A Policy Iteration Approach for Flock Motion Control

Shuzheng Qu*, Mohammed Abouheaf†, Wail Gueaieb* and Davide Spinello‡
*School of Electrical Engineering and Computer Science
‡Department of Mechanical Engineering
University of Ottawa, Ottawa, Ontario, Canada K1N 6N5
Email: {fqu096, wgueaieb, dspinell}@uottawa.ca
†College of Technology, Architecture & Applied Engineering
Bowling Green State University, Bowling Green, 43402, OH, USA
Email: mabouhe@bgsu.edu

***Abstract*—The flocking motion control is concerned with managing the possible conflicts between local and team objectives of multi-agent systems. The overall control process guides the agents while monitoring the flock-cohesiveness and localization. The underlying mechanisms may degrade due to overlooking the unmodeled uncertainties associated with the flock dynamics and formation. On another side, the efficiencies of the various control designs rely on how quickly they can adapt to different dynamic situations in real-time. An online model-free policy iteration mechanism is developed here to guide a flock of agents to follow an independent command generator over a time-varying graph topology. The strength of connectivity between any two agents or the graph edge weight is decided using a position adjacency dependent function. An online recursive least squares approach is adopted to tune the guidance strategies without knowing the dynamics of the agents or those of the command generator. It is compared with another reinforcement learning approach from the literature which is based on a value iteration technique. The simulation results of the policy iteration mechanism revealed fast learning and convergence behaviors with less computational effort.**

***Index Terms*—robotics, multi-agent, reinforcement learning, control systems, machine learning**

I. Introduction

Cooperative control schemes can sometimes be challenged by the increasing complexity of the dynamics and connectivity of their underlying systems [1]. This can be manifested in many applications where conflicting objectives may coexist, such as fleets of self-driving vehicles, unmanned aircraft, and large warehouse robots [2], [3]. The autonomy of such interacting systems relies on how well the pre-tuned control strategies can cope with the unmodeled dynamics or even perform under unexpected operational scenarios. Many distributed control approaches have been investigated and widely adopted in this domain [4], [5]. The flocking motion control problem is one category of these problems where each agent decides its action based on a compromise between a set of competing, and possibly conflicting, objectives [6]. In the context of the current manuscript, The objectives of the flock control scheme can be summarized as the guidance of agents towards a common goal, preventing collision among the flock by imposing a minimum safety distance between the agents, and the cohesion of the flock members towards a common speed [7], [8]. Herein, a model-free Reinforcement Learning (RL) process is used to develop a control strategy to guide agents communicating over a time-varying graph topology. Further, the control strategies are adapted online in real-time without prior knowledge of the agent dynamics. This research integrates ideas from optimal control, adaptive control, graph theory, reinforcement learning, and fuzzy logic to solve the aforementioned flocking motion control problem.

The study of multi-agent systems pertains to the behavior analysis of a large number of agents, such as flocks of flying birds and fish schools. Since the early works in [9], [10], researchers have been reflecting back these ideas to develop distributed control approaches for cooperative systems. Distributed estimators have been employed to solve a leader-follower tracking problem under noisy environments in [11]. Sliding surface control is a common approach that is used to handle the coordination tasks within multi-agent systems [11]. Adaptive control mechanism is used to trace the peaks of unknown fields for a multi-agent system under uncertain environments in [12]. A server-based approach is employed to control a group of embedded control systems in [13]. Another distributed control scheme is developed by combining consensus algorithms and attraction/repulsion potential functions in [14]. The consensus control problems for multi-agent systems communicating over fixed graph topologies are solved using pinning gain control ideas in [15]–[17].

Machine learning provide useful tools like reinforcement learning, fuzzy systems, and neural networks which are employed to solve many adaptive control problems [18], [19]. The RL processes enable the agent to learn the best strategy-to-follow through interactions with its dynamic environment [20]. This is often done using one of two two-step techniques known as Value Iteration (VI) and Policy Iteration (PI). The implementation of these techniques is often performed using means of adaptive critics [21]. PI processes are developed for systems interacting over graphs using actor-critic neural structures in [22], [23]. On a relevant side, fuzzy logic is applied to solve control problems for systems of imprecise,

This work was partially supported by NSERC Grant EGP 537568-2018.

This is the postscript version of the published paper. (doi: 10.1109/ROSE52750.2021.9611776)

vague, or uncertain dynamics. It provides not only a way of involving the designer experience in solving such problems but it can also lead to a balanced strategy for the conflict-resolution of multiple optimization criteria [24]. Fuzzy logic is employed to develop an obstacle avoidance scheme to solve a leader-follower problem in [25], [26]. A fuzzy logic scheme based on an extended Takagi-Sugeno-Kang (TSK) inference system is introduced in a collision-avoidance task in a flock motion control application [8]. A combined Fuzzy-RL system is tested in [27] for the online tuning of the rule consequences of a zero-order TSK fuzzy system.

The implementation of PI solutions using regression models, like the online and offline least squares methods, may result in unstable computational processes [28], [29]. This problem becomes more challenging when a multi-agent system is considered where it is required to solve a set of coupled Bellman optimality equations simultaneously [30]. Recursive least squares (RLS) regression approaches provide adequate mechanisms to handle such concerns taking into account the online processing of the sensory data feedback [31]. The RLS approach has been utilized in many applications, like in real-time signal processing and channel equalization [32], [33], adaptive control [34], and wireless communication [35], to name a few. A generalized data estimator that is based on RLS is employed for data analysis, mining and prediction in [36]. The RLS technique exhibited better convergence features and faster parameter tracking abilities when compared with other gradient-based search techniques [33]. A distributed estimator based on a communication diffusion algorithm is developed to optimize a global RLS criteria for a multi-agent system in [37]. RLS is applied to tune neuro-fuzzy structures for different optimization applications and adaptive systems [38]–[41]. A self-tuning scheme based on RLS learning approach is used to adapt the fuzzy rules of a TSK fuzzy system in [39]. A generalized RLS approach is adopted to train a neural network structure in [42].

