OPTIMAL CONTROL APPROACHES FOR CONSENSUS AND PATH PLANNING IN MULTI-AGENT SYSTEMS

A Dissertation

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

POORYA SHOBEIRY

Dr. Ming Xin, Dissertation Supervisor

MAY 2020

The undersigned, appointed by the dean of the Graduate School, have examined the dissertation entitled

OPTIMAL CONTROL APPROACHES FOR CONSENSUS AND PATH PLANNING IN MULTI-AGENT SYSTEMS

presented by Poorya Shobeiry

a candidate for the degree of Doctor of Philosophy,

and hereby certify that, in their opinion, it is worthy of acceptance.

Professor Ming Xin

Professor Craig Kluever

Professor Noah D. Manring

Professor Roger Fales

Professor Konstantin Makarov

ACKNOWLEDGEMENT

I would like to express my gratitude and appreciation for Professor Ming Xin whose guidance, support and encouragement has been invaluable throughout this study. I also wish to thank all the members of my committee who have been a great source of support and guidance all these years.

Contents

LIST OF FIGURES

ABSTRACT

Optimal control is one of the most powerful, important and advantageous topics in control engineering. The two challenges in every optimal control problem are defining the proper cost function and obtaining the best method to minimize it. In this study, innovative optimal control approaches are developed to solve the two problems of consensus and path planning in multi-agent systems (MASs). The consensus problem for general Linear-Time Invariant systems is solved by implementing an inverse optimal control approach which enables us to start by deriving a control law based on the stability and optimality condition and then according to the derived control define the cost function. We will see that this method in which the cost function is not specified *a priori* as the conventional optimal control design has the benefit that the resulting control law is guaranteed to be both stabilizing and optimal. Three new theorems in related linear algebra are developed to enable us to use the algorithm for all the general LTI systems. The designed optimal control is distributed and only needs local neighbor-to-neighbor information based on the communication topology to make the agents achieve consensus and track a desired trajectory.

Path planning problem is solved for a group are Unmanned Aerial Vehicles (UAVs) that are assigned to track the fronts of a fires in a process of wildfire management. We use Partially Observable Markov Decision Process (POMDP) in order to minimize the cost function that is defined according to the tracking error. Here the challenge is designing the algorithm such that (1) the UAVs are able to make decisions autonomously on which fire front to track and (2) they are able to track the fire fronts which evolve over time in random directions. We will see that by defining proper models, the designed algorithms provides real-time calculation of control variables which enables the UAVs to track the fronts and find their way autonomously. Furthermore, by implementing Nominal Belief-state Optimization (NBO) method, the dynamic constraints of the UAVs is considered and challenges such as collision avoidance is addressed completely in the context of POMDP.

1. Introduction

In this research, innovative optimal control approaches are developed to solve the two important problems of consensus and path planning among multi-agent systems (MASs). Consensus is important when the objective of a mission is to get to some specific state or track a given desired trajectory. Path planning is generally called to any process of finding a sequence of valid configuration that moves the agent through an unnecessarily given route. Here we focus on a special application of having a group of Unmanned Aerial Vehicles (UAVs) track fire fronts in order to collect data of their spread as an important part of wildfire management process. Consensus problem for general Linear Time-Invariant systems is discussed in chapter 2 and 3. Path planning algorithm for fire fighter UAVs is developed in chapter 4 via a special case of Markov Decision Process (MDP) named Partially Observable Markov Decision Process (POMDP). Here we briefly introduce the two problems, back ground of each one, our approaches and advantages of them.

Cooperative control problem for multi-agent systems (MASs) has received tremendous attention in the last two decades owing to its wide range of applications. Furthermore, the concept of distributed multi-agent cooperative systems was developed with the aid of rapid progress in communication, sensing, and actuation. The cooperative teamwork provides much more flexibility and robustness in performance to accomplish certain missions compared to single-agent systems and therefore, it has been applied to many practical engineering problems such as mobile robots, unmanned aerial vehicles (UAVs), autonomous underwater vehicles (AUVs), spacecraft, and automated highway as summarized in [1] . A comprehensive overview on the progress of multi-agent coordination has been provided by Cao *et al* [2].

The technical core of cooperative control of MASs is the concept of consensus. It is said that multiple vehicles have achieved consensus when they agree on the value of a common variable of interest. A good example is a dinner meeting among a group of friends. Suppose that five friends want to spend an evening together and have dinner. They all know the place (the particular restaurant) but they are uncertain about the time of the meeting. One solution is to make a conference call in which all five attend and decide about the meeting time. This centralized approach is not a real solution because they still need to set a time for attending the conference call all together. The common variable of interest (also called the "coordination variable") is the time the friends want to meet. The distributed solution is for each individual to call a subgroup of friends and make a decision on a preferred time. Here we suppose that each individual has phone numbers of only two of his/her friends. That means each individual can only communicate with two friends. This calling process continues and the preferred time is updated each time until it converges to a final consistent meeting time. Figure 1 shows this process of agreement.

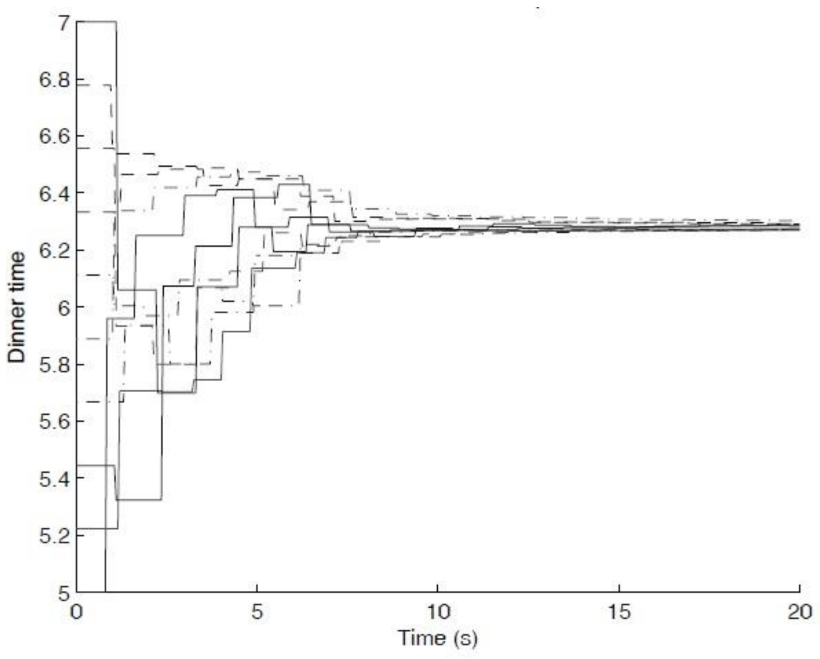

Figure 1. Meet-for-dinner consensus problem[3]

According to the meet-for-dinner example, consensus among agents is equivalent to achieving a common state by neighbor-to-neighbor interaction and information exchange. The information and interaction exchange among the agents is represented by communication topology which is modeled via graph theory. The classical consensus problem and the associated graph and matrix theories were introduced and extensively developed primarily on single or double integrator systems [4, 5]. The basic consensus algorithm was rapidly generalized to the case of higher-order systems [6, 7]. Other recent progress on the average consensus algorithms includes developing novel weighting strategies [8] and new classes of fixed-time protocols [9].

Since many of the practical applications involve formation and tracking problems of mobile vehicles, the integrator systems (both single and double-integrator) have received the most attention and have been thoroughly studied during the last decade [10-12]. Considering the constraints of real applications such as limited communication ranges and limited bandwidth, the concept of switching topology (instead of time-fixed topology) emerged and the conditions for achieving consensus under the switching topology were extensively investigated [13-18]. Besides, the leader-follower algorithm as the most straightforward approach for solving the tracking problem has been studied and developed as well [19-23].

Nonlinear analysis has also been taken into consideration in consensus analysis. Intermittent communication (i.e. the case when the communication among the agents is attacked disconnects at frequent time intervals) in nonlinear singular MASs has been addressed by Xie and Mu [24] as a crucial issue. They have used multi-Lyapunov function approach, the observer-based intermittent feedback control protocol to solve the consensus problem. Deng *et al*. [25] have studied another practical problems called “packet dropout” which occurs due to unreliable wireless communication. They have applied an iterative learning control method to design the control protocol. Other studies have been accomplished on applying model reference adaptive control (MRAC) in order to design the optimal control for Heterogeneous Nonlinear Multiagent Systems whose Dynamics is partially unknown [26] and solving Lag Group Consensus problem via adaptive control approach [27].Some other methods to address the consensus problem include designing observers [28, 29] and compensators [30].

When communication does not occur continuously, the system is called “discrete-time”. The basic algorithms for consensus in such systems have been derived in [3, 4]. Many novel approaches have been developed recently such as constructing adaptive algorithms to drive the local parameters estimators [31] and developing a new form of Krasovskii-LaSalle theorem for obtaining an exponential convergence [32].

Multi-agent cooperative control has also been investigated from the optimization perspective. Consensus can be optimized by improving the network topology via the graph theory and analysis. The research along this line is mainly focused on making the most efficient configuration to attain the desired property of the representing graph. The objectives include finding the fastest convergence rate by maximizing the second smallest non-negative eigenvalue of the Laplacian matrix as described by Kim and Mesbahi [33] or obtaining the optimal topology (optimal Laplacian matrix), for example via the linear quadratic regulator (LQR) approach [34]. Linear matrix inequality (LMI) and iterative LMI are also utilized to solve the consensus problem [35-37]. Tuna [38, 39] showed that the LQR can synchronize agents if the pair of the state and input matrices (for each agent) is stabilizable. However, there was no discussion on the exact approach of constructing proper cost functions and defining the desired trajectory. Wang and Xin [40] utilized an inverse optimal control method (introduced by Bernstein [41]) for a system of agents to address not only consensus but also obstacle/collision avoidance. This similar optimal control was applied to cooperative control of multiple autonomous robots [42] and the flocking problem [43]. However, the same simplified double-integrator dynamics is assumed in these works for the MAS model. Recently, many innovations have been made in solving the consensus problem by the optimal control theory. Xie and Lin [44] solved the problem of global optimal consensus for higher-order integrators with bounded controls starting from any arbitrary initial states and in the presence of actuator saturation. Dehshalie *et al*. [45] used optimal control theory to design a fault tolerant control law for

MASs with single and multi-input actuators under both directed and undirected communication topologies. A numerical approach was developed by Bailo *et al*. [46] to solve the consensus problem of the nonlinear multi-agent system of Cucker-Smale type and the first-order optimality conditions were obtained by using Barzilai-Borwein (BB) gradient descent method. Recently, the model predictive control has been used to solve cooperative control problems such as controlling connected and autonomous vehicles in the intelligent transportation system [47].

In chapter 2, we generalize the approach described in [40, 43, 48] in order to embrace general LTI systems with single inputs. The advantage of our approach is that, aside from being applicable to general systems (rather than mere integrator systems), the optimal control is derived based on an inverse optimal control approach in which the cost function is not a priori and is obtained after the control law is derived and satisfies stability and optimality conditions. The approach will be explained thoroughly in section 2.4. The importance of this approach is that the control law is guaranteed to be both stabilizing and optimal. The other important advantage of our algorithm is that the optimal control is obtained pure analytically without using any numerical approaches. Consensus, trajectory tracking, and minimization of control effort are achieved by constructing proper cost functional via the same inverse optimal control method. The optimal cooperative control law is distributed, which only needs neighboring agents' information. Both asymptotic stability and optimality are achieved.

In chapter 3, the same approach develops to embrace the case of multi-input systems. We will see that an important challenge would be choosing the most appropriate transformation matrix in order to convert the general LTI system to a Controllable

Canonical Form (CCF) with specific properties. Three separate theorems in related linear algebra will be developed to enable us to obtain the best transformation matrix. Optimal control is achieved through the same approach and allow us to solve the consensus problem for all the general Linear Time-invariant Multi Agent systems.

In chapter 4 a novel path planning algorithm is introduced based on Partially Observable Markov Decision Process (POMDP). The algorithm is implemented for making fire fighter UAVs track the fronts of wildfires. Wildfires are known for fast and randomly evolving over time. Therefore, accurate data collection of the spreading pattern of the fire fronts is very crucial in wildfire management. Many path-planning algorithms have already been developed for autonomous system of vehicles and robots [49-59]. The approach introduced here has the following advantages, which makes it different and profitable:

1) It is designed according to the concepts of POMDP, which means that the system makes decisions only according to some observations when there is no clearly known states (a realistic model for the case of managing wildfires).
2) In response to the collected information, the approach implements real-time calculation in obtaining control commands.
3) Using "Receding Horizon" technique, the algorithm has a "look-ahead" quality in a sense that for the current time-step, the controls are calculated over a certain time horizon.
4) The dynamics constraints are taken into consideration in the introduced approach and the control variables are calculated within certain limits.

5) A discrete-time model for evolving fire front is defined to enable the UAVs to track them when they evolve in random direction.
6) The cost function is defined according to the tracking error.

Each item has already been developed and used separately. For example in [60-63] the look-ahead property has been defined but the constraint have not been considered in all of them or the cost functions have been defined without considering the tracking error. The innovation of the algorithm described in this chapter is that it solves path-planning problem in a POMDP framework considering all the features mentioned above. This is novel because in other studies, where POMDP is used to solve general path-planning problems, either not all these features have been taken into account or separate algorithms are designed to overcome the challenges such as collision avoidance or practical constraints in UAV motions. In [64] the similar approach is implemented but no constraints has been considered. Collision avoidance has been taken into account in [65] but the target tracking has not been involved. Other approaches have been also suggested for addressing collision avoidance [66-69] among which the most recent ones are Bezier curve optimization [70, 71], Particle Swarm Optimization (PSO) [72] and nonlinear Model Predictive Control (MPC) [73]. Here we extend the approach described in [74] in order to enable the algorithm to embrace the case of tracking randomly evolving fire fronts. In [74], it is assumed that target motion models are the same for all the targets. In our algorithm, no specific motion model for the fire fronts is required and the UAVs are able to track the fronts only according to data of the momentary positions of them. This makes the algorithm very compatible for being implemented in case of tracking any randomly maneuvering targets and since for each time-step a simple linear model is constructed for the target, the algorithm remains computationally fast, efficient and inexpensive. The approach computes the control variable of UAVs considering collision avoidance and the dynamic constraints and with a “look-ahead” quality in a POMDP framework and the cost function is defined according to the tracking error.

POMDP problems are known as intractable problems, which means they cannot be exactly solved but keeping the essence of the theory, approximation methods can be implemented to enable us to provide optimal solutions. There are many approximation approaches such as heuristic Expected Value-To-Go [75], policy rollout [76], hindsight and foresight optimization [77-80] and parametric optimization [79, 81]. In this chapter we use Nominal Belief-state Optimization (NBO) which is the most appropriate method for tracking problems because it is less computationally intensive and allows us to define the cost function to be analytically solvable. Moreover, since it is a special case of hindsight and foresight optimization, it can be amicably extended to embrace the case of randomness in the tracking problem.

2. Single-Input Systems

2.1 Overview

In this chapter, the consensus problem for a general linear time-invariant (LTI) multi-agent systems with a single input is studied in a new optimal control framework. The approach is derived from a modified linear quadratic regulator (LQR) method by an innovative design of the cost function by using an inverse optimal control formulation. Three cost terms are constructed to address the consensus, control effort, and cooperative tracking, respectively. This formulation allows a closed-form feedback control law with guaranteed asymptotic stability and optimality. The two important advantages of this approach are (1) the optimal control law is derived pure analytically and (2) it is distributed, which means that it only requires local information based on the communication topology to enable the agents to achieve consensus and track a desired trajectory, rather than centralized control laws in which all agents' information is required.

This chapter is organized as follows. In Section 2.2, some preliminaries and fundamental concepts of graph theory are reviewed. The system is described and the problem is formulated in Section 2.3. Main results are presented in Section 2.4 and the algorithm of the approach is also provided in this Section which can be used as a quick reference for practical applications. Two examples are shown in Section 2.5 to demonstrate the performance of the proposed method.

2.2 Preliminaries

Since communication between agents is modeled by graphs, some basic notions of the graph theory are worth reviewing briefly. $G=(N,E)$, represents a graph with the nonempty set of nodes N and the nonempty set of the edges E where $E \subseteq N \times N$. A graph is called "directed" if all the pairs are ordered. A directed path is a sequence of ordered edges in the form of $(i_1,i_2),(i_3,i_4),...,$ where $i_j \in N$. (i_1,i_2) means that agent 2 can receive information from agent 1 but not vice versa. In an undirected graph, (i_1,i_2) means that both agent 1 and agent 2 can obtain information from each other. An undirected graph is called "connected" if a path can be found between each pair of nodes. A directed graph is "strongly connected" if there is a path from every node to every other node. Figure 2 shows the differences between these types of graphs.

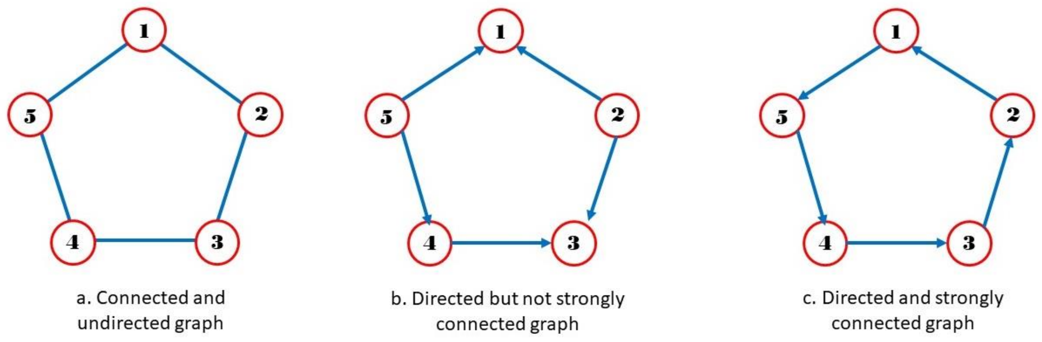
1
2
3
4
5
a. Connected and undirected graph
b. Directed but not strongly connected graph
c. Directed and strongly connected graph

Figure 2. Different types of graphs

Communication topology is described by the adjacency matrix $\boldsymbol{A}_d = \left[a_{ij}^{(d)}\right]_{n_g \times n_g}$ in which $a_{ii}^{(d)} = 0$ and

$$a_{ij}^{(d)} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{if } (i,j) \notin E \end{cases} \tag{1}$$

where n_g is the number of agents. For an undirected graph, the adjacency matrix is symmetric. The Laplacian matrix of a system of agents is defined as:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}_d \tag{2}$$

where $\boldsymbol{D}$ is the diagonal degree matrix with the entries of $d_{ii} = \sum_{j=1}^{n} a_{ij}^{(d)}$. For an undirected graph, the Laplacian matrix $\boldsymbol{L}$ is symmetric, and has a simple zero eigenvalue with an associated eigenvector $\boldsymbol{1}_{n_g}$ (a column vector of all ones) and all the other eigenvalues are positive if and only if the graph is connected. Every row sum of $\boldsymbol{L}$ is zero which means:

$$\boldsymbol{L}\boldsymbol{1}_{n_g} = \boldsymbol{0} \tag{3}$$

2.3 Problem statement and formulation

A general controllable LTI system is said to achieve consensus when all the agents' states converge to the same common value, (i.e. $\|\boldsymbol{X}_i - \boldsymbol{X}_j\| \to \boldsymbol{0}$) at the same time, where $\boldsymbol{X}_i$ is the state vector of agent *i*.

In other words, consensus is reached if there exists a real-valued function $\boldsymbol{X}_{cs}(t)$ such that $\boldsymbol{X}(t) - \boldsymbol{X}_{cs}(t) \to \boldsymbol{0}$ as $t \to \infty$. $\boldsymbol{X}$ is the state vector of the whole system of agents defined as: $\boldsymbol{X} = \left[X_{11}, \ X_{12}, \ \cdots, \ X_{1n_g}, \ X_{21}, \ X_{22}, \ \cdots, \ X_{2n_g}, \ \cdots, \ X_{n1}, \ X_{n2}, \ \cdots, \ X_{nn_g} \right]^T$ where X_{ij} represents the *i*th state of the *j*th agent.

