applied sciences

MDPI

Article

Data-Driven Model-Free Tracking Reinforcement Learning Control with VRFT-based Adaptive Actor-Critic

Mircea-Bogdan Radac * and Radu-Emil Precup

Department of Automation and Applied Informatics, Politehnica University of Timisoara, Timisoara 300006, Romania; radu.precup@upt.ro

* Correspondence: mircea.radac@upt.ro; Tel.: +40-256-403-240

Received: 28 January 2019; Accepted: 24 April 2019; Published: 30 April 2019

Abstract: This paper proposes a neural network (NN)-based control scheme in an Adaptive Actor-Critic (AAC) learning framework designed for output reference model tracking, as a representative deep-learning application. The control learning scheme is model-free with respect to the process model. AAC designs usually require an initial controller to start the learning process; however, systematic guidelines for choosing the initial controller are not offered in the literature, especially in a model-free manner. Virtual Reference Feedback Tuning (VRFT) is proposed for obtaining an initially stabilizing NN nonlinear state-feedback controller, designed from input-state-output data collected from the process in open-loop setting. The solution offers systematic design guidelines for initial controller design. The resulting suboptimal state-feedback controller is next improved under the AAC learning framework by online adaptation of a critic NN and a controller NN. The mixed VRFT-AAC approach is validated on a multi-input multi-output nonlinear constrained coupled vertical two-tank system. Discussions on the control system behavior are offered together with comparisons with similar approaches.

Keywords: adaptive actor-critic; model-free control; data-driven control; reinforcement learning; approximate dynamic programming; output reference model tracking; multi-input multi-output systems; vertical tank systems; Virtual Reference Feedback Tuning

1. Introduction

Data-driven or data-based control techniques rely on data collected from the process in order to learn and tune controllers that prevent control performance degradation due to mismatch between the true process and its model—the main issue with model-based control design approaches [1]. The data-driven controller learning objective can be achieved either by using highly adaptive simplified phenomenological models [2,3], or by using no model at all, except for common structural assumptions about the true process such as linearity or nonlinearity. The latter approach can be considered a true model-free one, with several representative techniques having first emerged from classical control theory, such as: Virtual Reference Feedback Tuning (VRFT), [4], Iterative Feedback Tuning [5], Simultaneous Perturbation Stochastic Approximation [6], Model-free Iterative Learning Control [7,8]. Most of the above approaches relying on instruments specific to optimal control with several recent applications [9–17].

Reinforcement learning (RL) [18] is a powerful data-driven technique that solves optimal control problems with parallel developments in the machine learning and control systems communities in which RL is better known as Adaptive (Approximate) Dynamic Programming (ADP) [19] or neuro-dynamic programming [20]. Reinforcement Q-learning [21] with function approximators (FAs)

is a particular version of Action Dependent Heuristic Dynamic Programming implemented without a process model [22,23], which is only one of the several types of adaptive actor-critic (AAC) ADP designs [24–27], besides Heuristic Dynamic Programming [28], Dual Heuristic Programming [29] and all of their action-dependent versions.

For learning high performance control, Action Dependent Heuristic Dynamic Programming (a form of continuous input-state space Q-learning) uses the Q-function as an extension of the cost (value) function and only needs to efficiently explore the input-state space of the unknown process, hence the model-free data-driven label is justified. The class of model-free AAC designs used with FAs is attractive over the majority of the model-based AAC designs, where a partially known nonlinear input-affine state-space representation is at least necessary [22,23]. The main disadvantages of the Action Dependent Heuristic Dynamic Programming schemes are that many transition samples are needed from the process—since the Q-function estimation is more informative, it needs to explore the action space in addition to the state space—and the lack of convergence guarantees in the absence of a process model, when generic FAs are used. Data-driven RL/ADP formulated in terms of control systems theory has also offered recent results regarding different applications and stability and learning convergence, in both model-free and model-based settings [30,31].

In output reference model (ORM) tracking control, the output of the controlled process should track a reference model's output regarded as a frequently changing time-varying learning goal. This control objective can also be formulated in an optimal control setup. An initial stabilizing state feedback controller that achieves suboptimal ORM tracking control is highly desirable in practice since it could accelerate the learning process. In fact, most of the AAC learning control architectures start the controller learning with respect to some objective using an initial controller, but lack systematic guidelines for obtaining such initial controller.

VRFT is one solution to design data-driven model-free feedback controllers, commonly using input-output data. Its linear time-invariant framework typically needs much fewer samples than model-free AAC designs to obtain an initial controller. Unfortunately, a linear controller cannot ensure good ORM tracking for nonlinear processes acting in wide operating ranges. Since AAC should essentially learn a nonlinear state-feedback controller, it is of interest to obtain such an initial (possibly suboptimal) controller, and this will be shown possible using the VRFT design and tuning framework. This would be significant since model-free AAC approaches are data-hungry in practice and any initial suboptimal solution would shorten the convergence time. Under such motivation, the combination of VRFT and AAC is used to achieve ORM tracking control. The resulting AAC design consists of two neural networks (NNs), one for the controller called the actor NN and one for the cost function approximation called the critic NN. The correction signals during the adaptive learning are backpropagated through the larger NN resulted from cascading the actor and the critic NNs, hence the AAC architecture belongs to the deep reinforcement learning approaches from the literature [32].

The mixed VRFT-AAC approach developed in this paper is applied to a real-world Multi-Input Multi-Output (MIMO) nonlinear coupled constrained laboratory vertical two-tank system for water level control. The approach proposed as follows is novel with respect to the state-of-the-art since:

- it introduces an original nonlinear state-feedback neural network-based controller for ORM tracking, tuned with VRFT, serving as initialization for the AAC learning controller that further improves the ORM tracking and accelerates convergence to the optimal controller. This leads to the novel VRFT-AAC combination;
- the case study proves implementation of the novel mixed control learning approach for ORM tracking. The MIMO validation scenario also demonstrates good decoupling ability of learned controllers, even under constraints and nonlinearities. Comparisons with a model-free batch fitted Q-learning scheme and with a model-based batch-fitted Q-learning approach are also offered. Statistical characterization case studies in different learning settings are given.
- theoretical analysis ensures that the AAC learning scheme preserves the CS stability throughout the updates and converges to the optimal control.

The paper is organized as follows: the next section formulates the ORM tracking control problem in an optimal control framework and offers a way to solve it using VRFT (Section 3) and AAC design (Section 4). Validation case study, useful implementation details, comparisons with similar control learning techniques, thorough investigations and discussions of the observed results, are all presented in Section 5. The concluding remarks are highlighted in Section 6.

2. Output Model Reference Control for Unknown Systems

Let the discrete-time nonlinear unknown open-loop minimum-phase state-space deterministic strictly causal process be

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

where k indexes the discrete time, $\mathbf{x}_k = [x_{k,1} \ \dots \ x_{k,n}]^T \in X \subset \mathfrak{R}^n$ is the n-dimensional state vector (upper T is matrix transpose), $\mathbf{u}_k = [u_{k,1}, \dots, u_{k,m}]^T \in U \subset \mathfrak{R}^m$ is the control input signal, $\mathbf{y}_k = [y_{k,1}, \dots, y_{k,p}]^T \in Y \subset \mathfrak{R}^p$ is the measurable controlled output, $\mathbf{f} : X \times U \to X$ is an unknown nonlinear system function, $\mathbf{g} : X \to Y$ is an unknown nonlinear output function of the states, and the initial conditions are not considered for analysis at this point. It is further assumed that the definition domains X, U, Y are compact convex. The following assumptions common to the data-driven problem formulation [1] are:

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

A2: System (1) is internally stable on $X \times U$.

Assumptions *A1* and *A2* are common in the data-driven control literature and difficult to assess when unknown process models are assumed. They may be supported from the experience on the process operation or from the literature. If no knowledge exists whatsoever, control can be tried in the constraining domains related to the minimum safety operating conditions of the process, which is required minimum information on the process variables. Internal stability is sufficient for output feedback control design and necessary for state-feedback control design using input-state samples.

Concerning the controllability and full state observability assumption *A1* imposed to the process, if the observability cannot be verified analytically, data-driven observers can be built using past samples of either the inputs and outputs and/or of the partially measurable state, as shown for linear systems in [33,34] and used for nonlinear systems in [35]. State measurement requires more insight on the process than several pure input-output representations.

Equation (1) is a general form for most controlled processes in practice and it is not restrictive. In this form, it obeys the definition of a deterministic Markov decision process.

The discrete-time known open-loop stable minimum-phase state-space deterministic strictly causal ORM is

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

where $\mathbf{x}_k^m = [x_{k,1}^m, \dots, x_{k,n_m}^m]^T \in X_m \subset \mathfrak{R}^{n_m}$ is the state vector of the ORM, $\mathbf{r}_k = [r_{k,1}, \dots, r_{k,p}]^T \in R_m \subset \mathfrak{R}^p$ is the reference input signal, $\mathbf{y}_k^m = [y_{k,1}^m, \dots, y_{k,p}^m]^T \in Y_m \subset \mathfrak{R}^p$ is the ORM's output, $\mathbf{f}^m : X_m \times R_m \to X_m$, $\mathbf{g}^m : X_m \to Y_m$ are known nonlinear maps. Initial conditions are zero unless stated otherwise. Note that $\mathbf{r}_k, \mathbf{y}_k, \mathbf{y}_k^m$ have the same size p for square feedback control systems. If the ORM (2) is linear time-invariant in particular, it is always possible to express the ORM as an input-output linear time-invariant transfer matrix $\mathbf{y}_k^m = \mathbf{M}(z)\mathbf{r}_k$, where $\mathbf{M}(z)$ is an asymptotically stable unit gain (i.e., $\mathbf{M}(1) = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix) rational transfer matrix and $\mathbf{r}_k$ is the reference input that drives both the feedback control system and the ORM. To extend the process (1) with the ORM (2), we consider the reference input $\mathbf{r}_k$ as a set of measurable exogenous signals that evolve according to $\mathbf{r}_{k+1} = \mathbf{h}^m(\mathbf{r}_k)$, with unknown $\mathbf{h}^m : \mathfrak{R}^m \to \mathfrak{R}^m$ but measurable $\mathbf{r}_k$. Piecewise constant $\mathbf{r}_k$ can be

modeled for example as $\mathbf{r}_{k+1} = \mathbf{r}_k$ and it will be used throughout this paper. Then the extended state-space model with output equations is

$$\begin{aligned} \mathbf{x}_{k+1}^E &= \begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{x}_{k+1}^m \\ \mathbf{r}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{f}^m(\mathbf{x}_k^m, \mathbf{r}_k) \\ \mathbf{h}^m(\mathbf{r}_k) \end{bmatrix} = \mathbf{F}(\mathbf{x}_k^E, \mathbf{u}_k), \mathbf{x}_k^E \in X^E = X \times X_m \times R_m, \\ \mathbf{y}_k &= \mathbf{g}(\mathbf{x}_k^E), \\ \mathbf{y}_k^m &= \mathbf{g}^m(\mathbf{x}_k^E). \end{aligned} \tag{3}$$

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 $\mathbf{x}_i$ be [36]

$$J(\mathbf{x}_i^E, U_{i,\infty}) = \sum_{k=i}^{\infty} \gamma^{k-i} V(\mathbf{x}_k^E, \mathbf{u}_k), U_{i,\infty} = \{\mathbf{u}_i, \ldots, \mathbf{u}_\infty\}, \tag{4}$$

where i indexes the starting time for $\mathbf{x}_i^E$, the discount factor $0 < \gamma \leq 1$ ensures the convergence of $J(\mathbf{x}_i^E, U_{i,\infty})$ [23] and sets the controller's (or interacting agent's) horizon, the stage cost $V > 0$ depends on $\mathbf{x}_k^E$ and $\mathbf{u}_k$ and captures the distance relative to some pre-specified learning goal (target) usually constant in many applications. The unknown control inputs $\mathbf{u}_i$, $\mathbf{u}_{i+1}$, ..., should minimize $J(\mathbf{x}_i^E, U_{i,\infty})$. A control sequence (or a controller) rendering a finite c.f. are called *admissible*.

ORM tracking control requires that the undisturbed process output $\mathbf{y}_k$ (also the control system output) tracks the ORM's output $\mathbf{y}_k^m = \mathbf{M}(z)\mathbf{r}_k$. For stage cost $V_{MR} = \|\mathbf{y}_k^m(\mathbf{x}_k^E) - \mathbf{y}_k(\mathbf{x}_k^E)\|_2^2$ in Equation (4) (measurable $\mathbf{y}_k$ depends via unknown $\mathbf{g}(\ )$ on $\mathbf{x}_k$, but not on $\mathbf{x}_{k+1}$), we introduce the discounted infinite-horizon model reference tracking c.f.

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

where $\boldsymbol{\varepsilon}_k(\mathbf{x}_k^E, \theta)$ is the model reference tracking error vector, $\theta \in \Re^{n_\theta}$ is a parameterization of a nonlinear feedback admissible controller [23] defined as $\mathbf{u}_k \stackrel{def}{=} \mathbf{C}(\mathbf{x}_k^E, \theta)$, which used in Equation (5) reflects the influence of θ on all system trajectories outcomes. This controller coupled with Equation (3) ensures that the output of Equation (1) tracks the ORM's output. J_{MR}^{∞} in (5) also serves as the value function of using the controller $\mathbf{C}$. For finite J_{MR}^{∞} when $\gamma = 1$, $\boldsymbol{\varepsilon}_k$ must be a square-summable sequence which can be obtained with an asymptotically stabilizing controller that ensures $\lim_{k\to\infty} \|\mathbf{y}_k^m(\mathbf{x}_k^E) - \mathbf{y}_k(\mathbf{x}_k^E)\|_2^2 = 0$. In the general case when $\gamma < 1$, J_{MR}^{∞} will be finite with any stabilizing controller that renders a finite upper bounded $\boldsymbol{\varepsilon}_k$. Herein, admissible controller for Equations (4) and (5) means the controller that ensures a finite c.f. J_{MR}^{∞}.

A nonlinear reference model $\mathbf{M}$ could have been used for tracking purposes as well; however, imposing a linear time-invariant one for the feedback control system ensures indirect feedback linearization of the controlled process. It is extremely beneficial to work with linearized feedback control systems because their behavior generalizes well in wide operating ranges [37]. The ORM tracking problem concerns the control system behavior from the reference input to the controlled output, neglecting potential load disturbances [38]. Extension of the proposed theory to nonlinear ORMs is not difficult. Under classical control rules, the process's delay and non-minimum-phase character should be included in $\mathbf{M}$. However, the non-minimum-phase zeroes make $\mathbf{M}$ non-invertible in addition to requiring their knowledge via identification [38], affecting the subsequent VRFT design, motivating the minimum phase assumption on the process.

3. Nonlinear State-Feedback VRFT for Approximate ORM Tracking Control Using Neural Networks

An initial controller for the system (3) to achieve approximate ORM tracking employs the VRFT concept. Under assumptions *A1* and *A2*, for tuning a nonlinear state-feedback controller, the designer may employ an input-state-output dataset of the form $\left\{\tilde{\mathbf{u}}_k, \tilde{\mathbf{x}}_k, \tilde{\mathbf{y}}_k\right\}, k = \overline{0, N-1}$, gathered from the process in an open-loop experiment lasting for N sample time steps, where persistently exciting $\tilde{\mathbf{u}}_k$ excites all the significant process dynamics. To achieve linear ORM tracking for a nonlinear process, a nonlinear state-feedback controller is more suitable than a linear one, being able to cope with the process nonlinearities.

