Concurrent Learning and Lyapunov-based Updates of Deep Neural Networks for Euler-Lagrange Dynamic Systems

Sujata Basyal*, Jonathan Ting*, Kislaya Mishra*, Brendon C. Allen*

***Abstract*—This paper presents a deep neural network (DNN)- and concurrent learning (CL)-based adaptive control architecture for an Euler-Lagrange dynamic system that guarantees system performance for the first time. The developed controller includes two DNNs with the same output-layer weights to ensure feasibility of the control system. In this work, a Lyapunov- and CL-based update law is developed to update the output-layer DNN weights in real-time; whereas, the inner-layer DNN weights are updated offline using data that is collected in real-time. A Lyapunov-like analysis is performed to prove that the proposed controller yields semi-global exponential convergence to an ultimate bound for the output-layer weight estimation errors and for the trajectory tracking errors.**

***Index Terms*—Deep Neural Networks (DNNs), Concurrent Learning (CL), Euler-Lagrange dynamics, Lyapunov methods.**

I. Introduction

The control of a wide-range of uncertain nonlinear systems has been the focus of numerous research efforts over the years (cf. [1]–[6]). One approach to compensate for system uncertainty is to develop robust controllers, which develop control laws that overcome the worst-case uncertainty. However, controller efficiency and overall system performance can be improved for an uncertain system by implementing an adaptive control scheme [7], [8]. To elaborate, adaptive controllers adjust the control output online using available information to achieve a control objective despite uncertainties in the dynamic model [9]–[12].

Neural networks (NNs) and deep neural networks (DNNs) are non-model based adaptive techniques that are capable of compensating for uncertainty in a wide-class of dynamic systems by approximating the uncertain dynamic model [13], [14]. A key distinction between NNs and DNNs is that DNNs have multiple-hidden layers, which allows DNNs to approximate complex continuous functions better than NNs [15]–[17]. The main limitation to the widespread adoption of DNN-based adaptive controllers was that the multi-hidden-layer structure of DNNs had long prevented the integration of DNN-based controllers with a rigorous nonlinear stability analysis, which had prevented performance guarantees [16], [18]. Recent breakthroughs have overcome this DNN limitation by developing Lyapunov-based update laws that adaptively update the output-layer DNN weights in real-time, which has been shown to simultaneously ensure system performance and system responsiveness [19]–[23]. To further improve the DNN learning performance, the results in [19]–[23] updated the DNN input-layers offline using data collected in real-time and conventional data-driven training methods. After sufficient data is collected and the inner-layer DNN weights are re-trained, the updated inner-layer weights can be instantaneously updated in the DNN-based controller. The iterative updates of the inner-layer DNN weights, in turn, improves the output-layer weight estimation performance and yields real-time learning [19].

Despite the fact that the Lyapunov-based DNN update policies developed in [19]–[23] produced an excellent tracking performance (i.e., tracking performance was guaranteed), the update policies cannot guarantee that the output-layer weights will converge towards a small neighborhood around their ideal values. A potential solution is to augment the real-time DNN update law with concurrent learning (CL) inspired terms. CL is a data driven approach that uses recorded data to ensure parameter estimation convergence [24], [25]. Another benefit of CL is that it only requires finite excitation to ensure parameter estimation, which can be achieved in a finite time duration provide that the system is sufficiently excited. In the past, CL has only been applied to dynamic systems that are linearly parametrizable (i.e. dynamic systems that are linear in the uncertain parameters) such that the entire dynamic system can be expressed as $Y\theta = u$, where Y is a measurable regression matrix, θ is a vector of unknown constants, and u is the input into the system [11], [26], [27]. Consequently, an open problem is to develop a DNN- and CL-based adaptive update law for a DNN-based controller.

In this study, a Lyapunov- and DNN-based control framework is developed for an uncertain and nonlinear Euler-Lagrange dynamic system. The Euler-Lagrange dynamic model required two unique DNNs to be developed to ensure feasibility of the control system. Both DNNs are designed to approximate the uncertain terms in the dynamic model and to include the same output-layer weights; however, each DNN includes different inputs. Furthermore, a CL-based adaptive update law is developed to update the output-layer DNN weights in real-time and to ensure that the output-layer weights converge towards their ideal values. Note that combining CL with DNNs removes the linear in the uncertain parameter condition associated with CL, allowing CL to be implemented on a wider class of dynamic systems. Additionally, the inner-layer DNN features are designed to be retrained offline and to be updated iteratively. A non-smooth Lyapunov-based stability analysis is performed to ensure

semi-global exponential convergence to an ultimate bound for the trajectory tracking errors and the output-layer DNN weight estimation errors. Based on the stability analysis, the performance of the proposed controller is guaranteed even if the DNN weights are initially randomized.

II. System Dynamics

Consider an Euler-Lagrange dynamic system modeled as[1]

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F\dot{q} + d(t) = \tau(q,\dot{q},t), \quad (1)$$

where $q : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the generalized position, $\dot{q} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the generalized velocity, $\ddot{q} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the generalized acceleration, $M : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ denotes the inertia matrix, $C : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n\times n}$ denotes the centripetal-Coriolis matrix, $G : \mathbb{R}^n \to \mathbb{R}^n$ denotes the gravitational effects, $F \in \mathbb{R}^{n\times n}$ denotes the viscous damping coefficient matrix, $d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the unknown disturbances, and $\tau : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the generalized input torque. The dynamic system in (1) has the following properties.

Property 1. The inertia matrix $M(q)$ is bounded as $c_m \leq \|M(q)\| \leq c_M$, where c_m, $c_M \in \mathbb{R}_{>0}$ are known constants. Furthermore, $M(q)$ is symmetric and positive definite.

