

Article

Learning Output Reference Model Tracking for Higher-Order Nonlinear Systems with Unknown Dynamics [†]

Mircea-Bogdan Radac *  and Timotei Lala

Department of Automation and Applied Informatics, Politehnica University of Timisoara, 2 Bd. V. Parvan, 300223 Timisoara, Romania; timotei.lala@student.upt.ro

* Correspondence: bogdan.ttl@gmail.com; Tel.: +40-256-403240; Fax: +40-256-403214

[†] This paper is an extended version of our paper published in the 27th Mediterranean Conference on Control and Automation (MED 2019).

Received: 1 May 2019; Accepted: 9 June 2019; Published: 12 June 2019

Abstract: This work suggests a solution for the output reference model (ORM) tracking control problem, based on approximate dynamic programming. General nonlinear systems are included in a control system (CS) and subjected to state feedback. By linear ORM selection, indirect CS feedback linearization is obtained, leading to favorable linear behavior of the CS. The Value Iteration (VI) algorithm ensures model-free nonlinear state feedback controller learning, without relying on the process dynamics. From linear to nonlinear parameterizations, a reliable approximate VI implementation in continuous state-action spaces depends on several key parameters such as problem dimension, exploration of the state-action space, the state-transitions dataset size, and a suitable selection of the function approximators. Herein, we find that, given a transition sample dataset and a general linear parameterization of the Q-function, the ORM tracking performance obtained with an approximate VI scheme can reach the performance level of a more general implementation using neural networks (NNs). Although the NN-based implementation takes more time to learn due to its higher complexity (more parameters), it is less sensitive to exploration settings, number of transition samples, and to the selected hyper-parameters, hence it is recommending as the de facto practical implementation. Contributions of this work include the following: VI convergence is guaranteed under general function approximators; a case study for a low-order linear system in order to generalize the more complex ORM tracking validation on a real-world nonlinear multivariable aerodynamic process; comparisons with an offline deep deterministic policy gradient solution; implementation details and further discussions on the obtained results.

Keywords: approximate dynamic programming; reinforcement learning; data-driven control; model-free control; reference trajectory tracking; output reference model; multivariable control; aerodynamic rotor system; neural networks; learning systems

1. Introduction

The output reference model (ORM) tracking problem is of significant interest in practice, especially for nonlinear systems control, since by selection of a linear ORM, feedback linearization is enforced on the controlled process. Then, the closed-loop control system can act linearly in a wide range and not only in the vicinity of an operating point. Subsequently, linearized control systems are then subjected to higher level learning schemes such as the Iterative Learning Control ones, with practical implications such as primitive-based learning [1] that can extrapolate optimal behavior to previously unseen tracking scenarios.

On another side, selection of a suitable ORM is not straightforward because of several reasons. The ORM has to be matched with the process bandwidth and with several process nonlinearities such as, e.g., input and output saturations. From classical control theory, dead-time and non-minimum-phase characters of the process cannot be compensated for and must be reflected in the ORM. Apart from this information that can be measured or inferred from working experience with the process, avoiding knowledge of the process' state transition function (process dynamics)—the most time consuming to identify and the most uncertain part of the process—in designing high performance control is very attractive in practice.

Reinforcement Learning (RL) has developed both from the artificial intelligence [2], and from classical control [3–7], where it is better known as Adaptive (Approximate, Neuro) Dynamic Programming (ADP). Certain ADP variants can ensure ORM tracking control without knowing the state-space (transition function) dynamics of the controlled process, which is of high importance in the practice of model-free (herein accepted as unknown dynamics) and data-driven control schemes that are able to compensate for poor modeling and process model uncertainty. Thus, ADP relies only on data collected from the process called state transitions. While plenty of mature ADP schemes already exist in the literature, tuning such schemes for a particular problem requires significant experience. Firstly, it must be specified whether ADP deals with continuous (infinite) or discrete (finite) state-action spaces. Then, the intended implementation will decide upon online/offline and/or adaptive/batch processing, the suitable selection of the approximator used for the extended cost function (called the Q-function) and/or for the controller. Afterwards, linear or nonlinear parameterizations are sought. Exploration of the state-action spaces is critical, as well as the hyperparameters of the overall learning scheme such as the number of transition samples, trading off exploration with exploitation, etc. Although successful stories on RL and ADP applied to large state-action spaces are reported mainly with artificial intelligence [8], in control theory, most approaches use low-order processes as representative case studies and mainly in linear quadratic regulator (LQR)-like settings (regulating states to zero). While, in an ADP, the reference input tracking control problem has been tackled before for linear time-invariant (LTI) processes by the name of Linear Quadratic Tracking (LQT) [9,10], the ORM tracking for nonlinear processes was rarely addressed [11,12].

The iterative model-free approximate Value Iteration (IMF-AVI) proposed in this work belongs to the family of batch-fitted Q-learning schemes [13,14] known as action-dependent heuristic dynamic programming (ADHDP) that are popular and representative ADP approaches, owing to their simplicity and model-free character. These schemes have 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) [12,15–23].

Concerning the exploration issue in ADP for control, a suitable exploration that covers as well as possible the state-action space is not trivially ensured. Randomly generated control input signals will almost surely fail to guide the exploration in the entire state-action space, at least not in a reasonable amount of time. Then, a priori designed feedback controllers can be used under a variable reference input serving to guide the exploration [24]. The existence of an initial feedback stabilizing controller, not necessarily of a high performance one, can accelerate the transition samples dataset collection under exploration. This allows for offline IMF-AVI based on large datasets, leading to improved convergence speed for high-dimensional processes. However, such input–output (IO) or input-state feedback controllers were traditionally not to be designed without using a process model, until the advent of data-driven model-free controller design techniques that have appeared from the field of control theory: Virtual Reference Feedback Tuning (VRFT) [25], Iterative Feedback Tuning [26], Model Free Iterative Learning Control [27–29], Model Free (Adaptive) Control [30,31], with representative applications [32–34]. This work shows a successful example of a model-free output feedback controller used to collect input-to-state transition samples from the process for learning state-feedback ADP-based ORM tracking control. Therefore it fits with the recent data-driven control [35–43] and reinforcement learning [12,44,45] applications.

The case study deals with the challenging ORM tracking control for a nonlinear real-world two-inputs–two-outputs aerodynamic system (TITOAS) having six natural states that are extended with four additional ones according to the proposed theory. The process uses aerodynamic thrust to create vertical (pitch) and horizontal (azimuth) motion. It is shown that IMF-AVI can be used to attain ORM tracking of first order lag type, despite the high order of the multivariable process, and despite the pitch motion being naturally oscillatory and the azimuth motion practically behaving close to an integrator. The state transitions dataset is collected under the guidance of an input–output (IO) feedback controller designed using model-free VRFT (© 2019 IEEE [12]).

As a main contribution, the paper is focused on a detailed comparison of the advantages and disadvantages of using linear and nonlinear parameterizations for the IMF-AVI scheme, while covering complete implementation details. To the best of authors' knowledge, the ORM tracking context with linear parameterizations was not studied before for high-order real-world processes. Moreover, theoretical analysis shows convergence of the IMF-AVI while accounting for approximation errors and explains for the robust learning convergence of the NN-based IMF-AVI. The results indicate that the nonlinearly parameterized NN-based IMF-AVI implementation should be *de facto* in practice since, although more time-consuming, it automatically manages the basis function selection, it is more robust to dataset size and exploration settings, and generally more well-suited for nonlinear processes with unknown dynamics. The main updates with respect to our paper [12] include: detailed IMF-AVI convergence proofs under general function approximators; a case study for a low order linear system in order to generalize to the more complex ORM tracking validation on the TITOAS process; comparisons with an offline Deep Deterministic Policy Gradient solution; more implementation details and further discussions on the obtained results.

Section 2 is dedicated to the formalization of the ORM tracking control problem, while Section 3 proposes a solution to this problem using an IMF-AVI approach. Section 4 validates the proposed approach on the TITOAS system, with concluding remarks presented in Section 5.

2. Output Model Reference Control for Unknown Dynamics Nonlinear Processes

2.1. The Process

A discrete-time nonlinear unknown open-loop stable state-space deterministic strictly causal process is defined as [12,46]

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

where k indexes the discrete time, $\mathbf{x}_k = [x_{k,1}, \dots, x_{k,n}]^\top \in \Omega_X \subset \mathbb{R}^n$ is the n -dimensional state vector, $\mathbf{u}_k = [u_{k,1}, \dots, u_{k,m_u}]^\top \in \Omega_U \subset \mathbb{R}^{m_u}$ is the control input signal, $\mathbf{y}_k = [y_{k,1}, \dots, y_{k,p}]^\top \in \Omega_Y \subset \mathbb{R}^p$ is the measurable controlled output, $\mathbf{f} : \Omega_X \times \Omega_U \mapsto \Omega_X$ is an *unknown* nonlinear system function continuously differentiable within its domain, $\mathbf{g} : \Omega_X \mapsto \Omega_Y$ is an unknown nonlinear continuously differentiable output function. Initial conditions are not accounted for at this point. Assume known domains $\Omega_X, \Omega_U, \Omega_Y$ are compact convex. Equation (1) is a general un-restrictive form for most controlled processes. The following assumptions common to the data-driven formulation are [12,46]:

Assumption 1 (A1). (1) is fully state controllable with measurable states.

Assumption 2 (A2). (1) is input-to-state stable on known domain $\Omega_U \times \Omega_X$.

Assumption 3 (A3). (1) is minimum-phase (MP).

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. Input to state stability (A2) is

necessary if open-loop input-state samples collection is intended to be used for state space control design. Assumption A2 can be relaxed if a stabilizing state-space controller is already available and used just for the purpose of input-state data collection. A3 is the least restrictive assumption and it is used in the context of the VRFT design of a feedback controller based on input–output process data. Although solutions exist to deal with nonminimum-phase systems processes, the MP assumption simplifies the VRFT design and the output reference model selection (to be introduced in the following section).

Comment 1. [46] Model (1) accounts for a wide range of processes including fixed 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 $\mathbf{x}_{k,1}^u = \mathbf{u}_{k-1}, \mathbf{x}_{k,2}^u = \mathbf{u}_{k-2}, \dots, \mathbf{x}_{k,d}^u = \mathbf{u}_{k-d}$ 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 states in (1), i.e., \mathbf{x}_{k-d} , are similarly treated.

2.2. Output Reference Model Control Problem Definition

Let the discrete-time known open-loop stable minimum-phase (MP) state-space deterministic strictly causal ORM be [12,46]

$$M : \{\mathbf{x}_{k+1}^m = \mathbf{f}^m(\mathbf{x}_k^m, \mathbf{r}_k), \mathbf{y}_k^m = \mathbf{g}^m(\mathbf{x}_k^m)\}, \tag{2}$$

where $\mathbf{x}_k^m = [x_{k,1}^m, \dots, x_{k,n}^m]^\top \in \Omega_{X_m} \subset \mathbb{R}^n$ is the ORM state, $\mathbf{r}_k = [r_{k,1}, \dots, r_{k,p}]^\top \in \Omega_{R_m} \subset \mathbb{R}^p$ is the reference input signal, $\mathbf{y}_k^m = [y_{k,1}^m, \dots, y_{k,p}^m]^\top \in \Omega_{Y_m} \subset \mathbb{R}^p$ is the ORM output, $\mathbf{f}^m : \Omega_{X_m} \times \Omega_{R_m} \mapsto \Omega_{X_m}$, $\mathbf{g}^m : \Omega_{X_m} \mapsto \Omega_{Y_m}$ are *known* nonlinear mappings. Initial conditions are zero unless otherwise stated. Notice that $\mathbf{r}_m, \mathbf{y}_k, \mathbf{y}_k^m$ are size p for square feedback control systems (CSs). If the ORM (2) is LTI, it is always possible to express the ORM as an IO LTI transfer function (t.f.) $\mathbf{M}(z)$ ensuring $\mathbf{y}_k^m = \mathbf{M}(z)\mathbf{r}_k$, where $\mathbf{M}(z)$ is commonly an asymptotically stable unit-gain rational t.f. 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 process (1) coupled with the ORM (2). For this, we consider the reference input \mathbf{r}_k as a set of measurable exogenous signals (possibly interpreted as a disturbance) that evolve according to $\mathbf{r}_{k+1} = \mathbf{h}^m(\mathbf{r}_k)$, with known nonlinear $\mathbf{h}^m : \mathbb{R}^m \mapsto \mathbb{R}^m$, where \mathbf{r}_k is measurable. Herein, $\mathbf{h}^m(\cdot)$ is a generative model for the reference input (© 2019 IEEE [12]).

The class of LTI generative models $\mathbf{h}^m(\cdot)$ has been studied before in [9] but it is a rather restrictive one. For example, reference inputs signals modeled as a sequence of steps of constant amplitude cannot be modeled by LTI generative models. A step reference input signal with constant amplitude over time can be modeled as $\mathbf{r}_{k+1} = \mathbf{r}_k$ with some initial condition \mathbf{r}_0 . On the other hand, a sinusoidal scalar reference input signal r_k can be modeled only through a second order state-space model. To see this, let the Laplace transform of $\cos(\omega t)\sigma(t)$ ($\sigma(t)$ is the unit step function) be $H(s) = \ell\{\cos(\omega t)\sigma(t)\}$ with the complex Laplace variable s . If $sH(s)$ is considered a t.f. driven by the unit step function with Laplace transform $\ell\{\sigma(t)\} = 1/s$, then the LTI discrete-time state-space associated with $sH(s)$ acting as a generative model for r_k is of the form

$$\begin{aligned} \mathbf{o}_{k+1} &= \mathbf{A}\mathbf{o}_k + \mathbf{B}\sigma_k, \\ r_k &= \mathbf{C}\mathbf{o}_k + \mathbf{D}\sigma_k, \end{aligned} \tag{3}$$

with known $\mathbf{A} \in \mathbb{R}^{2 \times 2}, \mathbf{B} \in \mathbb{R}^{2 \times 1}, \mathbf{C} \in \mathbb{R}^{1 \times 2}, \mathbf{D} \in \mathbb{R}, \mathbf{o}_0 = [0, 0]^\top$, and $\sigma_k = \{1, 1, 1, \dots\}$ is the discrete-time unit step function. The combination of $H(s)$ driven by the Dirac impulse with $\ell\{\delta(t)\}$ could also have been considered as a generative model. Based on the state-space model above, modeling p sinusoidal reference inputs $\mathbf{r}_k \in \Omega_{R_m} \subset \mathbb{R}^p$ requires $2p$ states. Generally speaking, the generative model of the reference input must obey the Markov property.

