IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 7, 2022, accepted February 23, 2022, date of publication March 2, 2022, date of current version March 15, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3156092

HDP Algorithms for Trajectory Tracking and Formation Control of Multi-Agent Systems

ERNESTO F. M. FERREIRA[1], (Student Member, IEEE), JOÃO VIANA DA FONSECA NETO[2], AND RASTKO R. SELMIC[3], (Senior Member, IEEE)
[1]Educational Department, Federal Institute of Maranhão, Açailândia, Maranhão 65930-000, Brazil
[2]Electrical Engineering Department, Federal University of Maranhão, São Luís, Maranhão 65080-805, Brazil
[3]Department of Electrical and Computer Engineering, Concordia University, Montreal, QC H3G 1M8, Canada

Corresponding author: Ernesto F. M. Ferreira (ernesto.ferreira@ifma.edu.br)

This work was supported in part by the Institutional Doctoral Degree Program Abroad (PDSE) of Coordination for the Improvement of Higher Education Personnel (CAPES).

ABSTRACT A heuristic dynamic programming (HDP) algorithm for trajectory tracking and formation control of multi-agent systems (MAS) is presented in this paper. The selected HDP method allows for an online optimal control design. The multi-agent control problem is formulated as a leader-follower, where it is necessary for followers to maintain the assigned formation, while the leader follows the specified trajectory. Developed $\mathcal{QR}$-Solver and RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithms provide a new methodology to obtain the solution of the Hamilton-Jacobi-Bellman (HJB) equation. These proposed solutions are based on $\mathcal{QR}$ factorization to decrease the computational cost and avoid issues associated with numerical stability of conventional least squares (LS) method. The algorithms' performances such as convergence, numerical stability, and computational metrics, were experimentally evaluated for two control systems: aerial altitude control (one degree of freedom) and terrestrial robot (two degrees of freedom).

INDEX TERMS Heuristic dynamic programming, multi-agent systems, leader-followers, robots.

I. INTRODUCTION

To conduct large-scale tasks, such as inspection of electric power transmission lines, monitoring leaks in gas and oil pipelines, country border patrolling, agriculture, and transportation, the use of multiple robots or agents is more feasible and economical than using a single agent [1], [2]. The control tasks are carried out by multiple agents simultaneously, and new control algorithms must support such framework [3].

A multi-agent system (MAS) can be defined as a group of autonomous agents that interact with each other and share the same environment, which is perceived through sensors, where they act by carrying out certain actions [4]. In a multi-agent framework, two or more agents interact and work together to conduct certain tasks or satisfy a set of goals [5]. The scientific research and practical implementations of MAS are focused on development of standards, principles, and models that allow the creation of groups of autonomous or semi-autonomous agents, capable of interacting properly to accomplish common objectives [6].

When working with multiple mobile agents such as drones or robots, it is often necessary to establish a formation between the agents [7]. Formations of agents can be characterized by geometrical patterns to be applied by a multi-agent team. Formations appear in biological systems, such as the well-known V-shaped flight formation used by geese and other large migratory birds that allow longer flight range of a flock of birds compared to an individual bird. Formation control remains one of the most challenging problems in the control of MAS [8], [9].

Formation control of multiple agents is a challenging task due to environmental disturbances such as wind and various obstacles, leading to instabilities [10], [11]. There is a need to develop an online control method that provides optimal control solutions for the dynamic behavior of agents without full knowledge of the system parameters [12], [13].

A promising solution to solve the online control problem with plant uncertainties is the adaptive dynamic programming (ADP) method [14], [15]. This approach employs

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Tang.

a 'forward-in-time' mechanism that looks for an optimal control policy by successively adapting two parametric structures, i.e., an action network and a critic network, to approximate the solution of the Hamilton-Jacobi-Bellman (HJB) equation [16], [17].

In ADP, the action network calculates the control actions, whereas the critic network learns to approximate the value function. This function evaluates the effect of control on the future performance and provides guidelines on how to improve the control law. These structures are combined with reinforcement learning (RL), which has been used as a framework for solving real-time optimal decision problems [18], [19].

In MAS, the problem of trajectory tracking and formation control can be treated in isolation, and in the proposed control system, the trajectory and formation are modeled in a single dynamic system, where the trajectory reference is defined by the leader (leader-follower), and the formation reference is established according to the specific application.

The algorithm proposed in this work is intended for execution in microcontrollers that are present in the agents/robots of the MAS. The main objective of this work is to design a multi-agent control system using an HDP architecture for a real-time implementation, that guarantees the convergence and stability of the closed-loop system, while providing a disturbance rejection.

This study presents two ADP algorithms for multi-agent systems: the $\mathcal{QR}$-solver (developed in this research) and the RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR (initially developed for single-agent systems), as described in [20], [21]. We compare their performances, such as convergence of the gain, by analyzing the parameter θ of the DLQR and convergence of the trajectory and desired formation. The contribution of this study is the development of optimal control algorithms for MAS, focusing on embedded systems (terrestrial robots and aerial robots) with specifications that guarantee numerical stability, fast convergence time, and low memory use and computational cost, thus ensuring applicability in the real-time control.

The main advantage of using the optimal control approach over classical approaches to MAS, such as a consensus, is the ability to provide a solution for time-varying systems, where the dynamics of the agent and the environment can vary due to disturbances or noises [22], [23]. The optimal control approach also allows the optimization of a desired performance metric by minimizing the specific cost function.

This paper is organized as follows: In Section II, the problem of trajectory tracking and formation (distributed forms that agents/robots should take in space) for MAS is described. In sequence, the optimal solution for trajectory control and MAS formation is then demonstrated. In Section III, the two control algorithms used in this study, i.e. the $\mathcal{QR}$-Solver and the RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR, are described and a brief of Computational Complexity is introduced. Section IV contains computational simulations of these algorithms. Finally, some concluding remarks regarding the results of this work are presented.

II. PRELIMINARIES

Here we present preliminaries related to the trajectory tracking and formation control problem for MAS. We use a state-space model for a subsystem referred to as the leader, and an exosystem referred to as the followers. In sequence, the optimal control solution is demonstrated by minimizing a value function for the trajectory tracking and formation control of MAS.

The formation control problem can be defined by a shape or a relative state. The first component of MAS dictates the position of the leader, and the second component of MAS corresponds to the overall formation (followers) [24]. The dynamics of a MAS is represented by three sets of differential and algebraic equations that correspond to the leader, followers, and the formation error model.

A. LEADER MODEL

The leader agent model is given in the state-space form

$$\dot{v} = Ev + Hw, \tag{1}$$

where $v \in \mathbb{R}^q$ is a state vector of the leader, and $E \in \mathbb{R}^{q\times q}$ and $H \in \mathbb{R}^{q\times q}$ are the state and input matrices of the leader, respectively. The agent input $w \in \mathbb{R}^q$ is defined as $w = \hat{w}+\tilde{w}$, where $\hat{w}$ is a feedback control input $\hat{w} = -K_l(v - v^*)$ with the leader's desired trajectory v^* that satisfies $\dot{v}^* = Ev^*$, $\tilde{w}$ is the external disturbance input (such as a wind gust for aerial agents and slip for terrestrial agents), and K_l is the gain of the leader.

Assumption 1: The disturbance $\tilde{w}$ is upper bounded by W such that $||\tilde{w}(t)|| \leq W$.

Remark 1: The previous assumption specifies the upper bound on the disturbance and is a standard assumption in the literature [25], [26]. The assumption is not very restrictive, as most disturbances in practice have an upper bound, such as wind gust, friction, slip, and other unmodelled disturbances.

B. FOLLOWERS-AGENTS MODEL

The dynamics of the followers is given by

$$\dot{x}_i = A_i x_i + B_i u_i + D_i v, \tag{2}$$

where $x_i \in \mathbb{R}^{n_i}$ is the state of the agent i, A_i, B_i, and D_i are the dynamics matrices of the agent i, $u_i \in \mathbb{R}$ is the control signal, and v is the state of the leader, for $i = 1, 2, \ldots, N$. The formation error $e_i \in \mathbb{R}$ is given by

$$e_i = C_i x_i + F_i v \quad i = 1, 2, \ldots, N, \tag{3}$$

where C_i is the control matrix of the agents and F_i is the formation matrix. The output of each agent is $y_i = C_i x_i \in \mathbb{R}^{n_i}$, and $y_i^* = F_i v$ is the reference or expected trajectory for each agent.