Property 2. The centripetal-Coriolis matrix $C(q,\dot{q})$ is bounded as $\|C(q,\dot{q})\| \leq c_C \|\dot{q}\|$, where $c_C \in \mathbb{R}_{>0}$ is a known constant.

Property 3. The centripetal-Coriolis matrix $C(q,\dot{q})$ satisfies the relationship $C(q,\xi)v = C(q,v)\xi, \quad \forall \xi,\ v \in \mathbb{R}^n$.

Property 4. The terms $G(q)$, F, and $d(t)$ are upper bounded such that $\|G(q)\| \leq c_G$, $\|F\| \leq c_F$, and $\|d(t)\| \leq c_D$, $\forall t \geq t_0$ where c_G, c_F, $c_D \in \mathbb{R}_{>0}$ are known constants and $t_0 \in \mathbb{R}_{\geq 0}$ is the initial time.

Property 5. The inertia matrix and the centripetal-Coriolis matrix satisfy the relationship $\xi^T\left(\frac{1}{2}\dot{M}(q) - C(q,\dot{q})\right)\xi = 0,\ \forall \xi \in \mathbb{R}^n$.

III. Control Design

The control objectives in this work are to track a desired position, denoted by $q_d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, and a desired velocity, denoted by $\dot{q}_d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$.[2]

Property 6. The desired position q_d is a sufficiently smooth signal that is defined by the user. The desired position q_d, its derivative $\dot{q}_d$, and its double derivative $\ddot{q}_d$, are selected to be continuous and bounded such that $q_d,\ \dot{q}_d,\ \ddot{q}_d \in \mathcal{L}_\infty$.

The position tracking error, represented by $e : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, for the dynamics in (1) is defined as

$$e \triangleq q_d - q. \quad (2)$$

[1]For the notational brevity, all explicit dependence on time, t, within the terms $q(t)$, $\dot{q}(t)$, and $\ddot{q}(t)$ is suppressed.

[2]For notational brevity, all functional dependencies are hereafter suppressed unless required for clarity of exposition.

To facilitate the stability analysis, a filtered auxiliary tracking error, represented by $r : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, is defined as

$$r \triangleq \dot{e} + \alpha e, \quad (3)$$

where $\alpha \in \mathbb{R}_{>0}$ is a selectable control gain. Using the tracking error in (2) and the filtered error in (3), a composite error vector, denoted by $z : \mathbb{R}_{\geq 0} \to \mathbb{R}^{2n}$, can be defined as

$$z \triangleq \left[r^T,\ e^T\right]^T. \quad (4)$$

Taking the time derivative of (3), substituting in the definition of the tracking error from (2), multiplying it by the inertia matrix M, and substituting the dynamics in (1) yields the open-loop error system as

$$M\dot{r} = M\ddot{q}_d + C\dot{q} + G + F\dot{q} + \chi - \tau, \quad (5)$$

where $\chi : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes an auxiliary term, defined as $\chi \triangleq M\alpha\dot{e} + d$.

Property 7. Using Property 1, Property 4, and (4), the auxiliary term χ can be upper bounded as $\|\chi\| \leq c_a + c_b\|z\|$, where $c_a,\ c_b \in \mathbb{R}_{>0}$ are known constants.

Property 8. Using Property 2, the errors in (2) and (3), and the definition of z in (4), the centripetal-Coriolis matrix $C(q,\dot{q})$ can be upper bounded as $\|C\| \leq (c_d + c_e\|z\|)$ where $c_d,\ c_e \in \mathbb{R}_{>0}$ are known constants.

A. DNN Approximation

Another control objective of this work is to approximate the uncertain dynamics in (1) using a DNN. To aid the CL development, the uncertain dynamics in (1) can be represented by a single function, denoted as $f_1(x_1) : \mathbb{R}^{3n} \to \mathbb{R}^n$, that is defined as

$$f_1(x_1) \triangleq M\ddot{q} + C\dot{q} + G + F\dot{q}, \quad (6)$$

where $x_1 \triangleq \left[q^T, \dot{q}^T, \ddot{q}^T\right]^T$. Notice that substituting (6) into the dynamics in (1) yields

$$f_1(x_1) + d = \tau. \quad (7)$$

Based on the subsequent analysis, a second DNN must be developed to approximate a function that is denoted as $f_2(x_2) : \mathbb{R}^{3n} \to \mathbb{R}^n$ and is defined as

$$f_2(x_2) = M\ddot{q}_d + C\dot{q} + G + F\dot{q}, \quad (8)$$

where $x_2 \triangleq \left[q^T, \dot{q}^T, \ddot{q}_d^T\right]^T$. Notice that (8) has the same form as (6); however, the acceleration $\ddot{q}$ has been replaced by the desired acceleration $\ddot{q}_d$. Substituting (8) into the open-loop error system in (5) yields

$$M\dot{r} = \chi + f_2(x_2) - \tau. \quad (9)$$

In this work, two DNNs are developed and both are updated using a multiple timescale approach. Specifically, the inner-layer features of each DNN are trained offline using traditional training methods and data is collected concurrent to real-time execution to enable iterative updates of the inner-layer DNN features; whereas, the output-layer DNN weights

are updated in real-time using a Lyapunov- and CL-based update law. Furthermore, previously collected data can be used to pre-train the DNNs; however, the DNNs can also be randomized initially.