In this study, it is assumed that each agent has the same dynamics of

$$\dot{\boldsymbol{X}}_i = \boldsymbol{A}_g \boldsymbol{X}_i + \boldsymbol{B}_g \boldsymbol{U}_i \tag{4}$$

The dynamics of the whole system of agents is written as:

$$\dot{\boldsymbol{X}} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\boldsymbol{U} \tag{5}$$

where $\boldsymbol{A} = \boldsymbol{A}_g \otimes \boldsymbol{I}_{n_g} = \left(a_{ij}\right)_{n\times n} \otimes \boldsymbol{I}_{n_g}$, $\boldsymbol{B} = \boldsymbol{B}_g \otimes \boldsymbol{I}_{n_g} = \left(b_j\right)_{n\times 1} \otimes \boldsymbol{I}_{n_g}$. n is the dimension of the single agent's system and $\boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}_1 & \boldsymbol{U}_2 & \cdots & \boldsymbol{U}_{n_g} \end{bmatrix}^T$ is the input vector for the agents.

The final consensus state vector should satisfy the system equation (5). Since $\boldsymbol{U}_{cs} = \boldsymbol{0}_{n\cdot n_g \times 1}$ when the system achieves consensus, we have

$$\dot{\boldsymbol{X}}_{cs} = \boldsymbol{A}\boldsymbol{X}_{cs} + \boldsymbol{B}\boldsymbol{U}_{cs} = \boldsymbol{A}\boldsymbol{X}_{cs} \tag{6}$$

An error state vector is defined as:

$$\hat{\boldsymbol{X}} = \boldsymbol{X} - \boldsymbol{X}_{cs} \tag{7}$$

Taking the time derivative of the error state vector yields:

$$\dot{\hat{\boldsymbol{X}}} = \dot{\boldsymbol{X}} - \dot{\boldsymbol{X}}_{cs} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\boldsymbol{U} - \boldsymbol{A}\boldsymbol{X}_{cs} = \boldsymbol{A}(\boldsymbol{X} - \boldsymbol{X}_{cs}) + \boldsymbol{B}\boldsymbol{U} = \boldsymbol{A}\hat{\boldsymbol{X}} + \boldsymbol{B}\boldsymbol{U} \tag{8}$$

Consensus is said to be reached if the system (8) is asymptotically stable.

For any controllable LTI system, a transformation matrix $\boldsymbol{T}$ can be found to transform the system to a controllable canonical form (CCF) [82].

$$\boldsymbol{T} = \boldsymbol{M}\boldsymbol{W} \otimes \boldsymbol{I}_{n_g} \tag{9}$$

where

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B} & \cdots & \boldsymbol{A}^{n-1}\boldsymbol{B} \end{bmatrix} \tag{10}$$

is the controllability matrix and $\boldsymbol{W}$ is called flipped Toeplitz matrix and has the form of:

$$\boldsymbol{W} = \begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & 1 \\ a_2 & a_3 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1} & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{11}$$

a_i' s are the coefficients of the characteristic equation of the state matrix $\boldsymbol{A}_g$.

Using the transformation matrix (9), the state vector and the final consensus vector can be expressed as:

$$\boldsymbol{X} = \boldsymbol{T}\bar{\boldsymbol{X}} \tag{12.a}$$

$$\boldsymbol{X}_{cs} = \boldsymbol{T}\bar{\boldsymbol{X}}_{cs} \tag{12.b}$$

Subtracting (12.b) from (12.a) leads to:

$$\boldsymbol{X} - \boldsymbol{X}_{cs} = \boldsymbol{T}(\bar{\boldsymbol{X}} - \bar{\boldsymbol{X}}_{cs}) \Rightarrow \hat{\boldsymbol{X}} = \boldsymbol{T}\hat{\bar{\boldsymbol{X}}} \tag{13}$$

Using (13), the system (8) can be converted to a CCF as:

$$\dot{\hat{\bar{\boldsymbol{X}}}} = \bar{\boldsymbol{A}}\hat{\bar{\boldsymbol{X}}} + \bar{\boldsymbol{B}}\boldsymbol{U} \tag{14}$$

where

$$\bar{\boldsymbol{A}} = \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} \otimes \boldsymbol{I}_{n_g} \tag{15}$$

The last row of $\bar{\boldsymbol{A}}$ contains the coefficients of the characteristic equation of the system state matrix $\boldsymbol{A}_g$ and

$$\bar{\boldsymbol{B}} = \boldsymbol{T}^{-1}\boldsymbol{B} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^T_{1\times n} \otimes \boldsymbol{I}_{n_g} \tag{16}$$

Now the system (14) can be rewritten as:

$$\dot{\hat{\bar{\boldsymbol{X}}}} = \left(\bar{\boldsymbol{A}}_2 + \bar{\boldsymbol{B}}\boldsymbol{K}_1\right)\hat{\bar{\boldsymbol{X}}} + \bar{\boldsymbol{B}}\boldsymbol{U} \tag{17}$$

where

$$\bar{\boldsymbol{A}}_2 + \bar{\boldsymbol{B}}\boldsymbol{K}_1 = \bar{\boldsymbol{A}} \tag{18}$$

and

$$\bar{\boldsymbol{A}}_2 = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{n\times n} \otimes \boldsymbol{I}_{n_g}\,,\ \boldsymbol{K}_1 = \begin{bmatrix} -a_0 & -a_1 & \cdots & -a_{n-1} \end{bmatrix} \otimes \boldsymbol{I}_{n_g} \tag{19}$$

The new equivalent system has the form of

$$\dot{\hat{\bar{\boldsymbol{X}}}} = \bar{\boldsymbol{A}}_2\hat{\bar{\boldsymbol{X}}} + \bar{\boldsymbol{B}}\boldsymbol{U}_2 \tag{20}$$

where

$$\boldsymbol{U}_2 = \boldsymbol{K}_1\hat{\bar{\boldsymbol{X}}} + \boldsymbol{U} \tag{21}$$

The consensus problem becomes finding a feedback control input $\boldsymbol{U}_2$ such that the system (20) is asymptotically stable.

In this paper, the cooperative control problem is formulated in an optimal control framework as follows:

$$\begin{aligned} &\min : J = J_1 + J_2 + J_3 \\ &s.t : \dot{\hat{\bar{\boldsymbol{X}}}} = \bar{\boldsymbol{A}}_2\hat{\bar{\boldsymbol{X}}} + \bar{\boldsymbol{B}}\boldsymbol{U}_2 \end{aligned} \tag{22}$$

J_1 is the cost for state deviations or consensus cost given by

$$J_1 = \int_0^\infty \left(\hat{\bar{\boldsymbol{X}}}^T\boldsymbol{R}_1\hat{\bar{\boldsymbol{X}}}\right)dt \tag{23}$$

where $\boldsymbol{R}_1$ is a ($n \cdot n_g \times n \cdot n_g$) diagonal and positive semi-definite (P.S.D) matrix. An approach for properly constructing this matrix will be shown afterwards in section 4.4.

J_2 is the control effort cost given by

$$J_2 = \int_0^{\infty} \left(\boldsymbol{U}_2^T \boldsymbol{R}_2 \boldsymbol{U}_2 \right) dt \tag{24}$$

where $\boldsymbol{R}_2 = w_c^2 \boldsymbol{I}_{n_g}$ is positive definite (P.D) and w_c is the weighting parameter.

J_3 is the cost for tracking and has the form of

$$J_3 = \int_0^{\infty} h(\hat{\hat{\boldsymbol{X}}}) dt \tag{25}$$

where $h(\hat{\hat{\boldsymbol{X}}})$ is constructed by an inverse optimal control approach and contains the tracking penalty function, which will be described in the next section.

2.4 Main Results

2.4.1 Optimal control solution

The following Lemma is used in this paper to prove the asymptotic stability and optimality of the proposed cooperative control law.

Lemma 2.1 [41]: Consider the nonlinear dynamical system

$$\dot{\hat{\hat{\boldsymbol{X}}}}(t) = \boldsymbol{f}\left(\hat{\hat{\boldsymbol{X}}}(t), \boldsymbol{U}(t) \right), \quad \hat{\hat{\boldsymbol{X}}}(0) = \hat{\hat{\boldsymbol{X}}}_0, \;\; t \geq 0 \tag{26}$$

with $\boldsymbol{f}\left(\boldsymbol{0},\boldsymbol{0}\right) = \boldsymbol{0}$ and a cost functional given by:

$$\boldsymbol{J}\left(\hat{\hat{\boldsymbol{X}}}_0, \boldsymbol{U}(\bullet) \right) \triangleq \int_{\boldsymbol{0}}^{\infty} \boldsymbol{T}\left(\hat{\hat{\boldsymbol{X}}}(t), \boldsymbol{U}(t) \right) dt \tag{27}$$

where $\boldsymbol{U}(\bullet)$ is an admissible control. Let $D \subseteq \mathbb{R}^n$ be an open set and $\Omega \subseteq \mathbb{R}^m$. Assume that there exists a continuously differentiable function $V: D \to \mathbb{R}$ and a control law $\phi: D \to \Omega$ such that

$$V(\boldsymbol{0}) = 0 \tag{28}$$

$$V(\hat{\bar{\boldsymbol{X}}}) > 0, \quad \hat{\bar{\boldsymbol{X}}} \in D, \; \hat{\bar{\boldsymbol{X}}} \neq \boldsymbol{0} \tag{29}$$

$$\phi(\boldsymbol{0}) = \boldsymbol{0} \tag{30}$$

$$V'(\hat{\bar{\boldsymbol{X}}})\, \boldsymbol{f}\left(\hat{\bar{\boldsymbol{X}}}, \phi(\hat{\bar{\boldsymbol{X}}})\right) < 0, \; \hat{\bar{\boldsymbol{X}}} \in D, \; \hat{\bar{\boldsymbol{X}}} \neq \boldsymbol{0} \tag{31}$$

$$H\left(\hat{\bar{\boldsymbol{X}}}, \phi(\hat{\bar{\boldsymbol{X}}})\right) = 0, \; \hat{\bar{\boldsymbol{X}}} \in D \tag{32}$$

$$H(\hat{\bar{\boldsymbol{X}}}, \boldsymbol{U}) \geq 0, \; \hat{\bar{\boldsymbol{X}}} \in D, \; \boldsymbol{U} \in \Omega \tag{33}$$

where $H(\hat{\bar{\boldsymbol{X}}}, \boldsymbol{U}) \triangleq T(\hat{\bar{\boldsymbol{X}}}, \boldsymbol{U}) + V'(\hat{\bar{\boldsymbol{X}}})\, \boldsymbol{f}(\hat{\bar{\boldsymbol{X}}}, \boldsymbol{U})$ is the Hamiltonian function. The superscript ' denotes partial differentiation with respect to $\hat{\bar{\boldsymbol{X}}}$.

Then, with the feedback control

$$\boldsymbol{U}(\bullet) = \phi(\hat{\bar{\boldsymbol{X}}}(\bullet)) \tag{34}$$

the solution $\hat{\bar{\boldsymbol{X}}} \equiv \boldsymbol{0}$ of the closed loop system is locally asymptotically stable and there exists a neighborhood of the origin $D_0 \subseteq D$ such that

$$\boldsymbol{J}\left(\hat{\bar{\boldsymbol{X}}}_0, \phi(\hat{\bar{\boldsymbol{X}}}(\bullet))\right) = V(\hat{\bar{\boldsymbol{X}}}_0), \quad \hat{\bar{\boldsymbol{X}}}_0 \in D_0 \tag{35}$$

In addition, if $\hat{\bar{\boldsymbol{X}}}_0 \in D_0$ then the feedback control (34) minimizes $\boldsymbol{J}(\hat{\bar{\boldsymbol{X}}}_0, \boldsymbol{U}(\bullet))$ in the sense that

$$\boldsymbol{J}\left(\hat{\bar{\boldsymbol{X}}}_0, \phi(\hat{\bar{\boldsymbol{X}}}(\bullet))\right) = \min_{\boldsymbol{U}(\bullet)\in S(\hat{\bar{\boldsymbol{X}}}_0)} \boldsymbol{J}(\hat{\bar{\boldsymbol{X}}}_0, \boldsymbol{U}(\bullet)) \tag{36}$$

where $S(\hat{\bar{\boldsymbol{X}}}_0)$ denotes the set of asymptotically stabilizing controllers for each initial condition $\hat{\bar{\boldsymbol{X}}}_0 \in D$. Finally, if $D = \mathbb{R}^n$, $\Omega = \mathbb{R}^m$ and

$$V(\hat{\bar{\boldsymbol{X}}}) \to \infty \text{ as } \left\|\hat{\bar{\boldsymbol{X}}}\right\| \to \infty \tag{37}$$

The solution $\hat{\bar{\boldsymbol{X}}}(0) \equiv \boldsymbol{0}$ of the closed loop system is globally asymptotically stable.

Proof: Refer to [41].

Before presenting the main theorem, we define the tracking penalty function

$$g(\hat{\bar{\boldsymbol{X}}}) \triangleq \sum_{i=1}^{n} g_{tr_i}(\hat{\bar{\boldsymbol{X}}}) \tag{38}$$

$$g_{tr_i}(\hat{\bar{\boldsymbol{X}}}) = \begin{cases} \left(\bar{\boldsymbol{X}}_i - \bar{\boldsymbol{X}}_D\right)^T \boldsymbol{G}\left(\bar{\boldsymbol{X}}_i - \bar{\boldsymbol{X}}_D\right) & \text{if the agent } i \text{ has access to the reference} \\ 0 & \text{if not} \end{cases} \tag{39}$$

where $\boldsymbol{G}$ is a positive semi-definite weighting matrix with tunable elements w_{d_i} constructed by

$$\boldsymbol{G} = \begin{bmatrix} {w_{d_1}}^2 & w_{d_1}.w_{d_2} & \cdots & w_{d_1}.w_{d_n} \\ w_{d_1}.w_{d_2} & {w_{d_2}}^2 & \cdots & w_{d_2}.w_{d_n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{d_1}.w_{d_n} & w_{d_2}.w_{d_n} & \cdots & {w_{d_n}}^2 \end{bmatrix} \tag{40}$$

and $\bar{\boldsymbol{X}}_D$ is the reference trajectory along which the system is expected to follow. Combining this tracking penalty function with the formation cost function J_1 enables the system of agents to follow a specified desired trajectory consensually. Note that only one agent having access to the reference is sufficient to guarantee that the entire system follows the desired trajectory if the communication topology is connected.

In order to investigate the eigenvalues of $\boldsymbol{G}$, the following lemma from exterior algebra is required.

Lemma 2.2 [83, 84]: Let R be a commutative ring and m be a positive integer. The characteristic polynomial of any $\boldsymbol{G} \in \boldsymbol{M}_m(R)$ can be written as:

$$\det(t\boldsymbol{I}_m - \boldsymbol{G}) = t^m + c_1(\boldsymbol{G})t^{m-1} + \ldots + c_{m-1}(\boldsymbol{G})t + c_m(\boldsymbol{G}) \tag{41}$$

where $c_k(\boldsymbol{G}) = (-1)^k \operatorname{tr}(\Lambda^k(\boldsymbol{G}))$. $\Lambda^k(\boldsymbol{G})$ is the kth exterior power of $\boldsymbol{G}$ and $\boldsymbol{M}_m(R)$ is the set of all the linear mapping with rank m. Furthermore, $\Lambda^1(\boldsymbol{G}) = \boldsymbol{G}$ and $\Lambda^k(\boldsymbol{G}) = 0$ for $k > r$ where r is the rank of $\boldsymbol{G}$.

Proof: Refer to [83, 84]. □

According to Lemma 2.2, the characteristic polynomial of $\boldsymbol{G}$ can be written as:

$$P_{\boldsymbol{G}}(t) = \sum_{k=0}^{m} t^{m-k}(-1)^k \operatorname{tr}(\Lambda^k(\boldsymbol{G})) \tag{42}$$

The matrix $\boldsymbol{G}$ is symmetric and its rank is $r = 1$ since each column is a product of any other column and a constant. Therefore $\Lambda^k(\boldsymbol{G}) = 0$ for any $k \geq 2$ and since $\Lambda^1\boldsymbol{G} = \boldsymbol{G}$, (42) is reduced to:

$$t^m - \left(\sum_{i=1}^{m}\left(w_{d_i}\right)^2\right)t^{m-1} = 0 \tag{43}$$

which implies that the set of eigenvalues contains $(m-1)$ zeros and $\sum_{i=1}^{m}\left(w_{d_i}\right)^2$, therefore $\boldsymbol{G}$ is positive semi-definite.

Before providing the main results, the following two Lemmas are introduced.

Lemma 2.3 $\boldsymbol{L}^2$ is positive semi-definite (P.S.D) and $\boldsymbol{L}^2\boldsymbol{1}_{n\times 1} = \boldsymbol{0}_{n\times 1}$ if the graph is undirected and connected.

Proof: Refer to [40]. □

Lemma 2.4 $\alpha_k \boldsymbol{L}^2 + \beta_k \boldsymbol{L}$ is positive definite (P.D) and $\left(\alpha_k \boldsymbol{L}^2 + \beta_k \boldsymbol{L}\right)\boldsymbol{1}_{n\times 1} = \boldsymbol{0}_{n\times 1}$ if the graph is undirected and connected and

$$\alpha_k e_i^2 + \beta_k e_i > 0 \tag{44}$$

where e_i is the ith eigenvalue of $\boldsymbol{L}$.

Proof: Since $\boldsymbol{L}$ is the Laplacian matrix of an undirected and connected graph, there exist two matrices $\boldsymbol{Q}$ and $\boldsymbol{\Lambda}$ such that:

$$\boldsymbol{L} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} \tag{45}$$

where $\boldsymbol{Q}$ is the matrix of eigenvectors of $\boldsymbol{L}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix whose entries are the eigenvalues of $\boldsymbol{L}$. For $\boldsymbol{L}^2$ we can write:

$$\boldsymbol{L}^2 = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^{-1} \tag{46}$$

Using the same approach, $\alpha_k \boldsymbol{L}^2 + \beta_k \boldsymbol{L}$ can be written as:

$$\begin{aligned}
\alpha_k \boldsymbol{L}^2 + \beta_k \boldsymbol{L} &= \alpha_k \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^{-1} + \beta_k \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} = \boldsymbol{Q}(\alpha_k \boldsymbol{\Lambda}^2 + \beta_k \boldsymbol{\Lambda})\boldsymbol{Q}^{-1} \\
&= \boldsymbol{Q}\begin{bmatrix} \alpha_k e_1^2 + \beta_k e_1 & 0 & \cdots & 0 \\ 0 & \alpha_k e_2^2 + \beta_k e_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_k e_n^2 + \beta_k e_n \end{bmatrix}\boldsymbol{Q}^{-1}
\end{aligned} \tag{47}$$

where e_i is the ith eigenvalue of $\boldsymbol{L}$. Since $\boldsymbol{L}$ is the Laplacian matrix for a connected and undirected graph, it is P.S.D and $e_i \geq 0$. Therefore $\alpha_k \boldsymbol{L}^2 + \beta_K \boldsymbol{L}$ is P.D if $\alpha_k e_i^2 + \beta_k e_i > 0$.

In addition,

$$\left(\alpha_k \boldsymbol{L}^2 + \beta_k \boldsymbol{L}\right)\boldsymbol{1}_{n\times 1} = \alpha_k \boldsymbol{L}^2\boldsymbol{1}_{n\times 1} + \beta_k \boldsymbol{L}\boldsymbol{1}_{n\times 1} = (\alpha_k \boldsymbol{L})\boldsymbol{L}\boldsymbol{1}_{n\times 1} + \beta_k \boldsymbol{L}\boldsymbol{1}_{n\times 1} = \boldsymbol{0}_{n\times 1} \tag{48}$$

□

The main result of this paper is presented in the following theorem.

Theorem 2.1 For n_g identical agents with the same controllable LTI dynamics (4), and connected and undirected communication topology, by choosing proper weights, the feedback control law

$$\boldsymbol{U}^* = \bar{\boldsymbol{K}}\boldsymbol{T}^{-1}\boldsymbol{X} - \frac{1}{2w_c^2}\frac{\partial}{\partial \boldsymbol{X}_{(n)}} g(\boldsymbol{X}) \tag{49}$$

makes the system (5) achieve consensus and track the reference trajectory, while minimizing the cost functional (22). $\boldsymbol{T}$ is the transformation matrix given by (9). $\boldsymbol{X}_{(n)} = \begin{bmatrix} X_{n1} & X_{n2} & \cdots & X_{nn_g} \end{bmatrix}^T$ is the vector of the last states of the agents. $\bar{\boldsymbol{K}}$ in (49) is defined by

$$\bar{\boldsymbol{K}} = \left(-\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{\boldsymbol{T}}\boldsymbol{P} - \boldsymbol{K}_1 \otimes \boldsymbol{I}_{n_g}\right) \tag{50}$$

where $\boldsymbol{K}_1$ is defined in (19) and $\boldsymbol{P}$ is the solution of the algebraic Riccati equation (ARE) that will be given in the proof.

