Approximate Optimal Indirect Regulation of an Unknown Agent With a Lyapunov-Based Deep Neural Network

Wanjiku A. Makumi, Zachary I. Bell, and Warren E. Dixon, *Fellow, IEEE*

***Abstract*—An approximate optimal policy is developed for a pursuing agent to indirectly regulate an evading agent coupled by an unknown interaction dynamic. Approximate dynamic programming is used to design a controller for the pursuing agent to optimally influence the evading agent to a goal location. Since the interaction dynamic between the agents is unknown, integral concurrent learning is used to update a Lyapunov-based deep neural network to facilitate sustained learning and system identification. A Lyapunov-based stability analysis is used to show uniformly ultimately bounded convergence. Simulation results demonstrate the performance of the developed method.**

***Index Terms*—Deep neural networks, reinforcement learning, adaptive control, Lyapunov methods, nonlinear control systems.**

I. Introduction

THIS letter considers a class of indirect control problems where the states of a dynamic system are regulated by an influencing agent through an interaction dynamic. Specifically, this letter considers indirect herding as a subset of this class of problems. Unlike classical pursuit-evasion problems where the goal is achieved upon capture, herding problems consist of a pursuing agent intercepting and regulating an evading agent to a desired goal location, such as in [1], [2], [3], [4], [5].

Optimal solutions for indirect herding problems are sought in [6], [7], [8] using tools such as dynamic programming and calculus of variations. Drawbacks of such methods include computational inefficiency, due to the curse of dimensionality, and the need for known dynamics. Approximate dynamic programming (ADP) is an alternative approach that approximates the optimal value function, and thus the optimal control policy, via the Hamilton Jacobi Bellman (HJB) equation [9]. The solution to the HJB equation is typically estimated online using function approximation methods such as neural networks (NNs). A measure of suboptimality known as the bellman error (BE) is used as feedback to update the NNs and enhance the value function approximation online.

In the presence of model uncertainty, the BE can use an approximation of the model dynamics to update the value function approximation. The previous ADP indirect herding result in [10] used a NN to approximate the unknown drift dynamics; however, recent evidence shows that using a deep neural network (DNN) for system identification results in improved tracking performance [11].

Previously, [11] used a Lyapunov-based (Lb-) DNN for a control affine system with concurrent learning (CL). CL is an adaptive update scheme that uses input/output data to guarantee parameter convergence without requiring persistent excitation. CL requires estimates of the state derivatives if the true values are not known or measurable. In this letter, we are generalizing to a larger class of systems that are not control affine, and not even directly controlled, by using integral concurrent learning (ICL) to remove the need to measure the state derivatives. In existing literature, DNNs have never been used within an ICL-based system identification technique. The result in [10] used ICL solely for the output weights in a single-layer NN, but now we develop a framework that uses the integral data to additionally train the Lb-DNN inner features by optimizing an integral form of the loss.

In this letter, a multi-timescale Lb-DNN, similar to the one introduced in [11] and [12], is used for system identification. A multi-timescale framework is used to merge typically offline deep learning techniques with online adaptation to result in real-time deep learning. In contrast to those previous works, this multi-timescale framework consists of output-layer weights being updated in real-time via an ICL-based adaptive update law and inner-layer features being updated concurrent to real-time via iterative batch updates training on integrated data sets. The challenges associated with applying this framework to the ADP-based indirect herding problem include piecewise-in-time discontinuities in the dynamics' estimate, adaptation laws, and closed-loop error system from the iterative updates of the inner-layer features. These challenges

Manuscript received 17 March 2023; revised 19 May 2023; accepted 8 June 2023. Date of publication 26 June 2023; date of current version 12 July 2023. This work was supported in part by the Air Force Office of Scientific Research (AFOSR) under Grant FA9550-19-1-0169; in part by the Air Force Research Laboratory (AFRL) under Grant FA8651-21-F-1027; and in part by the Office of Naval Research under Grant N00014-21-1-2481. Recommended by Senior Editor T. Oomen. *(Corresponding author: Wanjiku A. Makumi.)*

Wanjiku A. Makumi and Warren E. Dixon are with the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: makumiw@ufl.edu; wdixon@ufl.edu).

Zachary I. Bell is with the Munitions Directorate, Air Force Research Laboratory, Eglin AFB, FL 32542 USA (e-mail: zachary.bell.10@us.af.mil).

Digital Object Identifier 10.1109/LCSYS.2023.3289474

restrict the adaptive update law used in [12] from being used in this problem, resulting in a new analysis used in this letter that considers piecewise-in-time discontinuities.

The primary contribution of this letter is the development of the indirect herding pursuit-evasion problem using an ICL-based multi-timescale Lb-DNN system identification approach to approximate the agents' unknown interaction dynamics online. The integral data collected online is used to update the output-layer weights and optimize the inner-layer features of the Lb-DNN in a new ICL-based deep learning technique. The deep learning and approximate optimal architecture is informed by a Lb-analysis that ensures uniformly ultimately bounded (UUB) convergence of the states as well as estimation of the control policy to within a neighborhood of the optimal control policy. Simulation results demonstrate the performance of the developed method and the improved function approximation compared to a single-layer NN.

II. Problem Formulation

The problem is formulated as a pursuing agent tasked with optimally intercepting and escorting an uncooperative evading agent to a desired goal state using unknown interaction dynamics between the pursuer and evader.[1] The evader dynamics are

$$\dot{z} = f(z, \eta), \tag{1}$$

where $z : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ denotes the state of the evader, $\eta : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ denotes the state of the pursuer, $t_0 \in \mathbb{R}_{\geq 0}$ denotes the initial time, and $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ denotes the non-affine, unknown locally Lipschitz interaction function. While the evader dynamics in (1) cannot be directly controlled, the evader can be influenced through interaction with the pursuer, which is directly controllable. The pursuer dynamics are

$$\dot{\eta} = h(z, \eta) + g(\eta)u, \tag{2}$$

where $h : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ denotes an unknown locally Lipschitz function representing the pursuer drift dynamics, $g : \mathbb{R}^n \to \mathbb{R}^{n \times m_\eta}$ denotes the known control effectiveness matrix, and $u : \mathbb{R}_{\geq t_0} \to \mathbb{R}^{m_\eta}$ is the pursuer's control input.