Based on the universal function approximation property, continuous functions that lie on compact sets can be approximated by DNNs. Consequently, (6) and (8) can be approximated by DNNs provided that $x_1,\ x_2 \in \Omega$, where $\Omega \subset \mathbb{R}^n$ is a simply connected compact set and $\mathbb{S}^n(\Omega)$ is a space where $f_1(x_1)$ and $f_2(x_2)$ are continuous. Therefore, there exists a DNN (called DNN1) with ideal weights, biases, and activation functions such that $f_1(x_1) \in \mathbb{S}^n(\Omega)$ can be be represented as

$$f_1(x_1) = W^T \sigma(\Phi_1(x_1)) + \epsilon_1(x_1), \tag{10}$$

where $W \in \mathbb{R}^{m\times n}$ denotes unknown ideal DNN output-layer weights for DNN1, $\sigma : \mathbb{R}^p \to \mathbb{R}^m$ denotes the unknown ideal vector of ideal activation functions for DNN1, $\Phi_1 : \mathbb{R}^n \to \mathbb{R}^p$ represents the ideal unknown inner-layers of DNN1 and contains the inner-layer activation functions and weight matrices, and $\epsilon_1 : \mathbb{R}^n \to \mathbb{R}^n$ denotes the unknown reconstruction error of DNN1. The inner-portion of DNN1, Φ_1, can be represented as

$$\Phi_1 = V_{L_1}^T \phi_{L_1}(V_{L_1-1}^T \phi_{L_1-1}(V_{L_1-2}^T \phi_{L_1-2}(...V_1^T \phi_1(x_1)), \tag{11}$$

where $L_1 \in \mathbb{N}$ represents the number of inner-layers associated with DNN1, V_i denotes the weights of the i^{th} inner-layer $\forall i \in [1, L_1]$, and ϕ_i denotes the ideal activation function vector of the i^{th} inner-layer $\forall i \in [1, L_1]$. The DNN estimate of the function f_1, denoted as $\hat{f}_{d1,k} : \Omega \to \mathbb{R}^n$, can be expressed as

$$\hat{f}_{d1,k} = \widehat{W}^T \hat{\sigma}_k \left(\hat{\Phi}_{1,k}(x_1)\right), \tag{12}$$

where $\widehat{W} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{m\times n}$ denotes the estimate of weights associated with the output-layer of the DNN, $k \in \mathbb{N}$ denotes the index for each training iteration of DNN1, $\hat{\sigma}_k : \mathbb{R}^p \to \mathbb{R}^m$ represents the k^{th} user-defined estimate of the activation functions associated with the output-layer of DNN1, and $\hat{\Phi}_{1,k} : \mathbb{R}^n \to \mathbb{R}^p$ represents the k^{th} estimate of the inner-layers of DNN1.

Similar to above, the function $f_2(x_2) \in \mathbb{S}^n(\Omega)$ can be represented using a DNN (called DNN2) as[3]

$$f_2(x_2) = W^T \sigma(\Phi_2(x_2)) + \epsilon_2(x_2), \tag{13}$$

where $\Phi_2 : \mathbb{R}^n \to \mathbb{R}^p$ represents the ideal unknown inner-layers of DNN2, and $\epsilon_2 : \mathbb{R}^n \to \mathbb{R}^n$ denotes the unknown reconstruction error for DNN2. To enable the CL-based DNN update law, the ideal output-layer weights of DNN1, W, and the ideal output-layer activation functions of DNN1, σ, are both used in DNN2. The result of using W and σ in DNN2 (instead of the ideal weights and activation functions for the output-layer of DNN2) is that ϵ_2 is larger. However, it should be noted that DNN1 learns $f_1(x_1)$, DNN2 learns $f_2(x_2)$, and that $f_1(x_1) = f_2(x_2)$ whenever $\ddot{q} = \ddot{q}_d$.

The inner-portion of DNN2, Φ_2, can be represented as

$$\Phi_2 = \overline{V}_{L_2}^T \overline{\phi}_{L_2}(\overline{V}_{L_2-1}^T \overline{\phi}_{L_2-1}(\overline{V}_{L_2-2}^T \overline{\phi}_{L_2-2}(...\overline{V}_1^T \overline{\phi}_1(x_2)), \tag{14}$$

where $L_2 \in \mathbb{N}$ represents the number of inner-layers associated with DNN2, $\overline{V}_i$ denotes the weights of the i^{th} inner-layer $\forall i \in [1, L_2]$, and $\overline{\phi}_i$ denotes the ideal activation function vector of the i^{th} inner-layer $\forall i \in [1, L_2]$. The DNN estimate of the function f_2, denoted as $\hat{f}_{d2,k} : \Omega \to \mathbb{R}^n$, can be expressed as

$$\hat{f}_{d2,i} = \widehat{W}^T \hat{\sigma}_i \left(\hat{\Phi}_{2,i}(x_2)\right), \tag{15}$$

where $i \in \mathbb{N}$ denotes the index for each training iteration of DNN2, $\hat{\sigma}_i : \mathbb{R}^p \to \mathbb{R}^m$ represents the i^{th} user-defined estimate of the activation functions associated with the output-layer of DNN2, and $\hat{\Phi}_{2,i} : \mathbb{R}^n \to \mathbb{R}^p$ represents the i^{th} estimate of the inner-layers of DNN2.