The paper tackles the problem of flock motion control. The contributions of this work are two-fold: First, a PI approach is advised to solve a set of coupled Bellman optimality equations in real-time. This is accomplished to adapt tracking strategies in a distributed fashion without any prior knowledge of the agents' dynamics. Second, an algorithm based on RLS is employed to implement the PI solution. The proposed solution offers faster convergence features when compared with a similar method based on a VI process [27]. The current work considers a time-dependent connectivity graph among the agents, unlike the fixed connectivity assumed in [27]. The remaining of the paper is organized as follows: Section II introduces the task in hand casted as an optimization problem along with a high-level overview of the proposed motion control approach. The control structure is described in details in Sections III to V. Section VI validates the proposed method through a number of test cases. Finally, Section VII offers some concluding remarks.

II. Problem Formulation

The problem in hand is to control the motion of M-agents communicating over an undirected time-varying graph topology. The objective is to guide the flock members to follow a leader while satisfying the following requirements simultaneously: 1) avoid collision between neighboring agents; and 2) adjust the velocities of agents to reach a consensus on common flock velocity. The leader $\tau \in \{1, 2, \dots, M\}$ could be an independent command source as well as a virtual dynamic trajectory. The leader's dynamics are independent of those of the followers. Hence, this problem is generalizable to different classes that range from guidance of unmanned vehicles to solving pursuer-evader games.

The dynamics of each agent are approximated by

$$\boldsymbol{g}^i_{k+1} = \boldsymbol{g}^i_k + T\,\boldsymbol{h}^i_k \qquad\qquad \boldsymbol{h}^i_{k+1} = \boldsymbol{h}^i_k + T\,\boldsymbol{c}^i_k$$

where $\boldsymbol{g}^i_k = [x^i_k \;\; y^i_k]^T \in \mathbb{R}^2$ and $\boldsymbol{h}^i_k = [v^{i,x}_k \;\; v^{i,y}_k]^T \in \mathbb{R}^2$ represent the position and linear velocity at discrete time step $k \in \mathbb{N}$, respectively, T is the sampling period, and $\boldsymbol{c}^i_k = [c^{i,x}_k, c^{i,y}_k]^T \in \mathbb{R}^2$ is the control signal vector of agent $i \in \{1, 2, \dots, M\}$. Agent i receives different measurements at time-step k and hence generates a corresponding overall control signal $\boldsymbol{c}^i_k$.

The local objective of the flock members is to keep a desired safety distance between each agent i and its neighbors. The team objectives involve reaching a consensus on flock-velocity and that the agents maintains an average proximity d_t from the leader. The objectives can be formally described by the following relations for each agent $i \in \{1, \dots, M\} \setminus \{\tau\}$ and $\gamma \in \{x, y\}$:

$$\lim_{k\to\infty} \Big(\sum_{i\in\{1,\dots,M\}\setminus\{\tau\}} \big\|\boldsymbol{g}^i_k - \boldsymbol{g}^\tau_k\big\| \Big) \Big/ \Big(M-1\Big) = d_t \tag{1a}$$

$$\lim_{k\to\infty} \Big|\gamma^i_k - \gamma^j_k\Big| \geq s, \;\; \forall j \in \mathcal{M}_i \tag{1b}$$

$$\lim_{k\to\infty} v^{i,\gamma}_k = v^{\sim,\gamma} \tag{1c}$$

where $\mathcal{M}_i$ is the set of neighborhood agents of agent i, s is the desired safety distance between each agent and its neighbors, and $v^{\sim,\gamma}$ is the flock's consensus speed. The significance of γ is to articulate that the x and y-components of the different signals are treated in a similar fashion. To reflect the system objectives, the control command of each agent $i \in \{1, \dots, M\}\setminus\{\tau\}$ is formulated as an aggregate of three auxiliary control signals as

$$c^{i,\gamma}_k = c^{i,\gamma}_{t,k} + c^{i,\gamma}_{v,k} + c^{i,\gamma}_{d,k}. \tag{2}$$

The tracking control signal $c^{i,\gamma}_{t,k}$ is decided as a function of the positions of each follower and leader, $c^{i,\gamma}_{t,k} = f^i_{t,\gamma}(\gamma^i, \gamma^\tau)$. The signal $c^{i,\gamma}_{v,k}$ achieves a consensus on a common flock velocity in real-time. This signal is determined in terms of the positions and velocities of each agent and those of its neighbors $c^{i,\gamma}_{v,k} = f^i_{v,\gamma}(v^{i,\gamma}_k, v^{j,\gamma}_k, \gamma^i_k, \gamma^j_k)$, $\forall j \in \mathcal{M}_i$. The separation control signal $c^{i,\gamma}_{d,k}$ is applied to maintain a desired separation distance between the agents. This value relies on a

function of the positions of each follower and its neighbors, such that

$$c_{d,k}^{i,\gamma} = f_{d,\gamma}^{i}(\gamma_k^i, \gamma_k^j) = \frac{\sum_{j\in\mathcal{M}_i} c_{d,k}^{ij,\gamma}}{|\mathcal{M}_i|}, \tag{3}$$

where $|\mathcal{M}_i|$ denotes the cardinality of $\mathcal{M}_i$ and $c_{d,k}^{ij,\gamma}$ represents the partial contribution of the separation control decision taken by agent i due to each agent $j \in \mathcal{M}_i$.

The communication information between the agents are exchanged over a time-varying graph topology, where the position and velocity neighborhood measurements are available locally to each agent. Further, the measurements related to the leader are accessed by the flock members. Due to the competing nature of the different objectives, due to their possibly conflicting nature, there is a compromise to be made so that the control action taken by each agent balances such conflicting goals. Hence, the control algorithm continuously updates the underlying strategies to improve the quality of attempted strategies.

III. Tracking Control Strategy

The section introduces a model-free policy iteration process to compute the tracking control strategy $c_{t,k}^{i,\gamma}$ for each agent i to satisfy objective (1a). This is accomplished in real-time and without knowing the dynamics of the leader or any of the followers.

