Non-Parametric Neuro-Adaptive Formation Control

Christos K. Verginis, Zhe Xu, and Ufuk Topcu

***Abstract*—We develop a learning-based algorithm for the distributed formation control of networked multi-agent systems governed by unknown, nonlinear dynamics. Most existing algorithms either assume certain parametric forms for the unknown dynamic terms or resort to unnecessarily large control inputs in order to provide theoretical guarantees. The proposed algorithm avoids these drawbacks by integrating neural network-based learning with adaptive control in a two-step procedure. In the first step of the algorithm, each agent learns a controller, represented as a neural network, using training data that correspond to a collection of formation tasks and agent parameters. These parameters and tasks are derived by varying the nominal agent parameters and a user-defined formation task to be achieved, respectively. In the second step of the algorithm, each agent incorporates the trained neural network into an online and adaptive control policy in such a way that the behavior of the multi-agent closed-loop system satisfies the user-defined formation task. Both the learning phase and the adaptive control policy are distributed, in the sense that each agent computes its own actions using only local information from its neighboring agents. The proposed algorithm does not use any a priori information on the agents' unknown dynamic terms or any approximation schemes. We provide formal theoretical guarantees on the achievement of the formation task.**

***Note to Practitioners*—This paper is motivated by control of multi-agent systems, such as teams of robots, smart grids, or wireless sensor networks, with uncertain dynamic models. Existing works develop controllers that rely on unrealistic or impractical assumptions on these models. We propose an algorithm that integrates offline learning with neural networks and real-time feedback control to accomplish a multi-agent task. The task consists of the formation of a pre-defined geometric pattern by the multi-agent team. The learning module of the proposed algorithm aims to learn stabilizing controllers that accomplish the task from data that are obtained from offline runs of the system. However, the learned controller might result in poor performance owing to potential data inaccuracies and the fact that learning algorithms can only approximate the stabilizing controllers. Therefore, we complement the learned controller with a real-time feedback-control module that adapts on the fly to such discrepancies. In practise, the data can be collected from pre-recorded trajectories of the multi-agent system, but these trajectories do need to accomplish the task at hand. The real-time feedback-control is a closed-form function of the states of each agent and its neighbours and the trained neural networks and can be straightforwardly implemented. The experimental results show that the proposed algorithm achieves greater performance than algorithms that use only the trained neural networks or only the real-time feedback-control policy. Our future research will address the sensitivity of the algorithm to the quality and quantity of the employed data as well as to the learning performance of the neural networks.**

This work was supported in part by the grants NSF CNS 2304863, CNS 2339774, IIS 2332476, 2214939, AFOSR FA9550-19-1-0005, and ONR N00014-23-1-2505.

C. K. Verginis is with Uppsala University, Uppsala, Sweden. E-mail:christos.verginis@angstrom.uu.se. Z. Xu is with Arizona State University, Tempe, Arizona, USA. E-mail: xzhe1@asu.edu. U. Topcu is with the University of Texas at Austin, Austin, Texas, USA. E-mail:utopcu@utexas.edu.

I. Introduction

During the last decades, decentralized control of networked multi-agent systems has attracted significant attention due to the great variety of its applications, including multi-robot systems, transportation, multi-point surveillance as well as biological systems [1]–[3]. In such systems, each agent calculates its own actions based on local information, as modeled by a connectivity graph, without relying on any central control unit. This absence of central control and global information motivates leader-follower architectures, where a team of agents (followers) aims at following a pre-assigned leader agent that holds information about the execution of a potential task. The coordination problem of leader–follower architectures has been the focus of many works [4]–[9] because of its numerous applications in various disciplines including autonomous vehicles coordination (satellite formation flying, cooperative search of unmanned aerial vehicles and synchronization of Euler–Lagrange systems), systems biology (control and synchronization in cellular networks), and power systems (control of renewable energy microgrids).

Although many works on distributed cooperative control consider known and simple dynamic models, there exist many practical engineering systems that cannot be modeled accurately and are affected by unknown exogenous disturbances. Thus, the design of control algorithms that are robust and adaptable to such uncertainties and disturbances is important. For multi-agent systems, ensuring robustness is particularly challenging due to the lack of global information and the interacting dynamics of the individual agents. A promising step towards the control of systems with uncertain dynamics is the use of data obtained a priori from system runs. However, engineering systems often undergo purposeful modifications (e.g., substitution of a motor or link in a robotic arm or exposure to new working environments) or suffer gradual faults (e.g., mechanical degradation), which might change the systems' dynamics or operating conditions. Therefore, one cannot rely on the aforementioned data to provably guarantee the successful control of the system. On the other hand, the exact incorporation of these changes in the dynamic model, and consequently, the design of new model-based algorithms, can be a challenging and often impossible procedure. Hence, the goal in such cases is to exploit the data obtained a priori and construct intelligent online policies that achieve a user-defined task while adapting to the aforementioned changes.

There are numerous works that focus on formation control with uncertain dynamics, exhibiting however certain limitations. Adaptive-control techniques usually consider conservative assumptions on the underlying dynamics, such as linear parametrizations with constant unknown terms [10], [11], globally Lipschitz functions, growth conditions, or known

This article has been accepted for publication in IEEE Transactions on Automation Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TASE.2025.3528501

upper bounds [6], [12]–[14]. Similarly, reinforcement-learning works approximate the unknown dynamic terms via single-layer neural networks, rendering, however, the performance of the closed-loop system dependent on the number of layers used [15], [16]. Works that do not adopt such assumptions, such as funnel control [17], design high-gain algorithms that might lead to unnecessarily large control inputs. More information on the related works is given in Section I-B.

A. Contributions

This paper addresses the distributed coordination of networked multi-agent systems governed by unknown nonlinear dynamics. Our main contribution lies in the development of a distributed learning-based control algorithm that provably *guarantees* the accomplishment of a given multi-agent formation task without any a priori information on the underlying dynamics. The algorithm draws a novel connection between distributed learning with neural-network-based representations and adaptive feedback control, and consists of the following steps. Firstly, it trains a number of neural networks, one for each agent, to approximate controllers for the agents that accomplish the given formation task. The data used to train the neural networks consist of pairs of states and control actions of the agents that are gathered from runs of the multi-agent system. Secondly, it uses an online adaptive feedback control policy that guarantees accomplishment of the given formation task. Both steps can be executed in a distributed manner in a sense that each agent uses only local information, as modelled by a connectivity graph. Our approach builds on a combination of controllers trained offline and online adaptations, which was recently shown to significantly enhance performance with respect to single use of the offline part [18]. Numerical experiments show the robustness and adaptability of the proposed algorithm to different formation tasks, interactions among the agents, and system dynamics. That is, the proposed algorithm is able to achieve the given formation task even when the neural networks are trained with data that correspond to different multi-agent dynamic models (resembling a change in the dynamics of the agents), as well as different formation tasks and interactions among the agents. The novelty of the proposed method lies in the type of integration of neural networks and adaptive control. In particular, while the majority of the related works use neural networks to approximate the multi-agent dynamics, with the approximation error dictating the closed-loop convergence properties, we use neural networks to learn distributed controllers from readily available data. These data can correspond to systems with modified dynamics as well as formation tasks and the neural-network controllers aim to retain the boundedness of the closed-loop system. The online adaptive control design successfully compensates then for the bounded uncertainties of the multi-agent system. This paper extends our preliminary version [19] by providing (1) formal guarantees on the theoretical correctness of the proposed algorithm, and (2) a larger variety of experimental results.

B. Related Work

Robust and adaptive control: A large class of works on multi-agent coordination with uncertain dynamics falls in the category of robust and adaptive control [6], [10]–[14], [20]–[25]. Standard adaptive-control methodologies, however, assume certain linear parametric forms for the unknown terms of the dynamics, limiting the dynamic uncertainties to unknown *constant* terms [10], [11], [20], [21]. Additionally, many works that do not employ parametric assumptions consider dynamic uncertainties and disturbances that are uniformly bounded [13], [14] or satisfy growth conditions [6], [12], [22]. The works [23], [24] use functions in the control design that are larger than the upper bounds of the unknown dynamic terms; such a condition requires some a priori information on these terms. The work [25] assumes that the unknown drift terms of the dynamics are passive, which is then exploited in the stability analysis. Multi-agent coordination with unknown nonlinear continuous dynamics has been also tackled in the literature by using the so-called funnel control, without using dynamic approximations [5], [17], [26], [27]. Nevertheless, funnel controllers depend on so-called reciprocal time-varying barrier functions that drive the control input unbounded when the error approaches a pre-specified funnel, creating thus unnecessarily large control inputs that cannot be realized by the system's actuators. In this paper, we develop a distributed control algorithm that does not employ such reciprocal terms and whose correctness does not rely on any of the aforementioned assumptions.

Learning-based control: A large variety of works focus on distributed learning-based control to achieve multi-agent coordination under uncertain dynamics [15], [16], [28]–[32]. Such works resort to neural-network approximations of the unknown dynamic terms. In particular, they assume that the unknown functions of the dynamics are approximated arbitrarily well as a single-layer neural network with *known* radial-basis activation functions and a vector of unknown but *constant* weights. However, the accuracy of such approximations depends on the size of that vector, i.e., the number of neural-network neurons, implying that an arbitrarily small approximation error might require arbitrarily many weights. Additionally, there are no guidelines for choosing the activation functions in practice. Multi-agent coordination with unknown dynamics has also been tackled via cooperative reinforcement learning with stochastic processes [33]–[43]. However, such works usually adopt the conservative assumption that the agents have access to the states and actions of all other agents in the learning, execution, or both phases [36], [37]. Moreover, these works exhibit scalability problems with respect to the number of agents [35], or assume the availability of time or state discretizations of the underlying continuous-time and continuous-state models. Additionally, the related works on multi-agent cooperative reinforcement learning usually consider common or team-average reward functions for the agents [33], [39], which cannot be easily extended to account for inter-agent formation specifications that we account for. When relative inter-agent formation specifications are considered, the environment becomes non-stationary creating problems in the theoretical convergence analysis [33].

In this work, we develop a distributed neuro-adaptive control algorithm for the formation control of continuous-time and -

state multi-agent systems with unknown nonlinear dynamics. In contrast to the related works in the literature, we do not assume linear parametrizations [10], [11], neural-network approximations [15], [16], global boundedness or growth conditions [6], [12], [13], passivity properties [25], or known upper bounds [23], [24] for the unknown dynamic terms. According to the best of our knowledge, the distributed formation-control problem with unknown dynamics has not been solved in the absence of the aforementioned assumptions.

The rest of the paper is organized as follows. Section II describes the considered problem. We provide our theoretical results in Section III, and Section IV verifies the proposed methodology through experimental evaluation. Finally, Section V concludes the paper.

II. Problem Formulation

Consider a networked multi-agent group comprised of a leader, indexed by $i = 0$, and N followers, with $\mathcal{N} := \{1, \dots, N\}$. The leading agent acts as an exosystem that generates a desired command/reference trajectory for the multi-agent group. The followers, which have to be controlled, evolve according to the 2nd-order dynamics

$$\dot{x}_{i,1} = x_{i,2} \tag{1a}$$
$$\dot{x}_{i,2} = f_i(x_i, t) + g_i(x_i, t)u_i, \tag{1b}$$

where $x_{i,1} \in \mathbb{R}^n$, $x_{i,2} \in \mathbb{R}^n$, $x_i := [x_{i,1}^\top, x_{i,2}^\top]^\top \in \mathbb{R}^{2n}$ is the ith agent's state, assumed available for measurement by agent i, $f_i : \mathbb{R}^{2n} \times [0, \infty) \to \mathbb{R}^n$, $g_i : \mathbb{R}^{2n} \times [0, \infty) \to \mathbb{R}^{n \times n}$ are unknown functions modeling the agent's dynamics, and u_i is the ith agent's control input. The functions $f_i(x_i, t)$ and $g_i(x_i, t)$ are assumed to be locally Lipschitz in x_i over $\mathbb{R}^{2n}$ for each fixed $t \geq 0$, and uniformly bounded in t over $[0, \infty)$ for each fixed $x_i \in \mathbb{R}^{2n}$, for all $i \in \mathcal{N}$. In contrast to the works of the related literature, we do not assume any knowledge of the structure, Lipschitz constants, or bounds of $f_i(\cdot)$ and $g_i(\cdot)$, and we do not use any scheme to approximate them. The lack of such assumptions renders the multi-agent coordination problem significantly difficult, since there is no apparent way to counteract the effect of the unknown drift terms $f_i()$. Moreover, in contrast to the funnel-based schemes, we do not resort to the use of reciprocal-like terms to dominate $f_i()$. Nevertheless, we do require the following assumption on the control directions $g_i(\cdot)$:

Assumption 1. *The matrices $g_i(x_i, t)$ are positive definite, for all $x_i \in \Omega_i$, $t \geq 0$, where $\Omega_i \subset \mathbb{R}^{2n}$ are compact sets, $i \in \mathcal{N}$.*

Assumption 1 is a sufficiently controllability condition for (1) and is adopted in numerous related works (e.g., [5], [26], [29], [44]). The dynamics (1), subject to Assumption 1, comprise a large class of nonlinear dynamical systems that capture contemporary engineering problems in mechanical, electromechanical and power electronics applications, such as rigid/flexible robots, induction motors and DC-to-DC converters, to name a few. Systems not covered by (1) or Assumption 1 consist of underactuated or non-holonomic systems, such as unicycle robots, underactuated aerial or underwater vehicles. Such systems require special attention and their study consist part of our future work. Finally, the 2nd-order model (1) can be easily extended to account for higher-order integrator systems [45].

