Accelerated Gradient Approach For Deep Neural Network-Based Adaptive Control of Unknown Nonlinear Systems

Duc M. Le, Omkar Sudhir Patil, Cristian F. Nino, and Warren E. Dixon, *Fellow, IEEE*

***Abstract*— Recent connections in the adaptive control literature to continuous-time analogs of Nesterov's accelerated gradient method have led to the development of new real-time adaptation laws based on accelerated gradient methods. However, previous results assume that the system's uncertainties are linear-in-the-parameters (LIP). To compensate for non-LIP uncertainties, our preliminary results developed a neural network (NN)-based accelerated gradient adaptive controller to achieve trajectory tracking for nonlinear systems; however, the development and analysis only considered single-hidden-layer NNs. In this article, a generalized deep NN (DNN) architecture with an arbitrary number of hidden layers is considered, and a new DNN-based accelerated gradient adaptation scheme is developed to generate estimates of all the DNN weights in real-time. A nonsmooth Lyapunov-based analysis is used to guarantee the developed accelerated gradient-based DNN adaptation design achieves global asymptotic tracking error convergence for general nonlinear control affine systems subject to unknown (non-LIP) drift dynamics and exogenous disturbances. A comprehensive set of simulation studies are conducted on a two-state nonlinear system, a robotic manipulator, and a complex 20-D nonlinear system to demonstrate the improved performance of the developed method. Our simulation studies demonstrate enhanced tracking and function approximation performance from both DNN architectures and accelerated gradient adaptation.**

***Index Terms*— Adaptive control, deep neural networks, Lyapunov methods, nonlinear systems, uncertain systems.**

I. Introduction

RECENT advances in deep learning have led to significant impacts across various technological fronts [1]. For example, deep learning has made significant contributions to applications such as natural language processing [2], object detection [3], and reinforcement learning [4], [5], [6]. These advances are attributed to the capability of deep-learning models to embed complex representations of functions into a series of nested parametrized layers of abstraction. Of particular interest in this article is the application of deep NN (DNN) architectures for the control of uncertain nonlinear dynamical systems. DNNs are universal function approximators that are capable of approximating continuous functions to a prescribed accuracy [7], [8], [9]. Typically, numerical optimization-based methods have been used to train DNNs by generating estimates of the DNN weights such that a cost function is minimized over a training dataset [10]. Although DNN training algorithms provide a method to estimate DNN weights, in the context of control of dynamical systems, training is typically performed offline and then held constant during implementation. The resulting controller uses the DNN as a fixed approximate model, with no guarantees on how close the model approximates the actual model experienced during real-time implementation nor what effects such a model mismatch may have on the stability of the controller. Moreover, such offline training often requires a large dataset, which could be expensive or not possible to obtain.

Recent results in [11], [12], [13], [14], and [15] develop the first DNN-based adaptive controllers that enable continuous learning based on weight adaptation laws that are derived from a Lyapunov-based stability analysis. In [11], [12], and [13], Lyapunov-based adaptation laws are developed to adjust the output layer of a fully connected DNN in real-time, while the inner layers are updated discretely using data-driven offline training algorithms. More recent results in [14], [15], and [16] develop the first Lyapunov-based adaptation laws that can be used to enable continuous learning by all the weights of a DNN in real time. In [14], a modular approach is used to develop constraints on the DNN weight adaptation laws, but this approach lacks constructive insights for the adaptive update law. Results in [15] provide the first insights on Lyapunov-derived weight adaptation design for fully connected DNNs which is then generalized to residual neural networks (ResNets) in [16]. The adaptation laws in [11], [12], [13], [14], [15], and [16] use gradient-based adaptation laws that are designed to cancel cross-terms resulting from a first-order Taylor's series approximation. However, as mentioned in [17] and [18], gradient-based adaptation laws may exhibit poor transient performance with oscillations and slow convergence in the weight estimates.

Higher-order accelerated gradient-based optimization methods, such as Nesterov's method [19] and the Heavy-ball method [20], have gained significant interest due to the

Manuscript received 21 November 2022; revised 13 July 2023 and 8 January 2024; accepted 23 April 2024. Date of publication 14 May 2024; date of current version 7 April 2025. This work was supported in part by the Air Force Office of Scientific Research (AFOSR) under Award FA9550-19-1-0169, in part by the Air Force Research Lab (AFRL) under Grant FA8651-21-F-1027, and in part by the Office of Naval Research (ONR) under Grant N00014-21-1-2481. *(Corresponding author: Duc M. Le.)*

Duc M. Le is with Aurora Flight Sciences, a Boeing Company, Cambridge, MA 02142 USA (e-mail: le.duc@aurora.aero).

Omkar Sudhir Patil, Cristian F. Nino, and Warren E. Dixon are with the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: patilomkarsudhir@ufl.edu; cristian1928@ufl.edu; wdixon@ufl.edu).

Digital Object Identifier 10.1109/TNNLS.2024.3395064

improved transient performance and faster convergence rates compared to first-order methods [21].[1] Motivated by the performance capabilities of accelerated gradient-based algorithms, results such as [4], [22], [23], [24], [25], [26], [27], [28], [29], [30] employ Nesterov's method, the Heavy-ball method, or a variant of the acceleration strategy for deep-learning applications. For example, results in [22] develop a nonlinear conjugate gradient-based adaptive momentum algorithm to improve DNN training for image classification. Results in [23] and [25] develop acceleration-based algorithms for a distributed optimization setting. Results in [24] and [28] apply the acceleration strategy for nonsmooth optimization problems. Results in [4] develop accelerated deep reinforcement learning (DRL) algorithms to train an agent to learn an optimal policy that is estimated by a DNN. In the context of DRL, the learned policies dictate how the agent performs sequential decision-making to achieve a desired task. The deliberate acceleration-based design developed in the aforementioned results of [4], [22], [23], [24], [25], [26], [27], [28], [29], and [30] show improved performance and convergence for a variety of deep-learning applications. However, such results are offline methods in the sense that the DNN parameters are learned by solving an optimization problem with a training dataset. As mentioned previously, offline DNN training approaches are ill-suited for the real-time control application investigated in this article.

To build intuition on the acceleration phenomena from higher-order algorithms, results in [31], [32], [33], and [34] make connections between discrete-time accelerated gradient methods and continuous-time analogs. Motivated by the improved convergence properties of accelerated gradient methods, insights from [31], [32], [33], and [34] led to the design and analysis of accelerated gradient algorithms using Lyapunov-based methods [35], [36], [37], [38]. The results in [35] develop an accelerated gradient-based adaptation scheme to estimate a system's parametric uncertainty in applications for model-reference adaptive control with linear dynamics. However, the result in [35] only guarantees asymptotic convergence in the tracking error while the parameter estimation errors are only guaranteed to remain bounded. Results in [36] develop a data-driven accelerated gradient-based method to achieve convergence in the parameter estimates, but this method is only applicable to parameter estimation in linear regression models where a tracking objective is not considered. Moreover, the result in [36] requires measurements of the regression error which is typically not measurable in the context of adaptive control due to the need for higher-order state derivatives. The results in [37] and [38] generalize the accelerated gradient-based adaptation to uncertain nonlinear systems. Results in [37] apply the variational framework in [32] to develop accelerated gradient-based adaptation for general nonlinear systems. Results in [37] also exploit connections between adaptive control and optimization, which provides insights into adaptation design, by comparing gradient- and mirror descent-based adaptation with and without the accelerated gradient method. However, similar to [35], the results in [37] only achieve asymptotic tracking error convergence. The results in [38] develop a data-driven accelerated gradient-based adaptation design to achieve exponential stability in both the tracking and parameter estimation objectives. Unlike [36], the results in [38] are developed for general uncertain Euler–Lagrange systems. Moreover, the need for direct measurements of the parameter estimation error is eliminated by employing a torque-filtering method that exploits algebraic relations in the system dynamics to reconstruct a measurable form of the parameter estimation error. Although the results in [35], [36], [37], and [38] show improved performance from the accelerated-gradient-based adaptation, the underlying assumption throughout the aforementioned results require the system's uncertainty to satisfy the linear-in-the-parameters (LIP) assumption. Hence, scenarios in which a system's dynamics exhibits unstructured or unknown (non-LIP) uncertainties motivate the use of NN-based adaptive control techniques.

In the conference version of this article in [39], the accelerated gradient strategy is used to develop a neural network (NN)-based adaptive controller to compensate for non-LIP model uncertainties. Specifically, an accelerated gradient-based adaptation law is developed to estimate NN weights in real time. However, the NN model considered in [39] is limited to NNs with only a single hidden layer. Building on our preliminary results, this article generalizes the NN model to deep architectures with an arbitrary number of hidden layers. In comparison to our preliminary results in [39], this article generalizes the mathematical development, control design, and stability analysis to account for DNN architectures and unknown exogenous disturbances in the system dynamics. Additionally, the literature review is broadened and comprehensive simulation studies are conducted. In addition to the two-state nonlinear system in [39], simulations were performed on a robotic manipulator and a more complex 20-D system. The simulation studies compare the developed method into both baseline DNN and single hidden-layer NN adaptation schemes.

In this article, a new DNN-based accelerated gradient adaptive controller is developed for trajectory tracking for general nonlinear control-affine systems subject to non-LIP uncertainties and exogenous disturbances. The constructive variational framework in [32] is used to design higher-order accelerated gradient-based adaptation laws for real-time estimation of all the weights of the DNN. The higher-order adaptation scheme is structured as two coupled first-order differential equations; the first adaptation law is used to generate auxiliary estimates of the DNN weights and these estimates are coupled to the second adaptation law which alters the search direction of the auxiliary estimates and adds acceleration to generate the true DNN weight estimates. Despite the potential improvements from the accelerated gradient strategy, the inherent coupled structure poses mathematical challenges with Lyapunov-based analysis. However, the perturbing effects resulting from the cross-terms injected into the analysis by the accelerated gradient-based adaptation strategy is compensated for by the

[1]First-order optimization methods are recursive difference equation driven by a cost function's gradient. Higher-order accelerated methods alter the search direction by using a weighted sum from the previous iteration to add a momentum-based term and accelerate convergence.

developed control input design. Additionally, Lyapunov-based analyses of the DNN-based adaptation poses challenges due to the nested nonlinear parametrization of DNN architectures. However, a recursive relation of the DNN architecture, similar to [15], is developed to facilitate the analysis and derivation of a general expression for the adaptation design that accounts for general fully connected DNNs with an arbitrary number of hidden layers. In comparison to [15], this article restructures the analysis and control development to facilitate the derivation of a general expression to compute the DNN adaptation law. An example is also provided to show the utility of the generalized adaptation law. The switching analysis in [38] is adopted in this article to allow for activation functions with discontinuous gradients, for example, rectified linear unit (ReLU) activation functions. A nonsmooth Lyapunov-based analysis is used to analyze the new accelerated gradient-based DNN adaptive control design. The tracking errors are guaranteed to achieve global asymptotic tracking error convergence despite the presence of non-LIP system uncertainties and exogenous disturbances.

The remainder of this article is organized as follows. Section II introduces the problem formulation and control objective. Specifically, an uncertain control-affine nonlinear system is considered and the control objective is to track a user-defined trajectory subject to the system uncertainty. Moreover, Section II introduces the DNN architecture used in the developed DNN-based accelerated gradient adaptation control scheme. Sections III–IV contain the new contributions of this article. Specifically, Section III includes the designs for the DNN-based adaptive controller and accelerated gradient DNN weight adaptation laws. A nonsmooth Lyapunov-based stability analysis is performed in Section IV to guarantee the tracking objective is achieved. Comparative numerical simulations are provided in Section V to demonstrate the improved performance of the developed DNN-based accelerated gradient-based adaptation design. First, to show the improved performance of the developed method, comparative simulations are conducted on a two-state nonlinear system and a practical robotic manipulator system with the same DNN configuration (i.e., the same number of layers, neurons, activation function, and initial weights). Then, a comprehensive simulation study is conducted on a more complex 20-D nonlinear system that compares various NN architectures (varying hidden-layers and neurons) and adaptation schemes. Additionally, an extension of the developed method to Euler–Lagrange systems are provided in the Appendix.

