mathematics

MDPI

Article

Hierarchical Cognitive Control for Unknown Dynamic Systems Tracking

Mircea-Bogdan Radac * and Timotei Lala

Department of Automation and Applied Informatics, Politehnica University of Timisoara, 300223 Timisoara, Romania; timotei.lala@student.upt.ro
* Correspondence: mircea.radac@upt.ro

Abstract: A general control system tracking learning framework is proposed, by which an optimal learned tracking behavior called 'primitive' is extrapolated to new unseen trajectories without requiring relearning. This is considered intelligent behavior and strongly related to the neuro-motor cognitive control of biological (human-like) systems that deliver suboptimal executions for tasks outside of their current knowledge base, by using previously memorized experience. However, biological systems do not solve explicit mathematical equations for solving learning and prediction tasks. This stimulates the proposed hierarchical cognitive-like learning framework, based on state-of-the-art model-free control: (1) at the low-level L1, an approximated iterative Value Iteration for linearizing the closed-loop system (CLS) behavior by a linear reference model output tracking is first employed; (2) an experiment-driven Iterative Learning Control (EDILC) applied to the CLS from the reference input to the controlled output learns simple tracking tasks called 'primitives' in the secondary L2 level, and (3) the tertiary level L3 extrapolates the primitives' optimal tracking behavior to new tracking tasks without trial-based relearning. The learning framework relies only on input-output system data to build a virtual state space representation of the underlying controlled system that is assumed to be observable. It has been shown to be effective by experimental validation on a representative, coupled, nonlinear, multivariable real-world system. Able to cope with new unseen scenarios in an optimal fashion, the hierarchical learning framework is an advance toward cognitive control systems.

Keywords: hierarchical control; reinforcement learning and approximate dynamic programming; iterative learning control; primitives; unknown dynamics; input-output observable system

Citation: Radac, M.-B.; Lala, T. Hierarchical Cognitive Control for Unknown Dynamic Systems Tracking. *Mathematics* **2021**, *9*, 2752. https://doi.org/10.3390/math9212752

Academic Editor: Elias August

Received: 28 September 2021
Accepted: 26 October 2021
Published: 29 October 2021

1. Introduction

Cognitive control systems are characterized by perception, learning, planning and memorization, defining a clear path toward general intelligent systems that are able to optimally handle new, unseen before situations and display adaptability. The best examples of collections of such complex control systems are offered by certain biological systems (living organisms, humans in particular) who intelligently solve complex tasks by combining knowledge gathered through experience via learning and planning mechanisms that are encoded in the brain. As such, the brain acts as a processing unit that is able, at a higher level, to project future strategies in order to achieve goals through reasoning and planning, decompose complex strategies into well-known possibly simpler strategies (planning), memorize new experiences to augment the current knowledge base, and process feedback signals by fusing visual, tactile, auditory, and olfactory information for task execution improvement through learning. Finally, it guides the lower-level neuromuscular control system to act in the environment to achieve higher-level goals.

This intelligence of living organisms has developed without solving explicit mathematical equations; therefore, this kind of complex intelligence serves as inspiration for developing model-free control, learning, and planning techniques. One suitable paradigm

for explaining the above techniques is the process of learning neuromotor skills, which rely on well-acknowledged control formulations such as those found in feed-forward- and feedback-based control and predictive-based control. A common trait of all these techniques is that they are almost always posed and solved in optimization-based settings [1]. In such settings (e.g., taking error-based learning), the error gradient is exploited to improve the control solutions. Alternatively, use-dependent learning addresses the permanent compromise between repeatable tasks whose improved execution is improvable for high-performance and non-repeatable tasks that require a significant relearning effort.

One illustrative natural behavior for some of the above concepts is the imitating behavior of biological systems: the first execution of an imitation task is nearly optimal. The brain can analyze, memorize, and decompose the imitation task into subtasks for each limb and plan for corresponding immediate future tracking tasks. The control of each limb is already encoded in the neuromuscular system through an inverse dynamical model that is not explicitly represented. Another illustrative situation occurs when new neuromotor tasks are solved by first decomposing them into tasks with known solutions and then recomposing the new solution from existing ones. The previously described behavior unites several concepts that can be formally called a primitive-based learning framework [1].

Primitive-based control has been researched for at least two decades, in various forms, by transforming the time scale [2,3], concatenation-based mechanisms [4–6], and decomposition/re-composition in time [7,8]. More recently, primitive-based control has been studied in [9–13], mostly as part of hierarchical learning frameworks [14–18]. However, the application of the Iterative Learning Control (ILC, a full list of the acronyms used in the paper is presented in abbreviations part) technique [19–25] over linearized feedback closed-loop control systems (CLSs) as a primer mechanism for primitive-based learning was proposed in [8]. An experiment-driven ILC (EDILC) variant was crafted in a norm-optimal tracking setting to learn reference-input controlled output pairs called primitives. Such a primitive tuple contains a reference input to the CLS (called the input primitive) coupled with its paired control system output (called the output primitive). Subsequently, the output primitive is usually shaped as a Gaussian function, but it can be any shape suitable for approximation purposes. Delayed copies of the output primitives are used to approximate a new trajectory that must be tracked. The coefficients that combine the output primitives linearly to approximate the new trajectory are used to combine the delayed and copied input primitives to obtain the optimal reference input. For the assumed linear CLS, this reconstructed reference input is optimal with respect to the criterion by which the original primitives are learned; therefore, when set as input to the CLS, it leads to theoretically perfect new trajectory tracking. It does this without repetitive-based relearning using the EDILC.

The primitive-based approach is sensible to the CLS's linearity assumption; therefore, a control that makes the CLS as linear as possible has to be learned. One framework to ensure this linearization is the approximate (adaptive) dynamic programming (ADP) [26–30], also known as reinforcement learning (RL) [31–35], designed for the output reference model (ORM) paradigm. Specifically, a model-free iterative Value Iteration (IVI) algorithm as a representative ADP/RL approach can serve this goal by using general function approximators such as neural networks (NNs). Hence, it is called the approximated IVI (AIVI).

With a linear ORM's output well tracked by the CLS output, an approximate linearized CLS from the reference input to the controlled output is ensured in a wide range, together with some inherent disturbance attenuation ability. Two major problems with ADP/RL are exploration and state knowledge [36–40]. The former is still an open issue, although much research is underway. Enhanced and accelerated exploration is achievable with pre-stabilizing controllers, which can also be designed based on the unknown dynamics assumption principles. The state availability was solved with virtual state-space models constructed from input-output (I/O) samples of general unknown but assumed observable nonlinear systems [41,42]. Using historical I/O samples from the system as an alternate sys-

tem, state representation has been used in more complex but unformalized environments, such as video games [43].

This work aims to integrate the learning control techniques and machine learning techniques previously presented in a hierarchical cognitive-based control framework tailored to extrapolate optimally learned tracking control to novel tracking tasks. Tracking should be achievable without relearning through trials by exploiting the already learned optimal tracking skills. It is obtained through a hierarchical learning architecture where (a) the closed-loop system at a lower level (called L1 level herein) is first linearized using the ORM principle. This is achieved through nonlinear state-feedback control, where a virtual state is built from finite sequences of I/O system data samples. Learning such a controller is performed using AIVI; (b) an EDILC is used at the secondary upper L2 level to learn the optimal tracking skills called primitives, by repetitions; (c) the tertiary and last L3 level is dedicated to extrapolating the optimally learned tracking skills, without repetitions, to new tracking tasks.

Progress and contributions with respect to work [8] are:

- At level L1, CLS linearization is strongly ensured using a virtual state-feedback neuro-controller learned by AIVI. The new state-space representation relies on a virtual transformation empowered by observability theory.
- At level L2, the model-free EDILC dedicated to primitive learning is improved in several aspects: (1) the gradient information is extracted using a single gradient experiment in the vicinity of the current iteration nominal trajectory, irrespective of the CLS's number of controlled outputs; (2) EDILC monotonic convergence is derived by optimally selecting the learning rate parameter in order to trade-off learning speed and robustness. This is achieved using two approximate CLS models for a double-safe mechanism; (3) proper initialization of the gradient-based search specific to EDILC results in fewer iterations for convergence.
- The primitive-based mechanism at level L3, for optimally predicting the reference input ensuring theoretical perfect tracking of a previously unseen desired trajectory, is designed to: (1) handle desired trajectories longer than the learned primitives, and (2) indirectly handle constraints on the CLS's output.

The following bottom-up presentation of the proposed hierarchical learning framework results in an effective validation of a hybrid software-electrical system that is coupled, multivariable, and has a delay. The case study is representative of many mechatronic systems.

2. The Linear ORM Tracking for Observable Systems with Unknown Dynamics

A common and general representation for a deterministic controlled system model in a discrete-time state space is

$$\begin{cases} \boldsymbol{s}_{k+1}^{ext} = \boldsymbol{E}\left(\boldsymbol{s}_{k}^{ext}, \boldsymbol{a}_k\right), \\ \boldsymbol{y}_k = \boldsymbol{v}_{k,1},\ \boldsymbol{y}_k^{ORM} = \boldsymbol{L}\boldsymbol{s}_k^{ORM}, \end{cases} \tag{1}$$

where $\boldsymbol{E}()$ is a partially known nonlinear state map, k is the sample index, $\boldsymbol{a}_k = [a_{k,1}, \ldots, a_{k,m_u}]^T \in \Omega_A \subset \Re^{m_u}$ lumps the m_u control action inputs of the system within a known domain Ω_A. A number of p measurable system outputs are formally collected in the partial output vector $\boldsymbol{y}_k = [y_{k,1}, \ldots, y_{k,p}]^T \in \Omega_Y \subset \Re^p$ with known domain Ω_Y. Another p measurable system outputs formally complete the remaining output as $\boldsymbol{y}_k^{ORM} = [y_{k,1}^{ORM}, \ldots, y_{k,p}^{ORM}]^T \in \Omega_{Y_{ORM}} \subset \Re^p$ with known domain $\Omega_{Y_{ORM}}$, to be contextualized later on. The state $\boldsymbol{s}_k^{ext} = \left[\boldsymbol{v}_k^T, \left(\boldsymbol{s}_k^{ORM}\right)^T, \boldsymbol{\rho}_k^T\right]^T$ lumps three types of state variables stemming from three independent systems. First, a virtual state-space system is defined as follows:

$$\begin{cases} \boldsymbol{v}_{k+1} = \boldsymbol{F}(\boldsymbol{v}_k, \boldsymbol{a}_k), \\ \boldsymbol{y}_k = \boldsymbol{v}_{k,1}, \end{cases} \tag{2}$$

which is a transformation of the true state–space system

$$\begin{cases} \boldsymbol{s}_{k+1} = \boldsymbol{f}(\boldsymbol{s}_k, \boldsymbol{a}_k), \\ \boldsymbol{y}_k = \boldsymbol{g}(\boldsymbol{s}_k), \end{cases} \tag{3}$$

where $\boldsymbol{s}_k = [s_{k,1} \dots s_{k,n}]^T \in \Omega_S \subset \Re^n$ gathering the n (which is unknown) true system states that cannot be measured, $\boldsymbol{a}_k$ are the control action inputs of the true system, and the measurable true system outputs are $\boldsymbol{y}_k$. The system functions as nonlinear maps $\boldsymbol{f}(\boldsymbol{s}_k, \boldsymbol{a}_k) : \Omega_S \times \Omega_A \to \Omega_S$, $\boldsymbol{g}(\boldsymbol{s}_k) : \Omega_S \to \Omega_Y$ are unknown and are assumed to be continuously differentiable. Some assumptions regarding (1) were introduced. The pair $(\boldsymbol{f}, \boldsymbol{g})$ is entirely state observable, meaning that the true state $\boldsymbol{s}_k$ from (3) is recoverable from the present and past I/O measurements of $\boldsymbol{a}_k, \boldsymbol{y}_k$. The observability property is used in the sense defined for linear systems. Further, (3) is assumed to be controllable, and the system's relative degree (delay included) is known and constant.

With unknown system functions $(\boldsymbol{f}, \boldsymbol{g})$, observability and controllability can only be inferred from the working experience with the system, from technical manuals, or from the literature. The system's relative degree and its non-minimum phase (NMP) type are recognizable from historical I/O response data.

Relying on the system's (3) complete observability, using *Lemma* 1 from [41] shows how to build transformation (2) whose virtual state vector is defined in terms of I/O samples of (3): $\boldsymbol{v}_k = [(\boldsymbol{Y}_{\overline{k,k-\tau}})^T, (\boldsymbol{A}_{\overline{k-1,k-\tau}})^T]^T \triangleq [(\boldsymbol{v}_{k,1})^T, (\boldsymbol{v}_{k,2})^T, \dots, (\boldsymbol{v}_{k,2\tau+1})^T]^T \in \underbrace{\Omega_Y \times \dots \times \Omega_Y}_{\tau+1 \text{ times}} \times \underbrace{\Omega_A \times \dots \times \Omega_A}_{\tau \text{ times}} \triangleq \Omega_v \subset \Re^{p(\tau+1)+m_u\tau}$. For a more detailed element-wise correspondence, the indexing $\boldsymbol{Y}_{\overline{k,k-\tau}} = [(\boldsymbol{y}_k)^T \dots (\boldsymbol{y}_{k-\tau})^T]^T \triangleq [(\boldsymbol{v}_{k,1})^T \dots, (\boldsymbol{v}_{k,\tau+1})^T]^T$, $\boldsymbol{A}_{\overline{k-1,k-\tau}} = [(\boldsymbol{a}_{k-1})^T \dots (\boldsymbol{a}_{k-\tau})^T]^T \triangleq [(\boldsymbol{v}_{k,\tau+2})^T \dots (\boldsymbol{v}_{k,2\tau+1})^T]^T$ is utilized. Remarkably, knowing Ω_Y, Ω_A, makes Ω_v known. The $\tau \in \mathbb{N}$ ($\mathbb{N}$ is the set of positive integers) indexes the historical I/O measurements, and its value is related to the order of (3); therefore, it is also correlated with the unknown observability index of (3).