We use an undirected graph $\mathcal{G} := (\mathcal{N}, \mathcal{E})$ to model the communication among the agents, with $\mathcal{N}$ being the index set of the agents, and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ being the respective edge set, with $(i, i) \notin \mathcal{E}$ (i.e., simple graph). The adjacency matrix associated with the graph $\mathcal{G}$ is denoted by $\mathcal{A} := [a_{ij}] \in \mathbb{R}^{N \times N}$, with $a_{ij} \in \{0, 1\}$, $i, j \in \{1, \dots, N\}$. If $a_{ij} = 1$, then agent i obtains information regarding the state x_j of agent j (i.e., $(i, j) \in \mathcal{E}$), whereas if $a_{ij} = 0$ then there is no state-information flow from agent j to agent i (i.e., $(i, j) \notin \mathcal{E}$). Furthermore, the set of neighbors of agent i is denoted by $\mathcal{N}_i := \{j \in \mathcal{N} : (i, j) \in \mathcal{E}\}$, and the degree matrix is defined as $\mathcal{D} := \text{diag}\{|\mathcal{N}_1|, \dots, |\mathcal{N}_N|\}$. Since the graph is undirected, the adjacency is a mutual relation, i.e., $a_{ij} = a_{ji}$, rendering $\mathcal{A}$ symmetric. The *Laplacian* matrix of the graph is defined as $\mathcal{L} := \mathcal{D} - \mathcal{A}$ and is also symmetric. The graph is *connected* if there exists a path between any two agents. For a connected graph, it holds that $\mathcal{L}\bar{1} = 0$, where $\bar{1}$ is the vector of ones of appropriate dimension.

Regarding the leader agent, we denote its state variables by $x_0 := [x_{0,1}^\top, x_{0,2}^\top]^\top \in \mathbb{R}^{2n}$, and consider the 2nd-order dynamics

$$\dot{x}_{0,1} = x_{0,2}$$
$$\dot{x}_{0,2} = u_0,$$

where $u_0 : [0, \infty) \to \mathbb{R}^n$ is a bounded command signal. However, the leader provides its state only to a subgroup of the N agents. In particular, we model the access of the follower agents to the leader's state via a diagonal matrix $\mathcal{B} := \text{diag}\{b_1, \dots, b_N\} \in \mathbb{R}^{N \times N}$; if $b_i = 1$, then the ith agent has access to the leader's state, whereas it does not if $b_i = 0$, for $i \in \mathcal{N}$. Thus, we also define the augmented graph as $\bar{\mathcal{G}} := (\mathcal{N} \cup \{0\}, \bar{\mathcal{E}})$, where $\bar{\mathcal{E}} := \mathcal{E} \cup \{(0, i) : b_i = 1\}$. We further define

$$H := (\mathcal{L} + \mathcal{B}) \otimes I_n,$$

where $\otimes$ denotes the Kronecker product, as well as the stacked vector terms

$$\bar{x}_1 := [x_{1,1}^\top, \dots, x_{N,1}^\top]^\top \in \mathbb{R}^{Nn}$$
$$\bar{x}_2 := [x_{1,2}^\top, \dots, x_{N,2}^\top]^\top \in \mathbb{R}^{Nn}$$
$$\bar{x} := [x_1^\top, \dots, x_N^\top]^\top \in \mathbb{R}^{2Nn}$$
$$\bar{x}_{0,1} := [x_{0,1}^\top, \dots, x_{0,1}^\top]^\top \in \mathbb{R}^{Nn}$$
$$\bar{x}_{0,2} := [x_{0,2}^\top, \dots, x_{0,2}^\top]^\top \in \mathbb{R}^{Nn}$$
$$\bar{x}_0 := [\bar{x}_{0,1}^\top, \bar{x}_{0,2}^\top]^\top \in \mathbb{R}^{2Nn}.$$

By further defining

$$f(\bar{x}, t) := [f_1(x_1, t)^\top, \dots, f_N(x_N, t)^\top]^\top \in \mathbb{R}^{Nn}$$
$$g(\bar{x}, t) := \text{diag}\{g_1(x_1, t), \dots, g_N(x_N, t)\} \in \mathbb{R}^{Nn \times Nn}$$
$$u := [u_1^\top, \dots, u_N^\top]^\top \in \mathbb{R}^{Nn},$$

the dynamics (1) can be written as

$$\dot{\bar{x}}_1 = \bar{x}_2 \tag{2a}$$
$$\dot{\bar{x}}_2 = f(\bar{x}, t) + g(\bar{x}, t)u. \tag{2b}$$

This article has been accepted for publication in IEEE Transactions on Automation Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TASE.2025.3528501

The goal of this work is to design a distributed control algorithm, where each agent has access only to its neighbors' information, to achieve a pre-specified geometric formation of the agents in $\mathbb{R}^n$. More specifically, consider for each agent $i \in \mathcal{N}$ the constant[1] vectors c_{ij}, $j \in \{0\} \cup \mathcal{N}_i$ prescribing a desired offset that agent i desires to achieve with respect to the leader ($j = 0$), and its neighbors ($j \in \mathcal{N}_i$)[2] . That is, each agent $i \in \mathcal{N}_i$ aims at achieving $x_{i,1} = x_{j,1} - c_{ij}$, for all $j \in \mathcal{N}_i$, and if $b_i = 1$ (i.e., the agent obtains information from the leader), $x_{i,1} = x_{0,1} - c_{i0}$. Note that, in the case of undirected graph, $c_{ij} = -c_{ji}$, for all $(i, j) \in \mathcal{E}$, and we assume that the set

$$\{\bar{x}_1 \in \mathbb{R}^{Nn} : x_{i,1} - x_{j,1} + c_{ij} = 0, \forall (i,j) \in \mathcal{E}, \\ b_i(x_{i,1} - x_{0,1} + c_{i0}) = 0, \forall i \in \mathcal{N}\}$$

is non-empty in order for the formation specification to be feasible.

Furthermore, we impose the following assumption on the graph connectivity:

Assumption 2. *The graph $\mathcal{G}$ is connected and there exists at least one $i \in \mathcal{N}$ such that $b_i = 1$.*

The aforementioned assumption dictates that $\mathcal{L} + \mathcal{B}$ is an irreducibly diagonally dominant M-matrix [46]. An M-matrix is a square matrix having its off-diagonal entries non-positive and all principal minors nonnegative, thus $\mathcal{L} + \mathcal{B}$ is positive definite.

We define now the error variables for each agent as

$$e_{i,1} \coloneqq \sum_{j \in \mathcal{N}_i} (x_{i,1} - x_{j,1} + c_{ij}) + b_i(x_{i,1} - x_{0,1} + c_{i0}), \quad (3)$$

for $i \in \mathcal{N}$, and the respective stack vector

$$e_1 \coloneqq [e_{1,1}^\top, \dots, e_{N,1}^\top]^\top.$$

Next, by employing the multi-agent graph properties, noticing that $(\mathcal{L} \otimes I_n)\bar{x}_{0,1} = 0$, and since $(\mathcal{L} + \mathcal{B})$ is positive definite and hence invertible, (3) can be written as

$$e_1 \coloneqq H(\bar{x}_1 - \bar{x}_{0,1} + c), \quad (4)$$

with $H = (\mathcal{L} + \mathcal{B}) \otimes I_n$ and

$$c \coloneqq \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix} \coloneqq H^{-1} \begin{bmatrix} \sum_{j \in \mathcal{N}_1} c_{1j} + b_1 c_{10} \\ \vdots \\ \sum_{j \in \mathcal{N}_N} c_{Nj} + b_N c_{N0} \end{bmatrix} \quad (5)$$

stacks the relative desired offsets c_i of the ith agent with respect to the leader, as dictated by the desired formation specification. In this way, the desired formation is expressed with respect to the leader state, and is thus achieved when the state $x_{i,1}$ of each agent approaches the leader state $x_{0,1}$ with the corresponding offset c_i, $i \in \mathcal{N}$. Therefore, the formation control problem is solved if the control algorithm drives the disagreement vector

$$\delta_1 \coloneqq \begin{bmatrix} \delta_{1,1} \\ \vdots \\ \delta_{N,1} \end{bmatrix} \coloneqq \bar{x}_1 - \bar{x}_{0,1} + c \quad (6)$$

to zero. However, the disagreement formation variables $\delta_{i,1}$, are global quantities and thus cannot be measured distributively by each agent based on the local measurements, as they involve information directly from the leader as well as from the whole graph topology via employing the inverse of $\mathcal{L} + \mathcal{B}$ in (5). Nevertheless, from (4), since $\mathcal{L} + \mathcal{B}$ is positive definite and hence invertible, one obtains

$$\|\delta_1\| \leq \frac{\|e_1\|}{\sigma_{\min}(H)}, \quad (7)$$

where $\sigma_{\min}(\cdot)$ denotes the minimum singular value. Therefore, convergence of e_1 to zero, which we aim to guarantee, implies convergence of δ_1 to zero. We further define the augmented errors for each agent

$$e_{i,2} \coloneqq \dot{e}_{i,1} + k_1 e_{i,1}, \quad (8)$$

where k_1 is a positive constant, the respective stacked vector

$$e_2 \coloneqq [e_{i,2}^\top, \dots, e_{N,2}^\top]^\top \in \mathbb{R}^{nN},$$

and the total error vector $e \coloneqq [e_1^\top, e_2^\top]^\top$. By using (4), the total error dynamics can be written as

$$\dot{e}_1 = -k_1 e_1 + e_2 \quad (9a)$$

$$\dot{e}_2 = H(f(\bar{x}(e), t) + g(\bar{x}(e), t)u - \ddot{\bar{x}}_{0,1}) - k_1^2 e_1 + k_1 e_2, \quad (9b)$$

where, with a slight abuse of notation, we express $\bar{x}$ as a function of e through (4).

Before proceeding, we define the tuple

$$\mathcal{F} \coloneqq (x_0(t), f, g, c, \bar{\mathcal{G}}, \bar{x}(0)) \quad (10)$$

as the "formation instance", characterized by the leader profile, the agent dynamics, the desired formation offsets, the graph topology, and the initial conditions of the agents.

III. Main Results

This section describes the proposed algorithm, which consists of two steps. The first step consists of offline learning of distributed controllers, represented as neural networks, using training data derived from offline runs. Note that these data might be derived from systems with modifications in the dynamics and operating conditions (e.g., a robotic manipulator that has undergone a change of motor or end-effector or the addition of a link) or even different formation tasks as dictated by c_{ij} - the increasing deployment of autonomous systems guarantees the availability of such data. In the second step, we design an adaptive feedback control policy that uses the neural networks and provably guarantees achievement of the formation specification. Unlike the majority of the related works, we do not assume global boundedness/Lipschitz or growth conditions for the drift terms $f_i(\cdot)$ and do not use any parametric schemes to approximate them. Additionally, we do

[1]Time-varying $c_{ij}(t)$ can be also considered with a minor adjustment of Assumption 3 in Sec. III.

[2]The formation specification can be also expressed via formation offsets with respect to the leader c_i, $i \in \mathcal{N}$. The error (3) becomes then $e_{i,1} = \sum_{j \in \mathcal{N}_i}(x_{i,1} - c_i - x_{j,1} + c_j) + b_i(x_{i,1} - x_{0,1} - c_i)$, $i \in \mathcal{N}$.

not resort to the use of reciprocal barrier functions that yield excessively large control inputs. The proposed algorithm is a novel integration of offline-trained controllers, represented as neural networks, and online adaptive control laws.