II. Problem Formulation

A. Mathematical Preliminaries

Let $\mathbb{R}$ and $\mathbb{Z}$ denote the set of real numbers and integers, respectively. Let $\mathbb{R}_{\geq 0} \triangleq [0, \infty)$ and $\mathbb{R}_{>0} \triangleq (0, \infty)$ denote the set of positive and strictly positive real numbers, respectively. Similarly, let $\mathbb{Z}_{\geq 0}$ and $\mathbb{Z}_{>0}$ denote the set of positive and strictly positive integers, respectively. The Euclidean norm of a vector $x \in \mathbb{R}^n$ is denoted by $\|x\| \triangleq \sqrt{x^T x}$. The vector space of essentially bounded Lebesgue measurable functions is denoted by $\mathcal{L}_\infty$. For $n, m \in \mathbb{Z}_{>0}$, let $\mathbb{R}^{n\times m}$ and I_n denote the space of $n \times m$ dimensional matrices and the $n \times n$ dimensional identity matrix, respectively. The right-to-left matrix product operator is denoted by $\overset{\curvearrowleft}{\prod}$, that is, given suitable matrices A_i for all $i = 1, \ldots, m$, $\overset{\curvearrowleft m}{\prod_{i=1}} A_i \triangleq A_m, \ldots, A_2 A_1$ and $\overset{\curvearrowleft m}{\prod_{i=a}} A_i \triangleq I$ if $a > m$. Given matrices $A \in \mathbb{R}^{p\times q}$ and $B \in \mathbb{R}^{r\times s}$, the Kronecker product is denoted by $A \otimes B \in \mathbb{R}^{pr\times qs}$. Let $\text{vec}(\cdot)$ denote the vectorization operator that transforms a matrix into a column vector, that is, for a matrix $A = [a_{i,j}] \in \mathbb{R}^{n\times m}$, $\text{vec}(A) \triangleq [a_{1,1}, \ldots, a_{1,m}, \ldots a_{n,1}, \ldots, a_{n,m}]^T \in \mathbb{R}^{nm}$. Given matrices $A \in \mathbb{R}^{k\times l}$, $B \in \mathbb{R}^{l\times m}$, and $C \in \mathbb{R}^{m\times n}$, the vectorization operator satisfies the following property [40, Proposition 7.1.9]:

$$\text{vec}(ABC) = \left(C^T \otimes A\right) \text{vec}(B). \tag{1}$$

B. Dynamic Model and Control Objective

Consider a control-affine nonlinear system modeled as

$$\dot{x} = f(x) + g(x)u + d(t) \tag{2}$$

where $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the system state, $f : \mathbb{R}^n \to \mathbb{R}^n$ denotes an unknown differentiable drift vector field, $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ denotes a known control effectiveness, $d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes an unknown exogenous disturbance, and $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ denotes a control input. To facilitate the subsequent stability analysis, the following assumptions are made.

Assumption 1: The exogenous disturbance $d(\cdot)$ can be bounded as $\|d(t)\| \leq \overline{d}$ for all $t \in \mathbb{R}_{\geq 0}$, where $\overline{d} \in \mathbb{R}_{>0}$ denotes a known constant.

Assumption 2: The control effectiveness matrix $g(x)$ is full row rank for all $x \in \mathbb{R}^n$.

The control objective is to use an accelerated gradient approach to design a DNN-based adaptive controller to track a user-defined desired trajectory $x_d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ despite uncertainty of the drift vector field in (2). Let the tracking error $e : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ be defined as

$$e \triangleq x - x_d. \tag{3}$$

The desired trajectory and its time derivative are assumed to be continuous and bounded, that is, $x_d(t) \in \Omega$, for all $t \in \mathbb{R}_{\geq 0}$, and $\dot{x}_d \in \mathcal{L}_\infty$, where $\Omega \subset \mathbb{R}^n$ denotes a known compact set.

C. DNN Architecture and Function Approximation

NN function approximation methods are well-suited for systems with unknown or unstructured uncertainties, that is, the uncertainty does not satisfy the typical LIP assumption in adaptive control (see [41], [42], [43]). To compensate for the unknown drift vector field in (2), a fully connected DNN-based feedforward estimate of the drift vector field is introduced in this section. Let the DNN architecture $\Phi : \mathbb{R}^n \times \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \to \mathbb{R}^{L_{k+1}}$ be defined as

$$\Phi(s, \theta) \triangleq \left(V_k^T \sigma_k \circ \cdots \circ V_1^T \sigma_1\right)\left(V_0^T \overline{s}\right) \tag{4}$$

where $\overline{s} \triangleq [s^T, 1]^T \in \mathbb{R}^{L_0+1}$ denotes a concatenated state input that is augmented by 1 to facilitate the inclusion of bias

terms, $\theta \triangleq [\text{vec}(V_k)^T, \ldots, \text{vec}(V_0)^T]^T \in \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}}$ denotes a concatenated vector of the DNN hidden-layer weights $V_i \in \mathbb{R}^{(L_i+1)\times L_{i+1}}$ for all $i \in \{0, \ldots, k\}$, where $k \in \mathbb{Z}_{>0}$ denotes the number of hidden layers in the DNN, $L_i \in \mathbb{Z}_{>0}$ for all $i \in \{0, \ldots, k+1\}$ denotes the number of neurons in each hidden layer, and $\sigma_i : \mathbb{R}^{L_i} \to \mathbb{R}^{L_i+1}$ for all $i \in \{1, \ldots, k\}$ denotes a vector of activation functions. The vector of activation functions can be composed of various activation functions and hence may be represented as $\sigma_i = [\varsigma_{L_i}, \ldots, \varsigma_1, 1]^T$ for all $i \in \{1, \ldots, k\}$, where $\varsigma_j : \mathbb{R} \to \mathbb{R}$ for all $j \in \{1, \ldots, L_i\}$ denotes a piecewise continuously differentiable activation function.[2] Note that for the subsequent function approximation of the unknown drift vector field, the input and output dimensions are defined as $L_0 \triangleq L_{k+1} \triangleq n$.

Let $\mathbb{C}(\Omega)$ denote the space of continuous functions on the set Ω. By the universal function approximation theorem in [44, Th. 3.2], the function space of DNNs is dense in $\mathbb{C}(\Omega)$. Then, for any drift vector field $f \in \mathbb{C}(\Omega)$ and prescribed function reconstruction error bound $\bar{\varepsilon} \in \mathbb{R}_{>0}$, there exist ideal DNN weights $\theta^* \triangleq [\text{vec}(V_k^*)^T, \ldots, \text{vec}(V_0^*)^T]^T \in \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}}$ and activation functions σ_i for all $i \in \{1, \ldots, k\}$ such that $\sup_{x_d\in\Omega} \| f(x_d) - \Phi(x_d, \theta^*)\| \leq \bar{\varepsilon}$. Then, the DNN architecture in (4) models the unknown drift vector field in (2) as

$$f(x_d) = \Phi\left(x_d, \theta^*\right) + \varepsilon(x_d) \quad \forall x_d \in \Omega \tag{5}$$

where $\varepsilon : \mathbb{R}^n \to \mathbb{R}^n$ denotes an unknown bounded function approximation error. The function approximation error is bounded such that $\sup_{x_d\in\Omega}\|\varepsilon(x_d)\| \leq \bar{\varepsilon}$. Since the ideal DNN weights θ^* are unknown, real-time adaptive weight estimates $\hat{\theta} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}}$ are generated to develop a DNN-based adaptive feedforward component $\Phi(x_d, \hat{\theta})$, where $\hat{\theta} \triangleq [\text{vec}(\hat{V}_k)^T, \ldots, \text{vec}(\hat{V}_0)^T]^T$. For brevity, the following short-hand notations are introduced:

$$\Phi^* \triangleq \Phi\left(x_d, \theta^*\right), \quad \hat{\Phi} \triangleq \Phi\left(x_d, \hat{\theta}\right). \tag{6}$$

The DNN weight estimation error $\tilde{\theta} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}}$ is defined as

$$\tilde{\theta}(t) \triangleq \theta^* - \hat{\theta}(t). \tag{7}$$

To facilitate the subsequent stability analysis, the following assumption is made.

Assumption 3 [45, Assumption 1]: The ideal DNN weights can be bounded as $\|\theta^*\| \leq \overline{\theta}$, where $\overline{\theta} \in \mathbb{R}_{>0}$ denotes a known constant.

III. Control Development

This section introduces the DNN-based adaptive control design. Section III-A introduces the closed-loop error system resulting from the DNN-based adaptive controller. The DNN $\hat{\Phi}$ in (6) is used to develop an adaptive feedforward term that compensates for the system uncertainty. Then, in Section III-B, an accelerated gradient approach is used to design an adaptation law for $\hat{\theta}$ in (6) that generates real-time weight estimates for fully connected DNNs with an arbitrary number of hidden layers.

[2]Some common choices in activation functions include the sigmoid, hyperbolic tangent, and ReLu activation functions.

A. Closed-Loop Error System

Based on the subsequent stability analysis, the control input is designed as[3]

$$u \triangleq g^+(x)\left[\dot{x}_d - \beta_e e - \beta_s \text{sgn}(e) - \hat{\Phi} - \rho(\|e\|)e + 2\hat{\Phi}'\left(\hat{\theta} - \hat{\nu}\right)\right] \tag{8}$$

where $\beta_e, \beta_s \in \mathbb{R}_{>0}$ denote user-defined parameters, $\hat{\nu} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}}$ denotes an auxiliary weight estimate that is subsequently defined, $\text{sgn}(\cdot)$ denotes the signum function, $g^+(x) \triangleq g^T(x)(g(x)g(x)^T)^{-1}$ denotes the right pseudo-inverse of $g(x)$,[4] and $\rho : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denotes a known strictly increasing function that satisfies $\|f(x) - f(x_d)\| \leq \rho(\|e\|)\|e\|$ for all $x, x_d \in \mathbb{R}^n$ [49, Lemma 5]. Taking the time derivative of (3), adding and subtracting $f(x_d)$, and substituting in (2) and (5), the open-loop error system can be expressed as

$$\dot{e} = \Phi^* + \varepsilon(x_d) + f(x) - f(x_d) + d(t) - \dot{x}_d + g(x)u. \tag{9}$$

Substituting (8) into (9) yields the closed-loop error system

$$\dot{e} = -k_e e - k_s \text{sgn}(e) + 2\hat{\Phi}'\left(\hat{\theta} - \hat{\nu}\right) + \Phi^* - \hat{\Phi} + \varepsilon(x_d) + f(x) - f(x_d) + d(t) - \rho(\|e\|)e. \tag{10}$$

The first-order Taylor's series approximation of Φ^* yields [45, eq. (22)]

$$\Phi^* - \hat{\Phi} = \hat{\Phi}'\tilde{\theta} + \mathcal{O}^2\left(\left\|\tilde{\theta}\right\|\right) \tag{11}$$

where $\mathcal{O}^2(\cdot)$ denotes higher-order terms resulting from the Taylor's series approximation.[5] Then, substituting (11) into (10), the closed-loop error system can be expressed as

$$\dot{e} = -\beta_e e - \beta_s \text{sgn}(e) + \hat{\Phi}'\tilde{\theta} + 2\hat{\Phi}'\left(\hat{\theta} - \hat{\nu}\right) - \rho(\|e\|)e + \chi \tag{12}$$

where $\chi : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes an auxiliary function defined as $\chi \triangleq \mathcal{O}^2(\|\tilde{\theta}\|) + \varepsilon(x_d) + f(x) - f(x_d) + d(t)$.

B. Accelerated Gradient-Based Adaptation

This section applies the variational framework in [32] to develop an accelerated gradient-based adaptation law that generates real-time weight estimates for the DNN in (8). The variational approach in [32] defines a Bregman–Lagrangian function and uses principles from the calculus of variations to generate a class of accelerated gradient update laws that minimizes a cost functional. Similar to the approach in [32],

[3]The subsequent development is based on the use of the discontinuous signum function, which could lead to a high-frequency response. Results such as [46], [47], [48] could be used as a continuous alternative.

[4]By Assumption 2, the right pseudo-inverse $x \mapsto g^+(x)$ exists for all $x \in \mathbb{R}^n$.

[5]Given suitable functions f and g, the notation $f(x) = \mathcal{O}^k(g(x))$ means that there exist constants $c, x_0 \in \mathbb{R}_{>0}$ such that $\|f(x)\| \leq c\|g(x)\|^k$ for all $x \geq x_0$.

the Bregman–Lagrangian function $\mathscr{L} : \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \times \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ is defined as

$$\mathscr{L}\left(\hat{\theta}, \dot{\hat{\theta}}, t\right) = e^{\overline{\alpha}(t)+\overline{\gamma}(t)}\left(D_h\left(\hat{\theta} + e^{-\overline{\alpha}(t)}\dot{\hat{\theta}}, \hat{\theta}\right) - e^{\overline{\beta}(t)}E\left(\hat{\theta}\right)\right) \tag{13}$$

where $\overline{\alpha}, \overline{\beta}, \overline{\gamma} : \mathbb{R}_{\geq 0} \to \mathbb{R}$ denote arbitrary continuously differentiable scaling functions, $E : \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \to \mathbb{R}$ denotes a user-defined loss function, and $D_h : \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \times \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \to \mathbb{R}_{\geq 0}$ denotes the Bregman divergence defined as $D_h(p,q) \triangleq (1/2)\|q-p\|^2$.[6] The scaling functions in (13) are selected as $\overline{\alpha} \triangleq 0$, $\overline{\beta} \triangleq \ln(\gamma_1\gamma_2)$, and $\overline{\gamma}(t) \triangleq \gamma_2 t$, where $\gamma_1, \gamma_2 \in \mathbb{R}_{>0}$ are user-defined parameters [35], [37] (see [32]). Then, (13) can be expressed as $\mathscr{L}(\hat{\theta}, \dot{\hat{\theta}}, t) = e^{\gamma_2 t}((1/2)\dot{\hat{\theta}}^T\dot{\hat{\theta}} - \gamma_1\gamma_2 E(\hat{\theta}))$. Let a cost functional $J : \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \to \mathbb{R}_{\geq 0}$ be defined as $J(\hat{\theta}) \triangleq \int \mathscr{L}(\hat{\theta}, \dot{\hat{\theta}}, \tau)d\tau$. Then, by the calculus of variations, the trajectories $t \mapsto \hat{\theta}(t)$ that minimize the cost functional J are the solutions of the Euler–Lagrange equation $(d/dt)((\partial\mathscr{L}/\partial\dot{\hat{\theta}})(\hat{\theta}, \dot{\hat{\theta}}, t)) - (\partial\mathscr{L}/\partial\hat{\theta})(\hat{\theta}, \dot{\hat{\theta}}, t) = 0$ [32, eq. (3)]. Then, computing the terms $(\partial\mathscr{L}/\partial\dot{\hat{\theta}})$, $(\partial\mathscr{L}/\partial\hat{\theta})$, and $(d/dt)(\partial\mathscr{L}/\partial\dot{\hat{\theta}})$ in the Euler–Lagrange equation yields the adaptation law

$$\ddot{\hat{\theta}} + \gamma_2\dot{\hat{\theta}} = -\gamma_2\gamma_1\frac{\partial}{\partial\hat{\theta}}E\left(\hat{\theta}\right). \tag{14}$$

