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

ERNESTO F. M. FERREIRA¹, JOÃO VIANA DA FONSECA NETO², AND RASTKO R. SELMIC³
¹Federal Institute of Maranhão, Açailândia - MA, Brasil (e-mail: ernesto.ferreira@ifma.edu.br)
²Federal University of Maranhão, São Luís - MA, Brasil (e-mail: jviana@dee.ufma.br)
³Concordia University, Montreal - QC, Canada (e-mail: rastko.selmic@concordia.ca)

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 \(QR\)-Solver and RLS-\(QR\)-HDP-DLQR algorithms provide a new methodology to obtain the solution of the Hamilton-Jacobi-Bellman (HJB) equation. These proposed solutions are based on \(QR\) 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 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 QR-solver (developed in this research) and the RLSₜ₋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 control (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 QR-Solver and the RLSₜ₋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 ecosystem 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

\[ \dot{v} = Ev + Hw, \]

(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 \( w = \bar{w} + \hat{w} \), where \( \bar{w} \) is a feedback control input \( \bar{w} = -K_1(v - v^*) \) with the leader’s desired trajectory \( v^* \) that satisfies \( \dot{v}^* = Ev^* \). \( \hat{w} \) is the external disturbance input (such as a wind gust for aerial agents and slip for terrestrial agents), and \( K_1 \) is the gain of the leader.

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

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

\[ \dot{x}_i = A_i x_i + B_i u_i + D_i v, \]

(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 \( i = 1, 2, ..., N \). The formation error \( e_i \in \mathbb{R} \) is given by

\[ e_i = C_i x_i + F_i v \quad i = 1, 2, ..., N, \]

(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

\[ u_i = -K_{zi} x_i - K_{zi} z_i, \]

(4)

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

\[ \dot{z}_i = G_1 z_i + G_2 e_i, \quad i = 1, 2, ..., N, \]

(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

\[
J = \int_0^\infty (|\xi_i(t)| + |u_i(t)|) dt,
\]

where \( \xi_i = [x_i \ z_i] \in \mathbb{R}^{2n_i} \) is the \( i \)-th subsystem. The optimal control gains are \( [K_{0i}^* \ K_{0i}^*] = R_i^{-1} B_i^T P_i^* \), where \( P_i^* \) is the unique solution to the algebraic Riccati equation (ARE)

\[
A_i^T P_i^* + P_i^* A_i + Q_i - P_i^* B_i R_i^{-1} B_i^T P_i^* = 0.
\]

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

\[
\dot{\xi}_i = G_1 \xi_i + G_2 \dot{\xi}_i, \quad i = 1, 2, ..., N,
\]

where \( \dot{\xi}_i = y_i + F_i \xi_i \). The dynamics of \( \xi_i \in \mathbb{R}^q \) is given by

\[
\dot{\xi}_i = E \xi_i + \sum a_{ij}(t)(\xi_j - \xi_i), \quad i = 1, 2, ..., N,
\]

with \( \xi_0 = v \). The \( i \)-th subsystem augmenting \( \xi_i \) with the internal model of (8) is

\[
\dot{\xi}_i = \tilde{A}_i \xi_i + \tilde{B}_i u_i + \hat{D} \psi_i,
\]

for \( i = 1, 2, ..., N \), where

\[
\tilde{A}_i = \begin{bmatrix} A_i & 0 \\ G_2 C_i & G_1 \end{bmatrix}, \quad \tilde{B}_i = \begin{bmatrix} B_i \\ 0 \end{bmatrix},
\]
\[
\hat{D}_i = \begin{bmatrix} D_i & 0 \\ 0 & G_2 F_i \end{bmatrix},
\]

and \( \psi_i = [v \ \ \xi] \in \mathbb{R}^{2q} \).

### 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

\[
|\xi_i(t + \delta t)|_{P_i^*} - |\xi_i(t)|_{P_i^*} = \int_t^{t+\delta t} \left[ -|\xi_i|_{\Delta} + 2\psi_i^T \hat{D}_i P_i^* \xi_i \\ + 2(u_i + K_i^{k+1} \xi_i)^T R_i K_i^{k+1} \xi_i \right] dt,
\]