The difference between the output-layer weights of DNN1 and DNN2 and the estimated output-layer weights can represented by $\widetilde{W} : \mathbb{R}_{\geq t_0} \to \mathbb{R}^{m\times n}$, and is defined as

$$\widetilde{W}(t) \triangleq W - \widehat{W}(t). \tag{16}$$

Property 9. There exists known constants $\overline{W}$, $\overline{\sigma}$, $\overline{\hat{\sigma}}_1$, $\overline{\hat{\sigma}}_2$, $\overline{\epsilon}_1$, $\overline{\epsilon}_2 \in \mathbb{R}_{>0}$ that upper bound the ideal output-layer weight matrix for DNN1, W, the unknown vector of ideal activation functions associated with the output-layer of DNN1, σ, the estimates of the unknown activation function vector $\hat{\sigma}_k$ and $\hat{\sigma}_i$ for DNN1 and DNN2, respectively, and the function approximation errors of DNN1, ϵ_1, and DNN2, ϵ_2, respectively, such that $\|W\| \leq \overline{W}$, $\|\sigma\| \leq \overline{\sigma}$, $\|\hat{\sigma}_k\| \leq \overline{\hat{\sigma}}_1, \forall k \in \mathbb{N}$, $\|\hat{\sigma}_i\| \leq \overline{\hat{\sigma}}_2, \forall i \in \mathbb{N}$, $\sup_{x_1(t)\in\Omega} \|\epsilon_1(x_1)\| \leq \overline{\epsilon}_1$, and $\sup_{x_2(t)\in\Omega} \|\epsilon_2(x_2)\| \leq \overline{\epsilon}_2$. Additionally, the DNN1 estimate is selected such that $\sup_{x_1(t)\in\Omega} \left\|\sigma(\Phi(x_1)) - \hat{\sigma}_k\left(\hat{\Phi}_{1,k}(x_1)\right)\right\| \leq \overline{\hat{\sigma}}, \forall k \in \mathbb{N}$, where $\overline{\hat{\sigma}} : \mathbb{R}_{\geq 0}$ represents a known constant.

B. Control Law Development

Substituting (13) into (9) yields a modified open-loop error system as

$$M\dot{r} = \chi + W^T \sigma(\Phi_2(x_2)) + \epsilon_2(x_2) - \tau. \tag{17}$$

Based on the open-loop error system in (17) and the subsequent stability analysis, the controller τ can be designed as

$$\tau \triangleq k_1 r + \left(k_2 + k_3 \|z\| + k_4 \|z\|^2\right) \text{sgn}(r) + e + \hat{f}_{d2,i}, \tag{18}$$

where $k_1, k_2, k_3, k_4 \in \mathbb{R}_{>0}$ are positive user-assigned control gains, and $\text{sgn}(\cdot)$ is the signum function. Substituting the controller τ from (18) into the open-loop error system in

[3] As will subsequently be shown, DNN1 is required to augment the DNN update law with CL-like terms and DNN2 is required to develop an adaptive, real-time, and DNN-based control structure.

(17) yields the closed-loop error system as

$$\begin{aligned} M\dot{r} &= \chi + W^T\sigma\left(\Phi_2\left(x_2\right)\right) + \epsilon_2\left(x_2\right) \\ &\quad -k_1 r - \left(k_2 + k_3\left\|z\right\| + k_4\left\|z\right\|^2\right)\operatorname{sgn}\left(r\right) \\ &\quad -e - \widehat{W}^T\hat{\sigma}_i\left(\hat{\Phi}_{2,i}\left(x_2\right)\right). \end{aligned} \tag{19}$$

Based on the subsequent stability analysis, the output-layer weight estimate law with CL-inspired terms is denoted by $\dot{\widehat{W}} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{m\times n}$ and is designed as

$$\begin{aligned} \dot{\widehat{W}} &= \operatorname{proj}\left(\Gamma\hat{\sigma}_i\left(\hat{\Phi}_{2,i}\left(x_2\right)\right)r^T \right. \\ &\quad +k_{cl}\Gamma\sum_{j=1}^{N}\hat{\sigma}_k\left(\hat{\Phi}_{1,k}\left(x_{1,j}\right)\right)\cdot \\ &\quad \left.\left(\tau_j - \widehat{W}^T\hat{\sigma}_k\left(\hat{\Phi}_{1,k}\left(x_{1,j}\right)\right)\right)^T\right), \end{aligned} \tag{20}$$

where $\Gamma \in \mathbb{R}^{m\times m}$ is a positive definite user-defined gain matrix, $k_{cl} \in \mathbb{R}_{>0}$ is a user-defined control gain, $\operatorname{proj}(\cdot)$ is a smooth projection function (cf. [28]) that ensures that the estimate of output-layer weights remains within user-selected bounds, $\{x_{1,j}, \tau_j\}_{j=1}^{N}$ denotes a history stack of previous states and control inputs corresponding to times $t_j \leq t, \forall j$, and $N \in \mathbb{N}$ represents the size of the history stack. It should be noted that x_1 contains the generalized acceleration $\ddot{q}$, which may be unmeasurable. Due to the fact that only prior values of x_1 must be known (i.e., $x_{1,j}$), the acceleration at each time t_j can be numerically calculated. As per Property 9, the definition of $\widetilde{W}$ in (16), and the update law in (20), $\widetilde{W}$, is upper bounded by a constant $\overline{\widetilde{W}} \in \mathbb{R}_{>0}$, such that $\left\|\operatorname{vec}\left(\widetilde{W}\right)\right\| \leq \overline{\widetilde{W}}$.