Based on the subsequent stability analysis, the loss function is defined as $E(\hat{\theta}) \triangleq (d/dt)((1/2)e^Te)$. Using (12) yields $(\partial E/\partial\hat{\theta}) = \hat{\Phi}'^Te$, where $\hat{\Phi}' \triangleq (\partial\hat{\Phi}/\partial\hat{\theta}) \in \mathbb{R}^{n\times\sum_{i=0}^{k}(L_i+1)L_{i+1}}$. To facilitate the subsequent analysis, let $\gamma_1 \triangleq \gamma_v$ and $\gamma_2 \triangleq \gamma_v\gamma_\theta$, where $\gamma_v, \gamma_\theta \in \mathbb{R}_{>0}$ are user-defined parameters. Then, by defining an auxiliary weight estimate as $\hat{\nu} \triangleq \hat{\theta} + (1/\gamma_2)\dot{\hat{\theta}}$, (14) can be expressed by two first-order differential equations as

$$\dot{\hat{\nu}} \triangleq \text{proj}\left(\gamma_v\hat{\Phi}'^Te\right) \tag{15}$$

$$\dot{\hat{\theta}} \triangleq -\text{proj}\left(\gamma_v\gamma_\theta\left(\hat{\theta} - \hat{\nu}\right)\right) \tag{16}$$

where the operator proj$(\cdot)$ denotes the projection algorithm defined in [41, eq. (E.2)] and is used in (15) and (16) to ensure the weight estimates remain bounded in the subsequent stability analysis, that is, $\hat{\nu}(t), \hat{\theta}(t) \in \Theta$ for all $t \in \mathbb{R}_{\geq 0}$, where $\Theta \triangleq \{\theta \in \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} : \|\theta\| \leq \overline{\theta}\}$ denotes a known convex set and $\overline{\theta}$ is known by Assumption 3.

The term $\hat{\Phi}'$ in (15) can be computed as follows. To facilitate the subsequent development, let $\hat{\sigma}_i : \mathbb{R}^{L_i} \to \mathbb{R}^{L_i+1}$ be defined recursively as

$$\hat{\sigma}_i \triangleq \begin{cases} \sigma_1\left(\hat{V}_0^T\overline{x}_d\right), & i = 1 \\ \sigma_i\left(\hat{V}_{i-1}^T\hat{\sigma}_{i-1}\right), & i = 2, \ldots, k. \end{cases} \tag{17}$$

Using (6) and the recursive relation in (17), the DNN estimate $\hat{\Phi}$ can be expressed as $\hat{\Phi} = \hat{V}_k^T\hat{\sigma}_k$. Then, recalling $\hat{\theta} = [\text{vec}(\hat{V}_k)^T, \ldots, \text{vec}(\hat{V}_0)^T]^T$, $\hat{\Phi}'$ can be expressed as

$$\hat{\Phi}' = \left[\frac{\partial\hat{\Phi}}{\partial\text{vec}\left(\hat{V}_k\right)}, \ldots, \frac{\partial\hat{\Phi}}{\partial\text{vec}\left(\hat{V}_0\right)}\right]. \tag{18}$$

Using the chain rule, the vectorization property in (1), and the facts that $(\partial\hat{\sigma}_1/(\partial\text{vec}(\hat{V}_0))) = \hat{\sigma}_1'(I_{L_1} \otimes \overline{x}_d^T)$ and $(\partial\hat{\sigma}_{i+1}/(\partial\text{vec}(\hat{V}_i))) = \hat{\sigma}_{i+1}'(I_{L_{i+1}} \otimes \hat{\sigma}_i^T)$ for all $i = 1, \ldots, k-1$, the terms in (18) can be computed as

$$\frac{\partial\hat{\Phi}}{\partial\text{vec}\left(\hat{V}_i\right)} = \begin{cases} \left(\prod_{j=1}^{\curvearrowleft k}\hat{V}_j^T\hat{\sigma}_j'\right)\left(I_{L_1} \otimes \overline{x}_d^T\right), & i = 0 \\ \left(\prod_{j=i+1}^{\curvearrowleft k}\hat{V}_j^T\hat{\sigma}_j'\right)\left(I_{L_{i+1}} \otimes \hat{\sigma}_i^T\right), & i = 1, \ldots, k \end{cases} \tag{19}$$

where $\hat{\sigma}_1' \triangleq (\partial\hat{\sigma}_1/\partial\hat{V}_0^T\overline{x}_d)$ and $\hat{\sigma}_i' \triangleq (\partial\hat{\sigma}_i/\partial\hat{V}_{i-1}^T\hat{\sigma}_{i-1})$ for $i = 2, \ldots, k$. The adaptation law in (15), (18), and (19) are expressed for fully connected DNNs with an arbitrary number of hidden layers. To provide more insights, the following example is provided.

Example 1: Consider the single-hidden-layer NN $\hat{\Phi}(x_d, \hat{\theta}) = \hat{V}_1^T\sigma_1(\hat{V}_0^T\overline{x}_d)$ (i.e., $k = 1$), where $\hat{V}_0 \in \mathbb{R}^{(n+1)\times L}$, $\hat{V}_1 \in \mathbb{R}^{(L+1)\times n}$, $L \in \mathbb{Z}_{>0}$, and $\hat{\theta} \triangleq [\text{vec}(\hat{V}_1)^T, \text{vec}(\hat{V}_0)^T]^T \in \mathbb{R}^{2Ln+L+n}$. Then, using (18) and (19), $\hat{\Phi}'$ can be computed as

$$\hat{\Phi}' = \left[\left(I_n \otimes \hat{\sigma}_1^T\right), \hat{V}_1^T\hat{\sigma}_1'\left(I_L \otimes \overline{x}_d^T\right)\right]. \tag{20}$$

Using (20), the update law in (15) yields

$$\dot{\hat{\nu}} = \text{proj}\left(\gamma_v\begin{bmatrix}\left(I_n \otimes \hat{\sigma}_1^T\right)^T \\ \left(\hat{V}_1^T\hat{\sigma}_1'\left(I_L \otimes \overline{x}_d^T\right)\right)^T\end{bmatrix}e\right). \tag{21}$$

Defining $\hat{\nu} \triangleq [\text{vec}(\hat{\nu}_1)^T, \text{vec}(\hat{\nu}_0)^T]^T$, using (1), and applying some algebraic manipulation, (21) can be expressed as

$$\dot{\hat{\nu}}_1 = \text{proj}\left(\gamma_v\hat{\sigma}_1e^T\right) \tag{22}$$

$$\dot{\hat{\nu}}_0 = \text{proj}\left(\gamma_v\overline{x}_de^T\hat{V}_1^T\hat{\sigma}_1'\right). \tag{23}$$

Remark 1: The DNN configuration in Example 1 is a special case where there is only one hidden layer. For the special case, the adaptation laws in (22) and (23) are equivalent to the output- and inner-layer weight adaptation laws developed in [45] for single-hidden-layer NNs. Hence, the adaptation law developed in (15) can be interpreted as a gradient-based adaptation law that has been generalized to fully connected DNNs. However, the fundamental difference in the developed method is that the adaptation law in (15) generates auxiliary weight estimates. The auxiliary estimates are coupled to the adaptation law in (16) which alters the search direction of the auxiliary estimates to generate the true DNN weight estimates.

Remark 2: The adaptation law developed in (14) is dependent on the selection of the loss function. The loss function $E = (1/2)(d/dt)(e^Te)$, defined below (14), was selected

[6] The formulation in [32] considers a non-Euclidean setting and defines the Bregman divergence as $D_h(p,q) = h(p) - h(q) - \nabla h(q)^T(p-q)$, where $h : \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \to \mathbb{R}_{\geq 0}$ denotes a distance-generating function. To obtain the Bregman divergence function defined in this article, $h(x) \triangleq (1/2)\|x\|^2$.

based on the subsequent stability analysis and yields the adaptation laws in (15) and (16). It is well known that including the weight estimation error $\tilde{\theta}$ in the adaptation law design can improve performance, and hence, including a term such as $(1/2)\tilde{\theta}^T\tilde{\theta}$ in the loss function can be beneficial. However, including the weight estimation error in the adaptation design poses challenges in the implementation because the weight estimation error is unknown; therefore, selections in loss functions that incorporate a measurable/computable form of the weight estimation error may be a potential avenue for future works.

IV. Stability Analysis

The closed-loop control design introduced in Section III employs a discontinuous robust sliding mode term and may also have potential discontinuities in $\hat{\Phi}'$ depending on the choice of activation functions (e.g., ReLU activation functions). As a result, the closed-loop system is nonsmooth and does not admit classical solutions. Hence, a nonsmooth Lyapunov-based analysis is used to analyze generalized solutions of the resulting closed-loop system and ensure the tracking objective is achieved [50] ([41, Th. A.8] for LaSalle–Yoshizawa invariance principles for smooth nonautonomous systems). Specifically, the switching analysis in [15] for DNN-based adaptive control is adopted to model the closed-loop system as a state-dependent switched system composed of a finite collection of smooth functions.

To facilitate the subsequent analysis, let $\varrho \in \mathcal{P}$ denote a switching index, where $\mathcal{P} \subset \mathbb{Z}$ denotes a finite set of possible switching indices. Then, the DNN function approximation in (5) can be represented as $f(x_d) = \Phi_\varrho^* + \varepsilon_\varrho(x_d)$, where $x_d \mapsto \Phi_\varrho^*$ is smooth for each $\varrho \in \mathcal{P}$ with the corresponding function reconstruction error $\varepsilon_\varrho(x_d)$. Similarly, the function χ defined below (12) can be represented by the switched function

$$\chi_\varrho \triangleq \mathcal{O}_\varrho^2\left(\left\|\tilde{\theta}\right\|\right) + \varepsilon_\varrho(x_d) + f(x) - f(x_d) + d(t) \quad (24)$$

which is continuous for each $\varrho \in \mathcal{P}$. By the use of the projection algorithm in (16), the DNN weight estimates can be upper bounded as $\|\hat{\theta}(t)\| \leq \overline{\theta}$ for all $t \in \mathbb{R}_{\geq 0}$. Hence, by Assumption 3 and using (7), the DNN weight estimation error can be bounded as $\|\tilde{\theta}(t)\| \leq 2\overline{\theta}$ for all $t \in \mathbb{R}_{\geq 0}$. Moreover, since $\tilde{\theta} \in \mathcal{L}_\infty$ and $\mathcal{O}_\varrho^2$ is continuous in each $\varrho \in \mathcal{P}$, it follows that $\mathcal{O}_\varrho^2$ can be upper bounded by a known constant for all $\varrho \in \mathcal{P}$. The exogenous disturbance $t \mapsto d(t)$ is upper bounded by a known constant by Assumption 1. The terms $f(x) - f(x_d)$ can be bounded as $\|f(x) - f(x_d)\| \leq \rho(\|e\|)\|e\|$ for all $x, x_d \in \mathbb{R}^n$ [49, Lemma 5], where $\rho(\cdot)$ was previously defined below (8). Then, χ_ϱ can be upper bounded as

$$\left\|\chi_\varrho\right\| \leq c + \rho(\|e\|)\|e\| \quad \forall \varrho \in \mathcal{P} \quad (25)$$

where $c \in \mathbb{R}_{>0}$ denotes a known constant and $\rho(\cdot)$ was defined below (8).

Based on the development introduced above, the adaptation law in (15) and the closed-loop error system in (12) can be represented as

$$\dot{\hat{v}} = \text{proj}\left(\gamma_v \hat{\Phi}_\varrho'^T e\right) \quad \forall \varrho \in \mathcal{P} \quad (26)$$

$$\dot{e} = -\beta_e e - \beta_s \text{sgn}(e) + \hat{\Phi}_\varrho' \tilde{\theta} + 2\hat{\Phi}_\varrho'\left(\hat{\theta} - \hat{v}\right) - \rho(\|e\|)e + \chi_\varrho \quad \forall \varrho \in \mathcal{P}. \quad (27)$$

For notational brevity, let $\Psi \in \mathbb{Z}_{>0}$ be defined as $\Psi \triangleq n + 2\sum_{i=0}^{k}(L_i+1)L_{i+1}$. Let $z \triangleq [e^T, (\hat{\theta}-\hat{v})^T, (\theta^*-\hat{v})^T]^T \in \mathbb{R}^\Psi$ denote a concatenated state, and let $\dot{z} = h_\varrho(z,t)$ for all $\varrho \in \mathcal{P}$ denote a collection of subsystems, where $h_\varrho : \mathbb{R}^\Psi \times \mathbb{R}_{\geq 0} \to \mathbb{R}^\Psi$. Then, the corresponding switched system is represented as

$$\dot{z} = h_{p(z)}(z,t) \quad (28)$$

where $z : \mathbb{R}_{\geq 0} \to \mathbb{R}^\Psi$ denotes a Filippov solution to (28), $p : \mathbb{R}^\Psi \to \mathcal{P}$ denotes a state-dependent switching signal, and $h_\varrho(z,t)$ is defined as

$$h_\varrho(z,t) = \begin{bmatrix} f_\varrho^{cl}(z,t) \\ -\text{proj}\left(\gamma_v\gamma_\theta\left(\hat{\theta}-\hat{v}\right)\right) - \text{proj}\left(\gamma_v\hat{\Phi}_\varrho'^T e\right) \\ -\text{proj}\left(\gamma_v\hat{\Phi}_\varrho'^T e\right) \end{bmatrix} \quad (29)$$

for all $\varrho \in \mathcal{P}$, where $f_\varrho^{cl} : \mathbb{R}^\Psi \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is defined as $f_\varrho^{cl}(z,t) \triangleq -\beta_e e - \beta_s \text{sgn}(e) + \hat{\Phi}_\varrho'\tilde{\theta} + 2\hat{\Phi}_\varrho'(\hat{\theta}-\hat{v}) - \rho(\|e\|)e + \chi_\varrho$. In the following theorem, nonsmooth Lyapunov-based analysis techniques developed in [50] are used to establish invariance properties of (28) to ensure the tracking objective is achieved.