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

\[
\Phi_i^k \left[ \begin{array}{c} P_i^k \\ K_i^{k+1} \\ D_i^T P_i^k \end{array} \right] = y_i^k,
\]

where

\[
\Phi_i^k = [\delta_{i,1} - 2 \Gamma_i \xi_i (I \otimes (K_i^k)^T R_i) - 2 \Gamma_i \xi_i u_i (I \otimes R_i), 2 \Gamma_i \xi_i q_i],
\]

\[
y_i^k = -\Gamma_i \xi_i (Q_i + (K_i^k)^T R_i K_i^k).
\]

Let us define \( \Gamma \) and \( \delta \) as

\[
\Gamma_{a,b} = \left[ \int_{t_0}^{t_1} a \otimes b \, d\tau, \int_{t_1}^{t_2} a \otimes b \, d\tau, ..., \int_{t_{j-1}}^{t_j} a \otimes b \, d\tau \right],
\]

\[
\delta_a = \left[ kron(a(t_1)) - kron(a(t_0)), ..., kron(a(t_j)) - kron(a(t_{j-1})) \right],
\]

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

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

\[
\xi = \hat{A}_c \xi + \hat{B}_c (1_N \otimes v),
\]

where \( \hat{A}_c = I_N \otimes [\tilde{A}_i - \tilde{B}_i K_i] \) and \( \hat{B}_c = I_N \otimes [0 \ F_i^T G_i^2]^T \).

The error of the close-loop system is given by

\[
e = \hat{C} \xi + F (1_N \otimes v).
\]

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 and formation control of multi-agent systems are presented. The QR-solver algorithm is based on the LS approach, and the RLS-Q-R-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. QR-Solver Algorithm

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 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 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 QR-solver, was developed using the Householder transformation to calculate the values of QR matrices in order to find the least squares solution [32]. The QR-solver is described in the next paragraph.

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

\[
\Phi_i^k = Q_i^k R_i^k, \tag{19}
\]

where \( Q_i^k \in \mathbb{R}^{n \times n} \) is an orthogonal matrix and \( R_i^k \in \mathbb{R}^{n \times n} \) is an upper triangular matrix with positive diagonal elements. The inverse is given by

\[
(\Phi_i^k)^{-1} = (Q_i^k R_i^k)^{-1} = (R_i^k)^{-1} (Q_i^k)^T. \tag{20}
\]

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

\[
R_i^k \begin{bmatrix} P_i^k \\ K_i^k \\ D_i^T P_i^k \end{bmatrix} = Q_i^T y_i, \tag{21}
\]

where \( R_i^k \) is an upper triangular matrix whose inverse can be easily found without computing the determinant.

The QR factorization procedure for computing \( P_i^k \) (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 QRSOLVER).

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 \( \Phi_i \) 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: QR-SOLVER**

**Setup (Initial Conditions)**
1 Dynamic System Matrices: \( A_i, B_i, C_i, D_i, E, F_i, G_i, G_2, H, H \).
2 Initial State: \( x_{0i}, \xi_0, v_{0i}, y_{0i} \).
3 Initial Values of P, Q, R and K: \( P_0, K_0, K_0 \).
4 Number of Iterations: \( N \); Sample time: \( t_s \); Noise: \( \omega \).

**Iterative Process**
5 for \( k \leftarrow 1 : N \)
   6 do
      Switch Signal
      \( H^k \leftarrow \) chosen graph (\( G \))
      Environment Simulation
      \( z_i^k \leftarrow A_i x_i^k + B_i u_i^k + D_i \psi_i^k \)
      \( e_i^k \leftarrow E v_i^k + H (K_0 v_i^k - v_i^k + e_i^k) \)
      \( z_i^k \leftarrow \lfloor (I_y \otimes E) - (H_i^2 \otimes I_y) \rfloor \xi_i^k + [H_i^2 \otimes I_y] \lfloor 1_y, v_i \rfloor \)
      Calculation of \( \delta_i^k \) and \( \Gamma_i^k \) to \( t_0 \) at \( t_s \)
   7 end

**Control Update** (for each agent \( i \))
8 for \( k \leftarrow 1 : N_{int} \)
   9 do
      Calculation of matrix \( \Phi_i \) and vector \( \Phi_i \)
      \( \Phi_i^k \leftarrow [I_{y_i} \otimes (K_i^k)^T R_i] - 2G_i u_i^k (I \otimes R_i) - 2G_i v_i^k \)
      \( y_i^k \leftarrow \Gamma_i \xi_i^k + vec(Q_i \otimes (K_i^k)^T R_i K_i^k) \)
      Calculation of new gain \( K_i \) using LS
      \( Q_i^k R_i^k \leftarrow \Phi_i^k \)
      \( R_i^k \leftarrow P_i^k \otimes (K_i^k)^{-1}; D_i^T P_i^k \leftarrow (Q_i^k)^T y_i^k \)
   10 end

**Repeat the use of** \( K_i^{k+1} \)

**B. RLS_s-QR-HDP-DLQ AR 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 \( \hat{\theta}_i = [P_i^k; K_i^{k+1}; D_i^T P_i^k]^T \). Let \( \Phi_i \) 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 \( \Phi_i \), and \( \Phi_i^{H/2} \) is the Hermitian transpose of \( \Phi_i^{1/2} \).

