Received: 13 August 2022 | Revised: 23 April 2023 | Accepted: 7 June 2023

DOI: 10.1002/oca.3028

RESEARCH ARTICLE

WILEY

Model-free optimal tracking over finite horizon using adaptive dynamic programming

Mayank Shekhar Jha | **Didier Theilliol** | **Philippe Weber**

Centre de Recherche en Automatique de Nancy (CRAN), UMR 7039, CNRS Faculté des Sciences et Technologies, Université de Lorraine, Vandoeuvre Cedex, France

Correspondence
Mayank Shekhar Jha, Centre de Recherche en Automatique de Nancy (CRAN), UMR 7039, CNRS Faculté des Sciences et Technologies, Université de Lorraine, B.P. 70239, 54506 Vandoeuvre Cedex, France.
Email: mayank-shekhar.jha@univ-lorraine.fr

Abstract

Adaptive dynamic programming (ADP) based approaches are effective for solving nonlinear Hamilton–Jacobi–Bellman (HJB) in an approximative sense. This paper develops a novel ADP-based approach, in that the focus is on minimizing the consecutive changes in control inputs over a finite horizon to solve the optimal tracking problem for completely unknown discrete time systems. To that end, the cost function considers within its arguments: tracking performance, energy consumption and as a novelty, consecutive changes in the control inputs. Through suitable system transformation, the optimal tracking problem is transformed to a regulation problem with respect to state tracking error. The latter leads to a novel performance index function over finite horizon and corresponding nonlinear HJB equation that is solved in an approximative iterative sense using a novel iterative ADP-based algorithm. A suitable Neural network-based structure is proposed to learn the initial admissible one step zero control law. The proposed iterative ADP is implemented using heuristic dynamic programming technique based on actor-critic Neural Network structure. Finally, simulation studies are presented to illustrate the effectiveness of the proposed algorithm.

KEYWORDS

actor critic, adaptive dynamic programming, model free, neural networks, nonlinear Hamilton Jacobi bellman, optimal tracking

1 | INTRODUCTION

Optimal control for nonlinear systems requires solving the nonlinear Hamilton–Jacobi–Bellman (HJB) equation analytically which is usually too difficult.[1,2] To that end, the usefulness of dynamic programming (DP) for solving HJB equation is well established.[3–5] Adaptive dynamic programming (ADP) based methods have emerged as effective approach for solving nonlinear HJB in approximative sense and solve the optimal control problems forward-in-time.[6–8] Several variants of adaptive-critic designs have been developed to implement ADP-based methods including heuristic dynamic programming (HDP), dual HDP and globalized-dual HDP.[9] While HDP employs a critic NN to approximate the optimal cost function, dual HDP uses critic NN to approximate the derivative of the optimal cost function, and globalized-dual HDP employs critic NN to estimate both of the optimal cost function as well as its derivative.[10] In a broad sense, the actor-critic structures are both included in most ADP-based methods.

For dynamical systems, it is usually required to the optimize the energy consumed (keeping control input to zero) along with desired performance, but it is also significantly important to minimize the consecutive changes in control inputs that is, Δu_t where u is the control input at time t. For time varying tracking problems, the control input values depend on

the desired trajectory. As such, the latter remain sensitive to the trajectory generator output and associated perturbations. Under constrain free conditions (non-saturating actuators) as considered in this paper, the control inputs are sensitive to severe changes in control values (depending on the reference trajectory). Thus, it becomes important to account for such changes in control inputs often brought in by trajectory generators, while designing optimal control for tracking. In fact, one of the undesirable effects of large control input change (frequent actuator solicitation) is actuator degradation.[11,12] ADP-based approaches are being extensively developed in various domains where smoothness in the control input profile is often desirable. This includes fault tolerant control,[13] smart buildings,[14] robust control,[15] time-delay systems,[16] health sector,[17] spacecraft rendezvous,[18] and so on.

Much of the existing work stands on the view of infinite horizon designs and seek to satisfy the required system properties including system stability over an infinite time horizon[19] provided an important proof of convergence for iterative ADP algorithm to solve discrete-time nonlinear HJB,[20] considered consecutive changes in control inputs within the cost function (infinite horizon) and provided the convergence proof,[21] extended the analysis to unknown affine systems,[22] extended the treatment for nonaffine nonlinear systems,[23] and[8] gave a holistic treatment and provided boundedness result for online HDP approach. Most of these works have considered the known system models.

Finite horizon control design seeks to satisfy various required properties within a finite horizon limit and remains closer to the reality and hence, more pertinent for physical systems.[24,25] Recent years have seen significant rise in finite horizon-based control.[26] In Reference 27, adaptive epsilon-ADP algorithm was proposed to solve near optimal control problem,[28] provided finite horizon iterative ADP-based optimal control,[29] considered a non-quadratic cost function and provided convergence proofs. However, the aforementioned works have considered the system models that are either fully known[20] or partially known (knowledge of control matrix).[30] presented an iterative ADP-based optimal control under fully unknown conditions but over infinite horizon. The recent works of References 31 and 32 present finite horizon optimal tracking design under unknown model conditions using iterative HDP conditions with convergence proofs. Although, complete unknown conditions are assumed, the knowledge of inverse system dynamics (imperative for feedforward control) is assumed to be available a priori and the inverse system dynamics problem is not addressed which is usually very difficult to obtain analytically.

In this context,[20] considered changes in control input within the cost function but over infinite time horizon and under known system conditions.

To the best knowledge of authors, none of the existing works have focused on minimizing the consecutive changes in control inputs over a finite horizon under complete unknown conditions within the ADP framework.

To bridge the existing scientific gap, this paper:

- Develops a novel ADP iterative algorithm by accounting for consecutive changes in control inputs within the cost function and provides novel proofs of convergence. It is noted that similar formulation was proposed in Reference 20 but over infinite horizon for optimal regulation problem under known conditions while this paper proposes the latter over finite horizon and optimal tracking problem under complete unknown conditions.
- To initialize the iterative algorithm under the assumptions of admissibility, an initial one step zero control law is required. This paper proposes a suitable NN-based algorithm to obtain the initial admissible one step zero control.

This section is followed by section 2 that presents problem formulation taking into account consecutive changes in control input within the cost function and deriving the corresponding nonlinear HJB equation. Then, in section 3, it is noted that unknown system dynamics as well as inverse dynamics need to be identified, section 4 presents the two novel contribution of the paper: novel algorithm to derive an initial one step zero control law, and the principal contribution in form of novel iterative ADP algorithm under finite horizon conditions with complete convergence analysis and proofs. Further, section 5, implements the proposed iterative ADP using actor-critic structure-based iterative HDP technique and finally, section 6 presents a simulation study and section 7 draws the conclusions.

2 | PROBLEM STATEMENT

The dynamic system is considered unknown, nonlinear, control affine in discrete time as:

$$\begin{aligned} x_{k+1} &= F\left(x_k, u_{p,k}\left(x_k\right)\right) \\ &= f\left(x_k\right) + g\left(x_k\right) \quad u_{p,k}\left(x_k\right). \end{aligned} \tag{1}$$

where $x_k \in \mathbb{R}^n$ is the global system state vector determining the system state trajectory, $u_p(x) \in \mathbb{R}^m$ is the control vector, $f(\cdot)$ and $g(\cdot)$ are nonlinear functions differentiable with respect to its arguments, $g(x)$ satisfies $\|g(x)\|_F \leq g_M \ \forall x \in \mathbb{R}^n$ where $g_M > 0$ is any positive constant. Moreover, $f + gu$ is considered Lipschitz continuous on a compact set Ω in $\mathbb{R}^n$ containing the origin, system (1) is stabilizable so that there exists a control sequence on compact set Ω that asymptotically stabilizes the system. For notational purposes, we denote the feedback control $u_{p,k}(x)$ as $u_{p,k}$. Optimal tracking problem consists of determining the optimal control law $u^*_{p,k}$ (or optimal control sequence) that makes the system (1) track a reference (desired) trajectory $r_k \in \Omega \subseteq \mathbb{R}^n$ generated by a trajectory generator $\phi(r) \in \mathbb{R}^n$ as:

$$r_{k+1} = \phi(r_k) \tag{2}$$

Assumption 1. Mapping between the state x_k and the desired trajectory $\phi(r_k)$ is one to one.

The steady state desired control $u_{d,k}$ corresponding to reference trajectory r_k can be defined as:

$$u_{d,k} = g^{-1}(r_k)\left[\phi(r_k) - f(r_k)\right] \tag{3}$$

where the inverse of control matrix $g^{-1}(r_k)$ is unknown but exits so that $g^{-1}(r_k)\, g(r_k) = I \in \mathbb{R}^{m\times m}$ with I being an identity matrix.

A reference control generator function $u_{d,k}$ depends on unknown inverse dynamics which must be identified (see section 3.2) to generate the desired reference control.

2.1 | System transformation and nonlinear HJB optimality

The tracking error is defined as:

$$e_k = x_k - r_k \tag{4}$$

The augmented error dynamics can be obtained from (1) to (3) as[33]:

$$\begin{bmatrix} e_{k+1} \\ r_{k+1} \end{bmatrix} = \begin{bmatrix} \left(f(e_k + r_k) + g(e_k + r_k) \quad u_{d,k} - \phi(r_k)\right) + g(e_k + r_k) \quad u_k \\ \phi(r_k) \end{bmatrix} \tag{5}$$

where $u_{d,k}$ is obtained from (3). Considering the augmented system (5), e_k and r_k can be treated as system variables and u_k is seen as feedback control input at k, given as:

$$u_k = u_{p,k} - u_{d,k} \tag{6}$$

Moreover, considering the fact the reference evolution does not depend on other system variables, (5) can be represented as:

$$e_{k+1} = F_e(e_k, u_k) \tag{7}$$

where $e_k = x_k - r_k$, $e_k \in \mathbb{R}^n$.

Now, starting from k, considering the initial state of the system at discrete time k as e_k, let the control sequence over finite horizon be noted as $\underline{u_k^{N-1}} = (u_k, u_{k+1}, \ldots u_{k+N-1})$ that leads to a system trajectory starting from e_k: $e_{k+1}, e_{k+2}, e_{k+3} \ldots e_{k+N}$. The cardinality of the control sequence that determines its length is denoted as $\left|\underline{u_k^{N-1}}\right|$. Then, $\left|\underline{u_k^{N-1}}\right| = N$ and the final state under the control sequence $\underline{u_k^{N-1}}$ is denoted as $e^{(final)}\left(e_k, \underline{u_k^{N-1}}\right)$. Then, $e^{(final)}\left(e_k, \underline{u_k^{N-1}}\right) = e_N$.

2.2 | Cost function penalizing consecutive changes in control input

For finite horizon optimal control problem, the optimality of the control sequence is assessed by a cost function that must penalize consecutive changes in control inputs, that is, Δu_t where u is the control input at time t.

To that end, the cost function in this paper seeks to penalize the consecutive changes in control inputs $u_k - u_{k-1}$. It is considered as:

$$J\left(e_k, \underline{u_k^{N-1}}\right) = \sum_{i=k}^{N-1} \eta\left(e_i, u_i, u_{i-1}\right) \tag{8}$$

where $\eta(\cdot)$ is the utility function and $\eta(0,0,0) = 0$, $\eta\left(e_i, u_i, u_{i-1}\right) \geq 0 \ \ \forall e_i, u_i, u_{i-1}$. The utility function is chosen as:

$$\eta\left(e_i, u_i, u_{i-1}\right) = \begin{bmatrix} \mathrm{e}^T{}_i & r_i^T \end{bmatrix} \begin{bmatrix} Q_x & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} e_i \\ r_i \end{bmatrix} + u_i^T S \ \ u_i + \left(u_i - u_{i-1}\right)^T R \left(u_i - u_{i-1}\right) \tag{9}$$

where Q_x, S, and R are symmetric and positive definite matrices of appropriate dimensions. The first term in the quadratic function penalizes system errors, the second term penalizes the control error and while the third term penalizes the consecutive changes in the control input.

Remark 1. From (6) it is observed that tracking control $u_{p,k}$ to the original system (1) consists of a feedback control u_k and a predetermined desired control $u_{d,k}$ associated with the desired reference trajectory r_k.

Remark 2. By considering the augmented error system (5) in the cost function (8), the optimal tracking problem of (1) has been transformed to an optimal regulation problem.

Defining $\mathrm{v}_k = u_k - u_{k-1}$ and noting the fact that $\mathrm{v}_0 = u_0$ and $\mathrm{v}_k = u_k = 0 \ \ \forall k < 0$, one can obtain:

$$\begin{aligned} u_k &= \mathrm{v}_k + \mathrm{v}_{k-1} + . \ \ldots \ \mathrm{v}_0 \\ &= \mathrm{v}_k + \sum_{l=0}^{k-1} \mathrm{v}_l \end{aligned} \tag{10}$$

where $\forall l < k$, control value at l is denoted as v_l. Moreover, for notational simplicity, $\sum_{l=0}^{k-1} \mathrm{v}_l$ is denoted as $\sum_{l=0}^{k-1} \mathrm{v}_l$. Then, the control action at any k, u_k becomes:

$$u_k = \mathrm{v}_k + \sum_{l=0}^{k-1} \mathrm{v}_l \tag{11}$$

Accordingly, the utility function in (9) becomes:

$$\eta\left(e_i, \mathrm{v}_i, \left(\mathrm{v}_i + \sum_{l=0}^{i-1} \mathrm{v}_l\right)\right) = \left\{ \begin{bmatrix} e_i^T & r_i^T \end{bmatrix} \overline{Q} \begin{bmatrix} e_i \\ r_i \end{bmatrix} + \mathrm{v}_i^T R \mathrm{v}_i + \left(\mathrm{v}_i + \sum_{l=0}^{i-1} \mathrm{v}_l\right)^T S \ \ \left(\mathrm{v}_i + \sum_{l=0}^{i-1} \mathrm{v}_l\right) \right\} \tag{12}$$

where $\overline{Q} = \begin{bmatrix} Q_x & 0 \\ 0 & 0 \end{bmatrix}$. After having defined control actions in (10), the control sequences are represented in accordance to (10). Then, consider $\underline{\mathrm{v}_k^{N-1}} = (\mathrm{v}_k, \mathrm{v}_{k+1}, .. \mathrm{v}_{N-1})$ as the sequence of control inputs from any $k > 0$ to $N-1$. Then, the sequence of control actions starting from k as defined in (10) can be denoted as: $\underline{\sum_{l=0} \mathrm{v}_l}_k^{N-1} = \left(\left(\mathrm{v}_k + \sum_{l=0}^{k-1} \mathrm{v}_l\right), \left(\mathrm{v}_{k+1} + \mathrm{v}_k + \sum_{l=0}^{k-1} \mathrm{v}_l\right), \ \ldots \ , \left(\mathrm{v}_{N-1} + \mathrm{v}_{N-2} + \ \ldots \ + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right)$. Additionally, the corresponding cumulative cost function (cf. (8)) can be defined as:

$$J\left(e_k, \underline{\mathrm{v}_k^{N-1}}, \underline{\sum_{l=0} \mathrm{v}_l}_k^{N-1}\right) = \sum_{i=k}^{N-1} \eta\left(e_i, \mathrm{v}_i, \left(\sum_{p=k}^{p=i} \mathrm{v}_p + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) \tag{13}$$

Remark 3. The difference of control errors in cost function (9) has been incorporated systematically in cost function (13). The problem is then transformed to that of finding control law $v^*(e_k)$ starting at k, that is optimal in sense of (13) and produces optimal control value v_k^* at k.