To quantify the control objective, a regulation error denoted by $e_z : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ is defined as

$$e_z \triangleq z - z_g, \tag{3}$$

where $z_g \in \mathbb{R}^n$ denotes a fixed user-defined goal location that is only known to the pursuer. It is not possible to directly control the error in (3). To address this, a backstepping formulation is used to design a virtual desired state that enables the pursuer to indirectly minimize (3) by tracking a virtual desired state denoted by $\eta_d : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$.[2] To quantify the pursuer's ability to track the virtual desired state, an auxiliary error $e_\eta : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ is defined as

$$e_\eta \triangleq \eta - \eta_d. \tag{4}$$

[1]This problem formulation follows the development in [10], including Assumptions 1 and 2 on the agent dynamics.

[2]Traditional backstepping cannot be used due to the nonlinear relationship in the dynamics; hence, additional error system development is motivated by backstepping approaches.

To quantify the virtual desired state objective, an additional auxiliary error $e_d : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ is defined as

$$e_d \triangleq \eta_d - z_g - k_d e_z, \tag{5}$$

where $k_d \in \mathbb{R}$ denotes a positive control gain. The time derivative of the virtual desired state η_d is designed as $\dot{\eta}_d \triangleq \mu_d$, where $\mu_d : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ is the subsequently designed virtual input that minimizes (5). To facilitate the minimization of (3)-(5), let $x \triangleq [e_z^\top, e_d^\top, e_\eta^\top]^\top$ and $x_d \triangleq [e_z^\top, e_d^\top, 0_{1\times n}]^\top$ denote the concatenated state and desired concatenated state, respectively. Additionally, the mappings $s_1, s_2 : \mathbb{R}^{3n} \to \mathbb{R}^n$ are defined as $s_1(x) \triangleq e_z + z_g$ and $s_2(x) \triangleq e_\eta + e_d + k_d e_z + z_g$. Using the error systems in (3)-(5), the evader and pursuer states are represented as $z = s_1(x)$ and $\eta = s_2(x)$, respectively.

Following the problem formulation in [10], a composite autonomous error system can be written as

$$\dot{x} = F(x) + G(x)\mu, \tag{6}$$

where $\mu \triangleq [\mu_\eta^\top \ \mu_d^\top]^\top \in \mathbb{R}^m$ is the total vector of control policies with $m = m_\eta + n$, where $\mu_\eta : \mathbb{R}_{\geq t_0} \to \mathbb{R}^{m_\eta}$ is defined as $\mu_\eta \triangleq u - u_d$, $u_d : \mathbb{R}_{\geq t_0} \to \mathbb{R}^{m_\eta}$ denotes a desired input defined as $u_d \triangleq g^+(\eta_d)(\mu_d - h(z, \eta_d))$ where locally Lipschitz pseudo inverse $g^+ : \mathbb{R}^n \to \mathbb{R}^{m_\eta \times n}$ is defined as $g^+ \triangleq (g^\top g)^{-1} g^\top$, and $F : \mathbb{R}^{3n} \to \mathbb{R}^{3n}$ and $G : \mathbb{R}^{3n} \to \mathbb{R}^{3n \times m}$ are defined as

$$F(x) \triangleq \begin{bmatrix} f(s_1(x), s_2(x)) \\ -k_d f(s_1(x), s_2(x)) \\ h(s_1(x), s_2(x)) - F_{sd}(x) \end{bmatrix},$$

and

$$G(x) \triangleq \begin{bmatrix} 0_{n\times m_\eta} & 0_{n\times n} \\ 0_{n\times m_\eta} & I_n \\ g(s_2(x)) & G_{sd}(x) \end{bmatrix},$$

where $F_{sd}(x) \triangleq g(s_2(x))g^+(s_2(x_d))h(s_1(x), s_2(x_d))$, and $G_{sd}(x) \triangleq g(s_2(x))g^+(s_2(x_d)) - I_n$. The pursuer's objective is achieved if $\eta \to \eta_d$ and $z, \eta_d \to z_g$; hence, e_z, e_η, and $e_d \to 0$.

The goal is to formulate an optimal control problem to regulate the states based on a given cost function. To minimize the errors in (3)-(5), μ_d and μ_η are designed to minimize the cost function

$$J(x, \mu) \triangleq \int_{t_0}^{\infty} Q(x) + P(x) + \mu^\top R \mu \ d\tau, \tag{7}$$

where $Q : \mathbb{R}^{3n} \to \mathbb{R}_{\geq 0}$ is a user-defined positive-definite (PD) function that satisfies $\underline{q}\|x\|^2 \leq Q(x) \leq \overline{q}\|x\|^2$ for all $x \in \mathbb{R}^{3n}$, where $\underline{q}, \overline{q} \in \mathbb{R}_{>0}$, $R \triangleq \text{blkdiag}\{R_\eta, R_d\}$, $R_\eta \in \mathbb{R}^{m_\eta \times m_\eta}$ and $R_d \in \mathbb{R}^{n\times n}$ are user-defined PD symmetric cost matrices, and $P : \mathbb{R}^{3n} \to \mathbb{R}$ is a positive semi-definite (PSD) user-defined penalty function described in [10].[3]

Following the standard actor-critic-based approximate optimal control framework (see [9], [13]) in [10], the optimal value function approximation $\widehat{V} : \mathbb{R}^{3n} \times \mathbb{R}^L \to \mathbb{R}$ is defined as

$$\widehat{V}(x, \widehat{W}_c) = \widehat{W}_c^\top \sigma(x), \tag{8}$$

[3]In this letter, both the Euclidean norm for vectors and the Frobenius norm for matrices are denoted by $\|\cdot\|$.

where $\widehat{W}_c \in \mathbb{R}^L$ is the critic weight estimate, and $\sigma : \mathbb{R}^{3n} \to \mathbb{R}^L$ is a user-selected bounded vector of basis functions. The control objective is to determine an approximation of the optimal control policy $\widehat{\mu} : \mathbb{R}^{3n} \times \mathbb{R}^L \to \mathbb{R}^m$, defined as

$$\widehat{\mu}\big(x, \widehat{W}_a\big) = -\frac{1}{2} R^{-1} G(x)^\top \nabla \sigma(x)^\top \widehat{W}_a, \tag{9}$$

where $\widehat{W}_a \in \mathbb{R}^L$ is the actor weight estimate, to minimize the cost given in (7). Minimizing this cost ensures that the errors in (3)-(5) are regulated to zero.