Consider next that the extended state-space model that consists of (1), (2), and the state-space generative model of the reference input signal is, in the most general form [12,46]:

$$\mathbf{x}_{k+1}^E = \begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{x}_{k+1}^m \\ \mathbf{r}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{f}^m(\mathbf{x}_k^m, \mathbf{r}_k) \\ \mathbf{h}^m(\mathbf{r}_k) \end{bmatrix} = \mathbf{E}(\mathbf{x}_k^E, \mathbf{u}_k), \mathbf{x}_k^E \in \Omega_{X^E}, \tag{4}$$

where \mathbf{x}_k^E is called the extended state vector. Note that the extended state-space fulfils the Markov property. 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 [6,12,46]

$$J_{MR}^\infty(\mathbf{x}_0^E, \boldsymbol{\theta}) = \sum_{k=0}^\infty \gamma^k \|\mathbf{y}_k^m(\mathbf{x}_k^E) - \mathbf{y}_k(\mathbf{x}_k^E, \boldsymbol{\theta})\|_2^2 = \sum_{k=0}^\infty \gamma^k \|\boldsymbol{\epsilon}_k(\mathbf{x}_k^E, \boldsymbol{\theta})\|_2^2. \tag{5}$$

In (5), the discount factor $0 < \gamma \leq 1$ sets the controller’s horizon, $\gamma < 1$ is usually used in practice to guarantee learning convergence to optimal control. $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^\top \mathbf{x}}$ is the Euclidean norm of the column vector \mathbf{x} . $v_{MR} = \|\mathbf{y}_k^m(\mathbf{x}_k^E) - \mathbf{y}_k(\mathbf{x}_k^E, \boldsymbol{\theta})\|_2^2$ is the stage cost where measurable \mathbf{y}_k depends via unknown \mathbf{g} in (1) on \mathbf{x}_k and v_{MR} penalizes the deviation of \mathbf{y}_k from the ORM’s output \mathbf{y}_k^m . In ORM tracking, the stage cost does not penalize the control effort with some positive definite function $W(\mathbf{u}_k) > 0$ since the ORM tracking instills an inertia on the CS that indirectly acts as a regularizer on the control effort. Secondly, if the reference inputs \mathbf{r}_k do not set to zero, the ORM’s outputs also do not. For most processes, the corresponding constant steady-state control will be non-zero, hence making $J_{MR}^\infty(\boldsymbol{\theta})$ infinite when $\gamma = 1$ [12,46].

Herein, $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ parameterizes a nonlinear state-feedback admissible controller [6] defined as $\mathbf{u}_k \triangleq \mathbf{C}(\mathbf{x}_k^E, \boldsymbol{\theta})$, which used in (4) 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 $\boldsymbol{\epsilon}_k$ is a square-summable sequence, ensured by an asymptotically stabilizing controller if $\gamma = 1$ or by a stabilizing controller if $\gamma < 1$. $J_{MR}^\infty(\boldsymbol{\theta})$ in (5) is the value function of using the controller $\mathbf{C}(\boldsymbol{\theta})$. Let the optimal controller $\mathbf{u}_k^* = \mathbf{C}(\mathbf{x}_k^E, \boldsymbol{\theta}^*)$ that minimizes (5) be [12,46]

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} J_{MR}^\infty(\mathbf{x}_0^E, \boldsymbol{\theta}). \tag{6}$$

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 [1]. Then the ORM tracking control problem of this work should make $v_{MR} \approx 0$ when \mathbf{r}_k drives both the CS and the ORM.

Under classical control rules, following *Comment 1*, the process time delay and non-minimum-phase (NMP) character should be accounted for in $\mathbf{M}(z)$. However, the NMP zeroes make $\mathbf{M}(z)$ non-invertible in addition to requiring their identification, thus placing a burden on the subsequent VRFT IO control design [47]. This motivates the MP assumption on the process.

Depending on the learning context, the user may select a piece-wise constant generative model for the reference input signal such as $\mathbf{r}_{k+1} = \mathbf{r}_k$, or a ramp-like model, a sine-like model, etc. In all cases, the states of the generative model are known, measurable and need to be introduced in the extended state vector, to fulfill the Markov property of the extended state-space model. In many practical applications, for the ORM tracking problem, the CS’s outputs are required to track the ORM’s outputs when both the ORM and the CS are driven by the piece-wise constant reference input signal expressed by a generative model of the form $\mathbf{r}_{k+1} = \mathbf{r}_k$. This generative model will be used subsequently in this paper for learning ORM tracking controllers. Obviously, the learnt solution will depend on the proposed reference input generative model, while changing this model requires re-learning.

3. Solution to the ORM Tracking Problem

For unknown extended process dynamics (4), minimization of (5) can be tackled using an iterative model-free approximate Value Iteration (IMF-AVI). A c.f. that extends $J_{MR}^\infty(\mathbf{x}_k^E)$ 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}_k^E and then following the control (policy) $\mathbf{u}_k = \mathbf{C}(\mathbf{x}_k^E)$ be [12,46]

$$Q^C(\mathbf{x}_k^E, \mathbf{u}_k) = v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q^C(\mathbf{x}_{k+1}^E, \mathbf{C}(\mathbf{x}_{k+1}^E)). \quad (7)$$

The optimal Q-function $Q^*(\mathbf{x}_k^E, \mathbf{u}_k)$ corresponding to the optimal controller obeys Bellman's optimality equation [12,46]

$$Q^*(\mathbf{x}_k^E, \mathbf{u}_k) = \min_{\mathbf{C}(\cdot)} \{v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q^C(\mathbf{x}_{k+1}^E, \mathbf{C}(\mathbf{x}_{k+1}^E))\}, \quad (8)$$

where the optimal controller and Q-functions are [12,46]

$$\mathbf{u}_k^* = \mathbf{C}^*(\mathbf{x}_k^E) = \arg \min_{\mathbf{C}} Q^C(\mathbf{x}_k^E, \mathbf{u}_k), Q^*(\mathbf{x}_k^E, \mathbf{u}_k) = \min_{\mathbf{C}(\cdot)} Q^C(\mathbf{x}_k^E, \mathbf{u}_k). \quad (9)$$

Then, for $J_{MR}^{\infty,*} = \min_{\mathbf{u}} J_{MR}^{\infty}(\mathbf{x}_k^E, \mathbf{u}_k)$ it follows that $J_{MR}^{\infty,*} = Q^*(\mathbf{x}_k^E, \mathbf{u}_k^* = \mathbf{C}^*(\mathbf{x}_k^E))$. Implying that finding Q^* is equivalent to finding the optimal c.f. $J_{MR}^{\infty,*}$.

The optimal Q-function and optimal controller can be found using either Policy Iteration (PoIt) or Value Iteration (VI) strategies. For continuous state-action spaces, IMF-AVI is one possible solution, using different linear and/or nonlinear parameterizations for the Q-function and/or for the controller. NNs are most widely used as nonlinearly parameterized function approximators. As it is well-known, VI alternates two steps: the Q-function estimate update step and the controller improvement step. Several Q-function parameterizations allow for explicit analytic calculation of the improved controller as the following optimization problem ((© 2019 IEEE [12])

$$\tilde{\mathbf{C}}(\mathbf{x}_k^E, \boldsymbol{\pi}) = \arg \min_{\mathbf{C}} Q^C(\mathbf{x}_k^E, \mathbf{u}_k, \boldsymbol{\pi}), \quad (10)$$

by directly minimizing $Q^C(\mathbf{x}_k^E, \mathbf{u}_k, \boldsymbol{\pi})$ w.r.t. \mathbf{u}_k , where the parameterization $\boldsymbol{\pi}$ has been moved from the controller into the Q-function. (10) is the controller improvement step specific to both the PoIt and VI algorithms. In these special cases, it is possible to eliminate the controller approximator and use only one for the Q-function Q . Then, given a dataset D of transition samples, $D = \{(\mathbf{x}_k^E, \mathbf{u}_k, \mathbf{x}_{k+1}^E)\}, k = 1, \dots, N$ the IMF-AVI amounts to solving the following optimization problem (OP) at every iteration j ((© 2019 IEEE [12])

$$\boldsymbol{\pi}_{j+1} = \arg \min_{\boldsymbol{\pi}} \sum_{k=1}^N (Q(\mathbf{x}_k^E, \mathbf{u}_k, \boldsymbol{\pi}) - v(\mathbf{x}_k^E, \mathbf{u}_k) - \gamma Q(\mathbf{x}_{k+1}^E, \tilde{\mathbf{C}}(\mathbf{x}_{k+1}^E, \boldsymbol{\pi}_j), \boldsymbol{\pi}_j))^2, \quad (11)$$

which is a Bellman residual minimization problem where the (usually separate) controller improvement step is now embedded inside the OP (11). More explicitly, for a linear parameterization $Q(\mathbf{x}_k^E, \mathbf{u}_k, \boldsymbol{\pi}) = \boldsymbol{\Phi}^\top(\mathbf{x}_k^E, \mathbf{u}_k) \boldsymbol{\pi}$ using a set of n_Φ basis functions of the form $\boldsymbol{\Phi}^\top(\mathbf{x}_k^E, \mathbf{u}_k) = [\Phi_1(\mathbf{x}_k^E, \mathbf{u}_k), \dots, \Phi_{n_\Phi}(\mathbf{x}_k^E, \mathbf{u}_k)]$, the least squares solution to (11) is equivalent to solving the following over-determined linear system of equations w.r.t. $\boldsymbol{\pi}_{j+1}$ in the least-squares sense ((© 2019 IEEE [12]):

$$\begin{bmatrix} \boldsymbol{\Phi}^\top(\mathbf{x}_1^E, \mathbf{u}_1) \\ \dots \\ \boldsymbol{\Phi}^\top(\mathbf{x}_N^E, \mathbf{u}_N) \end{bmatrix} \boldsymbol{\pi}_{j+1} = \begin{bmatrix} v(\mathbf{x}_1^E, \mathbf{u}_1) + \gamma \boldsymbol{\Phi}^\top(\mathbf{x}_2^E, \tilde{\mathbf{C}}(\mathbf{x}_2^E, \boldsymbol{\pi}_j)) \boldsymbol{\pi}_j \\ \dots \\ v(\mathbf{x}_N^E, \mathbf{u}_N) + \gamma \boldsymbol{\Phi}^\top(\mathbf{x}_{N+1}^E, \tilde{\mathbf{C}}(\mathbf{x}_{N+1}^E, \boldsymbol{\pi}_j)) \boldsymbol{\pi}_j \end{bmatrix}. \quad (12)$$

Concluding, starting with an initial parameterization $\boldsymbol{\pi}_0$, the IMF-AVI approach with linearly parameterized Q-function that allows explicit controller improvement calculation as in (10), embeds both VI steps into solving (12). Linearly parameterized IMF-AVI (LP-IMF-AVI) will be validated in the case study and compared to nonlinearly parameterized IMF-AVI (NP-IMF-AVI). Convergence of the generally formulated IMF-AVI is next analysed under approximation errors ((© 2019 IEEE [12]).

IMF-AVI Convergence Analysis with Approximation Errors for ORM Tracking

The proposed iterative model-free VI-based Q-learning Algorithm 1 consists of the next steps (© 2019 IEEE [12]).

Algorithm 1 VI-based Q-learning.

S1: Initialize controller C_0 and the Q-function value to $Q_0(\mathbf{x}_k^E, \mathbf{u}_k) = 0$, initialize iteration index $j = 1$

S2: Use one step backup equation for the Q-function as in (13)

S3: Improve the controller using the Equation (14)

S4: Set $j = j + 1$ and repeat steps S2, S3, until convergence

To be detailed as follows:

S1. Select an initial (not necessarily admissible) controller C_0 and an initialization value $Q_0(\mathbf{x}_k^E, \mathbf{u}_k) = 0$ of the Q-function. Initialize iteration $j = 1$.

S2. Use one step backup equation for the Q-function

$$\begin{aligned} Q_j(\mathbf{x}_k^E, \mathbf{u}_k) &= v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{C}_{j-1}(\mathbf{x}_{k+1}^E)) \\ &= \min_{\mathbf{u}} \{v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{u})\} \end{aligned} \quad (13)$$

S3. Improve the controller using the equation

$$\mathbf{C}_j(\mathbf{x}_k^E) = \arg \min_{\mathbf{u}} Q_j(\mathbf{x}_k^E, \mathbf{u}). \quad (14)$$

S4. Set $j = j + 1$ and repeat steps S2, S3, until convergence.