Remark 4. From v_k^*, the optimal control value u_k^* for the system (5) can be constructed as $u_k^* = v_k^* + \sum_{l=0}^{k-1} v_l$.

Definition 1. The control sequence starting from k, $\underline{v_k^{N-1}}$ is considered finite horizon admissible with respect to error state $e_k \in \Omega$ if $\underline{v_k^{N-1}}$ is continuous on compact set $\Omega_u \in \mathbb{R}^m$ for $\forall e_k \in \Omega$ and stabilizes (5) on Ω, $v_0 = 0$, and for every initial condition $e_k \in \Omega$, the cost $J\left(e_k, \underline{v_k^{N-1}}, \underline{\sum_{l=0} v_l^{N-1}}_k\right)$ is finite with the final state $e^{(final)}\left(e_k, \underline{v_k^{N-1}}\right) = 0$.

Assumption 2. In this work, the nonlinear augmented error system (5) is considered stabilizable on a compact set $\Omega \in \mathbb{R}^n$, that is, for all initial conditions $e_k \in \Omega$, there exists a control sequence $\underline{v_k^{N-1}}$ such that the final state $e^{(final)}\left(e_k, \underline{v_k^{N-1}}\right) = 0$.

2.3 | Associated nonlinear HJB optimality

Consider $\Theta_{e_k} = \left\{\underline{v_k} : e^{(final)}\left(e_k, \underline{v_k}\right) = 0\right\}$ be the set of all finite horizon admissible control sequences starting at k and let $\Theta_{e_k}^{(i)} = \left\{\underline{v_k^{k+i-1}} : e^{(final)}\left(e_k \underline{v_k^{k+i-1}}\right) = 0, \left|\underline{v_k^{k+i-1}}\right| = i\right\}$ be the set of all admissible finite horizon control sequences starting at k of length i. Then, the optimal cost function with respect to these admissible control sequences over the finite horizon $[k, N-1]$ is:

$$J^*(e_k) = \inf_{\underline{v_k^{N-1}}} \left\{ J\left(e_k, \underline{v_k^{N-1}}, \underline{\sum_{l=0} v_l^{N-1}}_k\right) : \underline{v_k^{N-1}} \in \Theta_{e_k}^{(N-k)} \right\} \tag{14}$$

It must be noted that cost function (13) can be written as combination of immediate one stage cost at k and cost due to rest of the remaining $N-k-1$ stages as:

$$J\left(e_k, \underline{v_k^{N-1}}, \underline{\sum_{l=0} v_l^{N-1}}_k\right) = \left(\begin{bmatrix} e_k^T & r_k^T \end{bmatrix} \overline{Q} \begin{bmatrix} e_k \\ r_k \end{bmatrix} + v_k^T R v_k + \left(v_k + \sum_{l=0}^{k-1} v_l\right)^T S \left(v_k + \sum_{l=0}^{k-1} v_l\right)\right) + \sum_{i=k+1}^{N-1} \eta\left(e_i, v_i, \left(\sum_{p=k}^{i} v_p + \sum_{l=0}^{k-1} v_l\right)\right) \tag{15}$$

According to Bellman's optimality principle,[3,4] the optimal cost function satisfies the Discrete time Hamilton Jacobi Bellman (DTHJB) optimal equation:

$$J^*(e_k) = \min_{v_k} \left\{ \begin{bmatrix} e_k^T & r_k^T \end{bmatrix} \overline{Q} \begin{bmatrix} e_k \\ r_k \end{bmatrix} + v_k^T R v_k + \left(v_k + \sum_{l=0}^{k-1} v_l\right)^T S \left(v_k + \sum_{l=0}^{k-1} v_l\right) + J^*(e_{k+1}) \right\} \tag{16}$$

This leads to the optimal control from e_k starting at k, $v^*(e_k)$ given by the gradient of the right hand side of (16) with respect to v_k as:

$$v^*(e_k) = -\frac{1}{2}(R+S)^{-1} \left[2S\left(\sum_{l=0}^{k-1} v_l\right) + g^T\left(e_{x,k} + r_k\right) \frac{\partial J^*(e_{k+1})}{\partial e_{k+1}} \right] \tag{17}$$

Substituting the value of optimal control in (16), the optimal cost can be derived as:

$$\begin{aligned} J^*(e_k) = \begin{bmatrix} e_k^T & r_k^T \end{bmatrix} \overline{Q} \begin{bmatrix} e_k \\ r_k \end{bmatrix} + \frac{1}{4}\left(2S\left(\sum_{l=0}^{k-1} \mathrm{v}_l\right) + g^T\left(e_{x,k} + r_k\right)\frac{\partial J^*(e_{k+1})}{\partial e_{k+1}}\right)^T \\ \times (R+S)^{-1} R (R+S)^{-1} \times \left(2S\left(\sum_{l=0}^{k-1} \mathrm{v}_l\right) + g^T\left(e_{x,k} + r_k\right)\frac{\partial J^*(e_{k+1})}{\partial e_{k+1}}\right) \\ + \left[-\frac{1}{2}(R+S)^{-1} \times 2S\left(\sum_{l=0}^{k-1} \mathrm{v}_l\right) + g^T\left(e_{x,k} + r_k\right)\frac{\partial J^*(e_{k+1})}{\partial e_{k+1}} + \sum_{l=0}^{k-1} \mathrm{v}_l\right]^T \\ \times S\left[-\frac{1}{2}(R+S)^{-1} \times 2S\left(\sum_{l=0}^{k-1} \mathrm{v}_l\right) + g^T\left(e_{x,k} + r_k\right)\frac{\partial J^*(e_{k+1})}{\partial e_{k+1}} + \sum_{l=0}^{k-1} \mathrm{v}_l\right] \\ + J^*(e_{k+1}) \end{aligned} \tag{18}$$

This equation cannot be solved exactly as no closed form solution exists.

Thus, the problem addressed in this paper is in two folds. First, the completely unknown dynamics problem of (1)–(3) is addressed by using NNs in section 3. Second, the optimal control problem of (18) is solved using a novel iterative ADP method and convergence analysis over the respective finite horizon is provided in section 4.

3 | SYSTEM IDENTIFIER-NN AND FEEDFORWARD CONTROLLER NN

For the completely unknown systems, the challenge lies in obtaining the complete or even the partial knowledge of the system dynamics. In this work, NNs are used to identify the complete system dynamics of (1) and a system identifier-NN (SI-NN) is constructed for that purpose.

Additionally, to obtain tracking control $u_{p,k}$, the knowledge of pre-determined desired reference control $u_{d,k}$ is needed (see Remark 1), which requires the knowledge of inverse mapping of the system as shown in (3). To that end, a feedforward controller neural network (FCN-NN) is proposed to identify the inverse mapping of the system that leverages the already trained SI-NN.

3.1 | System identifier-NN

Input–output data is used to identify the system dynamics using multi-layered feedforward NNs. According to the universal approximation property of NNs over compact set, system (1) has an NN-based representation on a compact set S. Denoting the weights between input layer and hidden layer as ϑ_m, ideal weights between the hidden layer and output layer as ω_m, a three-layer NN with n_1 number of neurons in each hidden layer is considered as:

$$x_{k+1} = \omega_m^{*T} \sigma_m\left(\vartheta_m^{*T} z_{m,k}\right) + \varepsilon_{m,k} \tag{19}$$

where $\omega_m^* \in \mathbb{R}^{n_1 \times n}$ and $\vartheta_m^* \in \mathbb{R}^{(n+m)\times n_1}$ are the constant ideal weight matrices of system (model) NN, $z_{m,k}^T = \begin{bmatrix} x_k^T & u_{p,k}^T \end{bmatrix}$ is the input to the NN, $\varepsilon_{m,k}$ is the NN function approximation error and $\sigma_m(\cdot)$ is the NN activation function chosen as the hyperbolic tangent function, that is, $\sigma_m(z) = (e^z - e^{-z}) / (e^z + e^{-z})$ so that $\sigma_m(z) \in [-1, 1]$ and $\sigma_m(z) \in \mathbb{R}^{n_1}$.

Assumption 3. (21): The activation function $\sigma(\cdot)$ and NN function approximation error $\varepsilon_{m,k}$ are considered upper bounded as $\|\sigma_m(\bullet)\| \leq \sigma_M$ and $\varepsilon_{m,k}^T \varepsilon_{m,k} \leq \lambda_M \tilde{x}^T \tilde{x}$ for some non-negative constants σ_M and λ_M.

Assumption 4. For the ease of analysis, only the output weights of NN are updated and the hidden weights are kept fixed.[21] Using Assumption 4 and considering $\overline{z}_{m,k} = \vartheta_m^{*T} z_{m,k}$, (19) can be rewritten as:

$$x_{k+1} = \omega_m^{*T} \sigma_m\left(\overline{z}_{m,k}\right) + \varepsilon_{m,k} \tag{20}$$

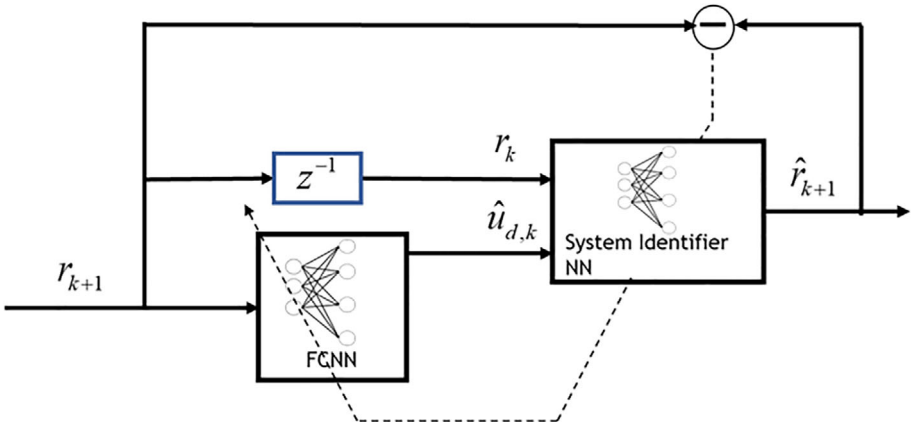

FIGURE 1 The feedforward neuro-controller NN training architecture.

The training procedure of SI-NN is very well established in the literature and major steps are provided in the Appendix A in Algorithm A1. After a sufficiently long training process, the system dynamics of (1) can be written as:

$$x_{k+1} = \hat{x}_{k+1} = \hat{\omega}_m^T \sigma_m \left(\bar{z}_{m,k}\right) \tag{21}$$

where $\hat{\omega}_m$, is the estimation of ideal weight matrix ω_m^*.

Next, the augmented system in (5) requires knowledge of the desired control (3).

3.2 | Feedforward controller neural network

To generate a desired control $u_{d,k}$ with respect to a given desired reference r_k (see (3)), the inverse mapping of system must be learnt. Under completely unknown conditions, this task is non-trivial. For this purpose, this work uses the feedforward neuro-control learning approach.[28] In this work, a FCN-NN is constructed to learn the inverse mapping using a feedforward NN. It is observed that the desired control can be obtained by setting in the original system (1), $u_{p,k} = u_{d,k}$, $x_k = r_k$ for all k that is,

$$\begin{aligned} r_{k+1} &= F\left(r_k, u_{d,k}\right) \\ u_{d,k} &= F^{-1}\left(r_{k+1}, r_k\right) \end{aligned} \tag{22}$$

where F^{-1} is the inverse dynamics of the system which is in general, very difficult to obtain, if not impossible. To that end, a fully connected feedforward NN structure is adopted for FCN-NN to learn the inverse dynamics F^{-1} by employing the structure shown in Figure 1.[30] The details of the architecture and major steps of NN training is provided in Appendix B. After a sufficiently long training, the inverse dynamics identification error approaches zero, the FCN-NN weights approach their ideal value in an asymptotic sense (the proof can be provided in sense of Theorem A3 in Appendix A and it is omitted here). Then, the inverse dynamics F^{-1} is learnt by the FCN-NN and the desired control $u_{d,k}$ in (22) can be expressed as:

$$u_{d,k} = \hat{u}_{d,k} \quad = \hat{\omega}_{F^{-1}}^T \sigma_{F^{-1}} \left(\bar{z}_{F^{-1},k}\right) \tag{23}$$

The desired control learnt here is needed in section 5 to implement the optimal tracking based on actor-critic structure. Next, section presents the major contributions of the paper.

4 | FINITE HORIZON TRACKING USING ITERATIVE ADP ALGORITHM

This section presents the principal contribution of this paper. A novel finite-horizon iterative ADP algorithm is proposed to solve (18) and the convergence to optimality is proved through convergence analysis.

However, the aforementioned algorithm requires presence of one-step zero control.[27,31]

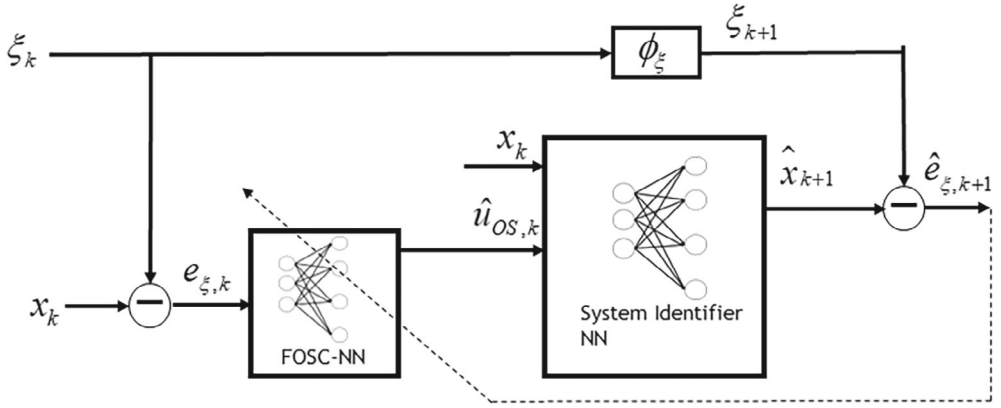

FIGURE 2 The feedforward one step-zero controller NN training architecture.