The cost function $h(\hat{\bar{\boldsymbol{X}}})$ in J_3 in Eq. (25) is constructed as:

$$h(\hat{\bar{\boldsymbol{X}}}) = \frac{1}{4w_c^2}\left\| \frac{\partial}{\partial \hat{\bar{\boldsymbol{X}}}_{(n)}} g(\hat{\bar{\boldsymbol{X}}}) \right\| - g'^T(\hat{\bar{\boldsymbol{X}}})(\bar{\boldsymbol{A}}_2 - \boldsymbol{SP})\hat{\bar{\boldsymbol{X}}} \tag{51}$$

where $\boldsymbol{S} = \bar{\boldsymbol{B}}\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^T$ and $\frac{\partial}{\partial \hat{\bar{\boldsymbol{X}}}_{(n)}} g(\hat{\bar{\boldsymbol{X}}})$ is the derivative of the tracking penalty function with respect to the last states of the agents.

Proof: According to the definition of the cost functional (22) and Lemma 2.1, $T(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U}_2)$ and $\boldsymbol{f}(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U}_2)$ are defined as:

$$T(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U}_2) = \hat{\hat{\boldsymbol{X}}}^{\boldsymbol{T}}\boldsymbol{R}_1\hat{\hat{\boldsymbol{X}}} + \boldsymbol{U}_2^{\boldsymbol{T}}\boldsymbol{R}_2\boldsymbol{U}_2 + h(\hat{\hat{\boldsymbol{X}}}) \tag{52}$$

$$\boldsymbol{f}(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U}_2) = \bar{\boldsymbol{A}}_2\hat{\hat{\boldsymbol{X}}} + \bar{\boldsymbol{B}}\boldsymbol{U}_2 \tag{53}$$

A Lyapunov function is chosen as:

$$V(\hat{\hat{\boldsymbol{X}}}) = \hat{\hat{\boldsymbol{X}}}^{\boldsymbol{T}}\boldsymbol{P}\hat{\hat{\boldsymbol{X}}} + g(\hat{\hat{\boldsymbol{X}}}) \tag{54}$$

For $V(\hat{\hat{\boldsymbol{X}}})$ to be a valid Lyapunov function, it should be continuously differentiable with respect to $\hat{\hat{\boldsymbol{X}}}$, which is obvious from the definition of $g(\hat{\hat{\boldsymbol{X}}})$.

The Hamiltonian function is constructed as

$$\begin{aligned} H(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U}_2,V'^{T}(\hat{\hat{\boldsymbol{X}}})) &= T(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U}_2) + V'^{T}(\hat{\hat{\boldsymbol{X}}})\,\boldsymbol{f}(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U}_2) \\ &= \hat{\hat{\boldsymbol{X}}}^{T}\boldsymbol{R}_1\hat{\hat{\boldsymbol{X}}} + \boldsymbol{U}_2^{T}\boldsymbol{R}_2\boldsymbol{U}_2 + h(\hat{\hat{\boldsymbol{X}}}) + \left(2\hat{\hat{\boldsymbol{X}}}^{T}\boldsymbol{P} + g'^{T}(\hat{\hat{\boldsymbol{X}}})\right)\left(\bar{\boldsymbol{A}}_2\hat{\hat{\boldsymbol{X}}} + \bar{\boldsymbol{B}}\boldsymbol{U}_2\right) \end{aligned} \tag{55}$$

Taking the derivate with respect to $\boldsymbol{U}_2$, the optimal control is determined as:

$$\begin{aligned} &\frac{\partial H}{\partial \boldsymbol{U}_2} = 0 \Rightarrow 2\boldsymbol{R}_2\boldsymbol{U}_2 + 2(\bar{\boldsymbol{B}}^{\boldsymbol{T}}\boldsymbol{P}\hat{\hat{\boldsymbol{X}}}) + \bar{\boldsymbol{B}}^{T}g'(\hat{\hat{\boldsymbol{X}}}) = 0 \\ &\Rightarrow \boldsymbol{U}_2^{*} = \phi(\hat{\hat{\boldsymbol{X}}}) = -\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{\boldsymbol{T}}\boldsymbol{P}\hat{\hat{\boldsymbol{X}}} - \frac{1}{2}\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{T}g'(\hat{\hat{\boldsymbol{X}}}) \end{aligned} \tag{56}$$

Next, all the conditions from (28) to (33) must be satisfied. Substituting $\boldsymbol{U}_2^{*}$ in the second term of the Hamiltonian function (55) yields:

$$\begin{aligned} &V'^{T}(\hat{\hat{\boldsymbol{X}}})\,\boldsymbol{f}(\hat{\hat{\boldsymbol{X}}},\phi(\hat{\hat{\boldsymbol{X}}})) = \hat{\hat{\boldsymbol{X}}}^{T}(\bar{\boldsymbol{A}}_2^{T}\boldsymbol{P} + \boldsymbol{P}\bar{\boldsymbol{A}}_2 - 2\boldsymbol{PSP})\hat{\hat{\boldsymbol{X}}} - \hat{\hat{\boldsymbol{X}}}^{T}\boldsymbol{PS}g'(\hat{\hat{\boldsymbol{X}}}) \\ &+ g'^{T}(\hat{\hat{\boldsymbol{X}}})(\bar{\boldsymbol{A}}_2 - \boldsymbol{SP})\hat{\hat{\boldsymbol{X}}} - \frac{1}{2}g'^{T}(\hat{\hat{\boldsymbol{X}}})\boldsymbol{S}g'(\hat{\hat{\boldsymbol{X}}}) \end{aligned} \tag{57}$$

where $\boldsymbol{S} \triangleq \bar{\boldsymbol{B}}\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{\boldsymbol{T}}$. Note that the term $\hat{\hat{\boldsymbol{X}}}^{\boldsymbol{T}}(2\boldsymbol{P}\bar{\boldsymbol{A}}_2)\hat{\hat{\boldsymbol{X}}}$ in (55) is a scalar and can be rewritten as $\hat{\hat{\boldsymbol{X}}}^{\boldsymbol{T}}(\bar{\boldsymbol{A}}_2^{\boldsymbol{T}}\boldsymbol{P} + \boldsymbol{P}\bar{\boldsymbol{A}}_2)\hat{\hat{\boldsymbol{X}}}$ in (57). The Hamiltonian function with the optimal control $\phi(\hat{\hat{\boldsymbol{X}}})$ becomes

$$\begin{gathered} H(\hat{\hat{\boldsymbol{X}}},\phi(\hat{\hat{\boldsymbol{X}}}),V'^{T}(\hat{\hat{\boldsymbol{X}}})) = \hat{\hat{\boldsymbol{X}}}^{T}(\bar{\boldsymbol{A}}_2^{T}\boldsymbol{P} + \boldsymbol{P}\bar{\boldsymbol{A}}_2 + \boldsymbol{R}_1 - \boldsymbol{PSP})\hat{\hat{\boldsymbol{X}}} + g'^{T}(\hat{\hat{\boldsymbol{X}}})(\bar{\boldsymbol{A}}_2 - \boldsymbol{SP})\hat{\hat{\boldsymbol{X}}} \\ +h(\hat{\hat{\boldsymbol{X}}}) - \frac{1}{4}g'^{T}(\hat{\hat{\boldsymbol{X}}})\boldsymbol{S}g'(\hat{\hat{\boldsymbol{X}}}) \end{gathered} \tag{58}$$

In order to satisfy the condition (32), Eq. (58) should be zero, which requires that:

$$\bar{\boldsymbol{A}}_2^{T}\boldsymbol{P} + \boldsymbol{P}\bar{\boldsymbol{A}}_2 + \boldsymbol{R}_1 - \boldsymbol{PSP} = \boldsymbol{0} \tag{59}$$

and

$$g'^{T}(\hat{\hat{\boldsymbol{X}}})(\bar{\boldsymbol{A}}_2 - \boldsymbol{SP})\hat{\hat{\boldsymbol{X}}} + h(\hat{\hat{\boldsymbol{X}}}) - \frac{1}{4}g'^{T}(\hat{\hat{\boldsymbol{X}}})\boldsymbol{S}g'(\hat{\hat{\boldsymbol{X}}}) = 0 \tag{60}$$

The importance of Eq. (60) is that it allows us to determine $h(\hat{\hat{\boldsymbol{X}}})$ and thus the cost function J_3.

Eq. (59) is an algebraic Riccati equation (ARE). In order to solve the ARE, we need to construct $\boldsymbol{R}_1$ such that it is P.S.D and $\boldsymbol{P}$ is P.D. Here we construct this matrix as follows:

$$\boldsymbol{R}_1 = \begin{bmatrix} w_1^2\boldsymbol{L}^2 & \boldsymbol{0}_{n_g\times n_g} & \boldsymbol{0}_{n_g\times n_g} & \cdots & \boldsymbol{0}_{n_g\times n_g} \\ \boldsymbol{0}_{n_g\times n_g} & w_2^2\boldsymbol{L}^2 - 2\boldsymbol{P}_{12} & \boldsymbol{0}_{n_g\times n_g} & \cdots & \boldsymbol{0}_{n_g\times n_g} \\ \boldsymbol{0}_{n_g\times n_g} & \boldsymbol{0}_{n_g\times n_g} & w_3^2\boldsymbol{L}^2 - 2\boldsymbol{P}_{23} & \cdots & \boldsymbol{0}_{n_g\times n_g} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}_{n_g\times n_g} & \boldsymbol{0}_{n_g\times n_g} & \boldsymbol{0}_{n_g\times n_g} & \cdots & w_n^2\boldsymbol{L}^2 - 2\boldsymbol{P}_{(n-1)n} \end{bmatrix}_{n\cdot n_g\times n\cdot n_g} \tag{61}$$

where w_i's are the tunable weights for the ith state and $\boldsymbol{P}_{(i-1)i}$, $1 \le i \le n$ are the entries above the main diagonal of the matrix $\boldsymbol{P}$ (the solution of the ARE (59)). These terms are subtracted from the diagonal entries of $\boldsymbol{R}_1$ to guarantee that ARE becomes a linear function of the Laplacian matrix $\boldsymbol{L}$. Expanding (59) leads to:

$$\left(\begin{bmatrix} \mathbf{0}_{n_g\times n_g} & \mathbf{0}_{n_g\times n_g} & \cdots & \mathbf{0}_{n_g\times n_g} \\ \mathbf{P}_{11} & \mathbf{P}_{12} & \cdots & \mathbf{P}_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{1(n-1)} & \mathbf{P}_{2(n-1)} & \cdots & \mathbf{P}_{n(n-1)} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n_g\times n_g} & \mathbf{P}_{11} & \cdots & \mathbf{P}_{1(n-1)} \\ \mathbf{0}_{n_g\times n_g} & \mathbf{P}_{12} & \cdots & \mathbf{P}_{2(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{n_g\times n_g} & \mathbf{P}_{1n} & \cdots & \mathbf{P}_{n(n-1)} \end{bmatrix} \\ + \begin{bmatrix} w_1^2\mathbf{L}^2 & \mathbf{0}_{n_g\times n_g} & \cdots & \mathbf{0}_{n_g\times n_g} \\ \mathbf{0}_{n_g\times n_g} & w_2^2\mathbf{L}^2 - 2\mathbf{P}_{12} & \cdots & \mathbf{0}_{n_g\times n_g} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{n_g\times n_g} & \mathbf{0}_{n_g\times n_g} & \cdots & w_n^2\mathbf{L}^2 - 2\mathbf{P}_{(n-1)n} \end{bmatrix} - \frac{1}{w_c^2}\begin{bmatrix} \mathbf{P}_{1n}^2 & \mathbf{P}_{1n}\mathbf{P}_{2n} & \cdots & \mathbf{P}_{1n}\mathbf{P}_{nn} \\ \mathbf{P}_{1n}\mathbf{P}_{nn} & \mathbf{P}_{2n}^2 & \cdots & \mathbf{P}_{2n}\mathbf{P}_{nn} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{1n}\mathbf{P}_{nn} & \mathbf{P}_{2n}\mathbf{P}_{nn} & \cdots & \mathbf{P}_{nn}^2 \end{bmatrix}\right) = \mathbf{0}_{n\cdot n_g\times n\cdot n_g} \quad (62)$$

The system of algebraic equations (62) allows us to solve the ARE analytically. $\mathbf{P}$ is determined by the following steps.

1) The equations for the entries on the main diagonal are first solved:

$$\begin{cases} w_1^2\mathbf{L}^2 - \dfrac{1}{w_c^2}\mathbf{P}_{1n}^2 = \mathbf{0}_{n_g\times n_g}, & i = 1 \\ 2\mathbf{P}_{(i-1)i} + w_i^2\mathbf{L}^2 - 2\mathbf{P}_{(i-1)i} - \dfrac{1}{w_c^2}\mathbf{P}_{in}^2 = \mathbf{0}_{n_g\times n_g}, & 2 \le i \le n \end{cases} \quad (63)$$

and therefore:

$$\mathbf{P}_{in} = w_i w_c \mathbf{L}, \quad 1 \le i \le n \quad (64)$$

2) The equations for each row in the system (62) are solved one by one. For example, for the first row we derive the following equation:

$$\mathbf{P}_{1i} = w_1 w_{(i+1)}\mathbf{L}^2 \quad 1 \le i \le n-1 \quad (65)$$

Then all the row equations are solved in a similar way to obtain all the entries of $\mathbf{P}$.

Note that all the entries are obtained analytically without any numerical iteration.

From the above solutions, it is seen that all the entries of $\mathbf{P}$ can be written as a linear combination of $\mathbf{L}$ and $\mathbf{L}^2$ (i.e. $\alpha_k\mathbf{L}^2 + \beta_k\mathbf{L}$ where both α_k and β_k are functions of state and control weighting parameters). Therefore, according to Lemma 4.4, one can always choose weighting parameters such that $\boldsymbol{R}_1$ is P.S.D and $\boldsymbol{P}$ is P.D.

Since $\boldsymbol{P}$ is P.D and $\boldsymbol{G}$ is P.S.D, $V(\hat{\hat{\boldsymbol{X}}}) > 0, \quad \hat{\hat{\boldsymbol{X}}} \in D, \ \hat{\hat{\boldsymbol{X}}} \neq \boldsymbol{0}$ and condition (29) is satisfied.

The condition (33) can be shown to hold as follows:

$$
\begin{aligned}
H(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U}_2,V'^{\boldsymbol{T}}(\hat{\hat{\boldsymbol{X}}})) = {} & \boldsymbol{U}_2{}^T\boldsymbol{R}_2\boldsymbol{U}_2 + h(\hat{\hat{\boldsymbol{X}}}) + \hat{\hat{\boldsymbol{X}}}{}^T\boldsymbol{R}_1\hat{\hat{\boldsymbol{X}}} + (2\hat{\hat{\boldsymbol{X}}}{}^T\boldsymbol{P} + g'^T(\hat{\hat{\boldsymbol{X}}}))(\bar{\boldsymbol{A}}_2\hat{\hat{\boldsymbol{X}}} + \bar{\boldsymbol{B}}\boldsymbol{U}_2) \\
& = \boldsymbol{U}_2{}^T\boldsymbol{R}_2\boldsymbol{U}_2 + h(\hat{\hat{\boldsymbol{X}}}) + \hat{\hat{\boldsymbol{X}}}{}^T\boldsymbol{R}_1\hat{\hat{\boldsymbol{X}}} + (2\hat{\hat{\boldsymbol{X}}}{}^T\boldsymbol{P} + g'^T(\hat{\hat{\boldsymbol{X}}}))(\bar{\boldsymbol{A}}_2\hat{\hat{\boldsymbol{X}}} + \bar{\boldsymbol{B}}\boldsymbol{U}_2) \\
& \quad + \hat{\hat{\boldsymbol{X}}}{}^T(\bar{\boldsymbol{A}}_2^{\boldsymbol{T}}\boldsymbol{P} + \boldsymbol{P}\bar{\boldsymbol{A}}_2 + \boldsymbol{R}_1 - \boldsymbol{PSP})\hat{\hat{\boldsymbol{X}}} \\
& = \boldsymbol{U}_2{}^T\boldsymbol{R}_2\boldsymbol{U}_2 + h(\hat{\hat{\boldsymbol{X}}}) + g'^T(\hat{\hat{\boldsymbol{X}}})(\bar{\boldsymbol{A}}_2\hat{\hat{\boldsymbol{X}}} + \bar{\boldsymbol{B}}\boldsymbol{U}_2) + 2\hat{\hat{\boldsymbol{X}}}{}^T\boldsymbol{P}\bar{\boldsymbol{B}}\boldsymbol{U}_2 + \hat{\hat{\boldsymbol{X}}}{}^T\boldsymbol{PSP}\hat{\hat{\boldsymbol{X}}} \\
& = \boldsymbol{U}_2{}^T\boldsymbol{R}_2\boldsymbol{U}_2 + \frac{1}{4}g'^T(\hat{\hat{\boldsymbol{X}}})\boldsymbol{S}g'(\hat{\hat{\boldsymbol{X}}}) + g'^T(\hat{\hat{\boldsymbol{X}}})\boldsymbol{SP}\hat{\hat{\boldsymbol{X}}} \\
& \quad + \hat{\hat{\boldsymbol{X}}}{}^T\boldsymbol{PSP}\hat{\hat{\boldsymbol{X}}} + (2\hat{\hat{\boldsymbol{X}}}{}^T\boldsymbol{P} + g'^T(\hat{\hat{\boldsymbol{X}}}))\bar{\boldsymbol{B}}\boldsymbol{U}_2 \\
& = \boldsymbol{U}_2{}^T\boldsymbol{R}_2\boldsymbol{U}_2 + \frac{1}{4}(2\hat{\hat{\boldsymbol{X}}}{}^T\boldsymbol{P} + g'^T(\hat{\hat{\boldsymbol{X}}}))\boldsymbol{S}(2\hat{\hat{\boldsymbol{X}}}{}^T\boldsymbol{P} + g'^T(\hat{\hat{\boldsymbol{X}}}))^T + (2\hat{\hat{\boldsymbol{X}}}{}^T\boldsymbol{P} + g'^T(\hat{\hat{\boldsymbol{X}}}))\bar{\boldsymbol{B}}\boldsymbol{U}_2
\end{aligned}
\tag{66}
$$

Since $\phi = -\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{\boldsymbol{T}}\boldsymbol{P}\hat{\hat{\boldsymbol{X}}} - \frac{1}{2}\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^T g'(\hat{\hat{\boldsymbol{X}}})$ from (56), we can write:

$$
\begin{aligned}
& 2\phi = -\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{\boldsymbol{T}}(2\boldsymbol{P}\hat{\hat{\boldsymbol{X}}} + g') \Rightarrow 2\boldsymbol{R}_2\phi = -\bar{\boldsymbol{B}}^T(2\boldsymbol{P}\hat{\hat{\boldsymbol{X}}} + g') = -\bar{\boldsymbol{B}}^T V' \\
& \Rightarrow \begin{cases} \bar{\boldsymbol{B}}^{\boldsymbol{T}}V' = -2\boldsymbol{R}_2\phi \\ V'^T\,\bar{\boldsymbol{B}} = -2\phi^T\boldsymbol{R}_2 \end{cases}
\end{aligned}
\tag{67}
$$

Using (67), (66) becomes

$$
\begin{aligned}
& \boldsymbol{U}_2^T\boldsymbol{R}_2\boldsymbol{U}_2 + \frac{1}{4}V'^{\boldsymbol{T}}\,\boldsymbol{S}V' + \boldsymbol{U}_2^T(\bar{\boldsymbol{B}}^T V') \\
& = \boldsymbol{U}_2^T\boldsymbol{R}_2\boldsymbol{U}_2 + \frac{1}{4}V'^T(\bar{\boldsymbol{B}}\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{\boldsymbol{T}})V' + \boldsymbol{U}_2^T\bar{\boldsymbol{B}}^T V' \\
& = \boldsymbol{U}_2^{\boldsymbol{T}}\boldsymbol{R}_2\boldsymbol{U}_2 + \frac{1}{4}(-2\phi^T\boldsymbol{R}_2)\boldsymbol{R}_2^{-1}(-2\boldsymbol{R}_2\phi) + \boldsymbol{U}_2^T\bar{\boldsymbol{B}}^T V' \\
& = \boldsymbol{U}_2^T\boldsymbol{R}_2\boldsymbol{U}_2 + \phi^T\boldsymbol{R}_2\phi + \boldsymbol{U}_2^T(-2\boldsymbol{R}_2\phi) \\
& = \boldsymbol{U}_2^T\boldsymbol{R}_2\boldsymbol{U}_2 + \phi^T\boldsymbol{R}_2\phi - 2\boldsymbol{U}_2^T\boldsymbol{R}_2\phi = \left(\boldsymbol{U}_2 - \phi(\hat{\hat{\boldsymbol{X}}})\right)^T\boldsymbol{R}_2\left(\boldsymbol{U}_2 - \phi(\hat{\hat{\boldsymbol{X}}})\right) \geq \boldsymbol{0}
\end{aligned}
\tag{68}
$$