A. Optimization Framework

The optimal control framework advises the tracking strategy while relying on tracking error measurements between the positions of the leader and agents. Each agent i uses an error vector where it stores the three recent tracking error measurements $\boldsymbol{Z}_k^{i,\gamma} = [\gamma_k^i - \gamma_k^\tau,\ \gamma_{k-1}^i - \gamma_{k-1}^\tau,\ \gamma_{k-2}^i - \gamma_{k-2}^\tau]^T \in \mathbb{R}^3$. The number of error instances in that vector can be customized by the designer in order to balance between the required accuracy and complexity of the problem. Herein, a time window of three most recent error samples is found to be sufficient for the task in hand. The goal is to adapt the tracking strategies in order to annihilate the average tracking error ϵ_t, which is defined as

$$\epsilon_t = \Bigg(\sum_{i\in\{1,\dots,M\}\setminus\{\tau\}} \|\boldsymbol{g}_k^i - \boldsymbol{g}_k^\tau\| \Bigg) \Big/ \Big(M - 1\Big) - d_t$$

A performance index $\varsigma_0^{i,\gamma} = \sum_{k=0}^{\infty} W_k^{i,\gamma}(\boldsymbol{Z}_k^{i,\gamma}, c_{t,k}^{i,\gamma})$ is considered to evaluate the quality of the tracking strategy at each time step for agent i, where $W_k^{i,\gamma}$ is a convex quadratic utility function given by

$$W_k^{i,\gamma}(\boldsymbol{Z}_k^{i,\gamma}, c_{t,k}^{i,\gamma}) = \frac{1}{2}\left[\boldsymbol{Z}_k^{i,\gamma T} \boldsymbol{J}^i \boldsymbol{Z}_k^{i,\gamma} + K^i (c_{t,k}^{i,\gamma})^2\right], \tag{4}$$

where $\boldsymbol{0} < \boldsymbol{J}^i \in \mathbb{R}^{3\times3}$ and $0 < K^i \in \mathbb{R}$ are weighting matrices for the tracking errors and control signal. The inequality symbols "$> \boldsymbol{0}$" and "$\geq \boldsymbol{0}$" refer to positive definite and positive semi-definite matrices, respectively.

The optimal control goal is to select the tracking strategy that minimizes the performance index $\varsigma^{i,\gamma}$ over the infinite horizon. First, a quadratic solving value function is assumed so that $Q^{i,\gamma}(\boldsymbol{Z}_k^{i,\gamma}, c_{t,k}^{i,\gamma}) \triangleq \varsigma_k^{i,\gamma}$. The structure of function $Q^{i,\gamma}$ is motivated by that of the utility function, such that

$$Q^{i,\gamma}(\boldsymbol{Z}_k^{i,\gamma}, c_{t,k}^{i,\gamma}) = \frac{1}{2}\begin{bmatrix} \boldsymbol{Z}_k^{i,\gamma T} & c_{t,k}^{i,\gamma} \end{bmatrix} \boldsymbol{G}^{i,\gamma} \begin{bmatrix} \boldsymbol{Z}_k^{i,\gamma} \\ c_{t,k}^{i,\gamma} \end{bmatrix}$$

$$\text{such that, } \boldsymbol{G}^{i,\gamma} \equiv \begin{bmatrix} \boldsymbol{G}^{i,\gamma}_{\boldsymbol{Z}^{i,\gamma}\boldsymbol{Z}^{i,\gamma}} & \boldsymbol{G}^{i,\gamma}_{\boldsymbol{Z}^{i,\gamma}c_t^{i,\gamma}} \\ \boldsymbol{G}^{i,\gamma}_{c_t^{i,\gamma}\boldsymbol{Z}^{i,\gamma}} & \boldsymbol{G}^{i,\gamma}_{c_t^{i,\gamma}c_t^{i,\gamma}} \end{bmatrix} \in \mathbb{R}^{4\times4}$$

where $\boldsymbol{G}^{i,\gamma} > \boldsymbol{0}$, $\boldsymbol{G}^{i,\gamma}_{c_t^{i,\gamma}c_t^{i,\gamma}} \in \mathbb{R}$, and $\boldsymbol{G}^{i,\gamma}_{c_t^{i,\gamma}\boldsymbol{Z}^{i,\gamma}} \in \mathbb{R}^{1\times3}$.

This solving structure along with the infinite-horizon performance index yield a temporal difference (Bellman) equation that is given by

$$Q^{i,\gamma}(\boldsymbol{Z}_k^{i,\gamma}, c_{t,k}^{i,\gamma}) = W_k^{i,\gamma}(\boldsymbol{Z}_k^{i,\gamma}, c_{t,k}^{i,\gamma}) + Q^{i,\gamma}(\boldsymbol{Z}_{k+1}^{i,\gamma}, c_{t,k+1}^{i,\gamma}). \tag{5}$$

Hence, Bellman's optimality condition is applied to find the optimal tracking strategy $c_{t,k}^{i,\gamma(*)} = \arg\min_{c_{t,k}^{i,\gamma}} \Big(Q^{i,\gamma}(\boldsymbol{Z}_k^{i,\gamma}, c_{t,k}^{i,\gamma})\Big)$. Thus,

$$c_{t,k}^{i,\gamma(*)} = -\left(\boldsymbol{G}^{i,\gamma}_{c_t^{i,\gamma}c_t^{i,\gamma}}\right)^{-1} \boldsymbol{G}^{i,\gamma}_{c_t^{i,\gamma}\boldsymbol{Z}^{i,\gamma}} \boldsymbol{Z}_k^{i,\gamma}. \tag{6}$$

Applying (6) into (5) yields the Bellman optimality equation

$$Q^{i,\gamma(*)}(\boldsymbol{Z}_k^{i,\gamma}, c_{t,k}^{i,\gamma(*)}) = W_k^{i,\gamma}(\boldsymbol{Z}_k^{i,\gamma}, c_{t,k}^{i,\gamma(*)}) + Q^{i,\gamma(*)}(\boldsymbol{Z}_{k+1}^{i,\gamma}, c_{t,k+1}^{i,\gamma(*)}). \tag{7}$$

Solving (6) and (7) simultaneously for each agent would solve the underlying guidance or optimal trajectory tracking problem. Therefore, approximate or regression methods are needed to implement the solutions of (6) and (7) in real-time.