Note, from the RLS_s-QR-HDP-DLQ algorithm, that the recursive updating equation of the correlation matrix \( \Phi_i \) is given by the following:

\[
\Phi_i = \mu \Phi_i - 1 + x_i x_i^H. \tag{24}
\]

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

\[
\omega_i = \Phi_i^{H/2} \hat{\theta}_i = \Phi_i^{-1/2} y_i. \tag{25}
\]

We obtain here a new form to represent the cross-correlation vector between the input \( x_i \) 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
\[ \Phi \hat{\theta}_i = \mu \Phi_{i-1} \hat{\theta}_{i-1} + \bar{x}_i d_i^*, \tag{27} \]

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:
\[ \mu \Phi_i^H = (\mu^{1/2} \Phi_{i-1}^H) \cdot (\mu^{1/2} \Phi_{i-1}^H), \tag{28} \]
\[ \mu \hat{\theta}_i^H \Phi_i = (\mu^{1/2} \hat{\theta}_{i-1}^H \Phi_{i-1}) \cdot (\mu^{1/2} \Phi_{i-1}^H). \tag{29} \]

Suppose one chooses a unitary rotation \( \Theta_i \) 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:
\[ A \Theta = B, \tag{30} \]

where
\[ A = \begin{bmatrix} \mu^{1/2} \Phi_{i-1}^{1/2} & \bar{x}_i \\ \mu^{1/2} \omega_i^{H} & 1 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} \Phi_i^H & 0 \\ b_{i1}^H & b_{i21}^* \\ b_{i31}^H & b_{i321}^* \end{bmatrix}. \tag{31} \]

To evaluate the elements of the unknown blocks \( B_i^{H}, b_{i11}^H, b_{i21}^H, b_{i31}^H, \) and \( b_{i321}^* \) of the post-array \( B \), the procedure is to square both sides of Eq. (30). According to the matrix factorization lemma, [31], \( \Theta_i \) 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
\[ A A^H = B B^H, \tag{32} \]
or in an expanded form
\[
\begin{bmatrix}
\mu^{1/2} \Phi_{i-1}^{1/2} & \bar{x}_i \\
\mu^{1/2} \omega_i^{H} & 1 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
\mu^{1/2} \Phi_{i-1}^H & \mu^{1/2} \omega_i^{-1} & 0 \\
\bar{x}_i^H & d_i^* & 1
\end{bmatrix}

= \begin{bmatrix}
B_i^H & 0 \\
b_{i11}^H & b_{i21}^* & b_{i31}^H \\
b_{i31}^H & 0 & b_{i321}^*
\end{bmatrix}. \tag{33}
\]

Comparing the respective terms on both sides of Eq. (33), the unknown block elements of a post-array \( B \) are determined. Consequently, Eq. (30) with the values of \( B_i^{H}, b_{i11}^H, b_{i21}^H, b_{i31}^H, \) and \( b_{i321}^* \) is expressed as follows:
\[
\begin{bmatrix}
\mu^{1/2} \Phi_{i-1}^{1/2} & \bar{x}_i \\
\mu^{1/2} \omega_i^{H} & 1 \\
0 & 1
\end{bmatrix}
\Theta_i = \begin{bmatrix}
\Phi_i^{1/2} & 0 \\
\omega_i^H & \xi_i \rho_i^{1/2} \\
\bar{x}_i^H \Phi_i^{1/2} & \rho_i^1
\end{bmatrix}. \tag{34}
\]