C. CONTROL OF AN AGENT

A decentralized, closed-loop controller, proposed in [27], that is used in this paper, is given by

$$u_i = -K_{xi} x_i - K_{zi} z_i, \tag{4}$$

where $K_i = [K_{xi} \;\; K_{zi}]$ is the gain of the i^{th} agent, $K_{xi} \in \mathbb{R}^{q\times q}$, $K_{zi} \in \mathbb{R}^{q\times q}$. An internal model $z_i \in \mathbb{R}^q$ of the i^{th} follower is given by

$$\dot{z}_i = G_1 z_i + G_2 e_i, \quad i = 1, 2, \ldots, N, \tag{5}$$

where the characteristic polynomial of G_1 is the same as the minimal polynomial of E, and the pair (G_1, G_2) is controllable and incorporates an internal model of the matrix E.

D. OPTIMAL PROBLEM FORMULATION

To control the multi-agent system, we developed an optimal controller that ensures stability of the system under disturbances. Moreover, as $v \equiv v^*$, the developed controller is optimal in a sense that it minimizes the cost function given by

$$J = \int_0^\infty \left(|\xi_i|_Q + |u_i|_R\right) dt, \tag{6}$$

where $\xi_i = [x_i \;\; z_i] \in \mathbb{R}^{2n_i}$ is the i^{th} subsystem. The optimal control gains are $\left[K_{xi}^* \;\; K_{zi}^*\right] = R_i^{-1}\bar{B}_i^T P_i^*$, where P_i^* is the unique solution to the algebraic Riccati equation (ARE)

$$\bar{A}_i^T P_i^* + P_i^* \bar{A}_i + Q_i - P_i^* \bar{B}_i R_i^{-1} \bar{B}_i^T P_i^* = 0. \tag{7}$$

The trajectory tracking and formation control methodologies are designed for a data-driven distributed controller using an ADP, under a switching network topology. The developed approach can approximate the control gains K_i^* for each follower without relying on the knowledge of the system matrices A_i, B_i, and D_i. The internal model (5) is modified and is given by

$$\dot{z}_i = G_1 z_i + G_2 \hat{e}_i, \quad i = 1, 2, \ldots, N, \tag{8}$$

where $\hat{e}_i = y_i + F_i \zeta_i$. The dynamics of $\zeta_i \in \mathbb{R}^q$ is given by

$$\dot{\zeta}_i = E\zeta_i + \sum a_{ij}(t)(\zeta_j - \zeta_i), \quad i = 1, 2, \ldots, N. \tag{9}$$

with $\zeta_0 = v$. The i^{th} subsystem augmenting ξ_i with the internal model of (8) is

$$\dot{\xi}_i = \bar{A}_i \xi_i + \bar{B}_i u_i + \bar{D}\psi_i, \tag{10}$$

for $i = 1, 2, \ldots, N$, where

$$\bar{A}_i = \begin{bmatrix} A_i & 0 \\ G_2 C_i & G_1 \end{bmatrix}, \quad \bar{B}_i = \begin{bmatrix} B_i \\ 0 \end{bmatrix},$$
$$\bar{D}_i = \begin{bmatrix} D_i & 0 \\ 0 & G_2 F_i \end{bmatrix},$$

and $\psi_i = \begin{bmatrix} v & \zeta \end{bmatrix} \in \mathbb{R}^{2q}$.

E. OPTIMAL SOLUTION FOR TRAJECTORY AND FORMATION CONTROL

Using the Iterative Technique for Riccati equation computations of [28], one can determine P_i^* and K_i^* by

$$\begin{aligned} &|\xi_i(t+\delta t)|_{P_i^k} - |\xi_i(t)|_{P_i^k} \\ &= \int_t^{t+\delta t} \Big[-|\xi_i|_\Delta + 2\psi_i^T \bar{D}_i^T P_i^k \xi_i \\ &\qquad + 2(u_i + K_i^k \xi_i)^T R_i K_i^{k+1} \xi_i\Big] d\tau, \end{aligned} \tag{11}$$

where $\Delta = Q_i + (K_i^k)^T R_i K_i^k$. Using the Kronecker product, the cost matrices of Eq. (11) imply that

$$\Phi_i^k \begin{bmatrix} P_i^k \\ K_i^{k+1} \\ D_i^T P_i^k \end{bmatrix} = y_i^k, \tag{12}$$

where

$$\Phi_i^k = [\delta_{\xi_i}, -2\Gamma_{\xi_i\xi_i}(I \otimes (K_i^k)^T R_i) - 2\Gamma_{\xi_i u_i}(I \otimes R_i), 2\Gamma_{\xi_i \psi_i}] \tag{13}$$

$$y_i^k = -\Gamma_{\xi_i\xi_i}(Q_i + (K_i^k)^T R_i K_i^k). \tag{14}$$

Let us define Γ and δ as

$$\begin{aligned}\Gamma_{a,b} = \Bigg[&\int_{t_0}^{t1} a \otimes b\, d\tau, \int_{t_1}^{t2} a \otimes b\, d\tau, \ldots, \\ &\int_{t_{f-1}}^{tf} a \otimes b\, d\tau\Bigg]\end{aligned} \tag{15}$$

$$\begin{aligned}\delta_a = [&kron(a(t_1)) - kron(a(t_0)), \ldots, \\ &kron(a(t_f)) - kron(a(t_{f-1}))],\end{aligned} \tag{16}$$

where $kron(a) = [a_i^2, a_1 a_2, \ldots, a_1 a_m, a_2^2, a_2 a_3, \ldots, a_{m-1} a_m, a_m^2]^T \in \mathbb{R}^{\frac{1}{2}m(m+1)}$.

Equation (12) provides an update of the optimal gain K, as well as the parameter P of the Riccati equation. The complete closed-loop dynamics for a MAS is given by

$$\begin{bmatrix} \dot{\xi} \\ \dot{\zeta} \end{bmatrix} = \bar{A}_c \begin{bmatrix} \xi \\ \zeta \end{bmatrix} + \bar{B}_c\left(\mathbf{1}_N \otimes v\right), \tag{17}$$

where $\bar{A}_c = \mathbf{I}_N \otimes \left[\bar{A}_i - \bar{B}_i K_i\right]$ and $\bar{B}_c = \mathbf{I}_N \otimes \left[\mathbf{0} \;\; F_N^T G_2^T\right]^T$.

The error of the close-loop system is given by

$$e = \bar{C}\xi + F\left(\mathbf{1}_N \otimes v\right). \tag{18}$$

More details and the proof of convergence of the LFS method can be found in [27].

III. ALGORITHMS FOR TRAJECTORY AND FORMATION CONTROL

In this section, two algorithms for trajectory tracking and formation control of multi-agent systems are presented. The QR-solver algorithm is based on the LS approach, and the RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithm uses a recursive least square (RLS). The restrictions and metrics associated with the analysis of computational complexity in the implementation of control algorithms are also presented.

A. $\mathcal{QR}$-SOLVER ALGORITHM

$\mathcal{QR}$-factorization has an important application in solving the LS problems [29], [30]. Often, a matrix is poorly conditioned, i.e., the matrix is extremely sensitive to errors owing to approximations that occur in numerical computations [31]. The main advantage of $\mathcal{QR}$ factorization is that there is no need to compute the inverse of variance or covariance matrices. Instead, it only needed to find an inverse of R, transpose Q, and the product is the LS coefficients. The $\mathcal{QR}$

factorization method is a stable elementary operation that can be applied to any type of matrix. For real-time applications, poor conditioning of the variance or covariance matrices of LS can lead to the agent/robot instabilities, causing a malfunction of the overall MAS. To overcome this problem, an algorithm, called $\mathcal{QR}$-solver, was developed using the Householder transformation to calculate the values of $\mathcal{QR}$ matrices in order to find the least squares solution [32]. The $\mathcal{QR}$-solver is described in the next paragraph.