Lemma 1. (© 2019 IEEE [12]) For an arbitrary sequence of controllers $\{\kappa_j\}$ define the VI-update for extended c.f. ξ_j as [48]

$$\xi_{j+1}(\mathbf{x}_k^E, \mathbf{u}_k) = v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \xi_j(\mathbf{x}_{k+1}^E, \kappa_j(\mathbf{x}_{k+1}^E)). \quad (15)$$

If $Q_0(\mathbf{x}_k^E, \mathbf{u}_k) = \xi_0(\mathbf{x}_k^E, \mathbf{u}_k) = 0$, then $Q_j(\mathbf{x}_k^E, \mathbf{u}_k) \leq \xi_j(\mathbf{x}_k^E, \mathbf{u}_k)$.

Proof. It is valid that

$$\begin{aligned} Q_1(\mathbf{x}_k^E, \mathbf{u}_k) &= v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \overbrace{Q_0(\mathbf{x}_{k+1}^E, \mathbf{C}_0(\mathbf{x}_{k+1}^E))}^0 = \\ &= v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \overbrace{\xi_0(\mathbf{x}_{k+1}^E, \kappa_0(\mathbf{x}_{k+1}^E))}^0 = \xi_1(\mathbf{x}_k^E, \mathbf{u}_k). \end{aligned} \quad (16)$$

Meaning that $Q_1(\mathbf{x}_k^E, \mathbf{u}_k) \leq \xi_1(\mathbf{x}_k^E, \mathbf{u}_k)$. Assume by induction that $Q_{j-1}(\mathbf{x}_k^E, \mathbf{u}_k) \leq \xi_{j-1}(\mathbf{x}_k^E, \mathbf{u}_k)$. Then

$$\begin{aligned} Q_j(\mathbf{x}_k^E, \mathbf{u}_k) &= v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{C}_{j-1}(\mathbf{x}_{k+1}^E)) \leq \\ &\leq v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q_{j-1}(\mathbf{x}_{k+1}^E, \kappa_{j-1}(\mathbf{x}_{k+1}^E)) \leq \\ &\leq v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \xi_{j-1}(\mathbf{x}_{k+1}^E, \kappa_{j-1}(\mathbf{x}_{k+1}^E)) = \xi_j(\mathbf{x}_k^E, \mathbf{u}_k), \end{aligned} \quad (17)$$

which completes the proof. Here, it was used that $\mathbf{C}_{j-1}(\mathbf{x}_{k+1}^E)$ is the optimal controller for $Q_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{u}_k)$ per (14), then, for any other controller $\mathbf{C}(\mathbf{x}_{k+1}^E)$ (in particular it can also be $\kappa_{j-1}(\mathbf{x}_{k+1}^E)$) it follows that

$$Q_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{C}_{j-1}(\mathbf{x}_{k+1}^E)) \leq Q_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{C}(\mathbf{x}_{k+1}^E)). \quad (18)$$

□

Lemma 2. (© 2019 IEEE [12]) For the sequence $\{Q_j\}$ from (13), under controllability assumption A1, it is valid that:

- (1) $0 \leq Q_j(\mathbf{x}_k^E, \mathbf{u}_k) \leq B(\mathbf{x}_k^E, \mathbf{u}_k)$ with $B(\mathbf{x}_k^E, \mathbf{u}_k)$ an upper bound.
 (2) If there exists a solution $Q^*(\mathbf{x}_k^E, \mathbf{u}_k)$ to (8), then $0 \leq Q_j(\mathbf{x}_k^E, \mathbf{u}_k) \leq Q^*(\mathbf{x}_k^E, \mathbf{u}_k) \leq B(\mathbf{x}_k^E, \mathbf{u}_k)$.

Proof. For any fixed admissible controller $\eta(\mathbf{x}_k^E)$, $Q^\eta(\mathbf{x}_k^E, \mathbf{u}_k) = v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q^\eta(\mathbf{x}_{k+1}^E, \eta(\mathbf{x}_{k+1}^E))$ is the Bellman equation. Update (13) renders

$$\begin{aligned}
 Q_j(\mathbf{x}_k^E, \mathbf{u}_k) &= v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{C}_{j-1}(\mathbf{x}_{k+1}^E)) \stackrel{(18)}{\leq} \\
 &\stackrel{(18)}{\leq} v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q_{j-1}(\mathbf{x}_{k+1}^E, \eta(\mathbf{x}_{k+1}^E)) \\
 Q_{j-1}(\mathbf{x}_{k+1}^E, \eta(\mathbf{x}_{k+1}^E)) &= v(\mathbf{x}_{k+1}^E, \eta(\mathbf{x}_{k+1}^E)) + \gamma Q_{j-2}(\mathbf{x}_{k+2}^E, \mathbf{C}_{j-2}(\mathbf{x}_{k+2}^E)) \stackrel{(18)}{\leq} \\
 &\stackrel{(18)}{\leq} v(\mathbf{x}_{k+1}^E, \eta(\mathbf{x}_{k+1}^E)) + \gamma Q_{j-2}(\mathbf{x}_{k+2}^E, \eta(\mathbf{x}_{k+2}^E)) \\
 &\dots \\
 Q_1(\mathbf{x}_{k+j-1}^E, \eta(\mathbf{x}_{k+j-1}^E)) &= v(\mathbf{x}_{k+j-1}^E, \eta(\mathbf{x}_{k+j-1}^E)) + \gamma \overbrace{Q_0(\mathbf{x}_{k+j}^E, \mathbf{C}_0(\mathbf{x}_{k+j}^E))}^0
 \end{aligned} \tag{19}$$

Replacing from the last inequality towards the first it follows that

$$\begin{aligned}
 Q_j(\mathbf{x}_k^E, \mathbf{u}_k) &\leq v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma v(\mathbf{x}_{k+1}^E, \eta(\mathbf{x}_{k+1}^E)) + \dots + \gamma^{j-1} v(\mathbf{x}_{k+j-1}^E, \eta(\mathbf{x}_{k+j-1}^E)) \\
 &\leq v(\mathbf{x}_k^E, \mathbf{u}_k) + \sum_{j=1}^{\infty} \gamma^j v(\mathbf{x}_{k+j}^E, \eta(\mathbf{x}_{k+j}^E)) = Q^\eta(\mathbf{x}_k^E, \mathbf{u}_k),
 \end{aligned} \tag{20}$$

then, setting $Q^\eta(\mathbf{x}_k^E, \mathbf{u}_k) = B(\mathbf{x}_k^E, \mathbf{u}_k)$ proves the first part of Lemma 2.

Among all admissible controllers, the optimal one renders the Q-function with the lowest value therefore $Q^*(\mathbf{x}_k^E, \mathbf{u}_k) \leq Q^\eta(\mathbf{x}_k^E, \mathbf{u}_k) = B(\mathbf{x}_k^E, \mathbf{u}_k)$. If $\eta(\mathbf{x}_k^E) = \mathbf{C}^*(\mathbf{x}_k^E)$ is the optimal controller, it follows that $Q_j(\mathbf{x}_k^E, \mathbf{u}_k) \leq Q^*(\mathbf{x}_k^E, \mathbf{u}_k)$. Then the second part of Lemma 2 follows as $0 \leq Q_j(\mathbf{x}_k^E, \mathbf{u}_k) \leq Q^*(\mathbf{x}_k^E, \mathbf{u}_k) \leq B(\mathbf{x}_k^E, \mathbf{u}_k)$. \square

Theorem 1. ((© 2019 IEEE [12]) For the extended process (4) under A1, A2, with c.f. (5), with the sequences $\{\mathbf{C}_j\}$ and $\{Q_j(\mathbf{x}_k^E, \mathbf{u}_k)\}$ generated by the Q-learning Algorithm 1, it is true that:

- (1) $\{Q_j(\mathbf{x}_k^E, \mathbf{u}_k)\}$ is a non-decreasing sequence for which $Q_{j+1}(\mathbf{x}_k^E, \mathbf{u}_k) \geq Q_j(\mathbf{x}_k^E, \mathbf{u}_k)$ holds, $\forall j, \forall (\mathbf{x}_k^E, \mathbf{u}_k)$ and
 (2) $\lim_{j \rightarrow \infty} \mathbf{C}_j(\mathbf{x}_k^E) = \mathbf{C}^*(\mathbf{x}_k^E)$ and $\lim_{j \rightarrow \infty} Q_j(\mathbf{x}_k^E, \mathbf{u}_k) = Q^*(\mathbf{x}_k^E, \mathbf{u}_k)$.

Proof. Let $Q_0(\mathbf{x}_k^E, \mathbf{u}_k) = \zeta_0(\mathbf{x}_k^E, \mathbf{u}_k) = 0$ and assume the update

$$\zeta_j(\mathbf{x}_k^E, \mathbf{u}_k) = v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \zeta_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{C}_j(\mathbf{x}_{k+1}^E)). \tag{21}$$

By induction it is shown that $Q_1(\mathbf{x}_k^E, \mathbf{u}_k) \geq \zeta_0(\mathbf{x}_k^E, \mathbf{u}_k)$ since

$$Q_1(\mathbf{x}_k^E, \mathbf{u}_k) = v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q_0(\mathbf{x}_{k+1}^E, \mathbf{C}_0(\mathbf{x}_{k+1}^E)) = v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \cdot 0 \geq 0 = \zeta_0(\mathbf{x}_k^E, \mathbf{u}_k). \tag{22}$$

Assume next that $Q_j(\mathbf{x}_k^E, \mathbf{u}_k) \geq \zeta_{j-1}(\mathbf{x}_k^E, \mathbf{u}_k)$ and show that

$$\begin{aligned}
 Q_{j+1}(\mathbf{x}_k^E, \mathbf{u}_k) - \zeta_j(\mathbf{x}_k^E, \mathbf{u}_k) &= v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q_j(\mathbf{x}_{k+1}^E, \mathbf{C}_j(\mathbf{x}_{k+1}^E)) - v(\mathbf{x}_k^E, \mathbf{u}_k) - \\
 \gamma \zeta_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{C}_j(\mathbf{x}_{k+1}^E)) &= \gamma [Q_j(\mathbf{x}_{k+1}^E, \mathbf{C}_j(\mathbf{x}_{k+1}^E)) - \zeta_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{C}_j(\mathbf{x}_{k+1}^E))] \geq 0.
 \end{aligned} \tag{23}$$

The expression above leads to $Q_{j+1}(\mathbf{x}_k^E, \mathbf{u}_k) \geq \tilde{\zeta}_j(\mathbf{x}_k^E, \mathbf{u}_k)$. Since by Lemma 1 one has that $Q_j(\mathbf{x}_k^E, \mathbf{u}_k) \geq \tilde{\zeta}_j(\mathbf{x}_k^E, \mathbf{u}_k)$ it follows that $Q_{j+1}(\mathbf{x}_k^E, \mathbf{u}_k) \geq Q_j(\mathbf{x}_k^E, \mathbf{u}_k)$, proving first part of Theorem 1.

Any non-decreasing upper bounded sequence must have a limit, thus $\lim_{j \rightarrow \infty} \mathbf{C}_j = \mathbf{C}_\infty$ and $\lim_{j \rightarrow \infty} Q_j(\mathbf{x}_k^E, \mathbf{u}_k) = Q_\infty(\mathbf{x}_k^E, \mathbf{u}_k)$ with \mathbf{C}_∞ an admissible controller. For any admissible controller $\boldsymbol{\eta}(\mathbf{x}_k^E) = \mathbf{C}_\infty(\mathbf{x}_k^E)$ that is non-optimal if follows from (20) that $Q^*(\mathbf{x}_k^E, \mathbf{u}_k) \leq Q_\infty(\mathbf{x}_k^E, \mathbf{u}_k)$. Still, part 2 of Lemma 2 states that $Q_j(\mathbf{x}_k^E, \mathbf{u}_k) \leq Q^*(\mathbf{x}_k^E, \mathbf{u}_k)$ implying $Q_\infty(\mathbf{x}_k^E, \mathbf{u}_k) \leq Q^*(\mathbf{x}_k^E, \mathbf{u}_k)$. Then from $Q_\infty(\mathbf{x}_k^E, \mathbf{u}_k) \leq Q^*(\mathbf{x}_k^E, \mathbf{u}_k) \leq Q_\infty(\mathbf{x}_k^E, \mathbf{u}_k)$ it must hold true that $Q_\infty(\mathbf{x}_k^E, \mathbf{u}_k) = Q^*(\mathbf{x}_k^E, \mathbf{u}_k)$ and $\mathbf{C}_\infty(\mathbf{x}_k^E) = \mathbf{C}^*(\mathbf{x}_k^E)$ which proves the second part of Theorem 1. \square

Comment 2. (© 2019 IEEE [12]) (13) is practically solved in the sense of the OP (11) (either as a linear or nonlinear regression) using a batch (dataset) of transition samples collected from the process using any controller, that is in *off-policy* mode. While the controller improvement step (14) can be solved either as a regression or explicitly analytically when the expression of $Q_j(\mathbf{x}_k^E, \mathbf{u}_k)$ allows it. Moreover, (13) and (14) can be solved batch-wise in either online or offline mode. When the batch of transition samples is updated with one sample at a time, the VI-scheme becomes adaptive.

Comment 3. (© 2019 IEEE [12]) Theorem 1 proves the VI-based learning convergence of the sequence of Q-functions $\lim_{j \rightarrow \infty} Q_j(\mathbf{x}_k^E, \mathbf{u}_k) = Q^*(\mathbf{x}_k^E, \mathbf{u}_k)$ assuming that the true Q-function parameterization is used. In practice, this is rarely possible, such as, e.g., in the case of LTI systems. For general nonlinear processes of type (1), different function approximators are employed for the Q-function, most commonly using NNs. Then the convergence of the VI Q-learning scheme is to a suboptimal controller and to a suboptimal Q-function, owing to the approximation errors. A generic convergence proof of the learning scheme under approximation errors is next shown, accounting for general Q-function parameterizations [49].

Let the IMF-AVI Algorithm 2 consist of the steps (© 2019 IEEE [12]).