Therefore, (68) verifies the condition (33), i.e. $H(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U}_2,V'^{\boldsymbol{T}}(\hat{\hat{\boldsymbol{X}}})) \geq 0$

Using (59) and (60), the condition (31) can be shown to hold as follows

$$V'^{\boldsymbol{T}}(\hat{\hat{\boldsymbol{X}}})\,\boldsymbol{f}(\hat{\hat{\boldsymbol{X}}},\phi(\hat{\hat{\boldsymbol{X}}})) = -\left[\hat{\hat{\boldsymbol{X}}}^T\boldsymbol{R}_1\hat{\hat{\boldsymbol{X}}} + h(\hat{\hat{\boldsymbol{X}}}) + (\hat{\hat{\boldsymbol{X}}}^T\boldsymbol{P} + \frac{1}{2}g'^T(\hat{\hat{\boldsymbol{X}}}))\boldsymbol{S}(\boldsymbol{P}\hat{\hat{\boldsymbol{X}}} + \frac{1}{2}g'(\hat{\hat{\boldsymbol{X}}}))\right] \tag{69}$$

Since $\hat{\hat{\boldsymbol{X}}}^T\boldsymbol{R}_1\hat{\hat{\boldsymbol{X}}}$ is P.S.D and $(\hat{\hat{\boldsymbol{X}}}^T\boldsymbol{P} + \frac{1}{2}g'^T(\hat{\hat{\boldsymbol{X}}}))\boldsymbol{S}(\boldsymbol{P}\hat{\hat{\boldsymbol{X}}} + \frac{1}{2}g'(\hat{\hat{\boldsymbol{X}}}))$ is P.D., $V'^{\boldsymbol{T}}(\hat{\hat{\boldsymbol{X}}})\,\boldsymbol{f}(\hat{\hat{\boldsymbol{X}}},\phi(\hat{\hat{\boldsymbol{X}}})) \le 0$ if $h(\hat{\hat{\boldsymbol{X}}}) \ge 0$. From equation (60), $h(\hat{\hat{\boldsymbol{X}}})$ is derived as

$$h(\hat{\hat{\boldsymbol{X}}}) = \frac{1}{4}g'^T(\hat{\hat{\boldsymbol{X}}})\boldsymbol{S}g'(\hat{\hat{\boldsymbol{X}}}) - g'^T(\hat{\hat{\boldsymbol{X}}})(\bar{\boldsymbol{A}}_2 - \boldsymbol{SP})\hat{\hat{\boldsymbol{X}}}$$

$$= \frac{1}{4w_c^2}\left\|\frac{\partial}{\partial\hat{\hat{\boldsymbol{X}}}_{(n)}}g(\hat{\hat{\boldsymbol{X}}})\right\| + \frac{\partial g}{\partial\hat{\hat{\boldsymbol{X}}}_{(n)}}\begin{bmatrix}\frac{w_1}{w_c}\boldsymbol{L} & \frac{w_2}{w_c}\boldsymbol{L} & \cdots & \frac{w_n}{w_c}\boldsymbol{L}\end{bmatrix}\begin{bmatrix}\hat{\hat{\boldsymbol{X}}}_{(1)}\\ \hat{\hat{\boldsymbol{X}}}_{(2)}\\ \vdots\\ \hat{\hat{\boldsymbol{X}}}_{(n)}\end{bmatrix} - \begin{bmatrix}\frac{\partial g}{\partial\hat{\hat{\boldsymbol{X}}}_{(1)}} & \frac{\partial g}{\partial\hat{\hat{\boldsymbol{X}}}_{(2)}} & \cdots & \frac{\partial g}{\partial\hat{\hat{\boldsymbol{X}}}_{(n-1)}}\end{bmatrix}\begin{bmatrix}\hat{\hat{\boldsymbol{X}}}_{(2)}\\ \hat{\hat{\boldsymbol{X}}}_{(3)}\\ \vdots\\ \hat{\hat{\boldsymbol{X}}}_{(n)}\end{bmatrix} \tag{70}$$

where $\hat{\hat{\boldsymbol{X}}}_{(k)}$ is the vector of the kth state of all the agents. Eq. (70) shows that one can always find proper weights such that $h(\hat{\hat{\boldsymbol{X}}}) \ge 0$. Specifically, for a given set of state weighting parameters (i.e. $w_1, w_2, ..., w_n$) the control weight parameter can be chosen small enough such that the positive term $\frac{1}{4w_c^2}\left\|\frac{\partial}{\partial\hat{\hat{\boldsymbol{X}}}_{(n)}}g(\hat{\hat{\boldsymbol{X}}})\right\|$ is always greater than the other terms that are sign-indefinite.

The conditions (28) and (30) are satisfied if $g(\hat{\hat{\boldsymbol{X}}}) = 0$ and $g'(\hat{\hat{\boldsymbol{X}}}) = \boldsymbol{0}$ when $\hat{\hat{\boldsymbol{X}}} = \boldsymbol{0}$. Note that after reaching consensus, when the agents go along a desired trajectory, we have: $\bar{\boldsymbol{X}}_{cs} = \bar{\boldsymbol{X}}_D$, which indicates $\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}}_D$ when $\hat{\hat{\boldsymbol{X}}} = \boldsymbol{0}$. Therefore, according to the definition of $g(\hat{\hat{\boldsymbol{X}}})$, $V(\hat{\hat{\boldsymbol{X}}}) = 0$ when $\hat{\hat{\boldsymbol{X}}} = \boldsymbol{0}$, and the condition (28) holds.

The second term in the optimal control (56) is calculated as

$$\frac{\partial g}{\partial \hat{\bar{\boldsymbol{X}}}_{(n)}} = \begin{cases} \left[\frac{\partial}{\partial \hat{\bar{\boldsymbol{X}}}_{(n)}} (\bar{\boldsymbol{X}}_i - \bar{\boldsymbol{X}}_D) \right] (\boldsymbol{G} + \boldsymbol{G}^T)(\bar{\boldsymbol{X}}_i - \bar{\boldsymbol{X}}_D) & \text{if the agent } i \text{ has access to the reference} \\ 0 & \text{if not} \end{cases}$$
$$= \begin{cases} \begin{bmatrix} 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{n_g \times n} (\boldsymbol{G} + \boldsymbol{G}^T)(\bar{\boldsymbol{X}}_i - \bar{\boldsymbol{X}}_D) & \text{if the agent } i \text{ has access to the reference} \\ 0 & \text{if not} \end{cases} \tag{71}$$

It is obvious that when all the agents go along the desired trajectory, (71) is zero and the condition (30) holds.

Now all the conditions (28)-(33) in Lemma 2.1 are satisfied and therefore, the control law $\phi(\hat{\bar{\boldsymbol{X}}})$ is an optimal control to minimize (22). In the meanwhile, the closed-loop system (20) is asymptotically stable. Furthermore, it is evident that the Lyapunov function (54) satisfies $V(\hat{\bar{\boldsymbol{X}}}) \to \infty$ as $\left\| \hat{\bar{\boldsymbol{X}}} \right\| \to \infty$. Thus, the closed-loop system is globally asymptotically stable.

The optimal control law for system (14) can be obtained from (21) as

$$\begin{aligned} \boldsymbol{U}^* &= (-\boldsymbol{R}_2^{-1} \bar{\boldsymbol{B}}^T \boldsymbol{P} - \boldsymbol{K}_1) \hat{\bar{\boldsymbol{X}}} - \frac{1}{2} \boldsymbol{R}_2^{-1} \bar{\boldsymbol{B}}^T g'(\hat{\bar{\boldsymbol{X}}}) \\ &= (-\boldsymbol{R}_2^{-1} \bar{\boldsymbol{B}}^T \boldsymbol{P} - [-a_0 \quad -a_1 \quad \cdots \quad -a_{n-1}] \otimes \boldsymbol{I}_{ng}) \hat{\bar{\boldsymbol{X}}} - \frac{1}{2} \boldsymbol{R}_2^{-1} \bar{\boldsymbol{B}}^T g'(\hat{\bar{\boldsymbol{X}}}) \\ &= \bar{\boldsymbol{K}} \hat{\bar{\boldsymbol{X}}} - \frac{1}{2} \boldsymbol{R}_2^{-1} \bar{\boldsymbol{B}}^T g'(\hat{\bar{\boldsymbol{X}}}) \end{aligned} \tag{72}$$

where $g'(\hat{\bar{\boldsymbol{X}}}) = \begin{bmatrix} \frac{\partial g}{\partial \hat{\bar{\boldsymbol{X}}}_{(1)}} & \frac{\partial g}{\partial \hat{\bar{\boldsymbol{X}}}_{(2)}} & \cdots & \frac{\partial g}{\partial \hat{\bar{\boldsymbol{X}}}_{(n)}} \end{bmatrix}^T$ and $\boldsymbol{X}_{(i)} = \begin{bmatrix} X_{i1} & X_{i2} & \cdots & X_{in_g} \end{bmatrix}^T$) .

Since $\boldsymbol{R}_2^{-1} = \frac{1}{w_c^2}\boldsymbol{I}_n$ and $\bar{\boldsymbol{B}} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^T_{1\times n} \otimes \boldsymbol{I}_{n_g}$, the second term of the controller (72) can be written as

$$\frac{1}{2}\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^T g'(\hat{\bar{\boldsymbol{X}}}) = \frac{1}{2w_c^2}\frac{\partial}{\partial\hat{\bar{\boldsymbol{X}}}_{(n)}} g(\hat{\bar{\boldsymbol{X}}}) \tag{73}$$

(72) is the optimal control for the system (14), which is equivalent to the system (20). The optimal cooperative control law can be written in a more compact form as

$$\boldsymbol{U}^* = \bar{\boldsymbol{K}}\boldsymbol{T}^{-1}\hat{\boldsymbol{X}} - \frac{1}{2w_c^2}\frac{\partial}{\partial\hat{\bar{\boldsymbol{X}}}_{(n)}} g(\hat{\bar{\boldsymbol{X}}}) \tag{74}$$

and $\bar{\boldsymbol{K}}$ is calculated by (50).

Applying the control (74) into Eq. (14) leads to

$$\begin{gathered}\dot{\hat{\boldsymbol{X}}} = \boldsymbol{A}\hat{\boldsymbol{X}} + \boldsymbol{B}\left(\bar{\boldsymbol{K}}\boldsymbol{T}^{-1}\hat{\boldsymbol{X}} - \frac{1}{2w_c^2}\frac{\partial}{\partial\hat{\bar{\boldsymbol{X}}}_{(n)}} g(\hat{\bar{\boldsymbol{X}}})\right) \\ \Rightarrow \dot{\hat{\boldsymbol{X}}} = \left(\boldsymbol{A} + \boldsymbol{B}\bar{\boldsymbol{K}}\boldsymbol{T}^{-1}\right)\hat{\boldsymbol{X}} - \left(\frac{1}{2w_c^2}\right)\boldsymbol{B}\frac{\partial}{\partial\hat{\bar{\boldsymbol{X}}}_{(n)}} g(\hat{\bar{\boldsymbol{X}}})\end{gathered} \tag{75}$$

Replacing $\hat{\boldsymbol{X}}$ with $\boldsymbol{X} - \boldsymbol{X}_{cs}$, we have

$$\begin{gathered}\left(\dot{\boldsymbol{X}} - \dot{\boldsymbol{X}}_{cs}\right) = \left(\boldsymbol{A} + \boldsymbol{B}\bar{\boldsymbol{K}}\boldsymbol{T}^{-1}\right)\left(\boldsymbol{X} - \boldsymbol{X}_{cs}\right) - \left(\frac{1}{2w_c^2}\right)\boldsymbol{B}\frac{\partial}{\partial\hat{\bar{\boldsymbol{X}}}_{(n)}} g(\hat{\bar{\boldsymbol{X}}}) \\ \Rightarrow \dot{\boldsymbol{X}} - \dot{\boldsymbol{X}}_{cs} = \boldsymbol{A}\boldsymbol{X} - \boldsymbol{A}\boldsymbol{X}_{cs} + \boldsymbol{B}\bar{\boldsymbol{K}}\boldsymbol{T}^{-1}\boldsymbol{X} - \boldsymbol{B}\bar{\boldsymbol{K}}\boldsymbol{T}^{-1}\boldsymbol{X}_{cs} - \left(\frac{1}{2w_c^2}\right)\boldsymbol{B}\frac{\partial}{\partial\hat{\bar{\boldsymbol{X}}}_{(n)}} g(\hat{\bar{\boldsymbol{X}}})\end{gathered} \tag{76}$$

According to (6), $\dot{\boldsymbol{X}}_{cs} = \boldsymbol{A}\boldsymbol{X}_{cs}$ and $\boldsymbol{U}_{cs} = \bar{\boldsymbol{K}}\boldsymbol{T}^{-1}\boldsymbol{X}_{cs} = \boldsymbol{0}_{n\cdot n_g\times 1}$. Thus, (76) is reduced to

$$\dot{\boldsymbol{X}} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\bar{\boldsymbol{K}}\boldsymbol{T}^{-1}\boldsymbol{X} - \left(\frac{1}{2w_c^2}\right)\boldsymbol{B}\frac{\partial}{\partial\hat{\bar{\boldsymbol{X}}}_{(n)}} g(\hat{\bar{\boldsymbol{X}}}) \tag{77}$$

$\frac{\partial g}{\partial \hat{\bar{\boldsymbol{X}}}_{(n)}}$ is calculated in (71). Since $\boldsymbol{X}_D$ is known to the agent *i* that has access to the desired trajectory, $\frac{\partial}{\partial \hat{\bar{\boldsymbol{X}}}_{(n)}} \boldsymbol{g}(\hat{\bar{\boldsymbol{X}}}) = \frac{\partial}{\partial \bar{\boldsymbol{X}}_{(n)}} \boldsymbol{g}(\bar{\boldsymbol{X}})$. Now the vector $(\bar{\boldsymbol{X}}_i - \bar{\boldsymbol{X}}_D)$ can be converted to the original state.

$$\frac{\partial g}{\partial \boldsymbol{X}_{(n)}} = \begin{cases} \begin{bmatrix} 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix}_{n_g \times n} (\boldsymbol{G} + \boldsymbol{G}^T)\boldsymbol{T}^{-1}(\boldsymbol{X}_i - \boldsymbol{X}_D) & \text{if the agent } i \text{ has access to the reference} \\ 0 & \text{if not} \end{cases} \tag{78}$$

Using (78), the optimal control law becomes (49)

$$\boldsymbol{U}^* = \bar{\boldsymbol{K}}\boldsymbol{T}^{-1}\boldsymbol{X} - \frac{1}{2w_c^2}\frac{\partial}{\partial \boldsymbol{X}_{(n)}} g(\boldsymbol{X})$$

and $\bar{\boldsymbol{K}}$ is defined by (50). □

Remark 2.1 In the conventional optimal control approach, the cost functional is given *a priori* and the optimal control law is derived by minimizing it. It can be seen that the approach described in Theorem 2.1 is an inverse optimal control approach. A Lyapunov function $V(\hat{\bar{\boldsymbol{X}}})$ is constructed first based on the stability conditions (28) and (29) and then the optimal control law $\phi(\hat{\bar{\boldsymbol{X}}})$ is derived from the optimality condition $\frac{\partial H}{\partial \boldsymbol{U}} = 0$. By satisfying the optimality condition (32), $h(\hat{\bar{\boldsymbol{X}}})$ in the cost functional J_3 is constructed while satisfying the stability condition (31). In other word, for this approach, the cost function $h(\hat{\bar{\boldsymbol{X}}})$ is not specified *a priori* as the conventional optimal control design. It is constructed inversely from the stability and optimality conditions (28) to (33). The benefit of this design is that the resulting control law is guaranteed to be both stabilizing and optimal. In addition, according to the Lemma 4.1, the minimum cost made by the derived control law $\phi(\hat{\bar{\boldsymbol{X}}})$ is equal to the introduced Lyapunov function as shown in (35).

2.4.2 Discussion on the distributed cooperative control law

It is necessary to show that the optimal cooperative control law (49) is a distributive control law. To this end, we first investigate the first term of the optimal cooperative control $\boldsymbol{U}^*$ in (49), which can be expanded as follows.

$$\begin{aligned}
\bar{\boldsymbol{K}}\boldsymbol{T}^{-1}\boldsymbol{X} &= \left(\begin{array}{l} -\dfrac{1}{w_c^2}\boldsymbol{I}_n \left[\boldsymbol{0}_{n_g\times n_g} \;\; \boldsymbol{0}_{n_g\times n_g} \;\; \cdots \;\; \boldsymbol{0}_{n_g\times n_g} \;\; \boldsymbol{I}_{n_g}\right] \begin{bmatrix} \boldsymbol{P}_{11} & \boldsymbol{P}_{12} & \cdots & \boldsymbol{P}_{1n} \\ \boldsymbol{P}_{21} & \boldsymbol{P}_{22} & \cdots & \boldsymbol{P}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{P}_{1n} & \boldsymbol{P}_{2n} & \cdots & \boldsymbol{P}_{nn} \end{bmatrix} \\ -\left[-a_0\boldsymbol{I}_{n_g} \;\; -a_1\boldsymbol{I}_{n_g} \;\; \cdots \;\; -a_{n-1}\boldsymbol{I}_{n_g}\right] \end{array} \right) \begin{bmatrix} \bar{t}_{11}\boldsymbol{I}_{n_g} & \bar{t}_{12}\boldsymbol{I}_{n_g} & \cdots & \bar{t}_{1n}\boldsymbol{I}_{n_g} \\ \bar{t}_{21}\boldsymbol{I}_{n_g} & \bar{t}_{22}\boldsymbol{I}_{n_g} & \cdots & \bar{t}_{2n}\boldsymbol{I}_{n_g} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{t}_{n1}\boldsymbol{I}_{n_g} & \bar{t}_{n2}\boldsymbol{I}_{n_g} & \cdots & \bar{t}_{nn}\boldsymbol{I}_{n_g} \end{bmatrix} \boldsymbol{X} \\
&= \left(-\frac{1}{w_c^2}\left[\boldsymbol{P}_{1n} \;\; \boldsymbol{P}_{2n} \;\; \cdots \;\; \boldsymbol{P}_{nn}\right] - \left[-a_0\boldsymbol{I}_{n_g} \;\; -a_1\boldsymbol{I}_{n_g} \;\; \cdots \;\; -a_{n-1}\boldsymbol{I}_{n_g}\right] \right) \begin{bmatrix} \bar{t}_{11}\boldsymbol{I}_{n_g} & \bar{t}_{12}\boldsymbol{I}_{n_g} & \cdots & \bar{t}_{1n}\boldsymbol{I}_{n_g} \\ \bar{t}_{21}\boldsymbol{I}_{n_g} & \bar{t}_{22}\boldsymbol{I}_{n_g} & \cdots & \bar{t}_{2n}\boldsymbol{I}_{n_g} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{t}_{n1}\boldsymbol{I}_{n_g} & \bar{t}_{n2}\boldsymbol{I}_{n_g} & \cdots & \bar{t}_{nn}\boldsymbol{I}_{n_g} \end{bmatrix} \boldsymbol{X} \\
&= \begin{bmatrix} -\bar{t}_{11}\left(\dfrac{1}{w_c^2}\boldsymbol{P}_{1n} - a_0\boldsymbol{I}_{n_g}\right) - \bar{t}_{21}\left(\dfrac{1}{w_c^2}\boldsymbol{P}_{2n} - a_1\boldsymbol{I}_{n_g}\right) - \cdots - \bar{t}_{n1}\left(\dfrac{1}{w_c^2}\boldsymbol{P}_{nn} - a_{n-1}\boldsymbol{I}_{n_g}\right), \\ -\bar{t}_{12}\left(\dfrac{1}{w_c^2}\boldsymbol{P}_{1n} - a_0\boldsymbol{I}_{n_g}\right) - \bar{t}_{22}\left(\dfrac{1}{w_c^2}\boldsymbol{P}_{2n} - a_1\boldsymbol{I}_{n_g}\right) - \cdots - -\bar{t}_{n2}\left(\dfrac{1}{w_c^2}\boldsymbol{P}_{nn} - a_{n-1}\boldsymbol{I}_{n_g}\right), \\ \cdots - \bar{t}_{1n}\left(\dfrac{1}{w_c^2}\boldsymbol{P}_{1n} - a_0\boldsymbol{I}_{n_g}\right) - \bar{t}_{2n}\left(\dfrac{1}{w_c^2}\boldsymbol{P}_{2n} - a_1\boldsymbol{I}_{n_g}\right) - \cdots - -\bar{t}_{nn}\left(\dfrac{1}{w_c^2}\boldsymbol{P}_{nn} - a_{n-1}\boldsymbol{I}_{n_g}\right) \end{bmatrix} \boldsymbol{X}
\end{aligned} \tag{79}$$

where $\bar{t}_{ij}$ are the entries of $\boldsymbol{T}^{-1}$. Note that $\bar{\boldsymbol{B}}$ is in the form of $\left[0 \;\; 0 \;\; \cdots \;\; 0 \;\; 1\right]\otimes\boldsymbol{I}_{n_g}$ and thus the product of $\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}$ in $\bar{\boldsymbol{K}}$ only keeps the last column of $\boldsymbol{P}$, i.e. $\boldsymbol{P}_{in}$, which is calculated by (64). Since $\boldsymbol{P}_{in}$'s are always linear functions of the Laplacian matrix $\boldsymbol{L}$ as shown in Eq. (64), when they are multiplied by the state vector $\boldsymbol{X}$ in (79), the feedback information exchange to implement the optimal control occurs only between the agent and its neighbors with whom it has the communication links defined by the Laplacian matrix $\boldsymbol{L}$. The other terms consisting of a_i's (characteristic equation coefficients of $\boldsymbol{A}_g$) and identity matrices in (79) relate to the agent's own state. The second term of $\boldsymbol{U}^*$ in (49) only relates to the state of the agent that has access to the reference. Therefore, from the above discussions, the whole optimal cooperative control law is distributed since each agent's control law only needs local information from its own and its communicating neighbors.