This section proposes a procedure, inspired from the one in Figure 1, to learn the one step zero control law using a NN. The novel architecture serves as a minor contribution of the paper, can be used obtain an initial one step control value u_{OS} for the error system (7) thus, avoiding the requirement of an analytical solution as in Reference 31.

4.1 | Initial one step-zero control

The objective is to learn an initial admissible control law. Using a one-step zero control law, the equivalent system (7) states are driven to zero (origin), that is, $F_e\left(e_k, u_{OS,k}\right) = 0$, where $u_{OS,k}$ is the control value that drives the system states to origin in one step.[31] used a similar strategy to assure presence of initial admissible control law for regulation purposes. However, a trained system network was used to get the initial one step control in form of a least square solution.

Here, a feedforward one step-zero controller NN (FOSC-NN) is introduced to learn the one step zero control law using a feedforward NN. The training of FOSC-NN is done using the structure shown in Figure 2 which comprises of trainable FOSC-NN, the already-trained SI-NN and system origin (where system states are zero) $\xi_k \in \mathbb{R}^n$.

Without the loss of generality, in this paper, $\xi_k = 0 \ \forall k$. The error $e_{\xi,k} \in \mathbb{R}^n$ is generated using original system states $x_k \in \mathbb{R}^n$ and the system-origin(s) $\xi_k \in \mathbb{R}^n$. Consider FOSC-NN as a three-layer NN with n_3 number of neurons in each hidden layer as:

$$\begin{aligned}\hat{u}_{OS,k} &= \omega_{OS}^{*T}\sigma_{OS}\left(\vartheta_{OS}^{*T}\ e_{\xi,k}\right) + \varepsilon_{OS,k} \\ &= \omega_{OS}^{*T}\sigma_{OS}\left(\bar{z}_{OS,k}\right) + \varepsilon_{OS,k}\end{aligned} \tag{24}$$

with $e_{\xi,k} \in \mathbb{R}^n$ as input and $\hat{u}_{OS,k}$ as the estimated one step control from FOSC-NN, $\omega_{OS}^* \in \mathbb{R}^{n_3 \times m}$ and $\vartheta_{OS}^* \in \mathbb{R}^{n \times n_3}$ are the constant ideal weight matrices of FOSC-NN, $\varepsilon_{OS,k}$ is the NN function approximation error and σ_{OS} is the NN activation function chosen as the hyperbolic tangent function, that is, $\sigma_{OS}(z) = \left(e^z - e^{-z}\right) / \left(e^z + e^{-z}\right)$ so that $\sigma_{OS}(z) \in [-1, 1]$ and $\sigma_{OS}(z) \in \mathbb{R}^{n_3}$. FOSC-NN is trained by feeding-in set of inputs $x_k \in \mathbb{R}^n$ and $\xi_k \in \mathbb{R}^n$, followed by adaptation of FOC-NN weights such that SI-NN (trained a priori) outputs zero error, that is, $\hat{e}_{k+1} = 0$. To that end, following steps are taken.

Algorithm 1. FOSC-NN training

Step 1: An array of the system origin values $\xi_k \in \mathbb{R}^n$ (in this paper, $\xi_k = 0, \forall k$), system states x_k are considered. Corresponding errors $e_{\xi,k} = x_k - \xi_k$ are generated.
Step 2: FOSC-NN is fed with input $e_{\xi,k}$, such that the output estimated the desired one step zero control value $u_{OS,k}$: $\hat{u}_{OS,k}$.
Step 3: The trained SI-NN with frozen weights is fed with inputs $z_{m,k}^T = \left[x_k^T\ \hat{u}_{OS,k}^T\right]$, so that the estimated output is $\hat{x}_{k+1}$. The output error is estimated as: $\hat{e}_{\xi,k+1} = \hat{x}_{k+1} - \xi_{k+1}$.
Step 4: FOSC-NN is trained using the error: $\hat{e}_{\xi,k+1}$ by minimizing the error function:

$$E_{OS,k+1} = (1/2)\hat{e}_{\xi,k+1}^T\hat{e}_{\xi,k+1}$$

so that $\widehat{e}_{\xi,k+1} = \widehat{x}_{k+1} - \xi_{k+1}$ is minimized in iterative sense. The weights of FCN-NN are adapted using a gradient descent-based optimization scheme:

$$\begin{aligned}\widehat{\omega}_{OS,k+1} &= \widehat{\omega}_{OS,k} - \alpha_{OS}\frac{\partial E_{OS,k+1}}{\partial \widehat{\omega}_{OS,k}} \\ &= \widehat{\omega}_{OS,k} - \alpha_{OS}\frac{\partial E_{OS,k+1}}{\partial \widehat{e}_{k+1}}\frac{\partial \widehat{e}_{\xi,k+1}}{\partial \widehat{u}_{OS,k}}\frac{\partial \widehat{u}_{OS,k}}{\widehat{\omega}_{OS,k}} \\ &= \widehat{\omega}_{OS,k} - \alpha_{OS}\frac{\partial \widehat{e}_{\xi,k+1}}{\partial \widehat{u}_{k}}\sigma_{OS}\left(\overline{z}_{OS,k}\right)\widehat{e}^{T}_{\xi,k+1}\end{aligned}$$

where $\alpha_{OS} > 0$ is the learning rate. It should be noted that the error $\widehat{e}_{\xi,k+1} = \widehat{x}_{k+1} - \xi_{k+1}$ is propagated through the trained SI-NN and $\frac{\partial \widehat{e}_{\xi,k+1}}{\partial \widehat{u}_{OS,k}} = \frac{\partial \widehat{x}_{k+1}}{\partial \widehat{u}_{OS,k}}$ is obtained through application of backpropagation algorithm from output of SI-NN $\widehat{x}_{k+1}$ to its input $\widehat{u}_{OS,k}$.

After a sufficiently long training procedure, the FOSC-NN identification error approaches zero and the network weights approach their ideal values in an asymptotic sense (the proof can be provided in sense of Theorem A3 in Appendix A and it is omitted here). Then, the one-step zero control law is learnt by the FOSC-NN that drives system states to zero (origin). (24) can be approximated as:

$$\widehat{u}_{OS,k} \;= \widehat{\omega}^{T}_{OS}\sigma_{OS}\left(\overline{z}_{OS,k}\right) \tag{25}$$

Thus, it is noted that given tracking error $e_k \in \mathbb{R}^n$, the trained FOSC-NN outputs control action $\widehat{u}_{OS,k}$ such that $F_e\left(e_k, \widehat{u}_{OS,k}\right) = 0$.

Remark 5. In theory, since $\vartheta^{T}_{OS,k}$ can be enough small, if $\widehat{\omega}_{OS,k}$ is appropriately set, then $F_e\left(e_k, \widehat{u}_{OS,k}\right) = 0$ can be assured. Therefore, in the theoretical analysis that follows, it will be considered that $F_e\left(e_k, \widehat{u}_{OS,k}\right) = 0$ is applicable. However, in practice, as remarked in Reference 31, the states are driven to the origin but remain deviated from the origin due to inertia of the control. In practice, a fine-tuning behavior can be applied to regulate the states to zero after the initial one step zero control.

4.2 | Derivation of iterative ADP algorithm

This section presents the principal contribution of this paper. For notational simplicity, at ith iteration $V^{(i)}$ shall denote the cost function value and $v^{(i)}$ denotes the minimum control input value.

First, consider the initial iterative index $i = 0$, such that initial cost function for all states $V^{(0)}(\cdot) = 0$. Then, from (27), an initial admissible control law of single (cardinality of one) control vector $u^{(0)}_{OS}(e_k)$ can be obtained from trained FOSC-NN as:

$$u^{(0)}_{OS}(e_k) = \widehat{\omega}^{T}_{OS}\sigma_{OS}\left(\widehat{\vartheta}^{T}_{OS}\;\; e_k\right) \tag{26}$$

which is subject to the final state condition constraint (see Assumption 1): $F_e\left(e_k, u^{(0)}_{OS}(e_k)\right) = 0$ or $e_{k+1} = 0$. Now, considering the transformation of control input in (11) that takes into account the cumulated control value till $k-1$: $\sum_{l=0}^{k-1}\mathrm{v}_l$, the transformed control law at k, $v^{(0)}(e_k)$ can be expressed as: $v^{(0)}(e_k) = u^{(0)}_{OS}(e_k) - \sum_{l=0}^{k-1}\mathrm{v}_l$. The cost function can be updated iteratively using $v^{(0)}(e_k), u^{(0)}_{OS}(e_k)$ and $V^{(0)}(\cdot)$:

$$\begin{aligned}V^{(1)}(e_k) &= \min_{\mathrm{v}_k}\left\{\boldsymbol{\eta}\left(e_k, \mathrm{v}_k, \left(\mathrm{v}_k + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right) + V^{(0)}(e_{k+1})\right\} \\ &= \min_{\mathrm{v}_k}\left\{\boldsymbol{\eta}\left(e_k, \mathrm{v}_k, \left(\mathrm{v}_k + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right) + 0\right\} \qquad \text{subject to}: F_e\left(e_k, u^{(0)}_{OS}(e_k)\right) = 0 \\ &= \boldsymbol{\eta}\left(e_k, v^{(0)}(e_k), \left(v^{(0)}(e_k) + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right)\end{aligned} \tag{27}$$

Second, consider the iterative index $i = 1$, the iterative control law $v^{(1)}(e_k)$ is calculated using the iterative cost function $V^{(1)}(e_k)$ as:

$$\begin{aligned} v^{(1)}(e_k) &= \underset{\mathrm{v}_k}{\operatorname{argmin}} \left\{ \boldsymbol{\eta}\left(e_k, \mathrm{v}_k, \left(\mathrm{v}_k + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) + V^{(1)}(e_{k+1}) \right\} \\ &= -\frac{1}{2}(R+S)^{-1}\left[2S\left(\sum_{l=0}^{k-1} \mathrm{v}_l\right) + g^T(e_k + r_k)\frac{\partial V^{(1)}(e_{k+1})}{\partial e_{k+1}}\right] \end{aligned} \tag{28}$$

followed by update of iterative cost function $V^{(2)}(e_k)$ using $v^{(1)}(e_k)$ as

$$\begin{aligned} V^{(2)}(e_k) &= \min_{\mathrm{v}_k} \left\{ \boldsymbol{\eta}\left(e_k, \mathrm{v}_k, \left(\mathrm{v}_k + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) + V^{(1)}(e_{k+1}) \right\} \\ &= \boldsymbol{\eta}\left(e_k, v^{(1)}(e_k), \left(v^{(1)}(e_k) + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) + V^{(1)}\left(F_e\left(e_k, \left(v^{(1)}(e_k) + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right)\right) \end{aligned} \tag{29}$$

In general, the closed form solutions are very difficult to obtain. To that end, for $i = 2,3,\ldots$, the above iterative procedure can be implemented as shown in (32):

$$\begin{aligned} v^{(i)}(e_k) &= \underset{\mathrm{v}_k}{\operatorname{argmin}} \left\{ \boldsymbol{\eta}\left(e_k, \mathrm{v}_k, \left(\mathrm{v}_k + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) + V^{(i)}(e_{k+1}) \right\} \\ &= -\frac{1}{2}(R+S)^{-1}\left[2S\left(\sum_{l=0}^{k-1} \mathrm{v}_l\right) + g^T(e_k + r_k)\frac{\partial V^{(i)}(e_{k+1})}{\partial e_{k+1}}\right] \end{aligned} \tag{30}$$

followed by cost function as shown in (33).

$$\begin{aligned} V^{(i+1)}(e_k) &= \min_{\mathrm{v}_k} \left\{ \boldsymbol{\eta}\left(e_k, \mathrm{v}_k, \left(\mathrm{v}_k + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) + V^{(i)}(e_{k+1}) \right\} \\ &= \boldsymbol{\eta}\left(e_k, v^{(i)}(e_k), \left(v^{(i)}(e_k) + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) + V^{(i)}\left(F_e\left(e_k, \left(v^{(i)}(e_k) + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right)\right) \end{aligned} \tag{31}$$

Remark 6. In the iterative ADP algorithm (32) and (33), superscript (i) denotes iteration index of control law and cost function, subscript k denotes the time index of augmented system's state and control trajectory.

4.3 | Convergence analysis of iterative ADP algorithm

In this section, convergence proofs for iterations between (32) and (33) is provided such that cost function $V^{(i)} \to J^*$ and the control law $v^{(i)} \to \mathrm{v}^*$ as $i \to \infty$.

Lemma 1. *Let $\left\{V^{(i)}(e_k)\right\}$ be the sequence of cost functions defined by (33). If there exists an initial one step zero control $u^{(0)}(e_k)$ such that $F_e\left(e_k, u^{(0)}(e_k)\right) = 0$, then there exits an upper bound Y such that $0 \le V^{(i)}(\mathrm{e}_k) \le Y$.*

Proof. Given time index k, from (33) we can get cost function with respect to state at k as:

$$V^{(i+1)}(e_k) = \min_{\mathrm{v}_k} \left\{ \boldsymbol{\eta}\left(e_k, \mathrm{v}_k, \left(\mathrm{v}_k + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) + V^{(i)}(e_{k+1}) \right\} \tag{32}$$

with $\mathrm{v}_{k-1}, \mathrm{v}_{k-2}, \ldots \mathrm{v}_0$ being any control actions before k. Then, it can be written over a time horizon of length i as:

$$V^{(i+1)}(e_k) = \min_{\underline{v_k^{k+i-1}}} \left\{ \boldsymbol{\eta}\left(e_k, v_k, \left(v_k + \sum_{l=0}^{k-1} v_l\right)\right) + \boldsymbol{\eta}\left(e_{k+1}, v_{k+1}, \left(v_{k+1} + v_k + \sum_{l=0}^{k-1} v_l\right)\right) + V^{(i-1)}(e_{k+2}) \right\}$$

$$\vdots$$

$$= \min_{\underline{v_k^{k+i-1}}} \left\{ \boldsymbol{\eta}\left(e_k, v_k, \left(v_k + \sum_{l=0}^{k-1} v_l\right)\right) + \boldsymbol{\eta}\left(e_{k+1}, v_{k+1}, \left(v_{k+1} + v_k + \sum_{l=0}^{k-1} v_l\right)\right) + \right.$$

$$\left. \ldots + \boldsymbol{\eta}\left(e_{k+i-1}, v_{k+i-1}, \left(v_{k+i-1} + \ldots + v_k + \sum_{l=0}^{k-1} v_l\right)\right) + V^{(1)}(e_{k+i}) \right. \tag{33}$$

where $V^{(1)}(e_{k+i}) = \min_{v_{k+i}} \boldsymbol{\eta}\left(e_{k+i}, v_{k+i}, \left(v_{k+i} + \ldots + v_k + \sum_{l=0}^{k-1} v_l\right)\right)$ is subject to $F_e\left(e_{k+i}, \left(v_{k+i} + \ldots + v_k + \sum_{l=0}^{k-1} v_l\right)\right) = 0$.