Assumption 1. It is assumed that the system is sufficiently excited over a finite duration of time such that $\lambda_{\min}\left(S_{el}\right) > \lambda_{ss} > 0, \forall t \geq T$, where $T \in \mathbb{R}_{>0}$ is a finite time, $\lambda_{\min}\left(\cdot\right)$ is the minimum eigenvalue of $(\cdot)$, $\lambda_{ss} \in \mathbb{R}_{>0}$ is a user-selected constant, and

$$S_{el} = \sum_{j=1}^{N}\hat{\sigma}_k\left(\hat{\Phi}_{1,k}\left(x_{1,j}\right)\right)\hat{\sigma}_k\left(\hat{\Phi}_{1,k}\left(x_{1,j}\right)\right)^T. \tag{21}$$

Data collected prior to an experiment can be used to pre-train the DNN estimates in (12) and (15). Pre-training DNN1 and DNN2 provides initial estimates for the inner-layer features, $\widehat{\Phi}_{1,1}\left(\cdot\right)$ and $\widehat{\Phi}_{2,1}\left(\cdot\right)$, and an initial estimate for the the output-layer weights $\widehat{W}\left(t_0\right)$. As previously stated, the DNNs in this project are updated using a multiple timescale approach. Specifically, the output-layer weights are updated in real-time using the proposed update law in (20). The summation terms in (20) (i.e., the CL-inspired terms) are included in the update law to ensure that the output-layer DNN weights converge towards their optimal values. State and input information are collected in real-time and used in the adaptive update law to train the output-layer DNN weights in real-time and to train the inner-layer DNN weights offline. After the inner-layer DNN weights for DNNs 1 and 2 are sufficiently trained, they are updated instantaneously to generate the updated DNN estimates, $\widehat{\Phi}_{1,k+1}\left(\cdot\right)$ and $\widehat{\Phi}_{2,i+1}\left(\cdot\right)$.

IV. Stability Analysis

To facilitate the subsequent analysis, a common Lyapunov-like function candidate is denoted by $V_L : \mathbb{R}^{n(2+m)} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ and is defined as

$$V_L\left(y,t\right) \triangleq \frac{1}{2}r^T M r + \frac{1}{2}e^T e + \frac{1}{2}\operatorname{trace}\left(\widetilde{W}^T\Gamma^{-1}\widetilde{W}\right), \tag{22}$$

where $y : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n(2+m)}$ is defined as

$$y \triangleq \left[r^T\ e^T\ \operatorname{vec}\left(\widetilde{W}\right)^T\right]^T, \tag{23}$$

where $\operatorname{vec}\left(\cdot\right)$ represents the vectorization operator. The positive definite and continuously differentiable Lyapunov-like function candidate in (22) can be bounded as

$$\beta_1\left\|y\right\|^2 \leq V_L \leq \beta_2\left\|y\right\|^2, \tag{24}$$

where $\beta_1,\ \beta_2 \in \mathbb{R}_{>0}$ are defined as $\beta_1 \triangleq \min\left(\frac{1}{2}\lambda_{\min}\left(M\right), \frac{1}{2}, \frac{1}{2}\lambda_{\min}\left(\Gamma^{-1}\right)\right)$ and $\beta_2 \triangleq \max\left(\frac{1}{2}\lambda_{\max}\left(M\right), \frac{1}{2}, \frac{1}{2}\lambda_{\max}\left(\Gamma^{-1}\right)\right)$, and $\lambda_{\max}$ is the maximum eigenvalue of $(\cdot)$. Moreover, V_L can also be bounded as

$$\eta_1\left\|z\right\|^2 \leq V_L \leq \eta_2\left\|z\right\|^2 + \eta_3, \tag{25}$$

where $\eta_1, \eta_2 \in \mathbb{R}_{>0}$ are defined as $\eta_1 \triangleq \min\left(\frac{1}{2}\lambda_{\min}\left(M\right), \frac{1}{2}\right)$ and $\eta_2 \triangleq \max\left(\frac{1}{2}\lambda_{\max}\left(M\right), \frac{1}{2}\right)$, and $\eta_3 : \mathbb{R}_{>0}$ is defined as

$$\eta_3 \triangleq \frac{1}{2}\lambda_{\max}\left(\Gamma^{-1}\right)\overline{\widetilde{W}}^2. \tag{26}$$

Theorem 1. *For the nonlinear and uncertain dynamic model in (1) that satisfies Properties 1-9 and Assumption 1, the control input in (18) and the adaptation law in (20) ensure that the closed-loop error system in (19) yields a bounded result $\forall t \in [t_0, T)$ in the sense that*

$$\left\|y\left(t\right)\right\| \leq \max\left(\sqrt{\frac{\beta_2}{\beta_1}}\left\|y\left(t_0\right)\right\|, \sqrt{\frac{v_1}{\delta_1\beta_1}}\right), \tag{27}$$

and semi-global exponential convergence to an ultimate bound $\forall t \in [T, \infty)$ in the sense that

$$\begin{aligned} \left\|y\left(t\right)\right\|^2 &\leq \tfrac{\beta_2}{\beta_1}\left\|y\left(T\right)\right\|^2 e^{-\delta_2(t-T)} \\ &\quad + \tfrac{v_2}{\delta_2\beta_1}\left(1 - e^{-\delta_2(t-T)}\right), \end{aligned} \tag{28}$$

provided that the following gain conditions are satisfied

$$\begin{aligned} k_2 &\geq c_a + \bar{\epsilon}_2 + \overline{W}\overline{\sigma} + \overline{W}\bar{\sigma}_2, \\ k_3 &\geq c_e, \\ k_4 &\geq c_b + c_d, \end{aligned} \tag{29}$$

where $\delta_1 \triangleq \frac{\lambda_1}{\eta_2}$, $v_1 \triangleq \frac{\lambda_1\eta_3}{\eta_2} + C_1\overline{\widetilde{W}}$, $\delta_2 \triangleq \frac{\lambda_2}{\beta_2}$, $v_2 \triangleq \frac{C_1^2}{2k_{cl}\lambda_{ss}}$, $\lambda_1 \triangleq \min\left(k_1, \alpha\right)$, $\lambda_2 \triangleq \min\left(k_1, \alpha, \frac{1}{2}k_{cl}\lambda_{ss}\right)$, and $C_1 \in \mathbb{R}_{>0}$ is a known constant.