VRFT concept assumes that, if the controlled output $\mathbf{y}_k$ produced in an open-loop experiment conducted on the stable process is both the control system's output and the ORM's output, then the closed-loop control system will match the reference model [4,39–42]. Let $\tilde{\mathbf{r}}_k = \mathbf{M}(z)^{-1}\tilde{\mathbf{y}}_k$ be the virtual reference input that generates $\tilde{\mathbf{y}}_k$ when filtered through $\mathbf{M}(z)$ which is assumed to be invertible with respect to the inverse filtering operation. It is called virtual since it is never set as a reference input to the closed-loop control system and it is only used in the offline controller tuning. The virtual states of the ORM are computable from Equation (2) as $\tilde{\mathbf{x}}^m_{k+1} = \mathbf{f}^m(\tilde{\mathbf{x}}^m_k, \tilde{\mathbf{r}}_k)$ serving to reconstruct the virtual extended state as $\tilde{\mathbf{x}}^E_k = [\left(\tilde{\mathbf{x}}_k\right)^T \left(\tilde{\mathbf{x}}^m_k\right)^T \left(\tilde{\mathbf{r}}_k\right)^T]^T$ A controller that produces $\tilde{\mathbf{u}}_k$ when fed by $\tilde{\mathbf{x}}^E_k$ achieves the ORM tracking. VRFT translates the model reference tracking c.f. in Equation (5) to a controller identification c.f. A finite-time controller identification c.f. is [4]

$$J^N_{VR}(\theta) = \sum_{k=0}^{N-1} \left\| \tilde{\mathbf{u}}_k - C(\tilde{\mathbf{x}}^E_k, \theta) \right\|^2. \tag{6}$$

Let the optimal controller parameter vector θ^* be the solution to the optimization problem $\theta^* = \arg\min_{\theta} J^N_{VR}(\theta)$. *Theorem 2* in [41] shows that if the controller parameterization is rich enough, then θ^* also minimizes J^∞_{MR}, proven for input-output models only. Motivated by [41], a formal proof is given as incentive for using state-feedback controllers tuned by nonlinear multi-input multi-output (MIMO) VRFT. Several other assumptions are considered:

A3: The process (1) has an equivalent input-output form $\mathbf{y}_k = \mathbf{P}(\mathbf{y}_{k-1}, \ldots, \mathbf{y}_{k-ny}, \mathbf{u}_{k-1}, \ldots, \mathbf{u}_{k-nu})$, where *ny*, *nu* are unknown process orders and the nonlinear map $\mathbf{P}$ is invertible with respect to $\mathbf{u}$, meaning that for given $\mathbf{y}_k$, $\mathbf{u}_k$ is recoverable as $\mathbf{u}_{k-1} = \mathbf{P}^{-1}(\mathbf{y}_k)$. Zero initial conditions are assumed at this point. Also, the ORM (2) has an equivalent input-output form $\mathbf{y}^m_k = \mathbf{M}(\mathbf{y}^m_{k-1}, \ldots, \mathbf{y}^m_{k-nym}, \mathbf{r}_{k-1}, \ldots, \mathbf{r}_{k-nr})$ where *nym*, *nr* are known ORM's orders, $\mathbf{M}$ is a nonlinear invertible map with stable inverse, allowing the calculation of $\mathbf{r}_{k-1} = \mathbf{M}^{-1}(\mathbf{y}^m_k)$. Zero initial conditions are also assumed.

A4: Let the process (1) and the ORM (2) be formally written as $\mathbf{y}_k = \mathbf{s}(\mathbf{x}_k, \mathbf{u}_{k-1})$ and $\mathbf{y}^m_k = \mathbf{s}^m(\mathbf{x}^m_k, \mathbf{r}_{k-1})$, respectively, to capture simultaneously both the input-output dependence and the input-state-output one in a compact form. These expressions also reveal the relative degree one from input to output, without loss of generality. Assume zero initial conditions for (1) and assume the map $\mathbf{s}$ invertible with $\mathbf{x}_k, \mathbf{u}_{k-1}$ computable from $\mathbf{y}_k$ as $\mathbf{x}_k = (\mathbf{s}_x)^{-1}(\mathbf{y}_k)$, $\mathbf{u}_{k-1} = (\mathbf{s}_u)^{-1}(\mathbf{y}_k)$. Further assume that $\mathbf{s}^m$ is a continuously differentiable invertible map such that $\mathbf{x}^m_k, \mathbf{r}_{k-1}$ are computable from $\mathbf{y}^m_k$ as $\mathbf{x}^m_k = (\mathbf{s}^m_x)^{-1}(\mathbf{y}^m_k), \mathbf{r}_{k-1} = (\mathbf{s}^m_r)^{-1}(\mathbf{y}^m_k)$ and assume there exists positive constants $B^m_{sx} > 0, B^m_{sr} > 0$ such that $\|\frac{\partial \mathbf{s}^m(\mathbf{x}^m_k, \mathbf{r}_{k-1})}{\partial \mathbf{x}^m_k}\| < B^m_{sx}, \|\frac{\partial \mathbf{s}^m(\mathbf{x}^m_k, \mathbf{r}_{k-1})}{\partial \mathbf{r}_{k-1}}\| < B^m_{sr}$. Let zero initial conditions hold for (2). These inversion assumptions are natural for state-space systems such as (1) and (2) that have equivalent input-output models according to *A4*. For example, for given output $\mathbf{y}_k$ of (1), the input is uniquely determined as $\mathbf{u}_{k-1} = \mathbf{P}^{-1}(\mathbf{y}_k)$, after which the state can be generated by recursion from $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ of Equation (1). This is the sense of $\mathbf{x}_k = (\mathbf{s}_x)^{-1}(\mathbf{y}_k)$.

Moreover, let $\mathbf{s}$, $(\mathbf{s}_x)^{-1}$ be continuously differentiable and of bounded derivative to satisfy

$$\begin{array}{l} \|\frac{\partial \mathbf{s}(\mathbf{x}_k,\mathbf{u}_{k-1})}{\partial \mathbf{x}_k}\| < B_{sx}, \|\frac{\partial \mathbf{s}(\mathbf{x}_k,\mathbf{u}_{k-1})}{\partial \mathbf{u}_{k-1}}\| < B_{su}, \|\frac{\partial (\mathbf{s}_x)^{-1}(\mathbf{y}_k)}{\partial \mathbf{y}_k}\| < B_{sy}, \\ 0 < B_{sx}B_{sy} < 1. \end{array} \tag{7}$$

A5: Let a finite open-loop trajectory collected from the process be $D = \left\{\tilde{\mathbf{u}}_k, \tilde{\mathbf{x}}_k, \tilde{\mathbf{y}}_k\right\} \subset U \times X \times Y, k = \overline{0, N-1}$ where $\tilde{\mathbf{u}}_k$ is: (1) persistently exciting, for $\tilde{\mathbf{y}}_k$ to capture all process dynamics and (2) ensuring uniform exploration of the entire domain $U \times X \times Y$. Good exploration is achievable for large enough N.

A6: There exists a set of nonlinear parameterized state-feedback continuously differentiable controllers $\left\{\mathbf{C}(\mathbf{x}_k^E, \theta)\right\}$, a $\hat{\theta}$ for which $\hat{\mathbf{u}}_k = \mathbf{C}(\hat{\mathbf{x}}_k^E, \hat{\theta})$, and an $\varepsilon > 0$ for which

$$J_{VR}^N(\hat{\theta}) = \sum_{k=0}^{N-1} \left\| \widetilde{\mathbf{u}}_k - \mathbf{C}(\widetilde{\mathbf{x}}_k^E, \hat{\theta}) \right\|^2 < \varepsilon^2, \tag{8}$$

$$\|\frac{\partial \mathbf{C}(\mathbf{x}_k^E, \theta)}{\partial \mathbf{x}_k^E}\| < B_{cx}, \tag{9}$$

where $\tilde{\mathbf{x}}_k^E = [\left(\tilde{\mathbf{x}}_k\right)^T \left(\tilde{\mathbf{x}}_k^m\right)^T \left(\tilde{\mathbf{r}}_k\right)^T]^T$, $\hat{\mathbf{x}}_k^E = [\left(\hat{\mathbf{x}}_k\right)^T \left(\tilde{\mathbf{x}}_k^m\right)^T \left(\tilde{\mathbf{r}}_k\right)^T]^T$ and $\tilde{\mathbf{x}}_k^m = (\mathbf{s}_x^m)^{-1}(\tilde{\mathbf{y}}_k), \tilde{\mathbf{r}}_{k-1} = (\mathbf{s}_r^m)^{-1}(\tilde{\mathbf{y}}_k)$. Technically, $\left\{\hat{\mathbf{u}}_k, \hat{\mathbf{x}}_k, \hat{\mathbf{y}}_k\right\}$ are generated with $\hat{\mathbf{u}}_k = \mathbf{C}(\hat{\mathbf{x}}_k^E, \hat{\theta})$ in closed-loop, by processing the virtual signals $\tilde{\mathbf{x}}_k^m, \tilde{\mathbf{r}}_{k-1}$ obtained from $\tilde{\mathbf{y}}_k$.

Theorem 1: Under assumptions *A3–A6*, there exists a finite $B > 0$ such that

$$J_{MR}^N(\hat{\theta}) = \sum_{k=1}^{N} \left\| \hat{\mathbf{y}}_k - \widetilde{\mathbf{y}}_k \right\|^2 = \sum_{k=1}^{N} \left\| \mathbf{s}(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_{k-1}) - \mathbf{s}^m(\widetilde{\mathbf{x}}_k^m, \widetilde{\mathbf{r}}_{k-1}) \right\|_2^2 < B\varepsilon^2. \tag{10}$$

Proof: See Appendix A.

Corollary 1. The controller $\mathbf{C}(\mathbf{x}_k^E, \hat{\theta})$ obtained by minimizing the c.f. (6) is stabilizing and admissible for J_{MR}^{∞} in Equation (5) with $\gamma < 1$.

Proof. By Equation (8), properly identified $\mathbf{C}(\mathbf{x}_k^E, \hat{\theta})$ renders the finite-time $J_{MR}^N(\hat{\theta})$ (10) arbitrarily small. Secondly, a good exploration of $U \times X \times Y$ ensured by $D = \left\{\tilde{\mathbf{u}}_k, \tilde{\mathbf{x}}_k, \tilde{\mathbf{y}}_k\right\}$ reflects in good exploration of domains R_m, X_m by $\tilde{\mathbf{r}}_k, \tilde{\mathbf{x}}_k^m$ respectively. In (10), $\hat{\mathbf{u}}_k, \hat{\mathbf{x}}_k, \hat{\mathbf{y}}_k, \tilde{\mathbf{x}}_k^m, \hat{\mathbf{x}}_k^E, \tilde{\mathbf{x}}_k^E$ are all generated from the same $\tilde{\mathbf{r}}_k$. If (10) holds for many combinations $\tilde{\mathbf{r}}_k, \tilde{\mathbf{x}}_k^m$ rendered form exploratory data, then by the arguments of continuous differentiability and bounded derivatives of the maps (7) and by assumption *A4*, they will hold for any possible combination of $\mathbf{r}_k$ and $\mathbf{x}_k^m = \mathbf{f}^m(\mathbf{x}_{k-1}^m, \mathbf{r}_{k-1})$ generated from any $\mathbf{r}_k$.

To show this, note that both $\tilde{\mathbf{y}}_k = \mathbf{y}_k^m = \mathbf{s}^m(\tilde{\mathbf{x}}_k^m, \tilde{\mathbf{r}}_{k-1})$ and $\hat{\mathbf{y}}_k = \mathbf{y}_k = \mathbf{s}(\hat{\mathbf{x}}_k, \mathbf{u}_{k-1}) = \mathbf{s}(\hat{\mathbf{x}}_k, \mathbf{C}(\hat{\mathbf{x}}_{k-1}^E, \hat{\theta}))$ in (10) can be generated from the same $\tilde{\mathbf{r}}_k$ ($\hat{\mathbf{x}}_{k-1}^E$ contains $\tilde{\mathbf{r}}_{k-1}$). Using this fact, it follows from (10) that the ORM tracking errors $\|\mathbf{s}(\hat{\mathbf{x}}_k, \mathbf{C}(\hat{\mathbf{x}}_{k-1}^E(\tilde{\mathbf{r}}_{k-1}^{(1)}))) - \mathbf{s}^m(\tilde{\mathbf{x}}_k^m, \tilde{\mathbf{r}}_{k-1}^{(1)})\|$ and $\|\mathbf{s}(\hat{\mathbf{x}}_k, \mathbf{C}(\hat{\mathbf{x}}_{k-1}^E(\tilde{\mathbf{r}}_{k-1}^{(2)}))) - \mathbf{s}^m(\tilde{\mathbf{x}}_k^m, \tilde{\mathbf{r}}_{k-1}^{(2)})\|$ are bounded at each time step, for any two training pairs $\tilde{\mathbf{r}}_{k-1}^{(1)}$ and $\tilde{\mathbf{r}}_{k-1}^{(2)}$, since the sum in (10) is bounded. Then, for any $\mathbf{r}_k$ such that $\tilde{\mathbf{r}}_{k-1}^{(1)} \leq \mathbf{r}_k \leq \tilde{\mathbf{r}}_{k-1}^{(2)}$ (component wise), since $\mathbf{s}, \mathbf{s}^m$ are differentiable with bounded derivatives with respect to their arguments, it must hold that $\|\mathbf{s}(\hat{\mathbf{x}}_k, \mathbf{C}(\hat{\mathbf{x}}_{k-1}^E(\mathbf{r}_{k-1}))) - \mathbf{s}^m(\mathbf{x}_k^m, \mathbf{r}_{k-1})\| = \|\hat{\mathbf{y}}_k - \mathbf{y}_k^m\|$ is bounded. Which makes the controller $\mathbf{C}(\mathbf{x}_k^E, \hat{\theta})$ stabilizing for the control system in the

sense of bounded output when $\mathbf{r}_k$ is bounded. Then, it is an admissible one for the infinite horizon c.f. J_{MR}^{∞} with $\gamma < 1$. This proves the claim.

An NN can be used as a controller for nonlinear state-feedback control learning. Nonlinear VRFT is proposed in [41,42] and successfully applied to NN controllers in [41,43–45] but only for output feedback control and not for state-feedback control as in here.

Notice that VRFT control does not need the entire extended state $\tilde{\mathbf{x}}_k^E$ for feedback (i.e. including the virtual states of the ORM), the process' initial states would suffice for this purpose. However, state extension is required for preserving the Markov property of the system (3) in order to ensure the correct collection of the transition samples; this is not possible otherwise without special collection design such as using a zero-order-hold for two-by-two consecutive time samples [43]. Correct transition samples collection is required for adaptive actor-critic tuning approach of the same NN controller that is initially tuned via VRFT.