A. Neural-network learning

As discussed in Section I, we are inspired by cases where systems undergo changes that modify their dynamics and hence the underlying controllers no longer guarantee the satisfaction of a specific task. In such cases, instead of carrying out the challenging and tedious procedure of identification of the new dynamic models and design of new model-based controllers, we aim to exploit data from offline system trajectories and develop a distributed online policy that is able to adapt to the aforementioned changes and achieve the formation task expressed via the offsets c_{ij}, $(i,j) \in \bar{\mathcal{E}}$. Consequently, we assume the existence of data gathered from a finite set of T trajectories $\mathcal{J}$ generated by a priori runs of the multi-agent system. More specifically, we consider that $\mathcal{J}$ is decomposed as $\mathcal{J} = (\mathcal{J}_1, \ldots, \mathcal{J}_N)$, where $\mathcal{J}_i$ is the set of trajectories of agent $i \in \mathcal{N}$. Since the proposed control scheme is distributed, we consider that each agent i has access to the data from its own set of trajectories $\mathcal{J}_i$, which comprises the finite set

$$\mathcal{J}_i = \left\{ x_i^k(t), \{x^j\}_{j\in\mathcal{N}_i^k}, u_i^k\left(x_i^k(t), \{x^j\}_{j\in\mathcal{N}_i^k}, t\right) \right\}_{t\in\mathbb{T}_i}, \tag{11}$$

where $\mathbb{T}_i$ is a finite set of time instants, $x_i^k \in \mathbb{R}^{2n}$ is the state trajectory of agent i for trajectory k, $\mathcal{N}_i^k$ are the neighbors of agent i in trajectory k, with $\{x^j\}_{j\in\mathcal{N}_i^k}$ being their respective state trajectories (which agent i has access to, being their neighbor), and $u_i^k(x_i^k(t), \{x^j\}_{j\in\mathcal{N}_i^k}, t) \in \mathbb{R}^n$ is the control input trajectory of agent i, which is a function of time and of its own and its neighbors' states.

Each agent $i \in \mathcal{N}$ uses the data to train a neural network in order to approximate a controller that accomplishes the formation task. More specifically, each agent uses the tuples $\{x_i^k(t), \{x^j\}_{j\in\mathcal{N}_i^k}\}_{t\in\mathbb{T}_i}$ as input to a neural network, and $u_i^k\left(x_i^k(t), \{x^j\}_{j\in\mathcal{N}_i^k}, t\right)_{t\in\mathbb{T}_i}$ as the respective output targets, for all T trajectories. For the inputs corresponding to agents that are not neighbours of agent i in a trajectory k, we disable the respective neurons. The training is performed using the standard back-propagation procedure in order to minimize a distance metric among the inputs $\{x_i^k(t), \{x^j\}_{j\in\mathcal{N}_i^k}\}_{t\in\mathbb{T}_i}$ and output targets $u_i^k\left(x_i^k(t), \{x^j\}_{j\in\mathcal{N}_i^k}, t\right)_{t\in\mathbb{T}_i}$ [47] - usually the Mean-Square-Error (MSE) metric $E_{MSE}(\mathsf{x},\mathsf{y}) = \frac{1}{K}\sum_{k\in\{1,\ldots,K\}}(\mathsf{x}_k - \mathsf{y}_k)^2$ for vectors $\mathsf{x} = [\mathsf{x}_1, \ldots, \mathsf{x}_K]^\top$, $\mathsf{y} = [\mathsf{y}_1, \ldots, \mathsf{y}_K]^\top$ and $K \geq 2$. For a given $x \in \mathbb{R}^{2Nn}$, we denote by $u_{i,nn}(\bar{x})$ the output of the neural network of agent $i \in \mathcal{N}$, and $u_{nn}(\bar{x}) := [u_{1,nn}(\bar{x})^\top, \ldots, u_{N,nn}(\bar{x})^\top]^\top$. Being the outputs of neural networks, $u_{i,nn}(\bar{x})$ constitute composite functions of input, hidden, and output layers; these layers are usually linear mappings $\mathsf{f}_m(\star) = A\star + B$, where A and B are the weights of the neural network, and activation functions of the form $\mathsf{f}_a(\star) = (1+\exp(-\star))^{-1}$, or $\max(0,\star)$. For example, in the experiments of Section IV, we use neural networks of 4 fully connected linear layers, followed by batch normalization, and a ReLu activation function $\max(0,\star)$. More specifically, the outputs $u_{i,nn}(\bar{x})$ are given by $u_{i,nn}(\bar{x}) = y_{i,4}(\bar{x})$, where $y_{i,4}(\bar{x})$ is defined recursively as:

$$\begin{aligned} \nu_{i,k} &= A_{i,k} y_{i,k} + b_{i,k} \\ q_{i,k} &= \frac{\nu_{i,k} - \mathbb{E}[\nu_{i,k}]}{\sqrt{\mathrm{Var}[\nu_{i,k}] + \epsilon}} \\ y_{i,k} &= \max(0, q_{i,k}), \end{aligned}$$

for $i \in \mathcal{N}$, $k \in \{1, \ldots, 4\}$, where $A_{i,k} \in \mathbb{R}^{2Nn\times 2Nn}$ and $b_{i,k} \in \mathbb{R}^{2Nn}$ are the weight matrix and bias vector, respectively, for layer k, $\epsilon = 10^{-5}$, $y_{i,0} = \bar{x}$, $\mathbb{E}[\cdot]$ is the expected-value operator, $\mathrm{Var}[\cdot]$ is the variance operator, and $\max(0,*)$ is applied element-wise.

We stress that we do not require the training trajectories $\mathcal{J}$ to correspond to the formation instance $\mathcal{F}$ specified in (10). As verified in the numerical experiments of Section IV, each trajectory k might be derived from the execution of a formation instance $\mathcal{F}_k = (x_0^k, f^k, g^k, c^k, \bar{\mathcal{G}}^k, \bar{x}^k(0))$ that is different than the one specified in (10), i.e., different leader profile x_0^k, different parameters in the agent dynamics f^k, g^k, different formation offsets c^k or communication graph $\bar{\mathcal{G}}^k$, and different initial agent conditions $\bar{x}^k(0)$, for all $k \in \mathbb{K}$ and some index set $\mathbb{K} \subset \mathbb{N}$. We further note that, in practical applications, one can obtain state x_i and control u_i measurements by using onboard sensors. Therefore, the state-control data $\mathcal{J}_i$ in (11) are more accessible than data that correspond to the unknown functions $f_i(x_i,t)$ and $g_i(x_i,t)$. For instance, consider a multi-robot system equipped with fine-tuned stabilization controllers for a number of formation tasks. By using onboard sensors, one can then measure the robot states x_i^k (e.g., position and velocity) and control inputs u_i^k at the actuators at sampled time instants of the robot trajectories, without having access to the expression of the controllers themselves or to the state derivatives $\dot{x}_i$. Therefore, one does not have direct access to samples of the unknown functions $f_i(x_i,t)$ and $g_i(x_i,t)$.

Since the training trajectories are produced by the instances $\mathcal{F}_k$, which are different from $\mathcal{F}$, we do not expect the neural networks to learn how to achieve the formation task at hand, but rather to be able to adapt to the entire collection of tasks. The motivation for training the neural networks with different tasks and dynamics is the following. Since the tasks correspond to bounded trajectories, the respective stabilizing controllers compensate successfully for the dynamics in (1). Therefore, the neural networks aim to approximate distributed controllers the retain this property, i.e., the boundedness of the multi-agent dynamics (1). Loosely speaking, the neural networks aim to inherit a property exhibited by all stabilizing controllers of the offline trajectories that consist the training data. More technically, the neural networks aim to approximate stabilizing distributed controllers that achieve the formation task, having though access only to the offline-generated data (u_i^k, x_i^k, t^k). These data might correspond to different formation tasks, but exhibit the property of successfully compensating for the unknown dynamics, even if these differ in some parameters. By using such approximation, the online feedback-control policy, which is illustrated in the next section, is able to guarantee achievement of the formation task at hand, without

using any explicit information on the dynamics. We explicitly model the aforementioned approximation via the following assumption on the closed-loop system trajectory that is driven the by neural networks' output.

Assumption 3. *There exists $r > 0$ such that the stacked vector of outputs $u_{nn}(x)$ of the trained neural networks satisfies*

$$e_2^\top H(f(\bar{x}(e),t) + g(\bar{x}(e),t)u_{nn}(\bar{x}) - \ddot{\bar{x}}_{0,1}) \leq \kappa \|e_2\|^2 \quad (12)$$

for all e satisfying $\|e\| \leq r$, where κ is a positive constant.

Assumption 3 is a sufficient condition for the prevention of finite-time escape of the the error trajectory $e(t)$ when the agents apply only the neural-network controllers, i.e., of the solution of the differential equation $\ddot{e}_2 = H(f(\bar{x},t) + g(\bar{x},t)u_{nn}(\bar{x}) - \ddot{\bar{x}}_{0,1}) - k_1^2 e_1 + k_1 e_2$. Indeed, when the multi-agent system is driven solely by the neural-network controllers and satisfies (12), one can find a Lyapunov function $V(e) = e^\top G e$, for a suitable constant matrix $G \in \mathbb{R}^{2nN \times 2nN}$, satisfying[3] $\lambda_{\min}(G)\|e\|^2 \leq V(e) \leq \lambda_{\max}(G)\|e\|^2$ and $\dot{V} \leq \alpha \|e\|^2 \leq \frac{\alpha}{\lambda_{\min}\{G\}} V$, for all $\|e\| \leq r$ and a positive constant α. Therefore, we conclude that $\lambda_{\min}(G)\|e(t)\|^2 \leq V(e(t)) \leq V(e(0)) \exp\left(\frac{\alpha}{\lambda_{\min}\{G\}} t\right)$, which prevents any finite-time escape of $e(t)$. Further note that the constants r and κ in (12) are *unknown*. Finally, in the case of time-varying formation vectors $c_{ij}(t)$, (12) is modified to $e_2^\top H(f(\bar{x}(e),t) + g(\bar{x}(e),t)u_{nn}(\bar{x}) - \ddot{\bar{x}}_{0,1} + \ddot{c}(t)) \leq \kappa \|e_2\|^2$, where c is defined in (5), with the rest of the analysis remaining the same.

Assumption 3 is motivated by (i) the property of neural networks to approximate a continuous function arbitrarily well in a compact domain for a large enough number of neurons and layers [48], and (ii) the fact that the neural networks are trained with bounded trajectories. As mentioned before, the collection of tasks that the neural networks are trained with correspond to bounded trajectories. Hence, in view of the similarity of the dynamic terms, the neural networks are expected to approximate a control policy that maintains the boundedness of the state trajectories as per (12). Contrary to the related works (e.g., [4], [44], [49]–[52]), however, we do not adopt approximation schemes for the system dynamics. In fact, a standard assumption in the related literature is the approximation of an unknown function by a single-layer neural network as $\Theta(x)\vartheta + \epsilon(x)$, where $\Theta(x)$ is a *known* matrix of radial basis function, ϑ is a vector of unknown weights, and $\epsilon(x)$ is an approximation error. However, the performance of the respective controllers is inversely proportional to the size of $\epsilon(x)$, which shrinks with the number of neurons and is bounded in any contact set. Hence, promising performance is obtained in the expense of arbitrarily large vector ϑ, which can potentially lead to numerical issues in practise. In our case, Assumption 3 is merely a growth condition on the the solution of the system driven by $u_{nn}(\bar{x})$. In practice, (12) can be achieved by rich exploration of the state space by the leader agent x_0^k in the training data $\mathcal{F}^k$. In the numerical experiments of Section IV, we show that (12) holds true along the executed trajectories of the multi-agent system.

[3] $\lambda_{\min}$ and $\lambda_{\max}$ denote the minimum and maximum eigenvalues, respectively.

We note that the neural-network controllers u_{nn} can be replaced by other learning methodologies, as long as Assumption 3 holds. Nevertheless, the rich structure of neural networks makes them great candidates for approximating a control policy that satisfies (12).

B. Distributed Control Policy

We now design a distributed, adaptive feedback control policy to accomplish the formation task dictated by the graph topology $\bar{\mathcal{G}}$, the leader profile $x_0(t)$, and offsets c_{ij}, $(i,j) \in \bar{\mathcal{E}}$, given in Section II.