Then, we have

$$V^{(i+1)}(e_k) = \min_{\underline{v_k^{k+i}}} \sum_{j=0}^{i} \boldsymbol{\eta}\left(e_{k+j}, v_{k+j}, \left(\sum_{p=0}^{j} v_{k+p} + \sum_{l=0}^{k-1} v_l\right)\right) \tag{34}$$

Moreover, applying the definition of finite horizon cost function (c.f. (13)), we have:

$$V^{(i+1)}(e_k) = \min_{\underline{v_k^{k+i}}} J\left(e_k, \underline{v_k^{k+i}}, \left(\sum_{p=k}^{k+1} v_p + \sum_{l=0}^{k-1} v_l\right)\right)$$

$$\text{subject to } F_e\left(e_{k+i}, \left(v_{k+i} + \ldots + v_k + \sum_{l=0}^{k-1} v_l\right)\right) = 0 \tag{35}$$

Thus, $\forall i : 0 \leq V^{(i+1)}(e_k) \leq Y$ for some $Y \geq 0$. ∎

Moreover, taking into account the iteration between (32) and (33) at each stage starting at k until some $k+i$, this can be written in terms of minimum control value obtained at any iteration j, $v^{(i-j)}\left(e_{k+j}\right)$ as:

$$V^{(i+1)}(e_k) = \sum_{j=0}^{i} \boldsymbol{\eta}\left(e_{k+j}, v^{(i-j)}\left(e_{k+j}\right), \left(\sum_{p=0}^{j} v^{(i-p)} e_{(k+p)} + \sum_{l=0}^{k-1} v_l\right)\right) \tag{36}$$

Theorem 1. *Assuming there exists an initial one step zero finite-horizon admissible control law $u^{(0)}(e_k)$ such that $F_e\left(e_k, u^{(0)}(e_k)\right) = 0$, the cost function sequence $\left\{V^{(i)}\right\}$ obtained using iteration (32) and (33) is a monotonically non-increasing sequence such that $V^{(i+1)}(e_k) \leq V^{(i)}(e_k)$ for $\forall i \geq 1$.*

Proof. The proof is developed using mathematical induction. First, let $i = 1$, the cost function $V^{(1)}(e_k)$ is given by (29) subject to $F_e\left(e_k, \left(v^{(0)}(e_k) + \sum_{l=0}^{k-1} v_l\right)\right) = 0$. Let the corresponding sequence of finite horizon admissible control starting at k be denoted as: $\underline{\hat{v}_k^k}$. Then, $\underline{\hat{v}_k^k} = \left(v^{(0)}(e_k)\right)$ and the corresponding utility function is $\boldsymbol{\eta}\left(e_k, v^{(0)}(e_k), \left(v^{(0)}(e_k) + \sum_{l=0}^{k-1} v_l\right)\right)$. Now, it can be shown that there exists a finite horizon admissible control of length 2 so that corresponding cumulative cost $V^{(2)}(e_k)$ is related to $V^{(1)}(e_k)$. To that end, we construct finite horizon admissible control of length 2: $\underline{\hat{v}_k^{k+1}}$, so that $\underline{\hat{v}_k^{k+1}} = \left(v^{(0)}(e_k), -\left(v^{(0)}(e_k) + \sum_{l=0}^{k-1} v_l\right)\right)$. Then, using condition on final state we have $e_{k+1} = F_e\left(e_k, \left(v^{(0)}(e_k) + \sum_{l=0}^{k-1} v_l\right)\right) = 0$ and applying the 2nd control action in $\underline{\hat{v}_k^{k+1}}$, we get: the $e_{k+2} = F_e\left(e_{k+1}, \left(-\left(v^{(0)}(e_k) + \sum_{l=0}^{k-1} v_l\right) + v^{(0)}(e_k) + \sum_{l=0}^{k-1} v_l\right)\right) = (0,0) = 0$, which implies that $\underline{\hat{v}_k^{k+1}}$ is an admissible control. Then, the corresponding utility function $\boldsymbol{\eta}\left(e_{k+1}, -\left(v^{(0)}(e_k) + \sum_{l=0}^{k-1} v_l\right), 0\right) = \left(v^{(0)}(e_k) + \sum_{l=0}^{k-1} v_l\right)^T R\left(v^{(0)}(e_k) + \sum_{l=0}^{k-1} v_l\right)$. Now, we can obtain the cumulative cost function as:

$$J\left(e_k, \underline{\hat{\mathrm{v}}_k^{k+1}}, \left(\sum_{p=k}^{k+1}\mathrm{v}_p + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right) = \boldsymbol{\eta}\left(e_k, v^{(0)}(e_k), \left(v^{(0)}(e_k) + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right) + \boldsymbol{\eta}\left(e_{k+1}, -\left(v^{(0)}(e_k) + \sum_{l=0}^{k-1}\mathrm{v}_l\right), 0\right)$$

$$= V^{(1)}(e_k) + \left(v^{(0)}(e_k) + \sum_{l=0}^{k-1}\mathrm{v}_l\right)^T R\left(v^{(0)}(e_k) + \sum_{l=0}^{k-1}\mathrm{v}_l\right)$$

■

On the other hand, according to (37), we have:

$$V^{(2)}(e_k) = \min_{\underline{\hat{\mathrm{v}}_k^{k+1}}} J\left(e_k, \underline{\hat{\mathrm{v}}_k^{k+1}}, \left(\sum_{p=k}^{k+1}\mathrm{v}_p + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right)$$

$$= \min_{\underline{\hat{\mathrm{v}}_k^{k+1}}} \left(V^{(1)}(e_k) + \left(v^{(0)}(e_k) + \sum_{l=0}^{k-1}\mathrm{v}_l\right)^T R\left(v^{(0)}(e_k) + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right) \tag{37}$$

As R is an arbitrarily chosen non-negative matrix, it can be taken as small as desired, which leads us to conclusion that $V^{(2)}(e_k) \leq \min_{\underline{\hat{\mathrm{v}}_k^{k+1}}} J\left(e_k, \underline{\mathrm{v}_k^{k+1}}, \left(\sum_{p=k}^{k+i}\mathrm{v}_p + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right) = V^{(1)}(e_k)$. Therefore, theorem is valid for $i=1$. Now, let us assume that the theorem holds for some $i=q$, where $q \geq 1$. Then, using (38) we get:

$$V^{(q)}(e_k) = \sum_{j=0}^{q-1}\boldsymbol{\eta}\left(e_{k+j}, v^{(q-1-j)}\left(e_{k+j}\right), \left(\sum_{p=0}^{j} v^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right) \tag{38}$$

with the corresponding control sequence $\underline{\hat{\mathrm{v}}_q^{k+q-1}} = \left(v^{(q-1)}(e_k), v^{(q-2)}(e_{k+1}), \ldots v^{(0)}\left(e_{k+q-1}\right)\right)$. Then, for $i=q+1$, a control sequence of length $q+1$ can be constructed as $\underline{\hat{\mathrm{v}}_q^{k+q}} = \left(v^{(q-1)}(e_k), v^{(q-2)}(e_{k+1}), \ldots v^{(0)}\left(e_{k+q-1}\right), -\left(\sum_{p=0}^{q-1} v^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right)$ such that the corresponding system trajectory is given as: $e_{k+1} = F_e\left(e_k, \left(v^{(q-1)}(e_k) + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right); e_{k+2} = F_e\left(e_{k+1}, \left(v^{(q-1)}(e_k) + v^{(q-2)}(e_{k+1}) + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right), \ldots, e_{k+q} = F_e\left(e_{k+q-1}, \left(v^{(q-1)}(e_k) + \ldots + v^{(0)}\left(e_{k+q-1}\right) + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right)$. Now, as this is an admissible control, it leads to $e_{k+q} = 0$.

Then, last control from sequence $\underline{\hat{\mathrm{v}}_q^{k+q}}$ leads to:

$$e_{k+q+1} = F_e\left(e_{k+q}, \left(-\left(\sum_{p=0}^{q-1} v^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1}\mathrm{v}_l\right) + \sum_{p=0}^{q-1} v^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right) = F_e(0,0) = 0.$$

This shows $\underline{\hat{\mathrm{v}}_q^{k+q}}$ is a finite horizon admissible control. The corresponding utility function is:

$$\boldsymbol{\eta}\left(e_{k+q}, -\left(\sum_{p=0}^{q-1} v^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1}\mathrm{v}_l\right), 0\right) = \left(\sum_{p=0}^{q-1} v^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1}\mathrm{v}_l\right)^T R\left(\sum_{p=0}^{q-1} v^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1}\mathrm{v}_l\right) \tag{39}$$

Now, we can obtain the cumulative cost function as:

$$J\left(e_k, \underline{\hat{\mathrm{v}}_k^{k+q}}, \left(\sum_{p=k}^{k+q}\mathrm{v}_p + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right) = \boldsymbol{\eta}\left(e_k, v^{(q-1)}(e_k), \left(v^{(q-1)}(e_k) + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right)$$

$$+ \ldots + \boldsymbol{\eta}\left(e_{k+q-1}, v^{(0)}\left(e_{k+q-1}\right), \left(\sum_{p=0}^{q-1} v^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1}\mathrm{v}_l\right)\right) + \boldsymbol{\eta}\left(e_{k+q}, -\left(\sum_{p=0}^{q-1} v^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1}\mathrm{v}_l\right), 0\right).$$

From (40) and (41), it can be seen that:

$$J\left(e_k, \underline{\hat{\mathrm{v}}_k^{k+q}}, \left(\sum_{p=k}^{k+q} \mathrm{v}_p + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) = V^{(q)}(e_k) + \left(\sum_{p=0}^{q-1} \nu^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1} \mathrm{v}_l\right)^T R \left(\sum_{p=0}^{q-1} \nu^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1} \mathrm{v}_l\right) \tag{40}$$

On the other hand, according to (37), we have:

$$\begin{aligned} V^{(q+1)}(e_k) &= \min_{\underline{\hat{\mathrm{v}}_k^{k+q}}} J\left(e_k, \underline{\hat{\mathrm{v}}_k^{k+q}}, \left(\sum_{p=k}^{k+q} \mathrm{v}_p + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) \\ &\leq J\left(e_k, \underline{\hat{\mathrm{v}}_k^{k+q}}, \left(\sum_{p=k}^{k+q} \mathrm{v}_p + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) \end{aligned} \tag{41}$$

This implies: $V^{(q+1)}(e_k) \leq V^{(q)}(e_k) + \left(\sum_{p=0}^{q-1} \nu^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1} \mathrm{v}_l\right)^T R \left(\sum_{p=0}^{q-1} \nu^{(q-1-p)} e_{(k+p)} + \sum_{l=0}^{k-1} \mathrm{v}_l\right)$. Now, as R is an arbitrarily chosen non-negative matrix. Without loss of generality, it can be taken as small as desired. This leads us to the conclusion that $V^{(q+1)}(e_k) \leq V^{(q)}(e_k)$ and completes the proof by mathematical induction. ■

Thus, $\left\{V^{(i)}\right\}$ is a monotonically nonincreasing sequence bounded below and as such the limit $\lim_{i\to\infty} V^{(i)}(e_k) = V^{(\infty)}(e_k)$ exists.

Theorem 2. *Assuming there exists an initial one step zero finite-horizon admissible control law $u_{OS}^{(0)}(e_k)$ such that $F_e\left(e_k, u_{OS}^{(0)}(e_k)\right) = 0$, given the sequence of cost function $\left\{V^{(i)}\right\}$ as in (33) with initial value $V^{(0)}(\cdot) = 0$, if e_k is controllable, then as $i \to \infty$, the sequence converges toward the optimal cost function J^*, that is, $\lim_{i\to\infty} V^{(i)}(e_k) = V^{(\infty)}(e_k) = J^*(e_k)$.*

Proof. From (14) and (37) or Lemma 1, we get

$$\begin{aligned} J^*(e_k) &= \inf_{\underline{\mathrm{v}_k^{N-1}}} \left\{ J\left(e_k, \underline{\mathrm{v}_k^{N-1}}, \underbrace{\sum_{l=0}^{N-1} \mathrm{v}_l}_{k}\right) : \underline{\mathrm{v}_k^{N-1}} \in \Theta_{e_k}^{(N-k)} \right\} \\ &\leq \min_{\underline{\mathrm{v}_k^{k+i}}} \left\{ J\left(e_k, \underline{\mathrm{v}_k^{k+i}}, \left(\sum_{p=k}^{k+1} \mathrm{v}_p + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) : \underline{\mathrm{v}_k^{k+i-1}} \in \Theta_{e_k}^{(i)} \right\} \\ &= V^{(i)}(e_k) \end{aligned} \tag{42}$$

From (44), we have $J^*(e_k) \leq V^{(i)}(e_k)$. Letting $i \to \infty$, we get:

$$J^*(e_k) \leq V^{(\infty)}(e_k) \tag{43}$$

Now, using definition of $J^*(e_k)$ for any arbitrary $\delta > 0$, there exists an admissible control sequence (not necessarily a minimizing one) $\underline{\tau_k} \in \Theta_{e_k}$ such that

$$J\left(e_k, \underline{\tau_k}, \left(\sum_{p=k} \tau_p + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) \leq J^*(e_k) + \delta \tag{44}$$