Notice that in the proposed state-feedback VRFT design, knowledge of the output function $\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k)$ in Equation (1) is again not needed since $\tilde{\mathbf{y}}_k$ is used to calculate the virtual reference $\tilde{\mathbf{r}}_k$, while the controller only uses $\tilde{\mathbf{x}}_k^E$ for feedback purposes.

4. Adaptive Actor-Critic Learning for ORM Tracking Control

If the system dynamics (3) is known, for a finite-time horizon version of the c.f. (4), numerical dynamic programming solutions can be employed backwards in time only with finite state and action spaces of moderate size, an issue referred to as the "curse of dimensionality". For infinite horizon c.f.s, Policy Iteration and Value Iteration [23] can be used even for large and/or continuous state and action spaces, where FAs such as NNs are one option.

If the system dynamics in (3) is unknown, the minimization of the c.f. (4) becomes an RL problem. To solve it model-free, an informative c.f. for each state-action pair is defined, called the Q-function (or action-value function). With this respect, the action-value function of acting $\mathbf{u}_k$ in state $\mathbf{x}_k^E$ and then following the control (policy) $\mathbf{u}_k = \mathbf{C}(\mathbf{x}_k^E)$ is defined as

$$Q^{\mathbf{C}}(\mathbf{x}_k^E, \mathbf{u}_k) = V(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q^{\mathbf{C}}(\mathbf{x}_{k+1}^E, \mathbf{C}(\mathbf{x}_{k+1}^E)). \tag{11}$$

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

$$Q^*(\mathbf{x}_k^E, \mathbf{u}_k^*) = \min_{\mathbf{u}_k}\Big(V(\mathbf{x}_k^E, \mathbf{u}_k) + \gamma Q^*(\mathbf{x}_{k+1}^E, \mathbf{u}_{k+1}^*)\Big), \tag{12}$$

with the optimal controller and optimal Q-function

$$\mathbf{u}_k^* = \mathbf{C}^*(\mathbf{x}_k^E) = \operatorname{argmin}_{\mathbf{C}} Q^{\mathbf{C}}(\mathbf{x}_k^E, \mathbf{u}_k) = \operatorname{argmin}_{\mathbf{u}} Q^*(\mathbf{x}_k^E, \mathbf{u}). \tag{13}$$

Then $J^*(\mathbf{x}_k^E) = Q^*(\mathbf{x}_k^E, \mathbf{u}_k^*)$, where $J^*(\mathbf{x}_k^E) = \min_{\mathbf{u}} J(\mathbf{x}_k^E, \mathbf{u})$ is the minimum value c.f. out of the c.f.s defined in Equation (4). Notice that c.f. (4) encompasses (5) thus making the ORM tracking problem consistent with the above equations. The optimal Q-function can be found using Policy Iteration or Value Iteration in a model-free manner, using, e.g., NNs as FAs. The optimal Q-function estimate and the optimal controller estimate can be updated from the transition samples in several ways: in online/offline mode, batch mode, or sample-by-sample update [23,46]. A particular class of online RL approaches is represented by the temporal difference-based AAC design that differs from the batch PI and VI approaches, as it avoids alternate batch back-up of the Q-function FA and of the controller FA.

4.1. Adaptive Actor-Critic Design

The proposed AAC design is a gradient-based scheme designed to converge to the optimal Q-function and optimal controller estimates. Let the temporal-difference error be measured from data as

$$\delta_k(\mathbf{x}_{k-1}^E, \mathbf{u}_{k-1}) = V(\mathbf{x}_{k-1}^E, \mathbf{u}_{k-1}) + \gamma \hat{Q}_{k-1}(\mathbf{x}_k^E, \mathbf{C}_k(\mathbf{x}_k^E)) - \hat{Q}_{k-1}(\mathbf{x}_{k-1}^E, \mathbf{u}_{k-1}) \tag{14}$$

where the continuous function $\hat{Q}_k(\cdot,\cdot)$ in its arguments is the Q-function estimate at time k, time at which some controller $\mathbf{C}_k(\mathbf{x}_k)$ is also available. *From this point onward, for notation simplicity,* $\mathbf{x}_k$ *or plain* $\mathbf{x}$ *are used instead of* ${\mathbf{x}_k}^E$ The proposed AAC design attempts the Q-function update to online minimize the c.f. $E_{c,k} = 0.5\ \delta_k{}^2$, while the controller attempts to online minimize the Q-function using gradient descent. Taxonomically, the proposed AAC belongs to the online Policy Iteration schemes where the policy evaluation step (of Bellman error residual minimization type) interleaves with the policy improvement step. The update laws for the AAC design from input-state data are:

$$\mathbf{u}_k = \mathbf{C}_k(\mathbf{x}) = \mathbf{C}_{k-1}(\mathbf{x}) - \alpha_a \cdot \left.\frac{\partial \hat{Q}_{k-1}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{u}_{k-1}}, \forall \mathbf{x}, \tag{15}$$

$$\begin{aligned} &\hat{Q}_k(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) = \hat{Q}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \alpha_c \delta_k = \\ &= \hat{Q}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \alpha_c \Big(V(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \gamma \hat{Q}_{k-1}(\mathbf{x}_k, \mathbf{u}_k = C_k(\mathbf{x}_k)) - \hat{Q}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \Big), \end{aligned} \tag{16}$$

where $\alpha_a > 0$, $\alpha_c > 0$ are learning rates. The controller $\mathbf{C}(\mathbf{x}_k)$ is imagined as a function (or as an infinitely dense table) mapping any state to a control action.

Comment 1: In particular, for any admissible controller $\mathbf{C}_0$, repeated calls of (16), under proper exploration (translated to visiting all the pairs $(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \in X^E \times U$ often and to generating the sample $\mathbf{x}_k$), and under proper selection of $\alpha_c > 0$, will update $\hat{Q}_k(\mathbf{x}, \mathbf{u})$, $\forall \mathbf{x} \in X^E$ until $\delta_k = 0$ at which point (11) must hold and the converged $Q^{\mathbf{C}_0}(\mathbf{x}_k, \mathbf{u}_k)$ evaluates $\mathbf{C}_0$. This is an *online off-policy model-free* policy evaluation step. Then $Q_0(\mathbf{x}, \mathbf{u}) = Q^{\mathbf{C}_0}(\mathbf{x}, \mathbf{u})$ can be an initialization for AAC. Whereas, an initial admissible controller can be obtained for example using VRFT as shown later in the case study.

Comment 2: The converged Q-function of an admissible control $\mathbf{C}_0$, be it $Q^{\mathbf{C}_0}(\mathbf{x}_k, \mathbf{u}_k)$, is positive by definition since it accumulates stage costs $V > 0$. Moreover, it is always greater than the optimal Q-function, i.e., $Q^{\mathbf{C}_0}(\mathbf{x}_k, \mathbf{u}_k) \geq Q^*(\mathbf{x}_k, \mathbf{u}_k) > 0$ and obeys the Bellman equation.

Comment 3: From (15), it follows that

$$Q_{k-1}(\mathbf{x}, \mathbf{C}_k(\mathbf{x})) \leq Q_{k-1}(\mathbf{x}, \mathbf{C}_{k-1}(\mathbf{x})), \forall \mathbf{x} \in X^E \tag{17}$$

for a small enough $\alpha_a > 0$.

Lemma 1. Starting from an admissible controller $\mathbf{C}_0$ with corresponding Q-function initialization $\hat{Q}_0(\mathbf{x}, \mathbf{u}) = Q^{\mathbf{C}_0}(\mathbf{x}, \mathbf{u})$, the sequence $\left\{\hat{Q}_k(\mathbf{x}, \mathbf{u})\right\}$ is monotonic and non-increasing ensuring that

$$\hat{Q}_k(\mathbf{x}, \mathbf{u}) \leq \hat{Q}_{k-1}(\mathbf{x}, \mathbf{u}), (\mathbf{x}, \mathbf{u}) \in X^E \times U \tag{18}$$

Proof. $\hat{Q}_0(\mathbf{x}, \mathbf{u})$ is initialization for Equation (16) and obeys the Bellman Equation (11) for the admissible controller $\mathbf{C}_0$, i.e.,

$$\hat{Q}_0(\mathbf{x}_k, \mathbf{u}_k) = V(\mathbf{x}_k, \mathbf{u}_k) + \gamma \hat{Q}_0(\mathbf{x}_{k+1}, \mathbf{C}_0(\mathbf{x}_{k+1})). \tag{19}$$

Starting the AAC update law from initial state $\mathbf{x}_0$, $\mathbf{C}_1(\mathbf{x})$ is updated first by Equation (15), then it follows that

$$\hat{Q}_1(\mathbf{x}_0,\mathbf{u}) = \hat{Q}_0(\mathbf{x}_0,\mathbf{u}) + \alpha_c\left(\underbrace{V(\mathbf{x}_0,\mathbf{u}) + \gamma\hat{Q}_0(\mathbf{x}_1,\mathbf{C}_1(\mathbf{x}_1))}_{\overset{(18)}{=}\hat{Q}_0(\mathbf{x}_0,\mathbf{C}_0(\mathbf{x}_0))} - \hat{Q}_0(\mathbf{x}_0,\mathbf{u})\right) = \hat{Q}_0(\mathbf{x}_0,\mathbf{u}) \le \hat{Q}_0(\mathbf{x}_0,\mathbf{u}), \forall\mathbf{u}. \tag{20}$$

Since $\mathbf{x}_0$ can be any state $\mathbf{x} \in X^E$, then Equation (18) holds for $k = 1$. Assume by induction that (18) holds for some k. Using *Comment 3*, it follows that

$$\begin{aligned}&\hat{Q}_{k+1}(\mathbf{x}_k,\mathbf{u}_k) = \hat{Q}_k(\mathbf{x}_k,\mathbf{u}_k) + \alpha_c\Big(V(\mathbf{x}_k,\mathbf{u}_k) + \gamma\hat{Q}_k(\mathbf{x}_{k+1},\mathbf{C}_{k+1}(\mathbf{x}_{k+1})) - \hat{Q}_k(\mathbf{x}_k,\mathbf{u}_k)\Big)\\ &\le \hat{Q}_k(\mathbf{x}_k,\mathbf{u}_k) + \alpha_c\Big(V(\mathbf{x}_k,\mathbf{u}_k) + \gamma\hat{Q}_k(\mathbf{x}_{k+1},\mathbf{C}_k(\mathbf{x}_{k+1})) - \hat{Q}_k(\mathbf{x}_k,\mathbf{u}_k)\Big) \le\\ &\overset{(18)}{\le} \hat{Q}_{k-1}(\mathbf{x}_k,\mathbf{u}_k) + \alpha_c\Big(V(\mathbf{x}_k,\mathbf{u}_k) + \gamma\hat{Q}_{k-1}(\mathbf{x}_{k+1},\mathbf{C}_k(\mathbf{x}_{k+1})) - \hat{Q}_{k-1}(\mathbf{x}_k,\mathbf{u}_k)\Big) = \hat{Q}_k(\mathbf{x}_k,\mathbf{u}_k),\end{aligned} \tag{21}$$

and since $\mathbf{x}_k, \mathbf{u}_k$ can be any pair $(\mathbf{x}, \mathbf{u}) \in X^E \times U$, the conclusion of *Lemma 1* follows.

Theorem 2. Let $\hat{Q}_0(\mathbf{x}, \mathbf{u}) > 0$, finite for any finite argument) be an initialization for the Q-function of an initial admissible controller $\mathbf{C}_0$. Starting with any $\mathbf{x}_0$ the control $\mathbf{u}_0 = \mathbf{C}_0(\mathbf{x}_0)$ is applied to the process. Specifically, the AAC update laws ((15), (16)) ensures that at time $k = 1$, $\mathbf{C}_1$ is updated from $\mathbf{C}_0$, $\hat{Q}_0(\mathbf{x}, \mathbf{u})$ is updated with (16) using $\mathbf{u}_1 = \mathbf{C}_1(\mathbf{x}_1)$ in the right-hand side, the control $\mathbf{u}_1$ is sent to the process, and then $k \leftarrow 2$, with the above strategy repeated for subsequent times. Claim: The feedback control system under time-varying control $\mathbf{C}_k$ is stabilized for $\gamma < 1$ and asymptotically stabilized for $\gamma = 1$.

Proof. It is valid for the first three time steps that

$$\hat{Q}_1(\mathbf{x}_0,\mathbf{u}_0) \overset{(16)}{=} \hat{Q}_0(\mathbf{x}_0,\mathbf{u}_0) + \alpha_c[V(\mathbf{x}_0,\mathbf{u}_0) + \gamma\hat{Q}_0(\mathbf{x}_1,\mathbf{u}_1 = \mathbf{C}_1(\mathbf{x}_1)) - \hat{Q}_0(\mathbf{x}_0,\mathbf{u}_0)] \overset{(18)}{\le} \hat{Q}_0(\mathbf{x}_0,\mathbf{u}_0). \tag{22a}$$

$$\hat{Q}_2(\mathbf{x}_1,\mathbf{u}_1) \overset{(16)}{=} \hat{Q}_1(\mathbf{x}_1,\mathbf{u}_1) + \alpha_c[V(\mathbf{x}_1,\mathbf{u}_1) + \gamma\hat{Q}_1(\mathbf{x}_2,\mathbf{u}_2 = \mathbf{C}_2(\mathbf{x}_2)) - \hat{Q}_1(\mathbf{x}_1,\mathbf{u}_1)] \overset{(18)}{\le} \hat{Q}_1(\mathbf{x}_1,\mathbf{u}_1). \tag{22b}$$

$$\hat{Q}_3(\mathbf{x}_2,\mathbf{u}_2) \overset{(16)}{=} \hat{Q}_2(\mathbf{x}_2,\mathbf{u}_2) + \alpha_c[V(\mathbf{x}_2,\mathbf{u}_2) + \gamma\hat{Q}_2(\mathbf{x}_3,\mathbf{u}_3 = \mathbf{C}_3(\mathbf{x}_3)) - \hat{Q}_2(\mathbf{x}_2,\mathbf{u}_2)] \overset{(18)}{\le} \hat{Q}_2(\mathbf{x}_2,\mathbf{u}_2). \tag{22c}$$

Cancelling the same terms in both sides of Equation (22a–c), since $\alpha_c > 0$, it follows that the sums in square parentheses are negative. These sums are further refined using *Lemma 1* as

$$V(\mathbf{x}_0,\mathbf{u}_0) + \gamma\hat{Q}_0(\mathbf{x}_1,\mathbf{u}_1) \le \hat{Q}_0(\mathbf{x}_0,\mathbf{u}_0), \tag{23a}$$

$$V(\mathbf{x}_1,\mathbf{u}_1) + \gamma\hat{Q}_1(\mathbf{x}_2,\mathbf{u}_2) \le \hat{Q}_1(\mathbf{x}_1,\mathbf{u}_1) \overset{(18)}{\le} \hat{Q}_0(\mathbf{x}_1,\mathbf{u}_1), \tag{23b}$$

$$V(\mathbf{x}_2,\mathbf{u}_2) + \gamma\hat{Q}_2(\mathbf{x}_3,\mathbf{u}_3) \le \hat{Q}_2(\mathbf{x}_2,\mathbf{u}_2) \overset{(18)}{\le} \hat{Q}_1(\mathbf{x}_2,\mathbf{u}_2), \tag{23c}$$

