN does not have integral component as the linear controllers. Ten times improvement is achieved with respect to the inferior linear controller.

Several regions where the two control channels interact through coupling are marked with vertical dotted rectangles in Fig. 3 and serve as disturbance rejection testing for the controllers. In these regions, the reference inputs to the CS do not switch simultaneously.

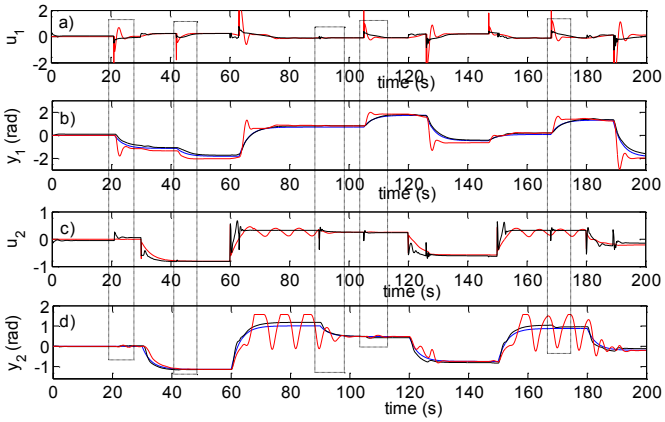

Fig. 3. NN controller and the linear MIMO controllers: a) $u_{k,1}^{VRFT}$ (red) and $u_{k,1}^{IMF-AVI}$ (black); b) $y_{k,1}^{VRFT}$ (red), $y_{k,1}^{IMF-AVI}$ (black), $y_{k,1}^m$ (blue); c) $u_{k,2}^{VRFT}$ (red) and $u_{k,2}^{IMF-AVI}$ (black); d) $y_{k,2}^{VRFT}$ (red), $y_{k,2}^{IMF-AVI}$ (black), $y_{k,1}^m$ (blue).

C. *Testing extended states of different orders*

A thorough case study is performed to assess the performance of IMF-AVI controllers used with extended states $\mathbf{x}_k^E$ that incorporate $\mathbf{z}_k$ built from different number of past values of $\{\mathbf{u}_k, \mathbf{y}_k\}$. Let

$$\begin{aligned}
\mathbf{z}_k^1 &= [y_{k,1}\ y_{k,2}\ y_{k-1,1}\ y_{k-1,2}\ y_{k-2,1}\ y_{k-2,2}\ u_{k-1,1}\ u_{k-1,2}]^T,\\
\mathbf{z}_k^2 &= [y_{k,1}\ y_{k,2}\ y_{k-1,1}\ y_{k-1,2}\ u_{k-1,1}\ u_{k-1,2}]^T,\\
\mathbf{z}_k^3 &= [y_{k,1}\ y_{k,2}\ y_{k-1,1}\ y_{k-1,2}\ y_{k-2,1}\ y_{k-2,2}\ u_{k-1,1}\ u_{k-1,2}\ u_{k-2,1}\ u_{k-2,2}]^T,\\
\mathbf{z}_k^4 &= [y_{k,1}\ y_{k,2}\ y_{k-1,1}\ y_{k-1,2}\ y_{k-2,1}\ y_{k-2,2}\ y_{k-3,1}\ y_{k-3,2}\ u_{k-1,1}\ u_{k-1,2}]^T,
\end{aligned} \tag{25}$$

be the alias states incorporated in the extended state-space vectors (used as inputs for the IMF-AVI controllers)

$$\begin{aligned}
&\mathbf{x}_k^{E,1} = [(\mathbf{z}_k^1)^T\ (\mathbf{x}_k^m)^T\ (\mathbf{r}_k)^T]^T, \mathbf{x}_k^{E,2} = [(\mathbf{z}_k^2)^T\ (\mathbf{x}_k^m)^T\ (\mathbf{r}_k)^T]^T,\\
&\mathbf{x}_k^{E,3} = [(\mathbf{z}_k^3)^T\ (\mathbf{x}_k^m)^T\ (\mathbf{r}_k)^T]^T, \mathbf{x}_k^{E,4} = [(\mathbf{z}_k^4)^T\ (\mathbf{x}_k^m)^T\ (\mathbf{r}_k)^T]^T.
\end{aligned} \tag{26}$$

The IMF-AVI NN controllers for $\mathbf{x}_k^{E,j}, j = \overline{1,4}$ are of sizes 12–6–2, 10–6–2, 14–6–2, 14–6–2, respectively.