Remark 2.2: According to (79), only the last column of $\boldsymbol{P}$ calculated by (64) is required in order to calculate the optimal control law.

2.4.3 Defining the proper desired trajectory

When the system is controllable, it is possible to reach any state in finite time but we still need to define a proper desired trajectory such that the agents are able to follow after reaching consensus. Since $\boldsymbol{X}_D$ (the desired trajectory) should satisfy the system dynamics, we can write:

$$\dot{\boldsymbol{X}}_D = \boldsymbol{A}_g \boldsymbol{X}_D + \boldsymbol{B}_g \boldsymbol{U}_D \Rightarrow \boldsymbol{B}_g \boldsymbol{U}_D = \dot{\boldsymbol{X}}_D - \boldsymbol{A}_g \boldsymbol{X}_D \tag{80}$$

Define two matrices $\boldsymbol{O}$ and $\boldsymbol{N}$ as follows

$$\boldsymbol{O} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}_{(n-1)\times n} \tag{81}$$

$$\boldsymbol{N} = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}_{1\times n} \tag{82}$$

Now by using (80), (81) and (82) the approach below is followed in order to obtain the system of ordinary differential equations for finding the desired trajectory:

$$\begin{aligned}
&-\boldsymbol{B}_g\boldsymbol{U}_D = -\boldsymbol{B}_g\boldsymbol{U}_D \\
&\Rightarrow -\boldsymbol{O}\boldsymbol{B}_g\boldsymbol{U}_D = -\boldsymbol{O}\boldsymbol{B}_g\boldsymbol{U}_D \\
&\Rightarrow -\boldsymbol{N}\boldsymbol{B}_g\boldsymbol{O}\boldsymbol{B}_g\boldsymbol{U}_D = -\boldsymbol{O}\boldsymbol{B}_g(\boldsymbol{N}\boldsymbol{B}_g)\boldsymbol{U}_D \\
&\Rightarrow \boldsymbol{N}\boldsymbol{B}_g\boldsymbol{O}(\boldsymbol{A}_g\boldsymbol{X}_D - \dot{\boldsymbol{X}}_D) = \boldsymbol{O}\boldsymbol{B}_g\boldsymbol{N}(\boldsymbol{A}_g\boldsymbol{X}_D - \dot{\boldsymbol{X}}_D) \\
&\Rightarrow \boldsymbol{N}\boldsymbol{B}_g\boldsymbol{O}\boldsymbol{A}_g\boldsymbol{X}_D - \boldsymbol{N}\boldsymbol{B}_g\boldsymbol{O}\dot{\boldsymbol{X}}_D = \boldsymbol{O}\boldsymbol{B}_g\boldsymbol{N}\boldsymbol{A}_g\boldsymbol{X}_D - \boldsymbol{O}\boldsymbol{B}_g\boldsymbol{N}\dot{\boldsymbol{X}}_D \\
&\Rightarrow (\boldsymbol{N}\boldsymbol{B}_g\boldsymbol{O}\boldsymbol{A}_g - \boldsymbol{O}\boldsymbol{B}_g\boldsymbol{N}\boldsymbol{A}_g)\boldsymbol{X}_D = (\boldsymbol{N}\boldsymbol{B}_g\boldsymbol{O} - \boldsymbol{O}\boldsymbol{B}_g\boldsymbol{N})\dot{\boldsymbol{X}}_D
\end{aligned} \tag{83}$$

Note that $\boldsymbol{N}\boldsymbol{B}_g$ is a scalar (the first entry of the column vector $\boldsymbol{B}_g$) and therefore it can be put anywhere in the equation.

Expanding the two sides of the equation (83) leads to:

$$\begin{bmatrix} -b_2 & b_1 & 0 & \cdots & 0 \\ -b_3 & 0 & b_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -b_n & 0 & 0 & \cdots & b_1 \end{bmatrix} \dot{\boldsymbol{X}}_D = \begin{bmatrix} b_1a_{21} - b_2a_{11} & b_1a_{22} - b_2a_{12} & \cdots & b_1a_{2n} - b_2a_{1n} \\ b_1a_{31} - b_3a_{11} & b_1a_{32} - b_3a_{12} & \cdots & b_1a_{3n} - b_3a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ b_1a_{n1} - b_na_{11} & b_1a_{n1} - b_na_{12} & \cdots & b_1a_{nn} - b_na_{1n} \end{bmatrix} \boldsymbol{X}_D \tag{84}$$

where b_j's and a_{ij}'s are the entries of the $\boldsymbol{B}_g$ and $\boldsymbol{A}_g$ respectively. There are $(n-1)$ differential equations with n unknowns, so the system (84) has infinitely many solutions. Once one of the states is set, the rest can be calculated by solving $(n-1)$ equations to obtain a unique solution as the desired trajectory.

2.4.4 The algorithm for finding the optimal cooperative control

The approach to find the optimal cooperative control $\boldsymbol{U}^*$ for a system of n_g agents with the general LTI dynamics $\dot{\boldsymbol{X}} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\boldsymbol{U}^*$ to achieve consensus and follow a desired trajectory can be summarized in the following algorithm:

1) Use (10) to compute the controllability matrix $\boldsymbol{M}$.

2) Check the rank of $\boldsymbol{M}$. If $\boldsymbol{M}$ is not full rank, the system is not controllable and the approach cannot be applied.

3) Find the characteristic equation of the state matrix $\boldsymbol{A}_g$.

4) Construct the flipped Toeplitz matrix $\boldsymbol{W}$ by (11).

5) Compute the transformation matrix $\boldsymbol{T} = \boldsymbol{MW} \otimes \boldsymbol{I}_{n_g}$ and its inverse $\boldsymbol{T}^{-1}$.

6) Construct the Laplacian matrix $\boldsymbol{L}$ according to the network topology and $\boldsymbol{R}_1$ as defined in (61).

7) Using $\boldsymbol{R}_1$, solve the ARE (59) to find the analytical solution of $\boldsymbol{P}$ given by (64). Note that only the last column of $\boldsymbol{P}$ is used in the optimal control as discussed in Remark 4.2.

8) Substitute the calculated $-2\boldsymbol{P}_{(i-1)i}$ into the diagonal entries of $\boldsymbol{R}_1$ (constructed in step 6) and choose proper weights $\{w_1, w_2, \cdots, w_n\}$ such that $\boldsymbol{R}_1$ becomes P.S.D.

9) Choose a proper weight w_c for the control and make the P.D. matrix $\boldsymbol{R}_2 = w_c^2 \otimes \boldsymbol{I}_{n_g}$.

10) Pick an appropriate desired trajectory $\boldsymbol{X}_D$ by solving the system of differential equations (84).

11) For any agent that has access to the reference vector, the tracking term of the controller is:

$$\begin{bmatrix} 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{n_g \times n} (\boldsymbol{G} + \boldsymbol{G}^T)\boldsymbol{T}^{-1}(\boldsymbol{X}_i - \boldsymbol{X}_D) \tag{85}$$

For the rest of the agents, this term will be zero.

12) The optimal control is designed as:

$$\boldsymbol{U}^* = (-\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P} - \begin{bmatrix}-a_0 & -a_1 & \cdots & -a_{n-1}\end{bmatrix} \otimes \boldsymbol{I}_{n_g})(\boldsymbol{T}^{-1} \otimes \boldsymbol{I}_{n_g})\boldsymbol{X} - \frac{1}{2w_c^2}\frac{\partial}{\partial \boldsymbol{X}_{(n)}} g(\boldsymbol{X}) \qquad (86)$$

The second term is what has been calculated in step 11). Choose w_{d_i} ' s such that the system has the best response.

2.5 Illustrative examples

In this section, the performance of the optimal cooperative control design is demonstrated through two examples. In the first example a very simple system is considered in order to illustrate how the algorithm works mathematically. In example 2, the algorithm is applied for solving a practical problem of synchronizing the attitude of a group of satellites.

Example 2.1

For the first example, we intentionally select a simple two-dimensional problem with only two agents in order to more clearly show the trajectories and consensus results in a phase plane. We assume that the system consists of two identical agents with the same dynamics of

$$\dot{\boldsymbol{X}}_i = \boldsymbol{A}_g\boldsymbol{X}_i + \boldsymbol{B}_g\boldsymbol{U}_i$$

where $$\boldsymbol{A}_g = \begin{bmatrix} 1 & 2 \\ -2 & 3 \end{bmatrix}, \quad \boldsymbol{B}_g = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \qquad (87)$$

The consensus problem is to make the system reach a specified point at the same time and stop there. They can communicate via the undirected link shown in Figure 3.

Figure 3. The two agents in example 2.1

The corresponding Laplacian matrix is:

$$\boldsymbol{L} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{88}$$

In order to obtain a point where the system can reach, the system (84) is solved. It is assumed that only the first agent has access to the desired trajectory, which is denoted by $\boldsymbol{X}_D = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$. For the system described in this example, the system is reduced to a simple ODE as follows:

$$\begin{aligned} &\begin{bmatrix} -b_2 & b_1 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} b_1 a_{21} - b_2 a_{11} & b_1 a_{22} - b_2 a_{12} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &\Rightarrow \begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} (2-1) & (-3-2) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &\Rightarrow -\dot{x}_1 - \dot{x}_2 = x_1 - 5x_2 \end{aligned} \tag{89}$$

Since for this example, the two agents are required to reach a fixed point and stop there, both derivatives should be zero when consensus is reached. In other words, $\dot{x}_1 = \dot{x}_2 = 0$. Therefore, the desired point should lie on the line $5x_2 - x_1 = 0$.

The optimal control law can be designed analytically by following the algorithm described in Section 2.4. The results are shown in Figures 4(a)-(d). The final point is set to be [5, 1] and the weights have been selected as: $w_1 = 1$, $w_2 = 2$, $w_c = 0.5$, $w_{d_1} = 1$, $w_{d_2} = 0.5$.

The initial conditions are $x_1(0) = [-1,1]$ and $x_2(0) = [0,1]$ for the first and second agents, respectively.

Figures 4. (a), (b) show the phase plane trajectories for the two agents. It can be seen that there will always be a stable focus lying on the line of $x_2 = 0.2x_1$ ([5, 1] for this example).

Figures 4. (c), (d) show the results for the first and second states of the two agents respectively. It can be seen that the first states finally reach the value of 5 and the second states reach 1 in finite time.

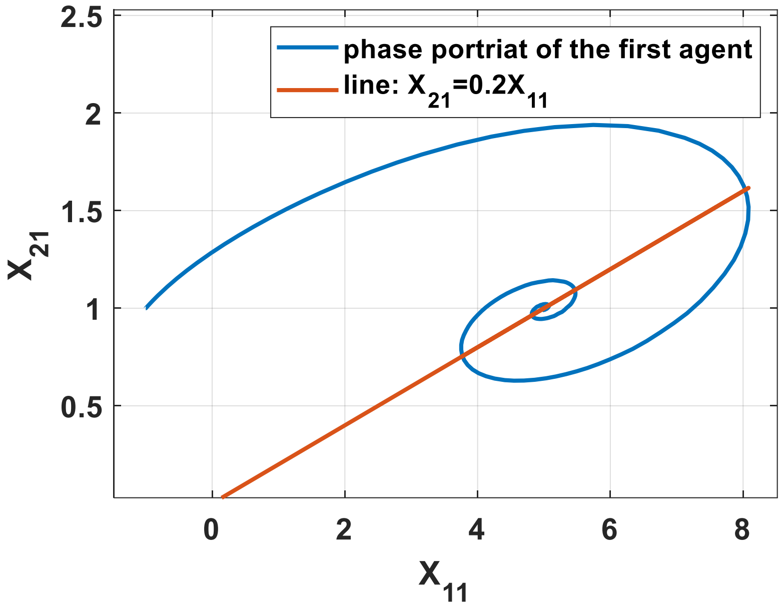

Figure 4 (a). Phase plane for the first agent

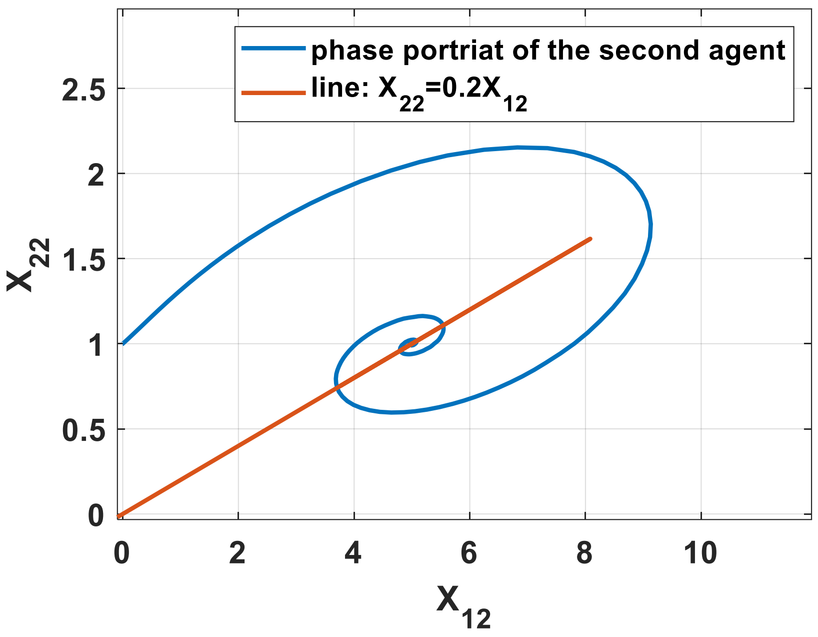

Figure 4 (b). Phase plane for the second agent

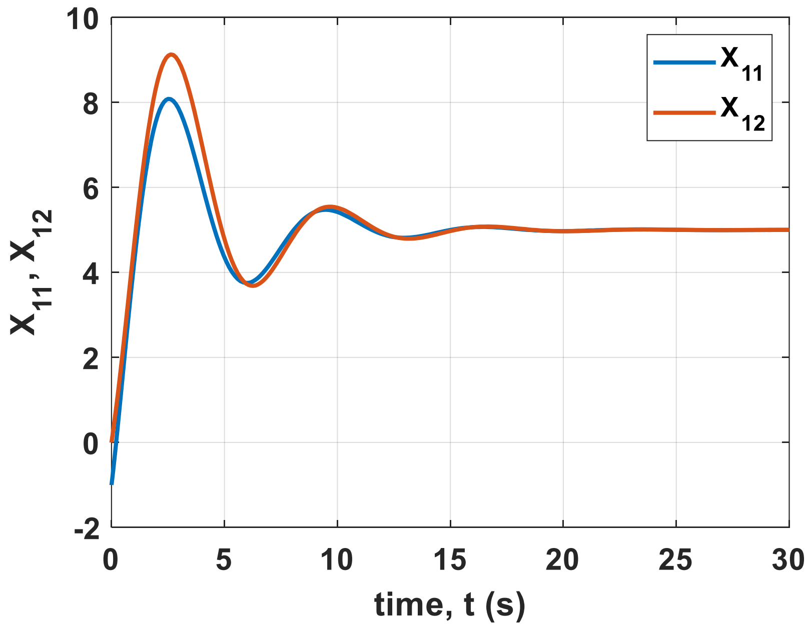

Figure 4 (c). The first states of the agents

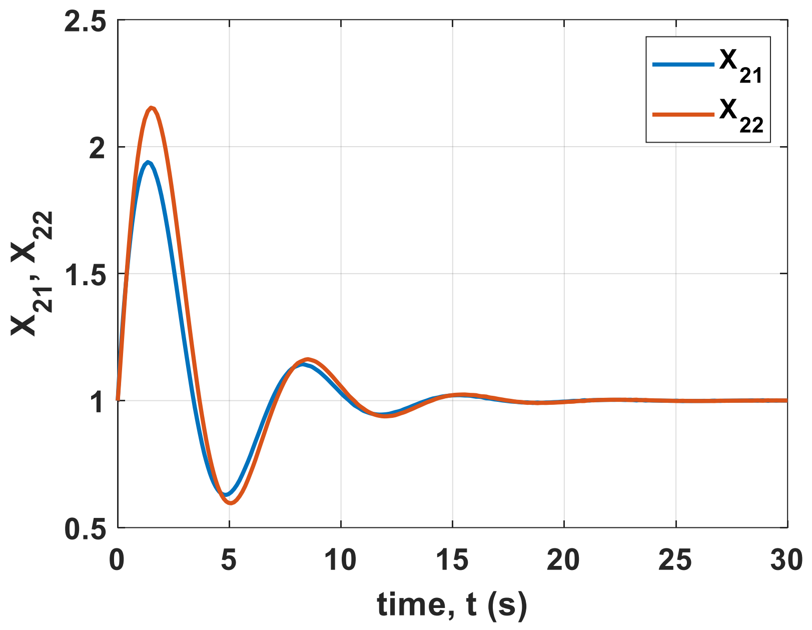

Figure 4 (d). The second states of the agents

Example 2.2 Satellite attitude control:

In satellite attitude control, appropriate orientation is an important problem. A sketch of the satellite control system and the model are shown in Figure 5 (taken from [85, 86]). The system can be considered as two separate masses (a large mass called "the body" and one "attached mass") which are connected to each other. Therefore it can be modeled by a simple mass-spring-damper system in which "k" is the spring constant and "d" is the viscous damping constant.

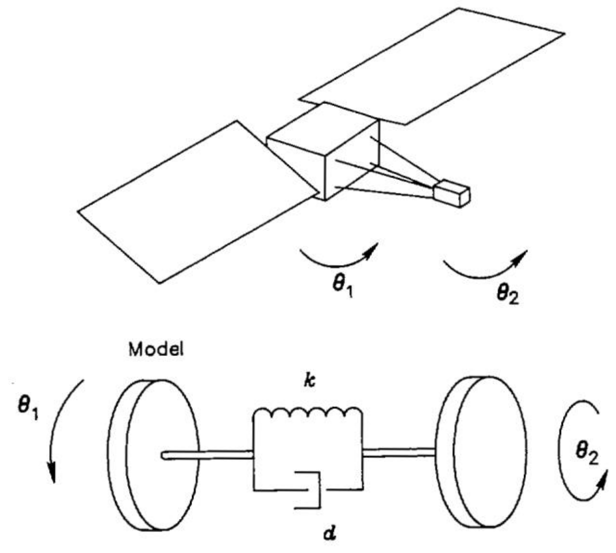

Figure 5. The satellite control system in example 2.2

The equations of motion are obtained as:

$$\begin{cases} J_1\ddot{\theta}_1 + d\left(\dot{\theta}_1 - \dot{\theta}_2\right) + k\left(\theta_1 - \theta_2\right) = T_C \\ J_2\ddot{\theta}_2 + d\left(\dot{\theta}_2 - \dot{\theta}_1\right) + k\left(\theta_2 - \theta_1\right) = 0 \end{cases} \tag{90}$$

Where T_C is the control torque, J_1 and J_2 are moments of inertia. The system state matrix and input state matrix are constructed based on these equations of motion as below:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{J_2} & -\frac{d}{J_2} & \frac{k}{J_2} & \frac{d}{J_2} \\ 0 & 0 & 0 & 1 \\ \frac{k}{J_1} & \frac{d}{J_1} & -\frac{k}{J_1} & -\frac{d}{J_1} \end{bmatrix} \quad , \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J_1} \end{bmatrix} \tag{91}$$

We choose $J_1 = J_2 = 1$, $k = 0.09$, $d = 0.022$ and the state vector is $\boldsymbol{X}_i = (\theta_2, \dot{\theta}_2, \theta_1, \dot{\theta}_1)^T$. Satellite 1 has access to the reference trajectory. Using the system of differential equations (115) lead to the obvious result that $\dot{x}_1 = x_2$ or $\dot{\theta}_2 = \dot{\theta}_2$ .Therefore the desired trajectory could be defined easily. We assume that the angular velocities are constant and equal to 0.35. Therefore, the angles are supposed to increase along the ramp $\theta = 0.35t$.