Now, if $\underline{\tau_k} \in \Theta_{e_k}^{(q)}$ such that control sequence $\underline{\tau_k}$ is of length q, that is, $\left|\underline{\tau_k}\right| = q$. Then, on one hand, from Theorem 1, we obtain:

$$\begin{aligned} V^{(\infty)}(e_k) &\leq V^{(q)}(e_k) \\ &= \min_{\underline{\mathrm{v}_k^{k+q-1}}} \left\{ J\left(e_k, \underline{\mathrm{v}_k^{k+q-1}}, \left(\sum_{p=k}^{k+q-1} \mathrm{v}_p + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) : \underline{\mathrm{v}_k^{k+q-1}} \in \Theta_{e_k}^{(q)} \right\} \end{aligned} \tag{45}$$

On the other hand, as $\underline{\tau_k} \in \Theta_{e_k}$ is an arbitrary admissible control sequence, we get:

$$\min_{\underline{\mathrm{v}_k^{k+q-1}}} \left\{ J\left(e_k, \underline{\mathrm{v}_k^{k+q-1}}, \left(\sum_{p=k}^{k+q-1} \mathrm{v}_p + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) : \underline{\mathrm{v}_k^{k+q-1}} \in \Theta_{e_k}^{(q)} \right\} \leq J\left(e_k, \underline{\tau_k^{k+q-1}}, \left(\sum_{p=k}^{k+q-1} \tau_p + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) \tag{46}$$

Then, from (46), (47), and (48), we get

$$V^{(\infty)}(e_k) \leq V^{(q)}(e_k) \leq J\left(e_k, \underline{\tau_k}, \left(\sum_{p=k} \tau_p + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) \leq J^*(e_k) + \delta$$
$$or, \quad V^{(\infty)}(e_k) \leq J^*(e_k) + \delta \tag{47}$$

As $\delta > 0$ is chosen arbitrarily, we have:

$$V^{(\infty)}(e_k) \leq J^*(e_k) \tag{48}$$

Combining (45) and (50), we can conclude that sequence $\left\{V^{(i)}(e_k)\right\}$ tends to $J^*(e_k)$ as $i \to \infty$ that is, $\lim_{i\to\infty} V^{(i)}(e_k) = V^{(\infty)}(e_k) = J^*(e_k)$. Hence, the theorem is proved. ∎

Corollary 1. Assuming there exists an initial one step zero finite-horizon admissible control law $u^{(0)}(e_k)$ such that $F_e\left(e_k, u^{(0)}(e_k)\right) = 0$, given the sequence of cost function $\left\{V^{(i)}\right\}$ as in *(33)* with initial value $V^{(0)}(\cdot) = 0$ and corresponding control sequence $\left\{v^{(i)}\right\}$ as in *(32)*. Then, for $\forall k$, following holds true: $V^{(\infty)}(e_k) = \min_{\mathrm{v}_k} \left\{ \boldsymbol{\eta}\left(e_k, \mathrm{v}_k, \left(\mathrm{v}_k + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) + V^{(\infty)}(e_{k+1}) \right\}$.

The proof follows directly from Theorem 2 and Bellman's optimality principle (c.f. (16)).

Corollary 2. Let the sequence of cost function $\left\{V^{(i)}\right\}$ as in *(33)* with initial value $V^{(0)}(\cdot) = 0$ and the corresponding control sequence $\left\{v^{(i)}\right\}$ as in *(32)*. Then as $i \to \infty$, the sequence converges toward the optimal control law v^*, that is, as $i \to \infty$ $\lim_{i\to\infty} v^{(i)}(e_k) = \mathrm{v}^*(e_k)$.

According to Theorem 2 and Corollary 1 and 2, the iterative procedure needs to be run until $i \to \infty$ to obtain the optimal cost function $J^*(e_k)$ and optimal control law $\mathrm{v}^*(e_k)$. In practice, it is impossible to implement this strategy.

To that end, an ε − optimal approach can be adopted such that iterative strategy is followed till a pre-defined tolerable error ε is reached. The algorithm is stopped when iterative cost function satisfies the prescribed error, that is:

$$\left|V^{(i)}(e_k) - J^*(e_k)\right| \leq \varepsilon \tag{49}$$

The length of the control sequence depends on the initial states of the system and tolerable error. However, as $J^*(e_k)$ is not known a priori, it is difficult to implement (51) as the stopping criteria. Thus, an alternative stopping criterion can be formulated as:

$$\left|V^{(i)}(e_k) - V^{(i+1)}(e_k)\right| \leq \varepsilon \tag{50}$$

The equivalence of (50) to (49) can be shown in the sense of References 28,31 and it is omitted in this paper.

5 | ACTOR CRITIC STRUCTURE-BASED IMPLEMENTATION

The optimal(sub) control law is learnt using iterations (32) and (33), which can be executed using adaptive-critic structures[10] by approximating value function $V^{(i+1)}$ and control policy function $v^{(i)}$ using NNs.

In this paper, the HDP architecture is used the execute the iterative ADP algorithm using actor-critic NNs. To that end, four feedforward fully connected NNs are used to execute the iterative ADP algorithm.

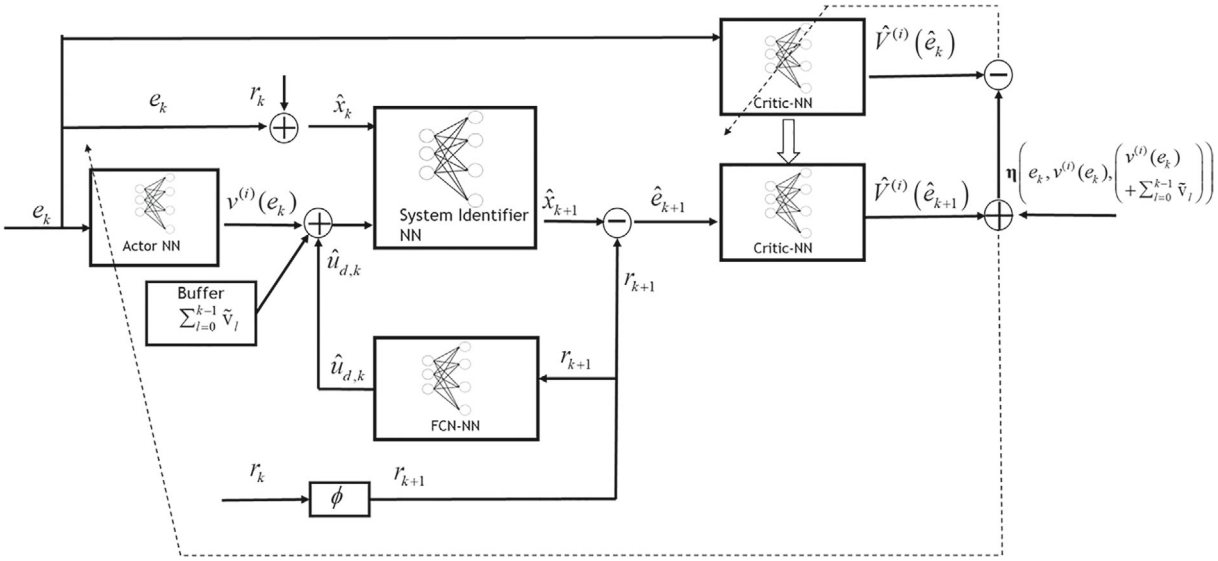

FIGURE 3 The actor critic-based implementation using iterative heuristic dynamic programming.

As shown in Figure 3, the four NNs are: SI-NN model (trained a priori, see (21)), FCN-NN (trained a priori, see (23)), critic network (critic-NN) and action network (action-NN). All NNs are chosen as fully connected feedforward NNs. It is noted that training of SI-NN and FCN-NN is done before implementing the iterative algorithm and weights of the SI-NN and FCN-NN are kept unchanged (frozen) during the implementation of proposed iterative ADP algorithm.

5.1 | System model

The trained SI-NN (with ideal weight sets) outputs the next state as ((21)): $\hat{x}_{k+1} = \omega_m^{*T}\sigma_m\left(\vartheta_m^{*T}z_{m,k}\right)$ with $z_{m,k}^T = \begin{bmatrix} x_k^T & u_{p,k}^T \end{bmatrix}$ as inputs.

5.2 | FCN-NN training

FCN-NN is trained using Algorithm B1 (see Appendix B). With trained FCN-NN, the desired control $u_{d,k}$ can be generated with respect to any given reference value r_k. The input control $u_{p,k}$ (see (6)) consists of two components: feedforward component $u_{d,k}$ and the feedback component $v^{(i)}(e_k) + \sum_{l=0}^{k-1}\mathrm{v}_l$ with respect to state e_k at an iteration i. While the former feedforward component is generated from trained FCN-NN network, the latter feedback component comprises of actor output: $v^{(i)}(e_k)$ and the accumulated control value till time $k-1$: $\sum_{l=0}^{k-1}\mathrm{v}_l$. The actor output at any iteration is learnt in iterative sense by the actor-NN structure using the HDP actor critic training scheme and accumulated control value $\sum_{l=0}^{k-1}\mathrm{v}_l$ is stored in a buffer till time k.

Then, with $\hat{x}_k = \hat{e}_k + r_k$ and $u_{p,k} = \left(v^{(i)}(e_k) + \sum_{l=0}^{k-1}\mathrm{v}_l\right) + u_{d,k}$, the state estimation at successive instant can be obtained from the trained SI-NN, as: $\hat{x}_{k+1} = \hat{e}_{k+1} + r_{k+1}$. While the feedforward component is generated from trained FCN-NN network at any time k, the feedback component is learnt in iterative sense using critic-NN and action-NN structure using the HDP architecture.

5.3 | Critic-NN

The critic-NN is used for approximating the cost function $V^{(i+1)}(e_k)$. The output of the critic network is denoted as:

$$\hat{V}^{(i+1)}(e_k) = \omega_C^{T(i+1)}\sigma_c\left(\vartheta_C^{T(i+1)}e_k\right) \tag{51}$$

However, from (33) we have:

$$V^{(i+1)}(e_k) = \boldsymbol{\eta}\left(e_k, v^{(i)}(e_k), \left(v^{(i)}(e_k) + \sum_{l=0}^{k-1} \mathrm{v}_l\right)\right) + \widehat{V}^{(i)}(e_{k+1}) \tag{52}$$

which can be constructed using the summation of immediate cost value $\boldsymbol{\eta}(\cdot)$ and estimated cost value from critic-NN. Moreover, $\sum_{l=0}^{k-1}\mathrm{v}_l$ is generated at k by summing the stored control values till $k-1$ in a buffer memory of appropriate size. Thus, the target error for critic-NN can be generated as:

$$e_{c,k}^{(i+1)} = \widehat{V}^{(i+1)}(e_k) - V^{(i+1)}(e_k) \tag{53}$$

with $e_{c,k}^{(i+1)}$ denoting the error at $(i+1)$th iteration at time index k. Then, the critic-NN minimizes the error function:

$$E_{c,k}^{(i+1)} = (1/2)e_{c,k}^{T(i+1)}e_{c,k}^{(i+1)} \tag{54}$$

and the weights are adapted according to gradient-based scheme as:

$$\begin{aligned}
\omega_C^{(i+1)}(j+1) &= \omega_C^{(i+1)}(j) - \alpha_c \frac{\partial E_{c,k}^{(i+1)}}{\partial \omega_C^{(i+1)}(j)} \\
&= \omega_C^{(i+1)}(j) - \alpha_c \frac{\partial E_{c,k}^{(i+1)}}{\partial e_{c,k}^{(i+1)}} \frac{\partial e_{c,k}^{(i+1)}}{\partial \widehat{V}^{(i+1)}(e_k)} \frac{\partial \widehat{V}^{(i+1)}(e_k)}{\partial \omega_C^{(i+1)}(j)} \\
\vartheta_C^{(i+1)}(j+1) &= \vartheta_C^{(i+1)}(j) - \alpha_c \frac{\partial E_{c,k}^{(i+1)}}{\partial \vartheta_C^{(i+1)}(j)} \\
&= \vartheta_C^{(i+1)}(j) - \alpha_c \frac{\partial E_{c,k}^{(i+1)}}{\partial e_{c,k}^{(i+1)}} \frac{\partial e_{c,k}^{(i+1)}}{\partial \widehat{V}^{(i+1)}(e_k)} \frac{\partial \widehat{V}^{(i+1)}(e_k)}{\partial \sigma_c\left(\vartheta_C^{T(i+1)} e_k\right)} \frac{\partial \sigma_c\left(\vartheta_C^{T(i+1)} e_k\right)}{\partial \vartheta_C^{(i+1)}(j)}
\end{aligned} \tag{55}$$

where $\alpha_c > 0$ is the learning rate and j is the weight updating/training inner loop iterative index.

Remark 7. It is noted that each iteration (outer loop) i, consists of a set of weight updating cycles indexed by j (inner loop) to learn the critic weights for the value function at that given ADP iteration step i, with respect to the error at time k.