Using (23c) in (23b) it follows that $V(\mathbf{x}_1,\mathbf{u}_1) + \gamma V(\mathbf{x}_1,\mathbf{u}_1) + \gamma^2\hat{Q}_2(\mathbf{x}_3,\mathbf{u}_3) \le \hat{Q}_0(\mathbf{x}_1,\mathbf{u}_1)$, which used in (23a) results in$V(\mathbf{x}_0,\mathbf{u}_0) + \gamma V(\mathbf{x}_1,\mathbf{u}_1) + \gamma^2 V(\mathbf{x}_2,\mathbf{u}_2) + \gamma^3\hat{Q}_2(\mathbf{x}_3,\mathbf{u}_3) \le \hat{Q}_0(\mathbf{x}_0,\mathbf{u}_0)$. Extending the exemplified reasoning backwards from infinity it follows that

$$\lim_{N\to\infty}\left(\sum_{i=0}^{N-1}\gamma^i V(\mathbf{x}_i,\mathbf{u}_i) + \gamma^N\hat{Q}_{N-1}(\mathbf{x}_N,\mathbf{u}_N)\right) \le \hat{Q}_0(\mathbf{x}_0,\mathbf{u}_0). \tag{24}$$

Since $\lim_{N\to\infty}\gamma^N\hat{Q}_{N-1}(\mathbf{x}_N,\mathbf{u}_N) = 0$ because $\hat{Q}_{N-1}(\mathbf{x}_N,\mathbf{u}_N)$ is bounded since resulting from a non-increasing sequence, it follows that $\lim_{N\to\infty}\sum_{i=0}^{N-1}\gamma^i V(\mathbf{x}_i,\mathbf{u}_i) \le \hat{Q}_0(\mathbf{x}_0,\mathbf{u}_0)$ is finite. The left term in the inequality is the cost of using the controller $\mathbf{u}_0 = \mathbf{C}_0(\mathbf{x}_0), \mathbf{u}_1 = \mathbf{C}_1(\mathbf{x}_1), \ldots$. Then it follows that

the control system remains stable under the time-varying control of the AAC updates. Moreover, for $\gamma = 1$, the sequence $\left\{ \sqrt{V(\mathbf{x}_i, \mathbf{u}_i)} \right\}, i = \overline{0, \infty}$ must be square-summable which implies that the control system is asymptotically stabilized by $\mathbf{C}_k$, thus proving the claim of the theorem.

Comment 4: The above proof resembles the stabilizing action-dependent value iteration of [30], but here relies on gradient-based updates of the Q-function estimate and of the controller, rather than on its minimization. The stability result of the *Theorem 2* is valid under continuous updates of the AAC laws (15), (16) under no exploration. It ensures that, starting from an admissible controller $\mathbf{C}_0$, the AAC updates (15), (16) preserve the control system stability. Since exploratory controls $\mathbf{u}_k$ are critical, a compromising solution is to perform the controller update (15) only for non- exploratory sampling instants, while the Q-function estimates are continuously updated per (16).

4.2. AAC Using Neural Networks As Approximators

In practice, specific FAs such as NNs are employed as approximators for the Q-function and for the controller, respectively. Using NNs as FAs (implying nonlinear features parameterization), the convergence of the learning scheme depends on a large extent to the selection of the learning parameters. However, the advantage of using generic NN architectures is that no manual or automatic feature selection is needed for parameterizing the Q-function and controller estimates.

The proposed AAC design herein uses two NNs to approximate the Q-function (critic, referred to herein as Q-NN), and the controller (actor, referred to herein as C-NN), respectively. Assume an initial admissible NN VRFT state-feedback controller exists. Let the critic NN FA and controller NN FA be parameterized as $\hat{Q}_k(\mathbf{x}_k^E, \mathbf{u}_k, \theta_k^c)$ and $\mathbf{C}_k(\mathbf{x}_k^E, \theta_k^a)$, respectively. With three-layer feed-forward NNs having one hidden layer, fully connected with bias, the critic and the controller are modeled by:

$$\hat{Q}_k = W_{c,n_{hc}+1}^k + \sum_{i=1}^{n_{hc}} W_{c,i}^k \sigma_i \left(\sum_{j=1}^{n_{ic}+1} V_{c,ji}^k I_j^k \right), \tag{25}$$

$$u_k^l = W_{a,n_{ha}+1}^{k,l} + \sum_{i=1}^{n_{ha}} W_{a,i}^{k,l} \sigma_i \left(\sum_{j=1}^{n_{ia}+1} V_{a,ji}^k I_j^k \right), l = \overline{1, m}, \tag{26}$$

with $\mathbf{W}_c = \left[W_{c,1}^k \ \ldots \ W_{c,n_{hc}+1}^k \right]^T$—the critic output layer weights having n_{hc} hidden neurons, $\mathbf{V}_c = \left[V_{c,ji}^k \right]_{i=1\ldots n_{hc}, j=1\ldots n_{ic}+1}$—the critic hidden layer weights matrix, $\mathbf{I}_c^k = \left[I_1^k \ldots I_{n_{ic}+1}^k \right]^T = \left[\left(\mathbf{x}_k^E \right)^T \ (\mathbf{u}_k)^T \ 1 \right]^T$—the critic input vector of size n_{ic} + 1 (the bias input is constant 1), $\mathbf{W}_a^{k,l} = \left[W_{a,1}^{k,l} \ \ldots W_{a,n_{ha}+1}^{k,l} \right]^T, l = 1..m$ – the output layer weights of the l-th controller output $\mathbf{u}_k = \left[u_k^1 \ldots u_k^m \right]^T$, having n_{ha} hidden neurons, and $\mathbf{V}_a = \left[V_{a,ji}^k \right]_{i=1\ldots n_{ha}, j=1\ldots n_{ia}+1}$—the controller hidden layer weights matrix, $\mathbf{I}_a^k = \left[I_1^k \ldots I_{n_{ia}+1}^k \right]^T = \left[\left(\mathbf{x}_k^E \right)^T \ 1 \right]^T$ – the controller input vector of size n_{ia} + 1 (bias input 1 included). $\sigma_i = \tanh_i$, is the hyperbolic tangent activation function at the output of i^{th} hidden neuron. The controller and critic are cascaded, with the actor output $\mathbf{u}_k$ as a part of the critic input alongside $\mathbf{x}_k^E$ The actor and critic weights are *formally* parameterized as $\theta_a^k = \left[\left(\mathbf{W}_a^{k,l} \right)^T \left(\mathbf{V}_a^k \right)^T \right]^T$ and $\theta_c^k = \left[\left(\mathbf{W}_c^k \right)^T \left(\mathbf{V}_c^k \right)^T \right]^T$, respectively.

The AAC's gradient descent tuning rules for the critic (25) and controller (26) as parameterized variants of (15) and (16) are

$$\begin{cases} \theta_k^c = \theta_{k-1}^c + \alpha_c \cdot \frac{\partial \hat{Q}(\mathbf{x},\mathbf{u},\theta)}{\partial \theta}\Big|_{(\mathbf{x}_{k-1}^E,\mathbf{u}_{k-1},\theta_c^{k-1})} \cdot \delta_k, \text{ detailed as :} \\ W_{c,i}^k = W_{c,i}^{k-1} + \delta_k \alpha_c \times \begin{cases} \sigma_i(\sum\limits_{j=1}^{n_{ic}+1} V_{c,ji}^{k-1} I_j^{k-1}), & \text{if } i \neq n_{hc}+1, \\ 1, & \text{if } i = n_{hc}+1, \end{cases} \\ V_{c,ji}^k = V_{c,ji}^{k-1} + \delta_k \alpha_c W_{c,i}^{k-1} I_j^{k-1} \sigma_i'(\sum\limits_{j=1}^{n_{ic}+1} V_{c,ji}^{k-1} I_j^{k-1}), \end{cases} \tag{27}$$

$$\begin{cases} \theta_k^a = \theta_{k-1}^a - \alpha_a \cdot \frac{\partial \hat{Q}(\mathbf{x},\mathbf{u},\theta)}{\partial \mathbf{u}}\Big|_{(\mathbf{x}_{k-1}^E,\mathbf{u}_{k-1},\theta_c^{k-1})} \cdot \frac{\partial \mathbf{u}(\mathbf{x},\theta)}{\partial \theta}\Big|_{(\mathbf{x}_{k-1}^E,\theta_a^{k-1})}, \text{as :} \\ W_{a,i}^{k,l} = W_{a,i}^{k-1,l} - \alpha_a \underbrace{\left(\sum_{i=1}^{n_{hc}} W_{c,i}^{k-1} V_{c,\xi i}^{k-1} \sigma_i'(\sum_{j=1}^{n_{ic}+1} V_{c,ji}^{k-1} I_j^{k-1})\right)}_{\frac{\partial \hat{Q}(\mathbf{x},\mathbf{u},\theta)}{\partial \mathbf{u}}|_{(\mathbf{x}_{k-1}^E,\mathbf{u}_{k-1},\theta_c^{k-1})}} \times \begin{cases} \sigma_i(\sum\limits_{j=1}^{n_{ia}+1} V_{a,ji}^{k-1} I_j^{k-1}), & \text{if } i \neq n_{ha}+1, \\ 1, & \text{if } i \neq n_{ha}+1, \end{cases} \\ V_{a,ji}^k = V_{a,ji}^k - \alpha_a \frac{\partial \hat{Q}(\mathbf{x},\mathbf{u},\theta)}{\partial \mathbf{u}}\Big|_{(\mathbf{x}_{k-1}^E,\mathbf{u}_{k-1},\theta_c^{k-1})} W_{a,i}^{k-1,l} I_j^{k-1} \sigma'_i(\sum\limits_{j=1}^{n_{ia}+1} V_{a,ji}^{k-1} I_j^{k-1}), \end{cases} \tag{28}$$

with α_a, α_c—the learning rate magnitudes of the critic and controller training rules, respectively and σ_i' is the derivative of σ_i w.r.t its argument and ξ is the index of u_k^l in the $\mathbf{I}_c^k$.

In many practical applications, the designer chooses to perform either full or partial adaptation of the NNs' weights, the latter implying only output weights adaptation. In this latter case, the Q-NN and C-NN parameterizations are:

$$\begin{aligned} \hat{Q}_k(\mathbf{x}_k^E, \mathbf{u}_k) &= \left(\mathbf{W}_c^k\right)^T \sigma(\mathbf{V}_c^k[\left(\mathbf{x}_k^E\right)^T \ (\mathbf{u}_k)^T \ 1]^T) = \left(\mathbf{W}_c^k\right)^T \mathbf{\Phi}_c^k(\mathbf{x}_k^E, \mathbf{u}_k), \\ \mathbf{u}_k &= \mathbf{C}_k(\mathbf{x}_k^E) = \left(\mathbf{W}_a^k\right)^T \sigma(\mathbf{V}_a^k[\left(\mathbf{x}_k^E\right)^T 1]^T) = \left(\mathbf{W}_a^k\right)^T \mathbf{\Phi}_c^k(\mathbf{x}_k^E), \end{aligned} \tag{29}$$

where $\mathbf{\Phi}_c^k(\mathbf{x}_k^E, \mathbf{u}_k)$, $\mathbf{\Phi}_a^k(\mathbf{x}_k^E)$ are the matrices of basis functions (or input features) and ${\mathbf{W}_c}^k$, ${\mathbf{W}_a}^k$ are the tunable output weights parameters, rendering $\hat{Q}_k, \mathbf{u}_k$ as linear combination of basis functions. This linear parameterization simplifies the convergence analysis but also requires manual features selection and training as a disadvantage.

As it is well-known, the AAC architecture performs online with the C-NN sending controls to the process and the Q-NN serving both to estimate the Q-function and to adaptively tune the C-NN. The closed-loop control system with the process (3) combined with the AAC tuning rules (27), (28) has the unique property that its dynamics is mainly driven by the reference input $\mathbf{r}_k$ viewed as a particular state of the extended state vector and, possibly, by exogenous unknown disturbances. Since $\mathbf{r}_k$ is user selectable, it can be used to drive the control system in a wide operating range to ensure efficient exploration of the state space. Enhanced exploration of the domain $X^E \times U$ can be performed by trying random actions in every state, usually as additive uniform random actions.

4.3. Convergence of the AAC Learning Scheme with NNs

While the results from Section 4.1 are formulated under generic functions for the Q-function and for the controller, the convergence to the optimal controller and optimal Q-function is not ensured. In the following, the convergence to of the AAC learning with NNs is shown. Linear parameterization

is the most widely used and supports tractable analysis. Let the output weight parameterization (29) of the Q-NN and C-NN lead to the update laws

$$\begin{aligned}\mathbf{W}_c^k &= \mathbf{W}_c^{k-1} + \alpha_c \delta_k \boldsymbol{\Phi}_c^k(\mathbf{x}_{k-1}^E, \mathbf{u}_{k-1}),\\ \mathbf{W}_a^k &= \mathbf{W}_a^{k-1} - \alpha_a \boldsymbol{\Phi}_a^k(\mathbf{x}_{k-1}^E)\left(\mathbf{W}_c^{k-1}\right)^T \frac{\partial \boldsymbol{\Phi}_c}{\partial \mathbf{u}}(\mathbf{x}_{k-1}^E, \mathbf{u}_{k-1}),\end{aligned} \tag{30}$$

be compactly written as

$$\begin{aligned}\mathbf{W}_c^k &= \mathbf{W}_c^{k-1} + \alpha_c \delta_k \boldsymbol{\Phi}_c^{k-1},\\ \mathbf{W}_a^k &= \mathbf{W}_a^{k-1} - \alpha_a \boldsymbol{\Phi}_a^{k-1}\left(\mathbf{W}_c^{k-1}\right)^T \boldsymbol{\Phi}_{c,u}^{k-1},\end{aligned} \tag{31}$$

where $\delta_k = \underbrace{V(\mathbf{x}_{k-1}^E, \mathbf{u}_{k-1})}_{U^{k-1}} + \left(\mathbf{W}_c^{k-1}\right)^T\left(\gamma \boldsymbol{\Phi}_c^k - \boldsymbol{\Phi}_c^{k-1}\right)$ Let the weights of the optimal Q-NN and optimal C-NN be $\mathbf{W}_c{}^*$, $\mathbf{W}_a{}^*$, the estimation errors being $\tilde{\mathbf{W}}_c^k = \mathbf{W}_c^k - \mathbf{W}_c^*, \tilde{\mathbf{W}}_a^k = \mathbf{W}_a^k - \mathbf{W}_a^*$ that render the estimation error dynamics

$$\begin{aligned}\tilde{\mathbf{W}}_c^k &= \tilde{\mathbf{W}}_c^{k-1} + \alpha_c \delta_k \boldsymbol{\Phi}_c^{k-1},\\ \tilde{\mathbf{W}}_a^k &= \tilde{\mathbf{W}}_a^{k-1} - \alpha_a \boldsymbol{\Phi}_a^{k-1}\left(\mathbf{W}_c^{k-1}\right)^T \boldsymbol{\Phi}_{c,u}^{k-1},\end{aligned} \tag{32}$$

Some assumptions follow:

A7. Let the estimation error of the critic be denoted as $\zeta_c^{k-1} = \left(\tilde{\mathbf{W}}_c^{k-1}\right)^T \boldsymbol{\Phi}_c^{k-1}$, let the critic's and actor's hidden activation layers be bounded as $\|\boldsymbol{\Phi}_c^{k-1}\|^2 \le \overline{\varphi}_c, \|\boldsymbol{\Phi}_a^{k-1}\|^2 \le \overline{\varphi}_a$ and let the critic's activation layer derivative w.r.t. $\mathbf{u}$ be bounded as $\|\boldsymbol{\Phi}_{c,u}^{k-1}\|^2 \le \overline{\varphi}_{c,u}$ where Frobenius norm was used, which is equivalent to the Euclidean norm when it is applied to vectors. The above upper bounds follow since the activation functions are bounded and so are their derivatives.