Algorithm 2 IMF-AVI.

- S1: Initialize controller $\tilde{\mathbf{C}}_0$ and Q-function value $\tilde{Q}_0(\mathbf{x}_k^E, \mathbf{u}_k) = 0, \forall(\mathbf{x}_k^E, \mathbf{u}_k)$. Initialize iteration $j = 1$
 - S2: Update the approximate Q-function using Equation (24)
 - S3: Improve the approximate controller using Equation (25)
 - S4: Set $j = j + 1$ and repeat steps S2, S3, until convergence
-

To be detailed as follows:

S1. Select an initial (not necessarily admissible) controller $\tilde{\mathbf{C}}_0$ and an initialization value $\tilde{Q}_0(\mathbf{x}_k^E, \mathbf{u}_k) = 0, \forall(\mathbf{x}_k^E, \mathbf{u}_k)$ of the Q-function. Initialize iteration $j = 1$.

S2. Use the update equation for the approximate Q-function

$$\begin{aligned} \tilde{Q}_j(\mathbf{x}_k^E, \mathbf{u}_k) &= v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \tilde{Q}_{j-1}(\mathbf{x}_{k+1}^E, \tilde{\mathbf{C}}_{j-1}(\mathbf{x}_{k+1}^E)) + \delta_j(\mathbf{x}_k^E, \mathbf{u}_k) \\ &= \min_{\mathbf{u}} \{v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \tilde{Q}_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{u})\} + \delta_j \end{aligned} \tag{24}$$

S3. Improve the approximate controller using

$$\tilde{\mathbf{C}}_j(\mathbf{x}_k^E) = \arg \min_{\mathbf{u}} \tilde{Q}_j(\mathbf{x}_k^E, \mathbf{u}) \tag{25}$$

S4. Set $j = j + 1$ and repeat steps S2, S3, until convergence.

Comment 4. (© 2019 IEEE [12]) In Algorithm 2, the sequences $\{\tilde{\mathbf{C}}_j(\mathbf{x}_k^E)\}$ and $\{\tilde{Q}_j(\mathbf{x}_k^E, \mathbf{u})\}$ are approximations of the true sequences $\{\mathbf{C}_j(\mathbf{x}_k^E)\}$ and $\{Q_j(\mathbf{x}_k^E, \mathbf{u}_k)\}$. Since the true Q-function and controller parameterizations are not generally known, (24) must be solved in the sense of the OP (11) with respect to the unknown \tilde{Q}_j , in order to minimize the residuals δ_j at each iteration. If the true parameterizations of the Q-function and of the controller were known, then $\delta_j = 0$ and the IMF-AVI

updates (24), (25) coincide with (13), (14), respectively. Next, let the following assumption hold. A4. ([12]) There exist two positive scalar constants $\underline{\psi}, \bar{\psi}$ such that $0 < \underline{\psi} \leq 1 \leq \bar{\psi} < \infty$, ensuring

$$\min_{\mathbf{u}} \{ \underline{\psi} v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \tilde{Q}_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{u}) \} \leq \tilde{Q}_j(\mathbf{x}_k^E, \mathbf{u}_k) \leq \min_{\mathbf{u}} \{ \bar{\psi} v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \tilde{Q}_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{u}) \}. \tag{26}$$

Comment 5. (© 2019 IEEE [12]) Inequalities from (26) account for nonzero positive or negative residuals δ_j , i.e., for the approximation errors in the Q-function, since $\tilde{Q}_j(\mathbf{x}_k^E, \mathbf{u}_k)$ can over- or under-estimate $\min_{\mathbf{u}} \{ v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \tilde{Q}_{j-1}(\mathbf{x}_{k+1}^E, \mathbf{u}) \}$ in (24). $\underline{\psi}, \bar{\psi}$ can span large intervals ($\underline{\psi}$ close to 0 and $\bar{\psi}$ very large). The hope is that, if $\underline{\psi}, \bar{\psi}$ are close to 1—meaning low approximation errors—then the entire IMF-AVI process preserves $\delta_j \approx 0$. In practice, this amounts to using high performance approximators. For example, with NNs, adding more layers and more neurons, enhances the approximation capability and theoretically reduces the residuals in (24).

Theorem 2. Let the sequences $\{\tilde{C}_j(\mathbf{x}_k^E)\}$ and $\{\tilde{Q}_j(\mathbf{x}_k^E, \mathbf{u}_k)\}$ evolve as in (24), (25), the sequences $\{C_j(\mathbf{x}_k^E)\}$ and $\{Q_j(\mathbf{x}_k^E, \mathbf{u}_k)\}$ evolve as in (13), (14). Initialize $\tilde{Q}_0(\mathbf{x}_k^E, \mathbf{u}_k) = Q_0(\mathbf{x}_k^E, \mathbf{u}_k) = 0, \forall(\mathbf{x}_k^E, \mathbf{u}_k)$ and let A3 hold. Then

$$\underline{\psi} Q_j(\mathbf{x}_k^E, \mathbf{u}_k) \leq \tilde{Q}_j(\mathbf{x}_k^E, \mathbf{u}_k) \leq \bar{\psi} Q_j(\mathbf{x}_k^E, \mathbf{u}_k) \tag{27}$$

Proof. (© 2019 IEEE [12]) First, the development proceeds by induction for the left inequality. For $j = 0$ it is clear that $\underline{\psi} Q_0(\mathbf{x}_k^E, \mathbf{u}_k) \leq \tilde{Q}_0(\mathbf{x}_k^E, \mathbf{u}_k)$. For $j = 1$, (13) produces $Q_1(\mathbf{x}_k^E, \mathbf{u}_k) = v(\mathbf{x}_k^E, \mathbf{u}_k)$ and left-hand side of (26) reads $\min_{\mathbf{u}} \{ \underline{\psi} v(\mathbf{x}_k^E, \mathbf{u}_k) + 0 \} \leq \tilde{Q}_1(\mathbf{x}_k^E, \mathbf{u}_k)$. Then $\underline{\psi} Q_1(\mathbf{x}_k^E, \mathbf{u}_k) \leq \tilde{Q}_1(\mathbf{x}_k^E, \mathbf{u}_k)$. Next assume that

$$\underline{\psi} Q_j(\mathbf{x}_k^E, \mathbf{u}_k) \leq \tilde{Q}_j(\mathbf{x}_k^E, \mathbf{u}_k) \tag{28}$$

holds at iteration j . Based on (28) used in (26), it is valid that

$$\min_{\mathbf{u}} \{ \underline{\psi} v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \underline{\psi} Q_j(\mathbf{x}_{k+1}^E, \mathbf{u}) \} \leq \min_{\mathbf{u}} \{ \underline{\psi} v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \tilde{Q}_j(\mathbf{x}_{k+1}^E, \mathbf{u}) \} \leq \tilde{Q}_{j+1}(\mathbf{x}_k^E, \mathbf{u}_k). \tag{29}$$

Notice from (29) that

$$\min_{\mathbf{u}} \{ \underline{\psi} v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma \underline{\psi} Q_j(\mathbf{x}_{k+1}^E, \mathbf{u}) \} = \underline{\psi} \min_{\mathbf{u}} \{ v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q_j(\mathbf{x}_{k+1}^E, \mathbf{u}) \} \stackrel{(13)}{=} \underline{\psi} Q_{j+1}(\mathbf{x}_k^E, \mathbf{u}_k) \tag{30}$$

From (29), (30) it follows that $\underline{\psi} Q_{j+1}(\mathbf{x}_k^E, \mathbf{u}_k) \leq \tilde{Q}_{j+1}(\mathbf{x}_k^E, \mathbf{u}_k)$ proving the left side of (27) by induction. The right side of (27) is shown similarly, proving Theorem 2. \square

Comment 6. (© 2019 IEEE [12]) Theorem 2 shows that the trajectory of $\{\tilde{Q}_j(\mathbf{x}_k^E, \mathbf{u}_k)\}$ closely follows that of $\{Q_j(\mathbf{x}_k^E, \mathbf{u}_k)\}$ in a bandwidth set by $\underline{\psi}, \bar{\psi}$. It does not ensure that $\{\tilde{Q}_j(\mathbf{x}_k^E, \mathbf{u}_k)\}$ converges to a steady-state value, but in the worst case, it oscillates around $Q^*(\mathbf{x}_k^E, \mathbf{u}_k) = \lim_{j \rightarrow \infty} Q_j(\mathbf{x}_k^E, \mathbf{u}_k)$ in a band that can be made arbitrarily small by using powerful approximators. By minimizing over \mathbf{u}_k both sides of (27), similar conclusions result for the controller sequence $\{\tilde{C}_j(\mathbf{x}_k^E)\}$ that closely follows $\{C_j(\mathbf{x}_k^E)\}$.

In the following Section, the IMF-AVI is validated on two illustrative examples. The provided theoretical analysis supports and explains the robust learning performance of the nonlinearly parameterized IMF-AVI with respect to the linearly parameterized one.

4. Validation Case Studies

4.1. ORM Tracking for a Linear Process

A first introductory simple example of IMF-AVI for the ORM tracking of a first-order process motivates the more complex validation for the TITOAS process and offers insight into how the IMF-AVI solution scales up with the higher-order processes.

Let a scalar discrete-time process discretized at $T_s = 0.1s$ be $x_{k+1} = 0.8187x_k + 0.1813u_k$. The continuous-time ORM $M(s) = 1/(s + 1)$ ZOH discretized at the same T_s leads to the extended process equivalent to (4), (output equations also given):

$$\begin{cases} x_{k+1} = 0.8187x_k + 0.1813u_k, \\ x_{k+1}^m = 0.9048x_k^m + 0.09516r_k, \\ r_{k+1} = r_k, \\ y_k = x_k, y_k^m = x_k^m, \end{cases} \Leftrightarrow \mathbf{x}_{k+1}^E = \mathbf{E}(\mathbf{x}_k^E, u_k) \quad (31)$$

where a piece-wise constant reference input generative model is introduced to ensure that the extended process (31) has full measurable state.

For data collection, the ORM's output y_k^m is collected along with: u_k , x_k and the reference input r_k . The measurable extended state vector is then $\mathbf{x}_k^E = [x_k, x_k^m, r_k]^T$. A discretized version of an integral controller with t.f. $0.25/s$ at sampling period $T_s = 0.1s$ closes the loop of the control system and asymptotically stabilizes it, while calculating the control input u_k based on the feedback error $e_k = r_k - y_k$. This CS setup is used for collecting transition samples of the form $D = \{(\mathbf{x}_k^E, \mathbf{u}_k, \mathbf{x}_{k+1}^E)\}$. Data is collected for 500 s, with normally distributed random reference inputs having variance $\sigma_r^2 = 0.0951$, modeled as piece-wise constant steps that change their values every 20 s. Normally distributed white noise having variance $\sigma_u^2 = 4.96$ is added on the command u_k at every time step to ensure a proper exploration by visiting as many combinations of states and actions as possible. Exploration has a critical role in the success of the IMF-AVI. A higher amplitude additive noise on u_k increases the chances of converging the approximate VI approach. The state transitions data collection is shown in Figure 1 for the first 1000 samples (100 s).

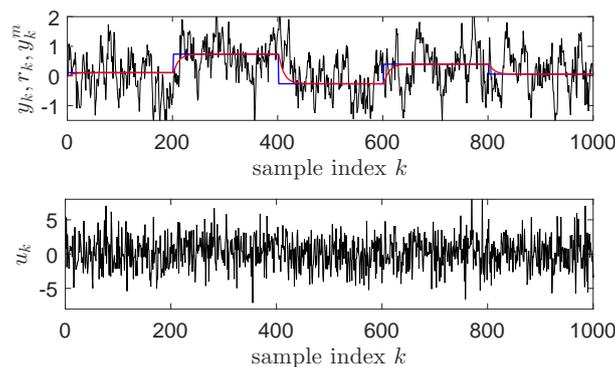


Figure 1. Closed-loop state transitions data collection for Example 1: (top) y_k (black), r_k (blue), y_k^m (red); (bottom) u_k .

Notice that a reference input modeled as a sequence of constant amplitude steps is used for exploration purposes, for which it may not be possible to write $r_{k+1} = h^m(r_k)$ as a generative model. To solve this, all transition samples that correspond to the switching times of the reference input are eliminated, therefore, $r_{k+1} = r_k$ can be considered as the piece-wise constant generative model of the reference input.

The control objective is to minimize $J_{MR}^\infty(\theta)$ from (5) using the stage cost $v(\mathbf{x}_k^E) = (y_k^m - y_k)^2$ (where the outputs obviously depend on the extended states as per (31)), with the discount factor

$\gamma = 0.9$. Thus the overall objective is to find the optimal state-feedback controller $\mathbf{u}_k^* = \mathbf{C}^*(\mathbf{x}_k^E)$ that makes the feedback CS match the ORM.