5.4 | Actor-NN

Actor-NN is used to calculate the control value $v^{(i)}(e_k)$ that minimizes the cost function with respect to e_k at iterative step i in accordance to (32). To that end, consider actor-NN as:

$$\widehat{v}^{(i)}(e_k) = \omega_a^{T(i)}\sigma_a\left(\vartheta_a^{T(i)} e_k\right) \tag{56}$$

The target value is given by (32) as:

$$v^{(i)}(e_k) = -\frac{1}{2}(R+S)^{-1}\left[2S\left(\sum_{l=0}^{k-1}\mathrm{v}_l\right) + g^T(e_k + r_k)\frac{\partial \widehat{V}^{(i)}(e_{k+1})}{\partial(e_{k+1})}\right] \tag{57}$$

where $\partial\widehat{V}^{(i)}(e_{k+1})$ can be produced the trained critic-NN (57) and $\sum_{l=0}^{k-1}\mathrm{v}_l$ is generated from the buffer memory at k. Moreover (59) requires control matrix $\widehat{g}(e_k + r_k)$ which cannot be obtained directly for unknown systems. To that end, the trained SI-NN can be used (see (1) and (21)) as:

$$\widehat{g}\left(x_k\right)=\frac{\partial\left(\widehat{x}_{k+1}\right)}{\partial\left(u_{p,k}\right)}=\frac{\partial\left(\omega_m^{*T}\sigma_m\left(\vartheta_m^{*T}z_{m,k}\right)\right)}{\partial\left(u_{p,k}\right)}=\omega_m^{*T}\frac{\partial\left(\sigma_m\left(\vartheta_m^{*T}z_{m,k}\right)\right)}{\partial\left(\vartheta_m^{*T}z_{m,k}\right)}\vartheta_m^{*T}\frac{\partial\left(z_{m,k}\right)}{\partial\left(u_{p,k}\right)} \tag{58}$$

It is clear that $\widehat{g}\left(x_k\right)$ can be obtained using backpropagation from outputs of SI-NN $\widehat{x}_{k+1}$ and input $u_{p,k}$. Further, the target error for actor-NN can be generated (61), followed by minimization of error function (62) and actor-NN weight update (63).

$$e_{a,k}^{(i)}=\widehat{v}^{(i)}\left(e_k\right)-v^{(i)}\left(e_k\right) \tag{59}$$

$$E_{a,k}^{(i)}=(1/2)e_{a,k}^{T(i)}e_{a,k}^{(i)} \tag{60}$$

$$\begin{aligned}\omega_a^{(i)}(j+1)&=\omega_a^{(i)}(j)-\alpha_c\frac{\partial E_{a,k}^{(i)}}{\partial\omega_a^{(i)}(j)}\\&=\omega_a^{(i)}(j)-\alpha_c\frac{\partial E_{a,k}^{(i)}}{\partial e_{a,k}^{(i)}}\frac{\partial e_{a,k}^{(i)}}{\partial\widehat{v}^{(i)}\left(e_k\right)}\frac{\partial\widehat{v}^{(i)}\left(e_k\right)}{\partial\omega_a^{(i)}(j)}\\\vartheta_a^{(i)}(j+1)&=\vartheta_a^{(i)}(j)-\alpha_a\frac{\partial E_{c,k}^{(i+1)}}{\partial\vartheta_a^{(i)}(j)}\\&=\vartheta_a^{(i)}(j)-\alpha_a\frac{\partial E_{a,k}^{(i)}}{\partial e_{a,k}^{(i)}}\frac{\partial e_{a,k}^{(i)}}{\partial\widehat{v}^{(i)}\left(e_k\right)}\frac{\partial\widehat{v}^{(i)}\left(e_k\right)}{\partial\sigma_a\left(\vartheta_a^{T(i)}e_k\right)}\frac{\partial\sigma_a\left(\vartheta_a^{T(i+1)}e_k\right)}{\partial\vartheta_a^{(i)}(j)}\end{aligned} \tag{61}$$

Here, $e_{a,k}^{(i)}$ denotes the error at (i)th iteration at time index k, $\alpha_c>0$ is the learning rate and j is the weight updating/training inner loop iterative index.

6 | SIMULATION STUDY

In this section, two examples are presented to demonstrates the theoretical result and assess the performance of the HDP algorithm. Following nonlinear system is considered as $x_{k+1}=f\left(x_k\right)+g\left(x_k\right)\ \ u_{p,k}\left(x_k\right)$.

Example 1. The nonlinear system is considered from Reference 28 with some modifications:

$$\boldsymbol{x}_{k+1}=\begin{bmatrix}sin\left(0.5x_{2,k}\right)x_{1k}^2\\cos\left(1.4x_{2,k}\right)sin\left(0.9x_{1,k}\right)\end{bmatrix}+\begin{bmatrix}1.1&0\\0&0.45\end{bmatrix}u_{p,k} \tag{62}$$

such that $f\left(x_k\right)=\left[sin\left(0.5x_{2,k}\right)x_{1k}^2\ ,\ \ cos\left(1.4x_{2,k}\right)sin\left(0.9x_{1,k}\right)\right]^T$, $g\left(x_k\right)=[1.1,\ \ 0.45]^T$ with $\boldsymbol{x}_k=[x_{1k}\ \ x_{2k}]^T\in\mathbb{R}^2$ as state variables and $u_{p,k}\in\mathbb{R}^2$ as control variable. The reference trajectory for the above system is considered as:

$$r_k=\begin{bmatrix}0.3cos(0.2k)\\0.1sin(0.4k)\end{bmatrix} \tag{63}$$

The cost function is defined by (9) with $Q_x=0.1I$, $S=2I$, and $R=I$ where I denotes identity matrix of suitable dimensions.

It is assumed that system dynamics completely unknown and the input–output data is available. SI-NN is established by a four layered fully connected feedforward network with: 3 input neurons, 16-first hidden layer neurons, 8-second hidden layer neurons and 2 outputs neurons. All activation functions are hyperbolic tangent functions. The initial weights of the NN are randomly initialized in the interval $[-0.1, 0.1]$. Training set consists of 2000 randomly sampled input–output data. The learning rate is kept as $\alpha_m=0.001$ and the training is done for 1000 epochs. Figure 4 shows the estimation of 100

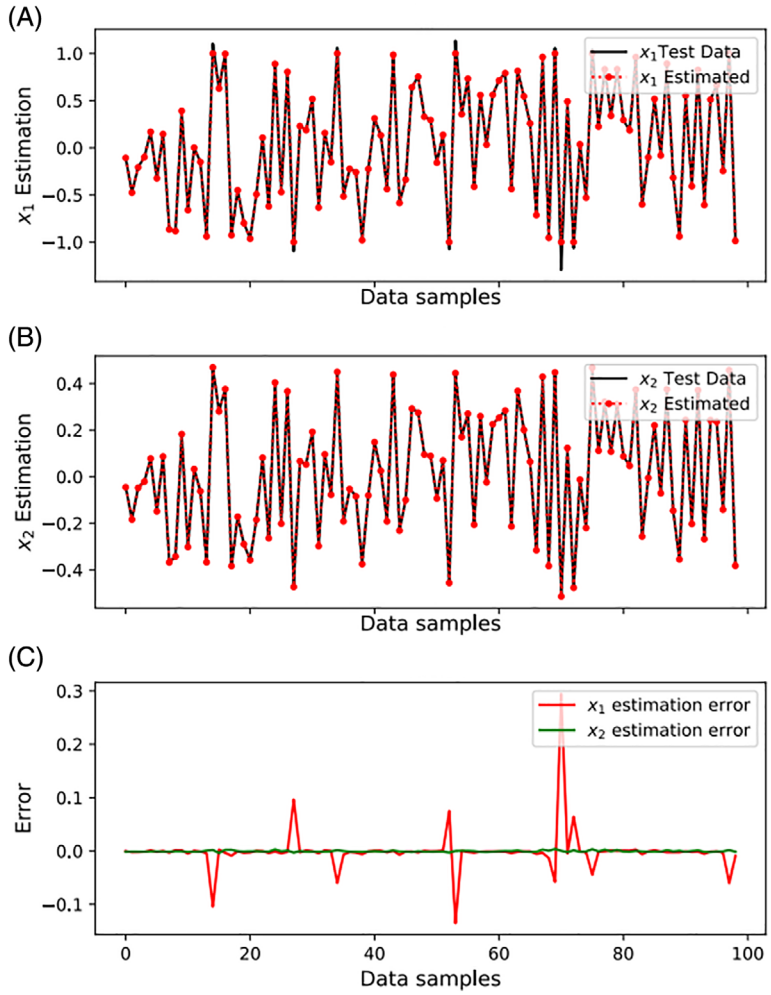

FIGURE 4 System states and training errors.

random data samples along with training errors which clearly remains close to zero and bounded. The SI-NN successfully learns the unknown nonlinear system dynamics (see Appendix A, Algorithm A1). Next, training of FCN-NN is done using the trained SI-NN (see Algorithm A1).

To that end, the FCN-NN is set up as shown in Figure 1 using a four layered fully connected NN with 16 hidden nodes in each layer, 2 input neurons, and one output neuron (2-16-16-1), with weights initialized in the interval $[-0.1, 0.1]$ and learning rate $\alpha_{F^{-1}} = 0.001$. A set of reference trajectories in the practical tracking region are generated comprising of 2000 data randomly generated. Next, the initial control policy is obtained by training FOSC-NN (see Algorithm 1) such that $F_e\left(e_k, u_{OS}^{(0)}(e_k)\right) = 0$ where $u_{OS}^{(0)}(e_k) = \widehat{\omega}_{OS}^T \sigma_{OS}\left(\widehat{\vartheta}_{OS}^T \; e_k\right)$ is the one-step zero control value is learnt (see (27) and (28)). FOSC-NN structure consists of five layered fully connected NN 2-32-16-8-1 with 3 hidden layers, 2 node input and one node output, with all activation functions chosen as nonlinear hyperbolic tangent function. The FOSC-NN is trained with randomly chosen 1000 data samples. Figure 5 shows the performance of trained FOSC-NN using 100 randomly sampled data. It is observed that FOSC-NN can learn the control law that drives the states near the origin (zero) in an approximative sense.

Next, the actor-critic NN weights are initialized randomly in $[-0.1, 0.1]$. The tolerance error in iterative algorithm is fixed as $\varepsilon = 10^{-6}$, maximum number of iterations is fixed as $n_i = 100$, critic and actor NN are chosen as five layered fully connected feedforward networks with: 3 input neurons, 16-8-4 hidden layer neurons and 2 outputs neurons, with activation functions set as hyperbolic tangent functions. The learning rates, tolerance errors, and maximal number of internal learning cycles (weight updating stochastic gradient descent step j) for the two networks are: $\alpha_c = \alpha_a = 0.001$, $\varepsilon_c = \varepsilon_a = 10^{-12}$, $n_c = n_a = 1000$.

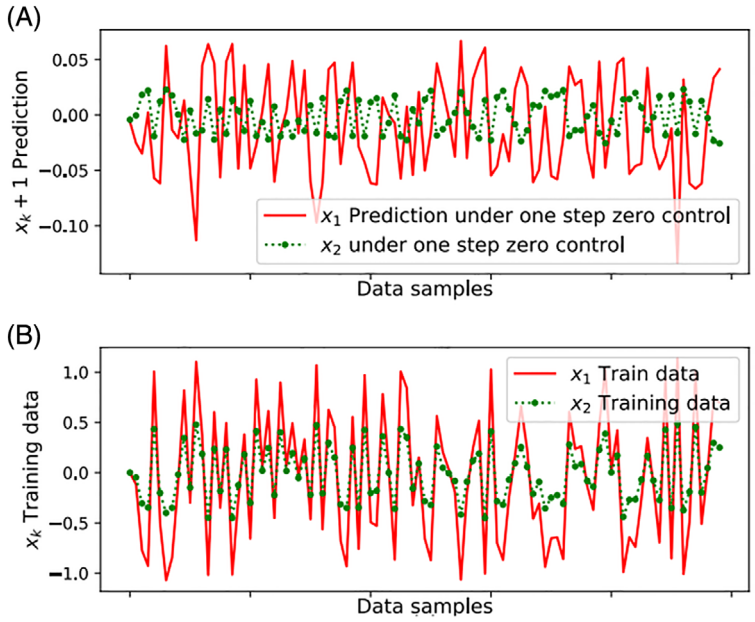

FIGURE 5 Learning of one step zero control law using feedforward one step-zero controller NN (FOSC-NN): (a) state prediction using initial control policy is obtained by training FOSC-NN, (b) randomly sampled data used for training FOSC-NN.

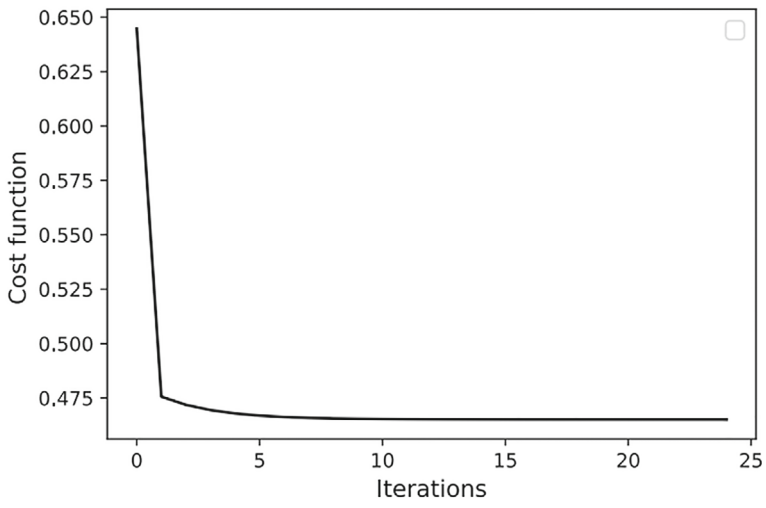

FIGURE 6 Cost function value with respect to number of iterations i for Example 1.

The iterative HDP algorithm is trained at $k = 0$ with an array of randomly chosen 10 initial states x_0. The critic and actor NN is trained for maximal $n_i = 30$ iterations (i.e., $i = 1, 2, \ldots 30$) with each iteration consisting of $n_c = n_a = 1000$ internal weight updating cycles. First, the critic-NN is trained to approximate the iterative cost function (33). The weights of critic NN are kept frozen while training the actor NN that approximates the iterative control law (32). This training procedure is repeated until the iterative cost function converges to an approximate optimal value, that is, $\left|V^{(i)}(x_0) - V^{(i+1)}(x_0)\right| \leq 10^{-12}$. The convergence shown in Figure 6 is attained in nearly 19 iterations.