We define the adaptation variables $\hat{d}_{i,1}$ for each agent $i \in \mathcal{N}$, with $\hat{d}_1 \coloneqq [\hat{d}_{1,1}, \ldots, \hat{d}_{N,1}]^\top \in \mathbb{R}^N$, and design the distributed control policy as

$$u_i = u_{i,nn}(\bar{x}) - (k_2 + \hat{d}_{i,1}) e_{i,2} \quad (13a)$$

where k_2 is a positive constant. We further design the updates of the adaptation variables $\hat{d}_{i,1}$ as

$$\dot{\hat{d}}_{i,1} \coloneqq \mu_{i,1} \|e_{i,2}\|^2 \quad (13b)$$

with $\hat{d}_{i,1}(0) > 0$ and $\mu_{i,1}$ are positive constants, for all $i \in \mathcal{N}$. The overall control algorithm is depicted in Fig. 1 for an agent $i \in \mathcal{N}$.

Remark 1 (Control design philosophy)**.** *The control design is inspired by adaptive control methodologies [53], where the time-varying coefficients $\hat{d}_{i,1}$ adapt, in coordination with the neural-network controllers, to the unknown dynamics in order to ensure closed-loop stability. In particular, by inspecting the proof of Theorem 1, it can be concluded that $\hat{d}_{i,1}$ aims to counteract the term $d_1 \coloneqq \frac{k_1 \|H^{-1}\|}{\underline{g}} + \frac{\kappa}{\underline{g}}$, where $\underline{g} \coloneqq \min_{i \in \{1,\ldots,N\}} \{\lambda_{\min}(g_i)\}$, $i \in \mathcal{N}$. Intuitively, $\hat{d}_{i,1}$ increases according to (13b) until it dominates the aforementioned term, leading to convergence of $e_{i,2}$ to zero, for all $i \in \mathcal{N}$.*

Note further that agent i's control policy (13) does not use any information on its own or its neighbors' dynamic terms $f_i(\cdot)$, $g_i(\cdot)$, or the constants r, κ of (12). Additionally, note that each agent uses only relative feedback from its neighbors, as can be verified by (3), (8) and (13).

Remark 2 (Control algorithm novelty)**.** *The proposed algorithm comprises an innovative integration of controllers represented as neural networks and adaptive feedback control. In contrast to the majority of related works, which employ neural networks to approximate the dynamics of the agents by assuming known radial-basis activation functions, large amount of weights, and small approximation errors, the proposed algorithm exploits the potential availability of offline-obtained data to train neural network controllers that simply maintain the boundedness of the closed-loop system (as per Assumption 3). The respective approximation errors are then successfully compensated by real-time adaptive control laws.*

Remark 3 (Control parameters)**.** *The proposed method can be easily applied in practice since it consists of a simple adaptive feedback control for each agent (see eq. (13)) and the well-known procedure of neural-network training. The majority of parameters involved in the control method concern the*

This article has been accepted for publication in IEEE Transactions on Automation Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TASE.2025.3528501

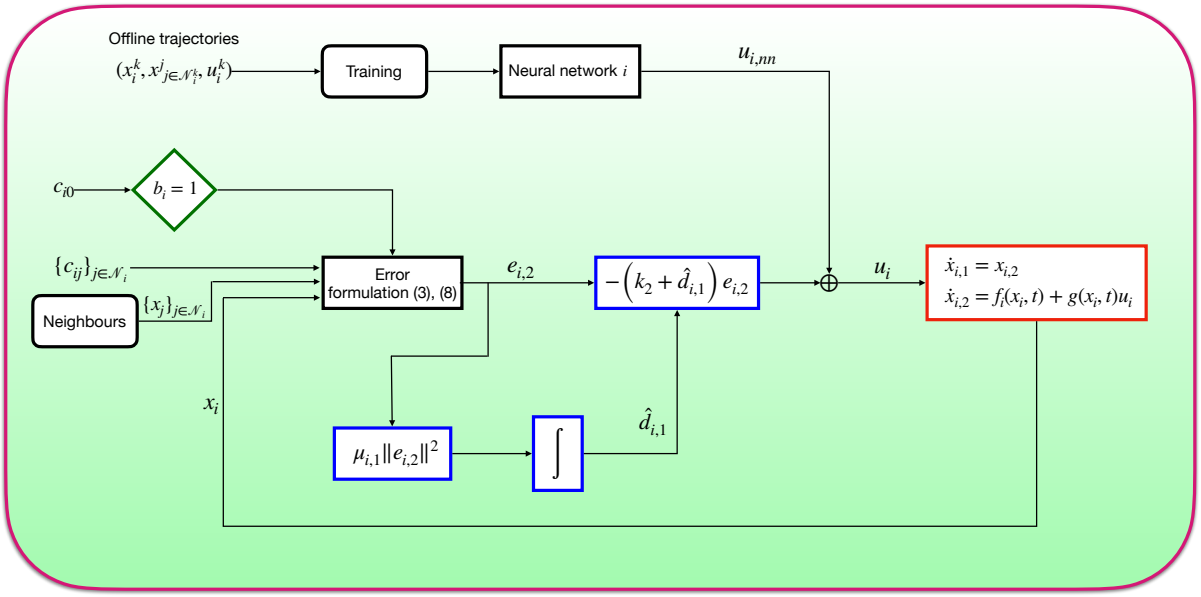

Fig. 1. Flow diagram for the proposed algorithm (shown for agent $i \in \mathcal{N}$.

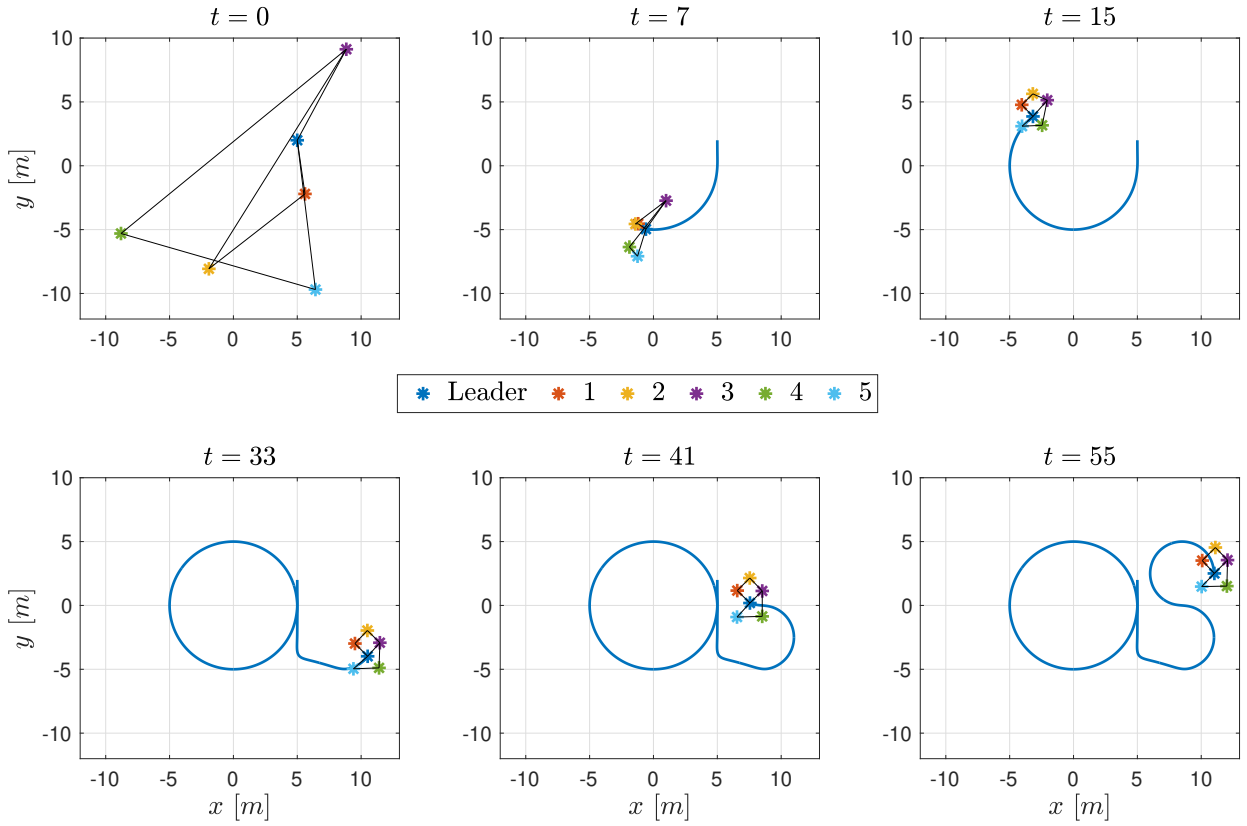

Fig. 2. Snapshots of Case 1's experiment in the x-y plane. The agents converge to the desired formation (see bottom-middle and bottom-right plots) around the leader, which follows a pre-specified trajectory (continuous blue line). The black lines represent the communication edge set $\bar{\mathcal{E}}$ of the agents.

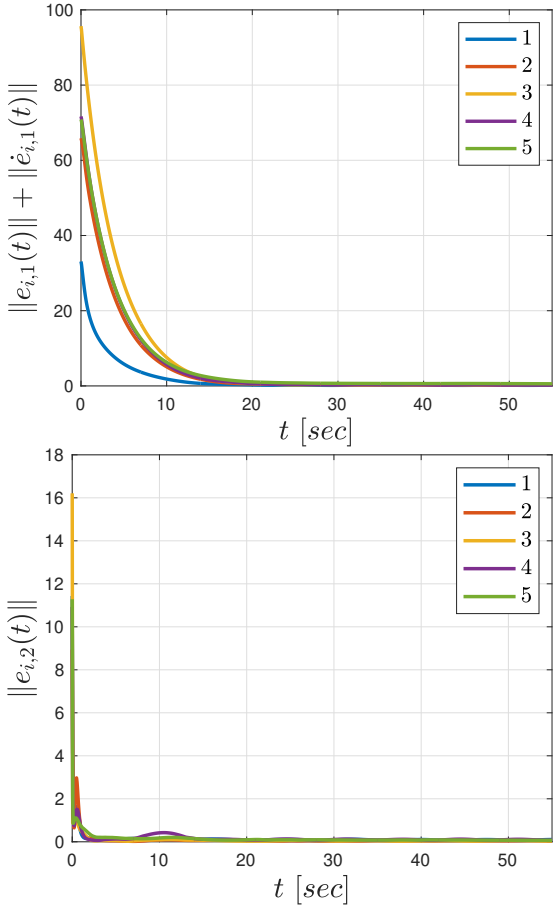

Fig. 3. Evolution of the error signals $\|e_{i,1}(t)\| + \|\dot{e}_{i,1}(t)\|$, and $\|e_{i,2}(t)\|$, for $i \in \{1, \ldots, 5\}$, and $t \in [0, 55]$, in the first experiment.

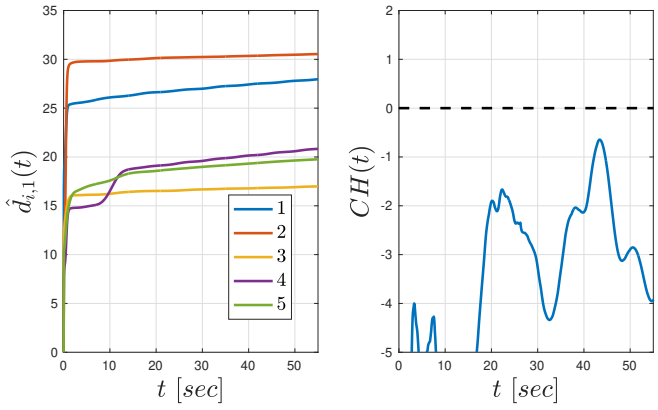

Fig. 4. Left: The evolution of the adaptation signals $\hat{d}_{i,1}(t)$ for $i \in \{1, \ldots, 5\}$, in Case 1. Right: The evolution of $CH(t)$ in Case 1.