Every non-singular $\Phi_i{}^k \in \mathbb{R}^{n\times n}$ has a $\mathcal{QR}$ factorization, which is given by

$$\Phi_i^k = \mathcal{Q}_i^k \mathcal{R}_i^k, \tag{19}$$

where $\mathcal{Q}_i^k \in \mathbb{R}^{n\times n}$ is an orthogonal matrix and $\mathcal{R}_i^k \in \mathbb{R}^{n\times n}$ is an upper triangular matrix with positive diagonal elements. The inverse is given by

$$(\Phi_i{}^k)^{-1} = (\mathcal{Q}_i^k \mathcal{R}_i^k)^{-1} = (\mathcal{R}_i^k)^{-1}(\mathcal{Q}_i^k)^T. \tag{20}$$

The algebraic manipulation of (20) was applied to the solution of (12), and with a non-singular $\Phi_i^k \in \mathbb{R}^{n\times n}$, one has

$$\mathcal{R}_i^k \begin{bmatrix} P_i^k \\ K_i^k \\ D_i^T P_i^k \end{bmatrix} = \mathcal{Q}_i^T y_i, \tag{21}$$

where $\mathcal{R}_i^k$ is an upper triangular matrix whose inverse can be easily be found without computing the determinant.

The $\mathcal{QR}$ factorization procedure for computing P_i^k (P of Riccati), K_i^k (gain of i^{th} follower), and $D_i^T P_i$ parameters of Eq. (21) is presented in Algorithm 1 (called $\mathcal{QR}$-solver).

Algorithm 1 proposes a computational structure with three main functional blocks. The first block is associated with **steps 1-5**, which is the setup of the algorithm with initialization of all system dynamics matrices, initial positions and velocity, gain matrices, simulation parameters, and finally state vectors. The second block establishes the formation topology in **step 8**. The data acquisition of the system is carried out in **steps 9-12**, and the following steps are based on the corresponding equations: **step 9** applies Eq. (4), **step 10** uses Eq. (17), **step 11** applies Eq. (1) and **step 12** uses Eq. (17). The control update of the agent i, represented by **steps 16-19**, is the third function block. The calculation of Φ_i in Eq. (13) is carried out in **step 16**, the value of y_i in **step 17** is calculated using Eq. (14), **step 18** refers to the LS solution of Eq. (12), and the new K_i value is obtained in **step 19**. We propose improvements in the numerical stability of the algorithm without compromising the convergence time.

Algorithm 1 $\mathcal{QR}$-Solver

```
    Setup (Initial Conditions)
 1: Dynamic System Matrices: $A_i, B_i, C_i, D_i, E, F_i, G_1, G_2, H, \mathcal{H}$.
 2: Initial State: $x_{i0}, \zeta_0, v_0, v_0^*$.
 3: Initial Values of $P, Q, R$ and $K$: $P_0, K_i, K_0$.
 4: Number of Iterations: $N$; Sample time: $t_s$; Noise: $\omega$.
 5: Vectors $\xi_i = [x_i^T\ z_i^T]^T$ and $\psi_i = [v^T\ \zeta_i^T]^T$

    Iterative Process
 6: for $k \leftarrow 1 : N$
 7:   do
        Switch Signal
 8:     $\mathcal{H}^k \leftarrow$ chosen graph ($\mathcal{G}$)
        Environment Simulation
 9:     $\dot{u}_i^k \leftarrow K_i x_i^k + \omega$
10:     $\dot{\xi}_i^k \leftarrow \bar{A}_i x_i^k + \bar{B}_i u_i^k + \bar{D}_i \psi_i^k$
11:     $\dot{v}^k \leftarrow E v^k + H(-K_0(v^k - v^*) + \tilde{v})$
12:     $\dot{\zeta}_i^k \leftarrow [(I_N \otimes E) - (\mathcal{H}^k \otimes I_q)]\zeta_i + [\mathcal{H}^k \otimes I_q](1_n \otimes v)$
        Calculation of $\delta$ and $\Gamma$ to ($t_0$ at $t_s$)
13: end

    Control Update (for each agent $i$)
14: for $k \leftarrow 1 : n_{int}$
15:   do
        Calculation of matrix $\Phi_i$ and vector $\Phi_i$
16:     $\Phi_i^k \leftarrow [\delta_{\xi_1}, -2\Gamma_{\xi_1,\xi_1}(I \otimes (K_i^k)^T R_i) - 2\Gamma_{\xi_1,u_1}(I \otimes R_i), -2\Gamma_{\xi_1,\psi_1}]$
17:     $y_i^k \leftarrow \Gamma_{\xi_1,\xi_1} vec(Q_i + (K_i^k)^T R_i K_i^k)$
        Calculation of new gain K using LS
18:     $\mathcal{Q}_i^k \mathcal{R}_i^k \leftarrow \Phi_i^k$
19:     $\mathcal{R}_i^k \left[P_i^k; K_i^{k+1}; D_i^T P_i^k\right] \leftarrow (\mathcal{Q}_i^k)^T y_i^k$
20: end

    Repeat the use of $K_i^{(k+1)}$
```

B. RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR ALGORITHM

According to the standard LS estimator, the optimal value of the parameter vector $\hat{\theta}_i$ is given by

$$\Phi_i \hat{\theta}_i = y_i, \tag{22}$$

where $\hat{\theta}_i = \left[P_i^k; K_i^{k+1}; \bar{D}_i^T P_i^k\right]^T$. Let Φ_i be expressed in its factored form, which is given by

$$\Phi_i = \Phi_i^{1/2} \Phi_i^{H/2}, \tag{23}$$

where $\Phi_i^{1/2}$ is a lower triangular matrix defined as the square root of Φ_i, and $\Phi_i^{H/2}$ is the Hermitian transpose of $\Phi_i^{1/2}$.

Note, from the RLS$_\mu$-HDP-DLQR algorithm, that the recursive updating equation of the correlation matrix Φ_i is given by the following:

$$\Phi_i = \mu \Phi_{i-1} + \bar{x}_i \bar{x}_i^H. \tag{24}$$

Pre-multiplying both sides of Eq. (23) by matrix $\Phi_i^{-1/2}$, a new vector variable is given by

$$\omega_i = \Phi_i^{H/2} \hat{\theta}_i = \Phi_i^{-1/2} z_i. \tag{25}$$

We obtain here a new form to represent the cross-correlation vector between the input $\bar{x}_i$ and the expected output d_i, which is given by

$$y_i = \mu z_{i-1} + \bar{x}_i d_i^*, \tag{26}$$

or equivalently

$$\Phi_i \hat{\theta}_i = \mu \Phi_{i-1} \hat{\theta}_{i-1} + \bar{x}_i d_i^*, \tag{27}$$

where the asterisk denotes a complex conjugate.

Eqs. (24), (27) are described in their Hermitian transposed forms, expressing each of the terms that appear in the second member of each equation in its transposed forms as:

$$\mu\Phi_{i-1}^{H} = (\mu^{1/2}\Phi_{i-1}^{1/2}).(\mu^{1/2}\Phi_{i-1}^{H/2}), \tag{28}$$

$$\begin{aligned}\mu\hat{\theta}_{i-1}^{H}\Phi_{i-1}^{H} &= (\mu^{1/2}\hat{\theta}_{i-1}^{H}\Phi_{i-1}^{1/2}).(\mu^{1/2}\Phi_{i-1}^{H/2})\\ &= (\mu^{1/2}\omega_{i-1}^{H})(\mu^{1/2}\Phi_{i-1}^{H/2}).\end{aligned} \tag{29}$$

Suppose one chooses a unitary rotation Θ_i transforming this pre-array to produce a zero block in the second entry of the post-array top-row block, as shown by the following form:

$$\mathcal{A}\Theta = \mathcal{B}, \tag{30}$$

where

$$\mathcal{A} = \begin{bmatrix} \mu^{1/2}\Phi_{i-1}^{1/2} & \bar{x}_i \\ \mu^{1/2}\omega_{i-1}^{H} & d_i \\ 0^T & 1 \end{bmatrix} \text{ and } \mathcal{B} = \begin{bmatrix} \mathcal{B}_{11i}^{H} & 0 \\ b_{21i}^{H} & b_{22i}^{*} \\ b_{31i}^{H} & b_{32i}^{*} \end{bmatrix}. \tag{31}$$