Definition 1. *The unknown observability index of (3) is the minimal value τ_{min} of τ for which state $\boldsymbol{s}_k$ is fully recoverable from the I/O measurements $\boldsymbol{Y}_{\overline{k,k-\tau}}$, $\boldsymbol{A}_{\overline{k-1,k-\tau}}$. The role of this index is similar to that of linear systems.*

Under the observability assumption, there exists a minimal value $\tau = \tau_{min}$ which makes $\boldsymbol{s}_k$ fully recovered from I/O data and uniquely associated with $\boldsymbol{v}_k$ which is a sort of alias for $\boldsymbol{s}_k$, but in a different dimensional space [41]. For $\tau > \tau_{min}$, the size increase of $\boldsymbol{v}_k$ does not add valuable information about $\boldsymbol{s}_k$. In practice, τ should be as large as possible without negatively affecting the computational complexity of the subsequent learning of state-feedback control based on high-dimensional $\boldsymbol{v}_k$.

Based on transformation (2), system (2) is also I/O controllable, having the same I/O as (1). The main feature of (2) is its complete state availability, making it fully state observable but without entirely known dynamics because the map $\boldsymbol{F}(.)$ is partially unknown. Input delay (dead-time) systems also hold transforms such as (2) to render them fully state observable.

The second component of $\boldsymbol{s}_k^{ext}$ is the state $\boldsymbol{s}_k^{ORM}$ of the ORM model which will be matched by the CLS by proper control design; let this known ORM state-space model be

$$\begin{cases} \boldsymbol{s}_{k+1}^{ORM} = \boldsymbol{G}\boldsymbol{s}_k^{ORM} + \boldsymbol{H}\boldsymbol{\rho}_k, \\ \boldsymbol{y}_k^{ORM} = \boldsymbol{L}\boldsymbol{s}_k^{ORM}, \end{cases} \tag{4}$$

where $\boldsymbol{s}_k^{ORM} = [s_{k,1}^{ORM}, \dots, s_{k,n_m}^{ORM}]^T \in \Omega_{S_{ORM}} \subset \Re^{n_m}$ gathers the n_m ORM states in known domain $\Omega_{S_{ORM}}$, $\boldsymbol{\rho}_k = [\rho_{k,1}, \dots, \rho_{k,p}]^T \in \Omega_{\boldsymbol{\rho}} \subset \Re^p$ lumps the ORM's inputs in their known

domain $\Omega_{\boldsymbol{\rho}}$; $\boldsymbol{\rho}_k$ will also serve as CLS input. The ORM outputs are collected in $\boldsymbol{y}_k^{ORM}$ introduced in the first place. Note that $\boldsymbol{\rho}_k$ completes the definition of $\boldsymbol{s}_k^{ext}$ in (1).

The tuple $(\boldsymbol{G}, \boldsymbol{H}, \boldsymbol{L})$ characterizes the linear ORM for which an I/O pulse transfer matrix (t.m.) is $\boldsymbol{y}_k^{ORM} = \boldsymbol{T}_{ORM}(q)\boldsymbol{\rho}_k$ (q is the unit time step advance operator). Information about (3), such as relative degree, non-minimum phase character, and bandwidth, must be synthesized within the ORM (4), as dictated by classical control rules. Commonly, this selection is better reflected in a $\boldsymbol{T}_{ORM}(q)$, as it is more straightforward and easier to interpret (bandwidth, overshoot).

To fulfill the Markovian property of (1), the exogenous reference input signal $\boldsymbol{\rho}_k$ is described by the known generation model dynamics $\boldsymbol{\rho}_{k+1} = \Gamma(\boldsymbol{\rho}_k)$, where $\boldsymbol{\rho}_k$ acts as a measurable state variable. Note that the extended model (1) has two outputs stemming from (3) and (4). In summary, the dynamics $\boldsymbol{E}(.)$ will be partially unknown owing to the partially unknown dynamics of its component $\boldsymbol{F}(.)$. The main feature of (1) is its full state observability, that is, the state is fully measurable, and the domain of $\boldsymbol{s}_k^{ext}$ is known as $\Omega_{S_{ext}} = \Omega_v \times \Omega_{S_{ORM}} \times \Omega_{\boldsymbol{\rho}}$.

The ORM tracking problem finally poses as the optimal control search over an infinite-horizon cost V_{ORM}^{∞}:

$$\begin{gathered} \boldsymbol{a}_k^* = \arg \min_{\boldsymbol{a}_k} V_{ORM}^{\infty}(\boldsymbol{a}_k),\ V_{ORM}^{\infty}(\boldsymbol{a}_k) = \sum_{k=0}^{\infty} \parallel \boldsymbol{y}_k(\boldsymbol{a}_k) - \boldsymbol{y}_k^{ORM} \parallel_2^2, \\ s.t.\ dynamics\ (1). \end{gathered} \tag{5}$$

The norm $\parallel . \parallel_2$ defined as the vector-wise Euclidean distance, penalizes at each sampling instant the CLS's output deviation from the ORM's output. Problem (5) is a type of imitation (or apprentice) learning with the ORM (3) acting as a teacher (or expert or supervisor) and with system (1) trying to mimic (or follow) the ORM dynamics. Assuming that (5) is solvable in the following, ADP/RL techniques are well suited for finding closed-form optimal control laws of the type $\boldsymbol{a}_k = \boldsymbol{C}\left(\boldsymbol{s}_k^{ext}\right)$ with the control action input as a nonlinear mapping of the extended state. An AIVI is in fact employed to solve (5) as a competitive alternative to the recently proposed Virtual State-Feedback Reference Tuning (VSFRT) [42]. This is considered the L1 level control and is presented next.

3. The L1 Level—ORM Tracking AIVI-Based Solution

AIVI is a type of model-free offline batch reinforcement Q-learning that is used to find the optimal control solution to (5). AIVI most often uses NNs as function approximators because of their easily customizable architecture, training software availability, and generalization capacity. As such, AIVI uses one NN to parameterize the nonlinear control action as $\boldsymbol{a}_k = \boldsymbol{C}\left(\boldsymbol{s}_k^{ext}, \boldsymbol{\psi}\right)$ and another NN to parameterize the well-known Q-function as $Q\left(\boldsymbol{s}_k^{ext}, \boldsymbol{a}_k, \boldsymbol{\pi}\right)$. Here, $\boldsymbol{\psi}$ and $\boldsymbol{\pi}$ formally gather all the NN tunable weights corresponding to some generic NN architectures. Starting from some initial solutions $\boldsymbol{\psi}^0$ and $\boldsymbol{\pi}^0$, two AIVI-specific steps are called iteratively, and in this particular order, the Q-function update and controller improvement. The steps are formally expressed as

$$\boldsymbol{\pi}^j = \arg \min_{\boldsymbol{\pi}} \frac{1}{B} \sum_{i=1}^{B} \left[Q\left(\boldsymbol{s}_k^{ext[i]}, \boldsymbol{u}_k^{[i]}, \boldsymbol{\pi}\right) - r\left(\boldsymbol{s}_k^{ext[i]}, \boldsymbol{a}_k^{[i]}\right) - Q\left(\boldsymbol{s}_{k+1}^{ext[i]}, \boldsymbol{C}\left(\boldsymbol{s}_{k+1}^{ext[i]}, \boldsymbol{\psi}^{j-1}\right), \boldsymbol{\pi}^{j-1}\right)\right]^2, \tag{6}$$

$$\boldsymbol{\psi}^j = \arg \min_{\boldsymbol{\psi}} \frac{1}{B} \sum_{i=1}^{B} Q\left(\boldsymbol{s}_k^{ext[i]}, \boldsymbol{C}\left(\boldsymbol{s}_k^{ext[i]}, \boldsymbol{\psi}\right), \boldsymbol{\pi}^j\right), \tag{7}$$

respectively, with penalty $r\left(\boldsymbol{s}_k^{ext[i]}, \boldsymbol{a}_k^{[i]}\right) = \boldsymbol{y}_k(\boldsymbol{a}_k) - \parallel \boldsymbol{y}_k^{ORM} \parallel_2^2$, over a batch consisting of B transition samples (also known tuples or experiences) of the form $\left\{\left(\boldsymbol{s}_k^{ext[i]}, \boldsymbol{a}_k^{[i]}, \boldsymbol{s}_{k+1}^{ext[i]}\right)\right\}$, $i = \overline{1, B}$. Here, j indexes the AIVI iteration. It is easily shown that (6) and (7) are equivalent to the NN training of the Q-function and controller, respectively. The transition samples set is collectable from the controlled system (5) under any sufficiently exploratory (i.e., it visits many state-action combinations which uniformly span their space) controller. Even

a random control strategy may be suitable for this purpose; however, a priori stabilizing controllers accelerate the exploration process: faster visiting more state-action combinations while simultaneously dealing with operational (e.g., safety) constraints. This off-policy offline is more data efficient than online on-policy reinforcement learning, where the exploration-exploitation dilemma throughout learning, combined with the lower data efficiency, is a major challenge. When all (or a random subset) of the transition samples are used in each AIVI step, the approach ensures the so-called experience replay.

The AIVI steps converge to the optimal virtual state feedback controller, ensuring that the CLS tracks the ORM in terms of reference input-controlled output at the L1 level [31,32,40,41]. It is assumed that the CLS is indirectly linearized in the ORM tracking sense. Thus, it is prepared to receive the L2 level learning phase, next presented.

4. The L2 Level—EDILC for Learning Primitives

The L2 level aims to learn pairs of reference inputs and CLS outputs called primitive pairs or simply primitives, where the CLS outputs have a basis function shape (e.g., Gaussian), to be used later for function approximation. To this end, we search for $\boldsymbol{\rho}_k$ which makes $\boldsymbol{y}_k$ track a desired trajectory $\boldsymbol{y}_k^d$. The CLS is assumed to be reset with each ILC iteration to zero initial conditions. The CLS response to non-zero (but constant) initial conditions and repeated disturbances is easily absorbed into $\boldsymbol{y}_k^d$. At the given ILC iteration j, the CLS reference input-controlled output relationship is written in lifted (or supervector) notation over an N-length experiment, as in

$$\boldsymbol{Y}^j = \boldsymbol{T}\boldsymbol{P}^j, \tag{8}$$

where the reference input at iteration j is $\boldsymbol{P}^j = [(\boldsymbol{P}_1^j)^T (\boldsymbol{P}_2^j)^T \dots (\boldsymbol{P}_p^j)^T]^T \in \Re^{pN\times 1}$, $\boldsymbol{P}_l^j = [\rho_{0,l}^j, \rho_{1,l}^j, \dots, \rho_{N-1,l}^j]^T \in \Re^{N\times 1}$, $l = \overline{1,p}$, the CLS output at iteration j is $\boldsymbol{Y}^j = [(\boldsymbol{Y}_1^j)^T (\boldsymbol{Y}_2^j)^T \dots (\boldsymbol{Y}_p^j)^T]^T \in \Re^{pN\times 1}$, $\boldsymbol{Y}_l^j = [y_{0,l}^j, y_{1,l}^j, \dots, y_{N-1,l}^j]^T \in \Re^{N\times 1}$, $l = \overline{1,p}$. The impulse response terms of the pulse transfer matrix (t.m.) operator $\boldsymbol{T}(q)$ characterizing the CLS are used to build the matrix $\boldsymbol{T}$. To see how $\boldsymbol{T}$ emerges, the $p \times p$ causal CLS at the current iteration j is (in time notation)

$$\begin{bmatrix} y_{k,1}^j \\ y_{k,2}^j \\ \dots \\ y_{k,p}^j \end{bmatrix} = \underbrace{\begin{bmatrix} T^{1,1}(q)T^{2,1}(q)\dots T^{p,1}(q) \\ T^{1,2}(q)T^{2,2}(q)\dots T^{p,2}(q) \\ \dots \\ T^{1,p}(q)T^{2,p}(q)\dots T^{p,p}(q) \end{bmatrix}}_{\boldsymbol{T}(q)} \begin{bmatrix} \rho_{k,1}^j \\ \rho_{k,2}^j \\ \dots \\ \rho_{k,p}^j \end{bmatrix}, \tag{9}$$

where $j = 0, 1, 2, \dots$ is the iteration index, $k = 0, 1, \dots$ is the discrete-time index, $y_{k,l}^j$ is the l^{th} output at time k in iteration j, $\rho_{k,l}^j$ is the l^{th} input at time k in iteration j, and $T^{i,l}(q)$ are the pulse transfer functions (t.f.) from input i to output l. The t.m. $\boldsymbol{T}(q)$ encompasses the feedback CLS with some kind of a multiple input multiple output (MIMO) (possibly nonlinear) feedback controller over a MIMO (possibly nonlinear) controlled system. The feedback CLS is square; that is, each reference input drives one controlled output. In general, $\boldsymbol{T}(q)$ was coupled.