Theorem 3. Under *A7*, the AAC learning scheme converges to a vicinity of the optimal controller and optimal Q-function since the estimation errors $\tilde{\mathbf{W}}_c, \tilde{\mathbf{W}}_a$ are uniformly ultimately bounded provided that $\alpha_c > 4\overline{\varphi}_c^2 - 2\overline{\varphi}_c + 1$, for $\overline{\varphi}_c > 1$.

Proof: See Appendix B.

Comment 5. Notice that the temporal difference error δ_k is calculated in terms of the Q-NN's output. Then δ_k is backpropagated to correct the Q-NN weights. Moreover, δ_k is further backpropagated to correct the C-NN weights, since the C-NN output is an input to the Q-NN. Hence, the resulted AAC architecture belongs to the deep learning approaches (the architecture is presented in Figure 1b of the next Section).

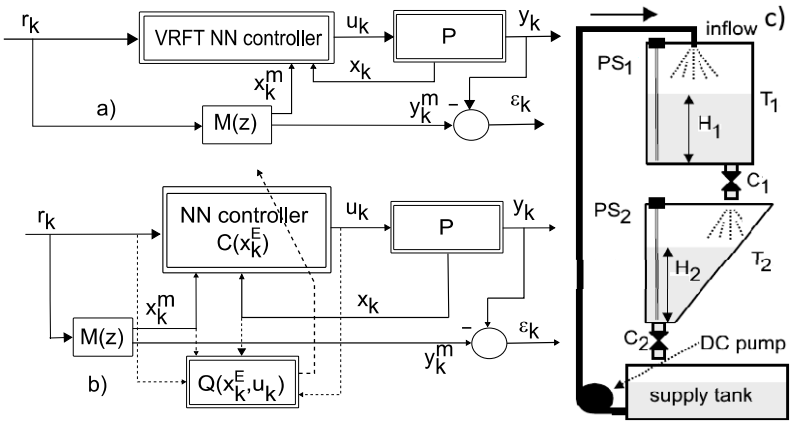

Figure 1. (**a**) control system with the VRFT controller; (**b**) control system with the VRFT NN controller further tuned by AAC design in a deep learning architecture; (**c**) the vertical tank system.

4.4. Summary of the Mixed VRFT-AAC Design Approach

The steps of the VRFT-AAC design approach are summarized next:

S1. Collect input-state-output samples from the open-loop stable process (1) in a dataset $D = \left\{\tilde{\mathbf{u}}_k, \tilde{\mathbf{x}}_k, \tilde{\mathbf{y}}_k\right\} \subset U \times X \times Y, k = \overline{0, N-1}$ where $\tilde{\mathbf{u}}_k$ persistently exciting, under conditions of *A5*.

S2. Obtain the initial state-feedback VRFT controller by minimizing the c.f. in Equation (6) as $\theta_k^a = \underset{\theta}{\text{argmin}} J_{VR}^N(\theta)$. When NNs are used, minimization of the c.f. (6) is equivalent to training the NN. The obtained controller is $\mathbf{C}_k(\mathbf{x}_k^E, \theta_k^a)$, which is a controller for both the process (1) and for the extended process (3). It is also a close initialization to the optimal controller that minimizes J_{MR}^∞ from (5), since VRFT identifies a controller that approximately minimizes J_{MR}^N from (10) as a finite horizon version of J_{MR}^∞. This is supported by *Theorem 1*.

S3. Close the control system loop on process (1) (it is equivalent to closing it on extended process (3)) using controller $\mathbf{C}_k(\mathbf{x}_k^E, \theta_k^a)$. The architecture is presented in Figure 2b). Use update (16) (in explicitly parameterized form, use (28)) under an exploratory reference input $\mathbf{r}_k$ in order to learn the Q-function of the controller $\mathbf{C}_k(\mathbf{x}_k^E, \theta_k^a)$. This serves as properly initializing $\hat{Q}_k(\mathbf{x}_k^E, \mathbf{u}_k, \theta_k^c)$ for the subsequent AAC tuning.

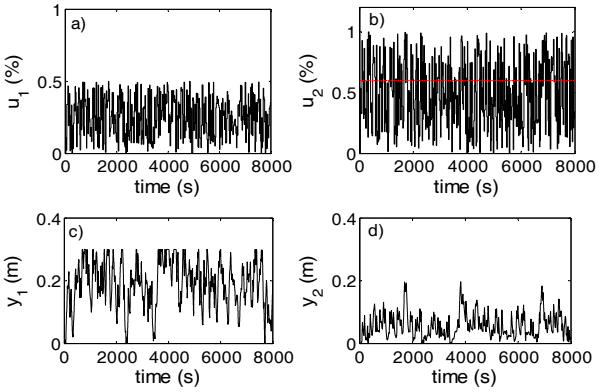

Figure 2. VRFT initial experiment data collection. Red line in (**b**) is the dead-zone threshold for u_2.

S4. Use the updates (15), (16), (27), (28), in explicitly parameterized form), in this exact order and under an exploratory reference input $\mathbf{r}_k$, to learn the optimal controller $\mathbf{C}^*(\mathbf{x}_k^E, \theta_k^{a*} = \mathbf{W}_a^*)$ and the optimal Q-function $\hat{Q}^*(\mathbf{x}_k^E, \mathbf{u}_k, \theta_k^{c*} = \mathbf{W}_c^*)$. Using the above updates for a finite time on a random learning scenario is called a learning episode.

S5. After every learning episode, measure the tracking performance on a standard test scenario. When the prescribed number of maximum tests is reached or the tracking performance on the standard scenario is not improving anymore, the controller learning is stopped. Otherwise, proceed to the next learning episode.

All implementation details of the above VRFT-AAC design are presented in the following Section 5.1 when validation is performed on the complex multivariable tank system case study.

5. Validation Case Study

5.1. AAC Design for a MIMO Vertical Tank System

The controlled process is a vertical MIMO two-tank system (Figure 1c) built around a three-tank laboratory equipment [47] with the continuous-time state-space equations

$$\begin{aligned} \dot{H}_1 &= \tfrac{k}{aw} u_1 - \tfrac{1}{aw}\overline{C}_1 H_1^{\alpha 1}(2.5\widetilde{u}_2 - 0.5), \\ \dot{H}_2 &= \tfrac{1}{aw}\overline{C}_1 H_1^{\alpha 1}(2.5\widetilde{u}_2 - 0.5) - \frac{1}{cw + \frac{H_2}{H_{2\max}} bw}\overline{C}_2 H_2^{\alpha 2}, \\ \widetilde{u}_2 &= \min(\max(u_2, 0.6), 1), \end{aligned} \tag{33}$$

with $a = 0.25$ [m], $w = 0.035$ [m], $c = 0.1$ [m], $b = 0.345$ [m] and $H_{1\max} = H_{2\max} = 0.35$ [m]. $x_1 = y_1 = H_1 \in [0, H_{1\max}]$ and $x_2 = y_2 = H_2 \in [0, H_{2\max}]$ are the water levels in the two tanks considered as system states and controlled outputs. The control inputs $u_1, u_2 \in [0, 1]$ (also expressible in [%]) are the duty cycles of the pump direct current (DC) motor and of the electrically controlled valve C_1, respectively. $k = 1.66 \cdot 10^{-4} [m^3/(s \cdot \%)]$ is the gain from the pump input to the inflow, $\overline{C}_1 = 5.65 \cdot 10^{-5} [m^{3-\alpha 1}/s]$, $\overline{C}_2 = 8 \cdot 10^{-5} [m^{3-\alpha 2}/s]$ are the resistances of the outflow orifices of the first (upper) and second (lower) tank, called T_1 and T_2, respectively, and $\alpha_1 = 0.29$, $\alpha_2 = 0.22$. The third equality in Equation (33) reflects the dead-zone plus saturation in the second control input u_2.

Features of this process include: no water level setpoint for the tank T_2 can be set if the water level in T_1 is zero; the electrical valve controlling the outflow from T_1 (which is inflow to T_2) is changed by $\widetilde{u}_2(u_2)$; no setpoint can be tracked for each tank if there is more outflow than inflow; T_1's outflow has a minimum value and can be zero only when $H_1 = 0$, as per first equality in (33). T_2's dynamics is slower than T_1's. Proper selection of the parameters $\overline{C}_1, \overline{C}_2$ through manual valves allows feasible control trajectories in the constrained input-state-output space. Discretization of (33) reveals its Markov form. The water level is measured using piezoelectric sensors (PS_1 and PS_2 in Figure 1c). Protection logic disables the pump voltage when water level exceeds the upper bound. The sampling period used for control experiments is $T_s = 0.5$ s. Model (33) is not used for control design.

For VRFT-based control design, the ORM is selected as the ZOH discretization of $\mathbf{M}(s) = diag(M_1(s), M_2(s))$. $M_1(s) = \omega_0^2/(s^2 + 2\varsigma\omega_0 s + \omega_0^2)$, with the damping factor $\varsigma = 1.0$ and the natural frequency $\omega_0 = 0.5 rad/s$ selects the speed and shape of the desired response $y_{k,1}^m$ while similar $M_2(s)$ with $\varsigma = 1.0$ and $\omega_0 = 0.2 rad/s$ describes $y_{k,2}^m$. For collecting the open-loop input-state-output data $\left\{\tilde{\mathbf{u}}_k, \tilde{\mathbf{x}}_k, \tilde{\mathbf{y}}_k\right\}_{k=0\ldots N-1}$, 16,000 samples have been generated from 16,000 samples of a uniformly random sequence of persistently exciting steps lasting for 20 s $\tilde{\mathbf{u}}_k = [\widetilde{u}_{k,1}\ \widetilde{u}_{k,2}]^T \in [0, 0.5] \times [0, 1]$, for an experiment of 8000 s. The collected data is displayed in Figure 2 and ensures the exploratory conditions from Assumption *A5*.

The controllable canonical state-space realizations ($\mathbf{A}_1$, $\mathbf{B}_1$, $\mathbf{C}_1$, $\mathbf{D}_1$) and ($\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2, \mathbf{D}_2$) and of $M_1(z)$ and $M_2(z)$ are, respectively:

$$\begin{aligned} \mathbf{A}_1 &= \begin{pmatrix} 1.5576 & -0.6065 \\ 1 & 0 \end{pmatrix}, \mathbf{B}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \mathbf{C}_1 = \begin{pmatrix} 0.0265 \\ 0.0224 \end{pmatrix}^T, \mathbf{D}_1 = 0, \\ \mathbf{A}_2 &= \begin{pmatrix} 1.8097 & -0.8187 \\ 1 & 0 \end{pmatrix}, \mathbf{B}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \mathbf{C}_2 = \begin{pmatrix} 0.0047 \\ 0.0044 \end{pmatrix}^T, \mathbf{D}_2 = 0, \end{aligned} \tag{34}$$

The ORM state will then be $\mathbf{x}_k^m = [x_{k,1}^m\ x_{k,2}^m\ x_{k,3}^m\ x_{k,4}^m]^T$. The virtual reference $\bar{\mathbf{r}}_k = \mathbf{M}(z)^{-1}\tilde{\mathbf{y}}_k$ is used as input to the state-space models (34) to obtain the ORM's virtual states $\tilde{\mathbf{x}}_k^m$. The extended virtual state vector $\tilde{\mathbf{x}}_k^E = [\left(\tilde{\mathbf{x}}_k\right)^T \left(\tilde{\mathbf{x}}_k^m\right)^T \left(\tilde{\mathbf{r}}_k\right)^T]^T \in \Re^8$ is used to offline compute the C-NN via VRFT by fitting the inputs $\tilde{\mathbf{x}}_k^E$ to the outputs $\tilde{\mathbf{u}}_k$. Note that using two second order ORM's produces four states in the extended state-space, which is disadvantageous. The ORM's orders should to be as low as possible (usually one) but a second order model offers greater flexibility in output response shaping.

The VRFT C-NN architecture (Figure 1a) is a feedforward 8–10–2 fully connected one with biases having $n_{ha} = 10$ hidden neurons with tanh( ) activation function and the output activation functions are linear. The weights are initialized with random uniform numbers in $[-1.5, 1.5]$. Since the NN training is performed offline, standard gradient backpropagation training with Levernberg-Marquardt [48] is used for maximum 50 epochs to learn a stabilizing VRFT C-NN controller $\mathbf{C}(\mathbf{x}_k^E)$ for the MIMO control system, that minimizes J_{VR}^N. 80% of the data is effectively used for training while the rest of 20% serves as validation data. Early stopping is used after six consecutive increases in the mean sum of squared error evaluated on the validation data. Other offline training algorithms such as

Broyden-Fletcher-Goldfarb-Shanno [49,50] and conjugate gradient [51,52] may be similarly efficient while their computational burden is prohibitive for online real-time training.

Results on a standard test scenario with the initial VRFT C-NN controller are shown in Figure 3. It is observed that the ORM tracking errors are bounded, since the VRFT controller is stabilizing (though not asymptotically) and validates the theoretical results of Theorem 1 and Corollary 1. Then it is an admissible controller for J_{MR}^{∞} in Equation (5) with $\gamma < 1$. The initial controller tuning using VRFT is attractive also because it has learned a feature matrix $\boldsymbol{\Phi}_a(\mathbf{x}_k^E)$, so from this point onwards, the designer can choose to perform either output weights adaptation or full weights adaptation.

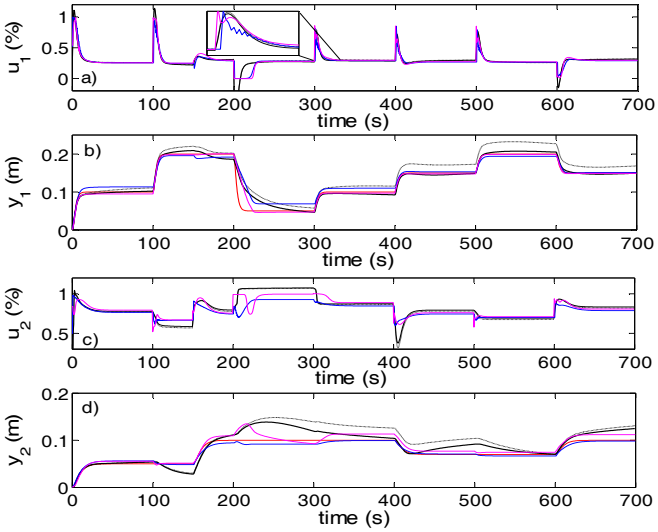

Figure 3. The initial VRFT controller (black dotted), the final adaptively learned controller (black solid), the BFQ controller (blue), the model-based BFQ controller (magenta) and the ORM outputs (red).