The state and control weightings parameters have been chosen as: $w_1 = 1$, $w_2 = 3$, $w_3 = 3$, $w_4 = 1$ and $w_c = \frac{1}{5}$. The tunable weights for the tracking penalty function are $w_{d_1} = 10$, $w_{d_2} = 500$, $w_{d_3} = 500$, $w_{d_4} = 5$ The initial condition of each state of θ_2 , $\dot{\theta}_2$, θ_1 , $\dot{\theta}_1$ are set to be $\begin{bmatrix} 6 & 5 & 8 & 2 & 7 \end{bmatrix}^T$ rad , $\begin{bmatrix} 0 & 1 & 0.5 & -1 & -0.5 \end{bmatrix}^T$ rad/s , $\begin{bmatrix} 8 & 7 & 7 & 6 & 7.5 \end{bmatrix}^T$ rad and $\begin{bmatrix} 0.6 & -1.5 & 1 & 0.8 & 1.5 \end{bmatrix}^T$ rad/s respectively.

The topology is assumed as shown in Figure 4. The graph is obviously connected and undirected.

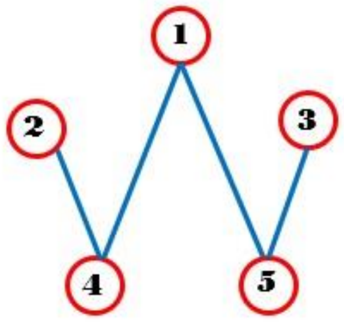

Figure 6. The topology of the satellites

The corresponding Laplacian matrix is:

$$\boldsymbol{L} = \begin{bmatrix} 2 & 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ -1 & -1 & 0 & 2 & 0 \\ -1 & 0 & -1 & 0 & 2 \end{bmatrix} \tag{92}$$

Figures 7(a)-(e) show the results. It can be seen that the rotational angles converge and increase along the ramp of $\theta = 0.35t$ and the angular rates converge to the constant value of 0.35 rad/s. It is worth noting that the responses of the satellite 1 are very smooth and do not have oscillations because it has access to the reference trajectory.

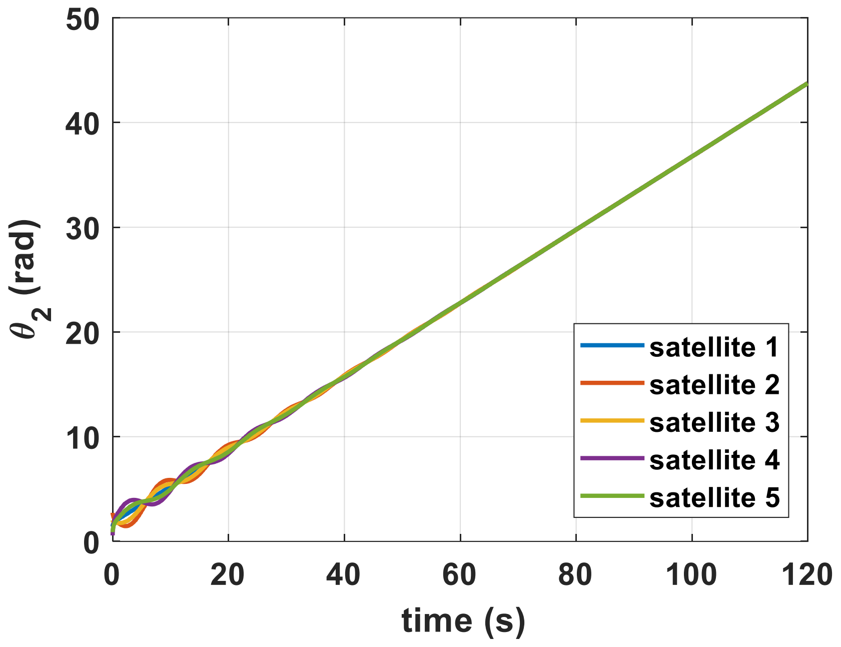

Figure 7 (a). The rotational angles of the attached masses (the first states)

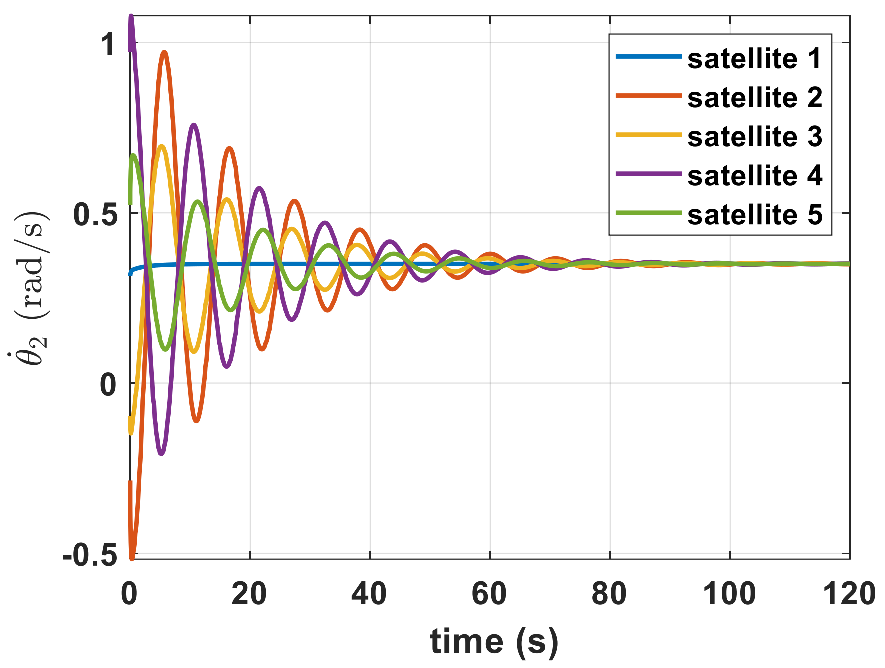

Figure 7(b). The angular rates of the attached masses (the second states)

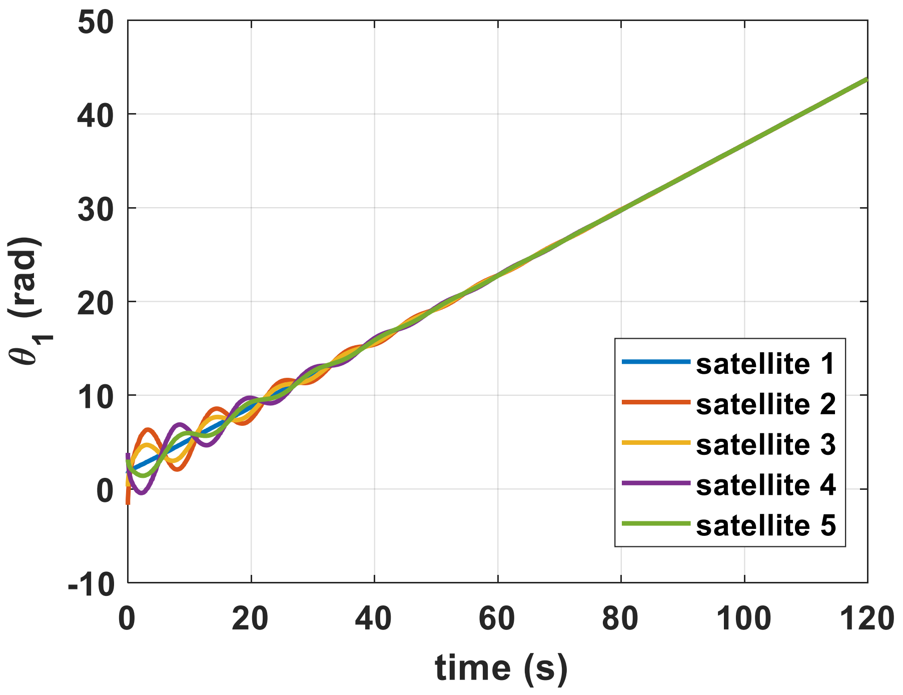

Figure 7(c). The rotational angles of the bodies (the third states)

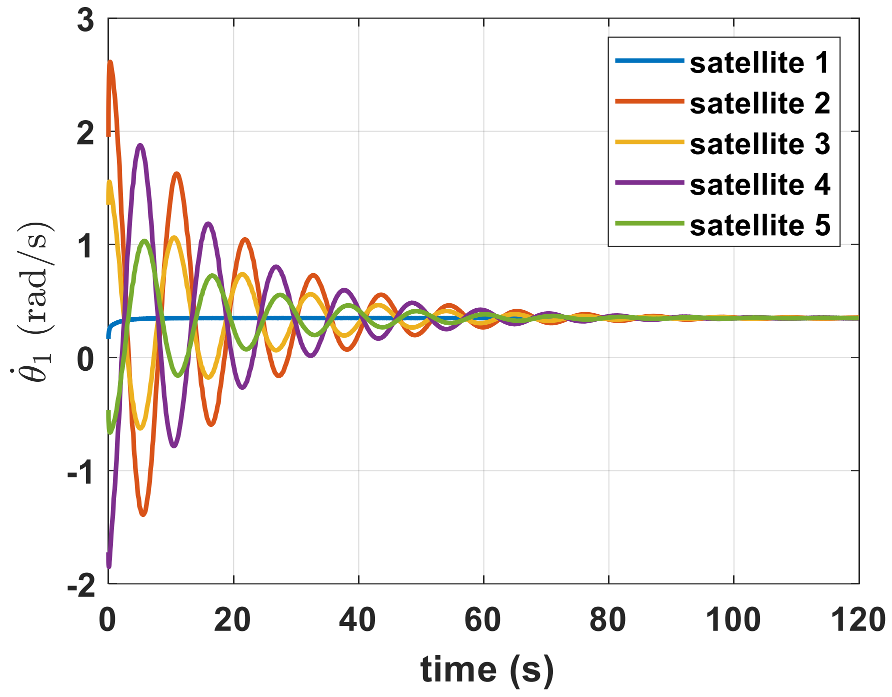

Figure 7(d). The angular rates of the bodies (the fourth states)

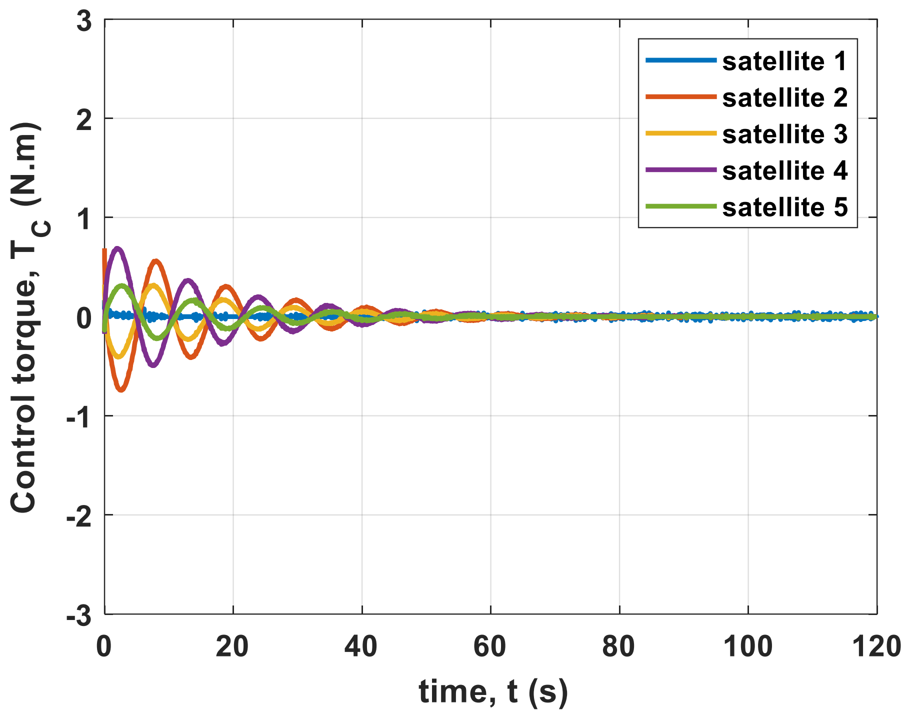

Figure 7(e). The control torque

3. Multi-Input Systems

3.1 Overview

In this chapter the algorithm described in chapter 2 will be generalized to embrace general Linear Time-Invariant multi-input systems. When the input matrix is no longer a column vector, the most important challenge would be the decision on choosing the proper transformation matrix. We will develop three new theorems which enable us to define the most appropriate transformation matrix and the important properties of it are derived. The Algebraic Riccati Equation (ARE) can no longer be solved analytically but we will see that the discussion on the distributed cooperative control law (developed in section 2.4.2) is still valid. It will be seen that the proper design strategy of the desired trajectory must also be redefined according to the new input matrix.

Problem statement and Formulation is described in section 3.2. Main results are provided in 3.3. In this section optimal control solution is derived and some issues regarding the distributed cooperative control law are discussed. Also, the approach for designing the proper desired trajectory is explained in this section. Finally, the application of the defined algorithm is illustrated through two examples in section 3.4.

3.2 Problem statement and formulation

When the system is a multi-input one, $\boldsymbol{B}$ is no longer a column vector (i.e. $m > 1$ and $\boldsymbol{B} = \boldsymbol{B}_g \otimes \boldsymbol{I}_{\boldsymbol{n}_g} = \left(b_{ij}\right)_{n\times m} \otimes \boldsymbol{I}_{\boldsymbol{n}_g}$) . Regarding to (9), any columns of the input matrix could be used to obtain a new transformation matrix. The converted state matrix, $\bar{\boldsymbol{A}}$ always remains unchanged but $\bar{\boldsymbol{B}}$ will be different by applying different transformation matrices. The challenge of constructing the best transformation matrix can be elucidated by using three following theorems:

Theorem 3.1 Let $\boldsymbol{B}_j$ s $j=1,2,...,m$ be the "m" columns of the input matrix $\boldsymbol{B}_g$ for each agent and $\boldsymbol{T}_j$ s be the associated transformation matrices (i.e. $\boldsymbol{T}_j = \boldsymbol{M}_j \boldsymbol{W} = \begin{bmatrix} \boldsymbol{B}_j & \boldsymbol{A}_g \boldsymbol{B}_j & \boldsymbol{A}_g{}^2 \boldsymbol{B}_j & \cdots & \boldsymbol{A}_g{}^{n-1} \boldsymbol{B}_j \end{bmatrix} \boldsymbol{W}, \quad 1 \le j \le m$) and $\boldsymbol{W}$ is constructed as (11). The transformation matrix $\boldsymbol{T}_t = \left(\sum_{j=1}^{m} \boldsymbol{T}_j \right) \otimes \boldsymbol{I}_{n_g}$ converts the system matrix, $\boldsymbol{A}$, into canonical form (15) (i.e. $\boldsymbol{T}_t^{-1} \boldsymbol{A} \boldsymbol{T}_t = \bar{\boldsymbol{A}}$).

Proof: For any $\boldsymbol{T}_j$, we have:

$$\boldsymbol{T}_j^{-1} \boldsymbol{A} \boldsymbol{T}_j = \bar{\boldsymbol{A}}$$

which means:

$$\boldsymbol{A} = \boldsymbol{T}_j \bar{\boldsymbol{A}} \boldsymbol{T}_j^{-1} = \boldsymbol{T}_1 \bar{\boldsymbol{A}} \boldsymbol{T}_1^{-1} = \boldsymbol{T}_2 \bar{\boldsymbol{A}} \boldsymbol{T}_2^{-1} = ... = \boldsymbol{T}_m \bar{\boldsymbol{A}} \boldsymbol{T}_m^{-1}$$

Applying the suggested transformation matrix, $\boldsymbol{T}_t = \sum_{j=1}^{m} \boldsymbol{T}_j$ we have:

$$\begin{aligned}
\left(\sum \boldsymbol{T}_j\right)^{-1} \boldsymbol{A} \left(\sum \boldsymbol{T}_j\right) &= \left(\boldsymbol{T}_1 + \boldsymbol{T}_2 + ... + \boldsymbol{T}_m\right)^{-1} \boldsymbol{A} \left(\boldsymbol{T}_1 + \boldsymbol{T}_2 + ... + \boldsymbol{T}_m\right) \\
&= \left(\boldsymbol{T}_1 + \boldsymbol{T}_2 + ... + \boldsymbol{T}_m\right)^{-1} \boldsymbol{T}_1 \bar{\boldsymbol{A}} \boldsymbol{T}_1^{-1} \left(\boldsymbol{T}_1 + \boldsymbol{T}_2 + ... + \boldsymbol{T}_m\right) \\
&= \left(\boldsymbol{T}_1 + \boldsymbol{T}_2 + ... + \boldsymbol{T}_m\right)^{-1} \left(\boldsymbol{T}_1 \bar{\boldsymbol{A}} + \boldsymbol{T}_1 \bar{\boldsymbol{A}} \boldsymbol{T}_1^{-1} \boldsymbol{T}_2 + ... + \boldsymbol{T}_1 \bar{\boldsymbol{A}} \boldsymbol{T}_1^{-1} \boldsymbol{T}_m\right) \\
&= \left(\boldsymbol{T}_1 + \boldsymbol{T}_2 + ... + \boldsymbol{T}_m\right)^{-1} \left(\boldsymbol{T}_1 \bar{\boldsymbol{A}} + \boldsymbol{T}_2 \bar{\boldsymbol{A}} \boldsymbol{T}_2^{-1} \boldsymbol{T}_2 + ... + \boldsymbol{T}_m \bar{\boldsymbol{A}} \boldsymbol{T}_m^{-1} \boldsymbol{T}_m\right) \\
&= \left(\boldsymbol{T}_1 + \boldsymbol{T}_2 + ... + \boldsymbol{T}_m\right)^{-1} \left(\boldsymbol{T}_1 \bar{\boldsymbol{A}} + \boldsymbol{T}_2 \bar{\boldsymbol{A}} + ... + \boldsymbol{T}_m \bar{\boldsymbol{A}}\right) \\
&= \left(\boldsymbol{T}_1 + \boldsymbol{T}_2 + ... + \boldsymbol{T}_m\right)^{-1} \left(\boldsymbol{T}_1 + \boldsymbol{T}_2 + ... + \boldsymbol{T}_m\right) \bar{\boldsymbol{A}} = \bar{\boldsymbol{A}}
\end{aligned}$$

Theorem 3.1 shows that the sum of transformation matrices, made by the columns of $\boldsymbol{B}_g$, is also a transformation matrix.

The following two theorems show useful properties of the converted input matrix of each agents $\bar{\boldsymbol{B}}$ when the transformation matrix $\boldsymbol{T} = \sum_{j=1}^{m} \boldsymbol{T}_j$ is applied.

Theorem 3.2 Let $\boldsymbol{M}_j$ be the controllability matrix associated with the "j"th column of $\boldsymbol{B}_g$ (i.e. $\boldsymbol{M}_j = \begin{bmatrix} \boldsymbol{B}_j & \boldsymbol{A}_g\boldsymbol{B}_j & \boldsymbol{A}_g{}^2\boldsymbol{B}_j & \cdots & \boldsymbol{A}_g{}^{n-1}\boldsymbol{B}_j \end{bmatrix}$ where $\boldsymbol{B}_j$ is the "j"th column of $\boldsymbol{B}_g$). Using $\boldsymbol{M}_t = \sum_{j=1}^{m} \boldsymbol{M}_j$ as the transformation matrix, the converted input matrix "$\bar{\boldsymbol{B}}_g$" has the following property:

$$\sum_{j=1}^{m} \boldsymbol{B}_j = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}_{1\times n}^{T}$$

In other words, the sum of columns of the converted matrix is always $\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}_{1\times n}^{T}$.

