

Article

Soft Real-Time Asynchronous Online Learning from Input–Output Data for UAV Model Reference Control Under Uncertain Dynamics and Faulty Actuation

Mircea-Bogdan Radac

Department of Automation and Applied Informatics, Politehnica University of Timisoara,
300223 Timisoara, Romania; mircea.radac@upt.ro

Highlights

What are the main findings?

- Learning control for high-dimensional unknown multivariable systems is based on input–output data and takes place in a transformed virtual state-space representation that is completely observable, thus overcoming the dimensionality curse.
- Asynchronous learning exploiting multi-tasking, and smart actor–critic parameterization is a key aspect in implementing the adaptive learning control in fast real-time systems that are subject to parametric variations and actuator faults.

What are the implications of the main findings?

- The proposed model reference tracking control approach learns decoupled control starting from well-known and straightforward performance indicators like overshoot and rise time.
- Online adaptive asynchronous learning under permanent stabilizing control is ensured, while its effectiveness can be extended to more complex systems even beyond the complexity of quadrotor UAV double-integrator coupled three-channel attitude control.

Abstract

An online off-policy asynchronous real-time model reference tracking control (OOART-MRTC) algorithm is proposed and validated for unmanned aerial vehicles (UAVs) characterized by faulty actuation and parametric uncertainty. The optimal control problem is posed based on approximate dynamic programming (ADP) and reinforcement learning (RL) theory, using a virtual state-space representation constructed exclusively on input–output true system data, which exploits the observability theory. OOART-MRTC learns control by interacting with the system, starting from an initial stabilizing controller derived from an approximate uncertain model. Learning convergence and stability under the proposed adaptive behavior are analyzed. Since the learning iterations cannot update within a sampling period, an asynchronous mechanism is proposed for updating the controller parameters, leveraging real-time control and multi-tasking. The complexity associated with the resulting high-dimensional system is solved by efficient linear parameterization and validated on a realistic case study where three coupled double integrators describe the UAV attitude control.

Academic Editor: Abdessattar Abdelkefi

Received: 5 January 2026

Revised: 5 February 2026

Accepted: 13 February 2026

Published: 15 February 2026

Copyright: © 2026 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license

Keywords: online learning; asynchronous; UAV; model reference control; approximate dynamic programming; reinforcement learning

1. Introduction

In the control system domain, one of the most influential methods is the adaptive (or approximate) dynamic programming (ADP) (sometimes coined as neural dynamic programming; see [1,2], whose goal is learning optimal control under system model uncertainty [3–11]. Several mainstream ADP controls have evolved, like dual heuristic programming (DHP), heuristic dynamic programming (HDP), and global DHP (GDHP), to cope with learning either in model-free mode or in model-based mode; see, e.g., the works of [12–16]. The model-free setting leads to the family of action-dependent HDP (ADHDP) methods, prominently known as Q-learning in the reinforcement learning domain, which is more aligned to Artificial Intelligence.

In ADP, the rationale is that the Hamilton–Jacobi–Bellman (HJB) equation is hard or impossible to solve analytically for general nonlinear systems. Then, ADP proposes to use a so-called actor–critic architecture that employs a) an actor that embodies the controller (sometimes called policy) and b) a critic to approximate the state value function “V” or the more extended state action value function “Q”. The critic’s role is to appreciate the actor’s performance, and the family of Policy Iteration (PoIt) and Value Iteration (VI) algorithms was developed to solve the HJB equation forward in time. For the general unknown nonlinear systems, the actor and the critic are represented by neural network (NN) function approximators. A third algorithm called Policy Gradient (PG) can also solve the HJB, but commonly it performs backward in time in an iterative fashion and it is also less efficient than PoIt or VI. Both PoIt and VI come in several implementation flavors, like on- or off-policy, on- or off-line, adaptive or not, and incremental or using experience replay, based on buffers of transition samples. These transition samples model the underlying system’s state transition, ideally captured by a (non)linear state-space representation which in ADP is a deterministic expression of the more general Markov Decision Process (MDP).

Two factors are essential to the underlying system representation whose control is sought via ADP methods. The first and most important is the system state (sometimes called observation) which is unmeasurable for unknown system dynamics. The observability theory deeply rooted in classic control points out the solution to arriving at virtual, equivalent state-space representations, where the hidden true state is aliased by a sequence of input–output data samples that are readily measured from the system. This approach preserves the system input and output, as it only changes the internal state representation that makes the equivalent (or virtual) model a fully observable one, thus alleviating the environment partial observability issue. The second factor is the system input that serves a two-fold objective: control action and action–state space exploration. The action–state exploration is a general requirement for learning convergence of the ADP methods and it is even more critical with nonlinear systems; as with linear ones, exploration of a narrow action–state region can well generalize the optimal learned control solution towards global certificates.

The ADP methods naturally lend themselves to systems with modeling uncertainty and incomplete state measurements. Unmanned aerial vehicle (UAV) systems fall under this category due to their nonlinearity and high complexity and dimensionality. UAV control using ADP or RL methods has been tackled before either in linear or nonlinear settings; however, the specifics of uncertain models, real-time online learning capability, incomplete state measurement and stability preservation along the learning process have

not been suitably approached before. In addition, the model reference control approach for UAVs, or for any kind of systems, is an essential instrument to synthesize the control performance specifications at a requirements level. It is therefore of interest to integrate such a feature in the aforementioned setting.

Early results on a 6DoF quadrotor UAV control with ADP were formulated in [17]. The learning takes place for a nonlinear model and uses NNs for the actor–critic. Although adaptive, the formulation misses the model reference setting and strong convergence and stability results. The work from [18] studied attitude and longitudinal motion of a quadrotor UAV based on a linearized model, in an adaptive control setting but without an actor–critic. Instead, classical LQR is combined with model reference adaptive control for a partially known model with uncertainty, and stability is assessed. The work’s innovation also includes faults on the control inputs actuation; however, all states are measured and the model-free design is not fully approached under partial observability. Stronger experimental evidence validated the approach. In another work, ref. [19] contributes a computationally-efficient method to store a Q-function generalization, a continuous action selection based on local Q-function approximation, and a combination of model identification and online learning for inner-loop control of a UAV system. A model reduction is performed to facilitate learning and overcome the curse of dimensionality. A reference tracking problem formulation was proposed; however, the model reference and convergence and stability analysis were not included. In [20], a novel RL for a thrust-vectoring controlled quadcopter was proposed, based on Proximal Policy Optimization (PPO). The reward was crafted to steer the flying robot towards its specified waypoint starting from its current location. The nonlinear dynamics complexity was used in the actor–critic; nevertheless, the model reference tracking, partial observability, and convergence and stability analysis were not research topics.

In [21], a quadratic programming neural dynamic controller is applied to UAVs for tracking control purposes. A complex numerical model was employed with a more conventional model-based design under uncertainty and validated in a thorough case study consisting of many flight modes. The observability issues and complete model-free design were not tackled. A deep RL (DRL) adaptive controller for an aerial robot was proposed in [22] and shown to be superior to traditional PID controllers; however, simulated environments do not pose the challenges associated with real-time interaction control. The authors of [23] relayed an improved tabular Q-learning for robust attitude control of a fixed-wing UAV, when affected by atmospheric disturbances, sensor noise, actuator faults, and other model uncertainty. Although the learning is adaptive, convergence and stability were not explicitly analyzed. In [24], a deep RL intelligent nonlinear controller for an experimental gliding UAV was validated using a modified DDPG algorithm. A sophisticated reward design combined with many simulation episodes leads to an optimal solution deemed superior to LQR and PPO. The results are missing convergence and stability analysis, although applied to a nonlinear problem model. A fully integrated RL approach combined with adaptive control was proposed in [25] for uncertain system control with reference model, in a real-time setting with stability analysis and actuator fault. However, full state measurement was considered even when the results were validated in a real-time UAV, and no partial environment observability was tackled. The authors of [26] proposed a robust optimal safe and stability certified control validated for UAV reference trajectory tracking under parametric uncertainty, under PPO RL. The real-time mechanisms are not analyzed, and the partial observability is not approached in this work. Ref. [27] proposed an optimal gain self-tuning approach for altitude, attitude, and longitudinal motion controllers of a 6DoF nonlinear drone, using the DRL mechanism combined with a custom reward. Theoretical stability is certified; however, the real-time interaction mechanism and the partial state observation are not considered, either.

One of the greater challenges with online real-time ADP and RL control concerns the interleaving of the update equations, data collection, and fixed sampling period control execution. In fact, this has been the major bottleneck against a more widespread adoption of the ADP methods. Even in the linear system case, the actor and critic updates may require working with many transition samples of high dimension to improve method conditioning, which makes the learning steps impossible to update within the system's sampling period. This forces the actor and critic structure update to be performed in asynchronous mode [28], using specialized software mechanisms. The issue becomes even more critical with generalized NN-based actor–critic approaches when real-time interaction is required [29–33]. The problem size becomes larger with the state-action size, which itself grows when using a virtual state representation comprised of present and past IO system samples. The environment's complete observability issue under real-time control, about how to build equivalent fully observable virtual states, has not been tackled systematically [34]. Another issue concerns the system stability preservation along the adaptive updates, which is necessary for safety insurance [35]. Addressing these two challenges may help with the adoption of ADP and RL methods in real-world systems and this idea drives the current contribution. Finally, in a model reference control context, the trajectory feasibility must account for the underlying dynamics, which certainly requires some qualitative system insight and consideration for several classic control rules.

The contributions in this work are enumerated as follows:

- A theory for learning optimal control dealing with uncertain model dynamics and actuation faults, in a model reference tracking control approach, with equivalent virtual state-space representation where the virtual state is encoded as a moving window of input–output system samples.
- The proposed reference model is different from those existing in the literature, as it originates from easily interpretable performance indicators like rise time and overshoot, which are widely adopted in the community.
- Learning convergence of the proposed online off-policy asynchronous real-time model reference tracking control (OOART-MRTC) algorithm, with stability certification alongside learning, when starting from initially stabilizing control.
- Validation of the OOART-MRTC on attitude control for a quadrotor UAV serves as a use case study, to show that a complex double-integrator with coupled channels transforms to a large virtual state-space representation consisting of tens of variables. Even under such dimensionality curse, the learning can still be effective, fast, convergent and stabilizing in online asynchronous mode.

Notation used: R^m is the set of real-valued m -dimensional vectors which are column vectors by default. Upper-right “ T ” indicates matrix transpose. The norm operator $\|\cdot\|$ is implicitly Euclidean, over some vector or induced over some matrix. I_n is the identity matrix with size $n \times n$.

The paper is organized as follows. The second section presents the general controlled system dynamics converted to a virtual state-space representation, the tracked reference model, the reference model driving input, and the resulting extended state space system. Once set up, the model reference tracking optimal control problem is formed and the parameterized ADP-based solution is derived under the OOART-MRTC algorithm. It is still in this section that the learning convergence analysis is performed. The third section is concerned with a complex results case study: a reference model tracking attitude control for a multivariable quadrotor UAV with an actuator fault and an uncertain model. A critical results discussion and interpretations are offered in the fourth section. The final section concludes the findings.

2. Materials and Methods

2.1. The Controlled Uncertain Dynamic System

The original system's state-space transition model with output equation is defined for the linear-time-invariant case as

$$\begin{aligned}x_{k+1} &= A_o x_k + B_o u_k, \\ y_k &= C_o x_k + D_o u_k,\end{aligned}\tag{1}$$

where $x_k \in \mathbb{D}_x \subset \mathbb{R}^n$ is the state within its domain $\mathbb{D}_x$, $u_k \in \mathbb{D}_u \subset \mathbb{R}^m$ is control input within its domain $\mathbb{D}_u$, $y_k \in \mathbb{D}_y \subset \mathbb{R}^p$ is the controlled output within its domain $\mathbb{D}_y$. Subscript k indexes the discrete time sample herein. Herein, the matrices A_o, B_o, C_o, D_o are of appropriate dimensions. They are assumed generally unknown and they are being referred to as the *true* matrices. Their values could be expressed under additive or multiplicative uncertain formulation. The matrix B_o could be expressed as $B_o = B_{nom} \cdot \Delta B \cdot \Lambda$, where Λ models an actuation fault present in the control input u_k whenever the value differs from $\Lambda = I_m$, and ΔB represents the deviation of the input matrix from its nominal value B_{nom} . This fault type described by multiplicative matrix Λ is a form of partial effectiveness loss and not a complete malfunction. Similar arguments are possible for representing uncertainty about the remaining state matrices A_o, C_o, D_o , using their nominal counterparts. The described form (1) is more general, as it is able to capture time delays in the variables u_k, x_k, y_k as well. This is provable by augmenting (1) with additional state components and it is a specific feature of discrete-time system representation.