The IMF-AVI is run on each dataset $\boldsymbol{\delta} = \{(\mathbf{x}_0^{E,j}, \mathbf{u}_0), (\mathbf{x}_1^{E,j}, \mathbf{u}_1), ...\}, j = \overline{1,4}$ for 200 iterations and the stabilizing controllers in terms of the value of J subject to $J < 2.27$ evaluated on the standard test scenario are saved. From these, the best controller in terms of smallest J is

selected. The value of $J = 2.27$ is the threshold performance of the linear VRFT MIMO controller run on the standard test scenario. All controllers with lower J are better. The smallest J values of the four controllers are $J_1 = 0.0221$, $J_2 = 0.0643$, $J_3 = 0.0204$, $J_4 = 0.0238$.

Natural conclusions are drawn: if not enough past IO samples of $\mathbf{u}_k, \mathbf{y}_k$ are used in $\mathbf{z}_k$, the control performance deteriorates since it corresponds to a partially observable process. A number of past IO samples of $\mathbf{u}_k, \mathbf{y}_k$ in $\mathbf{z}_k$ beyond the number ensuring full observability, does not improve the best achievable performance. Finally, we prove that starting from an initial stabilizing controller, performance can be dramatically improved using the IMF-AVI approach, which encourages its practical use. High control performance is obtained using only IO process data.

V. Conclusion

This paper has presented a successful application of a model-free Q-learning approach for position control of a nonlinear coupled aerodynamic system. The learned NN state-feedback control of a virtual state-space process that extends the initial process, indirectly controls the latter. The underlying data-based observability theory is confirmed and shows that IO control is achieved via virtual equivalent state-feedback control, using only IO data from the process. Future work will target the experimental validation on more nonlinear processes.

References

[1] M. C. Campi, A. Lecchini, and S. M. Savaresi, "Virtual reference feedback tuning: a direct method for the design of feedback controllers," *Automatica*, vol. 38, no. 8, pp. 1337–1346, Aug. 2002.

[2] H. Hjalmarsson, "Iterative feedback tuning - an overview," *Int. J. Adapt. Control Signal Process.*, vol. 16, pp. 373–395, June 2002.

[3] P. Janssens, G. Pipeleers, and J. L. Swevers, "Model-free iterative learning control for LTI systems and experimental validation on a linear motor test setup," in *Proc. 2011 American Control Conference*, San Francisco, CA, USA, 2011, pp. 4287–4292.

[4] M.-B. Radac and R.-E. Precup, "Optimal behaviour prediction using a primitive-based data-driven model-free iterative learning control approach," *Comput. Ind.*, vol. 74, pp. 95–109, Dec. 2015.

[5] R. Chi, Z.-S. Hou, S. Jin, and B. Huang, "An improved data-driven point-to-point ILC using additional on-line control inputs with experimental verification," *IEEE Trans. Syst., Man, Cybern.: Syst*., DOI: 10.1109/TSMC.2017.2693397, Apr. 2017.

[6] H. Abouaïssa, M. Fliess, and C. Join, "On ramp metering: towards a better understanding of ALINEA via model-free control," *Int. J. Control*, vol. 90, no. 5, pp. 1018–1026, May 2017.

[7] Z.-S. Hou, S. Liu, and T. Tian, "Lazy-learning-based data-driven model-free adaptive predictive control for a class of discrete-time nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 8, pp. 1914–1928, Aug. 2017.

[8] M.-B. Radac, R.-E. Precup, E. M. Petriu, and S. Preitl, "Iterative data-driven tuning of controllers for nonlinear systems with constraints," *IEEE Trans. Ind. Electron*., vol. 61, no. 11, pp. 6360–6368, 2014.