To evaluate the elements of the unknown blocks $\mathcal{B}_{11i}^{H}$, b_{21i}^{H}, b_{22i}^{*}, b_{31i}^{H}, and b_{32i}^{*} of the post-array $\mathcal{B}$, the procedure is to square both sides of Eq. (30). According to the matrix factorization lemma, [31], Θ_i is recognized as a unitary matrix; therefore, $\Theta_i\Theta_i^{H}$ is equal to the identity matrix for every i. Thus, Eq. (30) can be transformed into

$$\mathcal{A}\mathcal{A}^{H} = \mathcal{B}\mathcal{B}^{H}, \tag{32}$$

or in an expanded form

$$\begin{aligned}&\begin{bmatrix} \mu^{1/2}\Phi_{i-1}^{1/2} & \bar{x}_i \\ \mu^{1/2}\omega_{i-1}^{H} & d_i \\ 0^T & 1 \end{bmatrix}\begin{bmatrix} \mu^{1/2}\Phi_{i-1}^{H/2} & \mu^{1/2}\omega_{i-1} & 0 \\ \bar{x}_i^{H} & d_i^{*} & 1 \end{bmatrix}\\ &= \begin{bmatrix} \mathcal{B}_{11i}^{H} & 0 \\ b_{21i}^{H} & b_{22i}^{*} \\ b_{31i}^{H} & b_{32i}^{*} \end{bmatrix}\begin{bmatrix} \mathcal{B}_{11i} & b_{21i} & b_{31i} \\ 0^T & b_{22i} & b_{32i} \end{bmatrix}.\end{aligned} \tag{33}$$

Comparing the respective terms on both sides of Eq. (33), the unknown block elements of a post-array $\mathcal{B}$ are determined. Consequently, Eq. (30) with the values of $\mathcal{B}_{11i}^{H}$, b_{21i}^{H}, b_{22i}^{*}, b_{31i}^{H}, and b_{32i}^{*} is expressed as follows:

$$\begin{bmatrix} \mu^{1/2}\Phi_{i-1}^{1/2} & \bar{x}_i \\ \mu^{1/2}\omega_{i-1}^{H} & d_i \\ 0^T & 1 \end{bmatrix}\Theta_i = \begin{bmatrix} \Phi_{i}^{1/2} & 0 \\ \omega_{i}^{H} & \xi_i\rho_i^{1/2} \\ \bar{x}_i^{H}\Phi_i^{-H/2} & \rho_i^{1/2} \end{bmatrix}, \tag{34}$$

where ξ_i is the *a priori* estimation error given by $\xi_i = d_i - \theta_{i-1}^{H}\bar{x}_i$, and ρ_i is a real parameter [33]. The equations in this subsection form the Algorithm 2, labeled RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR.

Algorithm 2 has three main functional blocks. The first block is associated with **steps 1-6**, which sets up and initializes the algorithm, i.e., all system dynamic matrices, the initial positions and velocity, the gain matrices, the simulation parameters, the state vectors, and the forgetting factor of the RLS. The second block establishes the formation topology in **step 9**, and the data acquisition of the system that is represented by the model in **steps 10-13**. The following steps of this block are based on the following equations: **step 10** applies Eq. (4), **step 11** uses Eq. (17), **step 12** utilizes Eq. (1), and **step 13** applies Eq. (17). The control update of the agent i, represented by **steps 15-22**, is the third function block. In **step 15**, the vector $\bar{x}_i$ is assembled using by Kronecker product of the state vector x_i. In **step 16**, the target vector of the ADP is formed. In **step 17**, the matrix A_i is assembled according to Eq. (31). In **step 18**, the $\mathcal{QR}$ factorization is applied, generating the matrices $\mathcal{Q}_i$ and $\mathcal{R}_i$

Algorithm 2 RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR

```
    Setup (Initial Conditions)
 1: Dynamic System Matrices: $A_i, B_i, C_i, D_i, E, F_i, G_1, G_2, H, \mathcal{H}$.
 2: Initial State: $x_{i0}, \zeta_0, v_0, v_0^*$.
 3: Initial values of $P, Q, R$ and $K$: $P_0, K_i, K_0$.
 4: Number of Iterations: $N$; Sample time: $t_s$; Noise: $\omega$.
 5: Vectors $\xi_i = [x_i^T\, z_i^T]^T$ and $\psi_i = [v^T\, \zeta_i^T]^T$
 6: Forgetting Factor: $0 < \mu \le 1$.

    Iterative Process
 7: for $k \leftarrow 1 : N$
 8:   do    Switch Signal
 9:      $\mathcal{H}^k \leftarrow$ chosen graph ($\mathcal{G}$)
         Environment Simulation
10:      $\dot{u}_i \leftarrow K_i x_i + \omega$
11:      $\dot{\xi}_i \leftarrow \bar{A}_i x_i + \bar{B}_i u_i + \bar{D}_i \psi_i$
12:      $\dot{v} \leftarrow Ev + H(-K_0(v - v^*) + \tilde{v})$
13:      $\dot{\zeta}_i \leftarrow [(I_N \otimes E) - (\mathcal{H}^k \otimes I_q)]\zeta_i + [\mathcal{H}^k \otimes I_q](1_n \otimes v)$
         Calculation of $\delta$ and $\Gamma$ for ($t_0$ at $t_s$)
14: end

    Control Update (For each agent $i$)
    Basis Set - Kronecker Product
15: $\bar{x}_i \leftarrow [x_{1i}^2; x_{1i}x_{2i}; \dots; x_{4i}^2]$
    Target Assembly
16: $d(x,c,f,P) \leftarrow x_i^T Q x_i + u_i^T R u_i + \gamma x_{i+1}^T P x_{i+1}$
    $\mathcal{QR}$ decomposition-based RLS
17: $\mathcal{A}_i \leftarrow \begin{bmatrix} \mu^{1/2}\Phi_i^{1/2} & \bar{x}_i \\ \mu^{1/2}\omega_i^H & d(\cdot) \\ 0^T & 1 \end{bmatrix}$
18: $\mathcal{Q}_i\mathcal{R}_i \leftarrow \mathcal{A}_i$
19: $\mathcal{R}_i \rightarrow \begin{bmatrix} \Phi_{i+1}^{1/2} & 0 \\ \omega_{i+1}^H & \xi_i\rho_{i+1}^{1/2} \\ \bar{x}_i^H\Phi_{i+1}^{-H/2} & \rho_{i+1}^{1/2} \end{bmatrix}$
    Vector updating $\hat{\theta}$
20: $\hat{\theta}_{i+1}^H = \omega_{i+1}^H\Phi_{i+1}^{-1/2}$
    Recovery matrix $P$ from vector $\theta$
21: $P^{\hat{\theta}_{i+1}} \leftarrow [\hat{\theta}_1, \quad \hat{\theta}_{2/2}, \dots, \quad \hat{\theta}_{9/2}; \dots, \quad \hat{\theta}_{10}]$
    Feedback Optimal Gain $K$
22: $K_{i+1} \leftarrow -\gamma(R + \gamma B_d^T P^{\hat{\theta}_{i+1}} B_d)^{-1} B_d^T P^{\hat{\theta}_{i+1}} A_d$

23: End - Iterative Process
```

according to Eq.(34). In **step 19**, the values of Φ and ω are defined, and the calculation of $\hat{\theta}_i$ in **step 20** is applied using Eq.(25). In **step 21**, the matrix $P^{\hat{\theta}}$ is formed with the elements of the vector $\hat{\theta}_i$, and **step 22** applies the update of the K_i gain with the solution of HJB-Riccati equation.

C. COMPUTATIONAL COMPLEXITY

The amount of resources required to run an algorithm is closely related to computational complexity [34]. The complexity analysis, for applications with embedded systems (robots) in real-time, can be performed using metrics that are minimized and/or achieved. To evaluate the performance of the proposed algorithm, computational complexity metrics are established in terms of the numerical stability issues including ill-conditioned time vectors, the quantity of the operations, the storage scaling, and the dynamic system critical time.

1) NUMERICAL STABILITY

The numerical stability analyzes how errors introduced during the execution of an algorithm affect the result. It is a property of an algorithm rather than the problem being solved [35]. In computing, numerical stability refers to how a malformed input affects the execution of an algorithm. In a numerically stable algorithm, errors in the input lessen in significance as the algorithm executes, having little effect on the final output [36].

Some numerical algorithms may damp out the small fluctuations (errors) in the input data while others might amplify such errors. Calculations that are shown to attenuate approximation errors are called numerically stable. One of the common tasks of numerical analysis is to try to select algorithms which are robust, i.e., do not produce significantly different results for a very small change in the input data. In terms of numerical stability, the main concern is addressed by applying strategies to avoid ill-conditioned time vectors.

2) COMPUTATIONAL COST

The computational cost is the number of floating point operations (FLOPs) needed to execute a computational command, which can be a mathematical operation (e.g., addition, subtraction, or multiplication) or writing to, or reading from, the memory.