Assumption 1. *The system described by (1) is both controllable and observable.*

Following Assumption 1, an equivalent representation of (1), called a virtual system, is, in its state-space form

$$\begin{aligned}v_{k+1} &= A_v v_k + B_v u_k, \\ y_k &= C_v v_k + D_v u_k,\end{aligned}\tag{2}$$

where $v_k \in \mathbb{D}_v \subset \mathbb{R}^v$ is called a virtual state vector. The innovation behind v_k is that it consists of the input/output (IO) present and past values, i.e., $v_k := [y_k^T, y_{k-1}^T, \dots, y_{k-ny}^T, u_{k-1}^T, \dots, u_{k-nu}^T]^T$ with appropriate size of $v = (ny + 1) \times p + nu \times m$, where ny is the output order, nu is the input order. Most notably, (1) and (2) share the same inputs and outputs. This means that a suitable controller for system (2) in fact uses the IO data from the system (1) and elaborates the control for (2), which in fact is equivalent to controlling the original system (1). From the IO behavior perspective, systems (1) and (2) are alike [34,36]. What is more appealing about (2) is that v_k is a linear transform on x_k , which is captured by the next Theorem.

Theorem 1. *For an observable and controllable system (1), there exists a transformation matrix $\mathcal{T}$ which enables $x_k = \mathcal{T} v_k$.*

Proof of Theorem 1. See Appendix A. $\square$

In the light of Theorem 1 previously introduced, it is well-known that for the integer size $v \geq v_{\text{minimal}}$ (or equivalently for orders ny, nu greater than some minimal thresholds), more IO samples captured in v_k do not bring more information about the unique state value x_k . The value of v indirectly set by the values ny, nu trades off the high dimensionality of v_k with the inter-correlation between IO samples, i.e., it balances the exploration–exploitation factor. In practice, we start with v corresponding to a minimal

order known from historical analysis about the system (1) and increment it in unit steps, until there is no more gain in control performance.

2.2. The Reference Model to Be Tracked in Output and Matched in Dynamics

To force a desired control behavior for system (1) through its equivalent description (2), the most natural approach stemming from classic control is the reference model. This is, in fact, a supervised machine learning approach where one models the controller such that the closed loop matches a reference model's dynamics [37]. For the single input–single output case, a reference model is commonly described by the linear transfer function $H(s) = \frac{y^m(s)}{r(s)} = \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2} e^{-sT_m}$ parameterized in the natural frequency w_n , the damping ratio ξ , and the time delay T_m with Laplace argument s . $y^m(s)$ is the Laplace-domain reference model output, with $r(s)$ being the Laplace image of the reference input time signal. Generally, it is more intuitive to describe control performance in continuous-time and only afterwards discretize the transfer function to match the time-based description with systems (1) and (2) pertaining to the discrete-time representation. The parameter w_n is tightly related to the rise time while ξ determines the overshoot. Such relationships are found in families of characteristics or from tables. For a second order reference model,

ξ produces percent overshoot of $M_p = e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}} \times 100[\%]$, while the rise time is $t_r = \frac{\pi - \phi}{w_n \sqrt{1-\xi^2}}$ for $\phi = \arctan\left(\frac{\sqrt{1-\xi^2}}{\xi}\right)$.

In multivariable control, a transfer matrix reference model is commonly employed, which in most cases is also diagonal, in order to indirectly enforce control channel decoupling to the maximum extent possible, which greatly improves disturbance rejection resilience and debugging in the control performance. In model reference control, if we use a linear reference model for a nonlinear plant, a nonlinear controller is necessary if we want the closed loop to match the linear response of the reference model. A plain linear controller cannot achieve such a requirement. With a nonlinear controller, we aim for good closed-loop linearization. Such a behavior is very predictable, both in scale and in response, over wide operating ranges. This is a very desirable control systems feature. This kind of generalizability emerges from the linear systems superposition principle. A side note about the reference model: systems like (1) or (2) may also possess a non-minimum phase character, which is a kind of dynamics which should not be compensated for via the controller dynamics; hence, it must be accepted and forced as an explicit component in $H(s)$.

Eventually, $H(s)$ needs to be discretized and, for online ADP and RL control, it also needs to be transformed to a suitable state-space model. The final reference model state space would be described as

$$\begin{aligned} x_{k+1}^m &= A_m x_k^m + B_m r_k, \\ y_k^m &= C_m x_k^m + D_m r_k, \end{aligned} \quad (3)$$

with x_k^m the reference model's state of appropriate and known size, $r_k, y_k^m \in \Omega_y$ being the reference inputs and reference model outputs, respectively, pinpointing that they must live in the same space as the original system's outputs. The tuple (A_m, B_m, C_m, D_m) is one state-space realization out of infinitely many possible. This reference model is more flexible than most in the literature found in ADP and RL, since it stems from performance indicators which are familiar to most control system designers. The reference model output must be controllable and observable. For accuracy, we formally assume it.

Assumption 2. *The reference model (3) is both controllable and observable.*

2.3. The Driving Reference Input Model

A linear reference input model is next employed with the dynamics

$$r_{k+1} = A_r r_k, \quad (4)$$

where A_r is $p \times p$ and compatible with the number of controlled outputs. The reference input r_k drives both the closed-loop control system (to be designed for (1), based on (2), as shown in the next subsections) and the reference model.

2.4. The Extended State-Space System

To arrive at a final state-space model that complies with the ADP and RL formulation, we define $\chi_k = [v_k^T, (x_k^m)^T, r_k^T]^T$. Then the augmented system is

$$\begin{aligned} \chi_{k+1} &= \begin{bmatrix} A_v & 0 & 0 \\ 0 & A_m & B_m \\ 0 & 0 & A_r \end{bmatrix} \chi_k + \begin{bmatrix} B_v \\ 0 \\ 0 \end{bmatrix} u_k := A \chi_k + B u_k \\ y_k &= C_y \chi_k \\ y_k^m &= C_{ym} \chi_k \end{aligned} \quad (5)$$

For $\chi_k = [v_k^T, (x_k^m)^T, r_k^T]^T$ the size of the extended state vector results from the sizes of its independent components, while C_y, C_{ym} are appropriate matrices extracting the original system's outputs and the reference model outputs from χ_k , respectively. This augmented system is next used for learning optimal control: it is fully state observable and controllable. The augmented system from (5) is fully observable and controllable by construction. The reference input r_k from (4), although being a state within χ_k in (5) and not a traditional control input, is a user-set variable whose dynamics are selected beforehand. This r_k drives the reference model, which is selected to be fully observable and controllable; otherwise, its output is not to be tracked. The virtual state space $v_{k+1} = A_v v_k + B_v u_k$ in (2) is fully observable since its state is built from IO measured data. Its controllability follows, as it has the same input and output as (1), which is assumed controllable. For this to happen, sufficient past IO samples must fill the shift buffers from v_k .

2.5. The Model Reference Tracking Cost in the Optimal ADP/RL Control

To induce a control learning behavior, the undiscounted infinite-horizon cost-to-go is

$$J(\chi_k) = \sum_{i=0}^{\infty} \rho(\chi_{k+i}, u_{k+i}), \quad (6)$$

where $\rho(\chi_k, u_k) = \|y_k(\chi_k) - y_k^m(\chi_k)\|^2 + u_k^T R u_k$ captures the normed tracking error between the controlled system output and the reference model output, across all channels, and also penalizes the control effort. Generally, in the model reference tracking problem, no control penalty is used; hence, it is a little different from the classical LQR penalty. To solve the model reference tracking as an optimal control, we proceed as follows:

1. Define the controller-dependent cost $J_{\mathfrak{C}}(\chi_k) = \sum_{i=0}^{\infty} \rho(\chi_{k+i})$ to make sure that, starting from every state χ_k , the subsequent state transitions to $\chi_{k+1}, \chi_{k+2}, \dots$ are performed under the impact of the controller $u_k = \mathfrak{C}(\chi_k)$ in its most general form. This controller form will be linear in the case of linear system, i.e., $u_k = \mathfrak{C}(\chi_k) = K^T \chi_k$.
2. To solve for the optimal control $\mathfrak{C}^* = \arg \min_{\mathfrak{C}} J_{\mathfrak{C}}(\chi_k) = \sum_{i=0}^{\infty} \rho(\chi_{k+i})$, the ADP (RL) methods generally require the system model to be accurately known. To solve for the optimal control independently of the system model, introduce the extended cost $Q_{\mathfrak{C}}(\chi_k, u_k) = \rho(\chi_k, u_k) + \sum_{i=k+1}^{\infty} \rho(\chi_i)$. This is the well-known Q-function, and it obeys the Bellman equation as

$$Q_{\mathfrak{C}}(\chi_k, u_k) = \rho_i(\chi_k, u_k) + Q_{\mathfrak{C}}(\chi_{k+1}, \mathfrak{C}(\chi_{k+1})) = \rho_i(\chi_k, u_k) + J_{\mathfrak{C}}(\chi_{k+1}). \quad (7)$$

3. Smartly parameterize the Q-function and the controller using polynomials, or neural networks in the most general case.
4. Simultaneously learn the optimal Q-function and the optimal controller in an off-policy style, using a dataset of transition samples collected from the system under any controller, most often an existing stabilizing one that is suboptimal and it is employed for good exploration. This dataset is called the Experience Replay Buffer (ERB).

For a linear system like (5), the classical LQR theory proves that the cost of operation under a linear controller is $J_{\mathfrak{C}}(\chi_k) = \chi_k^T P \chi_k$, quadratic in the state, under the controller $\mathfrak{C}$, with $P \succcurlyeq 0$ some symmetric positive definite matrix, and the optimal controller linear in the state, i.e., $u_k = K^T \chi_k$.

Proposition 1. For the model reference tracking problem with the cost (6), the penalty complies with an LQR penalty formulation, having state and action penalty matrices $\bar{Q}, R$, as in $\rho_k = \chi_k^T \bar{Q} \chi_k + u_k^T R u_k$, where $\bar{Q} \succcurlyeq 0, R \succ 0$ are semipositive definite and positive definite, respectively.

Proof of Proposition 1. It follows immediately since $\rho_k = \|y_k(\chi_k) - y_k^m(\chi_k)\|^2 + u_k^T R u_k = (y_k(\chi_k) - y_k^m(\chi_k))^T (y_k(\chi_k) - y_k^m(\chi_k)) + u_k^T R u_k = \chi_k^T (C_y^T - C_{ym}^T)(C_y - C_{ym})\chi_k + u_k^T R u_k = \chi_k^T \bar{Q} \chi_k + u_k^T R u_k$, where $\bar{Q} \succcurlyeq 0, R \succ 0$. $\square$

Subsequently, it is straightforward to show that the extended Q-function cost is also quadratic in the argument $z_k = [\chi_k^T, u_k^T]^T$, i.e., $Q_{\mathfrak{C}}(\chi_k, u_k) = z_k^T H z_k$, with $H \geq 0$ some symmetric positive definite matrix, while the optimal control still remains linear in the state. The most important utility of the Q-function is that the control improvement is found directly from the condition $\frac{dQ_{\mathfrak{C}}(\chi_k, u_k)}{du_k} = 0$, and it is analytically solvable with the mentioned parameterization.