The Q-function estimate of the VRFT C-NN (i.e., the critic Q-NN) is next learned in a policy evaluation step in order to serve as a good initial estimate of the Q-function that is needed for the following AAC learning and also to fulfil the requirements of Lemma 1. This step is possible since the VRFT controller is admissible and, for properly selected learning rate, the weights of the Q-NN will converge. The critic Q-NN approximating the Q-function has similar architecture with the C-NN, of size 10–25–1 (eight states and two controls), with $n_{hc} = 25$. The critic Q-NN output weights are randomly drawn from a zero-mean normal distribution with variance $\sigma^2 = 90$ while the hidden layer weighs are uniformly randomly initialized in $[-1.5, 1.5]$. Setting $\gamma = 0.95$, the learning rates $\alpha_c = 0.01$ in (27) and $\alpha_a = 0$ in (28) (no controller tuning), all the Q-NN weights are updated using the gradient back-propagation in (27), by driving the MIMO control system with a sequence of uniformly random piecewise constant steps in $\mathbf{r}_k = [r_{k,1}\ r_{k,2}]^T \in [0.05, 0.25] \times [0.01, 0.2]$. This procedure also serves as a tuning step for α_c. With $r_{k,1}$ and $r_{k,2}$ lasting 20 s and 33 s, respectively, we ensure they do not switch simultaneously, to better reveal the coupling effects between the control channels. After 500 s, the critic weights stabilize, the output weights being shown in Figure 4 for 4500 s. This pre-tuned Q-NN will be used as initialization to the following case studies. After this intermediate tuning step of the Q-NN, the designer can choose for full weight adaptation of the Q-NN or only for output weights' adaptation of the Q-NN while the features matrix $\boldsymbol{\Phi}_c(\mathbf{x}_k^E, \mathbf{u}_k)$ is kept constant.

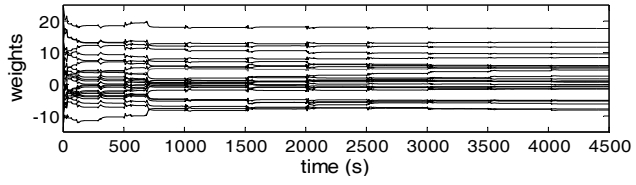

Figure 4. Q-NN output weights in Q-function learning of VRFT control.

The C-NN is now further tuned (in the architecture from Figure 1b to improve the ORM control performance. Setting $\alpha_a = 10^{-8}$ in (28) and $\alpha_c = 0.01$ in (27) (critic adaptation should generally be faster than actor adaptation), both the C-NN and Q-NN are adaptively trained online. Although carried out in an adaptive framework, the training unfolds on consecutive episodes where the feedback control system is driven by a sequence of random reference input steps for 700 s. The reference inputs are uniformly random piecewise constant steps in $\mathbf{r}_k = [r_{k,1}\ r_{k,2}]^T \in [0.05, 0.25] \times [0.01, 0.2]$, with $r_{k,1}$ and $r_{k,2}$ lasting 20 s and 33 s, respectively. Updates (27), (28) are skipped when either $r_{k,1}$ or $r_{k,2}$ switch, to preserve the Markov property of the extended model. The controller parameters at the end of an episode are the initial ones for the following episode, the Q-NN weights following the same transfer rule. To ensure enhanced exploration of the state-action space, the C-NN controller output is perturbed every third sample time with probing noise according to:

$$\mathbf{u}_k = \mathbf{C}(\mathbf{x}_k^E) + \begin{pmatrix} rand \\ rand \end{pmatrix} \cdot \Omega, \Omega = \begin{cases} 1 & \text{if } \mathrm{mod}(k,3) = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{35}$$

where *rand* is a normally distributed random number with zero-mean and variance $\sigma^2 = 3.56$ and $\mathrm{mod}(k,s)$ is the remainder after dividing k by s. This is in fact a form of ε^0-greedy exploration strategy useful to try many actions in the vicinity of the current state. A typical learning episode is shown in Figure 5. After each learning episode, the learning is stopped and the C-NN performance is measured on the standard test scenario from Figure 3 and the decrease of a finite-time version of the c.f. J_{MR}^{∞} from Equation (5), namely J_{MR}^{1400}, is aimed. This standard test scenario is *not* seen during training. The learning then resumes with the next episode. After maximum 30 learning episodes (meaning 21,000 s and 42,000 samples), the C-NN and Q-NN adaptations are stopped and the *learning trial* (comprising of learning episodes) converges under *Theorem 3*. The final adaptively learned C-NN and the initial VRFT controller are shown performing in Figure 3. The episodic learning allows us to test the improvement in control performance between episodes. Figure 6 illustrates the $J_{MR\_AC}^{1400}$ decrease over episodes of a convergent learning trial. Throughout each learning episode, under the AAC update laws, the control system preserves its stability, ensured by *Theorem 2*.

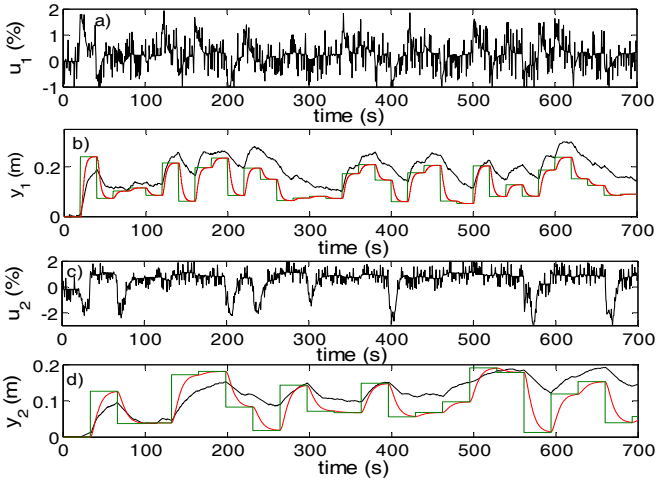

Figure 5. Typical training and learning episode. The perturbed control (black), the reference inputs to the CS (green) and the RM outputs (red).

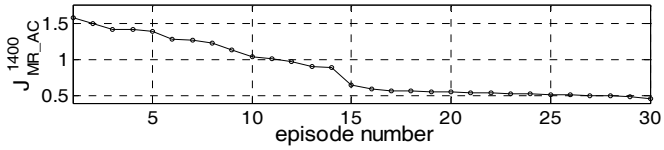

Figure 6. Evolution of $J_{MR\_AC}^{1400}$ with each episode for a typical trial of 30 episodes.

For comparisons, a model-free approximated batch-fitted Q-learning (BFQ) controller [53,54] is also proposed, using the same Q-NN and C-NN architectures with the same sizes. BFQ alternates offline training of the C-NN and the Q-NN, using 12,000 transition samples collected under the randomly perturbed model-free single-input single-output VRFT linear controllers $C_1(z) = (2.6092 + 0.1184z^{-1} - 2.3609z^{-2})/(1 - z^{-1})$ and $C_2(z) = (1.5735 + 0.2405z^{-1} - 1.3547z^{-2})/(1 - z^{-1})$, independently designed for the two tanks, respectively. BFQ implements a Value Iteration algorithm. The training settings assume that the weights of both NNs are initialized to uniform random numbers in $[-1.5, 1.5]$. Maximum 200 epochs are used for training with Levenberg-Marquardt on 80% effectively training data and 20% validation data. Early stopping is employed to prevent overfitting after six maximum increases of the mean sum of squared errors on the validation data. 200 iterations of BFQ take about 2 hours and the best controller is saved.

The value of J_{MR}^{1400} for the initial VRFT controller is $J_{MR\_VRFT}^{1400} = 1.58$, for the final adaptively learned controller is $J_{MR\_AAC}^{1400} = 0.45$ and for the BFQ controller is $J_{MR\_MBBFQ}^{1400} = 0.32$.

Additionally, a model-based approximated BFQ solution is also offered for comparisons. A first dimensionality reduction of the extended state space is performed by inspecting that, for both the ORMs, $x_{k+1,2}^m = x_{k,1}^m$ and $x_{k+1,4}^m = x_{k,3}^m$ from the state-space matrices in Equation (34) and, moreover, $y_{k,1}^m = \mathbf{C}_1\left[x_{k,1}^m \; x_{k,2}^m\right]^T \approx 2\mathbf{C}_1(1)x_{k,1}^m$, $y_{k,2}^m = \mathbf{C}_2\left[x_{k,3}^m \; x_{k,4}^m\right]^T \approx 2\mathbf{C}_2(1)x_{k,3}^m$. Then $x_{k,2}^m, x_{k,4}^m$ are considered approximate duplicates of $x_{k,1}^m, x_{k,3}^m$ and removed from the extended state vector, now defined as a reduced extended state vector $\mathbf{x}_k^{ER} = [x_{k,1}, x_{k,2}, x_{k,1}^m, x_{k,3}^m, r_{k,1}, \; r_{k,2}]^T \in \Re^6$ when used for feedback and controller learning. For $[0.05, 0.25] \times [0.03, 0.15] \times [0.5, 4.5] \times [1, 12] \times [0.05, 0.25] \times [0.03, 0.15]$ as the domain of $\mathbf{x}_k^{ER}$ and $[0, 1] \times [0.5, 1]$ the domain of $\mathbf{u}_k$, we generate a grid of $NP = 5 \times 5 \times 7 \times 7 \times 6 \times 6 \times 3 \times 3 = 396900$ linearly spaced points. 5 points for each of $x_{k,2}^m, x_{k,4}^m$ would have led to NP of almost 10 million. Let the discrete domains be denoted X_d^E and U_d, respectively. Domains of $x_{k,1}^m, x_{k,3}^m$ and $r_{k,1}, r_{k,2}$ are found by simulating the ORMs offline such that $y_{k,1}^m, y_{k,2}^m$ overlap the constrained domains of $x_{k,1} = y_{k,1}, x_{k,2} = y_{k,2}$. For a Q-NN of size 8–8–1, a C-NN of size 6–6–2 and $\gamma = 0.8$ found to ensure learning convergence, each iteration of the model-based BFQ trains both the Q-NN and the C-NN. For the Q-NN current iteration estimate (indexed by *iter*) denoted $\hat{Q}_{iter}\left(\mathbf{x}_k^{ER}, \mathbf{u}_k\right)$, the inputs patterns are $\left\{\left[\left(\mathbf{x}_k^{ER}\right)^T (\mathbf{u}_k)^T\right]^T\right\}$ and the target patterns are $\left\{V_{\mathrm{MR}}(\mathbf{x}_k^{ER}) + \gamma \min_{\mathbf{u} \in U_d} \hat{Q}_{iter-1}\left(\mathbf{F}(\mathbf{x}_k^{ER}, \mathbf{u}_k), \mathbf{u}\right)\right\}.$, evaluated for all the points in $X_d^E \times U_d$. For the current iteration C-NN $\mathbf{C}_{iter}\left(\mathbf{x}_k^{ER}\right)$, the input patterns are $\left\{\left(\mathbf{x}_k^{ER}\right)\right\}$ and the target patterns are $\left\{\mathbf{u}_k = \arg\min_{\mathbf{u} \in U_d} \hat{Q}_{iter}\left(\mathbf{x}_k^{ER}, \mathbf{u}\right)\right\}$. Note that evaluation of $\mathbf{F}(\mathbf{x}_k^{ER}, \mathbf{u}_k)$ to get $\mathbf{x}_{k+1}^{ER}$ uses the original extended state vector where $x_{k,2}^m, x_{k,4}^m$ are copies of generated $x_{k,1}^m, x_{k,3}^m$ and a piecewise constant reference input generative model is used where $r_{k+1,1} = r_{k,1}, r_{k+1,2} = r_{k,2}$. To keep the training computationally tractable and timely, only one third of uniformly sampled data points form the training set are used, differently for each of the Q-NN and the C-NN. The weights of both NNs are initialized to uniform random numbers in $[-1.5, 1.5]$. Maximum 200 epochs are used for training with Levenberg-Marquardt on 80% training data and 20% validation data. Early stopping is employed to prevent overfitting after six maximum increases of the mean sum of squared errors on the validation data. Just 14 iterations (taking about 20 min) of this approximate model-based BFQ produce the control results in Figure 3 (in magenta), with $J_{MR\_MFBFQ}^{1400} = 0.25$ naturally the smallest, with the best ORM tracking performance, since it uses the process model.

5.2. Statistical Investigations of the AAC Control Performance

Several thorough investigation case studies are considered, for which the initial C-NN tuned by VRFT and the initial Q-NN are the same. The investigation concerns full vs. partial tuning of the Q-NN and C-NN weights, while measuring the probing noise effect on the convergence. All statistics are measured on learning trials of maximum 50 episodes. The minimal and average $J_{MR\_AAC}^{1400}$ values on

100 trials are measured along with the success percentage of convergent learning trials. The average number of episodes until reaching the minimal $J^{1400}_{MR\_AAC}$ of a successful learning trial together with the standard deviation of the number of episodes in a successful learning trial, are both rendered in Table 1.

Table 1. AAC tuning statistics over maximum 50 episodes per trial.

Scenario		Avg. $J^{1400}_{MR\_AAC}$	Min. $J^{1400}_{MR\_AAC}$	Success Rate	Avg. Episodes	Std. Episodes
Full Tuning	**Perturbed**					
yes	yes	0.48	0.46	88%	25	4.9
yes	no	0.76	0.45	53%	43	6.3
no	yes	0.64	0.59	100%	16	3.2
no	no	0.526	0.50	100%	10	2.1

Case 1. Under learning rates $\alpha_c = 0.01$ in (27) and $\alpha_a = 10^{-8}$ in (28), with full adaptation of the Q-NN and of the C-NN, the learning process convergence starting from the initial C-NN tuned by VRFT and initial Q-NN is investigated. For a convergent learning trial, for the constant learning rates being used, the best performance did never drop below $J^{1400}_{MR\_AAC} = 0.46$, which is inferior to the BFQ performance, suggesting that the proposed adaptive learning strategy is prone to getting stuck in local minima under the adaptive gradient-based update rules. In fact, BFQ is generally advertised as being more data-efficient, although actor-critic learning architectures also allow alternative updates of the C-NN and Q-NN for improving data usage efficiency. The learning trials converges in about 88% of the cases, comparable with other perturbed AAC designs [55,56] given the wide operating range used for the controlled process.

Case 2. For full weights adaptation of both Q-NN and C-NN, without random perturbation of the control action, with $\alpha_c = 0.01$ in (27) and $\alpha_a = 10^{-7}$ in (28), the convergence rate drops to 53% and the best performance is $J^{1400}_{MR\_AAC} = 0.45$.

Case 3. In the case of output weights only tuning of both Q-NN and C-NN, with random perturbation of the control action, with $\alpha_c = 0.01$ in (27) and $\alpha_a = 10^{-6}$ in (28), 100% convergence rate was observed, but the performance never dropped below $J^{1400}_{MR\_AAC} = 0.59$.