The Q-function is linearly parameterized as $Q(\mathbf{x}_k^E, u_k) = \Phi^\top(\mathbf{x}_k^E, u_k)\boldsymbol{\pi}$, with the quadratic basis functions vector constructed by the unique terms of the Kronecker product of all input arguments of $Q(\mathbf{x}_k^E, u_k)$ as

$$\Phi^\top(\mathbf{x}_k^E, u_k) = [x_k^2, (x_k^m)^2, r_k^2, u_k^2, x_k x_k^m, x_k r_k, x_k u_k, x_k^m r_k, x_k^m u_k, r_k u_k] \tag{32}$$

with $\boldsymbol{\pi} \in \mathbb{R}^{10}$. The controller improvement step equivalent to explicitly minimizing the Q-function w.r.t. the control input u_k is $\tilde{u}_k^* = \tilde{\mathbf{C}}^*(\mathbf{x}_k^E, \boldsymbol{\pi}_j) = -\frac{1}{2\pi_{4,j}}[\pi_{7,j}, \pi_{9,j}, \pi_{10,j}]\mathbf{x}_k^E$. This improved linear-in-the-state controller is embedded in the linear system of equations (12) that is solved for every iteration of IMF-AVI. Each iteration produces a new $\boldsymbol{\pi}_{j+1}$ that is tested on a test scenario where the uniformly random reference inputs have amplitude $r_k \in [-1; 1]$ and switch every 10 s. The ORM tracking performance is then measured by the Euclidean vector norm $\|y_k^m - y_k\|_2$ while $\|\boldsymbol{\pi}_{j+1} - \boldsymbol{\pi}_j\|_2$ serves as a stopping condition when it drops below a prescribed threshold. The practically observed convergence process is shown in Figure 2 over the first 400 iterations, with $\|\boldsymbol{\pi}_{j+1} - \boldsymbol{\pi}_j\|_2$ still decreasing after 1000 iterations. While $\|y_k^m - y_k\|_2$ is very small right from the first iterations, making the process output practically overlap with the ORM’s output.

Comment 7. For LTI processes with an LQR-like c.f., an LTI ORM and an LTI generative reference input model, linear parameterizations of the extended Q-function of the form $Q(\mathbf{x}_k^E, \mathbf{u}_k) = \Phi^\top(\mathbf{x}_k^E, \mathbf{u}_k)\boldsymbol{\pi}$ is the well-known [9] form $Q(\mathbf{x}_k^E, \mathbf{u}_k) = [(\mathbf{x}_k^E)^\top, (\mathbf{u}_k)^\top]\mathbf{P}[(\mathbf{x}_k^E)^\top, (\mathbf{u}_k)^\top]^\top$ of the quadratic Q-function, with parameter $\boldsymbol{\pi} = \text{vec}(\mathbf{P})$ being the vectorized form of the symmetric positive-definite matrix \mathbf{P} and the basis function vector $\Phi^\top(\mathbf{x}_k^E, \mathbf{u}_k)$ is obtained by the nonrepeatable terms of the Kronecker product of all the Q-function input arguments.

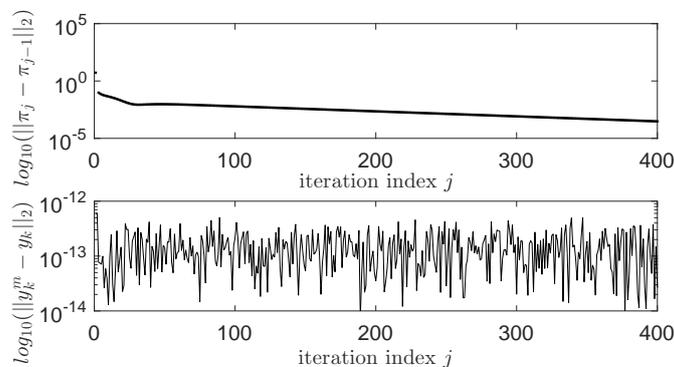


Figure 2. Convergence results of the linearly parameterized iterative model-free approximate Value Iteration (LP-IMF-AVI) for the linear process example.

4.2. IMF-AVI on the Nonlinear TITOAS Aerodynamic System

The ORM tracking problem on the more challenging TITOAS angular position control [50] (Figure 3) is aimed next. 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 [12,46]:

$$\begin{cases} \dot{\omega}_h = (\text{sat}(U_h) - M_h(\omega_h))/2.5 \cdot 10^{-5}, \\ \dot{K}_h = 0.216F_h(\omega_h)\cos(\alpha_v) - 0.058\Omega_h + 0.0178\text{sat}(U_v)\cos(\alpha_v), \\ \Omega_h = K_k / (0.0238\cos^2(\alpha_v) + 3 \cdot 10^{-3}), \\ \dot{\alpha}_h = \Omega_h, \\ \dot{\omega}_v = (\text{sat}(U_v) - M_v(\omega_v))/1.63 \cdot 10^{-4}, \\ \dot{\Omega}_v = \frac{1}{0.03} \begin{pmatrix} 0.2F_v(\omega_v) - 0.0127\Omega_v - 0.0935\sin\alpha_v + \\ -9.28 \cdot 10^{-6}\Omega_v|\omega_v| + 4.17 \cdot 10^{03}\text{sat}(U_h) - 0.05\cos\alpha_v + \\ -0.021\Omega_h^2\sin\alpha_v\cos\alpha_v - 0.093\sin\alpha_v + 0.05 \end{pmatrix} \\ \dot{\alpha}_v = \Omega_v, \end{cases} \tag{33}$$

where $\text{sat}()$ 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, $\alpha_h(\text{rad}) = y_1 \in [-\pi, \pi]$ is the azimuth angle output, $\alpha_v(\text{rad}) = y_2 \in [-\pi/2, \pi/2]$ is the pitch angle output, other states being described in [11,48]. The nonlinear static characteristics obtained by polynomial fitting from experimental data are for $\omega_v, \omega_h \in (-4000;4000)$ [46]:

$$\begin{aligned} M_v(\omega_v) &= 9.05 \times 10^{-12}\omega_v^3 + 2.76 \times 10^{-10}\omega_v^2 + 1.25 \times 10^{-4}\omega_v + 1.66 \times 10^{-4}, \\ F_v(\omega_v) &= -1.8 \times 10^{-18}\omega_v^5 - 7.8 \times 10^{-16}\omega_v^4 + 4.1 \times 10^{-11}\omega_v^3 + 2.7 \times 10^{-8}\omega_v^2 \\ &\quad + 3.5 \times 10^{-5}\omega_v - 0.014, \\ M_h(\omega_h) &= 5.95 \times 10^{-13}\omega_h^3 - 5.05 \times 10^{-10}\omega_h^2 + 1.02 \times 10^{-4}\omega_h + 1.61 \times 10^{-3}, \\ F_h(\omega_h) &= -2.56 \times 10^{-20}\omega_h^5 + 4.09 \times 10^{-17}\omega_h^4 + 3.16 \times 10^{-12}\omega_h^3 - 7.34 \times 10^{-9}\omega_h^2 \\ &\quad + 2.12 \times 10^{-5}\omega_h + 9.13 \times 10^{-3}. \end{aligned} \tag{34}$$

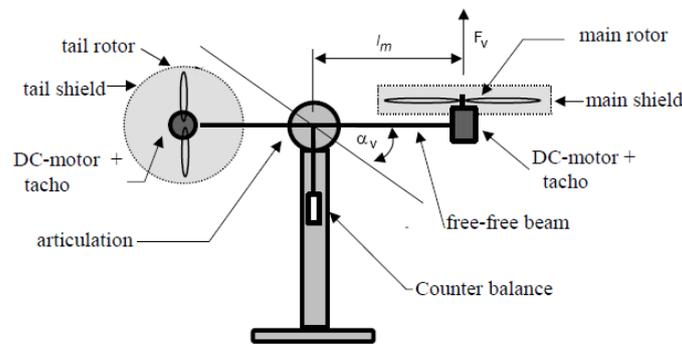


Figure 3. The two-inputs–two-outputs aerodynamic system (TITOAS) experimental setup [50].

A zero-order hold on the inputs and a sampler on the outputs of (33) lead to an equivalent MP discrete-time model of sampling time $T_s = 0.1\text{s}$ and of relative degree 1 (one), suitable for input-state data collection

$$P : \begin{cases} \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \\ \mathbf{y}_k = \mathbf{g}(\mathbf{x}_k) = [\alpha_{h,k}, \alpha_{v,k}]^\top, \end{cases} \tag{35}$$

where $\mathbf{x}_k = [\omega_{h,k}, \Omega_{h,k}, \alpha_{h,k}, \omega_{v,k}, \Omega_{v,k}, \alpha_{v,k}]^\top \in \mathbb{R}^6$ and $\mathbf{u}_k = [u_{k,1}, u_{k,2}]^\top \in \mathbb{R}^2$. The process' dynamics will not be used for learning the control in the following.

4.3. Initial Controller with Model-Free VRFT

An initial model-free multivariable IO controller is first found using model-free VRFT, as described in [11,24,32]. The ORM is $\mathbf{M}(z) = \text{diag}(M_1(z), M_2(z))$ where $M_1(z), M_2(z)$ are the

discrete-time counterparts of $M_1(s) = M_2(s) = 1/(3s + 1)$ obtained for a sampling period of $T_s = 0.1$ s. The VRFT prefilter is chosen as $\mathbf{L}(z) = \mathbf{M}(z)$. A pseudo-random binary signal (PRBS) of amplitude $[-0.1; 0.1]$ is used on both inputs $u_{k,1}, u_{k,2}$ to open-loop excite the pitch and azimuth dynamics simultaneously, as shown in Figure 4. The IO data $\{\tilde{\mathbf{u}}_k, \tilde{\mathbf{y}}_k\}$ is collected with low-amplitude zero-mean inputs $u_{k,1}, u_{k,2}$, in order to maintain the process linearity around the mechanical equilibrium, such that to fit the linear VRFT design framework (© 2019 IEEE [12]).

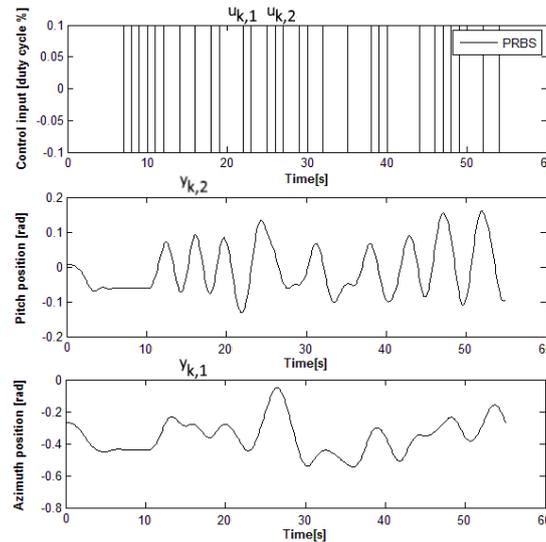


Figure 4. Open-loop input–output (IO) data from the two-inputs–two-outputs aerodynamic system (TITOAS) for Virtual Reference Feedback Tuning (VRFT) controller tuning [46].

An un-decoupling linear output feedback error diagonal controller with the parameters computed by the VRFT approach is [12,46]

$$\mathbf{C}(z, \boldsymbol{\theta}) = \begin{bmatrix} P_{11}(z)/(1 - z^{-1}) & 0 \\ 0 & P_{22}(z)/(1 - z^{-1}) \end{bmatrix}, \tag{36}$$

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

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

where the parameter vector $\boldsymbol{\theta}$ groups all the coefficients of $P_{11}(z), P_{22}(z)$. Controller (36) is obtained for $\boldsymbol{\theta}$ as the least squares minimizer of $J_{VR}(\boldsymbol{\theta}) = \sum_{k=1}^N \|\tilde{\mathbf{u}}_k^L - \mathbf{C}(z, \boldsymbol{\theta})\tilde{\mathbf{e}}_k^L\|_2^2$ where $\tilde{\mathbf{u}}_k^L = \mathbf{L}(z)\tilde{\mathbf{u}}_k = \mathbf{L}(z)[\tilde{u}_{k,1}, \tilde{u}_{k,2}]^T, \tilde{\mathbf{e}}_k^L = \mathbf{L}(z)\tilde{\mathbf{e}}_k = \mathbf{L}(z)[\tilde{e}_{k,1}, \tilde{e}_{k,2}]^T, [\tilde{e}_{k,1}, \tilde{e}_{k,2}]^T = (\mathbf{M}^{-1}(z) - \mathbf{I}_2)[\tilde{y}_{k,1}, \tilde{y}_{k,2}]^T$. Here, $J_{VR}(\boldsymbol{\theta})$ is an approximation of the c.f. J_{MR}^∞ from (5) obtained for $\gamma = 1$. The controller (36) will then close the feedback control loop as in $\mathbf{u}_k = \mathbf{C}(z, \boldsymbol{\theta})(\mathbf{r}_k - \mathbf{y}_k)$.

Notice that, by formulation, the VRFT controller tuning aims to minimize the undiscounted ($\gamma = 1$) J_{MR}^∞ from (5), but via the output feedback controller (36) that processes the feedback control error $\mathbf{e}_k = \mathbf{r}_k - \mathbf{y}_k$. The same goal to minimize (5) is pursued by the subsequent IMF-AVI design of a state-feedback controller tuning for the extended process. Nonlinear (in particular, linear) state-feedback controllers can also be found by VRFT as shown in [24,32], to serve as initializations for the IMF-AVI, or possibly, even for Polt-like algorithms. However, should this not be necessary, IO feedback controllers are much more data-efficient, requiring significantly less IO data to obtain stabilizing controllers.