Theorem 1: Consider a system modeled as in (2) and let Assumptions 1–3 hold. Then, the control input in (8) and accelerated gradient-based DNN weight adaptation laws in (15) and (16) ensure global asymptotic tracking in the sense that $\lim_{t\to\infty}\|e(t)\| = 0$ and $\lim_{t\to\infty}\|\hat{\theta}(t) - \hat{v}(t)\| = 0$, provided the following sufficient gain condition is satisfied

$$\beta_s > c \quad (30)$$

where c is a known constant defined in (25).

Proof: Consider a candidate common Lyapunov function $V_L : \mathbb{R}^\Psi \to \mathbb{R}_{\geq 0}$ defined as

$$V_L(z) \triangleq \frac{1}{2}e^T e + \frac{1}{2\gamma_v}\left(\hat{\theta}-\hat{v}\right)^T\left(\hat{\theta}-\hat{v}\right) + \frac{1}{2\gamma_v}\left(\theta^*-\hat{v}\right)^T\left(\theta^*-\hat{v}\right) \quad (31)$$

which satisfies the inequality $\underline{\alpha}(\|z\|) \leq V_L(z) \leq \overline{\alpha}(\|z\|)$, where $\underline{\alpha}, \overline{\alpha} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ are continuous positive definite functions. Let $F_\varrho : \mathbb{R}^\Psi \rightrightarrows \mathbb{R}^\Psi$ denote the Filippov regularization of (28) and be defined as $F_\varrho \triangleq \text{K}[h_\varrho](z,t)$, where the calculus of K[·] is defined in [51] and the notation $\rightrightarrows$ denotes a set-valued mapping. Then, the generalized time derivative of (31) can be computed as $\dot{\overline{V}}_L \triangleq \max_{p\in\partial V_L(z)}\max_{q\in F_\varrho'(z)} p^T q$ [50, Definition 3], where $F_\varrho'(z,t) \supseteq F_\varrho(z,t)$ denotes a bound on the regularization of (28), and ∂V_L denotes Clarke's generalized gradient of V_L [52, pp. 39]. Since $z \mapsto V_L(z)$ for all $z \in \mathbb{R}^\Psi$ is continuously differentiable, $\partial V_L(z) = \{\nabla V_L(z)\}$, where $\{\cdot\}$ denotes a singleton set and ∇ denotes the gradient. Additionally, the time derivative of V_L exists for almost all time, that is, $\dot{V}_L(z(t)) \overset{\text{a.e.}}{\in} \dot{\overline{V}}_L(z(t))$, where the notation $\overset{\text{a.e.}}{(\cdot)}$ denotes that the relation holds for almost all time.

Taking the generalized time derivative of (31), using (7), adding and subtracting $\hat{\theta}^T(1/\gamma_v)\mathrm{K}[\dot{\hat{v}}]$, and performing some algebraic manipulation yields

$$\dot{V}_L = e^T \mathrm{K}[\dot{e}] + \left(\hat{\theta} - \hat{v}\right)^T \frac{1}{\gamma_v}\mathrm{K}\left[\dot{\hat{\theta}}\right] - \left(\tilde{\theta} + 2\hat{\theta} - 2\hat{v}\right)^T \frac{1}{\gamma_v}\mathrm{K}\left[\dot{\hat{v}}\right]. \quad (32)$$

Substituting (16), (26), and (27) into (32) yields

$$\begin{aligned}\dot{V}_L = {} & e^T\left(-\beta_e e - \beta_s \mathrm{K}\left[\mathrm{sgn}\right](e) - \rho\left(\|e\|\right)e + \mathrm{K}\left[\chi_\varrho\right]\right) \\ & + e^T\left(\mathrm{K}\left[\hat{\Phi}'_\varrho\right]\tilde{\theta} + 2\mathrm{K}\left[\hat{\Phi}'_\varrho\right]\left(\hat{\theta} - \hat{v}\right)\right) \\ & - \left(\hat{\theta} - \hat{v}\right)^T \mathrm{K}\left[\mathrm{proj}\right]\left(\gamma_\theta\left(\hat{\theta} - \hat{v}\right)\right) \\ & - \left(\tilde{\theta} + 2\hat{\theta} - 2\hat{v}\right)^T \mathrm{K}\left[\mathrm{proj}\right]\left(\hat{\Phi}'^T_\varrho e\right). \end{aligned} \quad (33)$$

Since χ_ϱ and $\hat{\Phi}_\varrho$ are continuous functions for each $\varrho \in \mathcal{P}$, $\mathrm{K}[\chi_\varrho] = \{\chi_\varrho\}$, and $\mathrm{K}[\hat{\Phi}'_\varrho] = \{\hat{\Phi}'_\varrho\}$ for all $\varrho \in \mathcal{P}$. To bind the terms that involve proj$(\cdot)$ in (33), [41, Lemma E.1.IV] is invoked which states $-r^T\Gamma\mathrm{proj}(\tau) \leq -r\Gamma\tau$ for all $r \in \mathcal{R} \subset \mathbb{R}^m$, $\Gamma \in \mathbb{R}^{m\times m}$, and $\tau \in \mathbb{R}^m$, where Γ denotes a positive definite matrix and $\mathcal{R}$ denotes a convex set. The use of the projection algorithm in (15) and (16), the trajectories of $\hat{v}(t)$ and $\hat{\theta}(t)$, for all $t \in \mathbb{R}_{\geq 0}$, remain in the convex set Θ defined below (16). By Assumption 3, the ideal DNN weights can be bounded as $\|\theta^*\| \leq \overline{\theta}$. Moreover, note that $\mathrm{K}[\mathrm{proj}](\tau)$ computes the set of convex combinations of proj(τ) and τ at the points of discontinuity. Therefore, $-r^T\Gamma\mathrm{K}[\mathrm{proj}](\tau) \leq -r^T\Gamma\tau$, and hence, the terms with the proj$(\cdot)$ operator in (33) can be upper bounded as

$$\begin{aligned} & -\left(\hat{\theta} - \hat{v}\right)^T \mathrm{K}\left[\mathrm{proj}\right]\left(\gamma_\theta\left(\hat{\theta} - \hat{v}\right)\right) \\ & \quad \leq -\gamma_\theta\left(\hat{\theta} - \hat{v}\right)^T\left(\hat{\theta} - \hat{v}\right) \quad (34) \\ & -\left(\tilde{\theta} + 2\hat{\theta} - 2\hat{v}\right)^T \mathrm{K}\left[\mathrm{proj}\right]\left(\hat{\Phi}'_\varrho e\right) \\ & \quad \leq -\left(\tilde{\theta} + 2\hat{\theta} - 2\hat{v}\right)^T \hat{\Phi}'^T_\varrho e \\ & \quad = -e^T\left(\hat{\Phi}'_\varrho\tilde{\theta} + 2\hat{\Phi}'_\varrho\left(\hat{\theta} - \hat{v}\right)\right). \quad (35) \end{aligned}$$

Then using (25), (34), (35), and the facts that $e^T\mathrm{K}[\mathrm{sgn}](e) = \{\|e\|_1\}$ and $-\beta_s\|e\|_1 \leq -\beta_s\|e\|$, (33) can be upper bounded as

$$\dot{V}_L \overset{\text{a.e.}}{\leq} -\beta_e\|e\|^2 - (\beta_s - c)\|e\| - \gamma_\theta\left\|\hat{\theta} - \hat{v}\right\|^2. \quad (36)$$

Provided the sufficient gain condition in (30) is satisfied, (36) can be upper bounded as $\dot{V}_L \overset{\text{a.e.}}{\leq} -\beta_e\|e\|^2 - \gamma_\theta\|\hat{\theta} - \hat{v}\|^2$. From (31) and the fact that $\dot{V}_L \overset{\text{a.e.}}{\leq} 0$, it follows that $V_L \in \mathcal{L}_\infty$, which implies $z \in \mathcal{L}_\infty$, and hence $e, \hat{v}, \hat{\theta} \in \mathcal{L}_\infty$. Using (3), the fact that $e, x_d \in \mathcal{L}_\infty$ implies $x \in \mathcal{L}_\infty$. Using (7), the fact that $\theta^*, \hat{\theta} \in \mathcal{L}_\infty$ implies $\tilde{\theta} \in \mathcal{L}_\infty$. The fact that $x_d, \hat{\theta} \in \mathcal{L}_\infty$ implies $\hat{\Phi}, \hat{\Phi}' \in \mathcal{L}_\infty$. Using (8), the fact that $x, \dot{x}_d, e, \hat{\Phi}, \hat{\Phi}', \hat{\theta}, \hat{v} \in \mathcal{L}_\infty$ implies $u \in \mathcal{L}_\infty$. Using (15), the fact that $\hat{\Phi}', e \in \mathcal{L}_\infty$ implies $\dot{\hat{v}} \in \mathcal{L}_\infty$. Using (16), the fact that $\hat{\theta}, \hat{v} \in \mathcal{L}_\infty$ implies $\dot{\hat{\theta}} \in \mathcal{L}_\infty$. Invoking the LaSalle–Yoshizawa theorem extension for nonsmooth systems in [50, Th. 2] yields $\lim_{t\to\infty}\|e(t)\| = 0$ and $\lim_{t\to\infty}\|\hat{\theta}(t) - \hat{v}(t)\| = 0$. ■

V. Simulations

In this section, comprehensive simulation studies were conducted on various nonlinear systems to demonstrate the performance of the developed accelerated DNN adaptive control scheme. In Section V-A, comparative simulations were performed on a two-state nonlinear system. The simulation results compare the baseline DNN adaptive scheme from [15] to the developed method. Similarly, comparative simulations were performed on a two-link robotic manipulator in Section V-B to demonstrate the application of the developed method on a practical system. Then, to provide a comprehensive simulation study on a more complex 20-D nonlinear system, comparative simulations were conducted in Section V-C. The simulation study in Section V-C provides comparative simulations between the standard gradient adaptive controller for single hidden-layer NNs developed in [45], the baseline DNN adaptive scheme from [15], and the developed method.

A. Two-State Nonlinear System

Two comparative simulations were conducted on a two-state nonlinear system to demonstrate the performance of the developed method. The unknown drift vector field in (2) was modeled as [39]

$$f(x) = \begin{bmatrix} -x_1 + x_2 \\ -\frac{1}{2}x_1 - \frac{1}{2}x_2\left(1 - (\cos(2x_1) + 2)^2\right) \end{bmatrix} \quad (37)$$

where $x \triangleq [x_1, x_2]^T \in \mathbb{R}^2$ denotes the system state. The control effectiveness was modeled as $g = I_2$. The exogenous disturbance in (2) was modeled as random noise drawn from the distribution $\mathcal{N}(0, 1)$. Each simulation was conducted for 30 s with initial condition $x(0) = [5.0, 7.0]^T$. The desired trajectory $x_d \triangleq [x_{d,1}, x_{d,2}]^T$ was selected as $x_d(t) = \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}$ for all $t \in \mathbb{R}_{\geq 0}$. The DNN used in both simulations was configured with $L = 10$ neurons in each of the $k = 5$ hidden layers. The hyperbolic tangent activation function was used across all layers. The first simulation was performed with a baseline DNN adaptive controller which used a typical gradient-based scheme given by [15]

$$u \triangleq \dot{x}_d - \beta_e e - \beta_s \mathrm{sgn}(e) - \hat{\Phi} - \rho\left(\|e\|\right)e \quad (38)$$

$$\dot{\hat{\theta}} \triangleq \mathrm{proj}\left(\gamma_\theta\hat{\Phi}'^T e\right) \quad (39)$$

where $e \triangleq [e_1, e_2]^T$, $\hat{\theta} \triangleq [\mathrm{vec}(\hat{V}_5)^T, \ldots, \mathrm{vec}(\hat{V}_0)^T]^T$, and $\hat{\Phi}'$ was computed using (18) and (19). The second simulation was performed with the developed method in (8), (15), and (16). The bound for the projection operator was selected as $\overline{\theta} = 1000$. The robust state feedback term $\rho(\|e\|)$ in (8) was designed as $0.1(\|e\| + \|e\|^2)$ in both simulations were omitted to better focus on the performance resulting from the DNN-based adaptive terms. The DNN weight estimates $\hat{\theta}(0)$ (and also $\hat{v}(0)$ in the second simulation) were initialized randomly from the normal distribution $\mathcal{N}(0, 1)$. The controllers in