3) DATA STORAGE

The LS demands memory storage, which depends on the time size batch mode data processing. The same is true for the RLS processing mode, but in an amount that depends on the quantity output variable measurement and the regressors.

4) DYNAMIC SYSTEM CRITICAL TIME

The critical time of the dynamic system is the overall limitation to establish an online optimal decision, i.e., in our context, a constraint that the optimal controller (CPU time) must provide an action response faster than the time constant of the controlled dynamic system.

To ensure the functionality of the robotic embedded system, the response time (control output) of the control algorithm must be less than the dynamic system critical time. If this condition is not met, the control methodology must provide some response in time, i.e. an alternative response as the previous gain or a sub-optimal gain will be used to maintain the multi-agent system running.

IV. COMPUTATIONAL EXPERIMENTS

In this section we present the computational results of the algorithms and evaluate capability of the agents to follow the leader while maintaining the desired formation. With focus on performance for online control methodology in multi-agent systems, the $\mathcal{QR}$-solver and RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithms are evaluated in relation to convergence of parameter θ and the evolution of the control process. Furthermore, the reference policy is established using the offline Schur solution to the HJB-Riccati equation. The evaluations aim to verify the convergence behavior of the estimators.

Computational tests were carried out using MATLAB® and comparisons were performed to evaluate the accuracy and convergence of the solutions provided by the algorithms. To demonstrate the effect of the forgetting factor μ, several simulations were conducted. The convergence process of RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR for the HJB-Riccati equation solutions for different μ values was verified in [37].

We consider a multi-agent system consisting of three agents: the leader and two followers. The connectivity topology is shown in Figure 1.

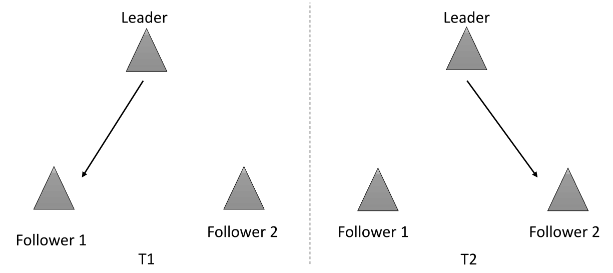

FIGURE 1. Topology of multi-agent system.

Based on Figure 1, the Laplacian matrices for multi-agent systems $T1$ and $T2$ are given by

$$L_{T1} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad L_{T2} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}.$$

A. DRONE MODEL IN *Z*-AXIS AND NUMERICAL SIMULATION

The first model is the altitude control of the mini-drone Parrot®. Considering only the quad-rotor altitude control, the equation that governs the dynamics of this system is given by [38]

$$F_z = F_c - F_g, \tag{35}$$

where F_z is the force applied along the Z-axis, F_c is the force of drone propellers, and F_g is the force of gravity on the drone. This is equivalent to

$$\frac{d^2z}{dt^2} = \ddot{z} = \frac{F_c}{m} - g, \tag{36}$$

for a multi-rotor moving only along the Z-axis, without any angular motion in which the four motors exert the same force, resulting in $F_c = 4k_f\omega^2$. The angular velocity of the blades is $\omega = 22000rpm$, the power constant is $k_f = 3.1 \times 10^{-8}$, mass is $m = 0.08kg$, and the gravity constant is $g = 9.81m/s^2$.

The matrices of the dynamic system are as follows:

$$A_i = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad B_i = \begin{bmatrix} \frac{F_c}{m} \\ 0 \end{bmatrix}, \tag{37}$$

where the states are the speed of the rotors (V_z) and the vertical position of the quad-rotor (P_z).

The simulation experiments are presented for the mini-drone Parrot® and the functionality of the $\mathcal{QR}$ decomposition is verified on the RLS estimator of the approximated state-value function for the online HDP-DLQR control system, as well as on the linear, least-square solution in $\mathcal{QR}$-solver. The numerical stability, trajectory, and formation convergence of the $\mathcal{QR}$-solver and RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithms are analyzed using a system with three agents, i.e., the leader and two followers.

The desired trajectory of the leader and the followers are shown in Figure 2, where the former starts from point **A** (2m high) to point **B** (3m high) and returns to the starting point. The followers execute the same form of trajectory following the formation points ($\mathbf{A}_1, \mathbf{B}_1$) for Follower 1 and ($\mathbf{A}_2, \mathbf{B}_2$) for Follower 2.

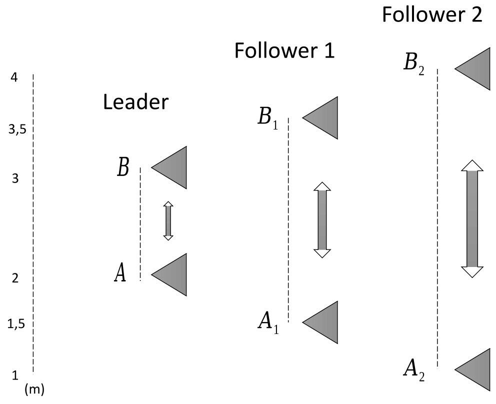

FIGURE 2. Formation and trajectory of leader and followers in Z-axis.

1) $\mathcal{QR}$-SOLVER EVALUATION

The control update using the $\mathcal{QR}$-solver algorithm is shown in Figure 3, for a cycle of 10 iterations. Figure 3 shows the convergence of p_{11}, p_{22}, p_{33}, and p_{44} of matrix P corresponding to components $\theta_1, \theta_5, \theta_8$, and θ_{10} of the parameter vector θ for Follower 1, respectively.

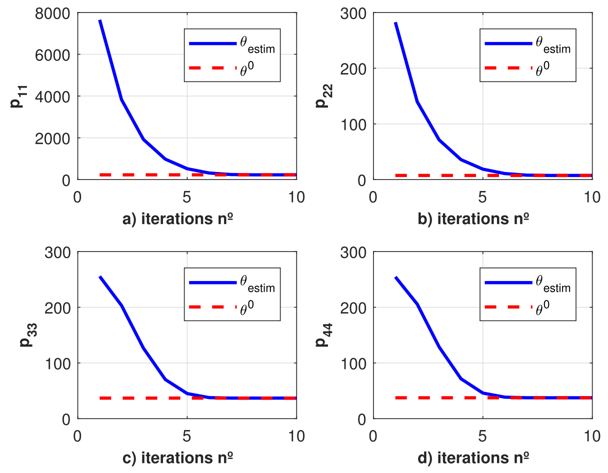

FIGURE 3. Evolution of the iterative process for the parameters p_{11}, p_{22}, p_{33}, and p_{44} for a cycle of 10 iterations, in the $\mathcal{QR}$-solver algorithm for Follower1.

The parameter P_{estim} reaches the optimal solution P^0 (Schur solution) after approximately 10 iterations of the control update. The evolution of the iterative process for the $\mathcal{QR}$-solver algorithm is shown in Figure 4 for a 10s cycle of the control process.

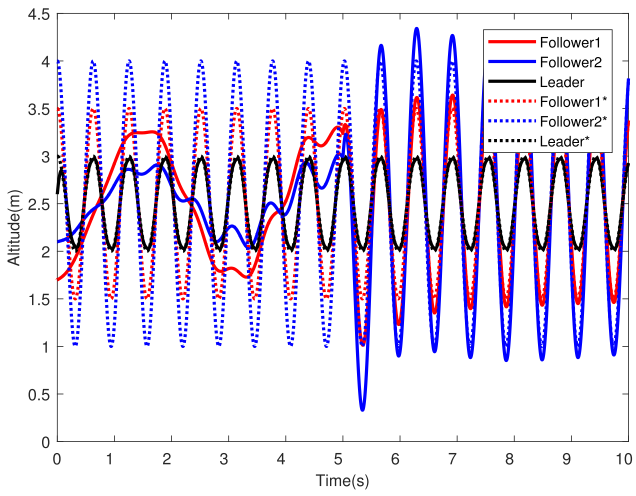

FIGURE 4. Evolution of the control process for position of the agents along the Z-axis during a 10s period, on the $\mathcal{QR}$-solver algorithm.

Figure 4 shows the trajectory of the elements (leader, Follower 1, and Follower 2) and the desired trajectory (leader*, Follower 1*, and Follower 2*) of the multi-agent system during the iterative process.

The leader moves from position **A** (2m) to position **B** (3m) several times, changing its altitude (Z-axis). The followers maintain the altitude distance with respect the flight formation plan shown in Figure 2. After the control update (gain update for the optimal solution) is applied within a 5s period, Follower 1 and Follower 2 trajectory tracking is set by the leader, and after $9s$ they reach the desired formation.