A linear state feedback controller of the form $u_k = K^T \chi_k$ with gain matrix K partitioned as $K^T = [K_1^T | K_2^T | K_3^T]$ renders the augmented closed-loop as

$$\chi_{k+1} = \begin{bmatrix} A_v + B_v K_1^T & B_v K_2^T & B_v K_3^T \\ 0 & A_m & B_m \\ 0 & 0 & A_r \end{bmatrix} \chi_k = A_{cl} \chi_k. \quad (8)$$

The Q-function is linearly parameterized by retaining the unique up-to-second-degree polynomials resulting from the argument products. Let $Q_{\mathfrak{C}}(z_k)$ be expressed for $z_k = [\chi_k^T, u_k^T]^T := [z_{k,1}, z_{k,2}, \dots, z_{k,t}]^T \in \mathbb{R}^t$ as a t -dimensional vector as

$$Q_{\mathfrak{C}}(z_k|\theta) = \theta_1 z_{k,1}^2 + \theta_2 z_{k,1} z_{k,2} + \dots + \theta_t z_{k,1} z_{k,t} + \theta_{t+1} z_{k,2}^2 + \theta_{t+2} z_{k,2} z_{k,3} + \dots + \theta_{\frac{t(t+1)}{2}} z_{k,t}^2 =: \varphi^T(z_k) \theta, \quad (9)$$

where $\theta \in \mathbb{R}^{\frac{t(t+1)}{2}} := [\theta_1, \dots, \theta_{\frac{t(t+1)}{2}}]^T$ lumps all the parameters together and $\varphi(z_k) \in \mathbb{R}^{\frac{t(t+1)}{2}} := [z_{k,1}^2, z_{k,1} z_{k,2}, \dots, z_{k,t}^2]^T$ is a basis function vector called nonlinear embedding.

Remark 1. It follows immediately from (9) that the Q-function takes the form $Q_{\mathfrak{C}}(z_k|\theta) = z_k^T H z_k$

with $H := \begin{bmatrix} \theta_1 & \theta_2/2 & \dots & \theta_t/2 \\ \theta_2/2 & \theta_{t+1} & \dots & \theta_{2t-1}/2 \\ \vdots & \vdots & \ddots & \vdots \\ \theta_t/2 & \theta_{2t-1}/2 & \dots & \theta_{\frac{t(t+1)}{2}} \end{bmatrix}$ being a positive definite symmetric matrix, to be proven later.

Let the transition samples database in the ERB be $DS = \{(\chi_k^{[i]}, u_k^{[i]}, \chi_{k+1}^{[i]})\}, i = 1, \dots, N_D$. For the previous iteration controller K_{i-1} , with known previous iteration Q-function parameter θ_{i-1} , we use a number N_T of transition samples from the ERB, to write the following VI update in the Q-function parameter space

$$\begin{bmatrix} \varphi^T(\chi_k^{[1]}, u_k^{[1]}) \\ \varphi^T(\chi_k^{[2]}, u_k^{[2]}) \\ \vdots \\ \varphi^T(\chi_k^{[N_T]}, u_k^{[N_T]}) \end{bmatrix} \theta_i = \begin{bmatrix} \rho_i(\chi_k^{[1]}, u_k^{[1]}) + \varphi^T(\chi_{k+1}^{[1]}, K_{i-1}^T \chi_{k+1}^{[1]}) \theta_{i-1} \\ \rho_i(\chi_k^{[2]}, u_k^{[2]}) + \varphi^T(\chi_{k+1}^{[2]}, K_{i-1}^T \chi_{k+1}^{[2]}) \theta_{i-1} \\ \vdots \\ \rho_i(\chi_k^{[N_T]}, u_k^{[N_T]}) + \varphi^T(\chi_{k+1}^{[N_T]}, K_{i-1}^T \chi_{k+1}^{[N_T]}) \theta_{i-1} \end{bmatrix}, \quad (10)$$

which can be expressed as an overdetermined system $\tilde{M}\theta_i = \tilde{N}$ and be solved accordingly as $\theta_i = (\tilde{M}^T \tilde{M})^{-1} \tilde{M}^T \tilde{N}$. In Value Iteration, we use parameterized notation $Q_i(x_k, u_k) = \varphi^T(\chi_k, u_k) \theta_i$ for the i -th iteration Q-function. In the right hand side of the VI update, $K_{i-1} = \min_K Q_{i-1}(\chi_{k+1}, u_k) = \min_K \varphi^T(\chi_{k+1}, K^T \chi_{k+1}) \theta_{i-1}$.

Remark 2. The major advantage of (10) is that it is model-free and independent of system matrices A, B.

Remark 3. Equation (10) is a policy estimate update and not a policy evaluation. If θ_i was used in the right-hand side, the available database DS helps evaluating the Q-function of any arbitrary policy K that fits into the expression of $\varphi^T(\chi_{k+1}^{[j]}, K^T \chi_{k+1}^{[j]})$.

Remark 4. To solve VI policy estimate update (10) using $\theta_i = (\tilde{M}^T \tilde{M})^{-1} \tilde{M}^T \tilde{N}$, the matrix $(\tilde{M}^T \tilde{M})$ needs to be invertible, requiring $\tilde{M}$ to be full column rank. This is achieved by sufficiently exploratory $u_k^{[j]}$, commonly obtained by noise injection, to decorrelate the tuples $(\chi_k^{[j]}, u_k^{[j]})$ from each other. Accordingly, this decorrelates the expansions $\varphi^T(\chi_k^{[j]}, u_k^{[j]})$, which are the columns for $\tilde{M}$.

With θ_i found from (10), we proceed with updating the controller and finding K_i from the condition $\frac{dQ_{\mathbb{Q}}(\chi_k, u_k | \theta)}{du_k} = 0$. Let this controller be

$$K_i = L(\theta_i), \quad (11)$$

where $L(\cdot)$ is some analytically derived expression of θ_i resulting from the solution $\frac{dQ_{\mathbb{Q}}(\chi_k, u_k^* | \theta)}{du_k} = 0$. It follows that the stationary point $u_k^* = K_i^T \chi_k$ is to be linear in the state. The analytical derivation is generally possible *only for well-posed parameterizations* such as the quadratic one used herein.

Example 1. Suppose that $\chi_k = [\chi_{k,1}, \chi_{k,2}]^T$ and $u_k \in \mathbb{R}$. Then the nonlinear embedding vector is $\varphi(z_k) = [(\chi_{k,1})^2, \chi_{k,1} \cdot \chi_{k,2}, \chi_{k,1} \cdot u_k, (\chi_{k,2})^2, \chi_{k,2} \cdot u_k, (u_k)^2]^T \in \mathbb{R}^6$ and $Q(z_k | \theta) = \theta_1 \chi_{k,1}^2 + \theta_2 \chi_{k,1} \cdot \chi_{k,2} + \theta_3 \chi_{k,1} \cdot u_k + \theta_4 \chi_{k,2}^2 + \theta_5 \chi_{k,2} \cdot u_k + \theta_6 u_k^2$. The solution to $\frac{dQ_{\mathbb{Q}}(\chi_k, u_k^* | \theta)}{du_k} = 0$ is $u_k^* = -[\frac{\theta_3}{\theta_6}, \frac{\theta_5}{\theta_6}] [\chi_{k,1}, \chi_{k,2}]^T$, which confirms the linear dependence in the state, of the form $u_k = K^T \chi_k$.

Next, we propose the online off-policy asynchronous real-time algorithm for the model reference tracking control (OOART-MRTC) problem.

2.6. The Summarized Algorithm—OOART-MRTC

The online off-policy asynchronous real-time algorithm used for model reference trajectory tracking is given below in Algorithm 1:

Algorithm 1. The OOART-MRTC algorithm

1. **Input** parameters: K_{-1}, θ_0 that characterizes $Q_0 := Q_{K_{-1}}$, the minibatch size N_T , orders ny, nu .
2. Reset the environment x_0^m, r_0, v_0 and fill the IO buffers with appropriate data.
3. With the initially stabilizing controller K_{-1} in closed loop, collect transitions to fill in the ERB, using exploratory noise for good state-action coverage.
4. Set update iteration index $i = 0$.
5. **repeat** (*in the main thread, this is the step function with time-critical operations*)
6. read y_k, y_k^m . Push the newest y_k to the outputs buffer.
7. form the current sample sliding window virtual state $v_k^T := [y_k^T, y_{k-1}^T, \dots, y_{k-ny}^T, u_{k-1}^T, \dots, u_{k-nu}^T]$ from the IO buffers.
8. construct the augmented state $\chi_k = [v_k^T, (x_k^m)^T, r_k^T]^T$.
9. calculate $u_k = K_i^T \chi_k$.
10. send $u_k + \eta_k$ to the system, where η_k is exploratory noise.
11. compute the penalty ρ_k at the current timestep.
12. push the transition sample $(\chi_{k-1}, u_{k-1}, \chi_k)$ in the ERB.
13. push the latest u_k to the inputs buffer.
14. set $\chi_{k-1} \leftarrow \chi_k$.
15. update the reference input based on the generative model $r_{k+1} = R(r_k)$.
16. update the reference model state according to $x_{k+1}^m = A_r x_k^m + B_r r_k$.
17. make $k = k + 1$.
18. sleep until the current timestep ΔT has elapsed.
19. **until** experiment is stopped.
- 20.
21. **repeat** (*in a separate thread where the learning iterative updates take place*)
22. sample the ERB with random N_T transitions.
23. fill the matrices in (10) using the transition samples.
24. **if** $i == 0$,
25. find θ_0 for K_{-1} , using the policy evaluation variant of (10) with θ_0 on both sides.
26. **else**
27. find θ_i based on K_{i-1}, θ_{i-1} using Equation (10).
28. find K_i from θ_i using Equation (11).
29. increment the iteration index $i = i + 1$.
30. **until** convergence or main thread is stopped

Specifics of the OOART-MRTC algorithm are discussed. The main computing thread first performs the time-critical operations (lines 6–10 in the OOART-MRTC algorithm), among the most important being outputs measurements, augmented state construction by moving a window over the IO data samples, control action computation, and sending to the environment, either under exploration mode (with noise) or without it. For real-time operation and interaction between the controller and the environment, resilient communication backbones are to be adopted, either low-level, like socket-based, or high-level, like Robot Operating System (ROS)-based [38]. Some less critical but still important management tasks are (lines 11–17 in the OOART-MRTC algorithm): the transition samples are collected in this thread also and pushed to the ERB, the reference input generative model and the reference model state-space are both updated for one timestep. The remaining time left from the sampling period ΔT is slept until the next interaction and control cycle begins.

In the parallel executing thread, the learning updates take place by sampling the ERB randomly for transitions and updating the critic and actor equations. Depending on the number of transitions, the time taken for these tasks may often exceed ΔT ; hence, the

asynchronous update is imperative. Dedicated software mechanisms for preventing race conditions on reading and writing variables are required (like mutexes); see [37]. The locked variables are (1) the actor weights K for reading, in order to make atomic inference on the actor in the control thread, without updating these weights from the actor-critic learning thread at the same time; (2) the ERB, for sampling in the actor-critic thread, to prevent race condition when updating it in the control main thread.

2.7. Learning Convergence Analysis for the OOART-MRTC

We formalize the convergence analysis by reparametrizing the VI updates and by setting some assumptions.