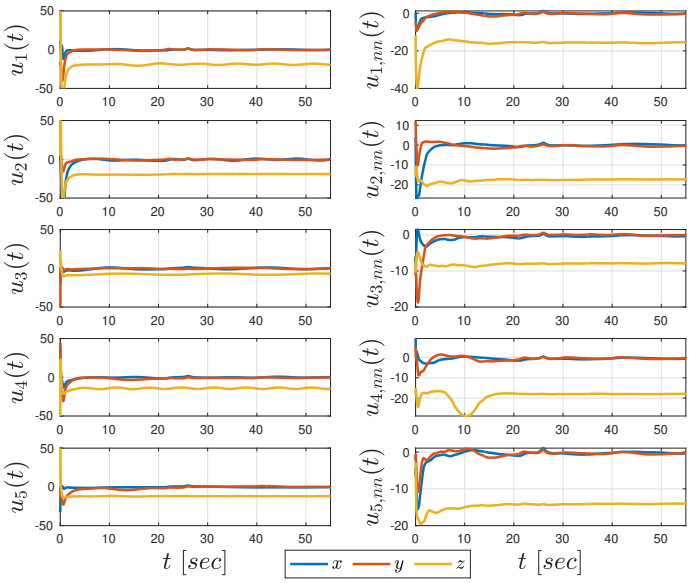

Fig. 5. The evolution of the control inputs $u_i(t)$ and the neural-network controllers $u_{i,nn}(t)$, for $i \in \{1, \ldots, 5\}$, in Case 1.

neural-network training since the online feedback controller (eq. (13) involves only k_2 and $\mu_{i,1}$ for each agent. The stability of the closed-loop system requires k_2 and $\mu_{i,1}$ to be positive; however, their selection affects the characteristics of the closed-loop trajectory, such as overshoot and convergence time, and the control-input characteristics. For example, large values for k_2 and $\mu_{i,1}$ will likely yield large overshoot and control-input magnitude but small error-convergence times.

The training procedure of neural networks naturally involves numerous parameters to adjust, such as input batch size, number of layers and neurons, number of training iterations, and learning rate. Since, however, the training is performed offline, these parameters can be adjusted until satisfactory training performance is obtained. Moreover, related works often provide guidelines regarding the choice of these parameters, (e.g., [54]).

Remark 4 (Computational aspects)**.** *The computational aspects of the proposed method concern primarily the part of offline neural-network training, since the feedback control in (13) comprises a closed-form expression and can be very efficiently run in real time. The complexity of the neural-network training depends on multiple factors, such as the architecture of the networks, the input data, the number of neurons, the optimization algorithm used and the number of training iterations. For stochastic gradient descent algorithms, which we use in the simulation studies of Sec. IV, the complexity is linear in the amount of data and number of iterations. Nevertheless, it should be noted that the training is performed offline, i.e., prior to execution, and hence it does not affect the scalability of the method. In the simulation studies of Sec. IV, the total time taken for training a single neural network did not exceed 5 minutes.*

The following theorem, whose proof is given in the appendix, guarantees the accomplishment of the formation task.

Theorem 1. *Let a multi-agent system evolve subject to the dynamics (1) under an undirected communication graph $\bar{\mathcal{G}}$. Under Assumptions 1-3, there exists a set $\bar{\Omega}_{\hat{x}} \subset \mathbb{R}^{N(2n+1)}$ such that, if $\big(e(0), \hat{d}_1(0)\big) \in \bar{\Omega}_{\hat{x}}$, the distributed control mechanism guarantees $\lim_{t\to\infty}(e_{i,1}, e_{i,2}) = 0$, for all $i \in \mathcal{N}$, as well as the boundedness of all closed-loop signals.*

Contrary to the works in the related literature (e.g., [5], [27]) we do not impose reciprocal terms in the control input that grow unbounded in order to guarantee closed-loop stability. The resulting controller is essentially a simple linear feedback on e_1, e_2 with time-varying adaptive control gains, accompanied by the neural network output that ensures the boundedness condition (12).

IV. Numerical Experiments

In this section, we carry out 4 case studies of numerical experiments to illustrate the effectiveness of the proposed algorithm. Comparative studies (3rd case) show that closed-loop stability requires both the neural networks and the real-time adaptive controllers under the (loose) assumptions on the dynamics. We further show the scalability and effectiveness of the proposed algorithm to large number of agents and its robustness with respect to time-varying graphs (4th case).

We consider $N = 5$ follower aerial vehicles in $\mathbb{R}^3$ with dynamics of the form (1), with

$$f_i(x_i, t) = \frac{1}{m_i}(\bar{g}_r + w_{i,1}(t) + w_{i,2}(x_i))$$
$$g_i(x_i, t) = \frac{\|x_i\| + 0.5\sin(0.1t) + 1}{m_i}$$

where $\bar{g}_r = [0, 0, 9.81]^\top$ is the gravity vector and $m_i \in \mathbb{R}$ is the mass of agent $i \in \mathcal{N}$. Furthermore, $w_{i,1}(t)$, $w_{i,2}(x_i)$ are chosen as

$$w_{i,1}(t) = \begin{bmatrix} A_{i,1}\sin(\eta_{i,1}t + \phi_{i,1}) \\ A_{i,2}\sin(\eta_{i,2}t + \phi_{i,2}) \\ A_{i,3}\sin(\eta_{i,3}t + \phi_{i,3}) \end{bmatrix}$$
$$w_{i,2}(x_i) = F_i y_i$$

with $y_i = [x_{i,2_1}^2, x_{i,2_2}^2, x_{i,2_3}^2, x_{i,2_1}x_{i,2_2}, x_{i,2_1}x_{i,2_3}, x_{i,2_2}, x_{i,2_3}]$, and we further use the notation $x_{i,2} = [x_{i,2_1}, x_{i,2_2}, x_{i,2_3}]^\top$ for all $i \in \mathcal{N}$. The terms m_i, $A_{i,\ell}$, $\eta_{i,\ell}$, $\phi_{i,\ell}$ are constants that take values in $(0, 1)$; similarly, $F_i \in \mathbb{R}^{3\times 6}$ is a constant matrix whose elements take values in $(0, 1)$. We evaluate the proposed algorithm in four test cases. In all of these cases, we choose the control gains of (13) as $k_1 = 0.1$, $k_2 = \mu_{i,1} = 0.5$. The form of the neural network and the learning procedure used can be found at the end of this section.

Case 1:The first case consists of the stabilization of the followers around the leader, which is assigned with the tracking of a reference time-varying trajectory profile $x_0(t)$. We consider a communication graph modeled by the edge set $\bar{\mathcal{E}}$ = { (1, 2), (2, 3), (3, 4), (4, 5), (1, 0), (3, 0), (5, 0) }, i.e., agents 1, 3, and 5 have access to the information of the leader. The stabilization is dictated by the formation constants $c_{1,2} = -c_{2,1} = [1, 1, 0]^\top$, $c_{2,3} = -c_{3,2} = [1, -1, 0]^\top$, $c_{3,4} = -c_{4,3} = [0, -2, 0]^\top$, $c_{4,5} = -c_{5,4} = [-2, 0, 0]^\top$, $c_{1,0} = [1, -1, 0]^\top$, $c_{3,0} = [-1, -1, 0]^\top$, $c_{5,0} = [1, 1, 0]^\top$. The aforementioned parameters, along with the agents' initial conditions, specify the first task's formation instance $\mathcal{F} := (x_0, f, g, c, \bar{\mathcal{G}}, \bar{x}(0))$. We generate data from 100 trajectories that correspond to different f, g, $\bar{x}(0)$ than in $\mathcal{F}$, but with the same leader profile x_0 and inter-agent formation offsets c and communication graph $\bar{\mathcal{G}}$. The differences in f and g are created by assigning random values, in $(0, 1)$, to the constants m_i, $A_{i,\ell}$, $\eta_{i,\ell}$, $\phi_{i,\ell}$, and F_i, for all $i \in \mathcal{N}$. We further assign the initial conditions for each agent as $x_{i,1}(0) = x_{0,1}(0) + \text{rand}(-4, 4)[1, 1, 1]^\top$, and $x_{i,2}(0) = \text{rand}(-2, 2)[1, 1, 1]^\top$, $i \in \mathcal{N}$; we set the leader agent's initial condition as $x_{0,1}(0) = [5, 2, 10]^\top$, $x_{0,2}(0) = [0.0039, -09836, 0]^\top$ for all trajectories. We use the generated data to train 5 neural networks, one for each agent. We test the control policy (13) using the task's formation instance $\mathcal{F}$. The results are depicted in Figs. 2-5; Fig. 2 depicts snapshots of the multi-agent formation in the x-y plane and Fig. 3 shows the evolution of the error signals $\|e_{i,1}(t)\| + \|\dot{e}_{i,1}(t)\|$ and $\|e_{i,2}(t)\|$ for $i \in \{1, \ldots, 5\}$. Fig. 4 shows the evolution of the adaptation variables $\hat{d}_{i,1}(t)$, $i \in \mathcal{N}$, and the signal $CH(t) = e_2(t)^\top H(f(\bar{x}(t), t) + g(\bar{x}(t), t)u(t) - \ddot{\bar{x}}_{0,1}(t)) - 100\|e_2\|$, which is always negative, verifying thus Assumption 3 for $\kappa = 100$. Finally, Fig. 5 depicts the evolution of the control inputs $u_i(t)$, $u_{i,nn}(t)$, $i \in \{1, \ldots, 5\}$. One concludes that the multi-agent system converges successfully to the pre-specified formation, whose x-y shape is depicted in the bottom-right plot of Fig. 2.

Case 2: The second case comprises a surveillance task, where the agents need to periodically surveil three areas in the environment. We choose the same communication graph as in the first case. Each area consists of 6 spherical regions of interest; the regions of interest of the first area are centered at $[-50, -50, -10]^\top$, $[-70, -50, 10]^\top$, $[-60, -40, 10]^\top$, $[-40, -40, 10]^\top$, $[-40, -60, 10]^\top$, $[-60, -60, 10]$; the regions of interest of the second area are centered at $[50, 50, 10]^\top$, $[40, 40, 10]^\top$, $[40, 60, 10]^\top$, $[50, 60, 10]^\top$, $[60, 50, 10]^\top$ $[50, 40, 10]^\top$; and the regions of interest of the third area are centered at $[50, -50, 10]^\top$, $[40, -40, 10]^\top$, $[60, -40, 10]^\top$, $[60, -50, 10]^\top$, $[40, -60, 10]^\top$, $[40, -50, 10]^\top$. The leader agent navigates sequentially to one of the regions in the areas, and by setting the constants c_{ij}, $(i, j) \in \bar{\mathcal{E}}$, according to the geometry of the regions, the followers aim to visit the remaining five regions in each area. More specifically, we set the formation constants as $c_{1,2} = -c_{2,1} = [10, 10, 0]^\top$, $c_{2,3} = -c_{3,2} = [20, 0, 0]^\top$, $c_{3,4} = -c_{4,3} = [0, -20, 0]^\top$, $c_{4,5} = -c_{5,4} = [-20, 0, 0]^\top$, $c_{1,0} = [20, 0, 0]^\top$, $c_{3,0} = [-10, -10, 0]^\top$, $c_{5,0} = [10, 10, 0]^\top$ for the first area, $c_{1,2} = -c_{2,1} = [0, 20, 0]^\top$, $c_{2,3} = -c_{3,2} = [10, 0, 0]^\top$, $c_{3,4} = -c_{4,3} = [10, -10, 0]^\top$, $c_{4,5} = -c_{5,4} = [-10, -10, 0]^\top$, $c_{1,0} = [10, 10, 0]^\top$, $c_{3,0} = [0, -10, 0]^\top$, $c_{5,0} = [10, 10, 0]^\top$ for the second area, and $c_{1,2} = -c_{2,1} = [20, 0, 0]^\top$, $c_{2,3} = -c_{3,2} = [0, -10, 0]^\top$, $c_{3,4} = -c_{4,3} = [-20, -10, 0]^\top$, $c_{4,5} = -c_{5,4} = [0, 10, 0]^\top$, $c_{1,0} = [10, -10, 0]^\top$, $c_{3,0} = [-10, 0, 0]^\top$, $c_{5,0} = [10, 0, 0]^\top$ for the third area.