where \( \xi_i \) is the a priori estimation error given by \( \xi_i = d_i - \hat{\theta}_i^H \bar{x}_i \), and \( \rho_i \) is a real parameter [33]. The equations in this subsection form the ALGORITHM 2, labeled RLS\textsubscript{HDP-DLQR}.
according to Eq. 31. In step 18, the $QR$ factorization is applied, generating the matrices $Q_i$ and $R_i$ according to Eq. (34). In step 19, the values of $\Phi$ and $\omega$ are defined, and the calculation of $\dot{\theta}_i$ in step 20 is applied using Eq. (25). In step 21, the matrix $P^\mu_i$ is formed with the elements of the vector $\dot{\theta}_i$, 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 $QR$-solver and RLS$_\mu$-$QR$-HDP-DLQR algorithms are evaluated in relation to convergence of parameter $\theta$ 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 $\mu$, several simulations were conducted. The convergence process of RLS$_\mu$-$QR$-HDP-DLQR for the HJB-Riccati equation solutions for different $\mu$ 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](image)

Based on Figure 1, the Laplacian matrices for multi-agent systems $T_1$ and $T_2$ are given by

$$L_{T_1} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad L_{T_2} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}.$$  

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

\[ F_z = F_c - F_g, \]  

(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

\[ \frac{d^2 z}{dt^2} = \ddot{z} = \frac{F_c}{m} - g, \]  

(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 \( \omega = 22000 \text{rpm} \), the power constant is \( k_f = 3.1 \times 10^{-8} \), mass is \( m = 0.08 kg \), and the gravity constant is \( g = 9.81 m/s^2 \).

The matrices of the dynamic system are as follows:

\[
A_i = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad B_i = \begin{bmatrix} F_c \\ \frac{m}{F_c} \end{bmatrix},
\]  

(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-dron Parrot® and the functionality of the \( QR \) 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 \( QR \)-solver. The numerical stability, trajectory, and formation convergence of the \( QR \)-solver and \( \text{RLS}_P-QR\)-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 \( (A_1, B_1) \) for Follower 1 and \( (A_2, B_2) \) for Follower 2.

1) \( QR \)-Solver Evaluation

The control update using the \( QR \)-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 \( \theta_{10} \) of the parameter vector \( \theta \) for Follower 1, respectively.

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

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 to 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 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 $\Gamma$ and $\delta$ of Eq. (15-16), necessary for the calculation of $\Phi_k^1$ and $y_k^1$ in Eq. (12).

2) RLS$_{\mu}$-QR-HDP-DLQR evaluation
The results of the RLS$_{\mu}$-QR-HDP-DLQR experiments are presented in an analogous form to those of the QR-solver. A cycle of 200 iterations of the control update and 20 s 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 $\theta_1, \theta_5, \theta_8,$ and $\theta_{10}$ of the parameter vector $\theta$ for Follower1.

The convergence of the $\theta$ 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$_{\mu}$-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.

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

| Algorithm          | $p_{11}$ | $p_{22}$ | $p_{33}$ | $p_{44}$ |
|--------------------|----------|----------|----------|----------|
| Schur              | 227.75   | 7.41     | 36.88    | 37.49    |
| QR-Solver          | 227.76   | 7.40     | 36.88    | 37.48    |
| RLS$_{\mu}$-QR-HDP-DLQR | 199.44   | 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 |
|--------------------|--------------|-------------|
| QR-Solver          | 1082         | One Time    |
| RLS$_{\mu}$-QR-HDP-DLQR | 792         | Each Iteration |

Analyzing the results of Experiment 1, it can be observed that the computational cost of RLS$_{\mu}$-QR-HDP-DLQR is lower than that of 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]

$$
A_t = \begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix} \quad \text{and} \quad B_t = \begin{bmatrix}
0 & 0 \\
1 & 0 \\
0 & 0 \\
0 & 1
\end{bmatrix},
$$

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 QR-solver and RLS$_\mu$-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](image)

1) $QR$-Solver Evaluation

The evolution of the iterative process for the $QR$-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 $\theta_1, \theta_5, \theta_8$, and $\theta_{10}$ of the parameter vector $\theta$ of Follower 1, respectively. The parameter $P_{estim}$ reaches the Schur solution $P^0$ in approximately 7 iterations.

![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 $QR$-solver algorithm of Follower 1.](image)

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.

![Figure 9. Evolution of the control process for the position of the agents along the X-axis and Y-axis for 100s, on the $QR$-solver algorithm.](image)

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

2) RLS$_\mu$-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 $\theta_1, \theta_5, \theta_8$, and $\theta_{10}$ of the parameter vector $\theta$ of Follower 1.