TABLE I
CONTROLLER CONFIGURATION

Parameters	DNN Adaptive [15]	Developed Method
Hidden layers, k	5	5
Neurons, L	10	10
Activation function	Hyperbolic tangent	Hyperbolic tangent
γ_ν	-	40
γ_θ	40	0.6
β_s	0.1	0.1
β_e	5	5

TABLE II
RMS TRACKING ERROR, CONTROL EFFORT, AND FUNCTION APPROXIMATION ERROR

RMS	DNN Adaptive	Developed Method
$\overline{e}$	0.0851	0.0277
$\overline{u}$	4.2761	2.6033
$\overline{\tilde{f}}$	2.9878	0.6029

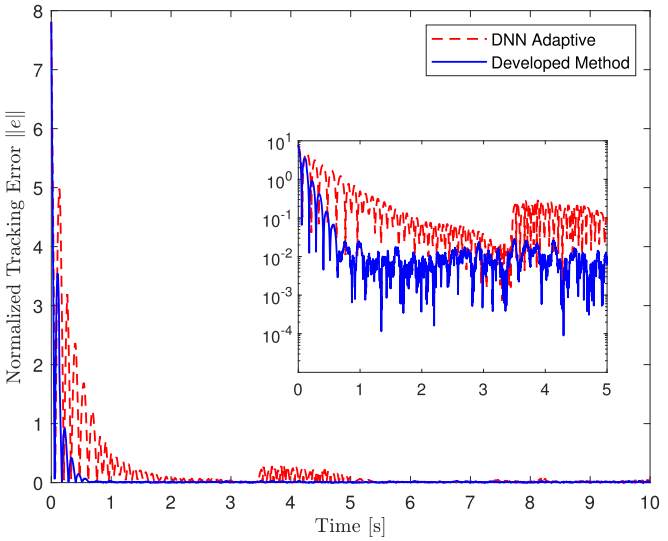

Fig. 1. Evolution of the normalized tracking errors $\|e\|$ for the two-state system. The simulation with the gradient-based DNN adaptation scheme in (38) and (39) is shown as a red dashed line, and the simulation using the developed accelerated gradient-based DNN adaptation scheme in (15) and (16) is shown as a blue solid line. The tracking errors are shown over a 10-s window rather than the entirety of the simulation to better exhibit the transient performance. Additionally, an inset of the simulation from 0 to 5 s is provided. Note the y-axis on the inset is on a logarithmic scale.

each simulation were configured similarly and are summarized in Table I.

Table II summarizes the performance in each simulation. In the leftmost column, $\overline{e}$, $\overline{u}$, and $\overline{\tilde{f}}$ denote the root mean square (rms) tracking error, control effort, and function approximation error, respectively. As shown in Fig. 1, the tracking errors converge toward the origin. The tracking errors using the baseline DNN adaptive controller in the first simulation converged to a neighborhood of the origin after approximately 5.4 s, whereas the tracking errors using the developed method converged after approximately 0.7 s. The rms tracking error for the baseline DNN and developed methods were 0.0851 and 0.0277, respectively. The developed method had a 67.41% decrease in the rms tracking error with a 39.12% decrease in the rms control effort in comparison to the baseline DNN adaptive method.

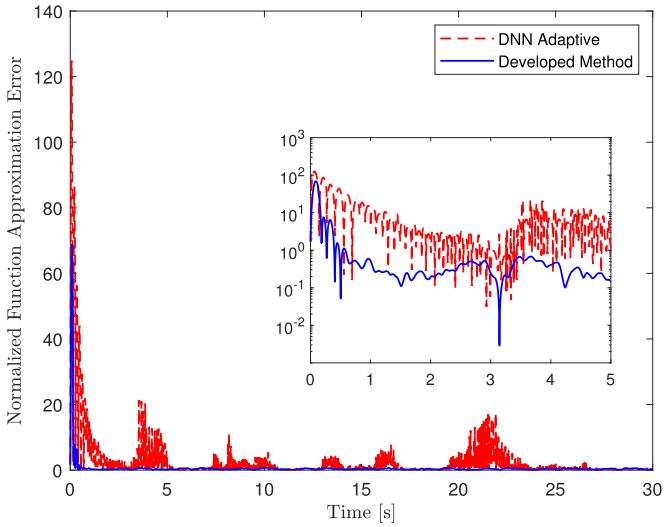

Fig. 2. Evolution of the normalized function approximation errors $\|f(x_d) - \hat{f}(x_d)\|$ for each simulation of the two-state system. Additionally, an inset of the simulation from 0 to 5 s is provided. Note the y-axis on the inset is on a logarithmic scale.

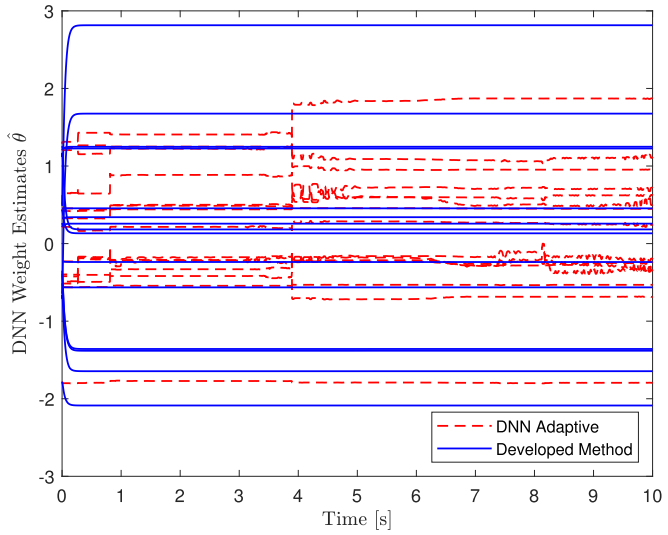

Fig. 3. Evolution of the DNN weight estimates in each simulation of the two-state system. Each of the corresponding weights was initialized at the same initial condition. To better exhibit the performance in the weight estimates, for both simulations, only 15 of the DNN weight estimates are shown over a 10-s window rather than the entirety of the simulation.

Fig. 2 illustrates the normalized function approximation error for each simulation. The simulation with the developed method showed a 78.82% decrease in the rms function approximation error in comparison to the simulation with the baseline DNN adaptive controller. To better illustrate the improved transient performance from the developed accelerated gradient-based adaptation design, Fig. 3 illustrates the evolution of the DNN weight estimates. The weight estimates from the baseline DNN adaptation exhibited oscillatory behavior which degraded the tracking and function approximation performance. As seen at approximately 3.9 s, and many other instances throughout the simulation, many of the weights in the baseline simulation had oscillations which resulted in increases to the tracking and function approximation errors

shown in Figs. 1 and 2, respectively. In comparison, the weight estimates from the developed method had improved transient performance as seen by the rapid convergence in the weight estimates and function approximation error after approximately 1.1 s.

B. Two-Link Robotic Manipulator

Similar to Section V-A, two comparative simulations were performed on a two-link planar revolute robot. The robot was modeled by the uncertain Euler–Lagrange dynamics[7]

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + F\dot{q} = \tau \tag{40}$$

where $q \triangleq [q_1, q_2]^T \in \mathbb{R}^2$, $\dot{q} \in \mathbb{R}^2$, and $\ddot{q} \in \mathbb{R}^2$ denote the angular position, velocity, and acceleration of each joint, respectively, $M(q) \in \mathbb{R}^{2\times 2}$ represents the inertia matrix, $V_m(q,\dot{q}) \in \mathbb{R}^{2\times 2}$ represents the centripetal–Coriolis matrix, $F \in \mathbb{R}^{2\times 2}$ represents friction effects, and $\tau \in \mathbb{R}^2$ denotes the torque inputs. In (40), the dynamics were modeled as [53]

$$M(q) = \begin{bmatrix} p_1 + 2p_3c_2, & p_2 + p_3c_2 \\ p_2 + p_3c_2, & p_2 \end{bmatrix} \tag{41}$$

$$V_m(q,\dot{q}) = \begin{bmatrix} -p_3s_2\dot{q}_2, & -p_3s_2(\dot{q}_1 + \dot{q}_2) \\ p_3s_2\dot{q}_1, & 0 \end{bmatrix} \tag{42}$$

$$F = \begin{bmatrix} f_1, & 0 \\ 0, & f_2 \end{bmatrix} \tag{43}$$

where c_2 denotes $\cos(q_2)$, s_2 denotes $\sin(q_2)$, and the nominal parameters of the two-link robot model in (41)–(43) were $p_1 = 3.473$, $p_2 = 0.196$, $p_3 = 0.242$, $f_1 = 5.3$, and $f_2 = 1.1$. The desired trajectory $q_d(t) \triangleq [q_{d,1}, q_{d,2}]^T \in \mathbb{R}^2$ was selected as $q_d \triangleq (1 - \exp(-0.1t)) \begin{bmatrix} (50\pi/180)\sin((\pi/2)t) \\ (50\pi/180)\sin((\pi/2)t) \end{bmatrix} \in \mathbb{R}^2$ [rad] for all $t \in \mathbb{R}_{\geq 0}$. Each simulation was performed for 45 s with the initial conditions $q(0) = [1.2217, -0.5236]^T$ [rad] and $\dot{q}(0) = [0, 0]^T$ [rad/s]. The DNN used in both simulations was configured similarly as in Section V-A (see hidden layers, neurons, and activation function in Table I). The DNN weight estimates $\hat{\theta}(0)$ (and also $\hat{v}(0)$ in the second simulation) were initialized randomly from the normal distribution $\mathcal{N}(0, 10)$. The first simulation was performed with the baseline DNN adaptive scheme given by [15][8]

$$\tau \triangleq \beta_r r + e + \hat{\Phi} + \beta_s \operatorname{sgn}(r) + \rho(\|y\|)\|y\|\operatorname{sgn}(r) \tag{44}$$

$$\dot{\hat{\theta}} \triangleq \operatorname{proj}\left(\gamma_\theta \hat{\Phi}'^T r\right) \tag{45}$$

where $e, r \in \mathbb{R}^2$ denote the tracking and filtered tracking errors defined in (47) and (48), respectively, $y \triangleq [e^T, r^T]^T \in \mathbb{R}^4$ denotes a concatenated state vector, $\rho(\|y\|) \triangleq 0.1 + 0.1\|y\| + 0.1\|y\|^2$, and $\beta_r \in \mathbb{R}_{>0}$ denotes a user-defined control parameter. The parameters in the first simulation were selected as $\beta_r = 40$, $\beta_s = 3$, $\gamma_\theta = 15$, and $\alpha = 15$. The second simulation was performed with the accelerated DNN adaptive scheme in (51)–(53). The parameters in the second simulation were selected as $\beta_r = 40$, $\beta_s = 3$, $\gamma_\theta = 10$, $\gamma_v = 0.2$ and $\alpha = 15$.

[7]See Appendix for an extension of the developed method to Euler–Lagrange systems.

[8]The DNN adaptive control scheme in [15] applies to nonlinear systems described by (2). The development and analysis on the extension to Euler–Lagrange systems in the Appendix can be used to derive (44) and (45).

TABLE III
RMS TRACKING ERROR, CONTROL EFFORT, AND FUNCTION APPROXIMATION ERROR

RMS	DNN Adaptive	Developed Method
$\overline{e}$ [deg]	0.0118	0.0079
$\overline{\tau}$ [Nm]	14.4857	12.3728
$\overline{\tilde{f}}$	10.4543	8.5989

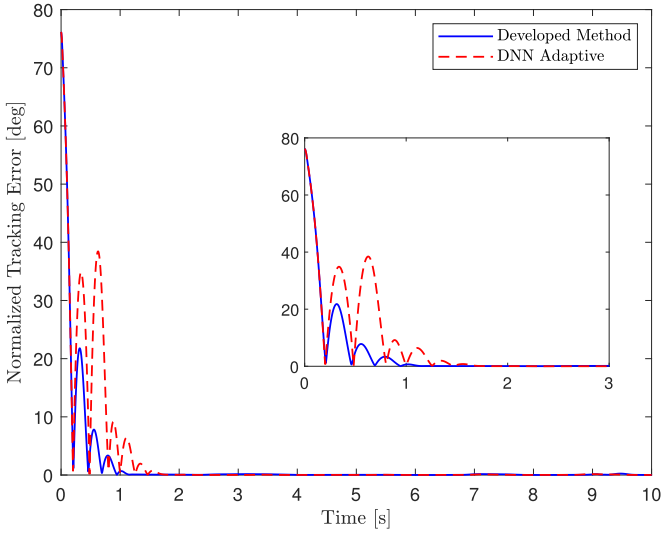

Fig. 4. Evolution of the normalized tracking errors $\|e\|$ for the robotic manipulator. The simulation with the gradient-based DNN adaptation scheme in (44) and (45) is shown as a red dashed line, and the simulation using the developed accelerated gradient-based DNN adaptation scheme in (51)–(53) is shown as a blue solid line. The tracking errors are shown over a 10-s window rather than the entirety of the simulation to better exhibit the transient performance. Additionally, an inset of the simulation from 0 to 3 s is provided.