III. System Identification

A challenge for the control objective is that the approximate optimal control formulation requires the dynamic model of the pursuer and the evader. Since the interaction dynamics and pursuer dynamics are unknown, an approximation of the composite dynamics $F(x)$ must be used to approximate the solution to the HJB equation. The interaction dynamics between the pursuer and evader in (1) must be estimated using data collected online and in real-time to achieve the control objective since interaction data will often be unavailable *a priori*. The result in [10] estimated $F(x)$ online using a single layer NN and CL; however, recent evidence has shown that DNNs can learn more complex features and improve function approximation performance [14]. The recent results in [11] and [15] demonstrated a novel method for estimating dynamics online using a multi-timescale Lb-DNN framework with CL for system identification and control. Building on the previous results, this section develops an advanced ICL-based multi-timescale Lb-DNN framework.

The ICL-based multi-timescale learning framework approximates functions online by pairing a Lb-ICL adaptive update law for the output-layer weights of a DNN with a concurrent to real-time iterative ICL batch update for the inner-layer features of the DNN. Specifically, data is collected online in batches and each batch iteratively updates the inner-layer features of the Lb-DNN concurrent to real-time control using integral history stack data in a user-defined loss function and an optimizer such as Adam [16]. Since the inner-layer features are updated concurrent to real-time, but not in real-time like the output-layer weights, the inner-layer features actively used by the controller are iteratively switched to the most recently updated inner-layer features after a batch update.

Motivated by improved function approximation, (1) and (2) can be stacked and represented as

$$\breve{\dot{x}} = \phi(\Phi(x))\theta + \epsilon_\theta(x) + \breve{G}(x, u), \tag{10}$$

where the concatenated state derivative vector is defined as $\breve{\dot{x}} \triangleq [k_d \dot{z} \ \dot{\eta}]^\top \in \mathbb{R}^{2\times n}$, and $\breve{G}(x,u) \triangleq [0_{n\times 1} \ g(x)u]^\top \in \mathbb{R}^{2\times n}$.[4] The drift dynamics are approximated on a compact set $\mathcal{C} \subset \mathbb{R}^n$ with a DNN where $\theta \triangleq [\theta_z^\top \ \ \theta_\eta^\top]^\top \in \mathbb{R}^{p\times n}$ denotes an unknown bounded ideal output-layer weight matrix with the subscripts z and η representing the evader and pursuer dynamics, respectively, and $p = p_z + p_\eta$ is the total number of rows of θ. Additionally, $\phi(\Phi(x)) \triangleq \begin{bmatrix} \phi_z^\top(\Phi_z(x)) & 0_{1\times p_\eta} \\ 0_{1\times p_z} & \phi_\eta^\top(\Phi_\eta(x)) \end{bmatrix}$ where $\phi : \mathbb{R}^{2p} \to \mathbb{R}^{2\times p}$ denotes the user-defined basis functions and $\Phi(x) : \mathbb{R}^{3n} \to \mathbb{R}^{2p}$ denotes a function that represents the ideal DNN inner-layer features as $\Phi \triangleq [\Phi_z^\top \ \ \Phi_\eta^\top]^\top$, and $\epsilon_\theta(x) : \mathbb{R}^{3n} \to \mathbb{R}^{2\times n}$ denotes the function approximation errors. The i^{th} DNN-based estimate of the system dynamics is defined as

$$\hat{\breve{x}}_i = \phi\big(\widehat{\Phi}_i(x)\big)\widehat{\theta} + \breve{G}(x,u), \tag{11}$$

where $\widehat{\theta} \in \mathbb{R}^{p\times n}$ is the output-layer ideal weight matrix θ estimate, and $\widehat{\Phi}_i : \mathbb{R}^{3n} \to \mathbb{R}^{2p}$ is the i^{th} iteration selection of the inner features consisting of estimated inner-layer weights and user-selected activation functions.

Assumption 1: There is a constant weight matrix θ and known positive constants $\overline{\theta}$, $\overline{\phi}$, $\overline{\nabla_x \phi}$, $\overline{\epsilon_\theta}$, and $\overline{\nabla_x \epsilon_\theta} \in \mathbb{R}_{\geq 0}$, such that $\|\theta\| \leq \overline{\theta}$, $\sup_{x\in\mathcal{C}} \|\phi(\cdot)\| \leq \overline{\phi}$, $\sup_{x\in\mathcal{C}} \|\nabla_x \phi(x)\| \leq \overline{\nabla_x \phi}$, $\sup_{x\in\mathcal{C}} \|\epsilon_\theta(x)\| \leq \overline{\epsilon_\theta}$, and $\sup_{x\in\mathcal{C}} \|\nabla_x \epsilon_\theta(x)\| \leq \overline{\nabla_x \epsilon_\theta}$ [17, Ch. 4].

Assumption 2: The inner-layer features selection of $\widehat{\Phi}_i$ ensures that $\Phi(x) - \widehat{\Phi}_i(x) \leq \tilde{\Phi}_i(x)$, where $\tilde{\Phi}_i : \mathbb{R}^{3n} \to \mathbb{R}^{2p}$ is the function approximation error of the i^{th} iteration inner-layer Lb-DNN features, and $\sup_{x\in\mathcal{C},\, i\in\mathbb{N}} \|\tilde{\Phi}_i(x)\| \leq \overline{\tilde{\Phi}}$, where $\overline{\tilde{\Phi}} \in \mathbb{R}_{\geq 0}$ is a bounded constant for all i. Using the Mean Value Theorem, $\|\phi(\Phi(x)) - \phi(\widehat{\Phi}_i(x))\| \leq \overline{\nabla_x \phi}\, \overline{\tilde{\Phi}}$ [11].