With each rational t.f. $T^{i,l}(q)$ admitting an infinite $T^{i,l}(q) = \cdots + t^{i,l}_{-2}q^2 + t^{i,l}_{-1}q + t^{i,l}_0 + t^{i,l}_1 q^{-1} + t^{i,l}_2 q^{-2} + \cdots$ power series expansion, a truncated lifted notation of length N for the ILC system is

$$\begin{bmatrix} y^j_{0,1} \\ y^j_{1,1} \\ \vdots \\ y^j_{N-1,1} \\ \hline \vdots \\ \hline y^j_{0,p} \\ y^j_{1,p} \\ \vdots \\ y^j_{N-1,p} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{T}^{1,1}\boldsymbol{T}^{2,1}\ldots\boldsymbol{T}^{p,1} \\ \hline \vdots \\ \hline \boldsymbol{T}^{1,p}\boldsymbol{T}^{2,p}\ldots\boldsymbol{T}^{p,p} \end{bmatrix}}_{\boldsymbol{T}} \begin{bmatrix} \rho^j_{0,1} \\ \rho^j_{1,1} \\ \vdots \\ \rho^j_{N-1,1} \\ \hline \vdots \\ \hline \rho^j_{0,p} \\ \rho^j_{1,p} \\ \vdots \\ \rho^j_{N-1,p} \end{bmatrix} \equiv \boldsymbol{Y}^j = \boldsymbol{T}\boldsymbol{P}^j, \tag{10}$$

where each $\boldsymbol{T}^{i,l} = \begin{bmatrix} t^{i,l}_0 & t^{i,l}_{-1} & \cdots & t^{i,l}_{1-N} \\ t^{i,l}_1 & t^{i,l}_0 & \cdots & t^{i,l}_{2-N} \\ \vdots & \vdots & \ddots & \vdots \\ t^{i,l}_{N-1} & t^{i,l}_{N-2} & \cdots & t^{i,l}_0 \end{bmatrix} \in \Re^{N\times N}$ is built from the impulse response coefficients of $T^{i,l}(q)$ and all the $\boldsymbol{T}^{i,l}$ build the $\boldsymbol{T} \in \Re^{pN\times pN}$ (note the difference between $\boldsymbol{T}$ and $\boldsymbol{T}(q)$).

Let the desired output trajectory be $\boldsymbol{Y}^d = [(\boldsymbol{Y}^d_1)^T\ (\boldsymbol{Y}^d_2)^T\ \ldots\ (\boldsymbol{Y}^d_p)^T]^T \in \Re^{pN\times 1}$, $\boldsymbol{Y}^d_l = [y^d_{0,l},\ y^d_{1,l},\ \ldots,\ y^d_{N-1,l}]^T \in \Re^{N\times 1}$, $l = \overline{1,p}$. The learning goal is formally expressed as an optimization problem without constraints as:

$$\boldsymbol{P}^* = \arg\min_{\boldsymbol{P}} J(\boldsymbol{P}),\ J(\boldsymbol{P}) = \frac{1}{N} \parallel \boldsymbol{E}(\boldsymbol{P}) \parallel_2^2 = \frac{1}{N} \parallel \boldsymbol{T}\boldsymbol{P} - \boldsymbol{Y}^d \parallel_2^2 + \frac{\lambda}{N} \parallel \boldsymbol{P} \parallel_2^2, \tag{11}$$

where the tracking error in the current iteration is $\boldsymbol{E}^j = \boldsymbol{Y}^d - \boldsymbol{T}\boldsymbol{P}^j$. The non-zero regularization coefficient λ may be useful in cases where, for example, $\boldsymbol{Y}^d$ does not acknowledge a possible NMP $\boldsymbol{T}(q)$. For simplicity, no regularization is used next ($\lambda = 0$). Directly working with $\boldsymbol{T}$ in practice raises several numerical difficulties: ill-conditioning, large size, and its identification is costly. A convenient numerical solution to (11) is $\boldsymbol{P}^* = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{Y}^d$. However, without using $\boldsymbol{T}$ explicitly, (11) is solved based on an ILC-specific iterative update law as a gradient descent search starting from an initial $\boldsymbol{P}^0$:

$$\begin{bmatrix} \boldsymbol{P}^{j+1}_1 \\ \boldsymbol{P}^{j+1}_2 \\ \cdots \\ \boldsymbol{P}^{j+1}_p \end{bmatrix} = \begin{bmatrix} \boldsymbol{P}^j_1 \\ \boldsymbol{P}^j_2 \\ \cdots \\ \boldsymbol{P}^j_p \end{bmatrix} - \boldsymbol{\xi} \begin{bmatrix} \frac{\partial J(\boldsymbol{P})}{\partial \boldsymbol{P}_1} \\ \vdots \\ \frac{\partial J(\boldsymbol{P})}{\partial \boldsymbol{P}_p} \end{bmatrix}\Bigg|_{\boldsymbol{P}=\boldsymbol{P}^j} = \begin{bmatrix} \boldsymbol{P}^j_1 \\ \boldsymbol{P}^j_2 \\ \cdots \\ \boldsymbol{P}^j_p \end{bmatrix} - \boldsymbol{\xi}\frac{\partial J\left(\boldsymbol{P}^j\right)}{\partial \boldsymbol{P}}, \tag{12}$$

with a positive-definite matrix $\boldsymbol{\xi} = diag(\xi_i),\ i = \overline{1,p}$ is a user-selectable learning rate. The gradient in (12) computed according to the cost in (11) is $\frac{2}{N}\cdot\boldsymbol{T}^T\boldsymbol{E}^j$ and is based on an unknown $\boldsymbol{T}$. The next EDILC Algorithm 1 extracts this gradient information experimentally, and its steps are:

Algorithm 1

Step 0. Initialize $\boldsymbol{P}^0 = \boldsymbol{Y}^d$, then for each $j = \overline{0, \epsilon}$ call the subsequent steps.

Step 1. In a nominal experiment, with the current iteration reference input $\boldsymbol{P}^j$, measure $\boldsymbol{E}^j$. Flip $\boldsymbol{E}^j$ to obtain $udf(\boldsymbol{E}^j)$: the upside-down flipped version of $\boldsymbol{E}^j$.

Step 2. In a so-called gradient experiment, use $\mu \cdot udf(\boldsymbol{E}^j)$ (with scaling factor μ) as an additive disturbance on the current iteration $\boldsymbol{P}^j$. With the disturbed reference input, the CLS output $\boldsymbol{Y}_G^j$. It is valid that $\boldsymbol{Y}_G^j = \boldsymbol{T}\left(\boldsymbol{P}^j + \mu \cdot udf\left(\boldsymbol{E}^j\right)\right) = \boldsymbol{Y}^j + \mu \boldsymbol{T} udf\left(\boldsymbol{E}^j\right)$. Irrespective of the number of control channels p, a single gradient experiment was performed.

Step 3. Calculate $\frac{\partial J(\boldsymbol{P}^j)}{\partial \boldsymbol{P}} = \frac{2}{N} \cdot udf\left(\frac{\boldsymbol{Y}_G^j - \boldsymbol{Y}^j}{\mu}\right)$, since $\boldsymbol{T}^T \boldsymbol{E}^j = udf\left(\boldsymbol{T} udf\left(\boldsymbol{E}^j\right)\right)$ provably holds.

Step 4. Call (8) to update $\boldsymbol{P}^j$.

Step 5. Go to Step 1 if $j < \epsilon$, otherwise exit with $\boldsymbol{P}^\epsilon$.

The output of the model-free EDILC algorithm is the primitive pair $\left\{\boldsymbol{P}^\epsilon \triangleq \widetilde{\boldsymbol{P}}^*, \boldsymbol{Y}\right\}$ where the *input primitive is* $\widetilde{\boldsymbol{P}}^*$, *the output primitive* is Y. Several such pairs can be learned, and they can be indexed $\left\{\widetilde{\boldsymbol{P}}^{*[\delta]}, \boldsymbol{Y}^{[\delta]}\right\}$. Note that $\boldsymbol{Y}$ is used instead of $\boldsymbol{Y}^d$ in the primitive because of the following reasons: after the algorithm, $\boldsymbol{Y}$ and $\boldsymbol{Y}^d$ are close; it is the pair $\left\{\widetilde{\boldsymbol{P}}^*, \boldsymbol{Y}\right\}$ that intrinsically encodes the behavior $\boldsymbol{T}(q)$. In addition, $\widetilde{\boldsymbol{P}}^*$ is not exactly the $\boldsymbol{P}^*$ solving (11) because EDILC stops after a finite number of iterations. Algorithm 1 runs in a mixed-mode: each iteration requires two real-time experiments which are proportional to the number N of samples of the desired trajectory. These computations are repeated for a number of iterations ϵ. We conclude that this algorithm complexity is $O\left(N^2\right)$. Note that the gradient computation has been greatly optimized: it requires a single gradient experiment than the previous variants of the EDILC algorithm in [8] which required a number of gradient experiments equal to the number of CLS inputs, in which case the complexity would have been $O\left(N^3\right)$.

Monotonic convergence is desirable for EDILC and is ensured by the proper selection of step size $\boldsymbol{\xi}$. The learning level L1 makes the CLS match $\boldsymbol{T}_{ORM}(q)$ to some extent; however, a mismatch is expected. The I/O samples from the EDILC trajectories used to learn $\left\{\widetilde{\boldsymbol{P}}^*, \boldsymbol{Y}\right\}$ offer the opportunity to identify a linear lower-order model for the CLS, which commonly has low-pass behavior. We denote the model identified from the I/O data as $\widetilde{\boldsymbol{T}}(q)$. Then, $\boldsymbol{T}_{ORM}(q)$ and $\widetilde{\boldsymbol{T}}(q)$ are both rough CLS estimates, which can be used to derive suitable step sizes for monotonic EDILC convergence. The most conservative selection of $\boldsymbol{\xi}$ is selected by solving the subsequent optimization:

$$\begin{aligned}
&\boldsymbol{\xi}_1^* = \arg \max_{\boldsymbol{\xi}} \textstyle\sum_{i=1}^{p} \xi_{1,i},\ s.t. \xi_{1,i} > 0,\ \| \boldsymbol{I}_{p\times p} - \tfrac{2}{N} \boldsymbol{T}_{ORM}(q) \boldsymbol{\xi}_1 \boldsymbol{T}_{ORM}^T\left(\tfrac{1}{q}\right) \|_\infty < 1;\\
&\boldsymbol{\xi}_2^* = \arg \max_{\boldsymbol{\xi}} \textstyle\sum_{i=1}^{p} \xi_{2,i}, s.t. \xi_{2,i} > 0,\ \| \boldsymbol{I}_{p\times p} - \tfrac{2}{N} \widetilde{\boldsymbol{T}}(q) \boldsymbol{\xi}_2 \widetilde{\boldsymbol{T}}^T\left(\tfrac{1}{q}\right) \|_\infty < 1;\\
&\boldsymbol{\xi}^* = diag[\xi_i],\ \xi_i = \min(\xi_{1,i}, \xi_{2,i}),\ i = \overline{1, p},
\end{aligned} \tag{13}$$

where $\| \ . \ \|_\infty$ is the norm calculated as the greatest singular value of a t.m. over the frequency domain, while the identity transfer matrix is $\boldsymbol{I}_{p\times p}$. Derivation of the convergence condition for the t.m. norm inequality constraints in (13) is presented in Appendix A.

$\boldsymbol{T}_{ORM}(q)$ is usually selected diagonally to ensure decoupling between the control channels through controller design, at least in the steady state, for example, by including integral action in the controllers. For example, for a first-order square reference model of size 2-by-2, the continuous-time unitary-gain t.m. is selected as $\boldsymbol{T}_{ORM}(s) = diag\left(\frac{e^{-sT_{m1}}}{1+sT_1}, \frac{e^{-sT_{m2}}}{1+sT_2}\right)$ (s is the continuous-time Laplace complex argument), where T_1 and T_2 are the time constants that impose the transient behavior, T_{m1}, T_{m2} are delays. The real achieved behavior in continuous time can be modeled as $\widetilde{\boldsymbol{T}}(s) = \left[\widetilde{T}_{11}\widetilde{T}_{12}; \widetilde{T}_{21}\widetilde{T}_{22}\right]$ with $\widetilde{T}_{11}(s) = \frac{k_{11}(T_{11}s+1)e^{-sT_{m11}}}{s^2+b_{11}s+c_{11}}$, $\widetilde{T}_{21}(s) = \frac{k_{21}s(T_{21}s+1)e^{-sT_{m21}}}{s^2+b_{21}s+c_{21}}$, $\widetilde{T}_{12}(s) = \frac{k_{12}s(T_{12}s+1)e^{-sT_{m12}}}{s^2+b_{12}s+c_{12}}$, $\widetilde{T}_{22}(s) = \frac{k_{22}(T_{22}s+1)e^{-sT_{m22}}}{s^2+b_{22}s+c_{22}}$ where the

second-order t.f.s on the anti-diagonal capture any coupling transient that possibly vanishes in time and oscillates at most, while the second-order t.f.s from the main diagonal captures any transient responses that are oscillatory at most and may or may not achieve the desired unit gains. Time delay and NMP zeros must be included in the ORM definition without difficulty, if present. The equivalent discretized causal behavior model $\widetilde{\boldsymbol{T}}(q)$ of the CLS $\widetilde{\boldsymbol{T}}(s)$, identifiable from I/O data, is (where $m_{ij} \in \mathbb{N}$ are the delay steps)

$$\widetilde{\boldsymbol{T}}(q) = \begin{bmatrix} \frac{b_{11}^1 q^{-1} + b_{11}^2 q^{-2}}{1 + a_{11}^1 q^{-1} + a_{11}^2 q^{-2}} q^{-m11} & \frac{b_{12}^0 + b_{12}^1 q^{-1} + b_{12}^2 q^{-2}}{1 + a_{12}^1 q^{-1} + a_{12}^2 q^{-2}} q^{-m12} \\ \frac{b_{21}^0 + b_{21}^1 q^{-1} + b_{21}^2 q^{-2}}{1 + a_{21}^1 q^{-1} + a_{21}^2 q^{-2}} q^{-m21} & \frac{b_{22}^1 q^{-1} + b_{22}^2 q^{-2}}{1 + a_{22}^1 q^{-1} + a_{22}^2 q^{-2}} q^{-m22} \end{bmatrix}. \tag{14}$$

During ILC operation, $\widetilde{\boldsymbol{T}}(q)$ is identified using I/O data either from the nominal or gradient experiment, or from both. The model can be re-estimated or can be continuously refined with every ILC iteration, with no computational burden, owing to the offline experimental nature of ILC. Model precision is not crucial because uncertainty only affects the monotonic convergence condition, and more cautious learning can be attempted. In any case, using low-order models such as (14) is recommended because the ORM control ensures that the CLS approximately matches $\boldsymbol{T}_{ORM}(q)$.