The $\mathcal{QR}$-solver showed satisfactory results for trajectory tracking and formation control. It required $5s$ to compute the new gain (batch solution), and this time is used to fill the matrices Γ and δ of Eq. (15-16), necessary for the calculation of Φ_i^k and y_i^k in Eq. (12).

2) RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR EVALUATION

The results of the RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR experiments are presented in an analogous form to those of the $\mathcal{QR}$-solver. A cycle of 200 iterations of the control update and $20s$ of the control process (sample time $0.1s$) for Algorithm 2 are shown in Figure 5 and Figure 6. Using the forgetting factor $\mu = 0.9142$, Figure 5 shows the convergence of parameters θ_1, θ_5, θ_8, and θ_{10} of the parameter vector θ for Follower1.

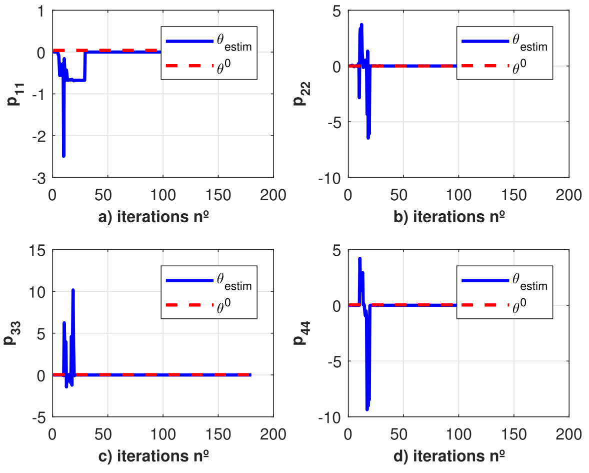

FIGURE 5. Evolution of the iterative process for the parameters p_{11}, p_{22}, p_{33}, and p_{44} for a cycle of 200 iterations, with forgetting factor $\mu = 0.9142$, on the RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithm, of Follower1.

The convergence of the θ parameters (p_{11}, p_{22}, p_{33}, and p_{44}) presents numerical instabilities in the first 50 iterations, after which the solution around iteration 80 is already admissible.

Figure 6 shows the results of the RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithm. Here, Follower 1 and Follower 2 spend more time acquiring the desired formation compared to the previous algorithm, which shows only a simulation within the interval of $10s$ to $20s$. This is a period during which followers track the desired trajectory and formation; the remainder of the simulation is not presented owing to a lack of significant information, and to stress the convergence of the trajectory tracking and formation control.

The RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithm also showed satisfactory results for multi-agent control despite the delay in assembly of the desired formation. As the advantage of this algorithm, if any variation in the process parameters occurs, the adaptability ensures that a new control policy P will be calculated for a controller gain K update (actor/critic architecture).

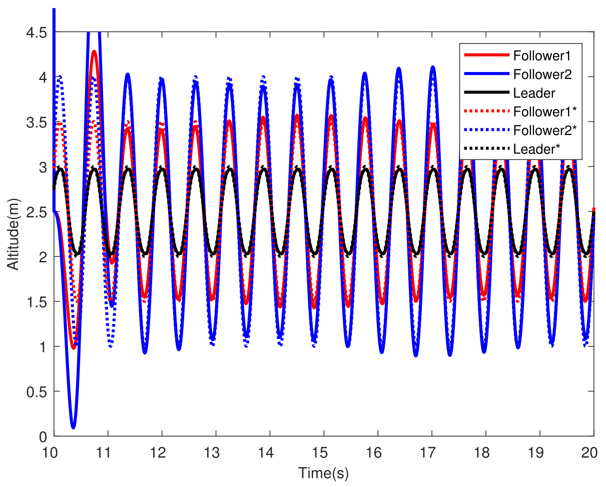

FIGURE 6. Evolution of the control process for position of agent along the Z-axis during a 20 s period, on the RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithm.

3) RESULTS OF THE NUMERICAL SIMULATION

The convergence results of the P matrix for the drone model of Algorithm 1 and Algorithm 2 are presented for the first follower (Follower1), along with the numerical values of Schur's solution in Table 1.

TABLE 1. Numerical values of the P matrix for drone model.

Algorithm	p_{11}	p_{22}	p_{33}	p_{44}
Schur	227.77	7.41	36.88	37.49
$\mathcal{QR}$-Solver	227.76	7.40	36.88	37.48
RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR*	199.43	7.41	35.64	38.01

The convergence values of the matrix P presented in the graphs of Figure 5, are multiplied by 10^{-3}. The computational costs for applying the control update of Algorithm 1 and Algorithm 2 in the drone model along the Z-axis is calculated using [31], and can be seen in Table 2.

TABLE 2. Results of computational cost during Experiment 1.

Algorithm	Cost (Flops)	Gain Update
$\mathcal{QR}$-Solver	1082	One Time
RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR	792	Each Iteration

Analyzing the results of Experiment 1, it can be observed that the computational cost of RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR is lower than that of $\mathcal{QR}$-solver, suggesting that the algorithm is more viable for real-time implementation with microcontrollers. However, this cost is repeated at each iteration of Algorithm 2, resulting in a higher cost in the long run.

B. TERRESTRIAL ROBOT MODEL IN X- AND Y-AXIS

To verify the operation of the control algorithms in models with more degrees of freedom (DoF), the model of a terrestrial robot with two DoF (X-axis and Y-axis) is chosen.

The state-space matrices A and B are given by [39]

$$A_i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \text{ and } B_i = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \tag{38}$$

where the states are the position and velocity along the X-axis (P_x, V_x) and the position and velocity along the Y-axis (P_y,V_y). The simulation results are presented for a terrestrial robot, and the numerical stability, trajectory, and formation convergence of the $\mathcal{QR}$-solver and RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithms were analyzed for a system with three agents – the leader and two followers.

The desired leader and followers trajectories are shown in Figure 7, where the leader trajectory is a circle along the X-axis and Y-axis with a $2m$ diameter. The followers must follow this circular trajectory, while maintaining the formation distance of $0.2m$ between each agent.

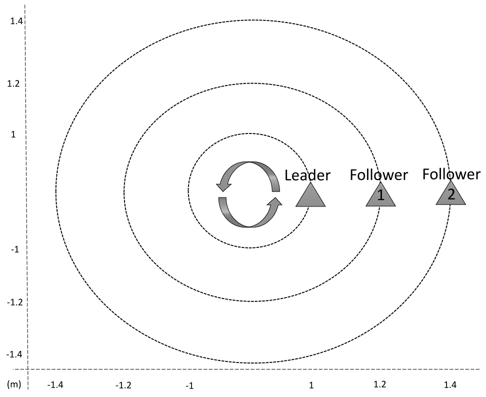

FIGURE 7. Formation and Trajectory of Leader and Followers in X-axis and Y-axis.

1) $\mathcal{QR}$-SOLVER EVALUATION

The evolution of the iterative process for the $\mathcal{QR}$-solver is shown in Figure 8 and Figure 9, for a cycle of 10 iterations of the control update part in Algorithm 1, and $10s$ for the iterative process. Figure 8 (a)-(d) shows the convergence behavior of the elements p_{11}, p_{22}, p_{33}, and p_{44} of matrix P corresponding to the components θ_1, θ_5, θ_8, and θ_{10} of the parameter vector θ of Follower 1, respectively. The parameter P_{estim} reaches the Schur solution P^0 in approximately 7 iterations.

Figure 9 shows the trajectory of the agents during the iterative process, showing the circular motion of the leader. The followers maintain the distance with respect to the desired formation shown in Figure 7.

After the control update has been applied (at $5s$), Follower 1 and Follower 2 tracking the trajectory are defined by the leader, and after $6s$ they reach the desired formation.

The $\mathcal{QR}$-solver algorithm used for the control of terrestrial robots has presented significant results for trajectory and formation control.