A RL technique based on Policy Iteration (PI) is adopted to provide an online solution for this problem. This is done recursively by solving the following temporal difference (Bellman) form:

$$Q^{i,\gamma(r)}(\boldsymbol{Z}_k^{i,\gamma}, c_{t,k}^{i,\gamma}) - Q^{i,\gamma(r)}(\boldsymbol{Z}_{k+1}^{i,\gamma}, c_{t,k+1}^{i,\gamma}) = W_k^{i,\gamma(r)}(\boldsymbol{Z}_k^{i,\gamma}, c_{t,k}^{i,\gamma}).$$

Then, the policy is updated using

$$c_{t,k}^{i,\gamma(r+1)} = -\left(\left(\boldsymbol{G}^{i,\gamma}_{c_t^{i,\gamma}c_t^{i,\gamma}}\right)^{-1} \boldsymbol{G}^{i,\gamma}_{c_t^{i,\gamma}\boldsymbol{Z}^{i,\gamma}}\right)^{(r)} \boldsymbol{Z}_k^{i,\gamma}. \tag{8}$$

This solution algorithmic form is implemented using a recursive least squares regression approach, which is explained below.

B. Recursive Least Squares

The PI solution is implemented in two steps. First, a given policy is evaluated. Second, the tracking strategy-to-follow is improved. Therefore, the RLS approach solves (5) for the optimal value $\boldsymbol{G}^{i,\gamma}$ or strategy using value function approximation since it is not possible to solve Bellman optimality equation analytically. The approximated value function $\tilde{Q}^{i,\gamma}$

is represented by

$$\tilde{Q}^{i,\gamma}(\boldsymbol{Z}_k^{i,\gamma}, \tilde{c}_{t,k}^{i,\gamma}) = \frac{1}{2}\begin{bmatrix} \boldsymbol{Z}_k^{i,\gamma^T} & \tilde{c}_{t,k}^{i,\gamma} \end{bmatrix} \boldsymbol{\Psi}^{i,\gamma} \begin{bmatrix} \boldsymbol{Z}_k^{i,\gamma} \\ \tilde{c}_{t,k}^{i,\gamma} \end{bmatrix}$$

$$\text{such that, } \boldsymbol{\Psi}^{i,\gamma} \equiv \begin{bmatrix} \boldsymbol{\Psi}^{i,\gamma}_{\boldsymbol{Z}^{i,\gamma}\boldsymbol{Z}^{i,\gamma}} & \boldsymbol{\Psi}^{i,\gamma}_{\boldsymbol{Z}^{i,\gamma}\tilde{c}_t^{i,\gamma}} \\ \boldsymbol{\Psi}^{i,\gamma}_{\tilde{c}_t^{i,\gamma}\boldsymbol{Z}^{i,\gamma}} & \boldsymbol{\Psi}^{i,\gamma}_{\tilde{c}_t^{i,\gamma}\tilde{c}_t^{i,\gamma}} \end{bmatrix} \in \mathbb{R}^{4\times 4},$$

where $\boldsymbol{\Psi}^{i,\gamma} > 0$, $\boldsymbol{\Psi}^{i,\gamma}_{\tilde{c}_t^{i,\gamma}\tilde{c}_t^{i,\gamma}} \in \mathbb{R}$, and $\boldsymbol{\Psi}^{i,\gamma}_{\tilde{c}_t^{i,\gamma}\boldsymbol{Z}^{i,\gamma}} \in \mathbb{R}^{1\times 3}$.

Let $\boldsymbol{\varphi}_k^{i,\gamma} = \bar{\boldsymbol{x}}_k^{i,\gamma} - \bar{\boldsymbol{x}}_{k+1}^{i,\gamma}$, where $\bar{\boldsymbol{x}}^{i,\gamma} = [0.5z_1^2\ z_1z_2\ z_1z_3\ z_1z_4\ 0.5z_2^2\ z_2z_3\ z_2z_4\ 0.5z_3^2\ z_3z_4\ 0.5z_4^2]$ and z_1 to z_4 correspond to the elements of vector $[\,\boldsymbol{Z}^{i,\gamma^T}\ \tilde{c}_t^{i,\gamma}\,]$. Also, let $\boldsymbol{\vartheta}^{i,\gamma} \in \mathbb{R}^{10\times 1}$ be the entries in the symmetric solution matrix $\boldsymbol{\Psi}^{i,\gamma}$ associated with the vector $\bar{\boldsymbol{x}}^{i,\gamma}$. Hence, it is required to solve the following equation

$$\boldsymbol{\varphi}_k^{i,\gamma}\boldsymbol{\vartheta}^{i,\gamma} = W_k^{i,\gamma}. \tag{9}$$

This is done using RLS technique for each agent i. Then, the approximated optimal strategy-to-follow is calculated by (8) after reconstructing $\boldsymbol{\Psi}^{i,\gamma}$ back from $\boldsymbol{\vartheta}^{i,\gamma}$, such that the improved control strategy follows $\tilde{c}_{t,k}^{i,\gamma} = -\left(\boldsymbol{\Psi}^{i,\gamma}_{\tilde{c}_t^{i,\gamma}\tilde{c}_t^{i,\gamma}}\right)^{-1} \boldsymbol{\Psi}^{i,\gamma}_{\tilde{c}_t^{i,\gamma}\boldsymbol{Z}^{i,\gamma}} \boldsymbol{Z}_k^{i,\gamma}$. The RLS approach solves for the unknown weights $\boldsymbol{\vartheta}_k^{i,\gamma} \in \mathbb{R}^{10\times 1}$ in real-time as follows [43], without requiring any prior knowledge about the agents' dynamics

$$\boldsymbol{\vartheta}_k^{i,\gamma} = \boldsymbol{\vartheta}_{k-1}^{i,\gamma} + \boldsymbol{L}_k^{i,\gamma}\left(W_k^{i,\gamma} - \boldsymbol{\varphi}_k^{i,\gamma}\boldsymbol{\vartheta}_{k-1}^{i,\gamma}\right)$$

$$\boldsymbol{L}_k^{i,\gamma} = \boldsymbol{P}_{k-1}^{i,\gamma}\boldsymbol{\varphi}_k^{i,\gamma^T}\left(1 + \boldsymbol{\varphi}_k^{i,\gamma}\boldsymbol{P}_{k-1}^{i,\gamma}\boldsymbol{\varphi}_k^{i,\gamma^T}\right)^{-1}$$

$$\boldsymbol{P}_k^{i,\gamma} = \left(\boldsymbol{I} - \boldsymbol{L}_k^{i,\gamma}\boldsymbol{\varphi}_k^{i,\gamma}\right)\boldsymbol{P}_{k-1}^{i,\gamma}$$

where $\boldsymbol{L}^{i,\gamma} \in \mathbb{R}^{10\times 1}$ is a gain adaptation vector, $\boldsymbol{P}^{i,\gamma} \in \mathbb{R}^{10\times 10}$ is a covariance matrix, and $\boldsymbol{I}$ is the identity matrix. A PI solution using the RLS method is shown in Algorithm 1. It is executed simultaneously by each agent i.