[9] C. Novara, S. Formentin, S. M. Savaresi, M. Milanese, "Data-driven design of two degree-of-freedom nonlinear controllers: The D2-IBC approach," *Automatica*, vol. 72, pp. 19–27, Oct. 2016.

[10] J. Bolder, S. Kleinendorst, and T. Oomen, "Data-driven multivariable ILC: enhanced performance by eliminating L and Q filters," *Int. J. Rob. Nonlin. Control*, doi: 10.1002/rnc.3611, 2016.

[11] Z. Wang, R. Lu, F. Gao, and D. Liu, "An indirect data-driven method for trajectory tracking control of a class of nonlinear discrete-time systems," *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4121–4129, May 2017.

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

[13] F. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circ. Syst. Mag.*, vol. 9, no. 3, pp. 76–105, Aug. 2009.

[14] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: an introduction," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, pp. 39–47, May 2009.

[15] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA, USA: Athena Scientific, 1996.

[16] J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks, "Adaptive dynamic programming," *IEEE Trans. Syst., Man, Cybern*., vol. 32, no. 2, pp. 140–153, May 2002.

[17] F. Lewis, and K. G. Vamvoudakis, "Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 41, no. 1, pp. 14–25, Feb. 2011.

[18] Z. Wang and D. Liu, "Data-based controllability and observability analysis of linear discrete-time systems," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2388–2392, Dec. 2011.

[19] Z. Ni, H. He, and X. Zhong, "Experimental studies on data-driven heuristic dynamic programming for POMDP," in *Frontiers of Intelligent Control and Information Processing*, D. Liu, C. Alippi, D. Zhao, and H. Zhang, Eds. Singapore: World Scientific Publishing, Lecture Notes in Computer Science, pp. 83–105, 2015.

[20] F. Ruelens, B. J. Claessens, S. Vandael, B. de Schutter, R. Babuška, and R. Belmans, "Residential demand response of thermostatically controlled loads using batch reinforcement learning," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2149–2159, Sep. 2017.

[21] V. Mnih, K. Kavukcouglu, D. Silver, A. A. Rusu, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb 2015.

[22] M.-B. Radac and R.-E. Precup, "Data-driven model-free slip control of anti-lock braking systems using reinforcement Q-learning," *Neurocomput.*, vol. 275, pp. 317–329, Jan. 2018.

[23] C. Watkins, and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, May 1992.

[24] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control," *Automatica*, vol. 43, no. 3, pp. 473–481, Mar. 2007.

[25] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *J. Mach. Learn. Res.*, vol. 6, pp. 503–556, Apr. 2005.

[26] R. Hafner and M. Riedmiller, "Reinforcement learning in feedback control. Challenges and benchmarks from technical process control," *Mach. Learn.*, vol. 84, no. 1, pp. 137–169, July 2011.

[27] M.-B. Radac, R.-E. Precup and R.-C. Roman, "Data-driven model reference control of MIMO vertical tank systems with model-free VRFT and Q-learning," *ISA Trans*., vol. 73, pp. 227–238, Feb. 2018.

[28] B. Luo, D. Liu, T. Huang, D. Wang, "Model-free optimal tracking control via critic-only Q-learning," *IEEE Trans. Neural Netw.*, vol. 27, no. 10, pp. 2134–2144, 2016.

[29] M. Fliess, J. Levine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *Int. J. Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[30] M. Milanese, C. Novara, K. Hsu, and K. Poolla, "The filter design from data (FD2) problem: Nonlinear set membership approach," *Automatica*, vol. 45, pp. 2350–2357, 2009.

[31] D. Liu, H. Javaherian, O. Kovalenko, and T. Huang, "Adaptive critic learning techniques for engine torque and air-fuel ratio control," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 988–993, Aug. 2008.

[32] *Two Rotor Aerodynamical System, User's Manual*. Krakow, Poland: Inteco Ltd., 2007.

[33] S. Zhou, M. K. Helva, and A. P. Schoellig, "Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking," in *Proc. 56th IEEE Conf. Dec. Control.*, Melbourne, Australia, 2017, pp. 5201–5207.