Table III summarizes the results in each simulation with the robotic manipulator. In the leftmost column, $\overline{e}$, $\overline{\tau}$, and $\overline{\tilde{f}}$ denote the rms tracking error, control effort, and function approximation error, respectively. The developed method showed improvements in the tracking objective, control effort, and function approximation in comparison to the baseline DNN adaptive scheme. Fig. 4 illustrates the normalized function approximation error for each simulation. The tracking errors using the developed method converge to a neighborhood of the origin in approximately 1.2 s, whereas the baseline DNN controller converges in approximately 1.8 s. In comparison to the baseline DNN controller, the rms tracking error using the developed method decreased by 33.31% with 14.59% less control effort. Additionally, the developed method showed a 17.75% improvement in terms of the function approximation errors. The improved function approximation by the developed method is attributed to the weight adaptation which is illustrated in Fig. 5. As seen in the top of Fig. 5, many of the weight estimates using the baseline DNN controller exhibited large oscillations during the transient period approximately from 0 to 1.8 s. In comparison, the weight estimates using the developed method at the bottom of Fig. 5 exhibited better transient performance with milder oscillations. From approximately 0.2–0.4 s, many of the weights in the top of Fig. 5 oscillate from magnitudes of approximately 30–74. The effects of the weight estimate oscillations on the tracking

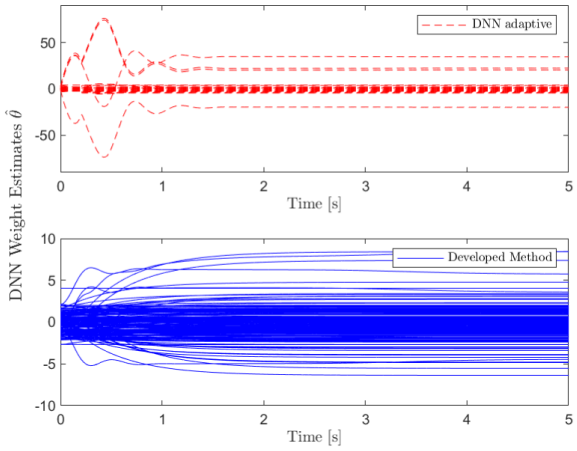

Fig. 5. Evolution of the DNN weight estimates in each simulation of the robot manipulator. Each of the corresponding weights was initialized at the same initial condition. To better exhibit the transient performance in the weight estimates, for both simulations, the estimates are shown over a 5-s window rather than the entirety of the simulation.

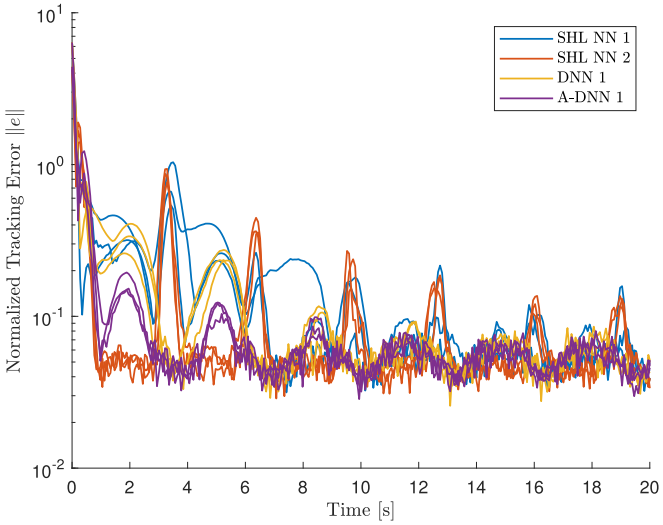

Fig. 6. Evolution of the normalized tracking error $\|e\|$ for the 20-D system in Case 1. The lines labeled "SHL NN 1" and "SHL NN 2" represent the simulations with an SHL NN adaptive controller, the line labeled "DNN 1" represents the baseline gradient-based adaptation scheme in (38) and (39), and the line labeled "A-DNN 1" represents the accelerated gradient-based DNN adaptation scheme in (15) and (16). To better illustrate the results visually, the tracking errors are shown over a 20-s window rather than the entirety (45 s) of the simulation, only 3 out of 25 simulations for each simulation configuration are shown, and the data points were downsampled.

error are evident as the tracking errors during the transient period are higher using the baseline DNN adaptive scheme. Corresponding to one of the peak oscillations, in Fig. 4, at approximately 0.4 s, the normalized tracking error using the baseline DNN scheme is 34.87°, whereas the error using the developed method was 21.79°.

C. Comprehensive Case Studies on 20-D Nonlinear System

In this section, a comprehensive simulation study was conducted on a more complex 20-D nonlinear system. The unknown drift vector field in (2) with $n = 20$ dimensions was modeled as $f(x) = A\phi(x)$, where $A \in \mathbb{R}^{n\times 6n}$ was a random matrix, and the nonlinear function $\phi : \mathbb{R}^n \to \mathbb{R}^{6n}$ was defined as[9] $\phi(x) \triangleq [1^T_{n\times 1}, x^T, (x \odot x)^T, (x \odot x \odot x)^T, \tanh(c_1 x), \sin(c_2 x)]^T$. The entries $a_{i,j}$ of the random matrix A were drawn from the normal distribution $\mathcal{N}(0, 1)$ and the constants c_1 and c_2 were drawn from $\mathcal{N}(0, 1)$. The desired trajectory was selected as $x_d(t) = [\sin(t), \ldots, \sin(t)]^T \in \mathbb{R}^n$.

1) Configurations: The simulation studies in this section were performed on two cases. In the first case (Case 1), to demonstrate the performance of both the acceleration strategy and DNN architectures, comparative simulations were conducted between three different NN-based adaptive controllers: the standard gradient adaptive controller for single hidden-layer NNs developed in [45], the baseline DNN adaptive scheme from [15], and the developed method in this article. The single hidden-layer NN-based adaptive controller is a special case of the baseline DNN controller in (38) and (39), where the NN architecture $\hat{\Phi}$ has $k = 1$ hidden layers. Two sets of simulations with different NN configurations were performed with the single hidden-layer NN adaptive controller: one set of simulations was configured with a single neuron (will be referred to as SHL NN 1), and the other had 5 neurons (SHL NN 2). Similar to Section V-A, the baseline DNN adaptive scheme used is shown in (38) and (39), and the developed method is in (8), (15), and (16). For both the baseline DNN adaptation scheme and the developed method, a set of simulations was conducted with DNNs configured with 5 layers and 1 neuron in each layer (i.e., $k = 5$ and $L_i = 1$ for $i = 1, \ldots, k$), and each of these sets of simulations will be referred to as DNN 1 and A-DNN 1, respectively. For each set of simulations (SHL NN 1, SHL NN 2, DNN 1, and A-DNN 1), 25 simulations were conducted for 45 s where each simulation had varying random initial conditions, that is, in every simulation, the NN weights and states were randomized from the normal distribution $\mathcal{N}(0, 1)$. Note that between all simulation configurations (i.e., between SHL NN 1, SHL NN 2, DNN 1, and A-DNN 1), the initial states were initialized at the same random initial conditions. The simulation configurations (NN architectures and controller parameters) are summarized in Table IV. Similarly, in the second case (Case 2), comparative simulations between the baseline DNN adaptive controller and the developed method was conducted for various DNN configurations, that is, the number of layers and neurons. The methodology of the simulation studies in Case 2 are similar to Case 1, and the configurations of each set of simulations is summarized in Table IV.

2) Results and Discussion: Table V summarizes the results of Case 1 with the 20-D nonlinear system. Fig. 6 illustrates the normalized tracking errors for a representative subset of simulations from each of the configurations. Additionally, Fig. 7 illustrates the normalized function approximation error for each configuration. Case 1 demonstrates the performance benefits of leveraging the DNN architectures and the accelerated gradient adaptation developed in this article. The simulations in SHL 2 demonstrate adding more neurons in

[9]The operator $\odot$ denotes the element-wise product operator. For example, given vectors $u = [u_1, \ldots, u_n]^T \in \mathbb{R}^n$ and $u = [v_1, \ldots, v_n]^T \in \mathbb{R}^n$, the element-wise product $u \odot v = [u_1 v_1, \ldots, u_n v_n]^T$.

TABLE IV

CONTROLLER CONFIGURATIONS†

Parameters	Case 1				Case 2			
	SHL NN 1	SHL NN 2	DNN 1	A-DNN 1	DNN 2	A-DNN 2	DNN 3	A-DNN 3
Hidden layers, k	1	1	5	5	5	5	20	20
Neurons, L	5	20	1	1	5	5	5	5
γ_ν	-	-	-	0.09	-	0.09	-	0.018
γ_θ	2	5	5	10	5	10	2	5
β_s	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
β_e	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5

† SHL NN (#) represents the single hidden-layer NN adaptation scheme developed in [45], DNN (#) represents the baseline DNN adaptation scheme developed in [15], and A-DNN (#) represents the developed method of this paper. In all simulations, the hyperbolic tangent activation function was used.

TABLE V

CASE 1 RESULTS: AVERAGE TRACKING ERROR, CONTROL EFFORT, AND FUNCTION APPROXIMATION ERROR†

Config.	$\bar{e}$	$\bar{u}$	$\bar{\tilde{f}}$	$\Delta\bar{e}$‡	$\Delta\bar{u}$	$\Delta\bar{\tilde{f}}$
SHL NN 1	0.2191	6.6425	3.2990	-	-	-
SHL NN 2	0.1984	7.1740	4.0018	-9.48	8.00	21.30
DNN	0.1937	6.3838	1.7866	-11.54	-3.90	-45.84
A-DNN	0.1815	6.5749	1.5310	-17.20	-1.02	-55.35

† For each configuration, 25 simulations were conducted at different random initial conditions (state and initial weights). The averages are calculated across all simulations for each configuration.

‡ The symbol $\Delta(\cdot)$ represents the relative percent change of $(\cdot)$. The relative percent change for the average tracking error, control input, and function approximation error are relative to SHL NN 1.

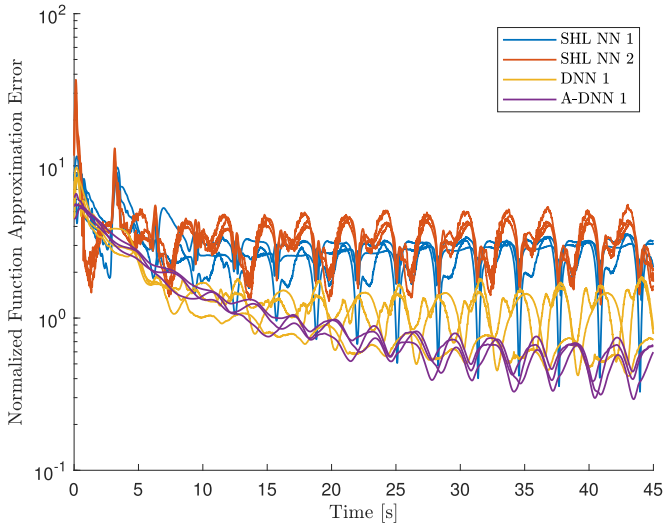

Fig. 7. Evolution of the normalized function approximation errors $\|f(x_d) - \hat{f}(x_d)\|$ for the 20-D system in Case 1. To better illustrate the results visually, only 3 out of 25 simulations for each simulation configuration is shown.

a hidden layer degraded the function approximation performance. The function approximation performance was 21.30% higher compared to SHL 1. Although the tracking performance in SHL 2 improved by 9.78%, the improvement is not indicative of better overall performance. As seen in Fig. 6, the tracking error exhibited high oscillatory behavior with larger amplitude errors throughout the simulations. For example, approximately from 1 to 3 s, the tracking errors resided at approximately 0.05 and then drastically increased to 0.9, which occurred throughout the simulations in correspondence to the poor function approximation. Alternatively, with the DNN architectures, both the DNN 1 and A-DNN 1 simulation

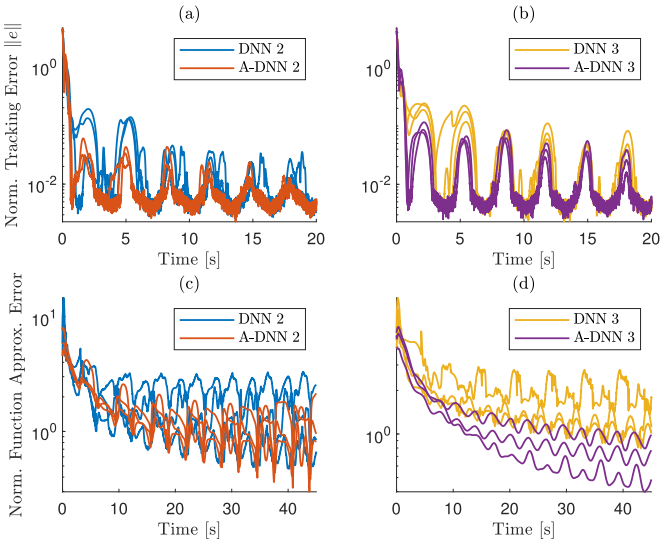

Fig. 8. (a) and (c) Evolutions of the normalized tracking and function approximation errors for the 20-D system in Case 2 for the DNN 2 and A-DNN 2 configurations. Similarly, (b) and (d) illustrate the normalized tracking and function approximation errors for DNN 3 and A-DNN 3. To better illustrate the results visually, the tracking errors are shown over a 20-s window rather than the entirety (45 s) of the simulation and only 3 out of 25 simulations for each simulation configuration are shown.

sets outperformed SHL NN 1 and SHL NN 2. The tracking performance in DNN 1 improved by 11.54% and 2.28% in comparison to SHL NN 1 and SHL NN 2, respectively. Similarly, in terms of the function approximation performance, DNN 1 improved by 45.84% and 55.35%, respectively. These results demonstrate the benefits of DNN architectures and their improved function approximation capabilities. In the final simulation set of Case 1, A-DNN 1 demonstrates the benefits of leveraging both DNN architectures along with accelerated gradient adaptation. In comparison to DNN 1, the tracking and function approximation errors improved by 6.40% and 14.31%.