Unlike the result in [11], which uses CL to learn the unknown ideal weights of the DNN, this result uses an ICL-based weight update policy. Following the ICL strategy in [18], let $\Delta t_\theta \in \mathbb{R}_{>0}$ be the time window of integration, where the integral of (10) at time $t_j \in [\Delta t_\theta, t]$ can be represented as $\Delta \breve{x}_j = \breve{x}(t_j) - \breve{x}(t_j - \Delta t_\theta) = \varphi_j \theta + \mathcal{E}_j + \mathcal{G}_j$ where $\varphi_j = \varphi(\widehat{\Phi}_i(t_j)) \triangleq \int_{t_j - \Delta t_\theta}^{t_j} \phi(\widehat{\Phi}_i(x(\tau)))d\tau$, $\mathcal{E}_j = \mathcal{E}(t_j) \triangleq \int_{t_j-\Delta t_\theta}^{t_j} \epsilon_\theta(x(\tau))d\tau$, and $\mathcal{G}_j = \mathcal{G}(t_j) \triangleq \int_{t_j-\Delta t_\theta}^{t_j} \breve{G}(x(\tau), u(\tau))d\tau$. An ICL-based parameter estimate update law is designed as

$$\dot{\widehat{\theta}}(t) = k_\theta \Gamma_\theta \sum_{j=1}^{M} \varphi_j^\top \big(\Delta \breve{x}_j - \mathcal{G}_j - \varphi_j \widehat{\theta}\big), \tag{12}$$

where $k_\theta, \Gamma_\theta \in \mathbb{R}_{>0}$ are update gains, and $M \in \mathbb{Z}_{>0}$ is the amount of data points saved for the history stack.

Remark 1: The i^{th} approximation of Φ is updated with the $i+1$th batch optimization using the ICL history stack data in the loss function

$$\mathcal{L}_{i+1} = \frac{1}{M} \sum_{j=1}^{M} \big\| \Delta \breve{x}_j - \mathcal{G}_j - \varphi_j \widehat{\theta} \big\|^2$$

and using Adam for the offline training optimization method.[5]

Assumption 3: There exists $T_1 \in \mathbb{R}_{>0}$ such that $T_1 > \Delta t_\theta$, and there exists a constant $\lambda_1 \in \mathbb{R}_{>0}$ that facilitates $\lambda_1 I_p \leq \sum_{j=1}^{M} \varphi_j^\top \varphi_j$, $\forall t \geq T_1$ [18].

IV. Online Learning

A. Bellman Error

The optimal value function $V^* : \mathbb{R}^{3n} \to \mathbb{R}_{\geq 0}$ and optimal control policy $\mu^* : \mathbb{R}^{3n} \to \mathbb{R}^m$ satisfy the HJB equation

$$0 = \nabla V^*(x)\big(F + G\mu^*\big) + Q(x) + P(x) + \mu^{*\top} R \mu^*, \tag{13}$$

[4] To streamline the subsequent development, a stacked matrix representation is used rather than a stacked vector representation.

[5] The estimates $\widehat{\Phi}_i$ are not computed *a priori*. Concurrent to real-time learning, input-output data is saved in a history stack, and then $\widehat{\Phi}$ is recomputed during the batch update.

where $V^*(0) = 0$. While (13) represents the HJB equation under optimal conditions, substituting the approximate terms from (8), (9), and (11), yields the BE

$$\delta\big(x,\widehat{\theta},\widehat{W}_c,\widehat{W}_a\big) \triangleq Q(x)+P(x)+\hat{\mu}^\top R\hat{\mu} + \nabla\widehat{V}\big(x,\widehat{W}_c\big)\big(\widehat{F}_i\big(x,\widehat{\theta}\big)+G(x)\widehat{\mu}\big(x,\widehat{W}_a\big)\big), \tag{14}$$

where

$$\widehat{F}_i\big(x,\widehat{\theta}\big) \triangleq \Bigg[\bigg(\frac{1}{k_d}\widehat{\theta}_z^\top\phi_z\big(\widehat{\Phi}_{z,i}(x)\big)\bigg)^\top, \Big(-\widehat{\theta}_z^\top\phi_z\big(\widehat{\Phi}_{z,i}(x)\big)\Big)^\top, \Big(\widehat{\theta}_\eta^\top\phi_\eta\big(\widehat{\Phi}_{\eta,i}(x)\big)\Big)^\top - \Big(g(x_\eta)g^+(\eta_d)\widehat{\theta}_\eta^\top\phi_\eta\big(\widehat{\Phi}_{\eta,i}(x_d)\big)\Big)^\top\Bigg]^\top$$

and $\widehat{\mu}(x,\widehat{W}_a) \triangleq [\widehat{\mu}_\eta^\top(x,\widehat{W}_a), \widehat{\mu}_d^\top(x,\widehat{W}_a)]^\top$ from (9). The pursuer controller is $\widehat{u}(x,\widehat{\theta},\widehat{W}_a) \triangleq \widehat{\mu}_\eta(x,\widehat{W}_a)+\widehat{u}_d(x,\widehat{\theta},\widehat{W}_a)$, where $\widehat{u}_d(x,\widehat{\theta},\widehat{W}_a) \triangleq g^+(\eta_d)(\widehat{\mu}_d(x,\widehat{W}_a)-\widehat{\theta}_\eta^T\phi_\eta(\widehat{\Phi}_{\eta,i}(x_d)))$. To facilitate the subsequent stability analysis, the BE can also be expressed in terms of the error $\widetilde{W}_c \triangleq W-\widehat{W}_c$ and $\widetilde{W}_a \triangleq W-\widehat{W}_a$. Subtracting (14) from (13) and, substituting (8) and (9), the BE in (14) can be rewritten as

$$\delta = -\omega^\top\widetilde{W}_c+\frac{1}{4}\widetilde{W}_a^\top G_\sigma\widetilde{W}_a-W^\top\nabla\sigma\widetilde{F}_i+O \tag{15}$$

where $\omega \triangleq \nabla\sigma(\widehat{F}_i+G\widehat{\mu})$, $\widetilde{F}_i \triangleq F-\widehat{F}_i$, $G_\sigma \triangleq \nabla\sigma G_R\nabla\sigma^\top$, $G_R \triangleq GR^{-1}G^\top$, and O is uniformly bounded over the compact set Ω.