This also leads to learning of the approximate optimal control law $v^*(x_0)$. The learnt approximate control law is applied to the system. Next, the state of the controlled system is initialized to be $\boldsymbol{x}_0 = [0.9 - 0.9]^T$ and the learnt control law is applied to the controlled system for 50-time steps. The reference trajectory is generated using (65) and the system state curves are shown in Figure 7A,B. The system is effectively able to track the reference trajectories. This demonstrates the

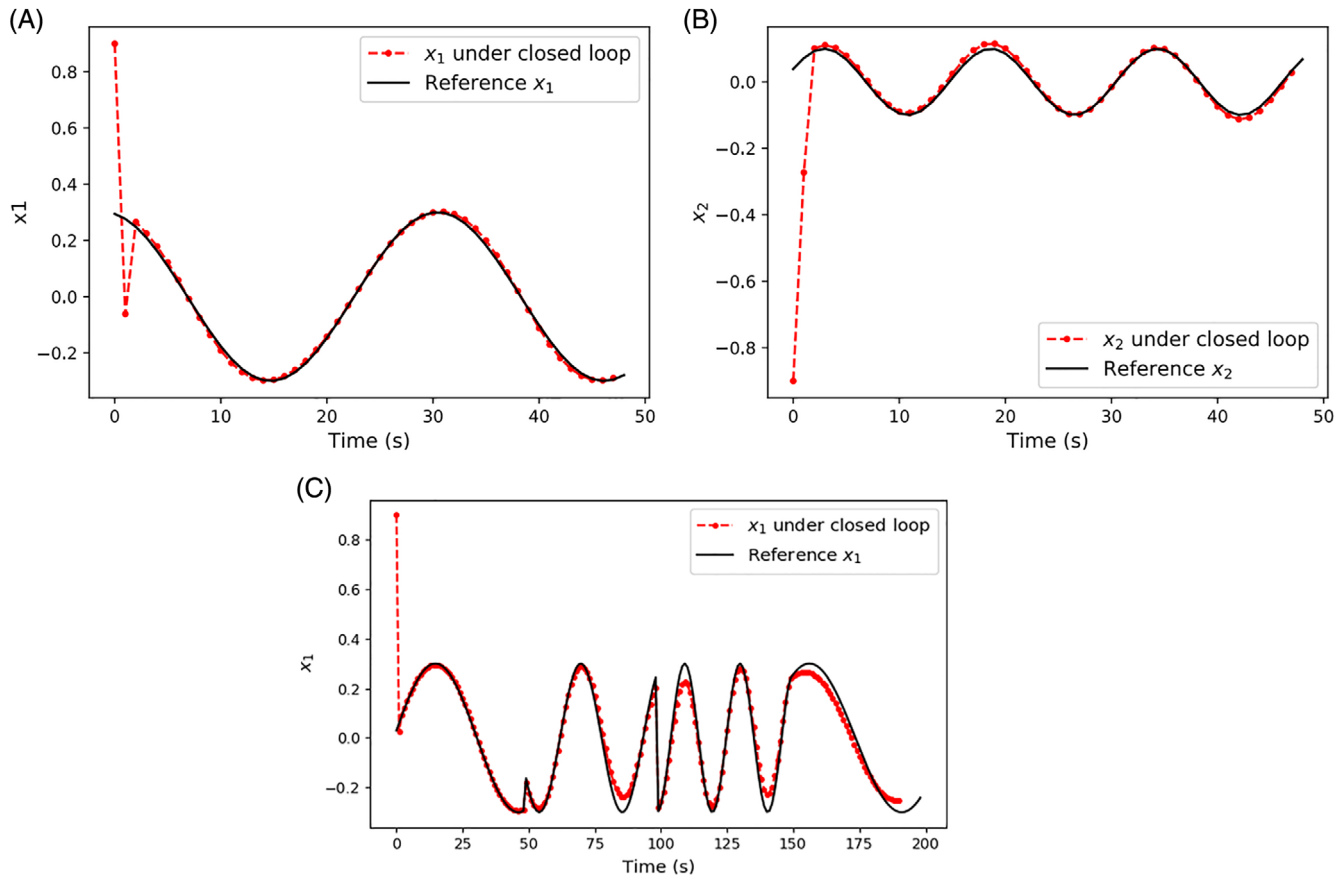

FIGURE 7 Reference tracking by system states of Example 1.

effectiveness of the proposed method. To further assess the efficiency of the overall procedure, Figure 7C shows tracking of a sinusoidal reference with four different arbitrary frequencies by the state x_1.

Example 2. To further check the effectiveness of the proposed algorithm, the control coefficient matrix is changed from a constant matrix to a function matrix. Consider the system inspired from Reference 31 with some modifications as:

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} sin\left(0.5x_{2,k}\right) x_{1k}^2 \\ cos\left(1.4x_{2,k}\right) sin\left(0.9x_{1,k}\right) \end{bmatrix} + \begin{bmatrix} -0.6sin\left(0.2x_{2,k}^2 + 0.1\right) & 0 \\ 0 & -0.4\left(1 - cos\left(0.1x_{1,k}\right)\right) \end{bmatrix} u_{p,k} \tag{64}$$

with $\boldsymbol{x}_k = [x_{1k} \;\; x_{2k}]^T \in \mathbb{R}^2$ as state variables $u_{p,k} \in \mathbb{R}^2$ as control variables. A similar approach is adopted wherein the SI-NN learns the unknown dynamics using Algorithm A1 (see Figure 8) FCN-NN is trained using SI-NN (Algorithm 1) and learns desire control profiles with respect to a given number of tracking references profiles.

Initial control policy is obtained by training FOSC-NN that provides the one-step zero control value (Algorithm 1). Actor-critic NNs are trained in similar manner with all parameters remaining same except: $\alpha_c = \alpha_a = 0.0001$, $\varepsilon_c = \varepsilon_a = 10^{-16}$, $n_c = n_a = 2000$. The iterative HDP algorithm is trained at $k = 0$ with an array of randomly chosen 100 initial states. The critic and actor NN is trained for maximal $n_i = 50$ iterations (i.e., $i = 1,2,\ldots 50$) until the iterative cost function converges to an approximate optimal value, that is, $\left|V^{(i)}(x_0) - V^{(i+1)}(x_0)\right| \leq 10^{-16}$ as shown in Figure 9. The approximate optimal control law $v^*(x_0)$ is learnt and applied to the system initialized at $\boldsymbol{x}_0 = [-0.5 \quad 0.9]^T$ for 100-time steps. The tracking performance as shown in Figure 10 demonstrates the effectiveness of the proposed approach.

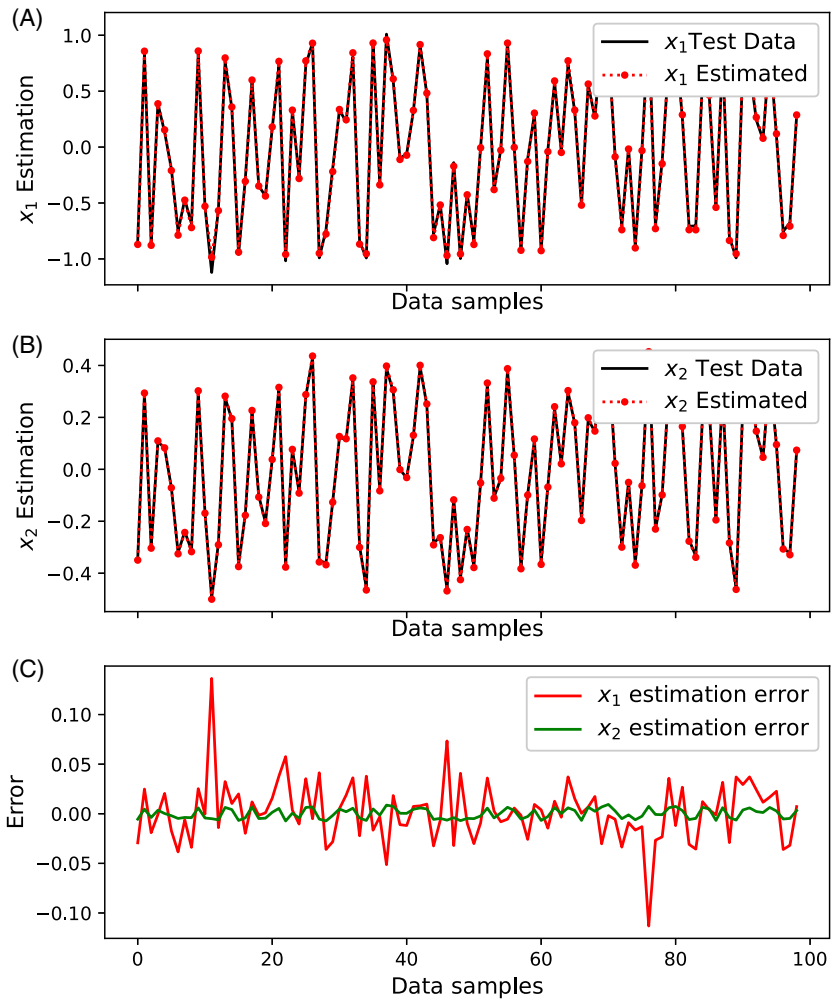

FIGURE 8 System states and training errors.

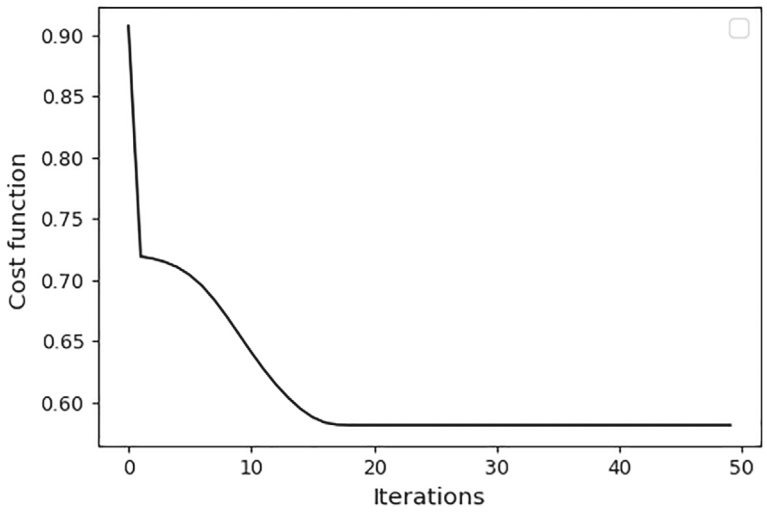

FIGURE 9 Cost function value with respect to number of iterations i for Example 2.

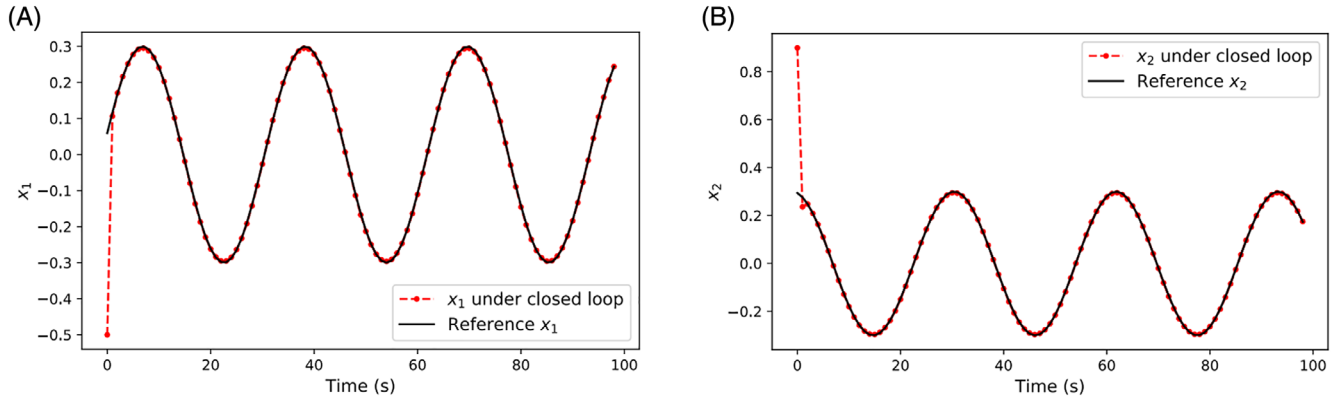

FIGURE 10 Reference tracking by system states of Example 1.

7 | CONCLUSIONS

The paper proposes a novel approach by considering consecutive changes in control inputs within the cost function while solving the optimal tracking problem for completely unknown discrete time nonlinear systems over a finite horizon. The latter leads to a novel performance index function over finite horizon and corresponding nonlinear Hamilton–Jacobi–Bellman equation that is solved in approximative iterative sense using a novel iterative ADP-based algorithm. The convergence of the novel algorithm to optimality(sub) over a finite horizon is established under the admissibility conditions through mathematically rigorous proofs. To find an initial one step zero control policy, a novel architecture is proposed to learn the one step zero control law which successfully brings the states near their origin (zero) in an approximative sense. The proposed iterative ADP is implemented using heuristic dynamic programming technique based on actor-critic NN structure. Finally, simulation studies are presented over two nonlinear systems with constant as well as varying control matrix to demonstrate the effectiveness of the proposed algorithm.

ACKNOWLEDGMENTS
The authors are grateful to the anonymous reviewers for their inputs.

AUTHOR CONTRIBUTION
MSJ conceived the idea and designed the analysis and proofs, wrote the paper, and performed the simulation analysis including the coding in Python, DT reviewed the paper, PW reveiwed the final manuscript.

DATA AVAILABILITY STATEMENT
Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

ORCID
Mayank Shekhar Jha https://orcid.org/0000-0002-6926-1386

REFERENCES
1. Lewis FL, Vrabie D, Syrmos VL. *Optimal control*. John Wiley & Sons; 2012.
2. Li N, Kolmanovsky I, Girard A. LQ control of unknown discrete-time linear systems—A novel approach and a comparison study. *Optim Control Appl Methods*. 2019;40(2):265-291.
3. Bellmann R. *Dynamic programming*. Princeton University Press; 1957.
4. Bertsekas DP, Bertsekas DP, Bertsekas DP, Bertsekas DP. *Dynamic programming and optimal control*. Vol 1. Athena scientific Belmont; 1995.
5. Su H, Zhang H, Zhang K, Gao W. Online reinforcement learning for a class of partially unknown continuous-time nonlinear systems via value iteration. *Optim Control Appl Methods*. 2018;39(2):1011-1028.
6. Werbos P. *Handbook of Intelligent Control*. Van Nostrand Reinhold New York Publishers; 1992:493-525.