Case 4. With output weights only tuning of both the Q-NN and C-NN, starting from the initialized C-NN tuned by VRFT and the initial C-NN, in the absence of random perturbation of the control action, with $\alpha_c = 0.01$ in (27) and $\alpha_a = 10^{-5}$ in (28), the AAC learning is 100% convergent in all trials but the performance never drops below $J^{1400}_{MR\_AAC} = 0.50$. For $\alpha_a = 10^{-6}$, the average number of episodes per trial increases only to 11.

The above four case studies are statistically characterized in Table 1. Concluding, full weight tuning of Q-NN and C-NN offers better performance (smaller $J^{1400}_{MR\_AAC}$) than when output weights only tuning is used. But full weight tuning lowers the convergence rate, as prone to stuck in local minima. In Case 1 vs. Case 2, the probing noise significantly improves the convergence, slightly improves the average $J^{1400}_{MR\_AAC}$ and decreases the average number of episodes per convergent trial. While full weights adaptation is more sensitive since even small corrections in the input-to-hidden layer weights may lead to learning divergence.

The output weights only tuning is more robust, with 100% convergence success rate to an improved solution, but with inferior achievable performance. The perturbing noise in this case worsens the average number of episodes per trial and the best achievable performance. Guaranteed convergence to an improved solution corresponding to a local minimum in Cases 3 and 4 is also caused by the good initial tuning offered by VRFT. Case 4 with output weights only tuning without probing noise offers the best compromise regarding convergence, performance and few episodes per trial (i.e., fewer transition samples until convergence).

The initial Q-function learning of the NN-VRFT controller is not necessary and learning convergence was obtained without this step also. For the selected critic learning rate, the weights converge fast enough, however, this step serves for tuning the critic learning rate and also for

initialization of the features matrix when output weights only adaptation is sought. This tuning step is achievable exactly because an initially stabilizing VRFT controller exists.

5.3. Comments on the AAC Learning Performance

The AAC learning data efficiency is clearly inferior to the BFQ strategy as a comparable model-free approach. The BFQ control is learned from scratch just from transition samples, whereas the proposed AAC controller learns from a NN-VRFT controller delivering an initial suboptimal ORM tracking solution. Both AAC and model-free BFQ are inferior to the model-based BFQ solution which exploits the process model knowledge.

AAC is a form of Action Dependent Heuristic Dynamic Programming which is also less data-efficient than other similar approaches such as Dual Heuristic Programming, where the learned co-state vector carries more information than the Q-function. On the other hand, AAC is less computationally demanding and requires less memory than Dual Heuristic Programming, BFQ and model-based BFQ, owing to AAC's adaptive implementation. However, AAC becomes competitive when used together with VRFT since the VRFT pre-tuning provides an initial controller close to the optimal one, which then can be fine-tuned using AAC. The initial NN VRFT controller ensures stabilized exploration over a wide operating range for ensuring ORM tracking in a wide range, which is equivalent to indirect feedback linearization. Then the combined VRFT-AAC design for ORM tracking is more attractive for practical data-driven applications [57,58].

6. Conclusions

A model-free combination of VRFT and AAC design approach was successfully validated to learn improved nonlinear state-feedback control for linear ORM tracking in a wide operating range. Learned controllers indirectly account for several nonlinearities such as actuator saturation plus dead-zone and output saturation, while they also show good decoupling abilities. AAC design shares similar conceptual framework with model-free techniques like Q-Learning, or SARSA, VRFT, Iterative Feedback Tuning and model-free Iterative Learning Control, by exploiting only the process model structure but not its parameters. The convergence of the proposed adaptive learning strategy relies on several key aspects: efficient exploration correlated with the size of the training dataset and with the process complexity, selected learning architecture and selection of the approximators with appropriate parameterizations. In a wider context, VRFT shows significant potential for obtaining close-to-optimal initially admissible controllers with respect to the ORM objective.

Future work targets the validation of the proposed tuning approach to other difficult nonlinear processes and its improvement using data-driven techniques.

Author Contributions: M.-B.R. developed the theoretical results, wrote the paper and performed the experiments; R.-E.P. revised the mathematical formulations, analyzed the algorithms and ensured the hardware and software support. All authors have read and approved the final paper.

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

Appendix A

Proof of Theorem 1.

Let $\mathbf{s}(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_{k-1}) - \mathbf{s}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{u}}_{k-1}) = \hat{\mathbf{y}}_k - \tilde{\mathbf{y}}_k = \Delta\mathbf{y}_k$, where $\hat{\mathbf{y}}_k = \mathbf{s}(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_{k-1} = C(\hat{\zeta}_{k-1}, \hat{\boldsymbol{\theta}}))$. In VRFT, it is also valid that $\tilde{\mathbf{y}}_k = \mathbf{s}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{u}}_{k-1}) = \mathbf{s}^m(\tilde{\mathbf{x}}_k^m, \tilde{\mathbf{r}}_{k-1})$ is the output of both the process and of the ORM driven by $\tilde{\mathbf{r}}_{k-1} = \mathbf{M}^{-1}(\tilde{\mathbf{y}}_k)$. By the mean value theorem, there is a $0 < \Gamma < 1$ making

$$\Delta\mathbf{y}_k = \frac{\partial \mathbf{s}(\mathbf{x}_k^\Gamma, \mathbf{u}_{k-1}^\Gamma)}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k - \tilde{\mathbf{x}}_k) + \frac{\partial \mathbf{s}(\mathbf{x}_k^\Gamma, \mathbf{u}_{k-1}^\Gamma)}{\partial \mathbf{u}_{k-1}}(\hat{\mathbf{u}}_{k-1} - \tilde{\mathbf{u}}_{k-1}), \tag{A1}$$

leading to

$$\|\Delta \mathbf{y}_k\| \leq B_{sx}\|\Delta \mathbf{x}_k\| + B_{su}\|\Delta \mathbf{u}_{k-1}\|, \tag{A2}$$

where $\mathbf{x}_k^{\Gamma} = \Gamma\hat{\mathbf{x}}_k + (1-\Gamma)\tilde{\mathbf{x}}_k$, $\mathbf{u}_{k-1}^{\Gamma} = \Gamma\hat{\mathbf{u}}_{k-1} + (1-\Gamma)\tilde{\mathbf{u}}_{k-1}$, and $\Delta\mathbf{x}_k = \hat{\mathbf{x}}_k - \tilde{\mathbf{x}}_k$, $\Delta\mathbf{u}_{k-1} = \hat{\mathbf{u}}_{k-1} - \tilde{\mathbf{u}}_{k-1}$.

Observe that (8) implies $\|\tilde{\mathbf{u}}_k - \mathbf{C}(\tilde{\mathbf{x}}_k^E, \hat{\boldsymbol{\theta}})\| < \varepsilon$, $\forall k = \overline{0, N-1}$. But $\Delta\mathbf{u}_k = \mathbf{C}(\hat{\mathbf{x}}_k^E, \hat{\boldsymbol{\theta}}) - \tilde{\mathbf{u}}_k = \mathbf{C}(\hat{\mathbf{x}}_k^E, \hat{\boldsymbol{\theta}}) - \mathbf{C}(\tilde{\mathbf{x}}_k^E, \hat{\boldsymbol{\theta}}) + \mathbf{C}(\tilde{\mathbf{x}}_k^E, \hat{\boldsymbol{\theta}}) - \tilde{\mathbf{u}}_k$ and by the MVT there is a $0 < \Gamma < 1$ such that

$$\Delta\mathbf{u}_k = \frac{\partial \mathbf{C}(\mathbf{x}_k^{E\Gamma}, \boldsymbol{\theta})}{\partial \mathbf{x}_k^E}(\hat{\mathbf{x}}_k^E - \tilde{\mathbf{x}}_k^E) + \mathbf{C}(\tilde{\mathbf{x}}_k^E, \hat{\boldsymbol{\theta}}) - \tilde{\mathbf{u}}_k, \tag{A3}$$

with $\mathbf{x}_k^{E\Gamma} = \Gamma\hat{\mathbf{x}}_k^E + (1-\Gamma)\tilde{\mathbf{x}}_k^E$, leading to

$$\|\Delta\mathbf{u}_k\| \leq B_{cx}\|\Delta\mathbf{x}_k^E\| + \varepsilon, \tag{A4}$$

for $\Delta\mathbf{x}_k^E = \hat{\mathbf{x}}_k^E - \tilde{\mathbf{x}}_k^E$. But $\Delta\mathbf{x}_k^E = \hat{\mathbf{x}}_k^E - \tilde{\mathbf{x}}_k^E = [\left(\hat{\mathbf{x}}_k - \tilde{\mathbf{x}}_k\right)^T \; 0^T \; 0^T]^T$ resulting in $\|\Delta\mathbf{x}_k^E\| = \|\Delta\mathbf{x}_k\|$ which transforms (A4) into

$$\|\Delta\mathbf{u}_k\| \leq B_{cx}\|\Delta\mathbf{x}_k\| + \varepsilon. \tag{A5}$$

Next, by the mean value theorem, there is a $0 < \Gamma < 1$ ensuring that

$$\Delta\mathbf{x}_k = \hat{\mathbf{x}}_k - \tilde{\mathbf{x}}_k = (\mathbf{s}_x)^{-1}(\hat{\mathbf{y}}_k) - (\mathbf{s}_x)^{-1}(\tilde{\mathbf{y}}_k) = \frac{\partial(\mathbf{s}_x)^{-1}(\mathbf{y}_k^{\Gamma})}{\partial\mathbf{y}_k}(\hat{\mathbf{y}}_k - \tilde{\mathbf{y}}_k), \mathbf{y}_k^{\Gamma} = \Gamma\hat{\mathbf{y}}_k + (1-\Gamma)\tilde{\mathbf{y}}_k.. \tag{A6}$$

It then follows that

$$\|\Delta\mathbf{x}_k\| \leq B_{sy}\|\Delta\mathbf{y}_k\|. \tag{A7}$$

Using (A5) and (A7) in (A2) it results

$$\|\Delta\mathbf{y}_k\| \leq B_{sx}B_{sy}\|\Delta\mathbf{y}_k\| + B_{su}(B_{cx}\|\Delta\mathbf{x}_{k-1}\| + \varepsilon) \leq B_{sx}B_{sy}\|\Delta\mathbf{y}_k\| + B_{su}B_{cx}B_{sy}\|\Delta\mathbf{y}_{k-1}\| + B_{su}\varepsilon, \tag{A8}$$

which is equivalent to

$$\|\Delta\mathbf{y}_k\| \leq \underbrace{\frac{B_{su}B_{cx}B_{sy}}{1 - B_{sx}B_{sy}}}_{B_1}\|\Delta\mathbf{y}_{k-1}\| + \underbrace{\frac{B_{su}}{1 - B_{sx}B_{sy}}}_{B_2}\varepsilon. \tag{A9}$$

One can write that

$$J_{MR}^N(\hat{\boldsymbol{\theta}}) = \sum_{k=1}^{N}\left\|\Delta y_k\right\|^2 \leq \left(B_2\sum_{k=1}^{N}\sum_{j=0}^{k-1}B_1^j\right)^2\varepsilon^2 = B\varepsilon^2. \tag{A10}$$

which is the conclusion (10), and the proof of *Theorem 1* is completed.

Appendix B

Proof of Theorem 3.

Let $\delta_k = U^{k-1} + \left(\mathbf{W}_c^{k-1}\right)^T\left(\gamma\boldsymbol{\Phi}_c^k - \boldsymbol{\Phi}_c^{k-1}\right)$ be expressed further as $\delta_k = \underbrace{U^{k-1} + (\mathbf{W}_c^*)^T\left(\gamma\boldsymbol{\Phi}_c^k - \boldsymbol{\Phi}_c^{k-1}\right)}_{E1} +$

$\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T\left(\gamma\boldsymbol{\Phi}_c^k - \boldsymbol{\Phi}_c^{k-1}\right)$. Let $\boldsymbol{\Phi}_c^k = \boldsymbol{\Phi}_c^{k-1} + \Delta\boldsymbol{\Phi}_c^k$ make $\delta_k = E1 + (\gamma-1)\zeta_c^{k-1} + \gamma\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T\Delta\boldsymbol{\Phi}_c^{k-1}$.

Define the Lyapunov function

$$L^{k-1} = \frac{1}{\alpha_c} tr\left\{\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T \tilde{\mathbf{W}}_c^{k-1}\right\} + \frac{1}{\alpha_a} tr\left\{\left(\tilde{\mathbf{W}}_a^{k-1}\right)^T \tilde{\mathbf{W}}_a^{k-1}\right\} = L_1 + L_2. \tag{A11}$$

($tr\{.\}$ meaning the matrix trace operator) leading to the first order differences in L_1 and L_2

$$\begin{aligned} \Delta L_1 &= \tfrac{1}{\alpha_c}[tr\left\{\left(\tilde{\mathbf{W}}_c^{k}\right)^T \tilde{\mathbf{W}}_c^{k}\right\} - tr\left\{\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T \tilde{\mathbf{W}}_c^{k-1}\right\}], \\ \Delta L_2 &= \tfrac{1}{\alpha_a}[tr\left\{\left(\tilde{\mathbf{W}}_a^{k}\right)^T \tilde{\mathbf{W}}_a^{k}\right\} - tr\left\{\left(\tilde{\mathbf{W}}_a^{k-1}\right)^T \tilde{\mathbf{W}}_a^{k-1}\right\}]. \end{aligned} \tag{A12}$$

Using estimation error dynamics (32), $\Delta L_1, \Delta L_2$ are refined further. Let

$$\begin{aligned} &\Delta L_1 = \tfrac{1}{\alpha_c}[tr\{2\alpha_c\delta_k \underbrace{\left(\boldsymbol{\Phi}_c^{k-1}\right)^T \tilde{\mathbf{W}}_c^{k}}_{\zeta_c^{k-1}} + \alpha_c\delta_k^2 \left(\boldsymbol{\Phi}_c^{k-1}\right)^T \boldsymbol{\Phi}_c^{k-1}] = 2\alpha_c\left(E1 + (\gamma-1)\zeta_c^{k-1} + \gamma\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T \Delta\boldsymbol{\Phi}_c^{k-1}\right)\zeta_c^{k-1} + \\ &+\alpha_c\left(E1 + (\gamma-1)\zeta_c^{k-1} + \gamma\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T \Delta\boldsymbol{\Phi}_c^{k-1}\right)^2 \|\boldsymbol{\Phi}_c^{k-1}\|^2 \le 2\alpha_c(\gamma-1)\left(\zeta_c^{k-1}\right)^2 + 2\alpha_c\left(E1 + \gamma\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T \Delta\boldsymbol{\Phi}_c^{k-1}\right)\zeta_c^{k-1} \\ &+\alpha_c\overline{\varphi}_c\left(2(\gamma-1)^2\left(\zeta_c^{k-1}\right)^2 + 2\left(E1 + \gamma\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T \Delta\boldsymbol{\Phi}_c^{k-1}\right)^2\right) = 2\alpha_c(\gamma-1)\left(\zeta_c^{k-1}\right)^2 - \underbrace{\left(\zeta_c^{k-1} - \alpha_c\left(E1 + \gamma\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T \Delta\boldsymbol{\Phi}_c^{k-1}\right)\right)^2}_{E2} \\ &+2\alpha_c\overline{\varphi}_c(\gamma-1)^2\left(\zeta_c^{k-1}\right)^2 + 2\alpha_c\overline{\varphi}_c\left(E1 + \gamma\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T \Delta\boldsymbol{\Phi}_c^{k-1}\right)^2 + \left(\zeta_c^{k-1}\right)^2 + \alpha_c^2\left(E1 + \gamma\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T \Delta\boldsymbol{\Phi}_c^{k-1}\right)^2 = \\ &== [1 + 2\alpha_c(\gamma-1) + 2\alpha_c\overline{\varphi}_c(\gamma-1)^2]\left(\zeta_c^{k-1}\right)^2 - E2 + \underbrace{(2\alpha_c\overline{\varphi}_c + \alpha_c^2)\left(E1 + \gamma\left(\tilde{\mathbf{W}}_c^{k-1}\right)^T \Delta\boldsymbol{\Phi}_c^{k-1}\right)^2}_{E3}. \end{aligned} \tag{A13}$$

One can show that the coefficient of $(\zeta_e{}^{k-1})$ in the first term, is a function of γ, i.e., $\widetilde{f}(\gamma) = 1 + 2\alpha_c(\gamma-1) + 2\alpha_c\overline{\varphi}_c(\gamma-1)^2$, which will have two real roots for $\alpha_c > 2\overline{\varphi}_c$. Let these roots be $\gamma_1 < \gamma_2$. By ensuring $\gamma_1 < 0, \gamma_2 \ge 1$ then for any γ such that $\gamma_1 < 0 < \gamma \le 1 < \gamma_2$, it means that $\widetilde{f}(\gamma) < 0$. A sufficient condition to ensure $\widetilde{f}(\gamma) < 0$ for all $0 < \gamma \le 1$, is to select $\alpha_c > \max\left\{2\overline{\varphi}_c, 4\overline{\varphi}_c^2 - 2\overline{\varphi}_c + 1\right\} = 4\overline{\varphi}_c^2 - 2\overline{\varphi}_c + 1$ for all $\overline{\varphi}_c > 1$.