As explained in [19], the user-selected state x_e can be used to evaluate the BE in (14) at off-trajectory points within the state space Ω. The extrapolated BEs are evaluated as $\delta_e \triangleq \delta(x_e,\widehat{\theta},\widehat{W}_c,\widehat{W}_a)$.

B. Actor and Critic Weight Update Laws

The on and off-trajectory BEs are used in the subsequently defined adaptive update laws to improve the actor and critic weight approximations online. The critic weight update law is defined as

$$\dot{\widehat{W}}_c \triangleq -\Gamma_c\Big(k_{c1}\frac{\omega}{\rho^2}\delta+\frac{k_{c2}}{N}\sum_{e=1}^{N}\frac{\omega_e}{\rho_e^2}\delta_e\Big), \tag{16}$$

and the least-squares gain matrix update law is defined as

$$\dot{\Gamma}_c \triangleq \beta_c\Gamma_c-\Gamma_c k_{c1}\frac{\omega\omega^\top}{\rho^2}\Gamma_c-\Gamma_c\frac{k_{c2}}{N}\sum_{e=1}^{N}\frac{\omega_e\omega_e^\top}{\rho_e^2}\Gamma_c, \tag{17}$$

where $\rho \triangleq 1+\gamma_1\omega^\top\omega$, $\rho_e \triangleq 1+\gamma_1\omega_e^\top\omega_e$, $\omega_e \triangleq \omega(\delta_e,\hat{\theta},\widehat{W}_a)$, and $k_{c1},k_{c2},\gamma_1,\beta_c \in \mathbb{R}_{>0}$ are user-defined learning gains. The actor weight update law is defined as

$$\dot{\widehat{W}}_a \triangleq -K_a k_{a1}\big(\widehat{W}_a-\widehat{W}_c\big)+K_a\frac{k_{c1}}{4}G_\sigma^\top\widehat{W}_a\frac{\omega^\top}{\rho^2}\widehat{W}_c - K_a k_{a2}\widehat{W}_a+K_a\frac{k_{c2}}{4N}\sum_{e=1}^{N}G_{\sigma e}^\top\widehat{W}_a\frac{\omega_e^\top}{\rho_e^2}\widehat{W}_c, \tag{18}$$

where $k_{a1},k_{a2} \in \mathbb{R}_{\geq 0}$ are user-defined learning gains, and $K_a \in \mathbb{R}^{L\times L}$ is a user-defined positive-definite symmetric matrix.

Assumption 4: There exist constants $T_2,\underline{c}_1,\underline{c}_2,\underline{c}_3 \in \mathbb{R}_{\geq 0}$ such that

$$\underline{c}_1 I_L \leq \inf_{t\in\mathbb{R}_{\geq t_0}}\frac{1}{N}\sum_{e=1}^{N}\frac{\omega_e\omega_e^\top}{\rho_e^2},$$

$$\underline{c}_2 I_L \leq \int_t^{t+T_2}\bigg(\frac{1}{N}\sum_{e=1}^{N}\frac{\omega_e(\tau)\omega_e^\top(\tau)}{\rho_e^2(\tau)}\bigg)d\tau,\ \forall t\in\mathbb{R}_{\geq t_0},$$

$$\underline{c}_3 I_L \leq \int_t^{t+T_2}\bigg(\frac{\omega(\tau)\omega^\top(\tau)}{\rho^2(\tau)}\bigg)d\tau,\ \forall t\in\mathbb{R}_{\geq t_0},$$

where T_2 and at least one of the constants $\underline{c}_1$, $\underline{c}_2$, or $\underline{c}_3$ is strictly positive [20].

Remark 2: See [10] for insight into Assumption (4).

V. Stability Analysis

Let $B_\zeta \subset \mathbb{R}^{3n+2L+np}$ represent a closed ball with a radius $\zeta \in \mathbb{R}_{>0}$ centered at the origin. Let $Z \in \mathbb{R}^{3n+2L+np}$ denote a concatenated state vector defined as $Z_L \triangleq [x^\top, \widetilde{W}_c^\top, \widetilde{W}_a^\top, Z_\theta^\top]^\top$ where $Z_\theta = \text{vec}(\tilde{\theta})$ and $\tilde{\theta} \triangleq \theta-\hat{\theta}$. Let $V_L : \mathbb{R}^{3n+2L+np}\times\mathbb{R}_{\geq t_0} \to \mathbb{R}$ denote a candidate Lyapunov function defined as

$$V_L(Z_L,t) \triangleq V^*(x,t)+\frac{1}{2}\widetilde{W}_c^\top\Gamma_c^{-1}(t)\widetilde{W}_c+\frac{1}{2}\widetilde{W}_a^\top K_a^{-1}\widetilde{W}_a + V_\theta(Z_\theta,t), \tag{19}$$

where $V_\theta(Z_\theta,t) \triangleq \frac{1}{2}\text{tr}(\tilde{\theta}^\top\Gamma_\theta^{-1}(t)\tilde{\theta})$, that can be bounded by class $\mathcal{K}$ functions $\underline{v}_l,\bar{v}_l : \mathbb{R}\to\mathbb{R}_{\geq 0}$ as

$$\underline{v}_l(\|Z_L\|) \leq V_L(Z_L,t) \leq \bar{v}_l(\|Z_L\|) \tag{20}$$

for all $t\in\mathbb{R}_{\geq t_0}$ where $Z_L \in \mathbb{R}^{3n+2L+np}$. The sufficient conditions for ultimate boundedness of Z are derived based on the subsequent analysis as