Assumption 3. *There exists an initial admissible controller K_{-1} with cost-to-go $J_{K_{-1}}(\chi) = \chi^T P_{-1} \chi$ with $P_{-1} = P_{-1}^T \succcurlyeq 0$ positive definite symmetric. Let $\chi_+ = A\chi + Bu$ define a transition sample tuple (χ, u, χ_+) . Then the corresponding Q -function is $Q_0(\chi, u) = \chi^T \bar{Q}\chi + u^T Ru + Q_0(\chi_+, K_{-1}\chi_+) = \chi^T \bar{Q}\chi + u^T Ru + J_{K_{-1}}(\chi_+) = \rho(\chi, u) + \chi_+^T P_{-1} \chi_+$.*

Define the unparameterized VI policy estimate update, which starts with Q_0 and uses the greedy updates, as

$$Q_i(\chi, u) = \rho(\chi, u) + \min_u Q_{i-1}(\chi_+, u). \quad (12)$$

Remark 5. *The value Q_0 is a true policy evaluation for K_{-1} hence strongly related to $J_{K_{-1}}$, while the subsequent Q_i will not represent true policy evaluations for the subsequent controllers K_i ; hence, these updates will not be tightly related to costs-to-go, but only loose approximations. Moreover, the existence of $P_{-1} = P_{-1}^T \succcurlyeq 0$ defining the value function of using a stabilizing K_{-1} is guaranteed.*

Theorem 2. *The sequence Q_i is monotonically decreasing as in $Q_i(\chi, u) \leq Q_{i-1}(\chi, u) \leq \dots \leq Q_0(\chi, u)$ and $Q_i \geq 0$. Moreover, $Q_i(\chi, u) = \begin{bmatrix} \chi \\ u \end{bmatrix}^T H_i \begin{bmatrix} \chi \\ u \end{bmatrix} := z^T H_i z$ preserves its quadratic form over the updates.*

Proof of Theorem 2. $Q_0(\chi, u) = \rho(\chi, u) + \chi_+^T P_{-1} \chi_+ \geq 0$ follows from $\bar{Q} \succcurlyeq 0, R > 0, P_{-1} \succcurlyeq 0$. Based on $\chi_+ = A\chi + Bu$, we have $Q_0(\chi, u) = \chi^T \bar{Q}\chi + u^T Ru + (A\chi + Bu)^T P_{-1} (A\chi + Bu)$. Moreover, $Q_0(\chi, u) = \begin{bmatrix} \chi \\ u \end{bmatrix}^T H_0 \begin{bmatrix} \chi \\ u \end{bmatrix} := z^T H_0 z$ is quadratic, with $H_0 = \begin{bmatrix} Q + A^T P_{-1} A & B^T P_{-1} A \\ A^T P_{-1} B & R + B^T P_{-1} B \end{bmatrix} \succcurlyeq 0$ and $H_0 = H_0^T$. $\square$

To find Q_1 from Q_0 based on (12) written as $Q_1(\chi, u) = \rho(\chi, u) + \min_u Q_0(\chi_+, u)$, we compute $u^* = \arg \min_u Q_0(\chi_+, u)$ from $\frac{dQ_0(\chi_+, u)}{du} = 0$ and arrive at

$$u^* = -(R + B^T P_{-1} B)^{-1} B^T P_{-1} A \chi_+ = K_0 \chi_+. \quad (13)$$

Then

$$\min_u Q_0(\chi_+, u) = Q_0(\chi_+, K_0 \chi_+) = \chi_+^T (\bar{Q} + K_0^T R K_0) \chi_+ + \chi_+^T (A^T + K_0^T B^T) P_{-1} (A + B K_0) \chi_+ := \chi_+^T P_0 \chi_+, \quad (14)$$

with

$$P_0 = \bar{Q} + K_0^T R K_0 + (A^T + K_0^T B^T) P_{-1} (A + B K_0) \succcurlyeq 0, \quad (15)$$

since $\bar{Q} \succcurlyeq 0, R > 0, P_{-1} \succcurlyeq 0$.

It follows that

$$Q_1(\chi, u) = \chi^T \bar{Q}\chi + u^T Ru + \chi_+^T P_0 \chi_+ \geq 0, \quad (16)$$

since $\bar{Q} \succcurlyeq 0, R > 0, P_0 \succcurlyeq 0$ and also that Q_1 is quadratic in its argument by writing $Q_1(\chi, u) = \begin{bmatrix} \chi \\ u \end{bmatrix}^T H_1 \begin{bmatrix} \chi \\ u \end{bmatrix} := z^T H_1 z$, with $H_1 = \begin{bmatrix} \bar{Q} + A^T P_0 A & B^T P_0 A \\ A^T P_0 B & R + B^T P_0 B \end{bmatrix} \succcurlyeq 0$ being positive definite symmetric.

It is also valid that

$$Q_1(\chi, u) = \rho(\chi, u) + Q_0(\chi_+, K_0 \chi_+) \leq \rho(\chi, u) + Q_0(\chi_+, K_{-1} \chi_+) := Q_0(\chi, u), \quad (17)$$

since K_0 is greedy w.r.t. Q_0 against any other controller, including K_{-1} .

To continue this rationale by induction, suppose that $Q_{i-1}(\chi, u) = \chi^T \bar{Q}\chi + u^T Ru + \chi_+^T P_{i-2} \chi_+ \geq 0$ holds for $\bar{Q} \succcurlyeq 0, R > 0, P_{i-2} \succcurlyeq 0$. Then

$$\min_u Q_{i-1}(\chi_+, u) = Q_{i-1}(\chi_+, K_{i-1} \chi_+) = \chi_+^T (\bar{Q} + K_{i-1}^T R K_{i-1}) \chi_+ + \chi_+^T (A^T + K_{i-1}^T B^T) P_{i-2} (A + B K_{i-1}) \chi_+ := \chi_+^T P_{i-1} \chi_+ \geq 0 \quad (18)$$

must hold because $\bar{Q} \succcurlyeq 0, R > 0, P_{i-2} \succcurlyeq 0$. This means that $P_{i-1} = \bar{Q} + K_{i-1}^T R K_{i-1} + (A^T + K_{i-1}^T B^T) P_{i-2} (A + B K_{i-1}) \succcurlyeq 0$, and the VI update

$$Q_i(\chi, u) = \chi^T \bar{Q}\chi + u^T Ru + \chi_+^T P_{i-1} \chi_+ \geq 0 \quad (19)$$

must also hold, since $\bar{Q} \succcurlyeq 0, R > 0, P_{i-1} \succcurlyeq 0$. The previous expression also takes the form $Q_i(\chi, u) = \begin{bmatrix} \chi \\ u \end{bmatrix}^T H_i \begin{bmatrix} \chi \\ u \end{bmatrix} := z^T H_i z$, proving that Q_i is itself quadratic at any iteration i .

Assume again by induction that $Q_{i-1}(\chi, u) \leq Q_{i-2}(\chi, u), \forall \chi, u$ and let

$$Q_i(\chi, u) = \rho(\chi, u) + \min_u Q_{i-1}(\chi_+, u), \quad (20)$$

$$Q_{i-1}(\chi, u) = \rho(\chi, u) + Q_{i-2}(\chi_+, K_{i-2} \chi_+), \quad (21)$$

where in the first equation above we used the greediness of K_{i-1} w.r.t. Q_{i-1} against any other controller, including K_{i-2} . By subtracting the second equation from the first, above, we obtain

$$Q_i(\chi, u) - Q_{i-1}(\chi, u) \leq Q_{i-1}(\chi_+, K_{i-2} \chi_+) - Q_{i-2}(\chi_+, K_{i-2} \chi_+) \leq 0, \quad (22)$$

because $Q_{i-1}(\chi_+, K_{i-2} \chi_+) \leq Q_{i-2}(\chi_+, K_{i-2} \chi_+), \forall \chi_+$, which results in

$$Q_i(\chi, u) \leq Q_{i-1}(\chi, u). \quad (23)$$

The above reasoning proves that $Q_i(\chi, u) \leq Q_{i-1}(\chi, u) \leq \dots \leq Q_0(\chi, u)$. $\square$

Theorem 3. Every K_i resulting as a byproduct of the VI update is stabilizing with every iteration.

Proof of Theorem 3. Recall that K_{-1} is admissible and stabilizing. Notice that

$$Q_1(\chi, u) = \chi^T \bar{Q}\chi + u^T Ru + \chi_+^T P_0 \chi_+ = \rho(\chi, u) + \chi_+^T [\bar{Q} + K_0^T R K_0 + (A^T + K_0^T B^T) P_{-1} (A + B K_0)] \chi_+, \quad (24)$$

and also recall that

$$Q_0(\chi, u) = \rho(\chi, u) + \chi_+^T P_{-1} \chi_+. \quad (25)$$

Compute $Q_1(\chi, u) - Q_0(\chi, u) = \chi_+^T [\bar{Q} + K_0^T R K_0 + (A^T + K_0^T B^T) P_{-1} (A + B K_0) - P_{-1}] \chi_+$ and since $Q_1(\chi, u) \leq Q_0(\chi, u)$ (proven by previous Theorem 2), it follows that

$$\bar{Q} + K_0^T R K_0 + (A^T + K_0^T B^T) P_{-1} (A + B K_0) - P_{-1} \preceq 0 \quad (26)$$

which is a discrete-time Lyapunov equation for the closed-loop system matrix $\bar{A} = A + B K_0$, where $\bar{Q} + K_0^T R K_0 \succcurlyeq 0$. This implies that K_0 is stabilizing. $\square$

By induction, suppose $Q_{i-1}(\chi, u) = \rho(\chi, u) + \chi_+^T P_{i-2} \chi_+$ holds and that by the optimal VI update

$$Q_i(\chi, u) = \rho(\chi, u) + \min_u Q_{i-1}(\chi_+, u) \quad (27)$$

we have

$$Q_i(\chi, u) = \rho(\chi, u) + \chi_+^T [\bar{Q} + K_{i-1}^T R K_{i-1} + (A^T + K_{i-1}^T B^T) P_{i-2} (A + B K_{i-1})] \chi_+. \quad (28)$$

Then, since $Q_i(\chi, u) - Q_{i-1}(\chi, u) \leq 0$, it must follow that

$$\bar{Q} + K_{i-1}^T R K_{i-1} + (A^T + K_{i-1}^T B^T) P_{i-2} (A + B K_{i-1}) - P_{i-2} \preceq 0, \quad (29)$$

which is a Lyapunov equation in discrete time for the closed-loop system matrix $\bar{A} = A + B K_{i-1}$. Its fulfilment implies that K_{i-1} is stabilizing, which concludes the proof. $\square$

Corollary 1. *The OOART-MRTC VI-like updates converge to the optimal Q-function $Q^*(\chi, u)$ and to the optimal control.*

Proof of Corollary 1. Using Theorem 2 and Theorem 3 results, we have $0 \leq Q_i(\chi, u) \leq Q_{i-1}(\chi, u) \leq \dots \leq Q_0(\chi, u)$ being monotonically decreasing and bounded above by Q_0 and below by 0, with K_i stabilizing at every iteration. Then the sequence Q_i must converge to a finite number, when $i \rightarrow \infty$, making $\lim_{i \rightarrow \infty} Q_i = Q^*$. Consequently, $\lim_{i \rightarrow \infty} K_i = K^*$, where Q^*, K^* defines the fixed point of the VI updates for which it is true that

$$Q^*(\chi, u) = \rho(\chi, u) + Q^*(\chi_+, K^* \chi_+). \quad (30)$$

$\square$

Remark 6. *The convergence analysis uses the system matrices A, B only to uncover the problem structure; however, the system knowledge is not really needed in the OOART-MRTC VI-like updates.*

3. Results: A Reference Model Tracking Attitude Control Case Study for a Multivariable Quadrotor UAV with an Actuator Fault and an Uncertain Model

A linearized quadrotor UAV attitude control model is

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{L}{I_x} & 0 & -\frac{L}{I_x} & 0 \\ 0 & \frac{L}{I_y} & 0 & -\frac{L}{I_y} \\ \frac{1}{I_z} & -\frac{1}{I_z} & \frac{1}{I_z} & -\frac{1}{I_z} \end{bmatrix} \Lambda \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \quad (31)$$

$$\Leftrightarrow x_{k+1} = A_{o_CT} x_k + B_{o_CT} \Lambda u_k,$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \Leftrightarrow y_k = C_{o_CT} x_k + D_{o_CT} u_k,$$

where ϕ, θ, ψ are the roll, pitch, and yaw attitude angle orientations of the body-fixed frame relative to the inertial frame, I_x, I_y, I_z are the inertial moments of the quadrotor, L is the rotor arm length relative to the mass center. Input variables $u_i, i = 1, \dots, 4$ are the *normalized* collective propeller forces. Here, Λ with the nominal value $\Lambda = I_4$ is the actuator health matrix that models a fault like the loss in actuation capacity, with positive

diagonal elements within the interval $\Lambda_{ii} \in (0,1]$. The model is a linearized one around hover position and neglects the inner propeller dynamics [18]. In this assumption, it is noted that the dynamics of the attitude angles behave like coupled double integrators about each of the three controlled outputs. With a sampling period of $T_s = 0.1[s]$, the discretized model with zero-order hold becomes compliant with description (1), with appropriate size matrices A_o, B_o, C_o, D_o (note the matrices $A_{o_CT}, B_{o_CT}, C_{o_CT}, D_{o_CT}$) in (31) are the continuous-time (CT) counterparts. The model is a four-input–three-output one with $m = 4, p = 3$.