Algorithm 1 RLS-Based PI Tracking Mechanism

Input:
Weighting matrices $\boldsymbol{J}^i$ and K^i
Number of search iterations $\mathcal{T}_n$
Initial tracking error vector $\boldsymbol{Z}_0^{i,\zeta}$
Initial weights $\boldsymbol{\Psi}^{i,\gamma(0)}$ and the corresponding $\boldsymbol{\vartheta}_0^{i,\gamma}$
Initial $\boldsymbol{P}_0^{i,\gamma}$ and $\boldsymbol{L}_0^{i,\gamma}$
Convergence error ξ and size of a moving window $\mathcal{W}$
Output:
Tuned weights $\boldsymbol{\Psi}^{i,\gamma(*)}$

$k \leftarrow 0$
RLS_weights_converged $\leftarrow$ false
while $k < \mathcal{T}_n$ **and** RSL_weights_converged = false **do**
  Compute control signal $\tilde{c}_{t,k}^{i,\gamma}$ and apply it to agent i
  Find $W_k^{i,\gamma}(\boldsymbol{Z}_k^{i,\gamma}, \tilde{c}_{t,k}^{i,\gamma})$ and $\bar{\boldsymbol{x}}_k^{i,\gamma}$
  Calculate $\boldsymbol{Z}_{(k+1)}^{i,\zeta}, \tilde{c}_{t,k+1}^{i,\gamma}$
  Find $\bar{\boldsymbol{x}}_{k+1}^{i,\gamma}$ and calculate $\boldsymbol{\varphi}_k^{i,\gamma}$
  $k \leftarrow k+1$
  Calculate $\boldsymbol{P}_k^{i,\gamma}$ and $\boldsymbol{L}_k^{i,\gamma}$
  Find $\boldsymbol{\vartheta}_k^{i,\gamma}$, then update $\boldsymbol{\Psi}_k^{i,\gamma}$
  if $k > \mathcal{W}$ **and** $\left\|\boldsymbol{\vartheta}^{i,\zeta\,(k+1-l)} - \boldsymbol{\vartheta}^{i,\zeta\,(k-l)}\right\| \le \xi,\ \forall l \in \{0,1,\ldots,\mathcal{W}\}$, **then**
    $\boldsymbol{\Psi}^{i,\gamma(*)} \leftarrow \boldsymbol{\Psi}^{i,\gamma(k+1)}$
    RSL_weights_converged $\leftarrow$ true
  end if
end while
return $\boldsymbol{\Psi}^{i,\gamma(*)}$

IV. Consensus Control Strategy

The second objective is to achieve cohesiveness among the moving agents through a consensus protocol, as expressed in (1c). This is done using the means of a time-varying communication graph topology $\mathcal{G} = \{\mathcal{M}, \mathcal{L}\}$, where $\mathcal{M} = \{\sigma_i\}_{i=1,\ldots,|\mathcal{M}|}$ is the set of graph nodes of cardinality $|\mathcal{M}|$, and $\mathcal{L} = \{(\sigma_i, \sigma_j) \in \mathcal{M}^2\}$ is the set of undirected edges [44]. The connection strength of each edge $(\sigma_i, \sigma_j) \in \mathcal{L}$ in the undirected graph is denoted as $s_{ij} = s_{ji}$, for $j \neq i$, where $s_{ii} = 0$. The consensus control strategy of each agent i is calculated using

$$c_{v,k}^{i,\gamma} = -\sum_{j\in\mathcal{M}_i} s_{ij}(v_k^{i,\gamma} - v_k^{j,\gamma}). \tag{10}$$

The graph connectivity weights are decided using a scalar pump function $\delta_a(\gamma^{ij})$, such that

$$\delta_a(\gamma^{ij}) = \begin{cases} 1 & ,\gamma^{ij} \in [0, a) \\ \frac{1}{2}\left[1 + \cos\left(\pi\frac{\gamma^{ij}-a}{r-a}\right)\right] & ,\gamma^{ij} \in [a, r) \\ 0 & ,\text{otherwise} \end{cases}$$

where $\gamma^{ij} = \gamma^i - \gamma^j$ is the relative distance between agents i and j. Note that the output of the scalar pump function also takes into account a connectivity communication range r between the agents, which somewhat contributes to the separation control objective [45]. The edge or connectivity weights are calculated as

$$s_{ij}(\gamma^{ij}) = \delta_a\left(\left\|\gamma^{ij}\right\|_\alpha / \|r\|_\alpha\right) \in [0, 1),\ j \neq i.$$

The α-norm is defined by

$$\left\|\gamma^{ij}\right\|_\alpha = \frac{1}{\mu}\left[\sqrt{1 + \mu\|\gamma^{ij}\|^2} - 1\right],$$

where μ is a positive real scalar. This control strategy is adaptive to the agents formation, where the connectivity between the agents can vary in time according to the agents proximity to one other.

TABLE I: Leader velocity

Time period [s]	Linear velocity [m/s]	Angular velocity [°/s]
[0, 20]	0.9	0.0
[20, 25]	0.9	0.0
[25, 45]	1.2	6.6
[45, 65]	1.2	0.0

V. Separation Control Strategy

The separation control strategy (1b) prevents agents from colliding by enforcing repulsive-attraction forces in order to control the localization of agents with respect to each other. A Fuzzy RL adaptation mechanism that uses a zero-order Tagaki-Sugeno (TS) fuzzy logic inference system which is implemented using an online value iteration is considered herein, as detailed in [27]. The RL approach adapts the consequences of the fuzzy rules in real-time. This works according to a temporal difference reward that penalizes the agents form getting closer to each other and vice versa. The approach aggregates the separation policies or decisions made for each agent i in reaction to the other agents in its neighborhood $\mathcal{M}_i$, which results in an overall signal $c_{d,k}^{i,\gamma}$ for each agent i in the γ-direction (recall that $\gamma \in \{x, y\}$), such that

$$c_{d,k}^{i,\gamma} = \frac{\sum_{j \in \mathcal{M}_i} \sum_{f=1}^{\mathcal{F}} \Theta_k^{ij,\gamma(f)} \eta^{ij,\gamma(f)}}{|\mathcal{M}_i|}, \tag{11}$$

where $\mathcal{F}$ is the total number of fuzzy rules. For each rule $f \in \{1, \ldots, \mathcal{F}\}$, $\Theta_k^{ij,\gamma(f)}$ is the firing strength and $\eta^{ij,\gamma(f)}$ is the consequence of that rule f. The latter is tuned online using the RL process presented in [27].