$$k_d \geq 1,\quad \lambda_{\min}\{H\}>0,\quad \sqrt{\frac{l}{\kappa}} \leq \underline{v}_l^{-1}(\bar{v}_l(\zeta)), \tag{21}$$

where $H \triangleq \begin{bmatrix} (\frac{k_{a1}+k_{a2}}{3}-\varphi_a) & -\frac{\varphi_{ac}}{2} & 0 \\ -\frac{\varphi_{ac}}{2} & \frac{k_{c2}\underline{c}}{3} & -\frac{\varphi_{c\theta}}{2} \\ 0 & -\frac{\varphi_{c\theta}}{2} & \frac{k_\theta\lambda_{\min}[\Sigma_\theta]}{2}\end{bmatrix}$, $\kappa \triangleq \min\{\frac{1}{2}\underline{q},\frac{1}{4}k_\theta\lambda_{\min}[\Sigma_\theta],\frac{1}{6}k_{c2}\underline{c},\frac{1}{6}(k_{a1}+k_{a2})\}$, $\underline{c} \triangleq (\frac{\beta_c}{2k_{c2}\overline{\Gamma}_c}+\frac{\underline{c}_1}{2})$, $\varphi_a \triangleq \frac{(k_{c1}+k_{c2})}{4}\overline{\|G_\sigma\|}\frac{k_\rho}{\sqrt{\gamma_1}}\|W\|+\frac{1}{2}\frac{1}{\lambda_{\min}\{K_a\}}\overline{\|\nabla WG_R\nabla\sigma^\top\|}$, $\Sigma_\theta \triangleq [\sum_{j=1}^{M}\varphi_j^\top\varphi_j]$, $\varphi_{ac} \triangleq k_{a1}+\frac{k_{c1}+k_{c2}}{4}\overline{\|G_\sigma\|\|W\|}\frac{k_\rho}{\sqrt{\gamma_1}}+\frac{1}{2}\frac{1}{\underline{\Gamma}_c}\overline{\|\nabla W\|\|G_R\|\|\nabla\sigma^\top\|}$, $\varphi_{c\theta} \triangleq (k_{c1}+k_{c2})\frac{k_\rho}{\sqrt{\gamma_1}}(\overline{\|W^\top\|\|\nabla\sigma\|\|\phi\|}(1+\frac{1}{k_d}+\overline{\|g\|\|g^+\|}))$, and $l\in\mathbb{R}_{>0}$ is a constant that depends on the bounded NN constants.

In contrast to the result in [10], the multi-timescale Lb-DNN identifier introduces piecewise-in-time discontinuities in the dynamics which complicates the stability analysis in the sense that common actor-critic methods cannot be readily applied in the stability analysis of the closed-loop system. The following theorem contains a Lyapunov-like stability analysis which considers functions containing discontinuities that are piecewise continuous in time.

Theorem 1: Provided all assumptions are satisfied, and conditions in (21) are met, then the error state x, the critic weight estimate error $\widetilde{W}_c$, the actor weight estimate error $\widetilde{W}_a$, and the parameter estimation error $\tilde{\theta}$ are UUB. Hence, the approximate control policy $\widehat{\mu}$ converges to a neighborhood of the optimal control policy μ^*.

Proof: Taking the time derivative of (19), and substituting (13), $\dot{V}^*(x) = \nabla V^*(F(x) + G(x)\mu)$, $\tilde{\dot{W}}_c \triangleq \dot{W} - \hat{\dot{W}}_c$, $\tilde{\dot{W}}_a \triangleq \dot{W} - \hat{\dot{W}}_a$, and $\dot{W} \triangleq \nabla W(x)(F(x) + G(x)\mu)$ yields

$$\dot{V}_L = \nabla V^*(F + G\mu) + \dot{V}_\theta(Z_\theta) - \frac{1}{2}\widetilde{W}_c^\top \left(\Gamma_c^{-1}\dot{\Gamma}_c\Gamma_c^{-1}\right)\widetilde{W}_c + \widetilde{W}_c^\top \Gamma_c^{-1}\left(\nabla W(F + G\mu) - \hat{W}_c\right) + \widetilde{W}_a^\top K_a^{-1}\left(\nabla W(F + G\mu) - \hat{W}_a\right).$$

Using (14), the update laws in (12) and (16)-(18) [10, eqs. (29)-(31)], $\widehat{W}_a = W - \widetilde{W}_a$, $\widehat{W}_c = W - \widetilde{W}_c$, Assumptions 1-4, and implementing bounding and completing the square yields $\dot{V}_L \leq -\kappa\|Z_L\|^2 - \kappa\|Z_L\|^2 + l - Z_v^\top H Z_v$, where $Z_v \triangleq [\,\|\widetilde{W}_a\|,\ \|\widetilde{W}_c\|,\ \|Z_\theta\|\,]^\top$. Specifically, Assumption 2 is used to bound the system identification parameter estimation term $\dot{V}_\theta$ in the Lyapunov function. Provided the sufficient conditions in (21) are met, then $\dot{V}_L$ can be bounded as

$$\dot{V}_L \leq -\kappa\|Z_L\|^2,\ \forall \|Z_L\| \geq \sqrt{\frac{l}{\kappa}} > 0. \tag{22}$$

As a result of the discontinuities in the update laws in (12) and (16)-(18) being piecewise continuous in time, and by using (21) and (22), [21, Th. 4.18] can be enforced to conclude that Z_L is UUB such that $\|Z_L\| \leq \underline{\nu}^{-1}(\overline{\nu}(\sqrt{\frac{l}{\kappa}}))$ and $\widehat{\mu}$ converges to a neighborhood around the optimal policy μ^*. Since $Z_L \in \mathcal{L}_\infty$, then x, $\widetilde{W}_c$, $\widetilde{W}_a$, $\tilde{\theta} \in \mathcal{L}_\infty$ and thus $\mu \in \mathcal{L}_\infty$. Moreover, since $x \in \mathcal{L}_\infty$, and since W is a continuous function of x, it follows that $W(x) \in \mathcal{L}_\infty$. Furthermore, since $x \in \mathcal{L}_\infty$, then e_η, e_z, $e_d \in \mathcal{L}_\infty$. Using (3)-(5), $z \in \mathcal{L}_\infty$, and $\eta_d \in \mathcal{L}_\infty$; hence, η, $(z - \eta) \in \mathcal{L}_\infty$ follows. Lastly, since η_d, μ, g^+, $\tilde{\theta} \in \mathcal{L}_\infty$, it follows that $\widehat{\theta}$, $u_d \in \mathcal{L}_\infty$ and $u \in \mathcal{L}_\infty$. ■