3.1. Initial Stabilizing Controller Derivation

For the current case study, the considered nominal system parametric values are $L_{nom} = 0.3[m], I_{x,nom} = 0.2 [kg \cdot m^2], I_{y,nom} = 0.2 [kg \cdot m^2], I_{z,nom} = 0.4 [kg \cdot m^2], \Lambda = I_4$ similar to [25]. To these nominal values, the nominal CT matrices $A_{nom_CT} =$

$$\begin{bmatrix} 0 & 0 & 0 & 0.98 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.98 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.98 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B_{nom_CT} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{L_{nom}}{I_{x,nom}} & 0 & -\frac{L_{nom}}{I_{x,nom}} & 0 \\ 0 & \frac{L_{nom}}{I_{y,nom}} & 0 & -\frac{L_{nom}}{I_{y,nom}} \\ \frac{1}{I_{z,nom}} & -\frac{1}{I_{z,nom}} & \frac{1}{I_{z,nom}} & -\frac{1}{I_{z,nom}} \end{bmatrix} \text{ are used to}$$

discretize the state space model (31) and obtain the nominal discrete-time matrices A_{nom}, B_{nom} . No uncertainty is considered about the output equation matrices $C_{nom} = C_o, D_{nom} = D_o$. The resulting nominal state space matrices are only used for searching an initial, stabilizing controller, with no focus on performance except stability certification.

For the sixth order nominal quadrotor model, six closed-loop poles equally spaced in the interval $[-1, -0.5]$ are allocated, let them be $p_i, i = 1, \dots, 6$. Their discrete counterparts are $p_{di} = e^{T_s p_i}$. For the nominal system matrices $A_{nom}, B_{nom}, C_{nom}$, we solve for $K_{10} \in \mathbb{R}^{6 \times 4}$ by placing the poles of $A_{nom} + B_{nom} K_{10}^T$ at the desired locations p_{di} . For $ny = nu = r = 2$, let the observability matrix from Theorem 1 be $\bar{O} = [C_{nom} A_{nom}; C_{nom}]$, the controllability matrix be $\bar{C} = [B_{nom} | A_{nom} B_{nom}]$, the matrix $\bar{N} = \begin{bmatrix} C_{nom} B_{nom} & C_{nom} A_{nom} B_{nom} \\ 0 & C_{nom} B_{nom} \end{bmatrix}$, and the transform matrix be $\mathcal{T} = [A_{nom}^2 \text{pinv}(\bar{O} A_{nom}) [I | 0] \bar{C} - A_{nom}^2 \text{pinv}(\bar{O} A_{nom}) \bar{N}] \in \mathbb{R}^{6 \times 17}$. It follows that $A_v = \text{pinv}(\mathcal{T}) A_{nom} \mathcal{T}, B_v = \text{pinv}(\mathcal{T}) B_{nom}, C_v = C_{nom} \mathcal{T}$. Note that the pole placement for MIMO systems requires controllability for proper control authority distribution over each input. Arbitrary pole locations may pose numerical challenges, both because there are infinite state feedback gains and owing to very close poles being mistaken for higher multiplicity. If pole placement is not numerically reliable, then the stabilization design should fall back to LQR instead of pole placement.

Next, we find a state-feedback gain in the equivalent transformed virtual state-space, using the transform matrix $\mathcal{T}$. Let this gain be $K_1 = \mathcal{T}^T K_{10} \in \mathbb{R}^{17 \times 4}$, which makes the virtual closed-loop matrix $A_v + B_v K_1^T \in \mathbb{R}^{17 \times 17}$ a stable one.

From the output equation of the augmented system model (8), i.e., $y_k = C_{nom} x_k$, we express the steady-state conditions

$$\begin{aligned} y_\infty &= C_{nom} x_\infty = C_{nom} \mathcal{T} v_\infty, \\ v_\infty &= C_{nom} \mathcal{T} [(A_v + B_v K_1^T) v_\infty + B_v K_2^T x_\infty^m + B_v K_3^T r_\infty]. \end{aligned} \quad (32)$$

To find the K_3 that assures unit gain from $r_\infty \rightarrow y_\infty$, we set $K_2 = 0$ and solve for $v_\infty = (I - C_{nom} \mathcal{T} (A_v + B_v K_1^T))^{-1} B_v K_3^T r_\infty$. Then, $y_\infty = I \cdot r_\infty$ for $I = C_{nom} \mathcal{T} (I - C_{nom} \mathcal{T} (A_v + B_v K_1^T))^{-1} B_v K_3^T$, and K_3 is found.

The initial controller $K_{-1}^T := [K_1^T | K_2^T | K_3^T] \in \mathbb{R}^{4 \times 26}$ is used to stabilize the extended nominal state-space system (5), in order to collect the transition samples and fill the ERB.

Note that the model reference dynamics part and the reference input model part from (5) are already stable; hence, the sole concern is to stabilize the virtual state-space dynamics (2).

3.2. The Data Collection Phase with the Uncertain Dynamics and with Actuator Fault

A total of 2500 transition samples are collected to fill the ERB, by interacting with the quadrotor system under the initial stabilizing controller K_0 under an exploratory setting. A fault in the actuator is simulated by making $\Lambda = \text{diag}(0.9, 0.9, 0.9, 0.9)$ and with the inertial parameters I_x, I_y, I_z increased by 10% more than their nominal values. Hence, $B_{o_CT} = B_{nom} \cdot \Delta B \cdot \Lambda$. The true state transition matrix A_o is the discrete-time counterpart of the discretized system (31) with A_{o_CT} . The interaction is next performed with a different system than the one used for stabilization.

The reference input is reset at $r_k = 0$ on all three channels corresponding to roll, pitch, and yaw. Zero-mean normally distributed noise is added to the four control input channels at each sampling step time; its variance is 0.9. Every 50 samples, each reference switches to a uniform random amplitude value within $[-1; 1]$; however, we decorrelate the switching times of the reference inputs, in order to excite each channel independently and make it behave as a disturbance about the other control channels. This helps, furthermore, to achieve a good exploration. The collection phase results are displayed in Figure 1.

The reference inputs about each control channel also drive the reference model outputs. To impose a stable, desirable control performance, we employ an overdamped reference model of the form $H(s) = \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2}$ with $w_n = 1.5$ [rad/s], $\xi = 0.9$ about each control channel (i.e., for each controlled angle ϕ, θ, ψ), which is discretized for the same $T_s = 0.1$ [s]. Here, ξ corresponds to percent overshoot of $M_p = e^{\frac{-\xi\pi}{\sqrt{1-\xi^2}}} \times 100$ [%] = 0.15[%], while the rise time is $t_r = \frac{\pi - \phi}{w_n \sqrt{1-\xi^2}} = 4.11$ [s] for $\phi = \arctan\left(\frac{\sqrt{1-\xi^2}}{\xi}\right)$. The discretized reference model will be of the form (3), where $x_k^m \in \mathbb{R}^6, A_m \in \mathbb{R}^{6 \times 6}, B_m \in \mathbb{R}^{6 \times 3}, C_m \in \mathbb{R}^{3 \times 6}, D_m = 0 \in \mathbb{R}^{3 \times 3}$. More precisely, the continuous-time transfer matrix is therefore of the form

$$\begin{aligned} \begin{bmatrix} y_1^m \\ y_2^m \\ y_3^m \end{bmatrix} &= \begin{bmatrix} \phi^m \\ \theta^m \\ \psi^m \end{bmatrix} \\ &= \begin{bmatrix} \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2} & 0 & 0 \\ 0 & \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2} & 0 \\ 0 & 0 & \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2} \end{bmatrix} \begin{bmatrix} r_\phi \\ r_\theta \\ r_\psi \end{bmatrix}, \end{aligned} \quad (33)$$

and then discretized accordingly at $T_s = 0.1$ [s]. Using the results of Theorem 1, the final and complete virtual state-space representation constructed for $ny = nu = 2$ is of the form (5), with $\chi_k = [v_k^T, (x_k^m)^T, r_k^T]^T \in \mathbb{R}^{26}$, including $v_k = [y_k^T, y_{k-1}^T, y_{k-2}^T, u_{k-1}^T, u_{k-2}^T]^T \in \mathbb{R}^{17}$, $x_k^m \in \mathbb{R}^6, r_k \in \mathbb{R}^3$.

Remark 7. The transition samples pertaining to the reference model can be generated offline (a posteriori) after the transition sample collection from the true quadrotor system, or they can be generated online in real-time along with the system trajectories. In our work, we choose to compute them online in real-time. Their offline generation is possible since the reference input trajectory r_k is stored.

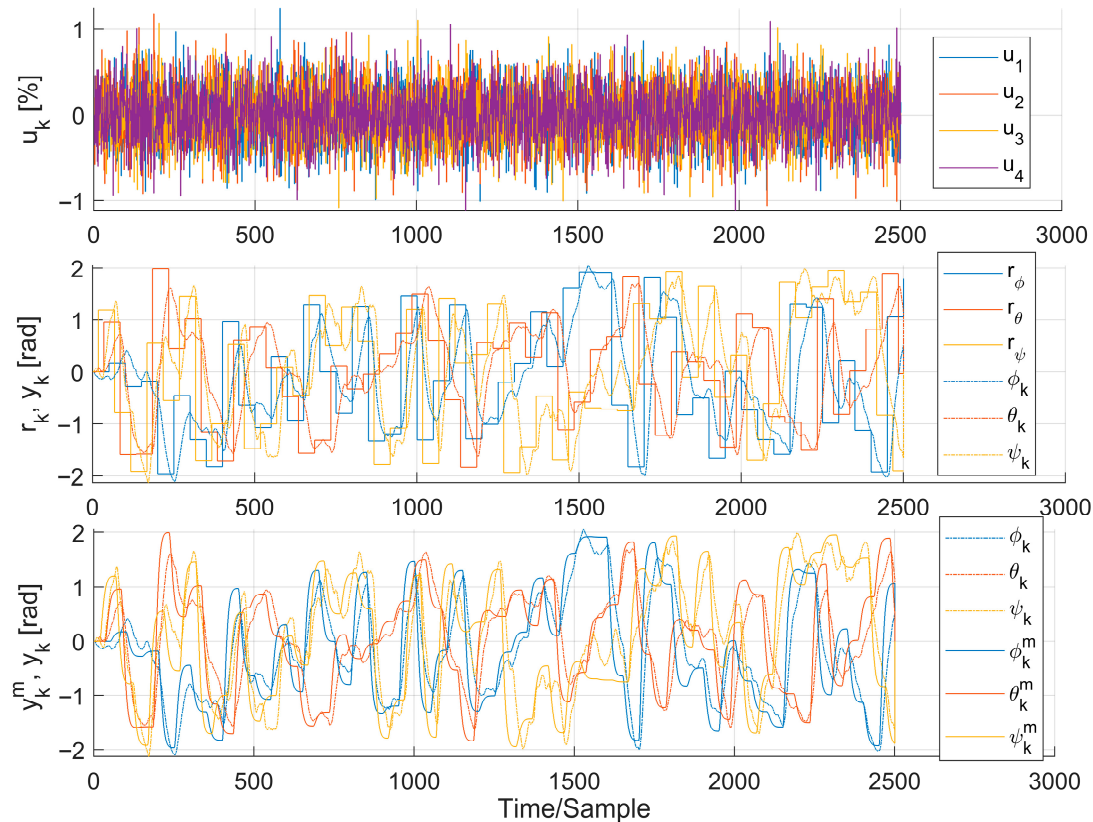


Figure 1. Transition sample collection phase. In the top subplot, the four control inputs are shown perturbed with normally distributed uncorrelated white noise, for exploration purposes. In the middle and bottom subplots, we see the reference inputs driving the model reference outputs and the controlled quadrotor outputs as the roll, pitch, and yaw angles.