VI. Results

A system of 10 Pioneer-3DX™ mobile robots is simulated to validate the proposed policy iteration approach in Algorithm 1, where one robot plays the role of a leader while the rest of the robots act as followers. The simulation is realized in CoppeliaSim™[1], a realistic robot simulation software that adopts some of the state-of-the-art physics engines to simulate physical phenomena, such as gravity, friction, etc. The robots maximum linear velocity and acceleration are set to 1.2 m/s and 2 m/s^2, respectively, while the angular velocity and acceleration are capped at ±150 °/s and ±150 °/s^2, respectively. Initially, the robots are scattered randomly in the environment, as seen in Fig. 1(a). The trajectory of the flock is shown in Fig. 1, where the leader's trajectory is marked in red. By default, the desired separation distance between agents and the average tracking proximity are taken as $s = 2$ m and $d_t = s$. During the 65 s simulation, the leader assumes different types of trajectories as shown in Table I. To further challenge the controller, 4 followers are decommisioned at the 20 s milestone, as can be seen in Figs. 1(d) to 1(f), and the desired separation distance is increased to $s = 2.5$ m at time 45 s. In other words, there are four stages in the simulation, distributed as $[0 \ldots 20\,\text{s} \ldots 25\,\text{s} \ldots 45\,\text{s} \ldots 65\,\text{s}]$. The weighting matrices of the utility function are set to $\boldsymbol{J}^i = 0.0001 \times \boldsymbol{I}_{3\times3}$ and $K^i = 0.01$, $\forall i$. The RLS simulation parameters are taken as $\mathcal{T}_n = 400$ and $\boldsymbol{P}_0^{i,\gamma} = 100 \times \boldsymbol{I}_{10}$, $\forall i$. The parameters of the position dependent adjacency function are fixed to $a = 1$, $r = 3.5$ m, and $\mu = 0.5$.

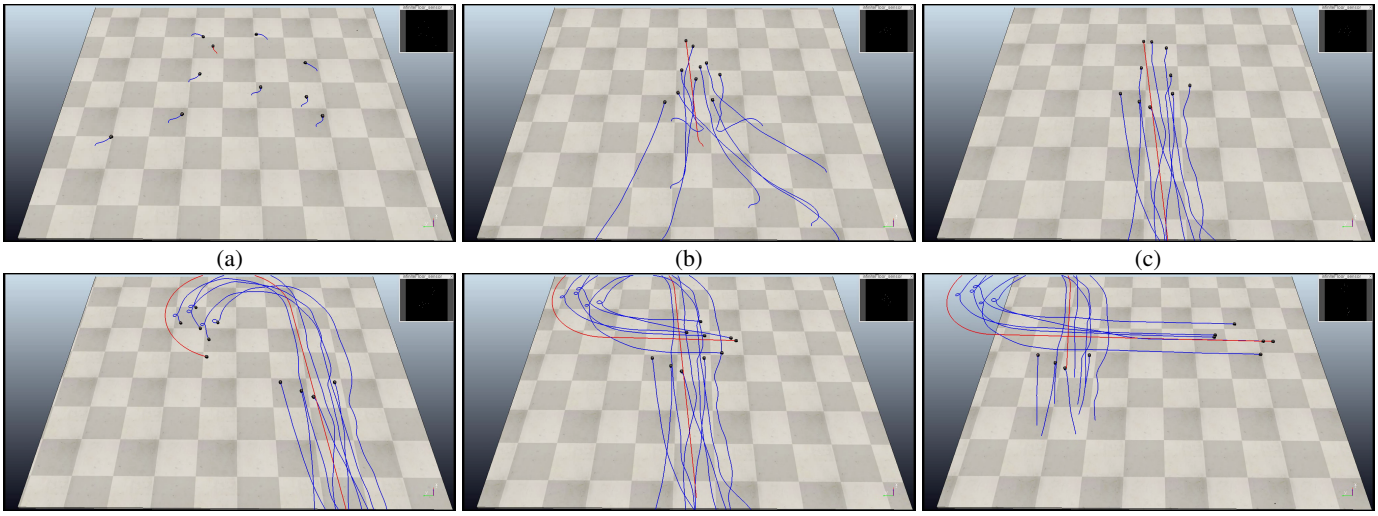
For a more quantitative assessment of the proposed control algorithm, its performance is contrasted against the Fuzzy-RL Value Iteration approach introduced in [27]. The results are shown in Fig. 2, where shaded areas indicate the standard deviation. It is clear from Fig. 2(c) that the followers' average velocity stabilizes in each phase of the simulation, and that in this aspect, the performance of the PI approach is relatively close to that of the VI. As for the average tracking error, it is noticed that it is generally smaller with the PI techniques, as revealed in Fig. 2(a). One can also observe how it is faster to converge towards the end of the simulation than with the VI method. The same remark is applicable to the average separation error between the followers and their neighbors, with the addition that the PI algorithm demonstrates a higher robustness here represented by the smaller standard deviation, as shown in Fig. 2(b).

Another interesting observation about the proposed RLS-based PI algorithm is its faster critic weight convergence along the x and y-directions when compared to its VI counterpart presented in [27]. Fig. 3 shows that the weights of the former method converge in less than 1 s and are not much affected later by the dynamic disturbances throughout the simulation, unlike the VI algorithm. This reflects a higher ability to adapt to time-varying graph topology in the flock decision process, which is encoded in the PI method, rather than relying on a fully connected-graph topology as it is the case with the VI approach.