Similarly to the first case, we generate data from 100 trajectories that correspond to different f, g, $\bar{x}(0)$ than in the task's formation instance $\mathcal{F}$; the differences in f, g are created by assigning random values, in $(0, 1)$, to the constants m_i, $A_{i,\ell}$,

This article has been accepted for publication in IEEE Transactions on Automation Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TASE.2025.3528501

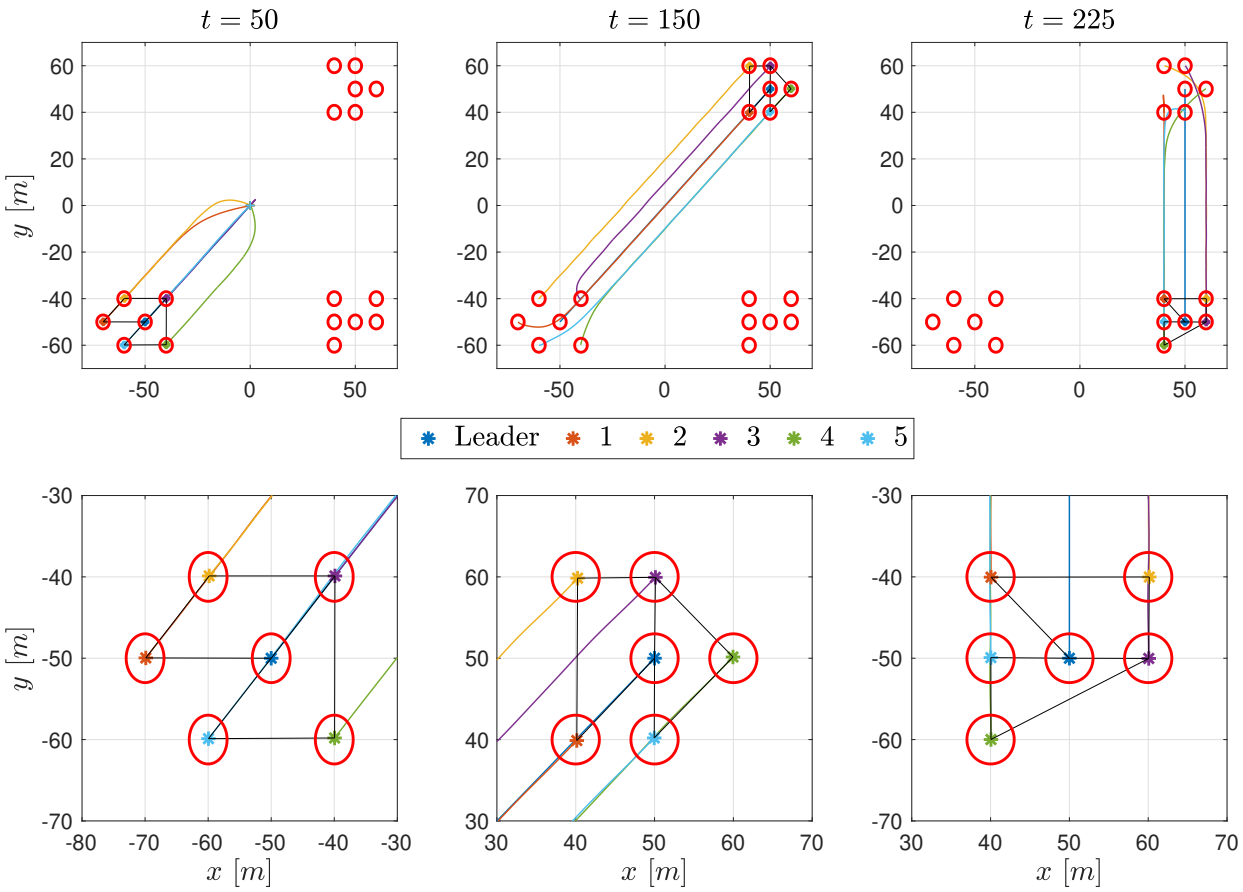

Fig. 6. Snapshots of Case 2's experiment (top) and their zoomed-in versions (bottom) in the x-y plane. The agents converge to the desired formation around the leader at $t = 50$, $t = 150$, and $t = 225$, which implies the visit of the regions of interest in the three areas. The black lines represent the communication edge set $\bar{\mathcal{E}}$ of the agents. The initial positions of the agents are depicted with " + " in the top-left plot.

$\eta_{i,\ell}$, $\phi_{i,\ell}$, and F_i, for all $i \in \mathcal{N}$. The initial conditions of the agents are set as $x_{i,1}(0) = x_{0,1}(0) + \text{rand}(-10, 10)[1, 1, 1]^\top$, and $x_{i,2}(0) = \text{rand}(-2, 2)[1, 1, 1]^\top$, $i \in \mathcal{N}$, and of the leader agent as $x_{0,1} = [0, 0, 10]^\top$, $x_{0,2} = [0, 0, 0]^\top$. We use the data to train 5 neural networks, one for each agent. We test the control policy (13) on $\mathcal{F}$, giving the results depicted in Figs. 6-9; Fig. 6 depicts snapshots of the agents' visit to the three areas (at $t = 50$, $t = 150$, and $t = 225$ seconds, respectively), and Fig. 7 depicts the evolution of the signals $\|e_{i,1}(t)\| + \|\dot{e}_{i,1}(t)\|$ and $\|\dot{e}_{i,2}(t)\|$, for all agents $i \in \{1, \ldots, 5\}$. Fig. 8 shows the evolution of the adaptation variables $\hat{d}_{i,1}(t)$, $i \in \mathcal{N}$, and the signal $CH(t) = e_2(t)^\top H(f(\bar{x}(t), t) + g(\bar{x}(t), t)u(t) - \ddot{\bar{x}}_{0,1}(t)) - 100\|e_2\|$, which is always negative, verifying thus Assumption 3 for $\kappa = 100$. Finally, Fig. 5 depicts the evolution of the control inputs $u_i(t)$, $u_{i,nn}(t)$, $i \in \{1, \ldots, 5\}$. As illustrated in the figures, the agents converge successfully to the three pre-specified formations, visiting the regions of interest in the three areas.

Case 3: The first two cases considered training data that correspond to the exact formation task, defined by the leader profile x_0 and the constants c_{ij}, and communication graph $\bar{\mathcal{G}}$. In the third case, we generate 120 different formation instances $\mathcal{F}^k := (x_0^k, f^k, g^k, c^k, \bar{\mathcal{G}}^k, x^k(0))$, $k \in \{1, \ldots, 120\}$, i.e., different trajectory profiles for the leader, different terms f^k and g^k for the agents, different communication graphs $\bar{\mathcal{G}}$, different formation constants c_{ij}, for $(i, j) \in \bar{\mathcal{E}}$, and different initial conditions for the agents. In every instance k, we set the parameters in f^k, and g^k as in the previous two cases, we set randomly the communication graph $\bar{\mathcal{G}}^k$ such that it satisfies Assumption 2, we set random offsets c_{ij} in the interval $(-5, 5)\bar{1}_3$, for $(i, j) \in \bar{\mathcal{E}}$, and the initial conditions of the agents as $x_{i,1}(0) = \text{rand}(-10, 10)\bar{1}_3$, $x_{i,2}(0) = \text{rand}(-2.5, 2.5)\bar{1}_3$, for all $i \in \{1, \ldots, 5\}$. Finally, the leader trajectory x_0 is set for each instance $k \in \{1, \ldots, 120\}$ as follows: we create four points in $\mathbb{R}^3$ randomly in $(-10, 10)$ in the x- and y- directions, and in $(1, 20)$ in the z direction. We then create a random sequence of these points, and set the leader trajectory as a smooth path that visits them according to that sequence, with a duration of 40 seconds.

We separate the 120 instances into 100 training and 20 test instances. We train next 5 neural networks, one for each agent, using data from system runs that correspond to the 100 first training instances $\mathcal{F}^k$, $k \in \{1, \ldots, 100\}$. We test the control policy on the 20 first training instances $\mathcal{F}^k$, $k \in \{1, \ldots, 20\}$, as well as on the 20 test instances that were not used in the training, i.e., $\mathcal{F}^k$, $k \in \{101, \ldots, 120\}$. In addition, we compare the performance of the proposed control algorithm with (i) a *no-neural-network* (no-NN) control policy, i.e., a policy that does not employ the neural network, (term $u_{i,nn}$ in (13a)), (ii) a non-adaptive control control policy $u_i = u_{i,nn} - k_2 e_{i,2}$,

This article has been accepted for publication in IEEE Transactions on Automation Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TASE.2025.3528501

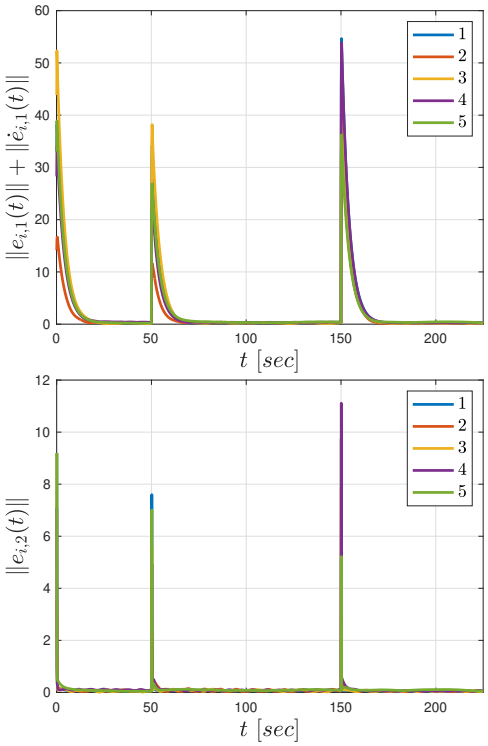

Fig. 7. Evolution of the error signals $\|e_{i,1}(t)\| + \|\dot{e}_{i,1}(t)\|$ and $\|e_{i,2}(t)\|$, for $i \in \{1, \ldots, 5\}$, and $t \in [0, 225]$, in Case 2.

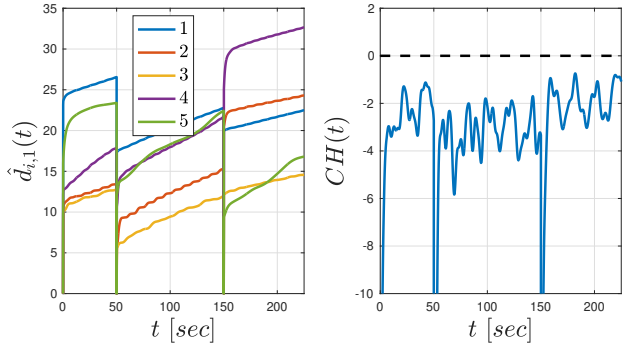

Fig. 8. Left: The evolution of the adaptation signals $\hat{d}_{i,1}(t)$ for $i \in \{1, \ldots, 5\}$, in Case 2. Right: The evolution of $CH(t)$ in Case 2.

i.e., without the adaptation terms $\hat{d}_{i,1}$, and (iii) the control algorithm of [12]. The comparison results are given in Fig. 10, which depicts the mean and standard deviation of the signal $\|\bar{e}_1(t)\| + \|\dot{\bar{e}}_1(t)\|$ for the 20 of the training instances (top), and for the 20 test instances (bottom). In both cases, the proposed control algorithm outperforms the other policies, which, in many of the instances, resulted in unstable closed-loop systems. The control gains of the algorithm of [12] were chosen so that the resulting control-input magnitude is similar to the proposed method. Nevertheless, it should be noted that in many instances the stability of the closed-loop system of [12] required control effort at least one order of magnitude larger than the proposed method. Moreover, from the comparison with the no-NN and non-adaptive policies, one can conclude that both the neural networks and the adaptation terms are necessary in order to achieve the desired

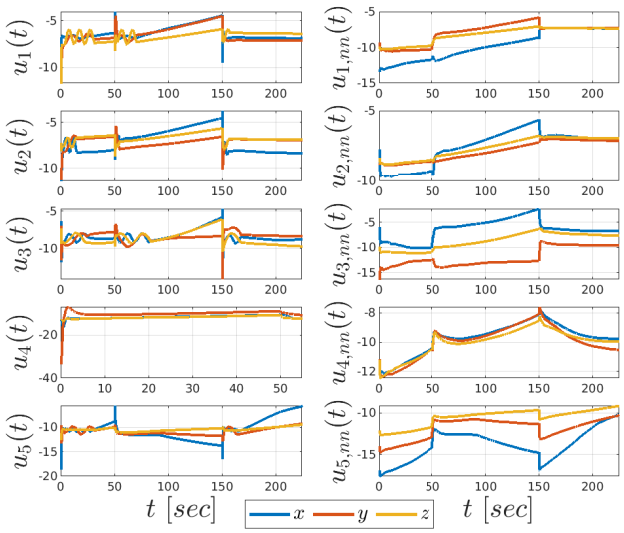

Fig. 9. The evolution of the control inputs $u_i(t)$ and the neural-network controllers $u_{i,nn}(t)$, for $i \in \{1, \ldots, 5\}$, in Case 2.

performance. When the algorithm does not use the learned neural networks (as in the no-NN policy), it fails to drive the multi-agent disagreement error to zero, as depicted in Fig. 10. The reason is that, without the neural networks, there is no way to accommodate the uncertainties imposed by the unknown functions $f(\cdot)$ and $g(\cdot)$. In particular, without the use of neural networks, Assumption 3 does not hold, i.e., there is no κ such that eq. (12) holds. Therefore, the adaptive controller cannot compensate accurately for the dynamic uncertainties and drive the multi-agent errors $e_1(t)$ and $e_2(t)$ to zero. Further note that there is no essential difference among training and test instances for the no-NN policy since there is no use of neural networks. Similarly, the non-adaptive policy is not sufficient to drive the errors to zero, since there are no adaptation terms to compensate for the unknown dynamic terms in real time.