3.3. The Learning Stage

The learning process starts from the initial state-feedback gain K_{-1} that was used for exploration and with θ_0 being initialized with random uniform values in $[0, 1]$. The penalty function is $\rho_k = \|y_k(\chi_k) - y_k^m(\chi_k)\|^2 + u_k^T R u_k$ with $R = 0.01 \cdot I_4$ being diagonal and positive definite.

At each iteration of the OOART-MRTC algorithm, in a parallel processing thread, we extract a minibatch of 1200 transitions and solve for θ_1 using Equation (10). Then K_0 is found based on Equation (11). This process is repeated until satisfactory convergence, i.e., until no more changes are observed in either θ or K . Solving such a large overdetermined system may exceed the allotted time available within a single sampling period. Thus, to make the computation feasible in online real-time, the algorithm learning updates are detached to a separate thread, according to the soft real-time software implementation principles exposed in [37]. Importantly, when extracting transition samples from the ERB, we filter out those samples which correspond to switching reference input values on any control channel. The reason is that the piecewise constant reference inputs comply with a generative model like $r_{k+1} = I_3 r_k$ only when the references are constant, not when they switch.

In Figure 2, we observe the learning results after 300 timesteps corresponding to about 150 actor-critic iterative updates that take place asynchronously. The model reference tracking is very accurate across each channel. Decoupling of all three control channels is ensured, which fulfils the desired control objective for this very high-order quadrotor UAV system, for which each of the three control channels behaves as a double integrator.

The actor weights K_i and critic weights θ_i throughout the learning iteration steps are visible in Figure 3. Both values of θ_i and the values of K_i stabilize after only few iterations. The model reference tracking is accurately satisfied even long before the actor and critic weights stabilize, as is visible in the samples from Figure 2. The actor has $4 \times 26 = 104$ parameters/weights and the critic has 465 parameters/weights.

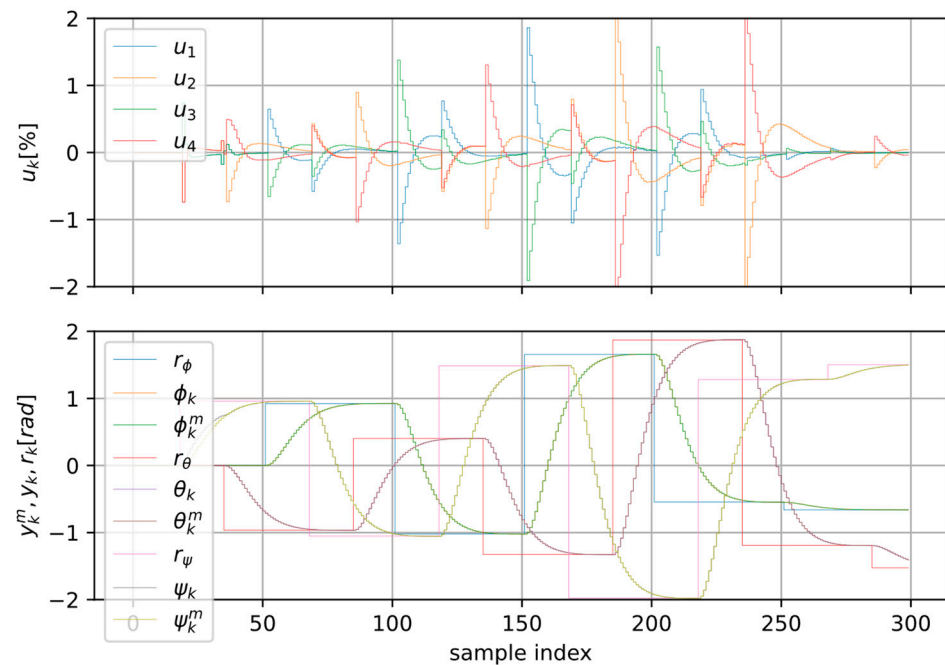


Figure 2. The control results after 300 timesteps (~150 asynchronous actor critic updates) reveal satisfactory and accurate reference model tracking, with decoupling assured across the three control channels that correspond to the roll, pitch, and yaw angles. Nota bene: the reference model outputs are barely distinguished from the actual controlled outputs, under highly accurate tracking.

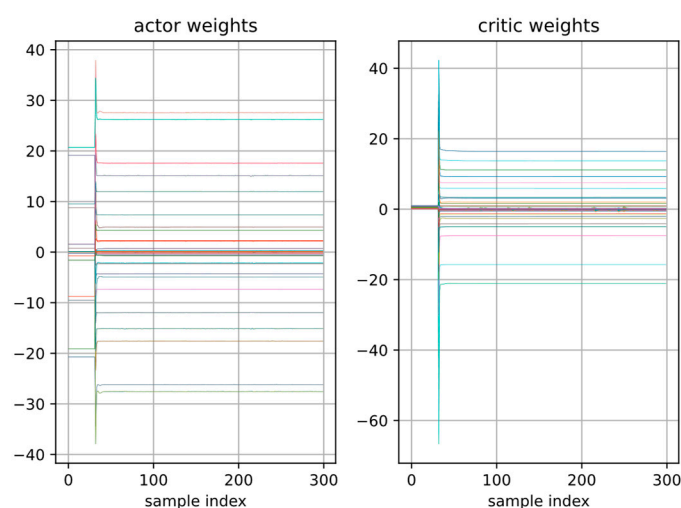


Figure 3. The evolution of K_i (left, 104 weights) and θ_i (right, 465 weights), monitored in the environment's real-time stepping thread, although the actor–critic updates take place asynchronously in a separate thread. Convergence to steady-state weight values is evident. The learning is delayed for ~25 time-steps, to substantiate the transition between before and after learning occurs.

The validated learning method based on OOART-MRTC relies strongly on the linearity assumption in the underlying system dynamics. The advantage is exposed clearly with the smart parameterization of the actor (controller) and the critic (extended Q-function). In the nonlinear dynamics case, neural networks are required [39–42]; however, such validation has already been performed for low-order systems [37]. Employing the virtual state representation leads to a very high-dimensional equivalent linear system which, although completely observable, comes with several challenges associated with the exploration. An efficient exploration implies that the sequences of inputs and outputs that form the augmented state vector must be decorrelated, which is only achievable with a sufficiently noisy excitation. While a good coverage of the state-action domain may not be easily attained in this case, the extrapolative capability that comes from the linear parameterization will make the optimality hold, at least in theory, at any scale around the equilibrium, no matter if the state transitions are collected in a confined smaller region, under persistent excitation. Hence, the clever parameterization compensates for the poorer exploration in this case.

3.4. Ablation Study 1: The Actor–Critic Update Time Relative to Environment Timestep

To monitor the actor–critic update time (ACUT) duration relative to the real-time sampling period of the environment ($T_s = 0.1$ [s] herein), the average ACUT duration is measured as a function of the batch size that is used to update the actor and critic weights. These batch sizes are {512, 768, 1024, 1536}. The *win-precise-time* (v. 1.4.2) and *time* libraries are used under Windows® and Python 3.10 for soft real-time environment stepping mechanisms and for function execution timing; the results are visible in Figure 4. Vectorized, batch-wise implementation with the *NumPy* library (v. 1.26.4) is used for code speed improvement when building the large-scale matrices and when solving the linear equation systems with the *numpy.linalg.lstsq* linear algebra package function.



Figure 4. The actor–critic update time (ACUT) duration function of the batch-size, relative to the environment timestep duration. The batch size here is the number of sampled points from the ERB used to update the actor and critic weights. For batch size 512, the ACUT is ~1.5 times the sampling period $T_s = 0.1$ [s].

Although we tested the learning with the batch sizes {512, 768, 1024, 1536}, we did not observe stabilization issues or significant control actions and tracking error bursts in the initial learning phase. This is in spite of increasing the ACUT from $1.5T_s$ to about $2T_s$. This indicates that the learning is robust and bump-less.

3.5. Ablation Study 2: Using VI-like Instead of PoIt-like Style in the First Critic Weight Update

In the OOART-MRTC algorithm, in the asynchronous actor–critic update thread, the critic weights θ_0 are found for K_{-1} using Equation (10) with θ_0 used on both sides. This corresponds to a PoIt-like critic update in the first step, and it is necessary, in order to evaluate the state-action Q-function of the initial stabilizing controller. The effects of this strategy shown with Figures 2 and 3 reveal the strong stabilization of the learning, despite an increased actor–critic update time duration.

To disclose the importance of this first critic update step, using PoIt-like strategy, we instead change it to a VI-like update, where the critic θ_0 is a randomly initialized vector within the unit magnitude uniform distribution. All subsequent critic updates are performed in VI-like style, which is known to converge, ultimately. However, a negative bursting effect is noticed, which is expected to alter the control stability, especially if the actor–critic update time duration increases. The learning behavior is shown in Figures 5 and 6.

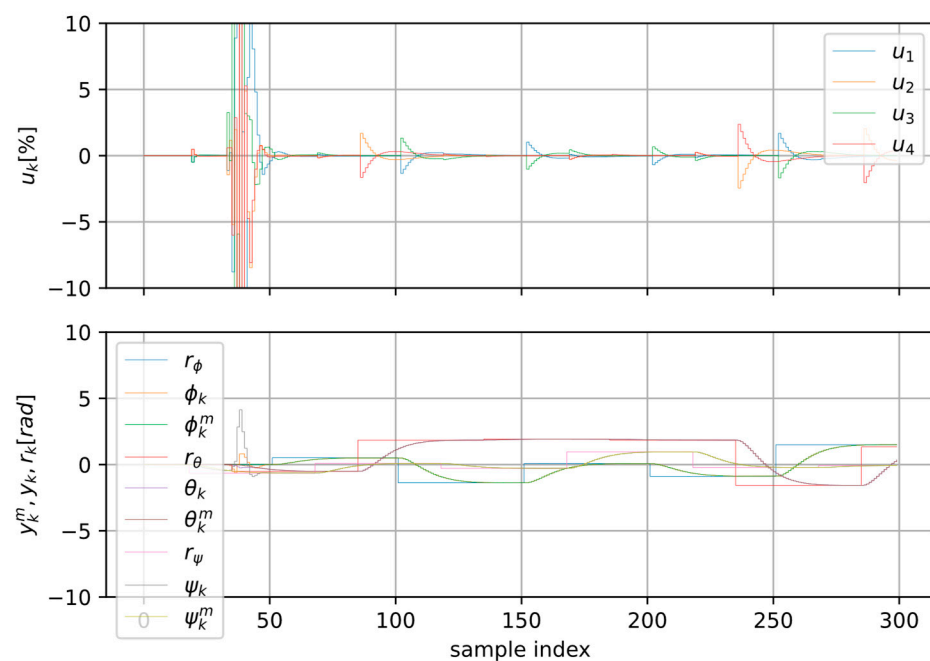


Figure 5. The control results after 300 timesteps (~150 asynchronous actor critic updates) reveal satisfactory and accurate reference model tracking, even when the first critic weight update is done in VI style, not in PoIt style. However, the bursting effect incurs a performance loss with the initial updates, and it is expected to worsen if the actor–critic update time duration increases.