The simulation study results for Case 2 are summarized in Table VI. The normalized tracking and function approximation errors are shown in Fig. 8(a) and (c), respectively, for both for simulations with DNN 2 and A-DNN 2. Similarly, DNN 3 and A-DNN 3 are shown in Fig. 8(b) and (d). In both the simulation sets with A-DNN 2 and A-DNN 3, tracking and function approximation performance improved in comparison to DNN 2 and DNN 3, respectively. In comparison to DNN 2, the tracking and function approximation performance improved by 0.47% and 15.75%, respectively. A-DNN 3 performed the best

TABLE VI
CASE 2 RESULTS: AVERAGE TRACKING ERROR, CONTROL EFFORT, AND FUNCTION APPROXIMATION ERROR

Config.	$\bar{e}$	$\bar{u}$	$\bar{\bar{f}}$	$\Delta\bar{e}^{\dagger}$	$\Delta\bar{u}$	$\Delta\bar{\bar{f}}$
DNN 2	0.1714	6.2782	2.1694	-	-	-
DNN 3	0.1808	6.1184	1.7637	5.48	-2.54	-18.70
A-DNN 2	0.17059	6.9319	1.8277	-0.47	10.41	-15.75
A-DNN 3	0.1674	6.3711	1.4863	-2.33	1.48	-31.49

† The relative percent change for the average tracking error, control input, and function approximation error are relative to DNN 2.

overall. In terms of tracking performance, A-DNN 3 improved by 2.33%, 7.81%, and 1.86% compared to DNN2, DNN3, and A-DNN 2, respectively. Moreover, in terms of function approximation performance, A-DNN 3 improved by 31.49%, 12.79%, and 15.74%.

Overall, the case studies conducted in this section demonstrate the performance improvements from DNN architectures and the developed accelerated gradient adaptation strategy. The study in Case 1 had a single hidden-layer NN scheme as the baseline and examined the effects of altering the NN architecture by adding more neurons in a hidden-layer and also adding more hidden layers. DNN architectures improved performance in comparison to shallow NNs. Leveraging the accelerated gradient strategy in tandem with DNN architectures showed the best overall performance. Similarly, the developed method improved performance in corresponding sets of experiments in Case 2 where different DNN architectures were compared with the baseline DNN adaptation against the developed method. Despite the improvement in the corresponding simulation sets, when cross-comparing A-DNN 2 to DNN 3, DNN 3 outperformed A-DNN 2 by 2.95% in function approximation. The primary difference between the simulations is the DNN configurations, where DNN 3 had more hidden layers. Hence, the DNN architecture and adaptation designs are independent attributing factors to performance. The developed method in this article focuses on adaptation for arbitrary fully-connected DNNs. Constructive methods for other DNN architectures in adaptive control may be an avenue for future research efforts.

VI. CONCLUSION

Recent connections in adaptive control to continuous-time analogs of accelerated gradient methods have led to the development of new real-time adaptation laws based on accelerated gradient methods. Motivated by the potential improvements in transient performance, an accelerated gradient approach was used to design an accelerated gradient-based DNN adaptive control scheme for trajectory tracking of uncertain nonlinear systems. A nonsmooth Lyapunov-based analysis was performed to show the developed methods yield global asymptotic tracking. Comprehensive simulation studies were conducted on a two-state nonlinear system, a two-link robotic manipulator, and a complex 20-D nonlinear system to demonstrate the improved performance from accelerated gradient-based DNN weight adaptation laws. The simulations of the two-state system showed the developed accelerated gradient-based adaptation outperformed the baseline gradient-based DNN adaptive scheme from [15] and had a 67.41% and 78.82% decrease in the rms tracking and function approximation errors, respectively. Similarly, the simulations on the robotic manipulator showed a 33.31% and 17.75% improvement. In the simulation with the 20-D system, comparative studies were conducted to investigate the effects of various NN architectures (varying number of hidden layers and neurons) and adaptation laws. In all cases, the DNN-based accelerated gradient method demonstrated improved tracking and function approximation performance.

Future work may involve analyzing new adaptation designs from the selection of the loss function in (14). As discussed in Remark 2, selecting a loss function that incorporates computable versions of the function approximation error in the adaptation design may improve performance. However, there are challenges in the analysis and implementation of such adaptation laws due to the unknown nature of the ideal DNN weights. Additionally, our findings in the case study in Section V-C indicate the selection of the DNN architecture is a contributing factor to performance. The developments in this article focus on adaptation laws given a class of fully connected DNNs with an arbitrary width and depth. Therefore, future research avenues may consider constructive methods (e.g., neural architecture search [54]) to generate an efficient DNN architecture for the adaptive controller. However, challenges will arise in developing stability-driven adaptation laws that are agnostic to varying DNN architectures (e.g., ResNets, convolutional NNs, and recurrent NNs) as the DNN architecture—along with the type of adaptation strategy—is embedded within the stability analysis.

APPENDIX

Consider a general uncertain Euler–Lagrange system modeled as [55, Ch. 2]

$$M(q)\ddot{q} + V_m(q)\dot{q} + F(\dot{q}) + G(q) = \tau \tag{46}$$

where $q, \dot{q}, \ddot{q} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the generalized position, velocity, and acceleration, respectively, $M : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ denotes a generalized inertia matrix, $V_m : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n\times n}$ denotes a generalized centripetal–Coriolis matrix, $G : \mathbb{R}^n \to \mathbb{R}^n$ represents a generalized vector of potential forces, $F : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ represents generalized dissipation effects, and $\tau : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the control input. The subsequent development is based on the assumption that q and $\dot{q}$ are measurable. The model in (46) satisfies the following properties and assumption.

Property 1: The inertia matrix $M(q)$ for all $q \in \mathbb{R}^n$ is positive-definite and satisfies $m_1\|\xi\|^2 \leq \xi^T M(q)\xi \leq m_2\|\xi\|^2$ for all $\xi \in \mathbb{R}^n$, where $m_1, m_2 \in \mathbb{R}_{>0}$ denote known constants.

Property 2: The inertia and centripetal-Coriolis matrices satisfy the following skew-symmetric relation $\xi^T(\dot{M}(q) - 2V_m(q,\dot{q}))\xi = 0$ for all $q, \dot{q}, \xi \in \mathbb{R}^n$.

Assumption 4 [56], [57]: The matrices $M(q)$, $V_m(q,\dot{q})$, $G(q)$, and $F(\dot{q})$ are bounded for all $q, \dot{q} \in \mathcal{L}_\infty$.

To quantify the tracking objective, the tracking error $e : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and auxiliary tracking error $r : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ are defined as

$$e \triangleq q_d - q \tag{47}$$

$$r \triangleq \dot{e} + \alpha e \tag{48}$$

respectively, where $\alpha \in \mathbb{R}_{\geq 0}$ denotes a user-defined parameter.

To facilitate the subsequent analysis, taking the time derivative of r, pre-multiplying by $M(q)$, using (46) and (48), and adding and subtracting the terms $M(q_d)\ddot{q}_d$, $V_m(q_d, \dot{q}_d)\dot{q}_d$, and $F(\dot{q}_d)$ yields the open-loop error system

$$M(q)\dot{r} = g(\zeta_d) - \tau - V_m(q, \dot{q})r + N(q, \dot{q}, q_d, \dot{q}_d, \ddot{q}_d) \tag{49}$$

where $\zeta_d \triangleq [q_d^T, \dot{q}_d^T, \ddot{q}_d^T]^T \in \mathbb{R}^{3n}$ denotes a concatenated state vector, the function $g : \mathbb{R}^{3n} \to \mathbb{R}^n$ denotes the unknown system dynamics defined as $g(\zeta_d) \triangleq M(q_d)\ddot{q}_d + V_m(q_d, \dot{q}_d)\dot{q}_d + F(\dot{q}_d)$, and the auxiliary function $N : \mathbb{R}^{5n} \to \mathbb{R}^n$ is defined as $N(q, \dot{q}, q_d, \dot{q}_d, \ddot{q}_d) \triangleq \alpha M(q)\dot{e} + (M(q) - M(q_d))\ddot{q}_d + (V_m(q, \dot{q}) - V_m(q_d, \dot{q}_d))\dot{q}_d + F(\dot{q} - \dot{q}_d) + V_m(q, \dot{q})\alpha e$. By the universal function approximation property, the DNN architecture in (4) is used to approximate the system uncertainties in (49) as

$$g(\zeta_d) = \Phi\left(\zeta_d, \theta^*\right) + \varepsilon(\zeta_d) \quad \forall \zeta_d \in \Omega \tag{50}$$

where Φ, θ^*, ε, and Ω were previously defined in Section II-C. Based on the subsequent stability analysis and following the development in Section III, the control input and adaptation laws are designed as

$$\tau \triangleq \beta_r r + e + \hat{\Phi} + \beta_s \operatorname{sgn}(r) + \rho(\|y\|)\|y\|\operatorname{sgn}(r) - 2\Phi'\left(\hat{\theta} - \hat{\nu}\right) \tag{51}$$

$$\dot{\hat{\nu}} \triangleq \operatorname{proj}\left(\gamma_\nu \hat{\Phi}'^T r\right) \tag{52}$$

$$\dot{\hat{\theta}} \triangleq -\operatorname{proj}\left(\gamma_\nu \gamma_\theta \left(\hat{\theta} - \hat{\nu}\right)\right) \tag{53}$$

where $y : \mathbb{R}_{\geq 0} \to \mathbb{R}^{2n}$ denotes a concatenated state vector defined as $y \triangleq [e^T, r^T]^T$. Substituting (50) and (51) into (49) and using (11) yields the closed-loop error system

$$M(q)\dot{r} = \hat{\Phi}'\tilde{\theta} - \beta_r r - e - \beta_s \operatorname{sgn}(r) - \rho(\|y\|)\|y\|\operatorname{sgn}(r) + 2\Phi'\left(\hat{\theta} - \hat{\nu}\right) - V_m(q, \dot{q})r + \tilde{N} \tag{54}$$

where $\tilde{N} \triangleq N + \varepsilon(\zeta_d) + \mathcal{O}^2(\|\tilde{\theta}\|)$. By Assumption 4, it can be shown that $\tilde{N}$ can be bounded as

$$\left\|\tilde{N}\right\| \leq a_0 + \rho(\|y\|)\|y\| \tag{55}$$

where $a_0 \in \mathbb{R}_{>0}$ denotes a known constant, and $\rho : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denotes a known positive definite and nondecreasing function. The following theorem analyzes the closed-loop error system in (54).[10]

Theorem 2: Consider a system modeled as in (46) that satisfies Properties 1 and 2. Let Assumption 3 and 4 hold. Then, the control input in (51) and accelerated gradient-based DNN weight adaptation laws in (52) and (53) ensure global asymptotic tracking in the sense that $\lim_{t\to\infty}\|e(t)\| = 0$, $\lim_{t\to\infty}\|r(t)\| = 0$, and $\lim_{t\to\infty}\|\hat{\theta}(t) - \hat{\nu}(t)\| = 0$, provided the sufficient gain condition $\beta_s > a_0$ is satisfied, where a_0 is a known constant defined in (55).

Proof: Consider the candidate Lyapunov function $V_L : \mathbb{R}^{2n+2\sum_{i=0}^{k}(L_i+1)L_{i+1}} \to \mathbb{R}_{\geq 0}$ defined as

$$V_L(\xi) \triangleq \frac{1}{2}r^T M r + \frac{1}{2}e^T e + \frac{1}{2\gamma_\nu}\left(\hat{\theta} - \hat{\nu}\right)^T\left(\hat{\theta} - \hat{\nu}\right) + \frac{1}{2\gamma_\nu}\left(\theta^* - \hat{\nu}\right)^T\left(\theta^* - \hat{\nu}\right) \tag{56}$$

which satisfies the inequality $\underline{\alpha}(\|\xi\|) \leq V_L(\xi) \leq \overline{\alpha}(\|\xi\|)$, where $\underline{\alpha}, \overline{\alpha} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ are continuous positive definite functions, and $\xi : \mathbb{R}_{\geq 0} \to \mathbb{R}^{2n+2\sum_{i=0}^{k}(L_i+1)L_{i+1}}$ denotes a concatenated state vector defined as $\xi \triangleq [r^T, e^T, (\hat{\theta} - \hat{\nu})^T, (\theta^* - \hat{\nu})^T]^T \in \mathbb{R}^{2n+2\sum_{i=0}^{k}(L_i+1)L_{i+1}}$. Taking the generalized time derivative of (56),[11] performing some algebraic manipulation, using (7), (48), (52)–(55) and Property 2 yields

$$\dot{V}_L \overset{\text{a.e.}}{\leq} -\beta_r\|r\|^2 - \alpha\|e\|^2 - \gamma_\theta\left\|\hat{\theta} - \hat{\nu}\right\|^2 \tag{57}$$

provided the sufficient gain condition $\beta_s > a_0$ is satisfied. Then, invoking the LaSalle–Yoshizawa theorem extension for nonsmooth systems in [50, Th. 2] yields $\lim_{t\to\infty}\|r(t)\| = 0$, $\lim_{t\to\infty}\|e(t)\| = 0$, and $\lim_{t\to\infty}\|\hat{\theta}(t) - \hat{\nu}(t)\| = 0$. Additionally, all closed-loop signals can be shown to be bounded by following the analysis in the proof of Theorem 1. ■

ACKNOWLEDGMENT

Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsoring agency.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 7553.