4.4. Input–State–Output Data Collection

ORM tracking is intended by making the closed loop CS match the same ORM $\mathbf{M}(z) = \text{diag}(M_1(z), M_2(z))$. With the linear controller (36) used in closed-loop to stabilize the

process, input–state–output data is collected for 7000 s. The reference inputs with amplitudes $r_{k,1} \in [-2; 2], r_{k,2} \in [-1.4; 1.1]$ model successive steps that switch their amplitudes uniformly random at 17 s and 25 s, respectively. On the outputs $u_{k,1}, u_{k,2}$ of both controllers $C_{11}(z), C_{22}(z)$, an additive noise is added at every 2nd sample as a uniform random number in $[-1.6; 1.6]$ for $C_{11}(z)$ and in $[-1.7; 1.7]$ for $C_{22}(z)$. These additive disturbances provide an appropriate exploration, visiting many combinations of input–states–outputs. The computed controller outputs are saturated to $[-1; 1]$, then sent to the process. The reference inputs $r_{k,1}, r_{k,2}$ drive the ORM [12,46]:

$$\begin{cases} x_{k+1,1}^m = 0.9672x_{k,1}^m + 0.03278r_{k,1}, \\ x_{k+1,2}^m = 0.9672x_{k,2}^m + 0.03278r_{k,2}, \\ \mathbf{y}_k^m = [y_{k,1}^m, y_{k,2}^m]^\top = [x_{k,1}^m, x_{k,2}^m]^\top. \end{cases} \quad (37)$$

Then the states of the ORM (also outputs of the ORM) are also collected along with the states and control inputs of the process, to build the process extended state (4). Let the extended state be:

$$\mathbf{x}_k^E = \underbrace{[x_{k,1}^m, x_{k,2}^m]}_{(\mathbf{x}_k^m)^\top}, \underbrace{[r_{k,1}, r_{k,2}]}_{\mathbf{r}_k^\top}, (\mathbf{x}_k)^\top. \quad (38)$$

Essentially, the collected \mathbf{x}_k^E and \mathbf{u}_k builds the transitions dataset $D = \{(\mathbf{x}_1^E, \mathbf{u}_1, \mathbf{x}_2^E), \dots, (\mathbf{x}_{70000}^E, \mathbf{u}_{70000}, \mathbf{x}_{70001}^E)\}$ for $N = 70,000$, used for the IMF-AVI implementation. After collection, an important processing step is the data normalization. Some process states are scaled in order to ensure that all states are inside $[-1; 1]$. The scaled process state is $\tilde{\mathbf{x}}_k = [\omega_{h,k}/7200, 25 \cdot \Omega_{h,k}, \alpha_{h,k}, \omega_{v,k}/3500, 40 \cdot \Omega_{v,k}, \alpha_{v,k}]^\top \in \mathbb{R}^6$ and $\mathbf{u}_k = [u_{k,1}, u_{k,2}]^\top \in \mathbb{R}^2$. Other variables such as the reference inputs, the ORM states and the saturated process inputs already have values inside $[-1; 1]$. The normalized state is eventually used for state feedback. Collected transition samples are shown in Figure 5 only for the process inputs and outputs, ORM’s outputs and reference inputs, for the first 400 s (4000 samples) out of 7000 s [12].

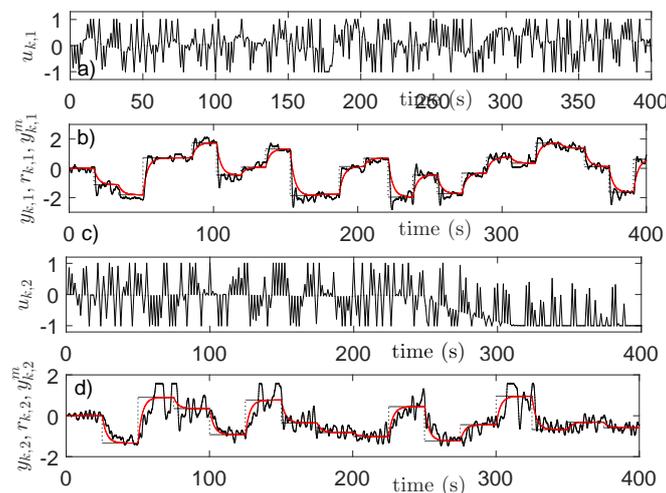


Figure 5. IO data collection with the linear controller: (a) $u_{k,1}$; (b) $y_{k,1}$ (black), $y_{k,1}^m$ (red), $r_{k,1}$ (black dotted); (c) $u_{k,2}$; (d) $y_{k,2}$ (black), $y_{k,2}^m$ (red), $r_{k,2}$ (black dotted).

Note that the reference input signals $r_{k,1}, r_{k,2}$ used as sequences of constant amplitude steps for ensuring good exploration, do not have a generative model that obeys the Markov assumption. To avoid this problem, the piece-wise constant reference input generative model $\mathbf{r}_{k+1} = \mathbf{r}_k$ is employed by eliminating from the dataset D all the transition samples that correspond to switching reference inputs instants (i.e., when at least one of $r_{k,1}, r_{k,2}$ switches) (© 2019 IEEE [12]).

4.5. Learning State-Feedback Controllers with Linearly Parameterized IMF-AVI

Details of the LP-IMF-AVI applied to the ORM tracking control problem are next provided. The stage cost is defined $v(\mathbf{x}_k^E) = (y_{k,1} - y_{k,1}^m)^2 + (y_{k,2} - y_{k,2}^m)^2$ and the discount factor in J_{MR}^∞ is $\gamma = 0.95$. The Q-function is linearly parameterized using the basis functions [12]

$$\Phi^\top(\mathbf{x}_k^E, \mathbf{u}_k) = [(x_{k,1}^m)^2, (x_{k,2}^m)^2, r_{k,1}^2, \dots, x_{k,6}^2, u_{k,1}^2, u_{k,2}^2, x_{k,1}^m x_{k,2}^m, x_{k,1}^m r_{k,1}, \dots, x_{k,1}^m u_{k,2}, x_{k,2}^m r_{k,1}, \dots, u_{k,1} u_{k,2}] \in \mathbb{R}^{78}. \tag{39}$$

This basis functions selection is inspired by the shape of the quadratic Q-function resulting from LTI processes with LQR-like penalties (see *Comment 7*). It is expected to be a sensible choice since the TITOAS process is a nonlinear one, therefore the quadratic Q-function may under-parameterize the true Q-function. Nevertheless, its computational advantage incentivizes the testing of such a solution. Notice that the controller improvement step at each iteration of the LP-IMF-AVI is based on explicit minimization of the Q-function. Solving the linear system of equations resulting after setting the derivative of $Q(\mathbf{x}_k^E, \mathbf{u}_k)$ w.r.t. \mathbf{u}_k equal to zero, it is obtained that (© 2019 IEEE [12])

$$\begin{aligned} \tilde{\mathbf{u}}_k^* &= \begin{bmatrix} u_{k,1}^* \\ u_{k,2}^* \end{bmatrix} = \tilde{\mathbf{C}}^*(\mathbf{x}_k^E, \boldsymbol{\pi}_j) = \begin{bmatrix} 2\pi_{j,11} & \pi_{j,78} \\ \pi_{j,78} & 2\pi_{j,12} \end{bmatrix}^{-1} \begin{bmatrix} F_1(\mathbf{x}_k^E) \\ F_2(\mathbf{x}_k^E) \end{bmatrix}, \\ F_1(\mathbf{x}_k^E) &= \pi_{j,22}x_{k,1}^m + \pi_{j,32}x_{k,2}^m + \pi_{j,41}r_{k,1} + \pi_{j,49}r_{k,2} + \pi_{j,56}x_{k,1} + \pi_{j,62}x_{k,2} + \pi_{j,67}x_{k,3} + \pi_{j,71}x_{k,4} + \pi_{j,74}x_{k,5} + \pi_{j,76}x_{k,6} = \boldsymbol{\pi}_{j,1}^\top \mathbf{x}_k^E, \\ F_2(\mathbf{x}_k^E) &= \pi_{j,23}x_{k,1}^m + \pi_{j,33}x_{k,2}^m + \pi_{j,42}r_{k,1} + \pi_{j,50}r_{k,2} + \pi_{j,57}x_{k,1} + \pi_{j,63}x_{k,2} + \pi_{j,68}x_{k,3} + \pi_{j,72}x_{k,4} + \pi_{j,75}x_{k,5} + \pi_{j,77}x_{k,6} = \boldsymbol{\pi}_{j,2}^\top \mathbf{x}_k^E. \end{aligned} \tag{40}$$

The improved controller is embedded in the system (12) of 70,000 linear equations with 78 unknowns corresponding to the parameters of $\boldsymbol{\pi}_{j+1} \in \mathbb{R}^{78}$. This linear system (12) is solved in least squares sense, with each of the 50 iterations of the LP-IMF-AVI. The practical convergence results are shown in Figure 6 for $\|\boldsymbol{\pi}_{j+1} - \boldsymbol{\pi}_j\|_2$ and for the ORM tracking performance in terms of a normalized c.f. $J_{test} = 1/N(\|y_{k,1} - y_{k,1}^m\|_2 + \|y_{k,2} - y_{k,2}^m\|_2)$ measured for samples over 200 s in the test scenario displayed in Figure 7. The test scenario consists of a sequence of piece-wise constant reference inputs that switch at different moments of time for the azimuth and pitch ($y_{k,1}$ and $y_{k,2}$, respectively), to illustrate the existing coupling behavior between the two control channels and the extent to which the learned controller manages to achieve the decoupled behavior requested but the diagonal ORM (© 2019 IEEE [12]).

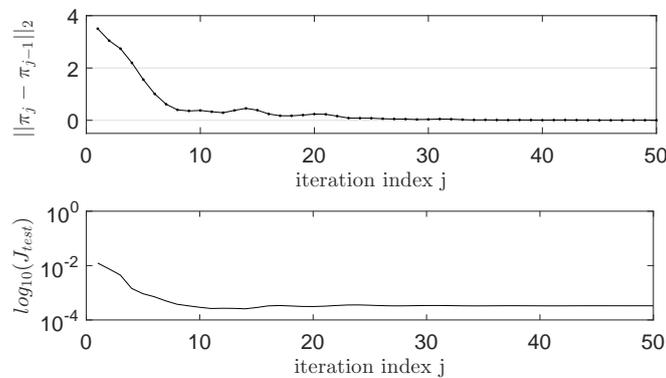


Figure 6. The LP-IMF-AVI convergence on TITOAS (© 2019 IEEE [12]).

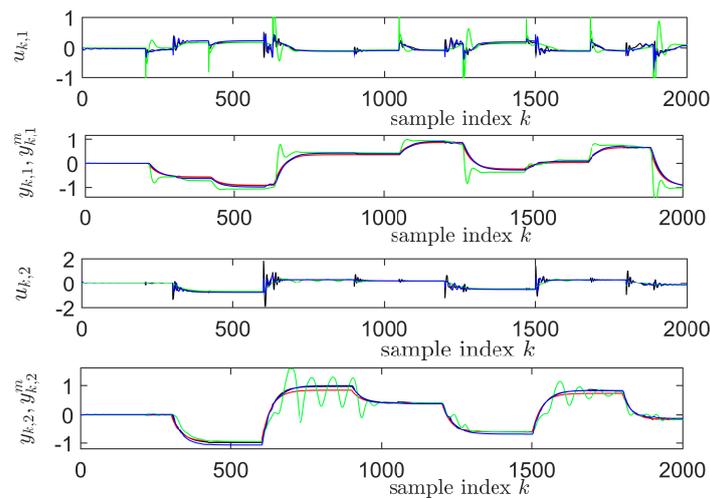


Figure 7. The IMF-AVI convergence on TITOAS: $y_{k,1}^m, y_{k,2}^m$, (red); $u_{k,1}, u_{k,2}, y_{k,1}, y_{k,2}$ for LP-IMF-AVI (black), for NP-IMF-AVI with NNs (blue), for the initial VRFT controller used for transitions collection (green) (© 2019 IEEE [12]).

The best LP-IMF-AVI controller found over the 50 iterations results in $J_{test} = 0.0017$ (tracking results in black lines in Figure 7), which is more than 6 times lower than the tracking performance of the VRFT controller used for transition samples collection, for which $J_{test} = 0.0103$ (tracking results in green lines in Figure 7) (© 2019 IEEE [12]). The convergence of the LP-IMF-AVI parameters is depicted in Figure 8.

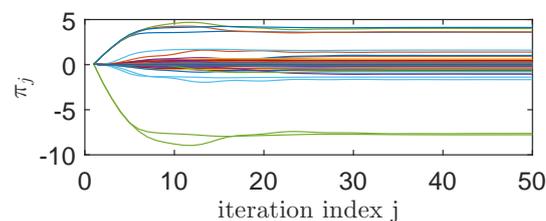


Figure 8. The LP-IMF-AVI parameters convergence.

4.6. Learning State-Feedback Controllers with Nonlinearly Parameterized IMF-AVI Using NNs

The previous LP-IMF-AVI for ORM tracking control learning scheme is next challenged by a NP-IMF-AVI implemented with NNs. In this case, two NNs are used to approximate the Q-function and the controller (the latter is sometimes avoidable, see the comments later on in this sub-section). The procedure follows the NP-IMF-AVI implementation described in [24,51]. The same dataset of transition samples is used as was previously used for the LP-IMF-AVI. Notice that the NN-based implementation is widely used in the reinforcement learning-based approach of ADP and is generally more scalable to problems of high dimension.

The controller NN (C-NN) estimate is a 10–3–2 (10 inputs because $\mathbf{x}_k^E \in \mathbb{R}^{10}$, 3 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 12–25–1 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 scaled conjugate gradient for a maximum of 500 epochs. The available dataset is randomly divided into training (80%) and validation data (20%). Early stopping during training is enforced after 10 increases of the training c.f. mean sum of squared errors (MSE) evaluated on the validation data. MSE is herein, for all networks, the default performance function used in training (© 2019 IEEE [12]).