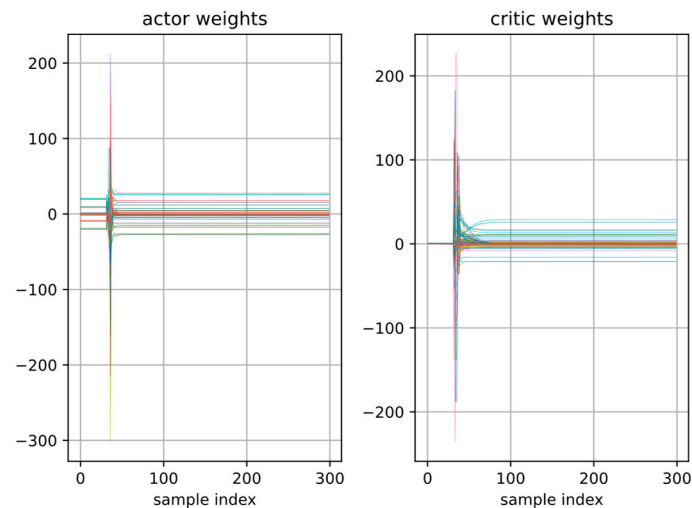


Figure 6. The evolution of K_i (left) and θ_i (right) is monitored in the environment's real-time stepping thread, although the actor–critic updates take place asynchronously in a separate thread. The learning ultimately takes place; however, the bursting effect in the weights points out a more fragile stabilization and it is expected to worsen if the actor–critic update time duration increases.

3.6. Ablation Study 3: Severe Actuator Loss and Performance Tracking Measurement

In the OOART-MRTC algorithm, we exacerbate the actuator loss, making $\Lambda = \text{diag}(0.9, 0.8, 0.5, 0.0)$. This means that one control action corresponding to one rotor is zero, as the propeller is stuck. This way, the actuation is asymmetric and saturating on the faulty propeller. We redo the OOART-MRTC learning, with maximal batch size of 1536, after data is collected with the new severely affected actuator. The controller designed in the first place based on the nominal model delivers the tracking performance in Figure 7, with measured RMSE averaged on three rollouts being 266.29×10^{-4} .

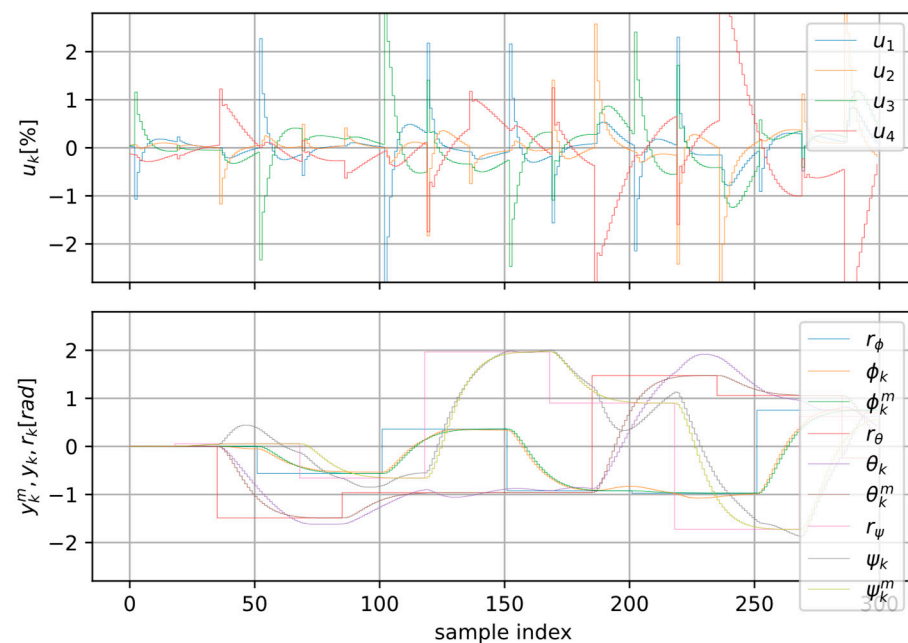


Figure 7. The control results with the initializing controller based on the nominal model. The asymmetric control action is visible, due to one actuator failure. This controller was used to collect the

initial 2500 transition samples, under exploratory noise, to build the database of transition samples for OOART-MRTC.

After learning takes place, the tracking is visible in Figure 8, with measured RMSE averaged on three rollouts being 1.62×10^{-4} , which is two orders of magnitude smaller than with the initial stabilizing controller used for transition sample collection. The asymmetric nature of the control action is clearly visible as they try to compensate for one actuator loss out of four. The fourth control action u_4 “learns” to become zero since it has no effect (see its nonzero action in Figure 7; however, its effect is also null there). Importantly, the result may not be feasible in the real world, since the altitude may not be held by such severe loss. Herein, the control configuration of the linearized operation near steady state permits such a result.

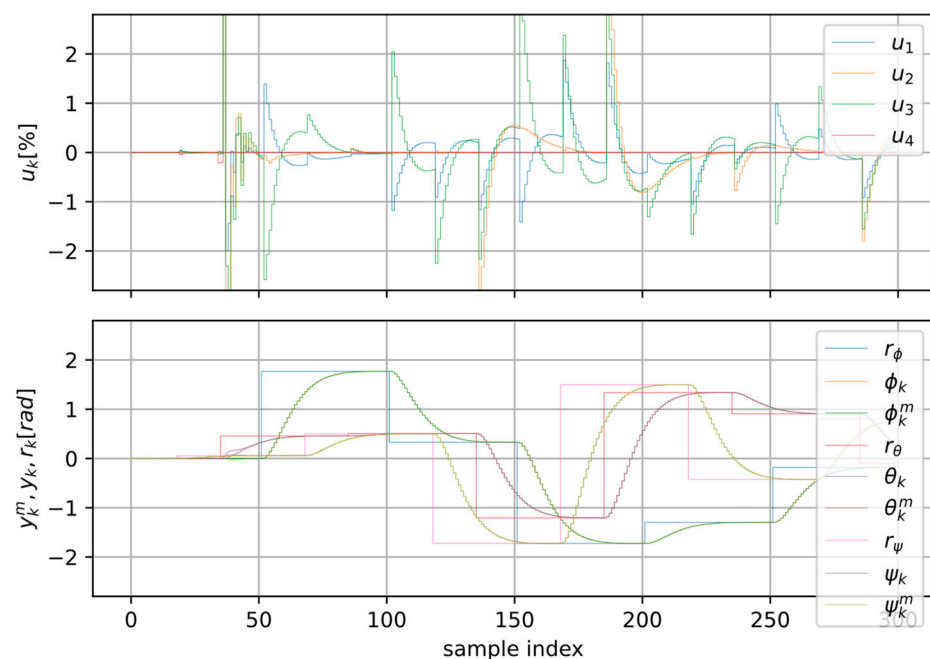


Figure 8. The control results with the learned OOART—MRTC controller, in the case of severe actuator fault. Three active control actions compensate for the fourth one, which is null.

4. Critical Discussions and Result Extensions

Regarding the transferability of the results towards other UAV types, the proposed framework is sufficiently general to accommodate other control configurations, since control necessities are ubiquitous. It is especially favorable towards the easily interpretable model reference tracking, and it can be extended to other performance requirements. Other potential control configurations include fixed-wing UAVs; ground-based unmanned vehicles (UGVs) like wheeled, tracked, legged/quadruped robots; unmanned surface vehicles (USVs); and unmanned underwater vehicles (UUVs).

Simulation experiments in this work were based on scenarios that favor real-world replication, to be performed near hovering point under prior stabilizing controllers, with step-like injected signals used as attitude setpoint excitations, leading to various quadrotor UAV attitude changes that generate the input–output data samples used for learning in the virtual embedded state-space representation. A dedicated flying space is required to allow for horizontal, constant-altitude motions. However, data collection can be relaxed near real operation missions, with injected attitude disturbances. These conditions are

reproducible with commercial autopilots (e.g., ArduPilot, PX4), by hooking up within software, at the attitude controller setpoints module, in order to produce synthetic, disturbance-like data injections. Then, by reading the attitude angles from filtered gyroscopes via Extended Kalman Filter (EKF) attitude estimators, together with the controller output signal, the learning data is logged to a database from where transition can be sampled in asynchronous mode, for learning. This learning takes place in a difference thread; hence, it should not affect the real-timeliness constraints for the fast attitude control loops. This aspect was validated with soft real-time validation in the case study. The commonly used control architecture in commercial autopilots, i.e., the P-type dominantly used for attitude and the inner cascaded PID-type rate controllers, can be replaced by the proposed state-feedback controller, which is, in fact, an equivalent mathematical representation.

There are, nevertheless, a number of limitations and corner cases that require further investigation. The linearity assumption holds to a limited extent in practice. In the concrete case of UAVs, aggressive maneuvers requiring drastic attitude changes will exacerbate the nonlinearity, leading to control robustness loss and potential instability issues. Secondly, there are other potential issues that could infringe assumptions like observability, for example, sensor bias, packet losses, and delayed measurements. For the attitude control, delayed measurements are a lesser problem if the delay is fixed, as the model (2) can be further augmented with additional states. However, it is problematic if it is time-varying. Packet loss is not a rough issue, at least in the attitude controller case, since the attitude controller belongs to the critical low-level communication infrastructure and it is carefully designed in practice. Moreover, its real-timeliness constraints are not affected when the learning takes place asynchronously. In real autopilots, the sensor bias is estimated and compensated using EKF, by fusing many sensors like optical flow, IMU, the altimeter, LiDAR, and the camera. Therefore, the attitude angles are relatively unbiased, but noisy.

Another potential limitation is with the fault and uncertainty modeling used here, which does not cover all corner cases of time-varying, asymmetric, saturating, or stuck. The multiplicative uncertainty partially accounts for asymmetric and time-varying cases. Actuator jamming is captured by zeroed entries in Λ . This is a further research direction.

5. Conclusions

For the class of low-order system applications where an initial prior stabilizing controller is available, exploration can be done to an extent which allows for asynchronous online learning of the optimal controller. This is more advisable than synchronous online learning since the asynchronous approach is unconstrained to run within a sampling period; hence, it can solve the update equations by taking into account significantly more transition samples. Using many transition samples with each iteration update improves the conditioning of the underlying problem (an overdetermined linear equation system in this case). The equivalent state-space representation based on input–output samples is a major advantage since it does not require the true system state to be measured; however, the virtual system makes the learning environment fully observable, even if the equivalent state-space representation becomes significantly higher-order. The tracking problem is augmented and articulated by a reference model that imposes a desired behavior for the closed-loop system. Learning convergence of the proposed OOART-MRTC VI-based algorithm was analyzed under parametric uncertainty and actuation fault. Future research will strive to validate the proposed method on highly complex, nonlinear, and high-order experimental systems.