[2] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 604–624, Apr. 2020.

[3] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[4] X. Wang, Y. Gu, Y. Cheng, A. Liu, and C. L. P. Chen, "Approximate policy-based accelerated deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 1820–1830, Jun. 2020.

[5] F. Tang, B. Niu, G. Zong, X. Zhao, and N. Xu, "Periodic event-triggered adaptive tracking control design for nonlinear discrete-time systems via reinforcement learning," *Neural Netw.*, vol. 154, pp. 43–55, Oct. 2022.

[6] Y. Zhao, H. Wang, N. Xu, G. Zong, and X. Zhao, "Reinforcement learning-based decentralized fault tolerant control for constrained interconnected nonlinear systems," *Chaos, Solitons Fractals*, vol. 167, Feb. 2023, Art. no. 113034.

[7] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[8] N. E. Cotter, "The Stone-Weierstrass theorem and its application to neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 4, pp. 290–295, Dec. 1990.

[9] M. H. Stone, "The generalized Weierstrass approximation theorem," *Math. Mag.*, vol. 21, no. 4, pp. 167–184, 1948.

[10] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[11] 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.*, vol. 6, pp. 193–198, Jan. 2022.

[10] The switched system development in (24)–(28) trivially extends to the extension to Euler–Lagrange systems. For brevity of exposition, the switched system development is excluded in the extension.

[11] See below (31) for definition of the generalized time derivative.

[12] G. Joshi and G. Chowdhary, "Deep model reference adaptive control," in *Proc. IEEE Conf. Decis. Control*, May 2019, pp. 4601–4608.

[13] G. Joshi, J. Virdi, and G. Chowdhary, "Design and flight evaluation of deep model reference adaptive controller," in *Proc. AIAA Scitech. Forum*, Jan. 2020, p. 1336.

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

[15] O. Patil, D. Le, M. Greene, and W. E. Dixon, "Lyapunov-derived control and adaptive update laws for inner and outer layer weights of a deep neural network," *IEEE Control Syst. Lett.*, vol. 6, pp. 1855–1860, Dec. 2022.

[16] O. S. Patil, D. M. Le, E. J. Griffis, and W. E. Dixon, "Deep residual neural network (ResNet)-based adaptive control: A Lyapunov-based approach," in *Proc. IEEE 61st Conf. Decis. Control (CDC)*, Dec. 2022, pp. 3487–3492.

[17] M. Krstić, P. V. Kokotović, and I. Kanellakopoulos, "Transient-performance improvement with a new class of adaptive controllers," *Syst. Control Lett.*, vol. 21, no. 6, pp. 451–461, Dec. 1993.

[18] T. E. Gibson, A. M. Annaswamy, and E. Lavretsky, "On adaptive control with closed-loop reference models: Transients, oscillations, and peaking," *IEEE Access*, vol. 1, pp. 703–717, 2013.

[19] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $O\left(\frac{1}{k}^2\right)$," *Sov. Math. Doklady*, vol. 269, pp. 543–547, Jun. 1983.

[20] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, Jan. 1964.

[21] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. Cham, Switzerland: Springer, 2003.

[22] B. Wang and Q. Ye, "Improving deep neural networks' training for image classification with nonlinear conjugate gradient-style adaptive momentum," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–13, Mar. 2023.

[23] H. Ye, C. He, and X. Chang, "Accelerated distributed approximate Newton method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, Nov. 2023.

[24] W. Tao, G.-W. Wu, and Q. Tao, "Momentum acceleration in the individual convergence of nonsmooth convex optimization with constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 1107–1118, Mar. 2022.

[25] H. Li, H. Cheng, Z. Wang, and G.-C. Wu, "Distributed Nesterov gradient and heavy-ball double accelerated asynchronous optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5723–5737, Dec. 2021.

[26] H. Ye, L. Luo, and Z. Zhang, "Accelerated proximal subsampled Newton method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4374–4388, Oct. 2021.

[27] H. Wang, Y. Luo, W. An, Q. Sun, J. Xu, and L. Zhang, "PID controller-based stochastic optimization acceleration for deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5079–5091, Dec. 2020.

[28] W. Tao, Z. Pan, G. Wu, and Q. Tao, "The strength of nesterovs extrapolation in the individual convergence of nonsmooth optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2557–2568, May 2020.

[29] H. Zhang, J. Qian, J. Gao, J. Yang, and C. Xu, "Scalable proximal Jacobian iteration method with global convergence analysis for nonconvex unconstrained composite optimizations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2825–2839, Sep. 2019.

[30] D. Chang, S. Sun, and C. Zhang, "An accelerated linearly convergent stochastic L-BFGS algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3338–3346, Nov. 2019.

[31] W. Su, S. Boyd, and E. J. Candes, "A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 5312–5354, 2016.

[32] A. Wibisono, A. C. Wilson, and M. I. Jordan, "A variational perspective on accelerated methods in optimization," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 47, pp. E7351–E7358, Nov. 2016.

[33] A. C. Wilson, B. Recht, and M. I. Jordan, "A Lyapunov analysis of accelerated methods in optimization," *J. Mach. Learn. Res.*, vol. 22, no. 113, pp. 1–34, 2021.

[34] H. Attouch and F. Alvarez, "The heavy ball with friction dynamical system for convex constrained minimization problems," in *Lecture Notes in Economics and Mathematical Systems*. Cham, Switzerland: Springer, 2000, pp. 25–35.

[35] J. E. Gaudio, A. M. Annaswamy, M. A. Bolender, E. Lavretsky, and T. E. Gibson, "A class of high order tuners for adaptive systems," *IEEE Control Syst. Lett.*, vol. 5, no. 2, pp. 391–396, Apr. 2021.

[36] D. E. Ochoa, J. I. Poveda, A. Subbaraman, G. S. Schmidt, and F. R. Pour-Safaei, "Accelerated concurrent learning algorithms via data-driven hybrid dynamics and nonsmooth odes," in *Proc. Conf. Learn. Dyn. Control*, 2021, pp. 866–878.

[37] N. M. Boffi and J. E. Slotine, "Implicit regularization and momentum algorithms in nonlinearly parameterized adaptive control and prediction," *Neural Comput.*, vol. 33, no. 3, pp. 590–673, Mar. 2021.

[38] D. M. Le, O. Sudhir Patil, P. M. Amy, and W. E. Dixon, "Integral concurrent learning-based accelerated gradient adaptive control of uncertain Euler–Lagrange systems," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2022, pp. 806–811.

[39] D. M. Le, O. S. Patil, C. F. Nino, and W. E. Dixon, "Accelerated gradient approach for neural network adaptive control of nonlinear systems," in *Proc. IEEE 61st Conf. Decis. Control (CDC)*, Dec. 2022, pp. 1–17.

[40] D. Bernstein, *Matrix Mathematics*. Princeton, NJ, USA: Princeton Univ. Press, 2005.

[41] M. Krstic, I. Kanellakopoulos, and P. V. Kokotovic, *Nonlinear and Adaptive Control Design*. Hoboken, NJ, USA: Wiley, 1995.

[42] P. Ioannou and J. Sun, *Robust Adaptive Control*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[43] J. Slotine and W. Li, *Applied Nonlinear Control*. Upper Saddle River, NJ, USA: Prentice-Hall, 1991.

[44] P. Kidger and T. Lyons, "Universal approximation with deep narrow networks," in *Proc. Conf. Learn. Theory*, 2020, pp. 2306–2327.

[45] F. L. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *IEEE Trans. Neural Netw.*, vol. 7, no. 2, pp. 388–399, Mar. 1996.

[46] P. M. Patre, W. MacKunis, K. Kaiser, and W. E. Dixon, "Asymptotic tracking for uncertain dynamic systems via a multilayer neural network feedforward and RISE feedback control structure," *IEEE Trans. Autom. Control*, vol. 53, no. 9, pp. 2180–2185, Oct. 2008.

[47] P. M. Patre, S. Bhasin, Z. D. Wilcox, and W. E. Dixon, "Composite adaptation for neural network-based controllers," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 944–950, Apr. 2010.

[48] O. S. Patil, A. Isaly, B. Xian, and W. E. Dixon, "Exponential stability with RISE controllers," *IEEE Control Syst. Lett.*, vol. 6, pp. 1592–1597, Nov. 2022.

[49] R. Kamalapurkar, J. A. Rosenfeld, J. Klotz, R. J. Downey, and W. E. Dixon, "Supporting lemmas for RISE-based control methods," 2013, *arXiv:1306.3432*.

[50] R. Kamalapurkar, J. A. Rosenfeld, A. Parikh, A. R. Teel, and W. E. Dixon, "Invariance-like results for nonautonomous switched systems," *IEEE Trans. Autom. Control*, vol. 64, no. 2, pp. 614–627, Feb. 2019.

[51] B. E. Paden and S. S. Sastry, "A calculus for computing Filippov's differential inclusion with application to the variable structure control of robot manipulators," *IEEE Trans. Circuits Syst.*, vols. CS-34, no. 1, pp. 73–82, Jan. 1987.

[52] F. H. Clarke, *Optimization and Nonsmooth Analysis*. Philadelphia, PA, USA: SIAM, 1990.

[53] A. Parikh, R. Kamalapurkar, and W. E. Dixon, "Integral concurrent learning: Adaptive control with parameter convergence using finite excitation," *Int. J. Adapt Control Signal Process*, vol. 33, pp. 1775–1787, Dec. 2019.

[54] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 550–570, Feb. 2023.

[55] R. Ortega, A. Loría, P. J. Nicklasson, and H. J. Sira-Ramirez, *Passivity-based Control of Euler–Lagrange Systems: Mechanical, Electrical and Electromechanical Applications*. Cham, Switzerland: Springer, 1998.

[56] K. Dupree, P. Patre, Z. Wilcox, and W. E. Dixon, "Asymptotic optimal control of uncertain nonlinear systems," *Automatica*, vol. 47, no. 1, pp. 99–107, 2011.

[57] N. Sharma, S. Bhasin, Q. Wang, and W. E. Dixon, "Predictor-based control for an uncertain Euler–Lagrange system with input delay," *Automatica*, vol. 47, no. 11, pp. 2332–2342, 2011.

Duc M. Le received the Ph.D. degree in mechanical engineering from the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL, USA, in 2022.

During his graduate studies, his research was on nonlinear controls and autonomy, with a focus on adaptive control and switched system control in a variety of applications. He joined Aurora Flight Sciences, a Boeing Company, as an Aerospace Controls Researcher in 2023. His research interests include, but are not limited to, adaptive control, deep learning, and guidance navigation and control (GNC).

Omkar Sudhir Patil received the Bachelor of Technology (B.Tech.) degree in production and industrial engineering from the Indian Institute of Technology (IIT), Delhi, in 2018, the Master of Science (M.S.) degree in mechanical engineering in 2022, and the Ph.D. degree in mechanical engineering from the University of Florida, Gainesville, Florida, in 2023.

In 2019, he joined the Nonlinear Controls and Robotics (NCR) Laboratory at the University of Florida under the guidance of Dr. Warren Dixon to pursue his doctoral studies. In 2023, he started working as a Postdoctoral Research Associate at NCR Laboratory, University of Florida. His research focuses on the development and application of innovative Lyapunov-based nonlinear, robust, and adaptive control techniques.

Dr. Patil was honored with the BOSS award for his outstanding bachelor's thesis project at IIT. During his Ph.D. studies, he was awarded the Graduate Student Research Award for outstanding research.

Cristian F. Nino received the B.S. degrees in mechanical engineering and mathematics and M.S. degree in mechanical engineering from the University of Florida, Gainesville, Florida, in 2021 and 2024, respectively, where he is currently pursuing the Ph.D. degree in mechanical engineering, under the supervision of Dr. Warren Dixon.

His research interests include encompass multi-agent robotics, autonomy, guidance and navigation, distributed estimation, robust adaptive nonlinear control, reinforcement learning, and geometric mechanics and control.

Dr. Nino was awarded the Science, Mathematics, and Research for Transformation (SMART) Scholarship, which has included three internships at Eglin Air Force Base in 2023. As a member of the Nonlinear Controls and Robotics (NCR) laboratory at the University of Florida.

Warren E. Dixon (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Clemson University, Clemson, SC, USA, in 2000.

His main research interest has been the development and application of Lyapunov-based control techniques for uncertain nonlinear systems.

Dr. Dixon is an ASME and IEEE Fellow for contributions to adaptive control of uncertain nonlinear systems. His work has been acknowledged by various early and mid-career awards and best paper awards. After working at Oak Ridge National Laboratory as a Eugene P. Wigner Fellow and research staff member, he joined the University of Florida in 2004, and is currently the Dean's Leadership Professor and Department Chair in the Department of Mechanical and Aerospace Engineering. His work has been acknowledged by various early and mid-career awards and best paper awards.