7. Werbos PJ. Intelligence in the brain: A theory of how it works and how to build it. *Neural Netw*. 2009;22(3):200-212.
8. Sokolov Y, Kozma R, Werbos LD, Werbos PJ. Complete stability analysis of a heuristic approximate dynamic programming control design. *Automatica*. 2015;59:9-18.
9. Prokhorov DV, Santiago RA, Wunsch DC. Adaptive critic designs: A case study for neurocontrol. *Neural Networks [Internet]*. 1995;8(9):1367-1372. http://www.sciencedirect.com/science/article/pii/0893608095000429
10. Prokhorov DV, Wunsch DC. Adaptive critic designs. *IEEE Trans Neural Netw*. 1997;8(5):997-1007.
11. Langeron Y, Grall A, Barros A. A modeling framework for deteriorating control system and predictive maintenance of actuators. *Reliab Eng Syst Saf [Internet]*. 2015;140:22-36. http://www.sciencedirect.com/science/article/pii/S0951832015000940
12. Khelassi A, Theilliol D, Weber P, Ponsart JC. Fault-tolerant control design with respect to actuator health degradation: An LMI approach. 2011 IEEE International Conference on Control Applications (CCA), IEEE; 2011.
13. Zhang C, Xu X, Zhang X. Dual heuristic programming with just-in-time modeling for self-learning fault-tolerant control of mobile robots. *Optim Control Appl Methods*. 2021;44:1215-1234.
14. Chen Z, Chen S, Zhang Y, Deng Q, Zeng X. Online and hard constrained adaptive dynamic programming algorithm for energy storage control in smart buildings. *Optim Control Appl Methods*. 2021;44:1074-1091.
15. Liu Y, Xing Z, Chen Z, Xu J. Data-based robust optimal control of discrete-time systems with uncertainties via adaptive dynamic programming. *Optim Control Appl Methods*. 2021;44:1290-1304.
16. Liang Y, Zhang H, Zhang K, Wang R. A novel neural network discrete-time optimal control design for nonlinear time-delay systems using adaptive critic designs. *Optim Control Appl Methods*. 2020;41(3):748-764.
17. Ali SF, Padhi R. Optimal blood glucose regulation of diabetic patients using single network adaptive critics. *Optim Control Appl Methods*. 2011;32(2):196-214.
18. Gu J, Zhou J. An off-policy approach for model-free stabilization of linear systems subject to input energy constraint and its application to spacecraft rendezvous. *Optim Control Appl Methods*. 2020;41(3):948-959.
19. Al-Tamimi A, Lewis FL, Abu-Khalaf M. Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof. *IEEE Trans Syst Man, Cybern Part B*. 2008;38(4):943-949.
20. Zhang H, Wei Q, Luo Y. A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm. *IEEE Trans Syst Man, Cybern Part B*. 2008;38(4):937-942.
21. Dierks T, Thumati BT, Jagannathan S. Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence. *Neural Netw*. 2009;22(5–6):851-860.
22. Wang D, Liu D, Wei Q, Zhao D, Jin N. Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming. *Automatica [Internet]*. 2012;48(8):1825-1832. http://www.sciencedirect.com/science/article/pii/S0005109812002221
23. Liu F, Sun J, Si J, Guo W, Mei S. A boundedness result for the direct heuristic dynamic programming. *Neural Netw*. 2012;32:229-235.
24. Amato F, Ariola M. Finite-time control of discrete-time linear systems. *IEEE Trans Automat Contr*. 2005;50(5):724-729.
25. Haimo VT. Finite time controllers. *SIAM J Control Optim*. 1986;24(4):760-770.
26. Zattoni E. Structural invariant subspaces of singular Hamiltonian systems and nonrecursive solutions of finite-horizon optimal control problems. *IEEE Trans Automat Contr*. 2008;53(5):1279-1284.
27. Wang FY, Jin N, Liu D, Wei Q. Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound. *IEEE Trans Neural Netw*. 2010;22(1):24-36.
28. Wang D, Liu D, Wei Q. Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach. *Neurocomputing*. 2012;78(1):14-22.
29. Heydari A, Balakrishnan SN. Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics. *IEEE Trans Neural Networks Learn Syst*. 2012;24(1):145-157.
30. Huang Y, Liu D. Neural-network-based optimal tracking control scheme for a class of unknown discrete-time nonlinear systems using iterative ADP algorithm. *Neurocomputing [Internet]*. 2014;125:46-56. http://www.sciencedirect.com/science/article/pii/S0925231213001641
31. Mu C, Wang D, He H. Data-driven finite-horizon approximate optimal control for discrete-time nonlinear systems using iterative HDP approach. *IEEE Trans Cybern*. 2018;48(10):2948-2961.
32. Song R, Xie Y, Zhang Z. Data-driven finite-horizon optimal tracking control scheme for completely unknown discrete-time nonlinear systems. *Neurocomputing*. 2019;356:206-216.
33. Park YM, Choi MS, Lee KY. An optimal tracking neuro-controller for nonlinear dynamic systems. *IEEE Trans Neural Netw*. 1996;7(5):1099-1110.

How to cite this article: Jha MS, Theilliol D, Weber P. Model-free optimal tracking over finite horizon using adaptive dynamic programming. *Optim Control Appl Meth*. 2023;1-25. doi: 10.1002/oca.3028

APPENDIX A

Consider the estimated system state vector $\hat{x}_{k+1}$, estimated NN weights $\hat{\omega}_{m,k}$ and the ideal weight matrix ω_m^* at k. Then, the estimated system state can be expressed as:

$$\hat{x}_{k+1} = \hat{\omega}_{m,k}^T \sigma_m\left(\bar{z}_{m,k}\right) \tag{A1}$$

Further, the system identification error $\tilde{x}_{k+1}$ can be expressed as:

$$\tilde{x}_{k+1} = \hat{x}_{k+1} - x_{k+1} \quad = \left(\hat{\omega}_{m,k}^T - \omega_m^{*T}\right)\sigma_m\left(\bar{z}_{m,k}\right) - \varepsilon_{m,k} = \tilde{\omega}_{m,k}^T \sigma_m\left(\bar{z}_{m,k}\right) - \varepsilon_{m,k} \tag{A2}$$

where $\tilde{\omega}_{m,k} = \hat{\omega}_{m,k} - \omega_m^*$ is the weight identification error at time k. Now, considering $\phi_{m,k} = \tilde{\omega}_{m,k}^T \sigma_m\left(\bar{z}_{m,k}\right)$, we have:

$$\tilde{x}_{k+1} = \phi_{m,k} - \varepsilon_{m,k} \tag{A3}$$

The weights are adapted to minimize the identification error $E_m = (1/2)\tilde{x}_{k+1}\tilde{x}_{k+1}^T$ using gradient descent scheme[21]:

$$\hat{\omega}_{m,k+1} = \hat{\omega}_{m,k} - \alpha_m \frac{\partial E_{m,k+1}}{\partial \tilde{x}_{k+1}} \frac{\partial \tilde{x}_{k+1}}{\partial \hat{\omega}_{m,k}} = \hat{\omega}_{m,k} - \alpha_m \sigma_m\left(\bar{z}_k\right)\tilde{x}_{k+1}^T \tag{A4}$$

where $\alpha_m > 0$ is the learning rate of the SI-NN.

Theorem A1. *Consider the SI-NN defined by (A3) to identify the system (1), with weights being updated by (A4). Under **Assumption 3**, the system identification error is asymptotically stable $\tilde{x}_k$ and weight identifier error $\tilde{\omega}_k$ is uniformly ultimately bounded.*

Proof. (21,30): The following positive definite Lyapunov candidate is considered:

$$G_{m,k} = \tilde{x}_k^T \tilde{x}_k + \left(\frac{1}{\alpha_m}\right) \mathrm{tr}\left\{\tilde{\omega}_{m,k}^T \tilde{\omega}_{m,k}\right\} \tag{A5}$$

The first order change in time of the candidate is given as:

$$G_{m,k} = \left(\tilde{x}_{k+1}^T \tilde{x}_{k+1} - \tilde{x}_k^T \tilde{x}_k\right) + \left(\frac{1}{\alpha_m}\right) \mathrm{tr}\left\{\tilde{\omega}_{m,k+1}^T \tilde{\omega}_{m,k+1} + \tilde{\omega}_{m,k}^T \tilde{\omega}_{m,k}\right\} \tag{A6}$$

From (A3) and (A4):

$$\Delta G_{m,k} = \phi_{m,k}^T \phi_{m,k} - 2\phi_{m,k}^T \varepsilon_{m,k} + \varepsilon_{m,k}^T \varepsilon_{m,k} + \alpha_m \sigma_m^T\left(\bar{z}_k\right)\sigma_m\left(\bar{z}_k\right)\tilde{x}_{k+1}^T \tilde{x}_{k+1} - \tilde{x}_k^T \tilde{x}_k - 2\varphi_{m,k}^T \tilde{x}_{k+1} \tag{A7}$$

Applying Cauchy–Schwarz inequality, it is noted that $\tilde{x}_{k+1}^T \tilde{x}_{k+1} = \left(\phi_{m,k} - \varepsilon_{m,k}\right)^T \left(\phi_{m,k} - \varepsilon_{m,k}\right) \leq 2\left(\phi_{m,k}^T \phi_{m,k} + \varepsilon_{m,k}^T \varepsilon_{m,k}\right)$ and using (A7), leads to:

$$\Delta G_{m,k} \leq -\phi_{m,k}^T \phi_{m,k} + \varepsilon_{m,k}^T \varepsilon_{m,k} + 2\alpha_m \sigma_m^T\left(\bar{z}_k\right)\sigma_m\left(\bar{z}_k\right)\left(\phi_{m,k}^T \phi_{m,k} + \varepsilon_{m,k}^T \varepsilon_{m,k}\right) - \tilde{x}_k^T \tilde{x}_k - 2\varphi_{m,k}^T \varphi_{m,k} \tag{A8}$$

As $\|\sigma_m(.)\| \leq \sigma_M$ and $\varepsilon_{m,k}^T \varepsilon_{m,k} \leq \lambda_M \tilde{x}_k^T \tilde{x}_k$) (see Assumption 4), we have:

$$\Delta G_{m,k} \leq -(1 - 2\alpha_m \sigma_M^2)\left\|\phi_{m,k}\right\|^2 - \left(1 - \lambda_M - \lambda_M 2\alpha_m \sigma_M^2\right)\|\tilde{x}_k\|^2 \tag{A9}$$

Then, if learning rate is chosen as $0 \leq \alpha_m \leq 1$ with $\alpha_m \leq \beta^2/2\sigma_M^2$ for some $\beta \neq 0$ such that $\max\left\{-\sqrt{(1-\alpha_m)/\alpha_m}, -1\right\} \leq \beta \leq \min\left\{\sqrt{(1-\alpha_m)/\alpha_m}, 1\right\}$, it is seen that $\Delta G_{m,k} \leq 0$. Thus, $\tilde{x}_k$ and $\tilde{\omega}_m$ are bounded on a compact set S if $\tilde{x}_k(0)$ and $\tilde{\omega}_m(0)$ are bounded on S. Moreover, as $k \to \infty$, $\tilde{x}_k \to 0$. ∎

For SI-NN training, following steps are followed:

Algorithm A1. SI-NN training ([21,30] for detailed steps)

Step1: An array of system states, control input at time k, $z_{m,k}^T = \begin{bmatrix} x_k^T & u_{p,k}^T \end{bmatrix}$, is generated randomly (see (19)) and the corresponding array of states x_{k+1} is considered available for supervised training.
Step 2: The system identifier NN is trained by minimizing the error function using stochastic gradient descent scheme (A4), until the error (A2) converges to zero or remains inferior to a certain constant tolerance (set a priori). E.
Step 3: The trained SI-NN is tested against a test set to assess overfitting and underfitting issues and retrained if required.

APPENDIX B

The procedure comprises of trainable FCN-NN as input to an already-trained SI-NN in a cascading manner. Consider FCN-NN as a three-layer NN with n_2 number of neurons in each hidden layer as:

$$\begin{aligned} \hat{u}_{d,k} &= \omega_{F^{-1}}^{*T} \sigma_{F^{-1}} \left(\vartheta_{F^{-1}}^{*T} z_{F^{-1},k} \right) + \varepsilon_{F^{-1},k} \\ &= \omega_{F^{-1}}^{*T} \sigma_{F^{-1}} \left(\bar{z}_{F^{-1},k} \right) + \varepsilon_{F^{-1},k} \end{aligned} \tag{B1}$$

with $\hat{u}_{d,k}$ as the estimate output of desired control from FCNN, $\omega_{F^{-1}}^* \in \mathbb{R}^{(n_2 \times m)}$ and $\vartheta_{F^{-1}}^* \in \mathbb{R}^{n \times n_2}$ are the constant ideal weight matrices of FCNN, $z_{F^{-1},k}$ is the input, $\varepsilon_{F^{-1},k}$ is the NN function approximation error and $\sigma_{F^{-1}}(\cdot)$ is the NN activation function chosen as the hyperbolic tangent function, that is, $\sigma_{F^{-1}}(z) = (e^z - e^{-z}) / (e^z + e^{-z})$ so that $\sigma_{F^{-1}}(z) \in [-1, 1]$ and $\sigma_{F^{-1}}(z) \in \mathbb{R}^{n_2}$.

FCN-NN is trained by feeding-in set of reference values r_{k+1} as input and optimizing the FCN-NN weights such that SI-NN (trained a priori) outputs the true estimate of r_{k+1}: $\hat{r}_{k+1}$.

To that end, following steps are taken.

Algorithm B1. FCN-NN training.

Step 1: A set of reference trajectories are generated using a known reference generator (2)
Step 2: FCN-NN is fed with input $z_{F^{-1},k} = r_{k+1}$, following which the trained SI-NN is fed as inputs $z_{m,k}^T = \begin{bmatrix} r_k^T & \hat{u}_{d,k}^T \end{bmatrix}$, so that the output is $\hat{r}_{k+1}$
Step 3: FCN-NN is trained using the error $\tilde{r}_{F^{-1},k+1} = \hat{r}_{k+1} - r_{k+1}$ by minimizing the error function:

$$E_{F^{-1},k+1} = (1/2) \tilde{r}_{F^{-1},k+1}^T \tilde{r}_{F^{-1},k+1} \tag{B2}$$

Step 4: The weights of FCN-NN are adapted using a gradient descent-based optimization scheme.

$$\begin{aligned} \hat{\omega}_{F^{-1},k+1} &= \hat{\omega}_{F^{-1},k} - \alpha_{F^{-1}} \frac{\partial E_{F^{-1},k+1}}{\partial \hat{\omega}_{F^{-1},k}} \\ \hat{\omega}_{F^{-1},k+1} &= \hat{\omega}_{F^{-1},k} - \alpha_{F^{-1}} \frac{\partial E_{F^{-1},k+1}}{\partial \tilde{r}_{F^{-1},\ k+1}} \frac{\partial \tilde{r}_{F^{-1},k+1}}{\partial \hat{u}_{d,k}} \frac{\partial \hat{u}_{d,k}}{\partial \tilde{\omega}_{F^{-1},k}} \\ \hat{\omega}_{F^{-1},k+1} &= \hat{\omega}_{F^{-1},k} - \alpha_{F^{-1}} \sigma_{F^{-1}} \left(\bar{z}_{F^{-1},k} \right) \frac{\partial \tilde{r}_{F^{-1},k+1}}{\partial \hat{u}_{d,k}} \tilde{r}_{F^{-1},k+1}^T \end{aligned} \tag{B3}$$

where $\alpha_{F^{-1}} > 0$ is the learning rate. It should be noted that the error $\tilde{r}_{F^{-1},k+1}$ is propagated through the trained SI-NN and $\frac{\partial \tilde{r}_{F^{-1},k+1}}{\partial \hat{u}_{d,k}}$ is obtained through application of backpropagation algorithm from output of SI-NN $\hat{r}_{k+1}$ to its input $\hat{u}_{d,k}$.