A better initial value than $\boldsymbol{P}^0 = \boldsymbol{Y}^d$ in the EDILC law (12) could theoretically be found based on the desired ORM $\boldsymbol{T}_{ORM}(q)$. This could further improve the convergence speed of the subsequent EDILC. The initialization is expressed as:

$$\boldsymbol{P}^0 = \arg \min_{\boldsymbol{P}} \frac{1}{N} (\boldsymbol{T}_{ORM}\boldsymbol{P} - \boldsymbol{Y}^d)^T \left(\boldsymbol{T}_{ORM}\boldsymbol{P} - \boldsymbol{Y}^d \right) + \frac{\lambda}{N} \boldsymbol{P}^T \boldsymbol{P}, \tag{15}$$

where the term $\boldsymbol{T}_{ORM}\boldsymbol{P}$ is evaluated by simulation, based on $\boldsymbol{T}_{ORM}(q)$, on a finite-time scenario with length N. However, experimental investigations reveal that this initialization renders oscillatory $\boldsymbol{P}^0$ near its ends because of noncausal filtering involved in the model-based solution of (15).

The EDILC learns primitives under constraints imposed on the operational variables, such as on the control action $\boldsymbol{a}_k$ (and also on its derivative) and on the system (and also the CLS) output $\boldsymbol{y}_k$ (and also on its derivative). The reference input constraints are not of interest in this type of tracking problem. These constraints are mostly of the inequality type owing to the underlying dynamical system. They can be directly handled either as hard constraints appended to (7) or as soft constraints by additional terms in the cost function of the optimization (7). It was shown [44] that inequality-type hard constraint handling is still possible with EDILC in an experimental-driven fashion with unknown system dynamics. For primitive-based learning with EDILC, inequality-type constraints on $\boldsymbol{a}_k$ are possible, but generally not an issue, as long as the CLS operates in the linearized range resulting from the level L1 learning phase. Any constraints on $\boldsymbol{a}_k$ (and possibly on its derivative) may negatively impact the level of L2 learning trajectory tracking performance. Therefore, the constraints on $\boldsymbol{y}_k$ are more interesting, mainly for safety reasons such as saturation, which when violated in mechanical systems could lead to permanent damage. The constraints on $\boldsymbol{y}_k$ will not be handled explicitly at this level L2, but rather indirectly at the next level L3 and deferred to a later section.

Concluding the level L2 learning aspects discussion, the final tertiary learning level L3 is presented next.

5. The L3 Level—New Desired Trajectories Optimally Tracked by Extrapolating Primitives' Behavior

Suppose a new desired trajectory $\boldsymbol{Y}^d$ is to be tracked; this time, without relearning by EDILC. This amounts to finding the optimal reference input $\boldsymbol{P}^*$ to obtain a CLS output that perfectly tracks $\boldsymbol{Y}^d$. A solution to use and extrapolate the behavior of learned primitive pairs was proposed in [8]. Because the solution uses delayed copies of the output primitives from the learned pairs serving as basis functions for approximating $\boldsymbol{Y}^d$, the reason for using

EDILC to learn output primitives that have good approximation capability is transparent. Among others, Gaussian shapes are well-known in function approximation theory; therefore, primitive pairs $\left\{\widetilde{\boldsymbol{P}}^{*[\delta]}, \boldsymbol{Y}^{[\delta]}\right\}$ where the output primitives $\boldsymbol{Y}^{[\delta]}$ have a Gaussian-like shape, are commonly learned with EDILC.

In the following, the desired trajectory $\boldsymbol{Y}^d$ is assumed to start from zero and end in non-zero values, component-wise. To accommodate for fine approximations whenever $\boldsymbol{Y}^d$ ends in a non-zero value and starts from zero with a non-zero slope, both $\boldsymbol{Y}^d$ and the copied primitive pairs $\left\{\widetilde{\boldsymbol{P}}^{*[\delta]}, \boldsymbol{Y}^{[\delta]}\right\}$ will undergo a time-extension process. The length of $\boldsymbol{Y}^d$ is N (same as the primitives' length), and with no generality loss, N is assumed even. A longer desired trajectory was deferred to a later discussion. The optimal reference input calculation Algorithm 2 is summarized below.

Algorithm 2

Step 0. Initialize $\boldsymbol{P}^0 = \boldsymbol{Y}^d$, then for each $j = \overline{0, \epsilon}$ call the subsequent steps.

Step 1. Let $y^d_{k,1}, y^d_{k,2}, \ldots, y^d_{k,p}$ be the components of $\boldsymbol{Y}^d$. In lifted notation, these trajectories are $\boldsymbol{Y}^d_1, \boldsymbol{Y}^d_2, \ldots, \boldsymbol{Y}^d_p$.

Step 2. Extend each $\boldsymbol{Y}^d_1, \ldots, \boldsymbol{Y}^d_p$ to size $2N$, padding with the first and last sample of $\boldsymbol{Y}^d_1, \ldots, \boldsymbol{Y}^d_p$, respectively with $N/2$ values to the right and to the left, respectively. We denote the extended trajectories $\boldsymbol{Y}^{d[e]}_1, \boldsymbol{Y}^{d[e]}_2, \ldots, \boldsymbol{Y}^{d[e]}_p$.

Step 3. Arrange $\boldsymbol{Y}^{d[e]} = [\boldsymbol{Y}^{d[e]^T}_1, \boldsymbol{Y}^{d[e]^T}_2, \ldots, \boldsymbol{Y}^{d[e]^T}_p]^T \in \Re^{2pN}$.

Step 4. The N-length input primitives $\left(\widetilde{\boldsymbol{P}}^{*[\delta]}_1\right), \left(\widetilde{\boldsymbol{P}}^{*[\delta]}_2\right), \ldots, \left(\widetilde{\boldsymbol{P}}^{*[\delta]}_p\right)$ and N-length output primitives $\left(\boldsymbol{Y}^{[\delta]}_1\right), \left(\boldsymbol{Y}^{[\delta]}_2\right), \ldots, \left(\boldsymbol{Y}^{[\delta]}_p\right)$, from the δ^{th} learned pair, are extended to size $2N$ by padding to the left and right $N/2$ zeros, respectively. Padding with zeros is employed because both $\widetilde{\boldsymbol{P}}^{*[\delta]}, \boldsymbol{Y}^{[\delta]}$ start and end at zero, owing to the Gaussian-like shape of $\boldsymbol{Y}^{[\delta]}$. We denote $\left\{\widetilde{\boldsymbol{P}}^{*[\delta e]}, \boldsymbol{Y}^{[\delta e]}\right\}$ as the extended primitives.

Step 5. Make M copies of the randomly occurring δ extended pairs $\left\{\widetilde{\boldsymbol{P}}^{*[\delta e]}, \boldsymbol{Y}^{[\delta e]}\right\}$, let them be called as $\left\{\widetilde{\boldsymbol{P}}^{*[\pi e]}, \boldsymbol{Y}^{[\pi e]}\right\}$, $\pi = \overline{1, M}$.

Step 6. Delay each component $\widetilde{\boldsymbol{P}}^{*[\pi e]}_1, \ldots, \widetilde{\boldsymbol{P}}^{*[\pi e]}_p$ and $\boldsymbol{Y}^{[\pi e]}_1, \ldots, \boldsymbol{Y}^{[\pi e]}_p$ of the π pairs by an integer uniform number $\theta \in [-N, N]$. Because the delay is a non-circular shift, padding is again required to either end, based on the sign of θ. The value used for padding for a number of $|\theta|$ samples is the first or last sample value of the unshifted signals. Let $\widetilde{\boldsymbol{P}}^{*[\theta_\pi e]}_1, \ldots, \widetilde{\boldsymbol{P}}^{*[\theta_\pi e]}_p, \boldsymbol{Y}^{[\theta_\pi e]}_1, \ldots, \boldsymbol{Y}^{[\theta_\pi e]}_p$ be the notation for the delayed input and output primitive of each of the π pairs.

Step 7. Obtain $\widetilde{\boldsymbol{P}}^{*[\theta_\pi e]} = [\widetilde{\boldsymbol{P}}^{*[\theta_\pi e]^T}_1, \ldots, \widetilde{\boldsymbol{P}}^{*[\theta_\pi e]^T}_p]^T \in \Re^{2pN}$ and $\boldsymbol{Y}^{[\theta_\pi e]} = [\boldsymbol{Y}^{[\theta_\pi e]^T}_1, \ldots, \boldsymbol{Y}^{[\theta_\pi e]^T}_p]^T \in \Re^{2pN}$. The basis function matrix is $\boldsymbol{Y}^{[b]} = \left[\boldsymbol{Y}^{[\theta_1 e]}, \ldots, \boldsymbol{Y}^{[\theta_M e]}\right] \in \Re^{2pN \times M}$. A visual summary of the extension with padding, delay with padding, and final concatenation is shown for the output primitives in Figure 1.

Step 8. Compute $\boldsymbol{w} = [w_1, \ldots, w_M]^T \in \Re^M$ as the minimizer of $\| \boldsymbol{Y}^{[b]}\boldsymbol{w} - \boldsymbol{Y}^{d[e]} \|_2$ with linear least-squares. Each component of $\boldsymbol{w}$ multiplies a specific $\boldsymbol{Y}^{[\theta_\pi e]}$.

Step 9. Based on Theorem 1 from [8], the optimal reference to ensure the tracking of $\boldsymbol{Y}^d$ is:

$$\widetilde{\boldsymbol{P}}^* = \sum_{\pi=1}^{M} w_\pi \widetilde{\mathbf{P}}^{*[\theta_\pi e]}, \tag{16}$$

Step 10. In (16), $\widetilde{\boldsymbol{P}}^* = [\widetilde{\boldsymbol{P}}^{*T}_1 \widetilde{\boldsymbol{P}}^{*T}_2 \ldots \widetilde{\boldsymbol{P}}^{*T}_p]^T \in \Re^{2pN}$. To obtain N-length channel-wise optimal reference inputs, the p components are clipped in the middle of their intervals.

Step 11. The clipped optimal references are set as CLS inputs, and the CLS output tracks $\boldsymbol{Y}^d$. The algorithm ends.

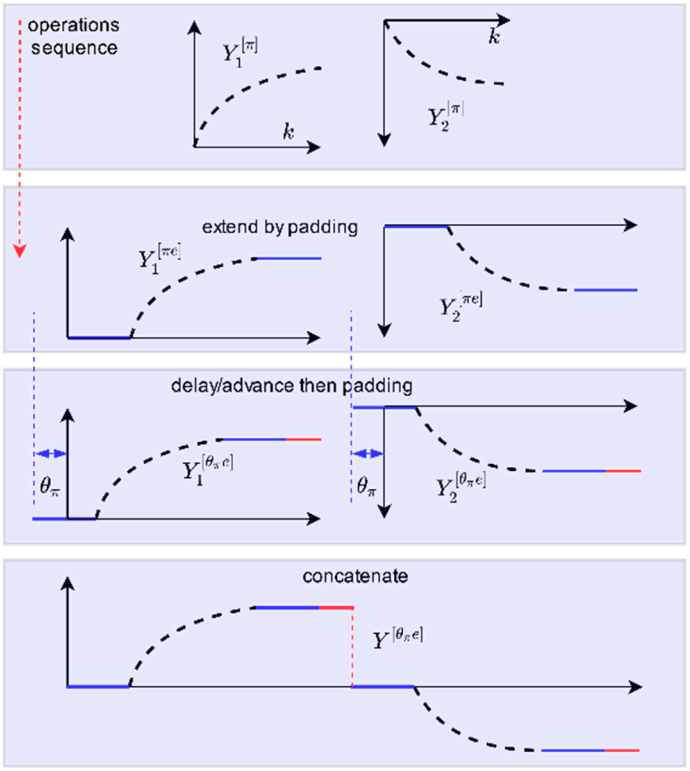

Figure 1. Extension with padding, delay with padding and final concatenation concepts, in the space of two output primitives. Similar operations are performed for the input primitives.

Algorithm 2 is entirely performed offline, not impacting the real-time control task. Of course, the planner is allowed sufficient computation time. If the trajectories to be tracked on each subinterval are very long, this may impact performance, but we have tried to solve this particular issue by dividing the longer trajectories in shorter, manageable trajectories. Essentially, Algorithm 2 has two complexity components: the least squares matrices preparation part which has complexity $O(N^3)$ and the least squares regression solution which combines matrix multiplication and the SVD factorization. For this numerical solution, the complexity is $O(M^2X)$ where M is the size of the regression vector $\boldsymbol{w}$ and $X = 2pN$ is the number of examples (N is the experiment length and p is the number of CLS inputs and outputs). Hence, the complexity of Algorithm 2 is $O(N^4)$ but since p is usually a small constant, it can be approximated to $O(N^3)$.

The optimality of $\widetilde{\boldsymbol{P}}^*$ in (16) with respect to the tracking performance and its relation to the true optimal solution $\boldsymbol{P}^*$ of (11) is analyzed next.