The NP-IMF-AVI proposed herein consists of two steps for each iteration j . The first one calculates the targets for the NN $Q(\mathbf{x}_k^E, \mathbf{u}_k, \boldsymbol{\pi}_j)$ (having inputs $[(\mathbf{x}_k^E)^\top, (\mathbf{u}_k)^\top]^\top$ and current iteration weights $\boldsymbol{\pi}_j$) as $\{v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q(\mathbf{x}_{k+1}^E, \mathbf{C}(\mathbf{x}_{k+1}^E, \boldsymbol{\theta}_{j-1}), \boldsymbol{\pi}_{j-1})\}$, for all transitions in the dataset. Resulting in the trained Q-function estimator NN $Q(\mathbf{x}_k^E, \mathbf{u}_k, \boldsymbol{\pi}_j)$ with parameter weights $\boldsymbol{\pi}_j$. The second step (the controller improvement) first calculates the targets for the controller $\mathbf{C}(\mathbf{x}_k^E, \boldsymbol{\theta}_j)$ (with inputs $(\mathbf{x}_k^E)^\top$) as $\{\mathbf{u}_k = \text{argmin}_{\mathbf{u} \in \Lambda} Q(\mathbf{x}_k^E, \mathbf{u}, \boldsymbol{\pi}_j)\}$. Note that additional parameterization for the controller NN weights $\boldsymbol{\theta}_j$ is needed. Training produces the improved controller characterized by the new weights $\boldsymbol{\theta}_j$. Here, the discrete set of control actions $\Lambda \subset \Omega_U$ used to minimize the Q-NN estimate for computing the controller targets is the Cartesian product of two identical sets of control actions, each containing 21 equally spaced values in $[-1;1]$, i.e., $\{-1, -0.9, \dots, 0.9, 1\}$.

A discount $\gamma = 0.95$ will be used and each iteration of the NP-IMF-AVI produces a C-NN that is tested on the standard test scenario shown in Figure 6 by measuring the same normalized c.f. J_{test} for $N = 2000$, on the same test scenario that was used in the case of the LP-IMF-AVI. The NP-IMF-AVI is iterated 50 times and all the stabilizing controllers that are better than the VRFT multivariable controller running on the standard test scenario described in Figure 7 (in terms of smaller J_{test}) are stored. The best C-NN across 50 iterations renders $J_{test} = 0.0025$. The tracking performance for the best NN controller found with the NP-IMF-AVI is shown in blue lines in Figure 7. The convergence process is depicted in Figure 9.

A gridsearch is next performed for the NP-IMF-AVI training process, by changing the dataset size from 30,000 to 50,000 to 70,000, combined with 17, 19, and 21 discrete values used for minimizing the Q-function over the two control inputs. For the case of 50,000 data with 17 uniform discrete possible values for each control input, $J_{test} = 0.0017$ which is the same with the best performance of the LP-IMF-AVI. Notice that neither the nonlinear state-feedback controller of the NP-IMF-AVI nor the linear state-feedback controller of the LP-IMF-AVI have integral component, while the linear output feedback controller tuned by VRFT and used for exploration has integrators.

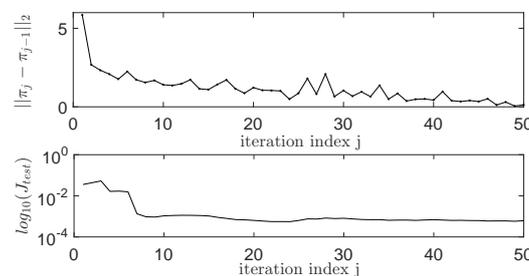


Figure 9. The nonlinearly parameterized IMF-AVI (NP-IMF-AVI) convergence.

Two additional approaches exist for dealing with a NP-IMF-AVI using two NNs, for each of the Q-NN approximator and for the C-NN. For example, [32] used to cascade the C-NN and the Q-NN. After training the Q-NN and producing the new weights $\boldsymbol{\pi}_j$, the weights of the Q-NN are fixed and only the weights $\boldsymbol{\theta}_j$ of the C-NN are trained, with all the targets equal to $\{0\}$ for all the inputs \mathbf{x}_k^E of the cascaded NN $Q(\mathbf{x}_k^E, \mathbf{C}(\mathbf{x}_k^E, \boldsymbol{\theta}_j), \boldsymbol{\pi}_j)$. In this way, the C-NN is forced to minimize the Q-NN. The disadvantage is the vanishing gradient problem of the resulted cascaded network that deepens through more hidden layers, therefore only small corrections are brought to the C-NN part that is further away from the Q-NN's output. Yet another solution [14] uses, for the controller improvement step, a single/several gradient descent step/steps $\boldsymbol{\theta}_j = \boldsymbol{\theta}_{j-1} - \alpha \frac{1}{N} \sum_{k=1}^N \frac{dQ}{d\boldsymbol{\theta}} \Big|_{(\boldsymbol{\theta}_{j-1}, \mathbf{x}_k^E)}$ with each iteration of NP-IMF-AVI, with step size $\alpha > 0$ and with gradient $\frac{1}{N} \sum_{k=1}^N \frac{dQ}{d\boldsymbol{\theta}} \Big|_{(\boldsymbol{\theta}_{j-1}, \mathbf{x}_k^E)}$ accumulated over all inputs \mathbf{x}_k^E of the cascaded NN $Q(\mathbf{x}_k^E, \mathbf{C}(\mathbf{x}_k^E, \boldsymbol{\theta}_{j-1}), \boldsymbol{\pi}_j)$, over fixed Q-NN weights. Essentially, the two approaches described above are equivalent and the number of gradient descent steps at each iteration is user-selectable. Also, no minimization by enumerating a finite set of control actions needs to be performed in either of the two above approaches. The above two equivalent

approaches are effectively a particular case of the Neural-Fitted Q-iteration with Continuous Actions (NFQCA) approach [14], more recently to be updated with some changes to Deep Deterministic Policy Gradient (DDPG) [52]. DDPG uses two NNs as well, for the Q-NN and for the C-NN. It was originally developed to work in online off-policy mode, hence the need to update the Q-NN and the C-NN in a faster way on a relatively small number of transition samples (called minibatch) randomly extracted from a replay buffer equivalent to the dataset D , in order to break the time correlation of consecutive samples. The effectiveness of DDPG in real-time online control has yet to be proven.

Two variants of *offline* off-policy DDPG called DDPG1 and DDPG2 are run for comparisons purposes. Both use minibatches of 128 transitions from the dataset D at each training iteration. While both use soft target updates of the Q-NN weights $\pi'_j = \tau\pi_j + (1 - \tau)\pi'_{j-1}$ and of the C-NN weights $\theta'_j = \tau\theta_j + (1 - \tau)\theta'_{j-1}$, with $\tau = 0.005$. π'_j and θ'_j are used to calculate the targets for the Q-NN training. At each iteration, DDPG1 makes one update step of the Q-NN weights in the negative direction of the gradient of the MSE w.r.t. π_j with step size $\alpha = 0.001$ and one update step of the C-NN weights in the negative direction of the gradient of the Q-NN's output w.r.t. θ_j with step size $\alpha = 0.001$. While DDPG2 differs in that the Q-NN training on each minibatch of each iteration is left to the same settings used for NP-IMF-AVI training (scaled conjugate gradient for maximum 500 epochs), only one gradient descent step is used to update the C-NN weights with the same $\alpha = 0.001$. The step-sizes were selected to ensure learning convergence. It was observed that DDPG1 has the slowest convergence (convergence appears after more than 20,000 iterations) since it performs only one gradient update step per iteration, DDPG2 has faster convergence speed (convergence appears after 5000 iterations) since it allows more gradient steps for Q-NN training, while NP-IMF-AVI has the highest convergence speed (convergence appears after 10 iterations), allowing more training in terms of gradient descent steps (with scaled conjugate gradient direction) for both Q-NN and for C-NN, at each iteration. This proves that, given the high-dimensional process, it is better to use the entire dataset D for offline training, as it was done with NP-IMF-AVI. On the other hand, the best performance with DDPG1 and DDPG2 is 0.003, not as good as the best one with the more computationally demanding NP-IF-AVI (0.0017), suggesting that minimizing the Q-NN by enumerating discrete actions to calculate the C-NN targets may actually escape local minima. The total learning time to convergence with DDPG1 and DDPG2 is about the same as with NP-IMF-AVI, which is to be expected since less calculations for DDPG1 takes more iterations until convergence appears. Notice that NP-IMF-AVI does not use soft target updates for its two NNs.

The additional NN controller is not mandatory and the NP-IMF-AVI can be made similar to the LP-IMF-AVI case. In this case, the minimization of the Q-function NN estimate is to be performed by enumerating the discrete set of control actions $\Lambda \subset \Omega_U$ and the targets calculation for the Q-function NN will use $\{v(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q(x_{k+1}^E, \underset{\mathbf{u} \in \Lambda}{\operatorname{argmin}} Q(\mathbf{x}_{k+1}^E, \mathbf{u}, \pi_j), \pi_{j-1})\}$. This approach merges the controller improvement step and the Q-function improvement step. However, for real-time control implementation after NP-IMF-AVI convergence, it is more expensive to find $\mathbf{u}_k^* = \underset{\mathbf{u} \in \Lambda}{\operatorname{argmin}} Q^*(\mathbf{x}_k^E, \mathbf{u}, \pi_j)$, since it requires evaluating the Q-function NN for a number of times proportional to the number of combinations of discrete control actions. Then only slower processes can be accommodated with this implementation. Whereas in the case when a dedicated controller NN is used, after NP-IMF-AVI convergence, the optimal control $\mathbf{u}_k^* = \mathbf{C}^*(\mathbf{x}_k^E, \theta_j)$ is calculated at once, through a single NN evaluation. This dedicated NN controller can also be obtained (trained) as a final step after the NP-IMF-AVI has converged to the optimal Q-function $Q^*(\mathbf{x}_k^E, \mathbf{u}, \pi_j)$ and the targets for the controller output are calculated as $\{\mathbf{u}_k^* = \underset{\mathbf{u} \in \Lambda}{\operatorname{argmin}} Q^*(\mathbf{x}_k^E, \mathbf{u}, \pi_j)\}$. Another original solution that uses a single NN Q-function approximator was proposed in [53], such that a quadratic approximation of the NN-fitted Q-function is used to directly derive a linear state-feedback controller with each iteration.

4.7. Comments on the Obtained Results

Some comments follow the validation of the LP-IMF-AVI and NP-IMF-AVI. The results of Figure 6 indicate that convergence of the LP-IMF-AVI is attained in terms of $\|\pi_j - \pi_{j-1}\|_2 \rightarrow 0$, however perfect ORM tracking is not possible, as shown by the nonzero constant value of J_{test} . On one hand, this is to be expected since the resulting linear state-feedback controller coupled with the process' nonlinear dynamics is not capable of ensuring a closed-loop linear behavior as requested by the ORM. On the other hand, the NN controller resulting from the NP-IMF-AVI implementation is a nonlinear state feedback controller, however the best obtained results are not better than (but on the same level with) those obtained with the linear state-feedback controller of the NP-IMF-AVI, although the nonlinear controller is expected to perform better in terms of lower J_{test} , due to its flexibility being able to compensate for the process nonlinearity. If this flexibility does not turn into an advantage, the reason lies with the additional NN controller parameterization (that introduces additional approximation errors) and with the training process that relies on approximate minimizations in the calculation step of the controller's targets.

The iterative evolution of J_{test} in case of both LP-IMF-AVI and NP-IMF-AVI show stabilization to constant nonzero values, suggesting that neither approach can provide perfectly ORM tracking controllers. For the LP-IMF-AVI, the responsibility lies with the under-representation error introduced by quadratic Q-function (and with the subsequent resulting linear state-feedback controller), while for NP-IMF-AVI, responsibility lies with the errors introduced by the additional controller approximator NN and the targets calculation in the controller improvement step.

Computational resources analysis indicate that the LP-IMF-AVI has learned only 78 parameters for to the Q-function parameterization, and no intermediate controller approximator is used. The run time for 50 iterations is about 345 seconds (including evaluation steps on the test scenario after each iteration). The NN-based NP-IMF-AVI needs to learn two NNs having 351 parameters (weights) for the Q-function NN and 41 parameters (weights) for the controller NN, respectively. Contrastingly, the runtime for the NP-IMF-AVI is about 3300 seconds, almost ten times more than in the case of the LP-IMF-AVI. Despite the larger parameter learning space, the converged behavior of the NN-based NP-IMF-AVI is very similar to that of the LP-IMF-AVI (see tracking results in Figure 7).

LP-IMF-AVI has shown an increased sensitivity to the transition samples dataset size: for fewer or more transition samples in the dataset, the LP-IMF-AVI diverges, under exactly the same exploration settings. But this divergence appears only after an initial convergence phase similar to that of Figure 6, and not from the very beginning. Whereas having fewer transition samples is intuitively disadvantageous for learning the true Q-function approximation, having a larger number of transition samples leading to divergence is unexpected. The reason is that non-uniform state-action space exploration affects the linear regression. Then, given a fixed dataset size, an increased amplitude of the additive disturbance used to stimulate exploration combined with a more often application of this disturbance (such as every 2nd sample) increases the convergence probability. These observations indicate again that the proposed linear parameterization using quadratic basis functions is insufficient for a correct representation of the true Q-function, thus failing the small approximation errors assumptions of Theorem 2. The connection between the convergence guarantees and the approximation errors have been analyzed in the literature [54–57].

In the light of the previous paragraph's observations, the NN-based NP-IMF-AVI proves to be significantly more robust throughout the convergence process, both to various transition samples dataset sizes and to different exploration settings (disturbance amplitude and frequency of its application, how often the reference inputs switch during the transition samples collection phase, etc.). This may well pay off for the additional controller approximator NN and for the extra computation time since the chances of learning high performance controllers will depend less on the selection of the many parameters involved. Moreover, manual selection of the basis functions is unnecessary with the NN-based NP-IMF-AVI, while the over-parameterization is automatically managed by the NN training mechanism.