Note further that the term E3 in (A13) is positive. Let ΔL_2 be further expressed as

$$\begin{aligned} &\Delta L_2 = \tfrac{1}{\alpha_a}[tr\left\{\left(\tilde{\mathbf{W}}_a^{k}\right)^T \tilde{\mathbf{W}}_a^{k}\right\} - tr\left\{\left(\tilde{\mathbf{W}}_a^{k-1}\right)^T \tilde{\mathbf{W}}_a^{k-1}\right\}] = \\ &= \tfrac{1}{\alpha_a}\left(\|\tilde{\mathbf{W}}_a^{k-1} - \alpha_a\boldsymbol{\Phi}_a^{k-1}\left(\mathbf{W}_c^{k-1}\right)^T \boldsymbol{\Phi}_{c,u}^{k-1}\|^2 - \|\tilde{\mathbf{W}}_a^{k-1}\|^2\right) \le \\ &\le \tfrac{1}{\alpha_a}\left(\left(\|\tilde{\mathbf{W}}_a^{k-1}\| + \|\alpha_a\boldsymbol{\Phi}_a^{k-1}\left(\mathbf{W}_c^{k-1}\right)^T \boldsymbol{\Phi}_{c,u}^{k-1}\|\right)^2 - \|\tilde{\mathbf{W}}_a^{k-1}\|^2\right) = \\ &= 2\alpha_a\|\tilde{\mathbf{W}}_a^{k-1}\|\|\boldsymbol{\Phi}_a^{k-1}\left(\mathbf{W}_c^{k-1}\right)^T \boldsymbol{\Phi}_{c,u}^{k-1}\| + \alpha_a\|\boldsymbol{\Phi}_a^{k-1}\left(\mathbf{W}_c^{k-1}\right)^T \boldsymbol{\Phi}_{c,u}^{k-1}\|^2 \le \\ &\le 2\alpha_a\overline{\varphi}_a\overline{\varphi}_{c,u}\|\tilde{\mathbf{W}}_a^{k-1}\|\|\mathbf{W}_c^{k-1}\| + \alpha_a\overline{\varphi}_a^2\overline{\varphi}_{c,u}^2\|\mathbf{W}_c^{k-1}\|^2 = E4 > 0. \end{aligned} \tag{A14}$$

Eventually,

$$\Delta L = \Delta L_1 + \Delta L_2 = [1 + 2\alpha_c(\gamma-1) + 2\alpha_c\overline{\varphi}_c(\gamma-1)^2]\left(\zeta_c^{k-1}\right)^2 - E2 + E3 + E4, \tag{A15}$$

can be shown negative if

$$[1 + 2\alpha_c(\gamma - 1) + 2\alpha_c\overline{\varphi}_c(\gamma - 1)^2]\left(\zeta_c^{k-1}\right)^2 + E3 + E4 < 0 \tag{A16}$$

holds, where, considering the first term as negative for $\alpha_c > 4\overline{\varphi}_c^2 - 2\overline{\varphi}_c + 1$ while $E3 > 0$, $E4 > 0$, it follows that it suffices to have

$$\left(\zeta_c^{k-1}\right)^2 > \frac{E3 + E4}{|1 + 2\alpha_c(\gamma - 1) + 2\alpha_c\overline{\varphi}_c(\gamma - 1)^2|} \tag{A17}$$

in order to make ΔL negative definite. Then by Lyapunov extension theorem [59], the AAC learning is stable and the NN estimation errors are uniformly ultimately bounded, concluding the proof.

References

1. Hou, Z.-S.; Wang, Z. From model-based control to data-driven control: Survey, classification and perspective. *Inf. Sci.* **2013**, *235*, 3–35. [CrossRef]
2. Fliess, M.; Join, C. Model-free control. *Int. J. Control* **2013**, *86*, 2228–2252. [CrossRef]
3. Hou, Z.-S.; Jin, S. Data-driven model-free adaptive control for a class of MIMO nonlinear discrete-time systems. *IEEE Trans. Neural Netw.* **2011**, *22*, 2173–2188. [PubMed]
4. 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]
5. Hjalmarsson, H. Iterative feedback tuning—An overview. *Int. J. Adapt. Control Signal Process.* **2002**, *16*, 373–395. [CrossRef]
6. Spall, J.C.; Cristion, J.A. Model-free control of nonlinear stochastic systems with discrete-time measurements. *IEEE Trans. Autom. Control* **1998**, *43*, 1198–1210. [CrossRef]
7. Butcher, M.; Karimi, A.; Longchamp, R. Iterative learning control based on stochastic approximation. In Proceedings of the 17th IFAC World Congress, Seoul, Korea, 6–11 July 2008; pp. 1478–1483.
8. Radac, M.-B.; Precup, R.-E.; Petriu, E.M. Optimal behaviour prediction using a primitive-based data-driven model-free iterative learning control approach. *Comp. Ind.* **2015**, *74*, 95–109. [CrossRef]
9. Li, Y.; Hou, Z.; Feng, Y.; Chi, R. Data-driven approximate value iteration with optimality error bound analysis. *Automatica* **2017**, *78*, 79–87. [CrossRef]
10. Radac, M.-B.; Precup, R.-E.; Petriu, E.M.; Preitl, S. Iterative data-driven tuning of controllers for nonlinear systems with constraints. *IEEE Trans. Ind. Electron.* **2014**, *61*, 6360–6368. [CrossRef]
11. Pang, Z.-H.; Liu, G.-P.; Zhou, D.; Sun, D. Data-based predictive control for networked nonlinear systems with network-induced delay and packet dropout. *IEEE Trans. Ind. Electron.* **2016**, *63*, 1249–1257. [CrossRef]
12. 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]
13. Qiu, J.; Wang, T.; Yin, S.; Gao, H. Data-Based Optimal Control for Networked Double-Layer Industrial Processes. *IEEE Trans. Ind. Electron.* **2017**, *64*, 4179–4186. [CrossRef]
14. 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. Syst.* **2017**, *49*, 687–696. [CrossRef]
15. 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]
16. Chi, R.; Huang, B.; Hou, Z.; Jin, S. Data-driven high-order terminal iterative learning control with a faster convergence speed. *Int. J. Robust Nonlinear Control* **2018**, *28*, 103–119. [CrossRef]
17. Jeng, J.-C.; Ge, G.-P. Data-based approach for feedback–feedforward controller design using closed-loop plant data. *ISA Trans.* **2018**, *80*, 244–256. [CrossRef] [PubMed]
18. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998.
19. Werbos, P.J. Approximate dynamic programming for real-time control and neural modeling. In *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*; White, D.A., Sofge, D.A., Eds.; Van Nostrand Reinhold: New York, NY, USA, 1992; pp. 493–525.

20. Bertsekas, D.P.; Tsitsiklis, J.N. *Neuro-Dynamic Programming*; Athena Scientific: Belmont, MA, USA, 1996.
21. Watkins, C.; Dayan, P. Q-learning. *Mach. Learn.* **1991**, *8*, 279–292. [CrossRef]
22. Wang, F.-Y.; Zhang, H.; Liu, D. Adaptive dynamic programming: An introduction. *IEEE Comput. Intell. Mag.* **2009**, *4*, 39–47. [CrossRef]
23. 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.
24. Prokhorov, D.V.; Wunsch, D.C. Adaptive critic designs. *IEEE Trans. Neural Netw.* **1997**, *8*, 997–1007. [CrossRef]
25. Wei, Q.; Lewis, F.; Liu, D.; Song, R.; Lin, H. Discrete-time local value iteration adaptive dynamic programming: Convergence analysis. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *48*, 875–891. [CrossRef]
26. Heydari, A. Revisiting approximate dynamic programming and its convergence. *IEEE Trans. Cybern.* **2014**, *44*, 2733–2743. [CrossRef] [PubMed]
27. Mu, C.; Ni, Z.; Sun, C.; He, H. Data-driven tracking control with adaptive dynamic programming for a class of continuous-time nonlinear systems. *IEEE Trans. Cybern.* **2017**, *47*, 1460–1470. [CrossRef]
28. Venayagamoorthy, G.K.; Harley, R.G.; Wunsch, D.C. Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator. *IEEE Trans. Neural Netw.* **2002**, *13*, 764–773. [CrossRef]
29. Ni, Z.; He, H.; Zhong, Z.; Prokhorov, D.V. Model-free dual heuristic dynamic programming. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 1834–1839. [CrossRef] [PubMed]
30. Heydari, A. Optimal triggering of networked control systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 3011–3021. [CrossRef] [PubMed]
31. Zhang, K.; Zhang, H.; Gao, Z.; Su, H. Online adaptive policy iteration based fault-tolerant control algorithm for continuous-time nonlinear tracking systems with actuator failures. *J. Frankl. Inst.* **2018**, *355*, 6947–6968. [CrossRef]
32. Mnih, V.; Kavukcouglu, 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] [PubMed]
33. Lewis, F.L.; Vamvoudakis, K.G. Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. *IEEE Trans. Syst. Man Cybern. B Cybern.* **2011**, *41*, 14–25. [CrossRef]
34. Wang, Z.; Liu, D. Data-based controllability and observability analysis of linear discrete-time systems. *IEEE Trans. Neural Netw.* **2011**, *22*, 2388–2392. [CrossRef]
35. Ruelens, F.; Claessens, B.J.; Vandael, S.; de Schutter, B.; Babuška, R.; Belmans, R. Residential demand response of thermostatically controlled loads using batch reinforcement learning. *IEEE Trans. Smart Grid* **2017**, *8*, 2149–2159. [CrossRef]
36. Liu, D.; Javaherian, H.; Kovalenko, O.; Huang, T. Adaptive critic learning techniques for engine torque and air-fuel ratio control. *IEEE Trans. Syst. Man Cybern. B Cybern.* **2008**, *38*, 988–993.
37. 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]
38. Campestrini, L.; Eckhard, D.; Bazanella, A.S.; Gevers, M. Data-driven model reference control design by prediction error identification. *J. Frankl. Inst.* **2017**, *354*, 2628–2647. [CrossRef]
39. Campestrini, L.; Eckhard, D.; Gevers, M.; Bazanella, A. Virtual reference feedback tuning for non-minimum phase plants. *Automatica* **2011**, *47*, 1778–1784. [CrossRef]
40. Formentin, S.; Savaresi, S.M.; Del Re, L. Non-iterative direct data-driven controller tuning for multivariable systems: Theory and application. *IET Control Theory Appl.* **2012**, *6*, 1250–1257. [CrossRef]
41. Yan, P.; Liu, D.; Wang, D.; Ma, H. Data-driven controller design for general MIMO nonlinear systems via virtual reference feedback tuning and neural networks. *Neurocomputing* **2016**, *171*, 815–825. [CrossRef]
42. Campi, M.C.; Savaresi, S.M. Direct nonlinear control design: The virtual reference feedback tuning (VRFT) approach. *IEEE Trans. Autom. Control* **2006**, *51*, 14–27. [CrossRef]
43. Esparza, A.; Sala, A.; Albertos, P. Neural networks in virtual reference tuning. *Eng. Appl. Artif. Intell.* **2011**, *24*, 983–995. [CrossRef]
44. 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]

45. 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]
46. Busoniu, L.; Ernst, D.; de Schutter, B.; Babuska, R. Approximate reinforcement learning: An overview. In Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, Paris, France, 11–15 April 2011; pp. 1–8.
47. Inteco Ltd. *Multitank System, User's Manual*; Inteco Ltd.: Krakow, Poland, 2007.
48. Hagan, M.T.; Menhaj, M.B. Training feed-forward networks with the Marquardt algorithm. *IEEE Trans. Neural Netw.* **1994**, *5*, 989–993. [CrossRef]
49. Liu, Q.; Liu, J.; Sang, R.; Li, J.; Zhang, T.; Zhang, Q. Fast neural network training on FPGA using quasi-Newton optimisation method. *IEEE Trans. Very Large Scale Integr. Syst.* **2018**, *26*, 1575–1579. [CrossRef]
50. Livieris, I.E. Improving the classification efficiency of an ANN utilizing a new training methodology. *Informatics* **2019**, *6*, 1. [CrossRef]
51. Møller, M.F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw.* **1993**, *6*, 525–533. [CrossRef]
52. Livieris, I.E.; Pintelas, P. A new conjugate gradient algorithm for training neural networks based on a modified secant equation. *Appl. Math. Comput.* **2013**, *221*, 491–502. [CrossRef]
53. 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]
54. 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] [PubMed]
55. 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]
56. Zhao, D.; Wang, B.; Liu, D. A supervised actor-critic approach for adaptive cruise control. *Soft Comput.* **2013**, *17*, 2089–2099. [CrossRef]
57. Radac, M.-B.; Precup, R.-E.; Petriu, E.M.; Preitl, S.; Dragos, C.-A. Data-driven reference trajectory tracking algorithm and experimental validation. *IEEE Trans. Ind. Inf.* **2013**, *9*, 2327–2336. [CrossRef]
58. 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]
59. Yang, Q.; Jagannathan, S. Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators. *IEEE Trans. Syst. Man Cybern. B Cybern.* **2012**, *42*, 377–390. [CrossRef] [PubMed]