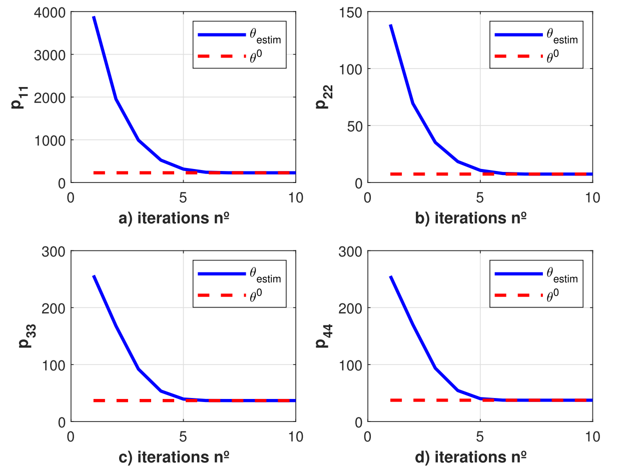

FIGURE 8. Evolution of the iterative process for the parameters p_{11}, p_{22}, p_{33}, and p_{44} for a cycle of 10 iterations, on the $\mathcal{QR}$-solver algorithm of Follower 1.

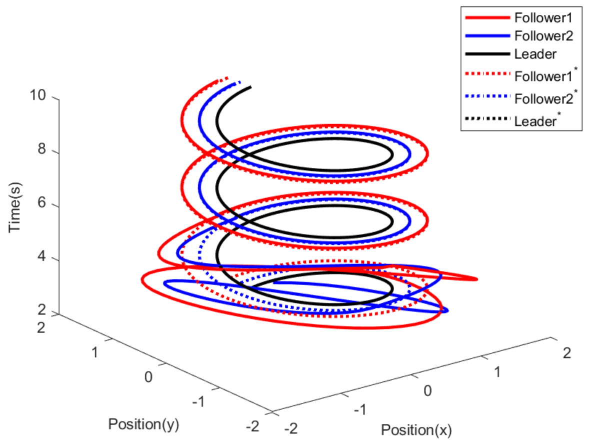

FIGURE 9. Evolution of the control process for the position of the agents along the X-axis and Y-axis for 100s, on the $\mathcal{QR}$-solver algorithm.

2) RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR EVALUATION

A cycle of 200 iterations of the control update of the Algorithm 2 is shown in Figure 10. Using the forgetting factor $\mu = 0.9142$, Figure 10 shows the convergence of θ_1, θ_5, θ_8, and θ_{10} of the parameter vector θ of Follower 1.

Figure 11 presents the results of the RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithm. During $20s$ of the control process, Follower 1 and Follower 2 begin a back and forth movement, and after $5s$, they copy the trajectory of the leader (circular motion). After $10s$, they start to follow the formation, and at close to $15s$, followers track the desired formation and trajectory.

The RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithm also showed good results for convergence of the trajectory and formation in multi-agent control.

3) RESULTS OF NUMERICAL SIMULATION

The convergence results of the P matrix for the terrestrial robot model of Algorithm 1 and Algorithm 2 are presented

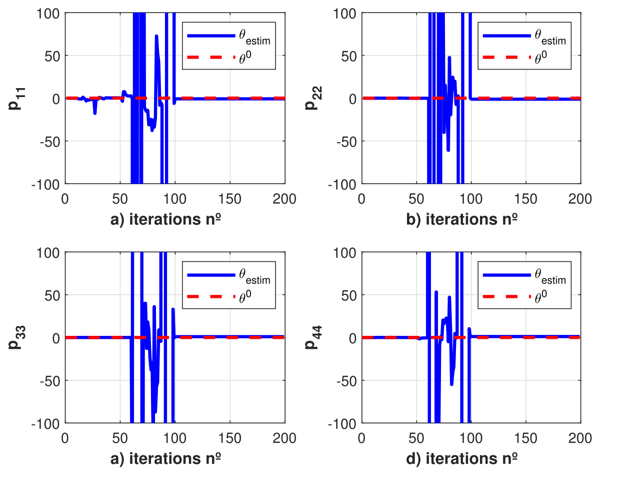

FIGURE 10. Evolution of the iterative process for the parameters p_{11}, p_{22}, p_{33}, and p_{44} for a cycle of 200 iterations, with forgetting factor $\mu = 0.9142$, on RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithm, of Follower 1.

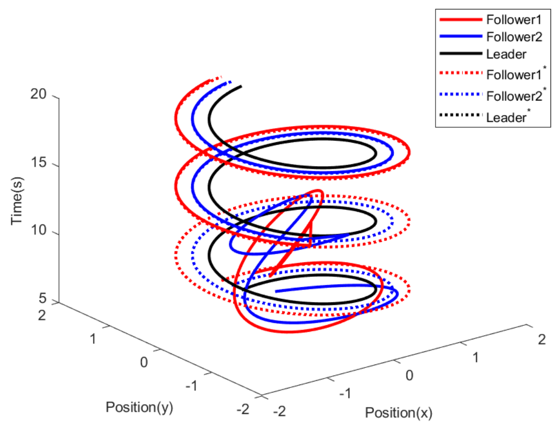

FIGURE 11. Evolution of the control process for position of agents in X-axis and Y-axis during 20 s on RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithm.

TABLE 3. Numerical values of the P matrix for Terrestrial Robot Model.

Algorithm	p_{11}	p_{22}	p_{33}	p_{44}
Schur	260.6	10.32	45.76	43.52
$\mathcal{QR}$-Solver	260.1	10.45	45.75	42.52
RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR*	255.52	9.82	46.86	50.63

TABLE 4. Result of computational cost in Experiment 2.

Algorithm	Cost (Flops)	Gain Update
$\mathcal{QR}$-Solver	2164	One Time
RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR	1584	Each Iteration

for the first follower (Follower 1), along with the numerical values of Schur's solution in Table 3.

The convergence values of matrix P presented in the graphs of Figure 10, are multiplied by 10^{-3}. The computational costs for applying the control update, of Algorithm 1 and Algorithm 2, in terrestrial robot model in X-axis and Y-axis are calculated using [31], and can be seen in Table 4.

Analyzing the results of Experiment 2, it was observed that the computational cost of RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR is lower than that of $\mathcal{QR}$-solve. However, in this experiment, Algorithm 2 required twice as much time as Algorithm 1 to achieve the convergence for the trajectory tracking and formation control.

C. COMPARISON BETWEEN MAS CONTROL ALGORITHMS

To evaluate the merits of the $\mathcal{QR}$-Solver and RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithms for trajectory tracking and formation control of MAS, a comparison is performed among the algorithms presented in this work and an approach based on a consensus algorithm [40]. The algorithms were compared in relation to time needed to reach the desired trajectory and mean absolute error of formation of N agents. The results obtained are shown in Table 5.

TABLE 5. Comparative between algorithms proposed and consensus-based*.

Algorithm	Time for trajectory(s)	Formation error (cm)
$\mathcal{QR}$-Solver	8.2	2.64
RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR	15.6	5.83
Consensus Based	8.6	2.8

*The comparative was carried out using the model in section IV-A.

The formation error of each agent is $E_{(L,1)} = h_{(L,1)} - d_{(L,1)}$, where $h_{(L,1)}$ and $d_{(L,1)}$ are the desired and measured distance between the leader and Follower 1, respectively.

We conclude that Algorithm 1 has a higher cost in applying the control update, generating a higher consumption burden on the CPU and memory resources, whereas Algorithm 2 has a lower cost that occurs on a recurring basis, generating a continuous usage of the CPU and memory resources, leading to greater energy consumption, and reducing the operating time of the agent/robot.

Simulations with a larger number of agents are performed to evaluate the convergence of the control algorithms. The achieved results show convergence of the algorithms $\mathcal{QR}$-Solver and RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR in trajectory tracking and formation control of systems with multiple agents and with any viable formation [41].

For collision avoidance, a well-established technique is to add a barrier function term in the optimal control cost function Eq. (6) (see [42]–[44]), which prevents collisions between agents.

V. CONCLUSION

We presented two algorithms for the trajectory tracking and formation control of multi-agent systems. Based on numerical simulations, the $\mathcal{QR}$-solver and RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithms show satisfactory results for trajectory tracking and formation convergence. The $\mathcal{QR}$-solver required less time to achieve convergence of the estimated parameters because of fewer updates. The algorithm based on HDP-DLQR is adaptable to exogenous perturbations in the plant in terms of the assigned trajectory and formation.

Based on the analysis, focusing on embedding the two algorithms in microcontrollers for real-world applications, the $\mathcal{QR}$-solver algorithm is computationally more stable and faster in terms of convergence of the gain – two essential features for real-time control applications. The RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithm has a superior adaptability, i.e., it is better at overcoming system uncertainties.