VI. Simulations

An example scenario is simulated to illustrate the performance of the developed ICL-Lb-DNN ADP architecture where an evader and pursuer are uniformly randomly placed in a 1000 × 1000 unit area with the goal of position control ($n = 2$). The goal region is set to a uniformly random location within a 100 unit radius of the pursuer while the evader is uniformly randomly initialized at least 500 units from the goal region. The pursuer must therefore leave the goal area to catch the evader, learn the interaction dynamics in real-time using the deep ICL learning architecture, and approximate the optimal influencing policy using ADP. The typical performance of the architecture in simulation is shown in Figure 1 to indirectly control the position of the evader, where the pursuer is initially in the top-right (blue circle with white plus) near the goal (orange circle) and the evader is initially in the bottom-left (orange circle with white plus).

Without loss of generality, the dynamics for the pursuer were $h(z, \eta) = 0_{2\times 1}$ and $g(\eta) = I_{2\times 2}$. The evader dynamics were $f(z, \eta) = (z - \eta)\exp(-\frac{1}{20{,}000}(z - \eta)^\top(z - \eta))$. In the simulation, the ICL-DNN function approximation was implemented using PyTorch, and all the history stack data was collected online in real-time (approximately 45 Hz). The ICL-Lb-DNN and the history stack remained on the graphics card for optimization using a maximum of approximately 1GB of

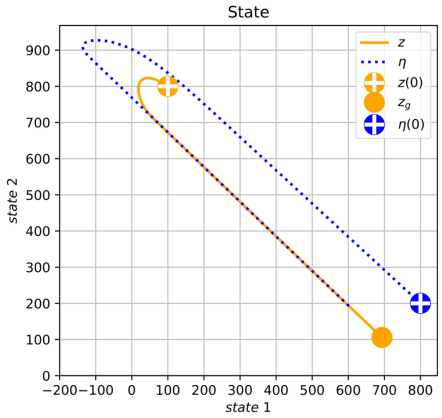

Fig. 1. Simulation example where the pursuer is initialized in the bottom-right (blue circle with white plus), the goal region is to the left of the pursuer (orange circle), and the evader is initialized in the top-left (orange circle with white plus). The pursuer trajectory and evader trajectory over the experiment are shown in blue and orange, respectively. Simulation shows evader initially flees towards top-left; however, the pursuer approximates the interaction dynamics and optimal policy in real-time and quickly escorts the evader to the goal region.

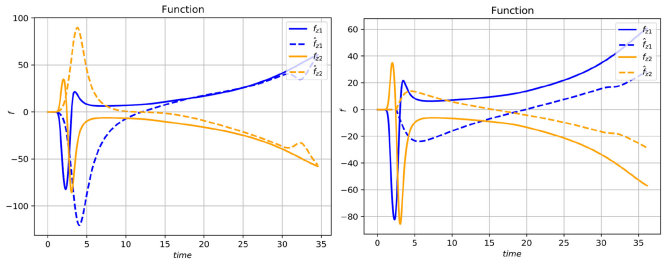

Fig. 2. Function approximation where the true values are shown in solid lines and the estimated values are shown in dashed lines. The ICL-DNN estimates quickly converged near the true values using the data collected online. The left figure shows the ICL-DNN approximation and right figure shows the ICL-SNN approximation demonstrating that the ICL-DNN outperforms the ICL-SNN.

memory. At each time step, the data was added to the history stack which was a sliding buffer containing the most recent second of data ($\Delta t_\theta = 1.0$ second). At each time step the integrals of the data were approximated using the trapezoidal rule to update the output weights using (12) and update the DNN inner-layer features using the loss discussed in Remark 1, where a single optimizer step was performed for each simulation step on a batch of integral data from the history stack using Adam with a linearly annealing learning rate (initialized to 0.001 and linearly decayed to 0.0001). To enable online optimization, the DNN was constrained to 3 inner layers, each with 64 neurons and hyperbolic tangent activation functions while the final layer had 64 output weights. The output weights and inner-layer weights were randomly initialized using a zero mean and standard deviation of 0.01 ($\hat{\theta}_z(0) \sim \mathcal{N}(0, 0.01)$) with $\Gamma_\theta = 0.1$ and $k_\theta = 1.0$. The function approximation results in Figure 2 show that in the 35 second simulation, the ICL-DNN function approximation converges to within 10% of the true value of the nonlinear interaction dynamics while the loss converges to 0.04. Additionally, the shallow NN (SNN) from [10], with 256 output weights, converges to within 50%

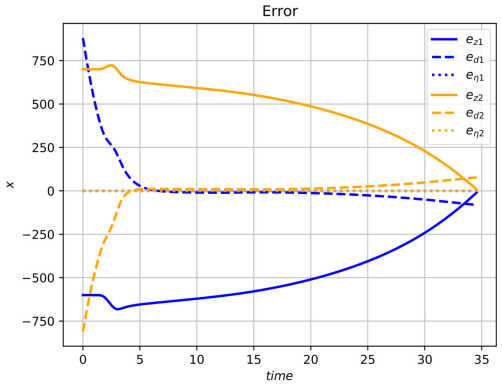

Fig. 3. The evader tracking error steadily decays after the evader initially flees. The auxiliary errors also converge to a small radius of the goal.

of the true value of the dynamics demonstrating the DNN outperforms the SNN used in [10].