Case 4: Finally, the fourth case shows the applicability of the proposed algorithm to large teams of agents and its robustness to time-varying graphs. In particular, we consider a team of 50 follower agents in $\mathbb{R}^3$, whose connectivity graph $\bar{\mathcal{G}}$ depends on their pair-wise distances; agents i and j are connected, i.e., $(i, j) \in \bar{\mathcal{E}}$, if $\|x_{i,1} - x_{j,1}\| \leq D_s$, where we choose $D_s = 15$. Similarly to Case 3, we generate 120 different formation instances $\mathcal{F}^k := (x_0^k, f^k, g^k, c^k, x^k(0))$, $k \in \{1, \ldots, 120\}$, making sure that the initial conditions $x^k(0)$ form connected connectivity graphs $\bar{\mathcal{G}}^k(0)$. Next, we follow the same procedure as in Case 3, separating the instances into 100 training and 20 test ones. The results are depicted in Fig. 11, which shows the convergence of the errors to zero, illustrating the robustness of the proposed algorithm to switching graphs. It is worth noting that the no-NN control and non-adaptive control policies lead to instability in most of the cases and hence we do not present the respective trajectories.

We now provide more details regarding the collection of data, the form of the neural networks, and the respective

This article has been accepted for publication in IEEE Transactions on Automation Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TASE.2025.3528501

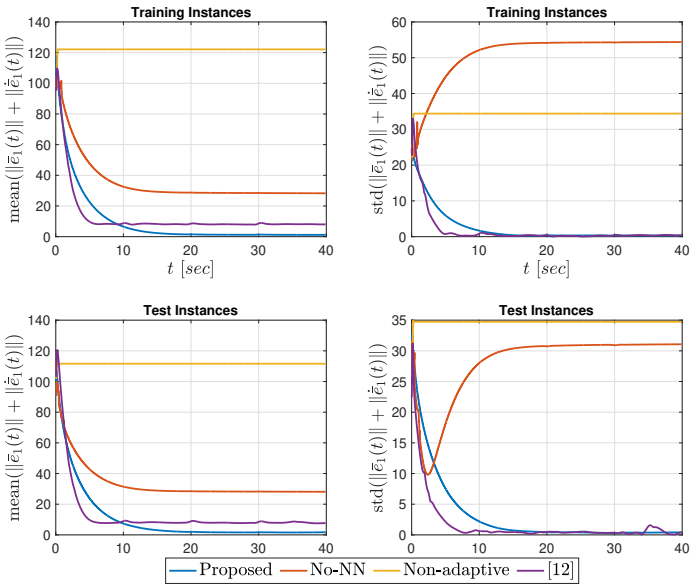

Fig. 10. Evolution of the mean (left) and standard deviation (right) of the signals $\|\bar{e}_1(t)\| + \|\dot{\bar{e}}_1(t)\|$ for the 20 training instances (top) and the 20 test instances (bottom) of Case 3.

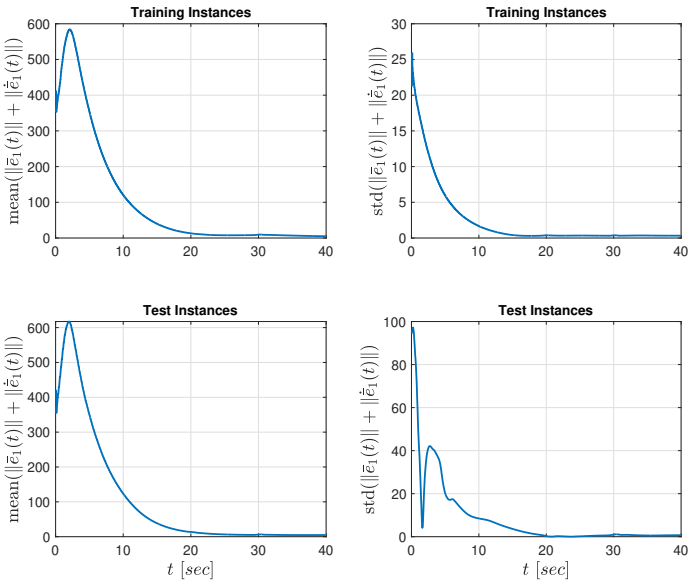

Fig. 11. Evolution of the mean (left) and standard deviation (right) of the signals $\|\bar{e}_1(t)\| + \|\dot{\bar{e}}_1(t)\|$ for the 20 training instances (top) and the 20 test instances (bottom) of Case 4.

training for the aforementioned experiments. For the execution of the trajectories that are used in the training of the neural networks, we use the control policies

$$u_i = g_i(x_i, t)^{-1}(u_0(t) - e_{i,2} - f_i(x_i, t)),$$

for all $i \in \mathcal{N}$. The data for the training of the neural networks consist of 100 system trajectories, sampled at 500 points, making a total of 50000 points. The neural networks we use consist of 4 fully connected layers of 512 neurons; each layer is followed by a batch-normalization module [55] and a ReLU activation function $\mathsf{f}_a(\star) = \max(0, \star)$. For the training, we deploy standard backpropagation using the Adam stochastic optimizer [54] and the Mean-Square-Error loss function; we choose the learning rate of Adam as 10^{-3}. Finally, we use a batch size of 256, and we train the neural networks until an average (per batch) loss of the order of 10^{-4} is achieved.

V. Conclusion and Future Work

We develop a learning-based control algorithm for the formation control of networked multi-agent systems with unknown nonlinear dynamics. The algorithm integrates distributed neural-network-based learning and adaptive control. We provide formal guarantees and perform extensive numerical experiments. Future efforts will focus on relaxing the considered assumptions and extending the proposed methodology to account for directed and time-varying communication graphs as well as underactuated systems.

References

[1] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, Jun. 2003.

[2] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[3] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, no. 7025, pp. 513–516, Feb. 2005.

[4] H. Modares, F. L. Lewis, W. Kang, and A. Davoudi, "Optimal synchronization of heterogeneous nonlinear systems with unknown dynamics," *IEEE Transactions on Automatic Control*, vol. 63, no. 1, pp. 117–131, Jun. 2017.

[5] C. P. Bechlioulis and G. A. Rovithakis, "Decentralized robust synchronization of unknown high order nonlinear multi-agent systems with prescribed transient and steady state performance," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 123–134, Feb. 2016.

[6] C. K. Verginis and D. V. Dimarogonas, "Adaptive leader-follower coordination of lagrangian multi-agent systems under transient constraints," *IEEE Conference on Decision and Control (CDC)*, pp. 3833–3838, 2019.

[7] J. Ni and P. Shi, "Adaptive neural network fixed-time leader–follower consensus for multiagent systems with constraints and disturbances," *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1835–1848, Feb. 2020.

[8] H. Zhang and F. L. Lewis, "Adaptive cooperative tracking control of higher-order nonlinear systems with unknown dynamics," *Automatica*, vol. 48, no. 7, pp. 1432–1439, Jul. 2012.

[9] J. Hu and W. X. Zheng, "Adaptive tracking control of leader–follower systems with unknown dynamics and partial measurements," *Automatica*, vol. 50, no. 5, pp. 1416–1423, May 2014.

[10] C. Chen, C. Wen, Z. Liu, K. Xie, Y. Zhang, and C. P. Chen, "Adaptive consensus of nonlinear multi-agent systems with non-identical partially unknown control directions and bounded modelling errors," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4654–4659, Nov. 2016.

[11] Y. Wang, Y. Song, and W. Ren, "Distributed adaptive finite-time approach for formation–containment control of networked nonlinear systems under directed topology," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 7, pp. 3164–3175, Jul. 2017.

[12] Z. Li, Z. Duan, and F. L. Lewis, "Distributed robust consensus control of multi-agent systems with heterogeneous matching uncertainties," *Automatica*, vol. 50, no. 3, pp. 883–889, Mar. 2014.

[13] Z.-J. Yang, "Robust consensus tracking of second-order nonlinear systems using relative position information by k-filter and disturbance observer based control," *International Journal of Systems Science*, vol. 49, no. 15, pp. 3117–3129, Oct. 2018.

[14] N. Rahimi and T. Binazadeh, "Distributed robust consensus control for nonlinear leader–follower multi-agent systems based on adaptive observer-based sliding mode," *Journal of Vibration and Control*, vol. 25, no. 1, pp. 109–121, Apr. 2018.

[15] Y. Liu and G.-H. Yang, "Neural learning-based fixed-time consensus tracking control for nonlinear multiagent systems with directed com-

munication networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 639–652, Apr. 2020.

[16] J. Qin, G. Zhang, W. X. Zheng, and Y. Kang, "Neural network-based adaptive consensus control for a class of nonaffine nonlinear multiagent systems with actuator faults," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 12, pp. 3633–3644, Mar. 2019.

[17] C. P. Bechlioulis and K. J. Kyriakopoulos, "Robust model-free formation control with prescribed performance for nonlinear multi-agent systems," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1268–1273, 2015.

[18] D. Bertsekas, "Alphazero, off-line training, and on-line play," in *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*, 1st ed. Athena Scientific, Belmont, MA, 2021, pp. 1–18.

[19] C. K. Verginis, Z. Xu, and U. Topcu, "Non-parametric neuro-adaptive coordination of multi-agent systems," *International Conference on Autonomous Agents and Multiagent Systems*, pp. 1747–1749, 2022.

[20] W. Liu and J. Huang, "Adaptive leader-following consensus for a class of higher-order nonlinear multi-agent systems with directed switching networks," *Automatica*, vol. 79, pp. 84–92, May 2017.

[21] H. Rezaee and F. Abdollahi, "Adaptive consensus control of nonlinear multiagent systems with unknown control directions under stochastic topologies," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3538–3547, Aug. 2017.

[22] C. K. Verginis and D. V. Dimarogonas, "Adaptive robot navigation with collision avoidance subject to 2nd-order uncertain dynamics," *Automatica*, vol. 123, p. 109303, Jan. 2021.

[23] C. Wang and H. Ji, "Robust consensus tracking for a class of heterogeneous second-order nonlinear multi-agent systems," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 17, pp. 3367–3383, Nov. 2015.

[24] M. Lu and J. Huang, "Cooperative global robust output regulation for a class of nonlinear multi-agent systems with a nonlinear leader," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3557–3562, Feb. 2016.

[25] U. Münz, A. Papachristodoulou, and F. Allgöwer, "Robust consensus controller design for nonlinear relative degree two multi-agent systems with communication constraints," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 145–151, Oct. 2010.

[26] C. K. Verginis, C. P. Bechlioulis, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Robust distributed control protocols for large vehicular platoons with prescribed transient and steady-state performance," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 299–304, Feb. 2017.

[27] C. K. Verginis, A. Nikou, and D. V. Dimarogonas, "Robust formation control in se (3) for tree-graph structures with prescribed transient and steady state performance," *Automatica*, vol. 103, pp. 538–548, May 2019.

[28] Z. Peng, D. Wang, H. Zhang, and G. Sun, "Distributed neural network control for adaptive synchronization of uncertain dynamical multiagent systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 8, pp. 1508–1519, Dec. 2013.

[29] G. Wen, C. P. Chen, Y.-J. Liu, and Z. Liu, "Neural network-based adaptive leader-following consensus control for a class of nonlinear multiagent state-delay systems," *IEEE Transactions on Cybernetics*, vol. 47, no. 8, pp. 2151–2160, Oct 2016.

[30] L. Cheng, Z.-G. Hou, M. Tan, Y. Lin, and W. Zhang, "Neural-network-based adaptive leader-following control for multiagent systems with uncertainties," *IEEE Transactions on Neural Networks*, vol. 21, no. 8, pp. 1351–1358, Jul. 2010.