***Proof*:**

$$\bar{\boldsymbol{B}}_g = \boldsymbol{M}_t^{-1}\boldsymbol{B}_g \Rightarrow \boldsymbol{B}_g = \boldsymbol{M}_t\bar{\boldsymbol{B}}_g = \left(\sum_{j=1}^{m} \boldsymbol{M}_j\right)\bar{\boldsymbol{B}}_g$$

$$\Rightarrow \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 & \cdots & \boldsymbol{B}_m \end{bmatrix} = \left(\sum_{j=1}^{m} \boldsymbol{M}_j\right)\begin{bmatrix} \bar{\boldsymbol{B}}_1 & \bar{\boldsymbol{B}}_2 & \cdots & \bar{\boldsymbol{B}}_m \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{m} \boldsymbol{M}_j\bar{\boldsymbol{B}}_1 & \sum_{j=1}^{m} \boldsymbol{M}_j\bar{\boldsymbol{B}}_2 & \cdots & \sum_{j=1}^{m} \boldsymbol{M}_j\bar{\boldsymbol{B}}_m \end{bmatrix}$$

Which means:

$$\boldsymbol{B}_1 + \boldsymbol{B}_2 + \cdots + \boldsymbol{B}_m = \sum_{j=1}^{m} \boldsymbol{M}_j\left(\bar{\boldsymbol{B}}_1 + \bar{\boldsymbol{B}}_2 + \cdots + \bar{\boldsymbol{B}}_m\right) \Rightarrow \sum_{j=1}^{m} \boldsymbol{B}_j = \sum_{j=1}^{m} \boldsymbol{M}_j\left(\sum_{j=1}^{m} \bar{\boldsymbol{B}}_j\right)$$

but $\sum_{j=1}^{m} \boldsymbol{M}_j = \begin{bmatrix} \sum_{j=1}^{m} \boldsymbol{B}_j & \boldsymbol{A}_g\sum_{j=1}^{m} \boldsymbol{B}_j & \boldsymbol{A}_g^2\sum_{j=1}^{m} \boldsymbol{B}_j & \cdots & \boldsymbol{A}_g^{n-1}\sum_{j=1}^{m} \boldsymbol{B}_j \end{bmatrix}$, therefore:

$$\sum_{j=1}^{m}\boldsymbol{B}_j=\left[\sum_{j=1}^{m}\boldsymbol{B}_j \quad \boldsymbol{A}_g\sum_{j=1}^{m}\boldsymbol{B}_j \quad \boldsymbol{A}_g^2\sum_{j=1}^{m}\boldsymbol{B}_j \quad \cdots \quad \boldsymbol{A}_g^{n-1}\sum_{j=1}^{m}\boldsymbol{B}_j\right]\left(\sum_{j=1}^{m}\bar{\boldsymbol{B}}_j\right)$$

and this yields:

$$\sum_{j=1}^{m}\bar{\boldsymbol{B}}_j=\begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix}_{1\times n}^T$$

Theorem 3.3 Let $\boldsymbol{T}_j$ s be the transformation matrices associated with the columns of the input matrix of each agent, $\boldsymbol{B}_j$ s (i.e. $\boldsymbol{T}_j=\boldsymbol{M}_j\boldsymbol{W}$ and $\boldsymbol{M}_j$ is as defined in Theorem 3.2 and $\boldsymbol{W}$ is flipped Toeplitz matrix (11)). Using $\boldsymbol{T}_t=\sum_{j=1}^{m}\boldsymbol{T}_j$ as the transformation matrix, the converted input matrix $\bar{\boldsymbol{B}}_g$ has the following property:

$$\sum_{j=1}^{m}\bar{\boldsymbol{B}}_j=\begin{bmatrix}0 & 0 & \cdots & 1\end{bmatrix}_{1\times n}^T$$

In other words, the sum of columns of the converted input matrix is always

$\begin{bmatrix}0 & 0 & \cdots & 1\end{bmatrix}_{1\times n}^T$.

Proof:

$$\boldsymbol{B}_g=\boldsymbol{T}_t^{-1}\boldsymbol{B}_g\Rightarrow\boldsymbol{B}_g=\boldsymbol{T}_t\bar{\boldsymbol{B}}_g=\left(\sum_{j=1}^{m}\boldsymbol{T}_j\right)\bar{\boldsymbol{B}}_g$$

$$\Rightarrow\begin{bmatrix}\boldsymbol{B}_1 & \boldsymbol{B}_2 & \cdots & \boldsymbol{B}_m\end{bmatrix}=\left(\sum_{j=1}^{m}\boldsymbol{T}_j\right)\begin{bmatrix}\bar{\boldsymbol{B}}_1 & \bar{\boldsymbol{B}}_2 & \cdots & \bar{\boldsymbol{B}}_m\end{bmatrix}=\left[\sum_{j=1}^{m}\boldsymbol{T}_j\bar{\boldsymbol{B}}_1 \quad \sum_{j=1}^{m}\boldsymbol{T}_j\bar{\boldsymbol{B}}_2 \quad \cdots \quad \sum_{j=1}^{m}\boldsymbol{T}_j\bar{\boldsymbol{B}}_m\right]$$

which means:

$$\boldsymbol{B}_1+\boldsymbol{B}_2+\cdots+\boldsymbol{B}_m=\sum_{j=1}^{m}\boldsymbol{T}_j\left(\bar{\boldsymbol{B}}_1+\bar{\boldsymbol{B}}_2+\cdots+\bar{\boldsymbol{B}}_m\right)\Rightarrow\sum_{j=1}^{m}\boldsymbol{B}_j=\sum_{j=1}^{m}\boldsymbol{T}_j\left(\sum_{j=1}^{m}\bar{\boldsymbol{B}}_j\right)$$

but $\sum_{j=1}^{m}\boldsymbol{T}_j=\left[\sum_{j=1}^{m}\boldsymbol{B}_j \quad \boldsymbol{A}_g\sum_{j=1}^{m}\boldsymbol{B}_j \quad \boldsymbol{A}_g^2\sum_{j=1}^{m}\boldsymbol{B}_j \quad \cdots \quad \boldsymbol{A}_g^{n-1}\sum_{j=1}^{m}\boldsymbol{B}_j\right]\boldsymbol{W}$, therefore:

$$\sum_{j=1}^{m}\boldsymbol{B}_j = \begin{bmatrix} \sum_{j=1}^{m}\boldsymbol{B}_j & \boldsymbol{A}_g\sum_{j=1}^{m}\boldsymbol{B}_j & \boldsymbol{A}_g^2\sum_{j=1}^{m}\boldsymbol{B}_j & \cdots & \boldsymbol{A}_2^{n-1}\sum_{j=1}^{m}\boldsymbol{B}_j \end{bmatrix}\boldsymbol{W}\left(\sum_{j=1}^{m}\bar{\boldsymbol{B}}_j\right)$$

According to the theorem 3.2 we have:

$$\boldsymbol{W}\sum_{j=1}^{m}\bar{\boldsymbol{B}}_j = \begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix}_{n\times 1} \Rightarrow \begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & 1 \\ a_2 & a_3 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1} & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}_{n\times n} \sum_{j=1}^{m}\bar{\boldsymbol{B}}_j = \begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix}_{n\times 1}$$

$$\Rightarrow \sum_{j=1}^{m}\bar{\boldsymbol{B}}_j = \begin{bmatrix}0 & 0 & \cdots & 1\end{bmatrix}_{1\times n}^{T}$$

Therefore, the appropriate transformation matrix for a multi-input system is defined as:

$$\boldsymbol{T}_t = \left(\sum_{j=1}^{m}\boldsymbol{T}_j\right)\otimes \boldsymbol{I}_{n_g} \tag{93}$$

Similar to single-input systems, the state and final consensus vectors are converted as:

$$\boldsymbol{X} = \boldsymbol{T}_t\bar{\boldsymbol{X}} \tag{94.a}$$

$$\boldsymbol{X}_{cs} = \boldsymbol{T}_t\bar{\boldsymbol{X}}_{cs} \tag{94.b}$$

By subtracting (94-b) from (94-a) the equation for converting the error state vector is derived:

$$\boldsymbol{X} - \boldsymbol{X}_{cs} = \boldsymbol{T}_t\left(\bar{\boldsymbol{X}} - \bar{\boldsymbol{X}}_{cs}\right) \Rightarrow \hat{\boldsymbol{X}} = \boldsymbol{T}_t\hat{\bar{\boldsymbol{X}}} \tag{95}$$

Using this transformation equation, the system is converted to:

$$\dot{\hat{\bar{\boldsymbol{X}}}} = \bar{\boldsymbol{A}}\hat{\bar{\boldsymbol{X}}} + \bar{\boldsymbol{B}}\boldsymbol{U} \tag{96}$$

where $\bar{\boldsymbol{A}}$ is the converted state matrix in form (15) and $\bar{\boldsymbol{B}}$, the converted input matrix, has the property that the sum of all its columns is $\begin{bmatrix}0 & 0 & \cdots & 1\end{bmatrix}_{1\times n}^{T}\otimes \boldsymbol{1}_{n_g}$ where $\boldsymbol{1}_{n_g}$ is a column vector of n_g ones.

We rewrite the converted system (96) as:

$$\dot{\hat{\boldsymbol{X}}} = \left(\bar{\boldsymbol{A}}_2 + \bar{\boldsymbol{B}}\boldsymbol{K}_1\right)\hat{\boldsymbol{X}} + \bar{\boldsymbol{B}}\boldsymbol{U} \tag{97}$$

where $\bar{\boldsymbol{A}}_2 + \bar{\boldsymbol{B}}\boldsymbol{K}_1 = \bar{\boldsymbol{A}}$. For multi-input systems, $\boldsymbol{K}_1$ is a $m \cdot n_g \times n \cdot n_g$ matrix and is constructed as follows:

$$\boldsymbol{K}_1 = \begin{bmatrix} -a_0 & -a_1 & \cdots & -a_{n-1} \\ -a_0 & -a_1 & \cdots & -a_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ -a_0 & -a_1 & \cdots & -a_{n-1} \end{bmatrix}_{m \times n} \otimes \boldsymbol{I}_{n_g} \tag{98}$$

Now the system (97) can be written as:

$$\dot{\hat{\boldsymbol{X}}} = \bar{\boldsymbol{A}}_2\hat{\boldsymbol{X}} + \bar{\boldsymbol{B}}\boldsymbol{U}_2 \tag{99}$$

where

$$\boldsymbol{U}_2 = \boldsymbol{K}_1\hat{\boldsymbol{X}} + \boldsymbol{U}, \quad \bar{\boldsymbol{A}}_2 = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{n \times n} \otimes \boldsymbol{I}_{n_g} \tag{100}$$

The problems is to find an input control to make (99) asymptotically stable.

Problem formulation is the same as what was described in section 2.3. All the definitions are valid for multi-input systems. The only difference is the size of the matrix $\boldsymbol{R}_2$, which is defined as:

$$\boldsymbol{R}_2 = w_c^2\boldsymbol{I}_{m \cdot n_g} \tag{101}$$

3.3 Main results

3.3.1 Optimal control solution

Lemma 1 is still used for obtaining the main results. All the conditions from (28) to (33) should be satisfied for the multi-input systems too.

Tracking penalty function and its weighting matrix are defined the same as (38)-(40).

Theorem 3.4 For a system of n_g identical agents each with a dynamics of (4), the following control law is an optimal control, which makes the system achieve consensus and follow a proper desired trajectory.

$$\boldsymbol{U} = \left(-\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P} - \boldsymbol{K}_1\right)\boldsymbol{T}_t^{-1}\boldsymbol{X} - \frac{1}{2w_c^2}\bar{\boldsymbol{B}}^T\frac{\partial}{\partial \boldsymbol{X}}g(\boldsymbol{X}) \tag{102}$$

where $\boldsymbol{R}_2$ is the control weighting matrix (102), $\boldsymbol{K}_1$ is defined as (98), $\boldsymbol{T}_t$ is the transformation matrix and is defined as (93) and $\frac{\partial}{\partial \boldsymbol{X}}g(\boldsymbol{X})$ is the derivative of the tracking penalty function with respect to the state vector $\boldsymbol{X}$. The cost function $h(\hat{\hat{\boldsymbol{X}}})$ is calculated as:

$$h(\hat{\hat{\boldsymbol{X}}}) = \frac{1}{4}g'^T(\hat{\hat{\boldsymbol{X}}})\bar{\boldsymbol{B}}\bar{\boldsymbol{B}}^T g'(\hat{\hat{\boldsymbol{X}}}) - g'^T(\bar{\boldsymbol{A}}_2 - \boldsymbol{SP})\hat{\hat{\boldsymbol{X}}} \tag{103}$$

where $\boldsymbol{S} = \bar{\boldsymbol{B}}\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^T$ and $g'(\hat{\hat{\boldsymbol{X}}}) = \frac{\partial}{\partial \hat{\hat{\boldsymbol{X}}}}\boldsymbol{g}(\hat{\hat{\boldsymbol{X}}})$ is the derivative of the tracking penalty function with respect to the state vector $\hat{\hat{\boldsymbol{X}}}$.

Proof: For the converted system (99), $\boldsymbol{T}(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U})$ and $\boldsymbol{f}(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U})$ can be defined according to the lemma 1 as:

$$T(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U}) = \hat{\hat{\boldsymbol{X}}}^T\boldsymbol{R}_1\hat{\hat{\boldsymbol{X}}} + \boldsymbol{U}_2^T\boldsymbol{R}_2\boldsymbol{U}_2 + h(\hat{\hat{\boldsymbol{X}}}) \tag{104}$$

$$\boldsymbol{f}(\hat{\hat{\boldsymbol{X}}},\boldsymbol{U}) = \bar{\boldsymbol{A}}_2\hat{\hat{\boldsymbol{X}}} + \bar{\boldsymbol{B}}\boldsymbol{U}_2 \tag{105}$$

Similar to what we did for single-input systems, a proper Lyapanov function is defined as:

$$V(\hat{\hat{\boldsymbol{X}}}) = \hat{\hat{\boldsymbol{X}}}^{\boldsymbol{T}} \boldsymbol{P} \hat{\hat{\boldsymbol{X}}} + g(\hat{\hat{\boldsymbol{X}}}) \tag{106}$$

Again, (106) is a valid Lyapanov function since it is continuously differentiable with respect to $\hat{\hat{\boldsymbol{X}}}$ (regarding to the definition of $\boldsymbol{G}$). On the other hand, $V(\hat{\hat{\boldsymbol{X}}}) > 0$, $\hat{\hat{\boldsymbol{X}}} \in \boldsymbol{D}$ and $\hat{\hat{\boldsymbol{X}}} \neq \boldsymbol{0}$ as long as both $\boldsymbol{P}$ and $\boldsymbol{G}$ are Positive Semi Definite (P.S.D).

The proof is basically the same as it is for theorem 4.1. The control input for the system (96) is written as:

$$\begin{aligned}
&\boldsymbol{U}_2 = -\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{\boldsymbol{T}}\boldsymbol{P}\hat{\hat{\boldsymbol{X}}} - \frac{1}{2}\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{T}\boldsymbol{g}'(\hat{\hat{\boldsymbol{X}}}) = \boldsymbol{K}_1\hat{\hat{\boldsymbol{X}}} + \boldsymbol{U} \\
&\Rightarrow \boldsymbol{U} = (-\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{\boldsymbol{T}}\boldsymbol{P} - \boldsymbol{K}_1)\hat{\hat{\boldsymbol{X}}} - \frac{1}{2}\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{T}\boldsymbol{g}'(\hat{\hat{\boldsymbol{X}}}) \\
&\boldsymbol{U} = (-\boldsymbol{R}_2^{-1}\bar{\boldsymbol{B}}^{\boldsymbol{T}}\boldsymbol{P} - \boldsymbol{K}_1)\hat{\hat{\boldsymbol{X}}} - \frac{1}{2w_c^2}\bar{\boldsymbol{B}}^{T}\frac{\partial}{\partial\hat{\hat{\boldsymbol{X}}}}\boldsymbol{g}(\hat{\hat{\boldsymbol{X}}})
\end{aligned} \tag{107}$$

and $\boldsymbol{K}_1$ is constructed as (98). Similarly, the control input (107) must be rewritten in terms of the original system. Representing the terms in the parenthesis with $\bar{\boldsymbol{K}}$ we can write:

$$\boldsymbol{U} = \boldsymbol{K}\hat{\boldsymbol{X}} = \bar{\boldsymbol{K}}\boldsymbol{T}_t^{-1}\hat{\boldsymbol{X}} - \frac{1}{2w_c^2}\bar{\boldsymbol{B}}^{T}\frac{\partial}{\partial\hat{\hat{\boldsymbol{X}}}}\boldsymbol{g}(\hat{\hat{\boldsymbol{X}}}) \tag{108}$$

Applying (108) into (8) we have:

$$\begin{aligned}
&\dot{\hat{\boldsymbol{X}}} = \boldsymbol{A}\hat{\boldsymbol{X}} + \boldsymbol{B}\left(\bar{\boldsymbol{K}}\boldsymbol{T}_t^{-1}\hat{\boldsymbol{X}} - \frac{1}{2w_c^2}\bar{\boldsymbol{B}}^{T}\frac{\partial}{\partial\hat{\hat{\boldsymbol{X}}}}\boldsymbol{g}(\hat{\hat{\boldsymbol{X}}})\right) \\
&\Rightarrow \dot{\hat{\boldsymbol{X}}} = \left(\boldsymbol{A} + \boldsymbol{B}\bar{\boldsymbol{K}}\boldsymbol{T}_t^{-1}\right)\hat{\boldsymbol{X}} - \left(\frac{1}{2w_c^2}\right)\boldsymbol{B}\bar{\boldsymbol{B}}^{T}\frac{\partial}{\partial\hat{\hat{\boldsymbol{X}}}}\boldsymbol{g}(\hat{\hat{\boldsymbol{X}}})
\end{aligned} \tag{109}$$

and since $\hat{\boldsymbol{X}} = \boldsymbol{X} - \boldsymbol{X}_{cs}$, (109) can be expanded as:

$$\begin{aligned}
&\left(\dot{\boldsymbol{X}} - \dot{\boldsymbol{X}}_{cs}\right) = \left(\boldsymbol{A} + \boldsymbol{B}\bar{\boldsymbol{K}}\boldsymbol{T}_t^{-1}\right)\left(\boldsymbol{X} - \boldsymbol{X}_{cs}\right) - \left(\frac{1}{2w_c^2}\right)\boldsymbol{B}\bar{\boldsymbol{B}}^{T}\frac{\partial}{\partial\hat{\hat{\boldsymbol{X}}}}\boldsymbol{g}(\hat{\hat{\boldsymbol{X}}}) \\
&\Rightarrow \dot{\boldsymbol{X}} - \dot{\boldsymbol{X}}_{cs} = \boldsymbol{A}\boldsymbol{X} - \boldsymbol{A}\boldsymbol{X}_{cs} + \boldsymbol{B}\bar{\boldsymbol{K}}\boldsymbol{T}_t^{-1}\boldsymbol{X} - \boldsymbol{B}\bar{\boldsymbol{K}}\boldsymbol{T}_t^{-1}\boldsymbol{X}_{cs} - \left(\frac{1}{2w_c^2}\right)\boldsymbol{B}\bar{\boldsymbol{B}}^{T}\frac{\partial}{\partial\hat{\hat{\boldsymbol{X}}}}\boldsymbol{g}(\hat{\hat{\boldsymbol{X}}})
\end{aligned} \tag{110}$$

But when the consensus is achieved, the control input approaches to zero which means:

$\boldsymbol{U}_{cs} = \boldsymbol{B}\bar{\boldsymbol{K}}\boldsymbol{T}_t^{-1}\boldsymbol{X}_{cs} = \boldsymbol{0}_{m\cdot n_g \times 1}$ This reduces (110) to:

$$\dot{\boldsymbol{X}} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\bar{\boldsymbol{K}}\boldsymbol{T}_t^{-1}\boldsymbol{X} - \left(\frac{1}{2w_c^2}\right)\boldsymbol{B}\bar{\boldsymbol{B}}^T \frac{\partial}{\partial\hat{\bar{\boldsymbol{X}}}}\boldsymbol{g}(\hat{\bar{\boldsymbol{X}}}) \tag{111}$$

The second term must also be conveyed in terms of the original system. Similar to the single-input systems, $\boldsymbol{X}_D$ is only available to agent “i” (the agent which has access to the desired trajectory), therefore $\frac{\partial}{\partial\hat{\bar{\boldsymbol{X}}}}\boldsymbol{g}(\hat{\bar{\boldsymbol{X}}}) = \frac{\partial}{\partial\bar{\bar{\boldsymbol{X}}}}\boldsymbol{g}(\bar{\boldsymbol{X}})$. This term is expanded as below:

$$\frac{\partial g}{\partial\hat{\bar{\boldsymbol{X}}}} = \begin{cases} \left[\frac{\partial}{\partial\hat{\bar{\boldsymbol{X}}}}(\bar{\boldsymbol{X}}_i - \bar{\boldsymbol{X}}_D)\right](\boldsymbol{G}+\boldsymbol{G}^T)(\bar{\boldsymbol{X}}_i - \bar{\boldsymbol{X}}_D) & \text{if the agent "i" has access to the reference} \\ 0 & \text{if not} \end{cases}$$

$$= \begin{cases} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{n\cdot n_g \times n} (\boldsymbol{G}+\boldsymbol{G}^T)(\bar{\boldsymbol{X}}_i - \bar{\boldsymbol{X}}_D) & \text{if the agent "i" has access to the reference} \\ 0 & \text{if not} \end{cases} \tag{112}$$

and finally, by converting $\bar{\boldsymbol{X}}_i - \bar{\boldsymbol{X}}_D$ back to the original state vectors, the term for tracking penalty function is rewritten as:

$$\frac{\partial g}{\partial \boldsymbol{X}} = \begin{cases} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{n \cdot n_g \times n} (\boldsymbol{G} + \boldsymbol{G}^T)\boldsymbol{T}_t^{-1}(\boldsymbol{X}_i - \boldsymbol{X}_D) & \text{if the agent "i" has access to the reference} \\ 0 & \text{if not} \end{cases}$$

(113)

Now the control law can be written in terms of the original system.

$$\boldsymbol{U} = \bar{\boldsymbol{K}}\boldsymbol{T}_t^{-1}\boldsymbol{X} - \frac{1}{2w_c^2}\bar{\boldsymbol{B}}^T \frac{\partial}{\partial \boldsymbol{X}} g(\boldsymbol{X}) \tag{114}$$

Since the system dynamics is a multi-input one and is in general form, there is no direct approach to construct the matrix $\boldsymbol{R}_1$ such that the Algebraic Riccati Equation (ARE) becomes a linear equation. The only approach is applying numerical methods. In this study, we used the methods described in [87].