Theorem 1. *When $\boldsymbol{Y}^{d[e]}$ is approximated with a bounded reconstruction error $\boldsymbol{\zeta}$, then $\boldsymbol{P}^*$ can be computed with arbitrary precision.*

Proof. The desired trajectory is decomposed as in $\boldsymbol{Y}^{d[e]} = \sum_{\pi=1}^{M} w_\pi \boldsymbol{Y}^{[\theta_\pi e]} + \boldsymbol{\zeta}$, where $\boldsymbol{\zeta}$ is the bounded reconstruction error. The optimal reference input ensuring perfect tracking is $\boldsymbol{P}^* = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{Y}^{d[e]}$ and is expressed as $\boldsymbol{P}^* = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\left(\sum_{\pi=1}^{M} w_\pi \boldsymbol{Y}^{[\theta_\pi e]} + \boldsymbol{\zeta}\right) = \sum_{\pi=1}^{M} w_\pi \widetilde{\boldsymbol{P}}^{*[\theta_\pi e]} + (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{\zeta}$. As $\| \boldsymbol{\zeta} \|_2 \to 0$, then $\| (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{\zeta} \|_2 \leq \| (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T \|_2 \| \boldsymbol{\zeta} \|_2 \to 0$ follows, where $\| \bullet \|_2$ applied to a matrix is the induced 2-norm by a vector, which represents the matrix's largest singular value, $\| \bullet \|_2$ applied to a vector is the Euclidean norm. □

Theorem 1 shows that a good approximation of the desired trajectory in terms of a small error leads to a reference input close to the optimal one. This ensures good tracking

performance. To achieve a small approximation error, the number of basis functions can be adjusted along with the desired shape.

Managing Longer Desired Trajectories

Suppose that each signal component $y^d_{k,1}, y^d_{k,2}, \ldots, y^d_{k,p}$ of $\boldsymbol{Y}^d$ has length Ω as a multiple of N (padding is again usable when Ω is not a multiple of N). One solution to obtain the desired optimal reference input that achieves CLS output tracking of $\boldsymbol{Y}^d$ is to extend the primitive pairs $\left\{\widetilde{\boldsymbol{P}}^{*[\delta e]}, \boldsymbol{Y}^{[\delta e]}\right\}$ to length Ω with appropriate padding. In addition, proportionally more M copies of these extended pairs were used. This will increase the size of $\boldsymbol{w}$ and the ill-condition the least squares. An intuitive mitigation is to perform tracking experiments on subintervals of length N as divisions of $\boldsymbol{Y}^d$.

For example, a longer trajectory of length $2N$, comprising two shorter trajectories of length N, is displayed in Figure 2 for a single input single output (SISO) system. The end of trajectory $\overline{00'}$ is the start of trajectory $\overline{0'B}$. A base reference frame in discrete-time is $< k0y^d >$ with a discrete-time index k.

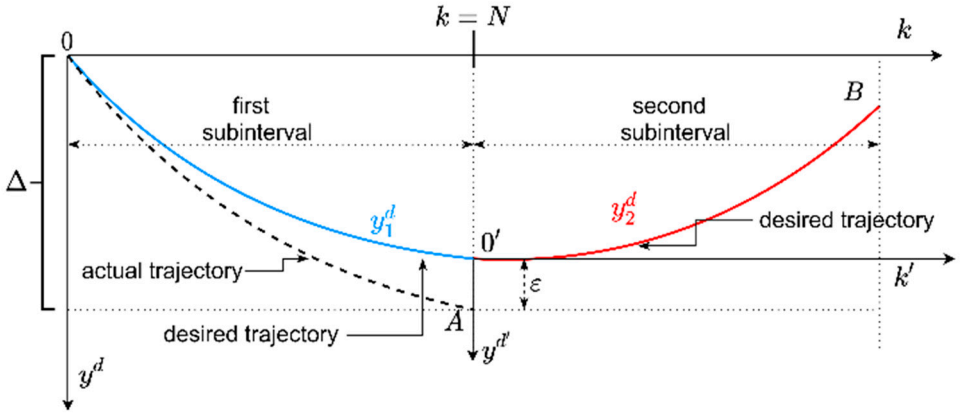

Figure 2. A trajectory consisting of two connected curves $\overline{00'}$ and $\overline{0'B}$; moving the reference frame to $0'$ makes the trajectory $\overline{0'B}$ start in zero.

One solution is to translate the origin of the base reference frame $< k0y^d >$ to the starting point of each piece of the desired trajectory of length N. In Figure 2, $< k0y^d >$ is translated to $< k'0'y^{d'} >$. This is equivalent to having the desired trajectory $\overline{0'B}$ starting from the origin. Thus, the planner always computes the optimal reference inputs for a desired trajectory starting in zero initial conditions. Considering that the tracked trajectory $\overline{0A}$ does not perfectly track the desired one $\overline{00'}$ and there exists a vertical distance ε between the endpoints of $\overline{0A}$ and $\overline{00'}$, then in the translated reference frame $< k'0'y^{d'} >$ the closed-loop CS will have a non-zero initial condition in the ε-vicinity near $0'$. In practice, ε is expected to be small. The CS response due to the non-zero initial condition vanishes over time.

After executing the tracking task on the second subinterval with desired trajectory $\overline{0'B}$, the execution is translated back in the base frame $< k0y^d >$ for visualization; afterwards, the reference frame $< k'0'y^{d'} >$ is moved in the start of the desired trajectory of the third subinterval, and the entire process repeats itself for all subintervals. After all subinterval tracking tasks are executed, the actual obtained trajectories are translated back to the base reference frame and then concatenated to be presented as a longer tracking task.

Inequality-type constraints on $\boldsymbol{Y}$ can be handled indirectly at the tertiary learning level. The aim of this learning level is to ensure the theoretical perfect tracking of $\boldsymbol{Y}^d$, meaning that $\boldsymbol{Y} = \boldsymbol{Y}^d$. This implies that any constraint on $\boldsymbol{Y}^d$ is imposed on $\boldsymbol{Y}$ also. We note that inequality constraints on the rate of $\boldsymbol{Y}$ are again not of particular interest when trajectory tracking is aimed; therefore, the magnitude constraints are more feasible in the lifted form as $\boldsymbol{Y}^d_{low} \leq \boldsymbol{Y}^d \leq \boldsymbol{Y}^d_{up}$. As a result, the constraints on $\boldsymbol{Y}$ are softly handled.

With the discussed aspects of the L3 learning level, the entire hierarchical three-level framework shown in Figure 3 was validated on a real-world system.

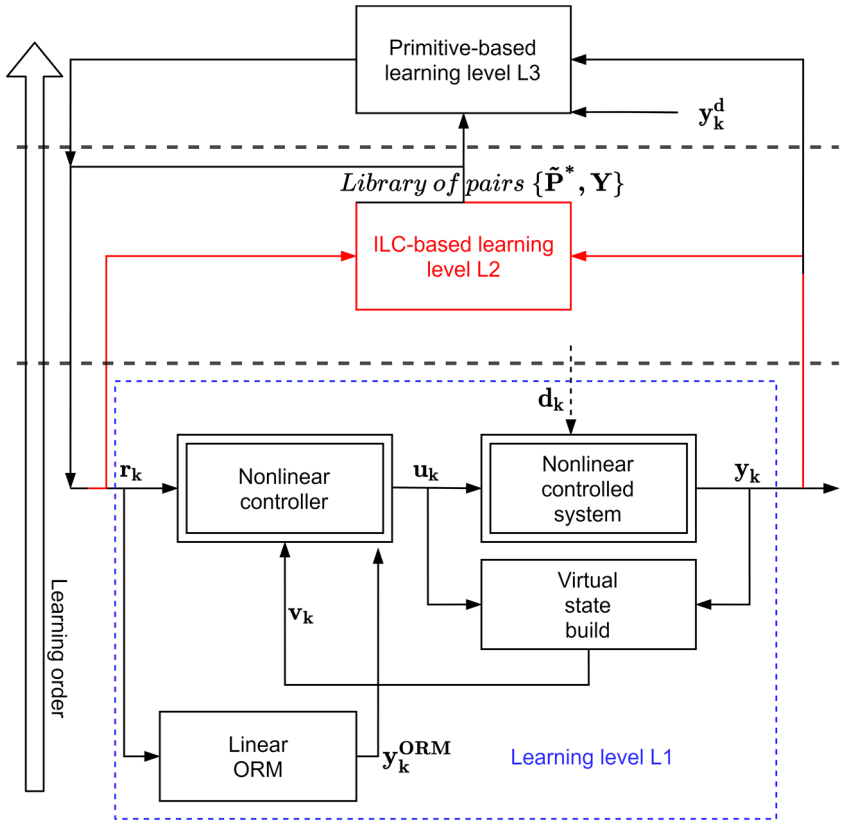

Figure 3. The hierarchical learning on levels L1–L3. Possible disturbances $\boldsymbol{d}_k$ may affect the nonlinear controlled system.

6. Validation Case Study

6.1. The Hybrid Software-Electrical Controlled System

Figure 4 shows the proposed multivariable nonlinear controlled system. The USB-based communication between an Arduino-based board and a computer (PC) manages the pulse-width modulated interface (PWM) used for voltage input action setting and for the analog-to-digital interface to read controlled voltages. Here, the PWM duty cycles are the system's input actions, $\boldsymbol{a} = [a_1, a_2]^T$. Their domain [0;1] is shifted to $[-0.5,0.5]$ for symmetric operation. Let V_{o1} and V_{o2} be the capacitors C_1 and C_2 voltages, respectively. The system controlled outputs are shifted to [–2.5; 2.5] V for symmetric operation; therefore, they are calculated on the PC as $y_1 = V_{o1} - 2.5$, $y_2 = V_{o2} - 2.5$. All communication, processing and control tasks are performed using MATLAB software package.

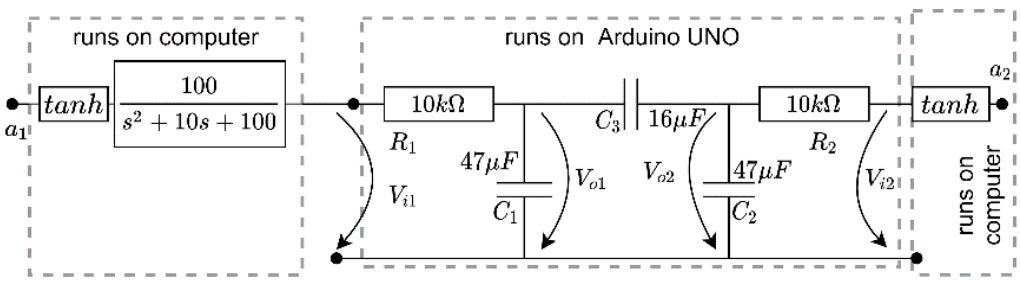

Figure 4. Conceptual description of the controlled system.

Part of this Hammerstein-like system is electrical (circuit parameters are $C1 = C2 =$ 47 μF,$C3 = 16.5$ μF, $R1 = R2 =$ 10kΩ,), while the input nonlinearities (characterized by a hyperbolic tangent function) and a linear t.f. on the first input action (which has a resonant mode at approximately 7.3 rad/s) run on a PC. The system is zero-order-hold sampled at $T_s = 0.1$ s, rendering the I/O as $\boldsymbol{a}_k = [a_{k,1}, a_{k,2}]^T$ and $\boldsymbol{y}_k = [y_{k,1}, y_{k,2}]^T$. Prior experiments reveal a unit step delay T_s to be included when defining the ORM for level L1 learning and also in defining $\boldsymbol{y}_k^d$ for level L2 learning [45].

6.2. ORM Tracking Learned for Feedback Linearization (Level L1)

First, a PI-type controller of the form $a_{k,1} = [0.15 + 0.1T_s/(q-1)](\rho_{k,1} - y_{k,1})$, $a_{k,2} = [0.35 + 0.25T_s/(q-1)\ ](\rho_{k,2} - y_{k,2})$ is employed to collect transition sample experiences from the CLS under noisy additive disturbances on the control action inputs [45]. These disturbances were used to enhance exploration. For a prescribed time of 1500 s, sequences of piecewise constant reference inputs with random amplitudes within some tunable domains drive the CLS dynamics. The state of a transition tuple $\left(\boldsymbol{s}_k^{ext}, \boldsymbol{a}_k, \boldsymbol{s}_{k+1}^{ext}\right)$ is $\boldsymbol{s}_k^{ext} = [y_{k,1}, y_{k,2}, \ldots, y_{k-5,1}, y_{k-5,2}, a_{k-1,1}, a_{k-1,2}, \ldots, a_{k-5,1}, a_{k-5,2}, s_{k,1}^m, s_{k,2}^m, \rho_{k,1}, \rho_{k,2}\ ]^T$. Transitions from the time steps when $\boldsymbol{\rho}_{k+1} = \boldsymbol{\rho}_k$ were considered. Approximately 14,000 transitions were formed and used for AIVI learning.

At the same time, to achieve indirect CLS linearization by ORM tracking, level L1 learning requires the selection of a linear ORM model as $\boldsymbol{T}_{ORM}(s) = diag(e^{-sT_s}/(s+1), e^{-sT_s}/(s+1))$, in the easy-interpretable continuous-time. Before discretizing $\boldsymbol{T}_{ORM}(s)$, note that it acknowledges the dead-time of one sampling period observed for the system, in addition to the relative degree one induced by the zero-order hold. This is read from the discrete-time I/O sample acquisition of the transition collection phase. The discretized t.f. $\boldsymbol{T}_{ORM}(q) = diag\left(\frac{0.0952q^{-2}}{1-0.9048q^{-1}}, \frac{0.0952q^{-2}}{1-0.9048q^{-1}}\right)$ is then expressed as a state-space model resulting from an observable canonical transformation. The resulting 4$^{\text{th}}$ order state-space uses two extra states denoted with ϖ, owing to the time delay of each t.f. from the $\boldsymbol{T}_{ORM}(q)$. The model used is ([45]):

$$\begin{cases} \begin{bmatrix} s_{k+1,1}^m \\ \varpi_{k+1,1} \\ s_{k+1,2}^m \\ \varpi_{k+1,2} \end{bmatrix} = \begin{bmatrix} 0.9048 & 0.0952 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9048 & 0.0952 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_{k,1}^m \\ \varpi_{k,1} \\ s_{k,2}^m \\ \varpi_{k,2} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \rho_{k,1} \\ \rho_{k,2} \end{bmatrix}, \\ \boldsymbol{y}_k^m = [y_{k,1}^m y_{k,2}^m]^T = [1,0,1,0] \times [s_{k,1}^m \varpi_{k,1} s_{k,2}^m \varpi_{k,2}]^T. \end{cases} \quad (17)$$

A single hidden layer feed-forward controller NN of size 28–15–2 is learned to output $\boldsymbol{a}_k$ for the input $\boldsymbol{s}_k^{ex}$, using the AIVI iteration steps, which implies NN training for both the controller and for a Q-function approximator NN, having a similar architecture but size 30–100–1. The CLS with the converged AIVI NN controller after six iterations of the AIVI approach is shown the ORM tracking ability in Figure 5. To check this convergence, the controller resulting after each iteration of the AIVI is tested on a scenario in which sequences of piecewise constant reference inputs drive the CLS. The ORM tracking performance was measured with a regularized finite-length version of V_{ORM}^{∞}, defined as $V_{test} = 1/N \cdot V_{ORM}^{N}$, with $N = 2000$. The convergence is illustrated in Figure 6. It occurs relatively quickly, and the best controller is shown in Figure 5.