[31] C. Yuan, H. He, and C. Wang, "Cooperative deterministic learning-based formation control for a group of nonlinear uncertain mechanical systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 319–333, Jan. 2018.

[32] J. Mei, W. Ren, B. Li, and G. Ma, "Distributed containment control for multiple unknown second-order nonlinear systems with application to networked lagrangian systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 9, pp. 1885–1899, Oct. 2014.

[33] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," in *Handbook of Reinforcement Learning and Control*, 1st ed., ser. Studies in Systems, Decision and Control. Springer, 2021, pp. 321–384.

[34] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," *International Conference on Machine Learning*, pp. 2681–2690, 2017.

[35] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, Oct 2019.

[36] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," *International Conference on Machine Learning*, pp. 1146–1155, 2017.

[37] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," *International Conference on Autonomous Agents and Multiagent Systems*, pp. 66–83, 2017.

[38] K. Zhang, Y. Liu, J. Liu, M. Liu, and T. Başar, "Distributed learning of average belief over networks using sequential observations," *Automatica*, vol. 115, pp. 108–857, May 2020.

[39] E. Dall'Anese, H. Zhu, and G. B. Giannakis, "Distributed optimal power flow for smart microgrids," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1464–1475, Apr. 2013.

[40] S. Kar, J. M. Moura, and H. V. Poor, "Qd-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus and innovations," *IEEE Transactions on Signal Processing*, vol. 61, pp. 1848–1862, Jan. 2013.

[41] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal nash equilibrium in team markov games," *Advances in Neural Information Processing Systems*, vol. 15, 2002.

[42] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, "Multi-agent reinforcement learning via double averaging primal-dual optimization," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[43] T. Doan, S. Maguluri, and J. Romberg, "Finite-time analysis of distributed td (0) with linear function approximation on multi-agent reinforcement learning," *International Conference on Machine Learning*, pp. 1626–1635, 2019.

[44] S. Huang, K. K. Tan, and T. H. Lee, "Nonlinear adaptive control of interconnected systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 243–246, Jan. 2006.

[45] C. K. Verginis and D. V. Dimarogonas, "Asymptotic tracking of second-order nonsmooth feedback stabilizable unknown systems with prescribed transient response," *IEEE Transactions on Automatic Control*, vol. 66, no. 7, pp. 3296–3302, Aug. 2020.

[46] P. Shivakumar and K. H. Chew, "A sufficient condition for nonvanishing of determinants," *Proceedings of the American Mathematical Society*, pp. 63–66, 1974.

[47] R. Hecht-Nielsen, "Theory of the backpropagation neural network," *Neural Networks for Perception*, pp. 65–93, 1992.

[48] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[49] K. G. Vamvoudakis and F. L. Lewis, "Multi-player non-zero-sum games: Online adaptive learning solution of coupled hamilton–jacobi equations," *Automatica*, vol. 47, no. 8, pp. 1556–1569, Aug. 2011.

[50] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, "Cooperative adaptive control for systems with second-order nonlinear dynamics," *Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches*, pp. 259–278, 2013.

[51] Y. Fei, P. Shi, and C.-C. Lim, "Neural network adaptive dynamic sliding mode formation control of multi-agent systems," *International Journal of Systems Science*, vol. 51, no. 11, pp. 2025–2040, Jun. 2020.

[52] D. Liu, C. Li, H. Li, D. Wang, and H. Ma, "Neural-network-based decentralized control of continuous-time nonlinear interconnected systems with unknown dynamics," *Neurocomputing*, vol. 165, pp. 90–98, Oct. 2015.

[53] M. Krstic, I. Kanellakopoulos, and P. Kokotovic, "Adaptive backstepping design," in *Nonlinear and Adaptive Control Design*, 1st ed. Wiley, New York, USA, 1995, pp. 87–122.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, May 2015.

[55] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *International Conference on Machine Learning*, pp. 448–456, 2015.

[56] H. K. Khalil, "Advanced stability analysis," in *Nonlinear Systems*, 3rd ed. Pearson Education, London, UK, 2002.

APPENDIX

We provide here the proof of Theorem 1.

Proof of Theorem 1. Let the continuously differentiable function

$$V_1 := \frac{k_1^2}{2\underline{g}} e_1^\top H^{-1} e_1 + \frac{1}{2\underline{g}} e_2^\top H^{-1} e_2. \tag{14}$$

where $\underline{g} = \min_{i\in\{1,\dots,N\}}\{\lambda_{\min}(g_i)\}$. By differentiating V_1 and using (9), one obtains

$$\begin{aligned}\dot{V}_1 =& -\frac{k_1^3}{\underline{g}} e_1^\top H^{-1} e_1 + \frac{k_1^2}{\underline{g}} e_1^\top H^{-1} e_2 \\ &+ \frac{1}{\underline{g}} e_2^\top \left(f(\bar{x},t) + g(\bar{x},t)u - \ddot{\bar{x}}_{0,1}\right) \\ &- \frac{k_1^2}{\underline{g}} e_2^\top H^{-1} e_1 + \frac{k_1}{\underline{g}} e_2^\top H^{-1} e_2,\end{aligned}$$

and by further using (13a),

$$\begin{aligned}\dot{V}_1 \leq& -k_1^3 e_1^\top H^{-1} e_1 + \frac{k_1 \|H^{-1}\|}{\underline{g}} \sum_{i\in\mathcal{N}} \|e_{i,2}\|^2 \\ &- \frac{1}{\underline{g}} \sum_{i\in\mathcal{N}} e_{i,2}^\top g_i(x_i,t)(k_2 + \hat{d}_{i,1}) e_{i,2} \\ &+ \frac{1}{\underline{g}} e_2^\top \left(f(\bar{x},t) + g(\bar{x},t)u_{nn}(\bar{x}) - \ddot{\bar{x}}_{0,1}\right).\end{aligned}$$

By using the positive definiteness of $g_i(x_i,t)$, the fact that $\underline{g} = \min_{i\in\mathcal{N}}\{\lambda_{\min}(g_i)\}$, and the fact that $\hat{d}_{i,1}(t)$ is positive, $i \in \mathcal{N}$, we obtain

$$\begin{aligned}\dot{V}_1 \leq& -k_1^3 e_1^\top H^{-1} e_1 + \frac{k_1 \|H^{-1}\|}{\underline{g}} \sum_{i\in\mathcal{N}} \|e_{i,2}\|^2 \\ &- \sum_{i\in\mathcal{N}} (k_2 + \hat{d}_{i,1}) \|e_{i,2}\|^2 \\ &+ \frac{1}{\underline{g}} e_2^\top \left(f(\bar{x},t) + g(\bar{x},t)u_{nn}(\bar{x}) - \ddot{\bar{x}}_{0,1}\right),\end{aligned}$$

and in view of Assumption 3, for $\|e\| \leq r$,

$$\begin{aligned}\dot{V}_1 \leq& -k_1^3 e_1^\top H^{-1} e_1 \\ &- \sum_{i\in\mathcal{N}} \left(k_2 + \hat{d}_{i,1} - \frac{k_1 \|H^{-1}\|}{\underline{g}} - \frac{\kappa}{\underline{g}}\right) \|e_{i,2}\|^2.\end{aligned} \tag{15}$$

By using $d_1 = \frac{k_1\|H^{-1}\|}{\underline{g}} + \frac{\kappa}{\underline{g}}$, (15) becomes

$$\dot{V}_1 \leq -k_1^3 e_1^\top H^{-1} e_1 - \sum_{i\in\mathcal{N}} (k_2 + \hat{d}_{i,1} - d_1) \|e_{i,2}\|^2. \tag{16}$$

In view of the aforementioned expression, the individual adaptation variables $\hat{d}_{i,1}$ aim to approximate d_1. Therefore, we define the adaptation errors $\widetilde{d}_1 := [\widetilde{d}_{1,1}, \dots, \widetilde{d}_{N,1}]^\top := \hat{d}_1 - \bar{d}_1 := [\hat{d}_{1,1} - d_1, \dots, \hat{d}_{N,1} - d_1]^\top$, and the overall state $\widetilde{x} := [e_1^\top, e_2^\top, \widetilde{d}_1^\top]^\top \in \mathbb{R}^{N(2n+1)}$. Let the continuously differentiable function

$$V_2(\widetilde{x}) := V_1(\widetilde{x}) + \frac{1}{2} \widetilde{d}_1^\top M_1^{-1} \widetilde{d}_1,$$

where $M_1 := \text{diag}\{\mu_{1,1}, \dots, \mu_{N,1}\}$. Note that $V_2(\widetilde{x})$ satisfies $W_{\underline{m}}(\widetilde{x}) \leq V_2(\widetilde{x}) \leq W_{\bar{m}}(\widetilde{x})$, where $W_{\underline{m}}(\widetilde{x}) := \underline{m}\|\widetilde{x}\|^2$, $W_{\bar{m}}(\widetilde{x}) := \bar{m}\|\widetilde{x}\|^2$ for some positive constants $\underline{m}$, $\bar{m}$. By differentiating V_2 and using (16), we obtain

$$\begin{aligned}\dot{V}_2 \leq& -e_1^\top K_1^2 H^{-1} K_1 e_1 - \sum_{i\in\mathcal{N}} (k_{i,2} + \hat{d}_{i,1} - d_1) \|e_{i,2}\|^2 \\ &+ \sum_{i\in\mathcal{N}} \frac{1}{\mu_{i,1}} \widetilde{d}_{i,1} \dot{\hat{d}}_{i,1},\end{aligned}$$

and by substituting (13b),

$$\dot{V}_2 \leq -k_1^3 e_1^\top H^{-1} e_1 - \sum_{i\in\mathcal{N}} k_2 \|e_{i,2}\|^2 =: -W_Q(\widetilde{x}).$$

Therefore, $V_2(t) \leq V_2(0)$, implying the boundedness of $e_1(t)$, $e_2(t)$, and $\widetilde{d}_1(t)$, for all $t \geq 0$. In view of (13), we also conclude the boundedness of $u(t)$ and $\dot{\hat{d}}_1(t)$, for all $t \geq 0$. By differentiating $\dot{V}_2$ and using (9) and (13), we further conclude the boundedness of $\ddot{V}_2(t)$, $t \geq 0$, which implies the uniform continuity of V_2. By employing Barbalat's Lemma (Theorem 8.4 of [56]), we conclude that $\lim_{t\to\infty} e_1(t) = \lim_{t\to\infty} e_2(t) = 0$.

In view of Assumptions 1 and 3, the aforementioned results hold under the conditions $x \in \Omega_x := \Omega_1 \times \cdots \times \Omega_N$ and $\|e\| \leq r$. Therefore, we need to establish that the proposed control algorithm and initial conditions do not force $e(t)$ to grow larger than r at any point in time $t \geq 0$. Alternatively, we need to establish that, for $\widetilde{x}(0) \in \bar{\Omega}$, it holds that $\widetilde{x}(t) \in \Omega_x$ and $\|e(t)\| \leq r$, for all $t \geq 0$. Let the set

$$\mathcal{M} := \{\widetilde{x} \in \mathbb{R}^{N(2n+1)} : V_2(\widetilde{x}) \leq V_0\},$$

where we choose V_0 as the largest constant for which $\mathcal{M} \subseteq \{\widetilde{x} \in \mathbb{R}^{N(2n+1)} : \widetilde{x} \in \Omega_x, \|e\| \leq r, \widetilde{d}_1 \leq V_2(\widetilde{x}(0))\}$. Then, for all $\widetilde{x}(0) \in \bar{\Omega}$, where $\bar{\Omega} \subseteq \mathcal{M}$, it follows that V_2 is bounded from above by $V_2(\widetilde{x}(0))$, which implies that $\widetilde{x} \in \Omega_x$ and $\|e(t)\| \leq r$, for all $t \geq 0$. Since $\widetilde{x} = [e^\top, \widetilde{d}_1]^\top = [e^\top, \hat{d}_1 - \bar{d}_1]^\top$ and $\bar{d}_1$ is constant, $\widetilde{x}(0) \in \bar{\Omega}$ implies $[e(0)^\top, \hat{d}_1(0)^\top]^\top \in \bar{\Omega}_{\hat{x}} := \{[e^\top, \hat{d}_1^\top]^\top \in \mathbb{R}^{N(2n+1)} : \widetilde{x} \in \Omega_x\}$, leading to the conclusion of the proof.

□