Proof: Let $y(t)$ be a Filippov solution of the differential inclusion $\dot{y} \in K[h](y)$ for $t \in [t_0, \infty)$, where $K[\cdot]$ is defined in [29] as Filippov's differential inclusion operator and $h : \mathbb{R}^{n(2+m)} \to \mathbb{R}^{n(2+m)}$ is defined as $h(y) \triangleq \left[\dot{r}^T, \dot{e}^T, \quad \text{vec}(\dot{\widehat{W}})^T \right]^T$. Due to the controller being discontinuous, the solution of the time-derivative of V_L exists almost everywhere (a.e.) within $t \in [t_0, \infty)$ such that $\dot{V}_L(y) \overset{\text{a.e.}}{\in} \dot{\tilde{V}}_L(y)$, where $\dot{\tilde{V}}_L$ denotes the generalized time-derivative of (22). The generalized time derivative $\dot{\tilde{V}}_L$ is defined as $\dot{\tilde{V}}_L \subseteq \bigcap_{\xi \in \partial V_L(y)} \xi^T \left[K[h]^T(y), 1\right]^T$, where $\partial V_L(y)$ represents the Clark's generalized gradient and simplifies to $\partial V_L(y) = \nabla V_L(y)$ since $V_L(y)$ is continuously differentiable in y. Note that ∇ denotes the gradient operator.

Taking the generalized time derivative of (22), using properties of the trace operator and the matrix vectorization operator, and substituting the definition of the auxiliary error in (3), the output-layer DNN weight estimation error in (16), the closed-loop error system in (19), and the adaptive update law in (20), yields

$$\begin{aligned}\dot{\tilde{V}}_L \quad \subseteq \quad & \tfrac{1}{2} r^T \dot{M} r + r^T \chi + r^T W^T \sigma\left(\Phi_2(x_2)\right) + r^T \epsilon_2 \\ & - \left(k_2 + k_3 \|z\| + k_4 \|z\|^2\right) r^T K[\text{sgn}(r)] \\ & - k_1 r^T r - r^T \widehat{W}^T K\left[\hat{\sigma}_i\left(\hat{\Phi}_{2,i}(x_2)\right)\right] \\ & - \alpha e^T e - r^T \widetilde{W}^T K\left[\hat{\sigma}_i\left(\hat{\Phi}_{2,i}(x_2)\right)\right] \\ & - \text{trace}\Bigg(k_{cl} \widetilde{W}^T \sum_{j=1}^{N} K\left[\hat{\sigma}_k\left(\hat{\Phi}_{1,k}(x_{1,j})\right)\right] \cdot \\ & \left(\tau_j - \widehat{W}^T K\left[\hat{\sigma}_k\left(\hat{\Phi}_{1,k}(x_{1,j})\right)\right]\right)^T\Bigg).\end{aligned} \tag{30}$$

Considering an arbitrary $i \in \mathbb{N}$ and an arbitrary $k \in \mathbb{N}$, such that $\hat{\sigma}_i$, $\hat{\sigma}_k$, $\hat{\Phi}_{1,k}$, and $\hat{\Phi}_{2,i}$ are continuous. Adding and subtracting $r^T W^T \hat{\sigma}_i\left(\hat{\Phi}_{2,i}(x_2)\right)$ into (30) and using (16) yields

$$\begin{aligned}\dot{\tilde{V}}_L \quad \subseteq \quad & \tfrac{1}{2} r^T \dot{M} r + r^T \chi + r^T \epsilon_2 - k_1 r^T r - \alpha e^T e \\ & + r^T W^T \left[\sigma\left(\Phi_2(x_2)\right) - \hat{\sigma}_i\left(\hat{\Phi}_{2,i}(x_2)\right)\right] \\ & - \left(k_2 + k_3 \|z\| + k_4 \|z\|^2\right) r^T K[\text{sgn}(r)] \\ & - \text{trace}\Bigg(k_{cl} \widetilde{W}^T \sum_{j=1}^{N} \hat{\sigma}_k\left(\hat{\Phi}_{1,k}(x_{1,j})\right) \cdot \\ & \left(\tau_j - \widehat{W}^T \hat{\sigma}_k\left(\hat{\Phi}_{1,k}(x_{1,j})\right)\right)^T\Bigg).\end{aligned} \tag{31}$$

Using Properties 5 and 7-9, noting that $\|r\|^2 \leq \|r\| \|z\|$ and $-r^T K[\text{sgn}(r)] \leq -\|r\|$, and using the gain conditions in (29) allows (31) to be further upper bounded as

$$\begin{aligned}\dot{V}_L \quad \overset{\text{a.e.}}{\leq} \quad & -k_1 \|r\|^2 - \alpha \|e\|^2 \\ & - \text{trace}\Bigg(k_{cl} \widetilde{W}^T \sum_{j=1}^{N} \hat{\sigma}_k\left(\hat{\Phi}_{1,k}(x_{1,j})\right) \cdot \\ & \left(\tau_j - \widehat{W}^T \hat{\sigma}_k\left(\hat{\Phi}_{1,k}(x_{1,j})\right)\right)^T\Bigg).\end{aligned} \tag{32}$$

Adding and subtracting terms into (32) and using (7), (10), (16), (21), Properties 4 and 9, and properties of the trace operator yields

$$\begin{aligned}\dot{V}_L \quad \overset{\text{a.e.}}{\leq} \quad & -k_1 \|r\|^2 - \alpha \|e\|^2 + C_1 \left\|\text{vec}\left(\widetilde{W}\right)\right\| \\ & - \text{trace}\left(k_{cl} \widetilde{W}^T S_{el} \tilde{W}\right).\end{aligned} \tag{33}$$