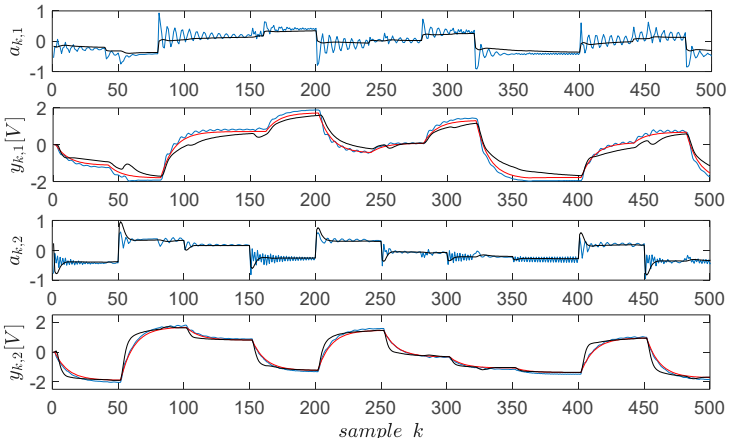

Figure 5. Tracking ORM results: $a_{k,1}$, $y_{k,1}$,$a_{k,2}$,$y_{k,2}$ with AIVI (blue); $a_{k,1}$,$y_{k,1}$,$a_{k,2}$,$y_{k,2}$ with PI control (black); $y^m_{k,1}$, $y^m_{k,2}$ (red).

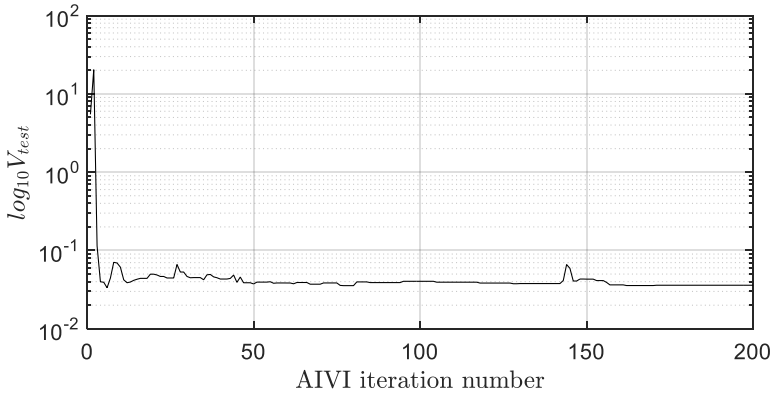

Figure 6. ORM tracking performance V_{test} throughout AIVI, measured on the test scenario for 200 iterations.

This concludes the L1 level indirect CLS linearization by matching the behavior of $\boldsymbol{T}_{ORM}(q)$. The subsequent EDILC L2 level learning benefits from this assumption.

6.3. Two Primitives Learned with EDILC (Level L2)

A number of $\delta = 2$ primitives of length $N = 400$ samples are learned using the EDILC Algorithm 1 with $\boldsymbol{P}^0 = \boldsymbol{Y}^d$, $\mu = 0.01$, $\epsilon = 30$, and the learning rate $\boldsymbol{\xi} = diag(299.13,\ 281.35)$ was determined by solving (13) using $\boldsymbol{T}_{ORM}(q)$ and an approximate CLS t.f. identified as [45]

$$\widetilde{\boldsymbol{T}}(q) = \begin{bmatrix} \frac{0.1278q^{-2}}{1-0.77258q^{-1}-0.1126q^{-2}} & \frac{0.02189q^{-2}-0.02101q^{-3}}{1-1.213q^{-1}+0.2206q^{-2}} \\ \frac{0.0756q^{-2}-0.05602q^{-3}}{1-0.8129q^{-1}+0.7474q^{-2}} & \frac{0.1554q^{-2}}{1-0.4374q^{-1}-0.422q^{-2}} \end{bmatrix}. \tag{18}$$

For the first primitive $\left\{\widetilde{\boldsymbol{P}}^{*[1]}, \boldsymbol{Y}^{[1]}\right\}$, the desired trajectory spanning N = 400 time steps ($k = \overline{0, 399}$) is $y^{d[1]}_{k,1} = \chi_1 \exp(\chi_2(k-\chi_3)^2)$, ($\chi_1 = 2$, $\chi_2 = -0.002$, $\chi_3 = 200$) and $y^{d[1]}_{k,2} = 0$, whereas the second primitive $\left\{\widetilde{\boldsymbol{P}}^{*[2]}, \boldsymbol{Y}^{[2]}\right\}$ is characterized by the same length and by $y^{d[2]}_{k,1} = 0$, $y^{d[2]}_{k,2} = y^{d[1]}_{k,1}$. That is, for each primitive, one desired output is Gaussian, while the other is zero. The first primitive $\left\{\widetilde{\boldsymbol{P}}^{*[1]}, \boldsymbol{Y}^{[1]}\right\}$ learned using EDILC is shown in Figure 7.

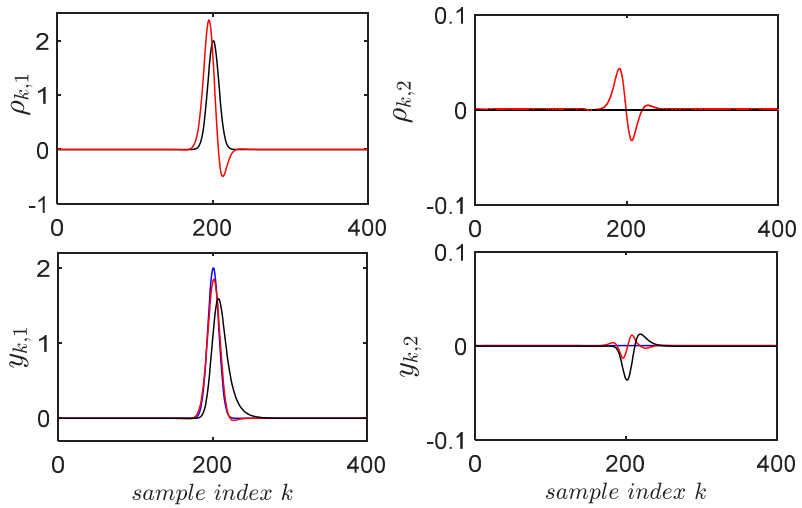

Figure 7. Learning results for $\{\tilde{\boldsymbol{P}}^{*[1]}, \boldsymbol{Y}^{[1]}\}$: $y_{k,1}^{d[1]}$ and $y_{k,2}^{d[1]}$ (blue); $y_{k,1}^{0}$, $y_{k,2}^{0}$, $\rho_{k,1}^{0}$, $\rho_{k,2}^{0}$ (black); $y_{k,1}^{30}$, $y_{k,2}^{30}$, $\rho_{k,1}^{*}$, $\rho_{k,2}^{*}$ (red).

6.4. Optimal Tracking Predicted from Primitives (Level L3)

At level L3, it is desired to track $y_{k,1}^{d} = max\{-0.7, \min[0.7, \chi_1 \sin(\chi_2 k T_s)\sin(\chi_3 k T_s)]\}$ ($\chi_1 = 1.1$, $\chi_2 = 0.6$, $\chi_3 = 0.02$) and $y_{k,2}^{d} = \chi_1 \sin(\chi_2 k T_s)\sin(\chi_3 k T_s)$, ($\chi_1 = -0.9$, $\chi_2 = 0.01$, $\chi_3 = 0.3$) for $k = \overline{0, \Omega - 1}$, where $\Omega = 1600 = 4 \times N$. Trajectory $y_{k,1}^{d}$ is clipped in amplitude with min/max operators to induce an output constraint. Then, the CLS output $\boldsymbol{Y}$ will be constrained because, by the primitive-based mechanism, level L3 ensures that $\boldsymbol{Y} = \boldsymbol{Y}^{d}$. Then, $\boldsymbol{Y}^{d}$ is divided into four subintervals, and the optimal reference input calculation by the primitive-based approach together with the trajectory tracking takes place on one subinterval of length N at a time. Each subinterval is offset to start the tracking at zero intervals. The parameters in Algorithm 2 are $N = 400$, $\delta = 2$ primitive pairs, $M = 400$ copies of the two primitives are employed, $\theta \in [-400, 400]$, to fulfill Theorem 1.

Figure 8 displays the extended, padded, delayed, once-again padded, and concatenated output primitives $\boldsymbol{Y}_1^{[\theta_\pi e]}$, $\boldsymbol{Y}_2^{[\theta_\pi e]}$. Notably, the basis function matrix is $\boldsymbol{Y}^{[b]} \in \Re^{1600 \times 400}$. The tracking results for each subinterval are re-offset in the original reference frame and displayed in Figure 9 where, for comparisons, the tracking results obtained with $\boldsymbol{P}_1 = \boldsymbol{Y}_1^{d}, \boldsymbol{P}_2 = \boldsymbol{Y}_2^{d}$ are also plotted. It is obvious that tracking is more accurate with the primitive-based approach (red line vs. blue line) than with respect to the case when it is not used (black line vs. blue line). The latter case corresponds to using only the CLS linearized with L1 level, with the reference inputs being set as usually, leaving the control system to reactively ensure trajectory tracking. Primitive-based tracking also better satisfies the constraints (in a soft, not a hard sense), which is enforced on the first axis only.

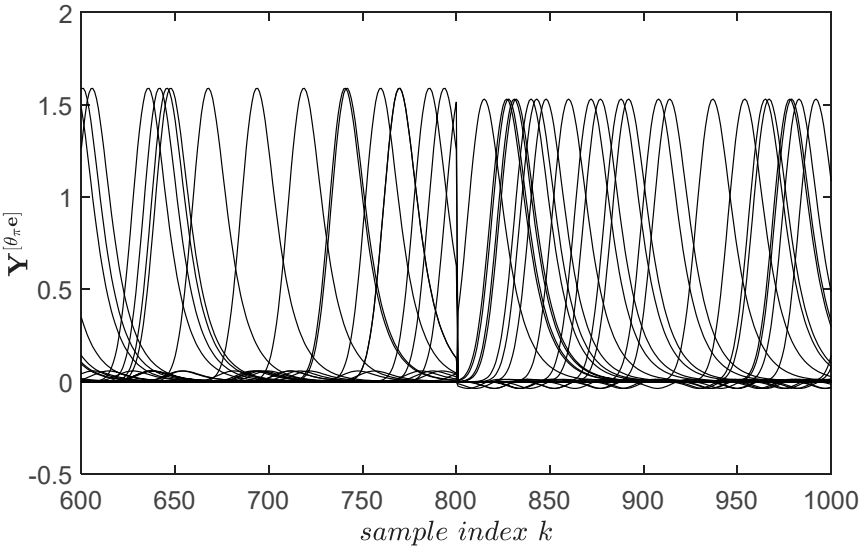

Figure 8. The 400 extended, padded, delayed and finally padded copies of $\boldsymbol{Y}_1^{[\theta_\pi e]}$, $\boldsymbol{Y}_2^{[\theta_\pi e]}$, zoomed in the sample index range [600;1000].