The efficient StaF kernels method from [20] was used to approximate the optimal policy online in real-time (approximately 45 Hz) while simultaneously estimating the dynamics using the ICL-DNN function approximation. The value function was approximated using 7 StaF kernels $\sigma(x, c(x)) = \left[\sigma_1(x, c_1(x)) \ldots \sigma_7(x, c_7(x))\right]^\top$ where each kernel $\sigma_q(x(t), c_q(x(t))) = \frac{x^\top(t)c_q(x(t))}{\|x(0)\|^2}$, $c_q(x(t)) = x(t) + \|x(0)\|\nu(x(t))d_q$, $\nu(x(t)) = \frac{x^\top(t)x(t)+0.01\|x(0)\|^2}{\|x(0)\|^2+x^\top(t)x(t)}$, and d_q are the vertices of a 6-simplex. The actor and critic weights were initialized as $\widehat{W}_a(0) = 1_7$ and $\widehat{W}_c(0) = 2\widehat{W}_a(0)$ while $\Gamma_c(0) = 5I_{7\times 7}$. The gains used to update the weights were selected as $k_{c1} = 0.9$, $k_{c2} = 0.1$, $K_a = 1.0$, $k_{a1} = 0.25$, $k_{a2} = 0.005$, $\beta_c = 0.001$, $\gamma_1 = 0.75$, and $N = 10$ extrapolation points were selected within a radius of $\nu(x)$ of x. The cost and control gains selected were $Q = 0.0001I_{6\times 6}$, $R = 0.01$, and $k_d = 1.3$. Using the selected gains resulted in excellent tracking performance as shown by the tracking errors in Figure 3 where the tracking error $e_z \to 0$. These results demonstrate that the DNN-ICL-based ADP architecture is an excellent approach for real-time approximation of the optimal policy when dynamics are unknown and highly nonlinear.

VII. Conclusion

A deep ICL-based implementation of ADP is presented to achieve an approximate optimal online solution to the indirect regulation herding problem for unknown agents. An ICL-based system identifier is facilitated by a Lb-DNN to estimate the unknown interaction dynamic between the pursuer and evader. A Lb-analysis is provided to prove UUB convergence of the evader to the desired goal location known by the pursuer. The simulation shows that the pursuer is able to intercept and regulate the evader towards the desired goal location and that the Lb-DNN system identifier outperforms the SNN system identifier. Further investigations into directly learning the Q-function are motivated from offline learning results such as [22]. Other potential future work includes investigating the optimality of the evader's behavior.

Acknowledgment

Any opinions, findings and 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] R. A. Licitra, Z. I. Bell, E. A. Doucette, and W. E. Dixon, "Single agent indirect herding of multiple targets: A switched adaptive control approach," *IEEE Contr. Syst. Lett.*, vol. 2, no. 1, pp. 127–132, Jan. 2018.
[2] R. A. Licitra, Z. I. Bell, and W. E. Dixon, "Single-agent indirect herding of multiple targets with uncertain dynamics," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 847–860, Aug. 2019.
[3] A. Pierson and M. Schwager, "Controlling noncooperative herds with robotic herders," *IEEE Trans. Robot.*, vol. 34, no. 2, pp. 517–525, Apr. 2018.
[4] V. S. Chipade and D. Panagou, "Multiagent planning and control for swarm herding in 2-d obstacle environments under bounded inputs," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1956–1972, Dec. 2021.
[5] K. Elamvazhuthi, Z. Kakish, A. Shirsat, and S. Berman, "Controllability and stabilization for herding a robotic swarm using a leader: A mean-field approach," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 418–432, Apr. 2021.
[6] P. Kachroo, S. A. Shedied, J. S. Bay, and H. Vanlandingham, "Dynamic programming solution for a class of pursuit evasion problems: The herding problem," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 1, pp. 35–41, Feb. 2001.
[7] A. D. Khalafi and M. R. Toroghi, "Capture zone in the herding pursuit evasion games," *Appl. Math. Sci.*, vol. 5, no. 39, pp. 1935–1945, 2011.
[8] S. A. Shedied, "Optimal trajectory planning for the herding problem: A continuous time model," *Int. J. Mach. Learn. Cybern.*, vol. 4, no. 1, pp. 25–30, 2013.
[9] R. Kamalapurkar, P. S. Walters, J. A. Rosenfeld, and W. E. Dixon, *Reinforcement Learning for Optimal Feedback Control: A Lyapunov-Based Approach*. Cham, Switzerland: Springer, 2018.
[10] P. Deptula, Z. I. Bell, F. M. Zegers, R. A. Licitra, and W. E. Dixon, "Approximate optimal influence over an agent through an uncertain interaction dynamic," *Automatica*, vol. 134, pp. 1–13, Dec. 2021.
[11] M. L. Greene, Z. I. Bell, S. Nivison, and W. E. Dixon, "Deep neural network-based approximate optimal tracking for unknown nonlinear systems," *IEEE Trans. Autom. Control*, vol. 68, no. 5, pp. 3171–3177, May 2023.
[12] R. Sun, M. L. Greene, D. M. Le, Z. I. Bell, G. Chowdhary, and W. E. Dixon, "Lyapunov-based real-time and iterative adjustment of deep neural networks," *IEEE Contr. Syst. Lett.*, vol. 6, pp. 193–198, 2021.
[13] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*. Stevenage, U.K.: Inst. Eng. Technol., 2013.
[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[15] Z. I. Bell, R. Sun, K. Volle, P. Ganesh, S. A. Nivison, and W. E. Dixon, "Target tracking subject to intermittent measurements using attention deep neural networks," *IEEE Control Syst. Lett.*, vol. 7, pp. 379–384, 2023.
[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[17] F. L. Lewis, S. Jagannathan, and A. Yesildirak, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Philadelphia, PA, USA: CRC Press, 1998.
[18] 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, no. 12, pp. 1775–1787, Dec. 2019.
[19] R. Kamalapurkar, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for approximate optimal regulation," *Automatica*, vol. 64, pp. 94–104, Feb. 2016.
[20] R. Kamalapurkar, J. A. Rosenfeld, and W. E. Dixon, "Efficient model-based reinforcement learning for approximate online optimal control," *Automatica*, vol. 74, pp. 247–258, Dec. 2016.
[21] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.
[22] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.