Data normalization is a frequently overlooked issue in ADP control but it is critical to successful design since it numerically affects both the regression solution in LP-IMF-AVI and the NN training in NP-IMF-AVI. A diagonal scaling matrix $\mathbf{S} = \text{diag}(s_1, \dots, s_{n+n_m+p})$ leads to the scaled extended state $\bar{\mathbf{x}}_k^E = \mathbf{S}\mathbf{x}_k^E$ resulting in the extended state-space model $\bar{\mathbf{x}}_{k+1}^E = \mathbf{S} \cdot \mathbf{E}(\mathbf{S}^{-1}\bar{\mathbf{x}}_k^E, \mathbf{u}_k)$ that still preserves the MDP property.

5. Conclusions

This paper proves a functional design for an IMF-AVI ADP learning scheme dedicated to the challenging problem of ORM tracking control for a high-order real-world complex nonlinear process with unknown dynamics. The investigation revolves around a comparative analysis of a linear vs. a nonlinear parameterization of the IMF-AVI approach. Learning high performance state-feedback control under the model-free mechanism offered by IMF-AVI builds upon the input–states–outputs transition samples collection step that uses an initial exploratory linear output feedback controller that is also designed in a model-free setup using VRFT. From the practitioners' viewpoint, the NN-based implementation of IMF-AVI is more appealing since it easily scales up with problem dimension and automatically manages the basis functions selection for the function approximators.

Future work attempts to validate the proposed design approach to more complex high-order nonlinear processes of practical importance.

Author Contributions: conceptualization, M.-B.R.; methodology, M.-B.R.; software, T.L.; validation, T.L.; formal analysis, M.-B.R.; investigation, T.L.; data curation, M.-B.R. and T.L.; writing—original draft preparation, M.-B.R. and T.L.; writing—review and editing, M.-B.R. and T.L.; supervision, M.-B.R.;

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Radac, M.B.; Precup, R.E.; Petriu, E.M. Model-free primitive-based iterative learning control approach to trajectory tracking of MIMO systems with experimental validation. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 2925–2938. [[CrossRef](#)]
2. Sutton, R.S.; Barto, A.G. In *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998.
3. Bertsekas, D.P.; Tsitsiklis, J.N. In *Neuro-Dynamic Programming*; Athena Scientific: Belmont, MA, USA, 1996.
4. Wang, F.Y.; Zhang, H.; Liu, D. Adaptive dynamic programming: an introduction. *IEEE Comput. Intell. Mag.* **2009**, *4*, 39–47. [[CrossRef](#)]
5. Lewis, F.; Vrabie, D.; Vamvoudakis, K.G. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst. Mag.* **2012**, *32*, 76–105.
6. Lewis, F.; Vrabie, D.; Vamvoudakis, K.G. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circ. Syst. Mag.* **2009**, *9*, 76–105. [[CrossRef](#)]
7. Murray, J.; Cox, C.J.; Lendaris, G.G.; Saeks, R. Adaptive dynamic programming. *IEEE Trans. Syst. Man Cybern.* **2002**, *32*, 140–153. [[CrossRef](#)]
8. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [[CrossRef](#)]
9. Kiumarsi, B.; Lewis, F.L.; Naghibi-Sistani, M.B.; Karimpour, A. Optimal tracking control of unknown discrete-time linear systems using input–output measured data. *IEEE Trans. Cybern.* **2015**, *45*, 2270–2279. [[CrossRef](#)]
10. Kiumarsi, B.; Lewis, F.L.; Modares, H.; Karimpour, A.; Naghibi-Sistani, M.B. Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics. *Automatica* **2014**, *50*, 1167–1175. [[CrossRef](#)]
11. Radac, M.B.; Precup, R.E.; Roman, R.C. Model-free control performance improvement using virtual reference feedback tuning and reinforcement Q-learning. *Int. J. Syst. Sci.* **2017**, *48*, 1071–1083. [[CrossRef](#)]

12. Lala, T.; Radac, M.B. Parameterized value iteration for output reference model tracking of a high order nonlinear aerodynamic system. In Proceedings of the 2019 27th Mediterranean Conference on Control and Automation (MED), Akko, Israel, 1–4 July 2019; pp. 43–49.
13. Ernst, D.; Geurts, P.; Wehenkel, L. Tree-based batch mode reinforcement learning. *J. Mach. Learn. Res.* **2005**, *6*, 2089–2099.
14. Hafner, R.; Riedmiller, M. Reinforcement learning in feedback control. Challenges and benchmarks from technical process control. *Mach. Learn.* **2011**, *84*, 137–169. [[CrossRef](#)]
15. Zhao, D.; Wang, B.; Liu, D. A supervised actor critic approach for adaptive cruise control. *Soft Comput.* **2013**, *17*, 2089–2099. [[CrossRef](#)]
16. Cui, R.; Yang, R.; Li, Y.; Sharma, S. Adaptive neural network control of AUVs with Control input nonlinearities using reinforcement learning. *IEEE Trans. Syst. Man Cybern.* **2017**, *47*, 1019–1029. [[CrossRef](#)]
17. Xu, X.; Hou, Z.; Lian, C.; He, H. Online learning control using adaptive critic designs with sparse kernel machines. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *24*, 762–775.
18. He, H.; Ni, Z.; Fu, J. A three-network architecture for on-line learning and optimization based on adaptive dynamic programming. *Neurocomputing* **2012**, *78*, 3–13 [[CrossRef](#)]
19. Modares, H.; Lewis, F.L.; Jiang, Z.P. H_∞ Tracking control of completely unknown continuous-time systems via off-policy reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 2550–2562. [[CrossRef](#)]
20. Li, J.; Modares, H.; Chai, T.; Lewis, F.L.; Xie, L. Off-policy reinforcement learning for synchronization in multiagent graphical games. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 2434–2445. [[CrossRef](#)]
21. Bertsekas, D. Value and policy iterations in optimal control and adaptive dynamic programming. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 500–509. [[CrossRef](#)]
22. Yang, Y.; Wunsch, D.; Yin, Y. Hamiltonian-driven adaptive dynamic programming for continuous nonlinear dynamical systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 1929–1940. [[CrossRef](#)]
23. Kamalapurkar, R.; Andrews, L.; Walters, P.; Dixon, W.E. Model-based reinforcement learning for infinite horizon approximate optimal tracking. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 753–758. [[CrossRef](#)]
24. Radac, M.B.; Precup, R.E. Data-driven MIMO Model-free reference tracking control with nonlinear state-feedback and fractional order controllers. *Appl. Soft Comput.* **2018**, *73*, 992–1003. [[CrossRef](#)]
25. Campi, M.C.; Lecchini, A.; Savaresi, S.M. Virtual Reference Feedback Tuning: A direct method for the design of feedback controllers. *Automatica* **2002**, *38*, 1337–1346. [[CrossRef](#)]
26. Hjalmarsson, H. Iterative Feedback Tuning—An overview. *Int. J. Adapt. Control Signal Process.* **2002**, *16*, 373–395. [[CrossRef](#)]
27. Janssens, P.; Pipeleers, G.; Swevers, J.L. Model-free iterative learning control for LTI systems and experimental validation on a linear motor test setup. In Proceedings of the 2011 American Control Conference (ACC), San Francisco, CA, USA, 29 June–1 July 2011; pp. 4287–4292.
28. Radac, M.B.; Precup, R.E. Optimal behavior prediction using a primitive-based data-driven model-free iterative learning control approach. *Comput. Ind.* **2015**, *74*, 95–109. [[CrossRef](#)]
29. Chi, R.; Hou, Z.S.; Jin, S.; Huang, B. An improved data-driven point-to-point ILC using additional on-line control inputs with experimental verification. *IEEE Trans. Syst. Man Cybern.* **2017**, *49*, 687–696. [[CrossRef](#)]
30. Abouaissa, H.; Fliess, M.; Join, C. On ramp metering: towards a better understanding of ALINEA via model-free control. *Int. J. Control* **2017**, *90*, 1018–1026. [[CrossRef](#)]
31. Hou, Z.S.; Liu, S.; Tian, T. Lazy-learning-based data-driven model-free adaptive predictive control for a class of discrete-time nonlinear systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 1914–1928. [[CrossRef](#)]
32. Radac, M.B.; Precup, R.E.; Roman, R.C. Data-driven model reference control of MIMO vertical tank systems with model-free VRFT and Q-learning. *ISA Trans.* **2018**, *73*, 227–238. [[CrossRef](#)]
33. Bolder, J.; Kleinendorst, S.; Oomen, T. Data-driven multivariable ILC: enhanced performance by eliminating L and Q filters. *Int. J. Robot. Nonlinear Control* **2018**, *28*, 3728–3751. [[CrossRef](#)]
34. Wang, Z.; Lu, R.; Gao, F.; Liu, D. An indirect data-driven method for trajectory tracking control of a class of nonlinear discrete-time systems. *IEEE Trans. Ind. Electron.* **2017**, *64*, 4121–4129. [[CrossRef](#)]
35. Pandian B.J.; Noel, M.M. Control of a bioreactor using a new partially supervised reinforcement learning algorithm. *J. Proc. Control* **2018**, *69*, 16–29. [[CrossRef](#)]
36. Diaz, H.; Armesto, L.; Sala, A. Fitted q-function control methodology based on takagi-sugeno systems. *IEEE Trans. Control Syst. Tech.* **2018**. [[CrossRef](#)]

37. Wang, W.; Chen, X.; Fu, H.; Wu, M. Data-driven adaptive dynamic programming for partially observable nonzero-sum games via Q-learning method. *Int. J. Syst. Sci.* **2019**. [CrossRef]
38. Mu, C.; Zhang, Y. Learning-based robust tracking control of quadrotor with time-varying and coupling uncertainties. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**. [CrossRef] [PubMed]
39. Liu, D.; Yang, G.H. Model-free adaptive control design for nonlinear discrete-time processes with reinforcement learning techniques. *Int. J. Syst. Sci.* **2018**, *49*, 2298–2308. [CrossRef]
40. Song, F.; Liu, Y.; Xu, J.X.; Yang, X.; Zhu, Q. Data-driven iterative feedforward tuning for a wafer stage: A high-order approach based on instrumental variables. *IEEE Trans. Ind. Electr.* **2019**, *66*, 3106–3116. [CrossRef]
41. Kofinas, P.; Dounis, A. Fuzzy Q-learning agent for online tuning of PID controller for DC motor speed control. *Algorithms* **2018**, *11*, 148. [CrossRef]
42. Radac, M.-B.; Precup, R.-E. Three-level hierarchical model-free learning approach to trajectory tracking control. *Eng. Appl. Artif. Intell.* **2016**, *55*, 103–118. [CrossRef]
43. Radac, M.-B.; Precup, R.-E. Data-based two-degree-of-freedom iterative control approach to constrained non-linear systems. *IET Control Theory Appl.* **2015**, *9*, 1000–1010. [CrossRef]
44. Salgado, M.; Clempner, J.B. Measuring the emotional state among interacting agents: A game theory approach using reinforcement learning. *Expert Syst. Appl.* **2018**, *97*, 266–275. [CrossRef]
45. Silva, M.A.L.; de Souza, S.R.; Souza, M.J.F.; Bazzan, A.L.C. A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems. *Expert Syst. Appl.* **2019**, *131*, 148–171. [CrossRef]
46. Radac, M.-B.; Precup, R.-E.; Hedrea E.-L.; Mituletu I.-C. Data-driven model-free model-reference nonlinear virtual state feedback control from input-output data. In Proceedings of the 2018 26th Mediterranean Conference on Control and Automation (MED), Zadar, Croatia, 19–22 June 2018; pp. 332–338.
47. Campestrini, L.; Eckhard, D.; Gevers, M.; Bazanella, M. Virtual reference tuning for non-minimum phase plants. *Automatica* **2011**, *47*, 1778–1784. [CrossRef]
48. Al-Tamimi, A.; Lewis, F.L.; Abu-Khalaf, M. Discrete-time nonlinear HJB Solution using approximate dynamic programming: Convergence proof. *IEEE Trans. Syst. Man Cybern. Cybern.* **2008**, *38*, 943–949. [CrossRef] [PubMed]
49. Rantzer, A. Relaxed dynamic programming in switching systems. *IEEE Proc. Control Theory Appl.* **2006**, *153*, 567–574. [CrossRef]
50. Inteco, LTD. *Two Rotor Aerodynamical System; User's Manual*; Inteco, LTD: Krakow, Poland, 2007. Available online: http://ee.sharif.edu/~lcs/lab/Tras_um_PCI.pdf (accessed on 12 June 2019).
51. Radac, M.-B.; Precup, R.-E. Data-driven model-free slip control of anti-lock braking systems using reinforcement Q-learning. *Neurocomputing* **2018**, *275*, 317–329. [CrossRef]
52. Lillicrap, T.P.; Hunt, J.J.; Pritzel A.; Heess N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *Mach. Learn.* **2016**, arXiv:1509.02971.
53. Ten Hagen, S.; Krose, B. Neural Q-learning. *Neural Comput. Appl.* **2003**, *12*, 81–88. [CrossRef]
54. Radac, M.B.; Precup, R.E. Data-Driven model-free tracking reinforcement learning control with VRFT-based adaptive actor-critic. *Appl. Sci.* **2019**, *9*, 1807. [CrossRef]
55. Dierks, T.; Jagannathan, S. Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update. *IEEE Trans. Neural Netw. Learn. Syst.* **2012**, *23*, 1118–1129. [CrossRef]
56. Heydari, A. Theoretical and numerical analysis of approximate dynamic programming with approximation errors. *J. Gui Control Dyn.* **2016**, *39*, 301–311. [CrossRef]
57. Heydari, A. Revisiting approximate dynamic programming and its convergence. *IEEE Trans. Cybern.* **2014**, *44*, 2733–2743. [CrossRef] [PubMed]