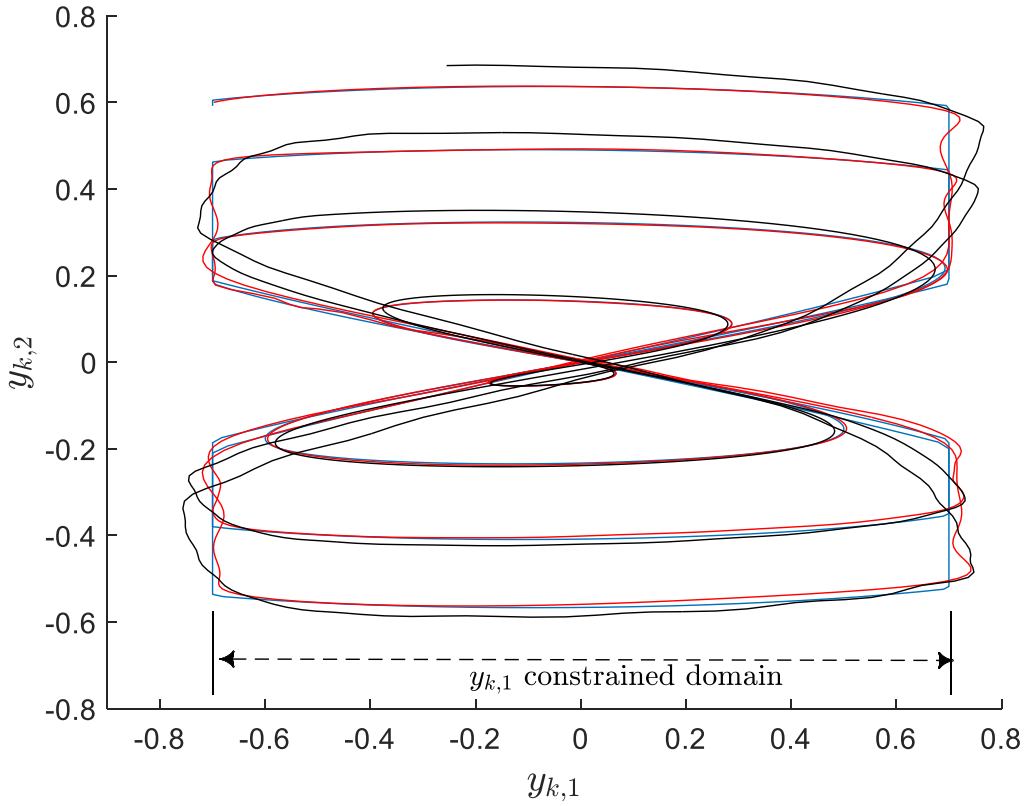

Figure 9. The recomposed tracking result on the long desired trajectory: $y_{k,1}^d$ and $y_{k,2}^d$ in blue; $y_{k,1}$ and $y_{k,2}$ (in red) obtained with optimal reference inputs $\rho_{k,1}^*$ and $\rho_{k,2}^*$, respectively; $y_{k,1}$ and $y_{k,2}$ (in black) are with $\rho_{k,1} = y_{k,1}^d, \rho_{k,2} = y_{k,2}^d$.

The tracking MSEs obtained with the primitive-based mechanism for the first and second axes are $0.44{\cdot}10^{-3}$ and $0.31{\cdot}10^{-3}$, respectively. When $\boldsymbol{P}_1 = \boldsymbol{Y}_1^d$, $\boldsymbol{P}_2 = \boldsymbol{Y}_2^d$, the tracking MSEs for the first and second axis are $6.50{\cdot}10^{-3}$ and $3.75{\cdot}10^{-3}$, respectively. A ten-fold improvement in tracking was realized using the primitive-based approach.

The observed improvement is of no surprise, since the primitive-based tracking solution behaves in an anticipative manner, whereas the tracking based on the L1 control

only is "reactive" and only decides based on the provided reference input. The anticipative behavior of the primitive-based solution inherits its character from the level L2 subjected to the ILC concept. It proves that an optimal behavior can be extrapolated successfully to new trajectories to be tracked, without requiring learning by repetitions with ILC.

7. Implementation Issues and Analysis of the Primitive-Based Framework

One of the most important aspects affecting the entire hierarchical learning performance is the CLS linearity assumption achieved at level L1. Therefore, it is critical to ensure that the CLS matches the linear ORM as accurately as possible. While the mismatch does not affect the L2 level too much because the EDILC displays good learning ability even for some nonlinear CLS (the perturbation-based gradient experiment performed in the vicinity of the nominal trajectory where linearity holds), the level L3 operations have proven to be more sensitive to this mismatch because the optimal reference reconstruction relies on the superposition principle. Therefore, all efforts in learning AIVI NN controllers are justified, both in terms of NN complexity and in terms of the required database size and exploration effort. Better parameterization (more specifically, a linear one using polynomials as basis functions, as opposed to a more generic, such as, e.g., a NN) of the Q-function may need fewer transition samples for learning; however, the downside is that it requires more insight about the underlying controlled system. This insight is still needed in terms of relative degree, system order, minimum-phase character, and in matching the system bandwidth with the linear ORM's bandwidth. Together, these impact the virtual state size (hence the learning space complexity and the function approximators architectural complexity). With careful AIVI design, the NN should be preferred because the underlying training management (e.g., early stopping) allows the more complex NN architectures to use just as much as needed in terms of approximation capacity. Another advantage is automatic feature selection, which is inherently encoded in the NN layers. Yet another aspect of offline AIVI learning is that it does not require online computational power (as in e.g., [46]). The resulting NN controllers take up to a few hours to learn offline; however, they are easily implementable afterwards on hardware with no special computing requirements for typical mechatronic systems. The efforts needed for this L1 level control, also regarded as limitations, are:

- a good estimation of the controlled system's relative degree. A too small degree is equivalent to partial observability and will incur a performance degradation in achieving linearized ORM tracking behavior. A higher degree will add information redundancy which is not a real issue except for the computational costs involving the resulted controller.
- some additional required information leading to a degree of subjectiveness in the design: the underlying system's minimum-phase character, and its bandwidth to be matched with the linear ORM's bandwidth. This information quantification necessitates some experience from the designer and some insight into the process phenomena.
- the controller computational complexity is not seen as an important issue. Indeed, there is more complexity than with a PI/PID controller with few parameters; however, the used NN architectures are shallow, allowing for less than 1ms inference time on a wide range of embedded devices, which would extend the application range of the control. The benefits in terms of nonlinear control performance well balance the costs.

Concerning level L2 learning, it should be used mostly in cases where $\boldsymbol{P}^0 = \boldsymbol{Y}^d$ does not render the output $\boldsymbol{Y}$ close to the basis function shape imposed by $\boldsymbol{Y}^d$. Otherwise, the impact on level L3 is not damaging, since essentially just about any shape of Y will eventually have some approximation capacity. The major question at this stage is how to learn primitives to be useful for level L3. This is discussed among the aspects of the next and final learning levels. The L2 level design is dependent only on the initial solution $\boldsymbol{P}^0$ and the linaerity assumption. The limitations associated with these are

- the L2 level CLS requires resetting conditioned which for some system may be more restrictive. ILC solutions for systems without perfect resetting such stochastic resetting

exists, but not in a model-free style. However, for most mechatronic systems, this is not an issue.
- The initialization for $\boldsymbol{P}^0$ can be optimized using, e.g., (15), for a neal-optimal approximate model-based initialization. This would require fewer EDILC iterations. On the other hand, model-based initialization based on non-causal filtering showed some oscillations at the beginning and at the end of $\boldsymbol{P}^0$ hence it was not applied here.
- The linearity assumption about the CLS. This must be ensured with the L1 level virtual state-feedback control. There is a certain degree of tolerance, as EDILC was applied before on smooth nonlinear systems and the underlying linear system's theory showed effectiveness and learnability.

Most aspects concern level L3 implementation details. The discussion about how to learn the proper primitives starts backward, with the new desired trajectory $\boldsymbol{Y}^d$. From Algorithm 2's steps, $\boldsymbol{Y}^d$ undergoes a transformation process (extrapolation plus padding, then concatenation), resulting in $\boldsymbol{Y}^{d[e]}$ as a single signal. The same is valid when obtaining $\boldsymbol{Y}^{[\theta_\pi e]} = [\boldsymbol{Y}_1^{[\theta_\pi e]^T}, \ldots, \boldsymbol{Y}_p^{[\theta_\pi e]^T}]^T$, except for the additional delay process involved. It is obvious that the final $\boldsymbol{Y}^{[\theta_\pi e]}$ should resemble a basis function (here, Gaussians were used, also known as radial basis functions), which means that only one output channel should be Gaussian and the rest should be zero. Therefore, the practical rule is that if the CLS has p channels, then a number of p primitives should be learned, each time with a different output channel being a Gaussian, while the other outputs are zero.

Learning Gaussian-like shapes as output primitive trajectories is advantageous because a Gaussian is described with only two design parameters: mean (center) and variance (width). With the function approximation, the width of the Gaussian-like output primitive influences the approximation capacity in relation to the shape of $\boldsymbol{Y}^d$: if $\boldsymbol{Y}^d$ is non-smooth, more copies of output primitives should be used when their widths are small. Conversely, when $\boldsymbol{Y}^d$ is smooth, a good approximation is achievable with fewer copied, wider output primitives. The Gaussian polarity (pointing upward or downward) is not important as long as the CLS linearity holds, because a negative weighting coefficient suitably reverses polarity and helps with the approximation.

Typically, the number of copies (parameter M) does not significantly affect the approximation quality. The number of delay/advance steps for each copied primitive (parameter θ) should be chosen from an interval such that the copied and time-shifted output primitives span the entire approximation range of interest for $\boldsymbol{Y}^d$. In summary, the limitations at this particular L3 level include:
- correlating the shape of the basis functions with the shape of the desired trajectory to be tracked on each subinterval.
- selecting the adequate number of copies for the learned primitives. While more is better in the sense that the approximation quality does not increase past a certain number M of copies, it could increase the computational cost for the underlying regression and reference input recomposition. We still argue that the computations are performed online and manageable.
- the tracking has to be of good quality for each subinterval, to avoid large deviations at the end, since this would shift the initial point for trajectory tracking of the next subinterval. This was found to depend largely on the ORM matching (CLS linearization quality) at the level L1.

Concluding, the discussed aspects also reveal the sensitive issues across the proposed framework. At level L1, it is important to ensure exceptional ORM matching for level L3 to accept behavior extrapolation. This is related to AIVI learning, which is strongly tied to exploration quality. This is an open issue in ADP/RL and requires further investigation. At level L2, with an impact on level L3, the basis function shape of the output primitives with respect to the desired trajectory shape is another trade-off and an open issue.

8. Conclusions

The three-level hierarchical cognitive-based learning framework validated here uses several tools from control systems and from machine learning to extrapolate an optimal tracking behavior to new tracking situations by avoiding explicit knowledge of the underlying system dynamics. This framework, inspired by intelligent life, could make progress toward more autonomous and adaptive learning control systems, as required by modern paradigms. It includes all traits of intelligence: memorization and learning from past experiences, generalizing behavior outside of the current knowledge base, and self-improvement in terms of specified performance. The framework's validation on more complex and different systems will prove its potential and may lead to faster adoption and industrial implementation.

Author Contributions: Conceptualization, M.-B.R. and T.L.; methodology, M.-B.R.; software, M.-B.R. and T.L.; validation M.-B.R. and T.L.; formal analysis, M.-B.R. and T.L.; investigation, M.-B.R. and T.L.; writing—original draft preparation, M.-B.R. and T.L.; writing—review and editing, M.-B.R. and T.L.; visualization, T.L.; supervision, M.-B.R.; project administration, M.-B.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI—UEFISCDI, project number PN-III-P1-1.1-TE-2019-1089, within PNCDI III.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

Acronyms	
CLS	closed loop system
CS	control system
EDILC	experiment-driven Iterative Learning Control
ADP	Adaptive Dynamic Programming
IVI	Iterative Value Iteration
NN	neural network
RL	Reinforcement Learning
ORM	Output Reference Model
NMP	non-minimum phase
I/O	input/output
VSFRT	Virtual State-Feedback Reference Tuning
MIMO	multiple input multiple output
SISO	single input single output
PWM	pulse width modulation
PC	personal computer
MSE	mean squared error

Appendix A

Since $\| \boldsymbol{E}(\boldsymbol{P}) \|_2^2 = \boldsymbol{E}^T\boldsymbol{E} = \boldsymbol{E}_1{}^T\boldsymbol{E}_1 + \ldots + \boldsymbol{E}_p{}^T\boldsymbol{E}_p$, the gradient $\partial J/\partial \boldsymbol{P}$ is

$$\frac{\partial J}{\partial \boldsymbol{P}} = \begin{bmatrix} \frac{\partial J}{\partial \boldsymbol{P}_1} \\ \vdots \\ \frac{\partial J}{\partial \boldsymbol{P}_p} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} \frac{\sum_{i=1}^{p} \boldsymbol{E}_i{}^T\boldsymbol{E}_i}{\partial \boldsymbol{P}_1} \\ \vdots \\ \frac{\sum_{i=1}^{p} \boldsymbol{E}_i{}^T\boldsymbol{E}_i}{\partial \boldsymbol{P}_p} \end{bmatrix}. \tag{A1}$$

It is provable by matrix calculus that $\frac{\partial \boldsymbol{E}_i{}^T \boldsymbol{E}_i}{\partial \boldsymbol{P}_j} = 2\left(\boldsymbol{T}^{j,i}\right)^T \boldsymbol{E}_i$, leading to the c.f. gradient

$$\frac{\partial J}{\partial \boldsymbol{P}} = \begin{bmatrix} \frac{\partial J}{\partial \boldsymbol{P}_1} \\ \vdots \\ \frac{\partial J}{\partial \boldsymbol{P}_p} \end{bmatrix} = \frac{2}{N} \underbrace{\begin{bmatrix} \left(\boldsymbol{T}^{1,1}\right)^T \left(\boldsymbol{T}^{1,2}\right)^T \dots \left(\boldsymbol{T}^{1,p}\right)^T \\ \left(\boldsymbol{T}^{2,1}\right)^T \left(\boldsymbol{T}^{2,2}\right)^T \dots \left(\boldsymbol{T}^{2,p}\right)^T \\ \vdots \\ \left(\boldsymbol{T}^{p,1}\right)^T \left(\boldsymbol{T}^{p,2}\right)^T \dots \left(\boldsymbol{T}^{p,p}\right)^T \end{bmatrix}}_{\boldsymbol{T}^T} \begin{bmatrix} \boldsymbol{E}_1 \\ \boldsymbol{E}_2 \\ \vdots \\ \boldsymbol{E}_p \end{bmatrix}. \tag{A2}$$