Funding: This research received no external funding.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADP	Approximate (adaptive) dynamic programming
HDP	Heuristic dynamic programming
DHP	Dual heuristic programming
GDHP	Global DHP
HJB	Hamilton–Jacobi–Bellman
PolIt	Policy Iteration
VI	Value Iteration
NN	Neural network
PG	Policy gGradient
MDP	Markov Decision Process
UAV	Unmanned aerial vehicle
LQR	Linear quadratic regulator
PPO	Proximal Policy Optimization
RL	Reinforcement learning
DRL	Deep RL
OOART-MRTC	Online off-policy asynchronous real-time RL for model reference tracking control

Appendix A

Based on model (1), we write

$$x_{k+2} = A_o x_{k+1} + B_o u_{k+1} = A_o^2 x_k + A_o B_o u_k + B_o u_{k+1}, \quad (A1)$$

and following the recursion, we generally obtain

$$x_{k+r} = A_o^r x_k + \underbrace{[B_o \ A_o B_o \ \dots \ A_o^{r-1} B_o]}_{\mathcal{C}} \begin{bmatrix} u_{k+r-1} \\ \vdots \\ u_{k+1} \\ u_k \end{bmatrix}, \quad (A2)$$

with $\mathcal{C}$ the controllability matrix. We can also write, for $k \leftarrow k - r$, that

$$x_k = A_o^r x_{k-r} + \mathcal{C} \begin{bmatrix} u_{k-1} \\ \vdots \\ u_{k-r+1} \\ u_{k-r} \end{bmatrix}. \quad (A3)$$

Based on the output equation, for a strictly causal system with $D_o = 0$, we find that $y_k = C_o x_k$ and

$$\begin{bmatrix} y_{k+r} \\ \vdots \\ y_{k+3} \\ y_{k+2} \\ y_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} C_o A_o^{r-1} \\ \vdots \\ C_o A_o^2 \\ C_o A_o \\ C_o \end{bmatrix}}_{\tilde{O}} A_o x_k + \underbrace{\begin{bmatrix} C_o B_o & \dots & C_o A_o^{r-2} B_o & C_o A_o^{r-1} B_o \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & C_o B_o & C_o A_o B_o \\ 0 & \dots & 0 & C_o B_o \end{bmatrix}}_{\tilde{N}} \begin{bmatrix} u_{k+r-1} \\ \vdots \\ u_{k+2} \\ u_{k+1} \\ u_k \end{bmatrix}, \quad (A4)$$

based on which, for $k \leftarrow k - r$, we get

$$\begin{bmatrix} y_k \\ \vdots \\ y_{k-r+3} \\ y_{k-r+2} \\ y_{k-r+1} \end{bmatrix} = \underbrace{\begin{bmatrix} C_o A_o^{r-1} \\ \vdots \\ C_o A_o^2 \\ C_o A_o \\ C_o \end{bmatrix}}_{\tilde{O}} A_o x_{k-r} + \underbrace{\begin{bmatrix} C_o B_o & \dots & C_o A_o^{r-2} B_o & C_o A_o^{r-1} B_o \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & C_o B_o & C_o A_o B_o \\ 0 & \dots & 0 & C_o B_o \end{bmatrix}}_{\tilde{N}} \begin{bmatrix} u_{k-1} \\ \vdots \\ u_{k-r+2} \\ u_{k-r+1} \\ u_{k-r} \end{bmatrix}. \quad (A5)$$

Solving for x_{k-r} in the last system of equations, we obtain

$$x_{k-r} = \text{pinv}(\bar{O}A_o)[I|0] \begin{bmatrix} y_{k,k-r+1} \\ y_{k-r} \end{bmatrix} - \text{pinv}(\bar{O}A_o)\bar{N}u_{k-1,k-r}, \quad (\text{A6})$$

where $y_{k,k-r+1} := [y_k^T, \dots, y_{k-r+2}^T, y_{k-r+1}^T]^T$, $u_{k-1,k-r} := [u_{k-1}^T, \dots, u_{k-r+1}^T, u_{k-r}^T]^T$, and $\text{pinv}(\cdot)$ is the pseudo-inverse operator for a matrix. In the last equation, dependence on y_{k-r} was artificially introduced using columns with zero-padding in the term $[I|0]$ found in the left hand-side multiplied matrix. The assumed controllability of system (1) implies both state and output controllability. Then, the first matrix row from $\bar{N}$, which is $[C_o B_o, \dots, C_o A_o^{r-1} B_o]$, is full row rank.

Suppose that $\exists r \geq n$ that makes the controllability matrix $\mathcal{C}$ full column rank and the observability matrix $\bar{O}$ full row rank. Based on x_{k-r} from (A6), replaced in (A3), we recover the present state as

$$x_k = [A_o^r \text{pinv}(\bar{O}A_o)[I|0] \quad \mathcal{C} - A_o^r \text{pinv}(\bar{O}A_o)\bar{N}] \begin{bmatrix} y_{k,k-r} \\ u_{k-1,k-r} \end{bmatrix} = \mathcal{T}v_k, \quad (\text{A7})$$

where $y_{k,k-r} = [y_{k,k-r+1}^T, y_{k-r}^T]^T$, which concludes the proof.

References

1. Bertsekas, D.P.; Tsitsiklis, J.N. Neuro-dynamic programming: An overview. In Proceedings of the 34th IEEE Conference on Decision and Control, New Orleans, LA, USA, 13–15 December 1995; Volume 1, pp. 560–564.
2. Van Roy, B. Neuro-dynamic programming: Overview and recent trends. In *Handbook of Markov Decision Processes*; Springer: Boston, MA, USA, 2002; pp. 431–459.
3. Werbos, P. Approximate dynamic programming for real-time control and neural modeling. In *Handbook of Intelligent Control*; Van Nostrand Reinhold: New York, NY, USA, 1992; pp. 493–525.
4. Si, J.; Barto, A.G.; Powell, W.B.; Wunsch, D. *Handbook of Learning and Approximate Dynamic Programming*; Wiley-IEEE Press: Hoboken, NJ, USA, 2004; Volume 2.
5. Al-Tamimi, A.; Abu-Khalaf, M.; Lewis, F.L. Heuristic dynamic programming nonlinear optimal controller. *Mach. Learn.* **2009**, *3*, 361–380.
6. Ni, Z.; He, H.; Wen, J.; Xu, X. Goal representation heuristic dynamic programming on maze navigation. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *24*, 2038–2050.
7. Wang, F.Y.; Zhang, H.; Liu, D. Adaptive dynamic programming: An introduction. *IEEE Comput. Intell. Mag.* **2009**, *4*, 39–47.
8. Wang, D.; Xin, P.; Zhao, M.; Qiao, J. Intelligent optimal control of constrained nonlinear systems via receding-horizon heuristic dynamic programming. *IEEE Trans. Syst. Man Cybern. Syst.* **2024**, *54*, 287–299.
9. Wang, Z.; Wei, Q.; Liu, D. A novel triggering condition of event-triggered control based on heuristic dynamic programming for discrete-time systems. *Optim. Control Appl. Methods* **2018**, *39*, 1467–1478.
10. Al-Dabooni, S.; Wunsch, D. The boundedness conditions for model-free HDP(λ). *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *30*, 1928–1942.
11. Sardarmehni, T.; Heydari, A. Sub-optimal switching in anti-lock brake systems using approximate dynamic programming. *IET Control Theory Appl.* **2019**, *13*, 1413–1424.
12. Prokhorov, D.V.; Wunsch, D.C. Adaptive critic designs. *IEEE Trans. Neural Netw.* **1997**, *8*, 997–1007.
13. 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.
14. Lewis, F.L.; Vrabie, D. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst. Mag.* **2009**, *9*, 32–50.
15. Wang, B.; Zhao, D.; Alippi, C.; Liu, D. Dual heuristic dynamic programming for nonlinear discrete-time uncertain systems with state delay. *Neurocomputing* **2014**, *134*, 222–229.
16. Wang, Y.; Jiao, X. Dual heuristic dynamic programming based energy management control for hybrid electric vehicles. *Energies* **2022**, *15*, 3235.
17. Stingu, E.; Lewis, F.L. An approximate Dynamic Programming based controller for an underactuated 6DoF quadrotor. In Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), Paris, France, 11–15 April 2011; pp. 271–278.

18. Dydek, Z.T.; Annaswamy, A.M.; Lavretsky, E. Adaptive control of quadrotor UAVs: A design trade study with flight evaluations. *IEEE Trans. Control Syst. Technol.* **2012**, *21*, 1400–1406.
19. Naruta, A.; Mannucci, T.; Van Kampen, E.J. Continuous state and action Q-learning framework applied to quadrotor UAV control. In Proceedings of the AIAA Scitech 2019 Forum, San Diego, CA, USA, 7–11 January 2019; p. 0145.
20. Deshpande, A.M.; Kumar, R.; Minai, A.A.; Kumar, M. Developmental reinforcement learning of control policy of a quadcopter UAV with thrust vectoring rotors. In Proceedings of the ASME 2020 Dynamic Systems and Control Conference, Pittsburgh, PA, USA, 4–7 October 2020; Volume 84287, p. V002T36A011.
21. Zhang, Z.; Chen, T.; Zheng, L.; Luo, Y. A quadratic programming based neural dynamic controller and its application to UAVs for time-varying tasks. *IEEE Trans. Veh. Technol.* **2021**, *70*, 6415–6426.
22. Barzegar, A.; Lee, D.J. Deep reinforcement learning-based adaptive controller for trajectory tracking and altitude control of an aerial robot. *Appl. Sci.* **2022**, *12*, 4764.
23. Zahmatkesh, M.; Emami, S.A.; Banazadeh, A.; Castaldi, P. Robust attitude control of an agile aircraft using improved Q-learning. *Actuators* **2022**, *11*, 374.
24. Din, A.F.U.; Mir, I.; Gul, F.; Akhtar, S. Development of reinforced learning based non-linear controller for unmanned aerial vehicle. *J. Ambient Intell. Humaniz. Comput.* **2023**, *14*, 4005–4022.
25. Annaswamy, A.M.; Guha, A.; Cui, Y.; Tang, S.; Fisher, P.A.; Gaudio, J.E. Integration of adaptive control and reinforcement learning for real-time control and learning. *IEEE Trans. Autom. Control* **2023**, *68*, 7740–7755.
26. Gu, S.; Kumar, R. Robust Optimal Safe and Stability Guaranteeing Reinforcement Learning Control for Quadcopter. *arXiv* **2024**, arXiv:2412.14003.
27. Moin, H.; Shah, U.H.; Khan, M.J.; Sajid, H. Fine-Tuning Quadcopter Control Parameters via Deep Actor-Critic Learning Framework: An Exploration of Nonlinear Stability Analysis and Intelligent Gain Tuning. *IEEE Access* **2024**, *12*, 173462–173474.
28. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; Volume 48, pp. 1928–1937.
29. Ramstedt, S.; Pal, C. Real-time reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 3073–3082.
30. Yuan, Y.; Mahmood, A.R. Asynchronous Reinforcement Learning for Real-Time Control of Physical Robots. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 5546–5552.
31. Abouheaf, M.; Boase, D.; Gueaieb, W.; Spinello, D.; Al-Sharhan, S. Real-time measurement-driven reinforcement learning control approach for uncertain nonlinear systems. *Eng. Appl. Artif. Intell.* **2023**, *122*, 106029.
32. Lemmel, J.; Grosu, R. Real-Time Recurrent Reinforcement Learning. *arXiv* **2023**, arXiv:2311.04830.
33. Reuer, K.; Landgraf, J.; Foesel, T.; O’Sullivan, J.; Beltrán, L.; Akin, A.; Norris, G.J.; Remm, A.; Kerschbaum, M.; Besse, J.C.; et al. Realizing a deep reinforcement learning agent for real-time quantum feedback. *Nat. Commun.* **2023**, *14*, 7138.
34. Radac, M.B.; Borlea, A.I. Virtual state feedback reference tuning and value iteration reinforcement learning for unknown observable systems control. *Energies* **2021**, *14*, 1006.
35. D’Amico, W.; La Bella, A.; Farina, M. An Incremental Input-to-State Stability Condition for a Class of Recurrent Neural Networks. *IEEE Trans. Autom. Control* **2023**, *69*, 2221–2236.
36. Radac, M.-B. Trajectory Tracking within a Hierarchical Primitive-Based Learning Approach. *Entropy* **2022**, *24*, 889.
37. Radac, M.B.; Chirla, D.P. Near real-time online reinforcement learning with synchronous or asynchronous updates. *Sci. Rep.* **2025**, *15*, 17158.
38. Radac, M.B.; Borsa, G.; Alexa, L.A. A soft real-time ROS2-based energy management system under asynchronous messaging and different node update rates. *Energy Convers. Manag. X* **2025**, *25*, 101272.
39. Liu, Y.; Liu, F.; Huang, R. Supervised optimal control in complex continuous systems with trajectory imitation and reinforcement learning. *Sci. Rep.* **2025**, *15*, 19479.
40. Dai, X.; Chen, R.; Guan, S.; Li, W.T.; Yuen, C. BuildingGym: An open-source toolbox for AI-based building energy management using reinforcement learning. *Build. Simul.* **2025**, *18*, 1909–1927.

41. Zhao, M.; Wang, D.; Li, M.; Gao, N.; Qiao, J. A new Q-function structure for model-free adaptive optimal tracking control with asymmetric constrained inputs. *Int. J. Adapt. Control Signal Process.* **2024**, *38*, 1561–1578.
42. Ayadi, W.; Alkhazraji, E.; Khaled, H.; Bouteraa, Y.; Abedini, M.; Mohammadzadeh, A. Adaptive heartbeat regulation using double deep reinforcement learning in a Markov decision process framework. *Sci. Rep.* **2025**, *15*, 35347.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.