![Figure 10. Evolution of the control process for the position of the agents along the X-axis and Y-axis for 200s, on the RLS$_\mu$-QR-HDP-DLQR algorithm.](image)
One Time

Time for Each Iteration

Formation

Cost (Flops)

Gain Update error (cm)

| Algorithm          | $P_{11}$ | $P_{22}$ | $P_{23}$ | $P_{44}$ |
|--------------------|---------|---------|---------|---------|
| Schur              | 260.6   | 10.32   | 45.76   | 43.52   |
| $QR$-Solver        | 260.1   | 10.45   | 45.75   | 42.52   |
| RLS$_{\mu}$-$QR$-HDP-DLQR$^3$ | 255.52  | 9.82    | 46.86   | 50.63   |

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.

### Table 4. Result of computational cost in Experiment 2

| Algorithm          | Cost (Flops) | Gain Update |
|--------------------|--------------|-------------|
| $QR$-Solver        | 2164         | One Time    |
| RLS$_{\mu}$-$QR$-HDP-DLQR$^3$ | 1584 | Each Iteration |

Analyzing the results of Experiment 2, it was observed that the computational cost of RLS$_{\mu}$-$QR$-HDP-DLQR is lower than that of $QR$-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 $QR$-Solver and RLS$_{\mu}$-$QR$-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) |
|--------------------|------------------------|----------------------|
| $QR$-Solver        | 8.2                    | 2.64                 |
| RLS$_{\mu}$-$QR$-HDP-DLQR$^3$ | 13.6               | 3.83                 |
| 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 QR-Solver and RLSµQR-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 QR-solver and RLSµQR-HDP-DLQR algorithms show satisfactory results for trajectory tracking and formation convergence. The 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 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 RLSµQR-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 QR-solver and RLSµQR-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.

ACKNOWLEDGMENT

The authors would like to thank the Federal University of Maranhão (UFMA), Concordia University, Federal Institute of Maranhão (IFMA), the Coordination of Undergraduate Personal Improvement (CAPES) through PDSE, FAPEMA, and the National Council for Scientific and Technological Development (CNPq) for the development of the infrastructure and financial support to carry out this work.

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. 169 590–169 613, 2019.
[2] S. S. Ge, X. Liu, C. Goh, and L. Xu, “Formation tracking control of multiagents in constrained space,” IEEE Transactions on Control Systems Technology, vol. 24, no. 3, pp. 992–1003, 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. 204 256–204 267, 2020.
[4] G. Weiss, Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence. MIT Press, 2000.
[5] Z. R. Bogdanowicz, “Flying swarm of drones over circular digraph,” IEEE Transactions on Aerospace and Electronic Systems, vol. 53, no. 6, pp. 2662–2670, 2017.
[6] M. Mesbahi and M. Egerstedt, Graph Theoretic Methods in Multiagent Networks. Princeton University 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 Transactions on Neural Networks and Learning Systems, vol. 29, no. 12, pp. 6015–6025, 2018.
[8] X. Dong and G. Hu, “Time-varying formation tracking for linear multiagent systems with multiple leaders,” IEEE Transactions on Automatic Control, vol. 62, no. 7, pp. 3638–3644, 2017.
[9] J. Hu and W. X. Zheng, “Adaptive tracking control of leader-follower systems with unknown dynamics and partial measurements,” in Automatica, vol. 50, February 2014, pp. 1416–1423.
[10] G. Wen, C. L. P. Chen, and Y. Liu, “Formation control with obstacle avoidance for a class of stochastic multiagent systems,” IEEE Transactions on Industrial Electronics, vol. 65, no. 7, pp. 5847–5855, 2018.
[11] G. Jing and L. Wang, “Multiagent flocking with angle-based formation shape control,” IEEE Transactions on Automatic Control, vol. 65, no. 2, pp. 817–823, 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 Transactions on Cybernetics, vol. 50, no. 9, pp. 3826–3839, 2020.
[13] W. Wang, Z. Man, Z. Cao, J. Zheng, and Y. Zhao, “Dynamics Modelling and Linear Control of Quadcopters,” Proceedings of the 2016 International Conference on Advanced Mechatronic Systems, 2016.
[14] F. L. Lewis, “Adaptive dynamic programming for feedback control,” in 2009 7th Asian Control Conference, Aug 2009, pp. 10–11.
[15] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, Feedback Control of Dynamic Systems. Prentice Hall, 2014.
[16] A. Boukhouta, J. Berger, W. B. Powell, and A. George, “An adaptive-learning framework for semi-cooperative multi-agent coordination,” in 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 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. 207 884–207 893, 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. 218 424–218 432, 2020.
[19] K. J. Astrom and B. Wittenmark, Adaptive Control. Dover Publications, 2008.
[20] P. H. M. Régo, J. V. F. Neto, and E. F. M. Ferreira, “Convergence of the standard rl method and udt factorisation of covariance matrix for solving the algebraic riccati equation of the dlqr via heuristic approximate dynamic programming,” International Journal of Systems Science, pp. 1–23, 2013.
[21] J. V. F. Neto, E. F. M. Ferreira, and P. H. M. Régo, “Online optimal dlqrdfig control system design via recursive least-square approach and state heuristic dynamic programming for approximate solution of the hjb equation,” in Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on, 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. 211 401–211 410, 2020.
[23] M. Kazim, A. T. Azar, A. Koubaa, and A. Zaidi, “Disturbance-rejection-based optimized robust adaptive controllers for uavs,” IEEE Systems Journal, vol. 15, no. 2, pp. 3097–3108, 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. 227 970–227 981, 2020.
[25] B. Wang, Z. A. Ali, and D. Wang, “Controller for uav to oppose different kinds of wind in the environment,” Journal of Control Science and Engineering, p. 10, 2020.
[26] H. Li, X. Xu, and Y. Zhao, “Efficient simulation budget allocation with bound information,” IEEE Transactions on Automatic Control, vol. 65, no. 1, pp. 207–222, 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 TRANSACTIONS ON AUTOMATIC CONTROL, 2018.
[28] D. Kleiman, “On an iterative technique for riccati equation computations,” IEEE Transactions on Automatic Control, vol. 13, no. 1, pp. 114–115, 1968.
[29] M. Bellanger, “A survey of qr based fast least squares adaptive filters: from principles to realization,” in [Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing, 1991, pp. 1833–1836 vol.3.