VII. Conclusion

A novel guidance mechanism based on policy iteration is introduced to solve a flocking motion problem in real-time without knowing the dynamics of the agents or those of the formation. This solution is complemented with an extended fuzzy system to ensure that the agents will stay close to each other and avoid colliding. Additionally, they interact simultaneously considering a time-varying graph topology in order to reach consensus on a common flock velocity. The proposed approach employ model-free strategies and it continuously evaluates and updates the guidance policies without waiting for data batches unlike least squares and batch least squares regression methods. The policy iteration solution based on recursive least squares exhibited better convergence characteristics when compared with another reinforcement learning process that is based on value iteration. A real-world robotics simulation software engine is employed to show the usefulness of the developed solution for a flock of Pioneer-3DX mobile robots.

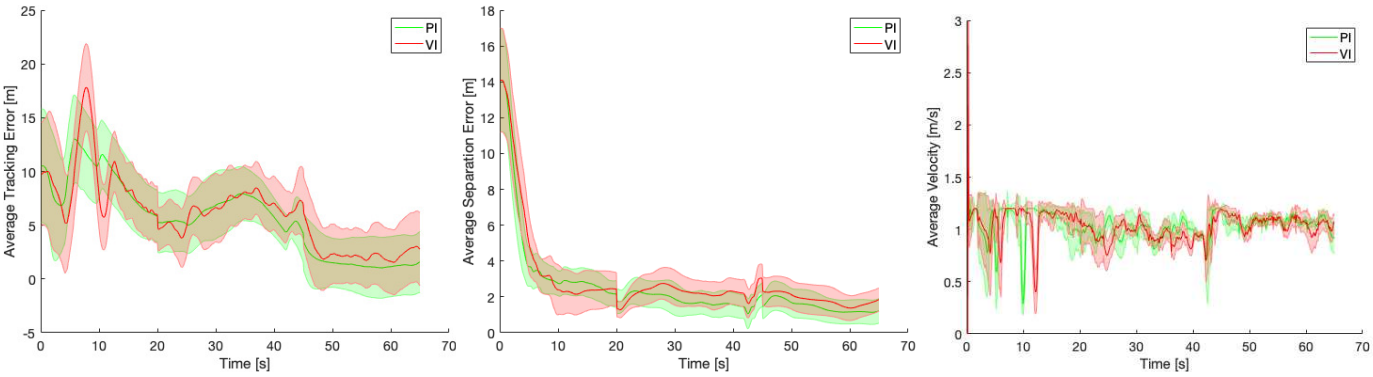
[1] https://coppeliarobotics.com [accessed: July 10, 2021]



(a) (b) (c)

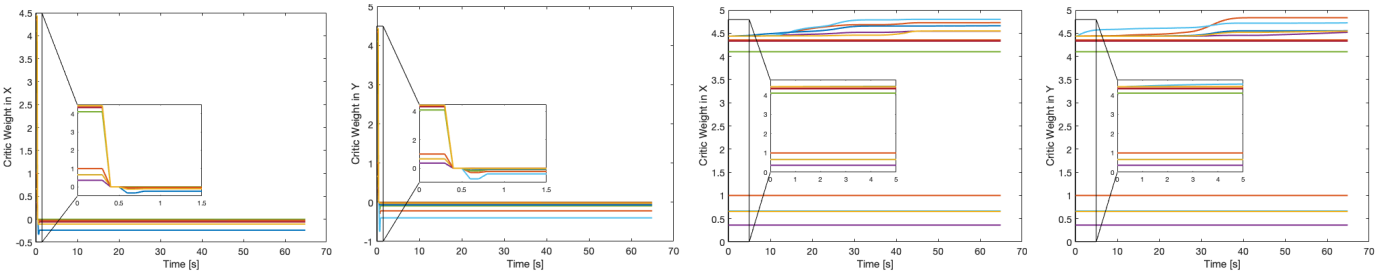
(d) (e) (f)

Fig. 1: Flock trajectory



(a) Average tracking error ϵ_t (b) Average separation error (c) Average follower velocity

Fig. 2: Comparison with the Value Iteration approach of [27]



(a) PI (x) (b) PI (y) (c) VI (x) (d) VI (y)

Fig. 3: Tracking critic weights of the proposed PI technique and the VI approach of [27] along the x and y-directions

REFERENCES

[1] D. Grundel, R. Murphey, P. Pardalos, and O. Prokopyev, *Cooperative Systems: Control and Optimization*. Springer Science & Business Media, Jan. 2007, vol. 588.

[2] Z. Qu, J. Wang, and R. A. Hull, "Cooperative control of dynamical systems with application to autonomous vehicles," *IEEE Transactions on Automatic Control*, vol. 53, no. 4, pp. 894–911, 2008.

[3] P. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses." *AI Magazine*, vol. 29, pp. 9–20, 03 2008.

[4] U. Halder and B. Dey, "Biomimetic algorithms for coordinated motion: Theory and implementation," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5426–5432.

[5] J. M. Soares, A. P. Aguiar, A. M. Pascoal, and A. Martinoli, "A distributed formation-based odor source localization algorithm - design, implementation, and wind tunnel evaluation," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1830–1836.

[6] C. Speck and D. J. Bucci, "Distributed UAV swarm formation control via object-focused, multi-objective sarsa," in *2018 Annual American Control Conference*, 2018, pp. 6596–6601.

[7] D. Gu and H. Hu, "Using fuzzy logic to design separation function in flocking algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 4, pp. 826–838, 2008.

[8] M. Abouheaf and W. Gueaieb, "Flocking motion control for a system of nonholonomic vehicles," in *2017 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*. IEEE, 2017, pp. 32–37.

[9] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

[10] A. A. Paranjape, S.-J. Chung, K. Kim, and D. H. Shim, "Robotic herding of a flock of birds using an unmanned aerial vehicle," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 901–915, 2018.

[11] J. Hu and G. Feng, "Distributed tracking control of leader–follower multi-agent systems under noisy measurement," *Automatica*, vol. 46, no. 8, pp. 1382–1387, 2010.

[12] M. Jadaliha, J. Lee, and J. Choi, "Adaptive control of multi-agent systems for finding peaks of unknown fields," in *Dynamic Systems and Control Conference*, vol. 44182, 2010, pp. 623–630.