The future work will involve mobile robots implementation of $\mathcal{QR}$-solver and RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR algorithms. For this, it is also necessary to add a collision avoidance routine to the presented algorithms, preventing the agents from colliding with any obstacle.

REFERENCES

[1] L. Bian, W. Sun, and T. Sun, "Trajectory following and improved differential evolution solution for rapid forming of UAV formation," *IEEE Access*, vol. 7, pp. 169599–169613, 2019.

[2] S. S. Ge, X. Liu, C.-H. Goh, and L. Xu, "Formation tracking control of multiagents in constrained space," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 3, pp. 992–1003, May 2016.

[3] N. Nascimento, P. Alencar, C. Lucena, and D. Cowan, "A metadata-driven approach for testing self-organizing multiagent systems," *IEEE Access*, vol. 8, pp. 204256–204267, 2020.

[4] G. Weiss, *Multiagent Systems—A Modern Approach to Distributed Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 2000.

[5] Z. R. Bogdanowicz, "Flying swarm of drones over circulant digraph," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 6, pp. 2662–2670, Dec. 2017.

[6] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multagent Networks*. Princeton, NJ, USA: Princeton Univ. Press, 2010.

[7] J. Yu, X. Dong, Q. Li, and Z. Ren, "Practical time-varying formation tracking for second-order nonlinear multiagent systems with multiple leaders using adaptive neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6015–6025, Dec. 2018.

[8] X. Dong and G. Hu, "Time-varying formation tracking for linear multiagent systems with multiple leaders," *IEEE Trans. Autom. Control*, vol. 62, no. 7, pp. 3658–3664, Jul. 2017.

[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] G. Wen, C. L. P. Chen, and Y.-J. Liu, "Formation control with obstacle avoidance for a class of stochastic multiagent systems," *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5847–5855, Jul. 2018.

[11] G. Jing and L. Wang, "Multiagent flocking with angle-based formation shape control," *IEEE Trans. Autom. Control*, vol. 65, no. 2, pp. 817–823, Feb. 2020.

[12] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[13] P. Wang, Z. Man, Z. Cao, J. Zheng, and Y. Zhao, "Dynamics modelling and linear control of quadcopter," *Proc. Int. Conf. Adv. Mech. Syst.*, Nov. 2016, pp. 498–503.

[14] F. L. Lewis, "Adaptive dynamic programming for feedback control," in *Proc. 7th Asian Control Conf.*, Aug. 2009, pp. 10–11.

[15] G. F. Franklin, J. D. Powel, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, 2014.

[16] A. Boukhtouta, J. Berger, W. B. Powell, and A. George, "An adaptive-learning framework for semi-cooperative multi-agent coordination," in *Proc. IEEE Symp. Adapt. Dyn. Program. Reinforcement Learn. (ADPRL)*, Apr. 2011, pp. 324–331.

[17] X. Yu, S. Jin, G. Liu, T. Lei, and Y. Ren, "A data-driven distributed adaptive control approach for nonlinear multi-agent systems," *IEEE Access*, vol. 8, pp. 207884–207893, 2020.

[18] W. Lin, W. Zhao, and H. Liu, "Robust optimal formation control of heterogeneous multi-agent system via reinforcement learning," *IEEE Access*, vol. 8, pp. 218424–218432, 2020.

[19] K. J. Astrom and B. Wittenmark, *Adaptive Control*. New York, NY, USA: Dover, 2008.

[20] P. H. M. Rêgo, J. V. F. Neto, and E. F. M. Ferreira, "Convergence of the standard RLS method and UDU^T factorisation of covariance matrix for solving the algebraic Riccati equation of the DLQR via heuristic approximate dynamic programming," *Int. J. Syst. Sci.*, vol. 46, pp. 1–23, Aug. 2013.

[21] J. V. F. Neto, E. F. M. Ferreira, and P. H. M. Rêgo, "Online optimal DLQR-DFIG control system design via recursive least-square approach and state heuristic dynamic programming for approximate solution of the HJB equation," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2013, pp. 3174–3179.

[22] Q. Zhang, J. Zhang, X. Wang, Y. Xu, and Z. Yu, "Wind field disturbance analysis and flight control system design for a novel tilt-rotor UAV," *IEEE Access*, vol. 8, pp. 211401–211410, 2020.

[23] M. Kazim, A. T. Azar, A. Koubaa, and A. Zaidi, "Disturbance-rejection-based optimized robust adaptive controllers for UAVs," *IEEE Syst. J.*, vol. 15, no. 2, pp. 3097–3108, Jun. 2021.

[24] X. Li, S. Xu, H. Gao, and H. Cai, "Distributed tracking of leader-follower multiagent systems subject to disturbed Leader's information," *IEEE Access*, vol. 8, pp. 227970–227981, 2020.

[25] B. Wang, Z. A. Ali, and D. Wang, "Controller for UAV to oppose different kinds of wind in the environment," *J. Control Sci. Eng.*, vol. 2020, p. 10, Jan. 2020.

[26] H. Li, X. Xu, and Y. Zhao, "Efficient simulation budget allocation with bound information," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 207–222, Jan. 2020.

[27] W. Gao, Z.-P. Jiang, F. L. Lewis, and Y. Wang, "Leader-to-formation stability of multiagent systems: An adaptive optimal control approach," *IEEE Trans. Autom. Control*, vol. 63, no. 10, pp. 3581–3587, Oct. 2018.

[28] D. L. Kleinman, "On an iterative technique for Riccati equation computations," *IEEE Trans. Autom. Control*, vol. AC-13, no. 1, pp. 114–115, Feb. 1968.

[29] M. Bellanger, "A survey of QR based fast least squares adaptive filters: From principles to realization," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, Apr. 1991, p. 1991.

[30] A. A. Rontogiannis and S. Theodoridis, "New fast inverse QR least squares adaptive algorithms," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, vol. 2, May 1995, pp. 1412–1415.

[31] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 2013.

[32] C. M. Rader and A. O. Steinhardt, "Hyperbolic householder transformations and the solution of least squares equations/sup," in *Proc. Nineteeth Asilomar Conf. Circuits, Syst. Comput.*, Nov. 1985, pp. 436–440.

[33] E. F. M. Ferreira, P. H. M. Rêgo, and J. V. F. Neto, "Numerical stability improvements of state-value function approximations based on RLS learning for online HDP-DLQR control system design," *Eng. Appl. Artif. Intell.*, vol. 63, pp. 1–19, Aug. 2017.

[34] S. Arora and B. Borak, *Computational Complexity: A Modern Approach*, 1st ed. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[35] R. M. Corless and N. Fillion, *A Graduate Introduction to Numerical Methods*, 1st ed. New York, NY, USA: Springer, 2013.

[36] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002.

[37] E. F. M. Ferreira, P. H. M. Rego, and J. V. F. Neto, "Computational performance of state-value function approximators based on RLS-HDP estimators for online DLQR control system design," in *Proc. UKSim-AMSS 18th Int. Conf. Comput. Modelling Simul. (UKSim)*, Apr. 2016, pp. 27–34.

[38] T. Bresciani, "Modelling, identification and control of a quadrotor helicopter," M.S. thesis, Dept. Autom. Control, Lund Univ., Lund, Sweden, 2008.

[39] A. Shalimoon, "Cooperative control of multi-agent systems: Consensus, flocking and formation control," M.S. thesis, Dept. Elect. Eng., California State Univ., Northridge, Long Beach, CA, USA, 2018.

[40] G. Lafferriere, A. Williams, J. Caughman, and J. J. P. Veerman, "Decentralized control of vehicle formations," *Syst. Control Lett.*, vol. 54, no. 9, pp. 899–910, Sep. 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167691105000204

[41] P. Tabuada, G. J. Pappas, and P. Lima, "Feasible formations of multi-agent systems," in *Proc. Amer. Control Conf.*, Jun. 2001, pp. 56–61.

[42] M. Igarashi and H. Nakamura, "Collision avoidance assist control for two-wheel vehicle robots by control barrier function," in *Proc. Int. Autom. Control Conf. (CACS)*, Nov. 2018, pp. 1–6.

[43] A. Hegde and D. Ghose, "Multi-uav collaborative transportation of payloads with obstacle avoidance," *IEEE Control Syst. Lett.*, vol. 6, pp. 926–931, 2022.

[44] K. Long, C. Qian, J. Cortes, and N. Atanasov, "Learning barrier functions with memory for robust safe navigation," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4931–4938, Jul. 2021.