[30] A. Rontogiannis and S. Theodoridis, “New fast inverse qr least squares adaptive algorithms,” in 1995 International Conference on Acoustics, Speech, and Signal Processing, vol. 2, 1995, pp. 1412–1415 vol.2.

[31] G. H. Golub and C. F. Van Loan, Matrix Computations, 4th ed. Johns Hopkins University Press, 2013.

[32] C. M. Rader and A. O. Steinhardt, “Hyperbolic householder transformations and the solution of least squares equations,” in Nineteenth Asilomar Conference on Circuits, Systems and Computers, 1985., Nov 1985, pp. 436–440.

[33] E. M. Ferreira, P. H. M. Régo, and J. V. F. Neto, “Numerical stability improvements of state-value function approximations based on rls learning for online hdp-dlqr control system design,” Engineering Applications of Artificial Intelligence, 2017.

[34] S. Arora and B. Borak, Computational complexity: A modern approach, 1st ed. Cambridge University Press, 2009.

[35] R. M. Corless and N. Fillion, A Graduate Introduction to Numerical Methods, 1st ed. Springer, 2013.

[36] N. J. Higham, Accuracy and Stability of Numerical Algorithms, 2nd ed. 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 Computer Modelling and Simulation (UKSim), 2016 UKSim-AMSS 16th International Conference on, April 2016, pp. 27–34.

[38] T. Bresciani, “Modelling, identification and control of a quadrotor helicopter,” Master’s thesis, Department of Automatic Control Lund University, 2008.

[39] A. Shalimoon, “Cooperative Control of Multi-Agent Systems: Consensus, Flocking and Formation Control,” Master’s thesis, California State University, Northridge, 2018.

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

[41] P. Tabuada, G. Pappas, and P. Lima, “Feasible formations of multi-agent systems,” in Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148), vol. 1, 2001, pp. 56–61 vol.1.

[42] M. Igarashi and H. Nakamura, “Collision avoidance assist control for two-wheel vehicle robots by control barrier function,” in 2018 International Automatic Control Conference (CACS), 2018, pp. 1–6.

[43] A. Hegde and D. Ghose, “Multi-uav collaborative transportation of payloads with obstacle avoidance,” IEEE Control Systems Letters, vol. 6, pp. 926–931, 2022.

[44] K. Long, C. Qian, J. Cortés, and N. Atanasov, “Learning barrier functions with memory for robust safe navigation,” IEEE Robotics and Automation Letters, vol. 6, no. 3, pp. 4931–4938, 2021.

***