[13] S. Oweis, S. Ganesan, and K. C. Cheok, "Server based control flocking for aerial-systems," in *IEEE International Conference on Electro/Information Technology*, 2014, pp. 314–319.

[14] Y. Jia and L. Wang, "Leader–follower flocking of multiple robotic fish," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1372–1383, 2015.

[15] M. I. Abouheaf, F. L. Lewis, K. G. Vamvoudakis, S. Haesaert, and R. Babuska, "Multi-agent discrete-time graphical games and reinforcement learning solutions," *Automatica*, vol. 50, no. 12, pp. 3038–3053, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109814004282

[16] M. I. Abouheaf and F. L. Lewis, "Multi-agent differential graphical games: Nash online adaptive learning solutions," in *52nd IEEE Conference on Decision and Control*, 2013, pp. 5803–5809.

[17] ——, "Approximate dynamic programming solutions of multi-agent graphical games using actor-critic network structures," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–8.

[18] R. S. Sutton, A. G. Barto, and R. J. Williams, "Reinforcement learning is direct adaptive optimal control," *IEEE Control Systems Magazine*, vol. 12, no. 2, pp. 19–22, 1992.

[19] M. Abouheaf, W. Gueaieb, and A. Sharaf, "Load frequency regulation for multi-area power system using integral reinforcement learning," *IET Generation, Transmission & Distribution*, vol. 13, no. 19, pp. 4311–4323, 2019. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-gtd.2019.0218

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., ser. Second. Massachusetts: MIT Press, 1998.

[21] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*, 1st ed. Massachusetts: Athena Scientific, 1996.

[22] M. Abouheaf, F. Lewis, M. Mahmoud, and D. Mikulski, "Discrete-time dynamic graphical games: Model-free reinforcement learning solution," *Control Theory and Technology*, vol. 13, no. 1, pp. 55–69, 2015.

[23] M. Abouheaf and M. Mahmoud, "Policy iteration and coupled riccati solutions for dynamic graphical games," *International Journal of Digital Signals and Smart Systems*, vol. 1, no. 2, pp. 143–162, 2017.

[24] H. Singh, M. M. Gupta, T. Meitzler, Z.-G. Hou, K. K. Garg, A. M. G. Solo, and L. A. Zadeh, "Real-life applications of fuzzy logic," *Advances in Fuzzy Systems*, vol. 2013, pp. 1–3, 2013.

[25] B. Innocenti, B. López, and J. Salvi, "A multi-agent architecture with cooperative fuzzy control for a mobile robot," *Robotics and Autonomous Systems*, vol. 55, no. 12, pp. 881–891, 2007.

[26] H. Zhang, J. Zhang, G.-H. Yang, and Y. Luo, "Leader-based optimal coordination control for the consensus problem of multiagent differential games via fuzzy adaptive dynamic programming," *IEEE Transactions on Fuzzy Systems*, vol. 23, pp. 152–163, Jan. 2014.

[27] S. Qu, M. Abouheaf, W. Gueaieb, and D. Spinello, "An adaptive fuzzy reinforcement learning cooperative approach for the autonomous control of flock systems," in *2021 International Conference on Robotics and Automation (ICRA)*, 2021.

[28] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, "Online least-squares policy iteration for reinforcement learning control," in *Proceedings of the 2010 American Control Conference*, 2010, pp. 486–491.

[29] R. Srivastava, R. Lima, K. Das, and A. Maity, "Least square policy iteration for ibvs based dynamic target tracking," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019, pp. 1089–1098.

[30] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*. John Wiley & Sons, 2012.

[31] B. Yin, M. Dridi, and A. El Moudni, "Approximate dynamic programming with recursive least-squares temporal difference learning for adaptive traffic signal control," in *IEEE Conference on Decision and Control*, 2015, pp. 3463–3468.

[32] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.

[33] D. Lee, M. Morf, and B. Friedlander, "Recursive least squares ladder estimation algorithms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 3, pp. 627–641, 1981.

[34] E. Wilson, C. Lages, and R. W. Mah, "On-line gyro-based, mass-property identification for thruster-controlled spacecraft using recursive least squares," in *The Midwest Symposium on Circuits and Systems (MWSCAS-2002)*, vol. 2, 2002, pp. 1–4.

[35] S. Shafigh, T. Zia, and N. Mouzehkesh, "Wireless accelerometer sensor data filtering using recursive least squares adaptive filter," in *2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2013, pp. 66–70.

[36] W. Xu and F. Liu, "Recursive algorithm of generalized least squares estimator," in *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, vol. 3, 2010, pp. 487–490.

[37] A. Rastegarnia, "Reduced-communication diffusion RLS for distributed estimation over multi-agent networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 1, pp. 177–181, 2020.

[38] R. Yusof, R. Z. Abdul Rahman, M. Khalid, and M. F. Ibrahim, "Optimization of fuzzy model using genetic algorithm for process control application," *Journal of the Franklin Institute*, vol. 348, no. 7, pp. 1717–1737, 2011, special issue on Modeling, Simulation and Applied Optimization. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0016003210002310

[39] J.-W. Yeh and S.-F. Su, "Efficient approach for RLS type learning in TSK neural fuzzy systems," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2343–2352, 2017.

[40] J.-W. Yeh, S.-F. Su, and I. Rudas, "Analysis of using rls in neural fuzzy systems," in *2011 IEEE International Conference on Systems, Man, and Cybernetics*, 2011, pp. 1831–1836.

[41] C.-C. Hu, H.-Y. Lin, and J.-H. Wen, "An adaptive fuzzy-logic variable forgetting factor rls algorithm," in *2005 IEEE 62nd Vehicular Technology Conference, 2005*, vol. 3. IEEE, 2005, pp. 1412–1416.

[42] Y. Xu, K.-W. Wong, and C.-S. Leung, "Generalized rls approach to the training of neural networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 19–34, 2006.

[43] K. J. Åström and B. Wittenmark, *Adaptive Control*. Courier Corporation, 2013.

[44] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro, "Time-varying graphs and dynamic networks," in *Ad-hoc, Mobile, and Wireless Networks*, H. Frey, X. Li, and S. Ruehrup, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 346–359.

[45] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.