The gradient-based search (8) is then expanded in a mathematical form for each reference input as

$$\begin{bmatrix} \boldsymbol{P}_1^{j+1} \\ \boldsymbol{P}_2^{j+1} \\ \vdots \\ \boldsymbol{P}_p^{j+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{P}_1^{j} \\ \boldsymbol{P}_2^{j} \\ \vdots \\ \boldsymbol{P}_p^{j} \end{bmatrix} - \frac{2}{N}\boldsymbol{\xi} \begin{bmatrix} \left(\boldsymbol{T}^{1,1}\right)^T \left(\boldsymbol{T}^{1,2}\right)^T \dots \left(\boldsymbol{T}^{1,p}\right)^T \\ \left(\boldsymbol{T}^{2,1}\right)^T \left(\boldsymbol{T}^{2,2}\right)^T \dots \left(\boldsymbol{T}^{2,p}\right)^T \\ \vdots \\ \left(\boldsymbol{T}^{p,1}\right)^T \left(\boldsymbol{T}^{p,2}\right)^T \dots \left(\boldsymbol{T}^{p,p}\right)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{E}_1^{j} \\ \boldsymbol{E}_2^{j} \\ \vdots \\ \boldsymbol{E}_p^{j} \end{bmatrix}. \tag{A3}$$

If $\boldsymbol{T}^{i,l}$ corresponds to $T^{i,l}(q)$, then $\left(\boldsymbol{T}^{i,l}\right)^T$ corresponds to the t.f. obtained by replacing q with $1/q$ in $T^{i,l}(q)$. Herein, $1/q$ must not be mistaken for the t.f. argument q^{-1} as an alternate notation to q. Replacing q qith $1/q$ in a causal t.f. leads to a noncausal one. Extending each vector and matrices from (A3) for $N \to \infty$, we write in the time domain as

$$\begin{bmatrix} \rho_{k,1}^{j+1} \\ \rho_{k,2}^{j+1} \\ \vdots \\ \rho_{k,p}^{j+1} \end{bmatrix} = \begin{bmatrix} \rho_{k,1}^{j} \\ \rho_{k,2}^{j} \\ \vdots \\ \rho_{k,p}^{j} \end{bmatrix} - \frac{2}{N}\boldsymbol{\xi} \underbrace{\begin{bmatrix} T^{1,1}(\frac{1}{q})T^{1,2}(\frac{1}{q}) \dots T^{1,p}(\frac{1}{q}) \\ T^{2,1}(\frac{1}{q})T^{2,2}(\frac{1}{q}) \dots T^{2,p}(\frac{1}{q}) \\ \vdots \\ T^{p,1}(\frac{1}{q})T^{p,2}(\frac{1}{q}) \dots T^{p,p}(\frac{1}{q}) \end{bmatrix}}_{\boldsymbol{T}^T(\frac{1}{q})} \begin{bmatrix} e_{k,1}^{j} \\ e_{k,2}^{j} \\ \vdots \\ e_{k,p}^{j} \end{bmatrix}. \tag{A4}$$

Using the previous equation in the tracking error dynamics across the iterations it follows that

$$\begin{bmatrix} e_{k,1}^{j+1} \\ e_{k,2}^{j+1} \\ \vdots \\ e_{k,p}^{j+1} \end{bmatrix} = \boldsymbol{T}(q) \begin{bmatrix} \rho_{k,1}^{j+1} \\ \rho_{k,2}^{j+1} \\ \vdots \\ \rho_{k,p}^{j+1} \end{bmatrix} - \begin{bmatrix} y_{k,1}^{d} \\ y_{k,2}^{d} \\ \vdots \\ y_{k,p}^{d} \end{bmatrix} = \boldsymbol{T}(q) \left[\begin{bmatrix} \rho_{k,1}^{j} \\ \rho_{k,2}^{j} \\ \vdots \\ \rho_{k,p}^{j} \end{bmatrix} - \frac{2}{N}\boldsymbol{\xi}\boldsymbol{T}^T(\tfrac{1}{q}) \begin{bmatrix} e_{k,1}^{j} \\ e_{k,2}^{j} \\ \vdots \\ e_{k,p}^{j} \end{bmatrix} \right] - \begin{bmatrix} y_{k,1}^{d} \\ y_{k,2}^{d} \\ \vdots \\ y_{k,p}^{d} \end{bmatrix} = \left(\boldsymbol{I}_{p\times p} - \frac{2}{N}\boldsymbol{T}(\tfrac{1}{q})\boldsymbol{\xi}\boldsymbol{T}^T(\tfrac{1}{q}) \right) \begin{bmatrix} e_{k,1}^{j} \\ e_{k,2}^{j} \\ \vdots \\ e_{k,p}^{j} \end{bmatrix}, \tag{A5}$$

where the frequency domain properties of $\| \boldsymbol{I}_{p\times p} - \frac{2}{N}\boldsymbol{T}\left(\frac{1}{q}\right)\boldsymbol{\xi}\boldsymbol{T}^T\left(\frac{1}{q}\right) \|_\infty$ are used for the convergence analysis.

References

1. Wolpert, D.M.; Diedrichsen, J.; Flanagan, J.R. Principles of sensorimotor learning. *Nat. Rev. Neurosci.* **2011**, *12*, 739–751. [CrossRef] [PubMed]
2. Kawamura, S.; Sakagami, N. Analysis on dynamics of underwater robot manipulators basing on iterative learning control and time-scale transformation. In Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, USA, 11–15 May 2002; IEEE: Piscataway, NJ, USA, 2002; Volume 2, pp. 1088–1094.
3. Ijspeert, A.J.; Nakanishi, J.; Schaal, S. Movement imitation with nonlinear dynamical systems in humanoid robots. In Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, USA, 11–15 May 2002; IEEE: Piscataway, NJ, USA, 2002; Volume 2, pp. 1398–1403.
4. Hoelzle, D.J.; Alleyne, A.G.; Wagoner Johnson, A.J. Bumpless transfer for a flexible adaptation of iterative learning control. In Proceedings of the American Control Conference, Can Francisco, CA, USA, 29 June–1 July 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 4305–4311.

5. Schöllig, A.; Hehn, M.; Lupashin, S.; D'Andrea, R. Feasiblity of motion primitives for choreographed quadrocopter flight. In Proceedings of the American Control Conference, Can Francisco, CA, USA, 29 June–1 July 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 3843–3849.
6. Grymin, D.J.; Neas, C.B.; Farhood, M. A hierarchical approach for primitive-based motion planning and control of autonomous vehicles. *Rob. Auton. Syst.* **2014**, *62*, 214–228. [CrossRef]
7. Wang, H.; Zou, Q. B-spline-decomposition-based approach to multiaxis trajectory tracking: Nanomanipulation example. *IEEE Trans. Control Syst. Technol.* **2014**, *22*, 1573–1580. [CrossRef]
8. 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] [PubMed]
9. Li, J.; Li, Z.; Li, X.; Feng, Y.; Hu, Y.; Xu, B. Skill learning strategy based on dynamic motion primitives for human-robot cooperative manipulation. *IEEE Trans. Cogn. Dev. Syst.* **2021**, *13*, 105–117. [CrossRef]
10. Kim, Y.L.; Ahn, K.H.; Song, J.B. Reinforcement learning based on movement primitives for contact tasks. *Robot. Comput. Integr. Manuf.* **2020**, *62*, 101863. [CrossRef]
11. Camci, E.; Kayacan, E. Learning motion primitives for planning swift maneuvers of quadrotor. *Auton. Robots* **2019**, *43*, 1733–1745. [CrossRef]
12. Yang, C.; Chen, C.; He, W.; Cui, R.; Li, Z. Robot learning system based on adaptive neural control and dynamic movement primitives. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 777–787. [CrossRef]
13. Huang, R.; Cheng, H.; Qiu, J.; Zhang, J. Learning physical human-robot interaction with coupled cooperative primitives for a lower exoskeleton. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 1566–1574. [CrossRef]
14. Liu, H.; Li, J.; Ge, S. Research on hierarchical control and optimisation learning method of multi-energy microgrid considering multi-agent game. *IET Smart Grid* **2020**, *3*, 479–489. [CrossRef]
15. Van, N.D.; Sualeh, M.; Kim, D.; Kim, G.W. A hierarchical control system for autonomous driving towards urban challenges. *Appl. Sci.* **2020**, *10*, 3543. [CrossRef]
16. Jiang, W.; Yang, C.; Liu, Z.; Liang, M.; Li, P.; Zhou, G. A hierarchical control structure for distributed energy storage system in DC micro-grid. *IEEE Access* **2019**, *7*, 128787–128795. [CrossRef]
17. Merel, J.; Botvinick, M.; Wayne, G. Hierarchical motor control in mammals and machines. *Nat. Commun.* **2019**, *10*, 5489. [CrossRef]
18. Wu, B.; Gupta, J.K.; Kochenderfer, M. Model primitives for hierarchical lifelong reinforcement learning. *Auton. Agent. Multi. Agent. Syst.* **2020**, *34*, 28. [CrossRef]
19. Chi, R.; Hou, Z.; 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.* **2019**, *49*, 687–696. [CrossRef]
20. Chi, R.; Zhang, H.; Huang, B.; Hou, Z. Quantitative data-driven adaptive iterative learning control: From trajectory tracking to point-to-point tracking. *IEEE Trans. Cybern.* **2020**, 1–15. [CrossRef]
21. Zhang, J.; Meng, D. Convergence analysis of saturated iterative learning control systems with locally lipschitz nonlinearities. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4025–4035. [CrossRef]
22. Li, X.; Chen, S.L.; Teo, C.S.; Tan, K.K. Data-based tuning of reduced-order inverse model in both disturbance observer and feedforward with application to tray indexing. *IEEE Trans. Ind. Electron.* **2017**, *64*, 5492–5501. [CrossRef]
23. Madadi, E.; Söffker, D. Model-free control of unknown nonlinear systems using an iterative learning concept: Theoretical development and experimental validation. *Nonlinear Dyn.* **2018**, *94*, 1151–1163. [CrossRef]
24. Zhang, D.; Wang, Z.; Masayoshi, T. Neural-network-based iterative learning control for multiple tasks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4178–4190. [CrossRef] [PubMed]
25. Zhao, H.; Peng, L.; Yu, H. Data driven distributed bipartite consensus tracking for nonlinear multiagent systems via iterative learning control. *IEEE Access* **2020**, *8*, 144718–144729. [CrossRef]
26. Fu, H.; Chen, X.; Wang, W.; Wu, M. MRAC for unknown discrete-time nonlinear systems based on supervised neural dynamic programming. *Neurocomputing* **2020**, *384*, 130–141. [CrossRef]
27. Wang, W.; Chen, X.; Fu, H.; Wu, M. Data-driven adaptive dynamic programming for partially observable nonzero-sum games via Q-learning method. *Int. J. Syst. Sci.* **2019**, *50*, 1338–1352. [CrossRef]
28. Na, J.; Lv, Y.; Zhang, K.; Zhao, J. Adaptive identifier-critic-based optimal tracking control for nonlinear systems with experimental validation. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, 1–14. [CrossRef]
29. Sardarmehni, T.; Heydari, A. Sub-optimal switching in anti-lock brake systems using approximate dynamic programming. *IET Control Theory Appl.* **2019**, *13*, 1413–1424. [CrossRef]
30. Liu, Y.; Zhang, H.; Yu, R.; Xing, Z. H∞ tracking control of discrete-time system with delays via data-based adaptive dynamic programming. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 4078–4085. [CrossRef]
31. Perrusquia, A.; Yu, W. Neural H_2 control using continuous-time reinforcement learning. *IEEE Trans. Cybern.* **2020**, 1–10. [CrossRef] [PubMed]
32. Huang, M.; Liu, C.; He, X.; Ma, L.; Lu, Z.; Su, H. Reinforcement learning-based control for nonlinear discrete-time systems with unknown control directions and control constraints. *Neurocomputing* **2020**, *402*, 50–65. [CrossRef]
33. Buşoniu, L.; de Bruin, T.; Tolić, D.; Kober, J.; Palunko, I. Reinforcement learning for control: Performance, stability, and deep approximators. *Annu. Rev. Control* **2018**, *46*, 8–28. [CrossRef]

34. Chen, C.; Sun, W.; Zhao, G.; Peng, Y. Reinforcement Q-Learning incorporated with internal model method for output feedback tracking control of unknown linear systems. *IEEE Access* **2020**, *8*, 134456–134467. [CrossRef]
35. Treesatayapun, C. Knowledge-based reinforcement learning controller with fuzzy-rule network: Experimental validation. *Neural Comput. Appl.* **2020**, *32*, 9761–9775. [CrossRef]
36. 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. Part B Cybern.* **2011**, *41*, 14–25. [CrossRef] [PubMed]
37. Wang, Z.; Liu, D. Data-based controllability and observability analysis of linear discrete-time systems. *IEEE Trans. Neural Netw.* **2011**, *22*, 2388–2392. [CrossRef]
38. Ni, Z.; He, H.; Zhong, X. Experimental studies on data-driven heuristic dynamic programming for POMDP. In *Frontiers of Intelligent Control and Information Processing*; Liu, D., Alippi, C., Zhao, D., Zhang, H., Eds.; World Scientific: Singapore, 2014; Chapter 3; pp. 83–106.
39. De Bruin, T.; Kober, J.; Tuyls, K.; Babuska, R. Integrating state representation learning into deep reinforcement learning. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1394–1401. [CrossRef]
40. Yang, Q.; Jagannathan, S. Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2012**, *42*, 377–390. [CrossRef]
41. Radac, M.B.; Lala, T. Robust control of unknown observable nonlinear systems solved as a zero-sum game. *IEEE Access* **2020**, *8*, 214153–214165. [CrossRef]
42. 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. [CrossRef]
43. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
44. 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]
45. Lala, T.; Radac, M.-B. Learning to extrapolate an optimal tracking control behavior towards new tracking tasks in a hierarchical primitive-based framework. In Proceedings of the 2021 29th Mediterranean Conference on Control and Automation, Virtual Conference, 22–25 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 421–427.
46. Radac, M.B.; Precup, R.E. Data-driven model-free tracking reinforcement learning control with VRFT-based adaptive actor-critic. *Appl. Sci.* **2019**, *9*, 1807. [CrossRef]