Since (33) holds for any arbitrary $i, k \in \mathbb{N}$, (33) holds for all $t \in \left[t_0, \infty\right)$. During the time interval $t = \left[t_0, T\right)$, sufficient excitation has not yet been achieved based on Assumption 1 and it can be conservatively assumed that S_{el} is only positive semidefinite $\forall t \in \left[t_0, T\right)$. Therefore, based on the discussion under (20), the definition of λ_1 in the theorem statement, and (4), (33) can be further bounded as

$$\dot{V}_L \overset{\text{a.e.}}{\leq} -\lambda_1 \|z\|^2 + C_1 \overline{\widetilde{W}}, \quad \forall t \in \left[t_0, T\right). \tag{34}$$

Using (24), (25), and the definitions of δ_1 and v_1 in the theorem statement, and solving the differential inequality in (34) yields

$$\begin{aligned}\|y(t)\|^2 \quad \leq \quad & \tfrac{\beta_2}{\beta_1} \|y(t_0)\|^2 e^{-\delta_1 (t - t_0)} \\ & + \tfrac{v_1}{\delta_1 \beta_1}\left[1 - e^{-\delta_1 (t - t_0)}\right], \forall t \in \left[t_0, T\right).\end{aligned} \tag{35}$$

The result in (27) can be obtained by inspection of (35).

During the time interval $t \in [T, \infty)$, sufficient learning has occurred according to Assumption 1. Therefore, the term S_{cl} is positive definite $\forall t \in [T, \infty)$ and it can be proven that

$$-\text{trace}\left(k_{cl} \widetilde{W}^T S_{el} \widetilde{W}\right) \leq -k_{cl} \lambda_{ss} \left\|\text{vec}\left(\widetilde{W}\right)\right\|^2. \tag{36}$$

Using (23), (36), and the definition of λ_2 in the theorem statement, and completing the squares yields an upper bound for (33) as

$$\dot{V}_L \overset{\text{a.e.}}{\leq} -\lambda_2 \|y\|^2 + \frac{C_1^2}{2 k_{cl} \lambda_{ss}}, \quad \forall t \in [T, \infty). \tag{37}$$

Using (24) and the definitions of δ_2 and v_2 in the theorem statement, and solving the differential inequality in (37) yields (28) in the theorem statement.

From (4), (27), and (28), it can be seen that $z, y \in \mathcal{L}_\infty$. Therefore, from (23) and the fact that $y \in \mathcal{L}_\infty$, it is clear that $e, r, \widetilde{W} \in L_\infty$. From (2), (3), and Property 6, it can be seen that $q, \dot{q} \in \mathcal{L}_\infty$. From the projection algorithm in (20), it is clear that $\widehat{W}^T \in L_\infty$, and from Property 9 and 15, it can be seen that $\hat{f}_{d2,i} \in \mathcal{L}_\infty, \forall i \in \mathbb{N}$. Therefore, the controller defined in (18) is bounded.

Lastly, recall that the DNNs in (12) and (15) require x_1 and x_2 to be bounded for all time, respectively. Furthermore,

recall that DNN2 defined in (15) is implemented in real-time in the controller; whereas, DNN1 is implemented using previously collected data in the CL portion of the DNN update law defined in (20). A complication with DNN1 is that x_1 contains $\ddot{q}$; however, since DNN1 is implemented using prior data, the data used in DNN1 (i.e., $x_{1,j}$) can be limited to times (i.e., t_j) when $\|\ddot{q}(t_j)\| \leq C_2$, where $C_2 \in \mathbb{R}_{>0}$ is a user defined constant. Based on (27), (28), and the fact that $\ddot{q}(t_j) \in \mathcal{L}_\infty, \forall t_j$, if $q(t_0), \dot{q}(t_0) \in \mathcal{L}_\infty$ (i.e., a semi-global result), then $q(t), \dot{q}(t) \in \mathcal{L}_\infty, \forall t \in [t_0, \infty)$, which ensures that $x_1(t_j) \in \mathcal{L}_\infty, \forall t_j$ and $x_2(t) \in \mathcal{L}_\infty, \forall t \in [t_0, \infty)$. ■

V. Conclusion

A DNN-based controller with a CL-inspired DNN weight update law was developed for an uncertain, Euler-Lagrange nonlinear dynamic system. To augment the DNN update law with CL-inspired terms, two DNNs had to be developed. The DNNs in this work were designed to have the output-layer weights updated in real-time using a Lyapunov- and CL-based adaptive update law; whereas, the inner-layer weights and biases were designed to be updated offline using traditional DNN training techniques. Real-time data collection was utilized to update the DNN's inner-layers and to implement the CL-inspired terms in the DNN real-time update law. The CL policy in the adaptive controller helps to ensure exponential convergence of the output-layer weights to to a small neighborhood containing the ideal DNN weights. A nonsmooth Lyapunov-like analysis was performed to guarantee semi-global exponential convergence to an ultimate bound for the trajectory tracking errors and the DNN output-layer estimation errors. In the future, simulations and experiments will be performed to further validate the effectiveness of the developed control law.

References

[1] R. A. Freeman and P. V. Kokotovic. *Robust Nonlinear Control Design: State-Space and Lyapunov Techniques*. Birkhäuser, Boston, MA, 1996.

[2] C. Abdallah, D.M. Dawson, P. Dorato, and M. Jamshidi. Survey of robust control for rigid robots. *IEEE Control System Mag.*, 11(2):24–30, 1991.

[3] B. C. Allen, C. A. Cousin, C. A. Rouse, and W. E. Dixon. Robust cadence tracking for switched FES-cycling with an unknown time-varying input delay. *IEEE Trans. Control Syst. Tech.*, 30(2):827–834, 2022.

[4] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, NJ, 3 edition, 2002.

[5] B. C. Allen, K. J. Stubbs, and W. E. Dixon. Robust cadence and power tracking on a switched FES cycle with an unknown electromechanical delay. *IEEE Trans. Control Syst. Tech.*, 31(1):451–458, 2023.

[6] B. C. Allen, K. J. Stubbs, and W. E. Dixon. Robust cadence tracking for switched FES-cycling using a time-varying estimate of the electromechanical delay. *Automatica*, 144, 2022.

[7] Miroslav Krstic, Ioannis Kanellakopoulos, and Peter V. Kokotovic. *Nonlinear and Adaptive Control Design*. John Wiley & Sons, New York, NY, USA, 1995.

[8] Miroslav Krstic and Petar V. Kokotovic. Control Lyapunov functions for adaptive nonlinear stabilization. *Syst. Control Lett.*, 26(1):17–23, 1995.

[9] C. Riano-Rios, R. Bevilacqua, and W. E. Dixon. Adaptive control for differential drag-based rendezvous maneuvers with an unknown target. *Acta Astronautica*, 181:733–740, April 2021.

[10] V. H. Duenas, C. Cousin, V. Ghanbari, and W. E. Dixon. Passivity-based learning control for torque and cadence tracking in functional electrical stimulation FES induced cycling. In *Proc. Am. Control Conf.*, pages 3726–3731, 2018.

[11] B. Allen, K. Stubbs, and W. E. Dixon. Data-based and opportunistic integral concurrent learning for adaptive trajectory tracking during switched fes-induced biceps curls. *IEEE Trans. Neural Syst. Rehabil. Eng.*, 30:2557–2566, 2022.

[12] B. C. Allen, K. J. Stubbs, and W. E. Dixon. Adaptive trajectory tracking during motorized and FES-induced biceps curls via integral concurrent learning. In *Proc. ASME Dyn. Syst. Control Conf.*, 2020.

[13] Ding Wang, Derong Liu, Hongliang Li, and Hongwen Ma. Neural-network-based robust optimal control design for a class of uncertain nonlinear systems via adaptive dynamic programming. *Inf. Sci.*, 282:167–179, 2014.

[14] F. L. Lewis, S. Jagannathan, and A. Yesildirak. *Neural network control of robot manipulators and nonlinear systems*. CRC Press, Philadelphia, PA, 1998.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[16] G. Joshi and G. Chowdhary. Deep model reference adaptive control. In *IEEE Conf. Decis. Control*, pages 4601–4608. IEEE, 2019.

[17] Girish Joshi, Jasvir Virdi, and Girish Chowdhary. Asynchronous deep model reference adaptive control. In *Conf. on Robot Learn.*, 2020.

[18] D. Le, M. Greene, W. Makumi, and W. E. Dixon. Real-time modular deep neural network-based adaptive control of nonlinear systems. *IEEE Control Syst. Lett.*, 6:476–481, 2022.

[19] R. Sun, M. Greene, D. Le, Z. Bell, G. Chowdhary, and W. E. Dixon. Lyapunov-based real-time and iterative adjustment of deep neural networks. *IEEE Control Syst. Lett.*, 6:193–198, 2022.

[20] J. Ting, S. Basyal, and B. C. Allen. Deep neural network based saturated adaptive control of muscles in a lower-limb hybrid exoskeleton. In *ASME Int. Mech. Eng. Congr. Expo. (IMECE)*, 2023.

[21] H. M. Sweatland, B. C. Allen, M. L. Greene, and W. E. Dixon. Deep neural network real-time control of a motorized FES cycle with an uncertain time-varying electromechanical delay. In *ASME Int. Mech. Eng. Congr. Expo. (IMECE)*, 2021.

[22] Girish Joshi, Jasvir Virdi, and Girish Chowdhary. Design and flight evaluation of deep model reference adaptive controller. In *AIAA Scitech 2020 Forum*, page 1336, 2020.

[23] E. Griffis, A. Isaly, D. M. Le, and W. E. Dixon. Deep neural network-based adaptive fes-cycling control: Part ii, a hybrid systems approach. In *Proc. IEEE Conf. Decis. Control*, 2022.

[24] G. Chowdhary and E. Johnson. Concurrent learning for convergence in adaptive control without persistency of excitation. In *Proc. IEEE Conf. Decis. Control*, pages 3674–3679, 2010.

[25] Girish V. Chowdhary and Eric N. Johnson. Theory and flight-test validation of a concurrent-learning adaptive controller. *J. Guid. Control Dynam.*, 34(2):592–607, March 2011.

[26] C. Riano-Rios, R. Bevilacqua, and W. E. Dixon. Differential drag-based multiple spacecraft maneuvering and on-line parameter estimation using integral concurrent learning. *Acta Astronautica*, 174:189–203, September 2020.

[27] Z. Bell, J. Nezvadovitz, A. Parikh, E. Schwartz, and W. Dixon. Global exponential tracking control for an autonomous surface vessel: An integral concurrent learning approach. *IEEE J. Ocean Eng.*, 45(2):362–370, April 2020.

[28] Z. Cai, M. S. de Queiroz, and D. M. Dawson. A sufficiently smooth projection operator. *IEEE Trans. Autom. Control*, 51(1):135–139, January 2006.

[29] A. F. Filippov. Differential equations with discontinuous right-hand side. In *Fifteen papers on differential equations*, volume 42 of *American Mathematical Society Translations - Series 2*, pages 199–231. American Mathematical Society, 1964.