$\boldsymbol{R}_1$ should be constructed such that it is Positive Semi-Definite (P.S.D). A good choice can be as below:

$$\boldsymbol{R}_1 = \begin{bmatrix} w_1^2\boldsymbol{L}^2 & \boldsymbol{0}_{n\times n} & \cdots & \boldsymbol{0}_{n\times n} \\ \boldsymbol{0}_{n\times n} & w_2^2\boldsymbol{L}^2 & \cdots & \boldsymbol{0}_{n\times n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}_{n\times n} & \boldsymbol{0}_{n\times n} & \cdots & w_n^2\boldsymbol{L}^2 \end{bmatrix}_{n\cdot n_g \times n \cdot n_g} \tag{115}$$

where w_i is the tunable weight for the “i”th state and $\boldsymbol{L}$ is the Laplacian matrix. $\boldsymbol{R}_1$ is P.S.D according to the lemma 2.3.

3.3.2 Discussion on the distributed cooperative control

Similar to the discussion for the single-input systems, $\boldsymbol{K}_1$ in (102) is only a constant gain and therefore the terms in the parenthesis $\left(-\boldsymbol{R}_2^{-1}\overline{\boldsymbol{B}}^T\boldsymbol{P}-\boldsymbol{K}_1\right)$ is a linear function of the Laplacian matrix. For investigating the effect of $\boldsymbol{T}_t^{-1}$ we expand the first term of (114) for a special case when $n=m=n_g=2$. According to the theorem 5, in a general form the two matrices of $\overline{\boldsymbol{B}}$ and $\boldsymbol{K}_1$ can be parametrically represented as below:

$$\overline{\boldsymbol{B}}=\begin{bmatrix} b_1 & -b_1 \\ b_2 & 1-b_2 \end{bmatrix}\otimes \boldsymbol{I}_2 \tag{116}$$

now the first term is expanded as:

$$\boldsymbol{K}_1=\begin{bmatrix} -a_0 & -a_1 \\ -a_0 & -a_1 \end{bmatrix}\otimes \boldsymbol{I}_2 \tag{117}$$

$$\begin{aligned}
\boldsymbol{U} &= (-\boldsymbol{R}_2^{-1}\overline{\boldsymbol{B}}^T\boldsymbol{P}-\boldsymbol{K}_1)\boldsymbol{T}_t^{-1}\boldsymbol{X} \\
&= \left(\left(\frac{-1}{w_c^2}\boldsymbol{I}_4\right)\left(\begin{bmatrix} b_1 & -b_1 \\ b_2 & 1-b_2 \end{bmatrix}\otimes \boldsymbol{I}_2\right)\begin{bmatrix} \boldsymbol{P}_{11} & \boldsymbol{P}_{12} \\ \boldsymbol{P}_{21} & \boldsymbol{P}_{22} \end{bmatrix}-\left(\begin{bmatrix} -a_0 & -a_1 \\ -a_0 & -a_1 \end{bmatrix}\otimes \boldsymbol{I}_2\right)\right)\begin{bmatrix} \bar{t}_{11}\boldsymbol{I}_2 & \bar{t}_{12}\boldsymbol{I}_2 \\ \bar{t}_{21}\boldsymbol{I}_2 & \bar{t}_{22}\boldsymbol{I}_2 \end{bmatrix}\boldsymbol{X} \\
&= \frac{-1}{w_c^2}\begin{bmatrix} (b_1\boldsymbol{P}_{11}+b_2\boldsymbol{P}_{21}+a_0\boldsymbol{I}_2)\bar{t}_{11}+(b_1\boldsymbol{P}_{12}+b_2\boldsymbol{P}_{22}+a_1\boldsymbol{I}_2)\bar{t}_{21} & (b_1\boldsymbol{P}_{11}+b_2\boldsymbol{P}_{21}+a_0\boldsymbol{I}_2)\bar{t}_{12}+(b_1\boldsymbol{P}_{12}+b_2\boldsymbol{P}_{22}+a_1\boldsymbol{I}_2)\bar{t}_{22} \\ (-b_1\boldsymbol{P}_{11}+(1-b_2)\boldsymbol{P}_{21}+a_0\boldsymbol{I}_2)\bar{t}_{11}+(-b_1\boldsymbol{P}_{12}+(1-b_2)\boldsymbol{P}_{22}+a_1\boldsymbol{I}_2)\bar{t}_{21} & (-b_1\boldsymbol{P}_{11}+(1-b_2)\boldsymbol{P}_{21}+a_0\boldsymbol{I}_2)\bar{t}_{21}+(-b_1\boldsymbol{P}_{12}+(1-b_2)\boldsymbol{P}_{22}+a_1\boldsymbol{I}_2)\bar{t}_{22} \end{bmatrix}\boldsymbol{X}
\end{aligned} \tag{118}$$

where $\bar{t}_{ij}$ s are the entries of $\boldsymbol{T}_t^{-1}$. Since any entry in $\boldsymbol{P}$ is a linear function of the Laplacian matrix, the terms containing its entries guarantee that information exchange occurs only among the neighbors of each agent. The other term of characteristic equation coefficients and identity matrices represents the state of the agent itself. Therefore the two terms of $\boldsymbol{K}_1$

and $\boldsymbol{T}_t^{-1}$ do not affect the communication topology since the control law only requires local information gathered from neighbors of each agent.

3.3.3 Defining the proper desired trajectory

It was said that the control input approaches to zero when consensus is achieved (i.e. $\boldsymbol{U}_{cs} = \boldsymbol{0}_{m.n_g \times 1}$). According to this fact, an appropriate desired trajectory could be designed along which our system is desired to follow. When all the inputs become zero it can be assumed that the input vector is a scalar. In other words, regardless of how many inputs the system has, when the agents achieve consensus and follow a desired trajectory they have no longer any inputs. Without loss of generality we consider that all the inputs become a constant of u (we know that $u = 0$) and the dynamics of each agent can be rewritten as:

$$\dot{\boldsymbol{X}}_i = \boldsymbol{A}_g \boldsymbol{X}_i + \boldsymbol{B}_g \boldsymbol{U}_i = \boldsymbol{A}_g \boldsymbol{X}_i + \left(\sum_{j=1}^{m} \boldsymbol{B}_j \right) u_i \tag{119}$$

where $\boldsymbol{B}_j$ s $j = 1, 2, ..., m$ are the "m" columns of the input matrix $\boldsymbol{B}_g$. Equation (119) means that the system can be seen as a single-input one upon achieving consensus. The original multi input system will be equivalent to a single-input system for which the input matrix is the sum of the columns of the original one when the agents achieve consensus. Let us call $\sum_{j=1}^{m} \boldsymbol{B}_j$ as $\boldsymbol{B}_2$.

The problem is to find an appropriate desired trajectory for a single-input system with a dynamics of $\dot{\boldsymbol{X}}_i = \boldsymbol{A}_g \boldsymbol{X}_i + \boldsymbol{B}_2 u_i$ along which the system could follow. The two matrices $\boldsymbol{N}$ and $\boldsymbol{O}$ are introduced similar to those for single-input systems.

$$\boldsymbol{N} = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}_{1\times n} \tag{120}$$

$$\boldsymbol{O} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}_{(n-1)\times n} \tag{121}$$

The desired trajectory $\boldsymbol{X}_D$ must satisfy the system therefore:

$$\dot{\boldsymbol{X}}_D = \boldsymbol{A}_g \boldsymbol{X}_D + \boldsymbol{B}_2 \boldsymbol{U}_D \Rightarrow \boldsymbol{B}_2 \boldsymbol{U}_D = \dot{\boldsymbol{X}}_D - \boldsymbol{A}_g \boldsymbol{X}_D \tag{122}$$

We start with the left hand side of the equation.

$$\begin{aligned} &-\boldsymbol{B}_2\boldsymbol{U}_D = -\boldsymbol{B}_2\boldsymbol{U}_D \\ &\Rightarrow -\boldsymbol{O}\boldsymbol{B}_2\boldsymbol{U}_D = -\boldsymbol{O}\boldsymbol{B}_2\boldsymbol{U}_D \\ &\Rightarrow -\boldsymbol{N}\boldsymbol{B}_2\boldsymbol{O}\boldsymbol{B}_2\boldsymbol{U}_D = -\boldsymbol{O}\boldsymbol{B}_2(\boldsymbol{N}\boldsymbol{B}_2)\boldsymbol{U}_D \\ &\Rightarrow \boldsymbol{N}\boldsymbol{B}_2\boldsymbol{O}(\boldsymbol{A}_g\boldsymbol{X}_D - \dot{\boldsymbol{X}}_D) = \boldsymbol{O}\boldsymbol{B}_2\boldsymbol{N}(\boldsymbol{A}_g\boldsymbol{X}_D - \dot{\boldsymbol{X}}_D) \\ &\Rightarrow \boldsymbol{N}\boldsymbol{B}_2\boldsymbol{O}\boldsymbol{A}_g\boldsymbol{X}_D - \boldsymbol{N}\boldsymbol{B}_2\boldsymbol{O}\dot{\boldsymbol{X}}_D = \boldsymbol{O}\boldsymbol{B}_2\boldsymbol{N}\boldsymbol{A}_g\boldsymbol{X}_D - \boldsymbol{O}\boldsymbol{B}_2\boldsymbol{N}\dot{\boldsymbol{X}}_D \\ &\Rightarrow (\boldsymbol{N}\boldsymbol{B}_2\boldsymbol{O}\boldsymbol{A}_g - \boldsymbol{O}\boldsymbol{B}_2\boldsymbol{N}\boldsymbol{A}_g)\boldsymbol{X} = (\boldsymbol{N}\boldsymbol{B}_2\boldsymbol{O} - \boldsymbol{O}\boldsymbol{B}_2\boldsymbol{N})\dot{\boldsymbol{X}}_D \end{aligned} \tag{123}$$

Since $\boldsymbol{N}\boldsymbol{B}_2$ is a scalar, it could be placed anywhere in the equation. Expanding the two sides of (123) we get:

$$\begin{bmatrix} -\bar{b}_2 & \bar{b}_1 & 0 & \cdots & 0 \\ -\bar{b}_3 & 0 & \bar{b}_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\bar{b}_n & 0 & 0 & \cdots & \bar{b}_1 \end{bmatrix} \dot{\boldsymbol{X}}_D = \begin{bmatrix} \bar{b}_1 a_{21} - \bar{b}_2 a_{11} & \bar{b}_1 a_{22} - \bar{b}_2 a_{12} & \cdots & \bar{b}_1 a_{2n} - \bar{b}_2 a_{1n} \\ \bar{b}_1 a_{31} - \bar{b}_3 a_{11} & \bar{b}_1 a_{32} - \bar{b}_3 a_{12} & \cdots & \bar{b}_1 a_{3n} - \bar{b}_3 a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{b}_1 a_{n1} - \bar{b}_n a_{11} & \bar{b}_1 a_{n1} - \bar{b}_n a_{12} & \cdots & \bar{b}_1 a_{nn} - \bar{b}_n a_{1n} \end{bmatrix} \boldsymbol{X}_D \tag{124}$$

where $\bar{b}_i$ s $1 \le i \le n$ are the entries of the column vector $\boldsymbol{B}_2$. The system of differential equations above has infinitely many solutions since it has $n-1$ differential equations with n unknowns. For selecting a proper desired trajectory we need to set at least one of the states. Once a state is set, the other states can be determined by solving the system of “n-1” equations with “n-1” unknowns.

3.4 Illustrative examples

Now application of the algorithm is shown through two examples. Like the examples in chapter 2, the first example is pure mathematical and has been designed to show how the algorithm works. In the second example, we implement the algorithm in order to synchronize the lateral motions of a group of aircrafts.

Example 3.1:

Suppose that we have two identical agents each with a dynamic equation of:

$$\boldsymbol{X}_i = \boldsymbol{A}_g \boldsymbol{X}_i + \boldsymbol{B}_g \boldsymbol{U}_i \tag{125}$$

where

$$\boldsymbol{A}_g = \begin{bmatrix} 1 & 2 \\ 1 & -1 \end{bmatrix}, \quad \boldsymbol{B}_g = \begin{bmatrix} 3 & 2 \\ 4 & 3 \end{bmatrix} \tag{126}$$

The two agents communicate with each other and the corresponding graph is as shown in Figure 8.

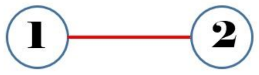

Figure 8. The graph associated with the two agents in Example 3.1

According to the graph, it is obvious that the Laplacian matrix has the form of:

$$\boldsymbol{L} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{127}$$

For finding the appropriate desired trajectory, we first make the equation (119) by adding the columns of the input matrix $\boldsymbol{B}_g$. The new equation will be:

$$\begin{aligned}&\dot{\boldsymbol{X}}_i = \boldsymbol{A}_g\boldsymbol{X}_i + \boldsymbol{B}_g\boldsymbol{U}_i = \boldsymbol{A}_g\boldsymbol{X}_i + \left(\sum_{j=1}^{m}\boldsymbol{B}_j\right)u_i = \boldsymbol{A}_g\boldsymbol{X}_i + \boldsymbol{B}_2u_i \\ &\Rightarrow \dot{\boldsymbol{X}}_i = \begin{bmatrix}1 & 2\\ 1 & -1\end{bmatrix}\boldsymbol{X}_i + \begin{bmatrix}5\\ 7\end{bmatrix}u_i\end{aligned} \tag{128}$$

now the system (124) should be solved to determine the desired trajectory. In this particular example, the system is reduced to two simple differential equations as below:

$$\begin{aligned}&\begin{bmatrix}-\bar{b}_2 & -\bar{b}_1\end{bmatrix}\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}\bar{b}_1a_{21} - \bar{b}_2a_{11} & \bar{b}_1a_{22} - \bar{b}_2a_{12}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}\\ &\Rightarrow \begin{bmatrix}-7 & -5\end{bmatrix}\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}(5-7) & (-5-14)\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}\\ &\Rightarrow -7\dot{x}_1 - 5\dot{x}_2 = -2x_1 - 19x_2\end{aligned} \tag{129}$$

where $\bar{b}_i$ s are the entries for $\boldsymbol{B}_2$. Suppose that the system is desired to reach a point. That means the terms on the left side of the equation (129) are all zeros and therefore any point of the line $-2x_1 - 19x_2 = 0$ or $x_2 = -(2/19)x_1$ can be assigned for the system to reach. We pick the point $(19,-2)$. The results are shown in Figures. 7 (a)-(d). The weighting parameters have been selected to be $w_1 = 5,\ w_2 = 2,\ w_c = 0.5,\ w_{d_1} = 2,\ w_{d_2} = 0.6$. Agent 1 has access to the reference desired trajectory and the initial conditions of the two agents have been set to $X_1(0) = (10,10),\ X_2(0) = (0,6)$. Figures. 7 (a) and (b) show the phase plane of the agents. There is always a stable focus, which exactly lies on the line $X_2 = -\dfrac{2}{19}X_1$.

In this example the focus lies exactly on the point $(19,-2)$. The states have been shown in Figures 9 (c) and (d). It is seen that the agent 2 will follow the first one and both reach the point $(19,-2)$ at the end.

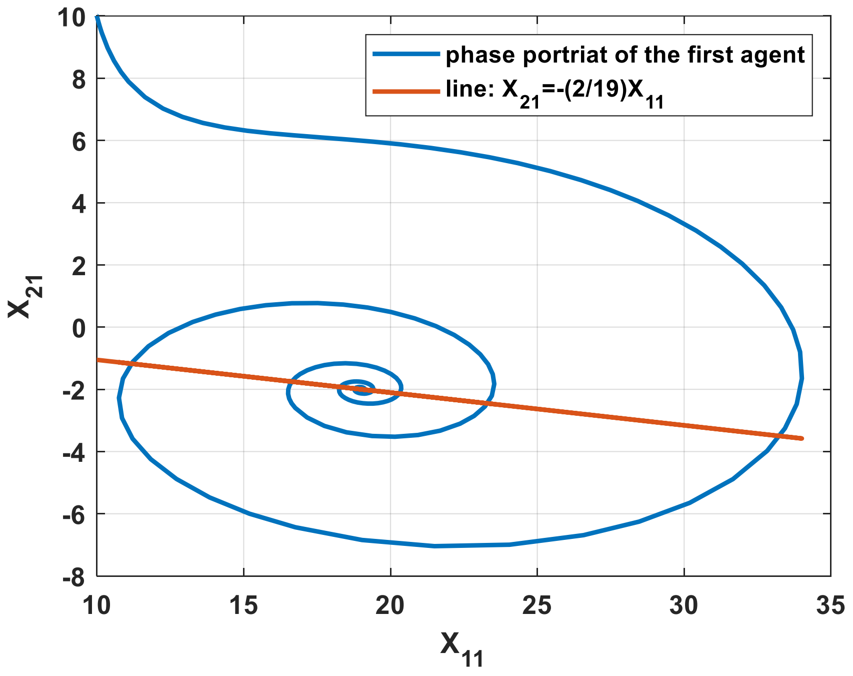

Figure 9 (a). Phase plane for the first agent

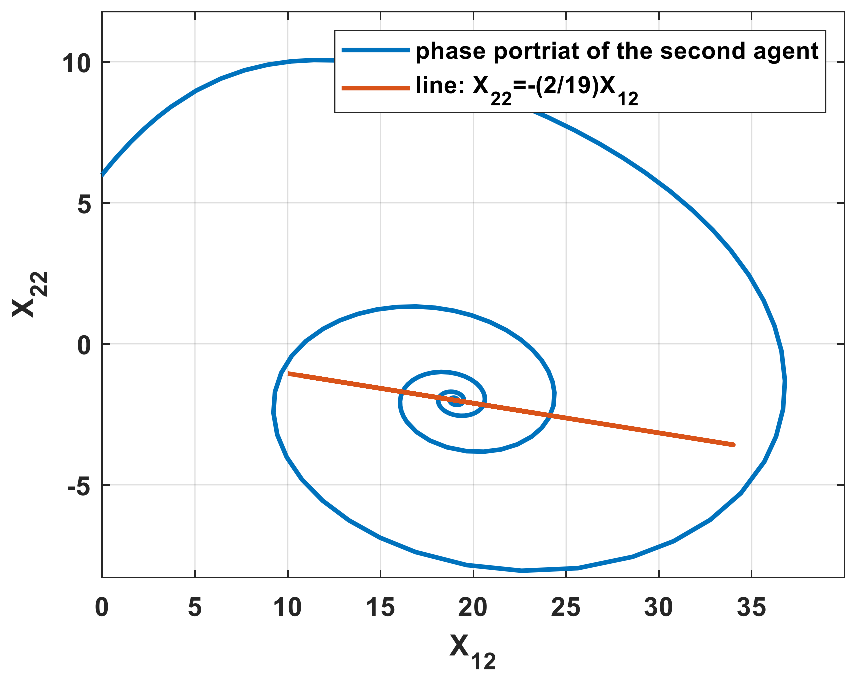

Figure 9 (b). Phase plane for the second agent

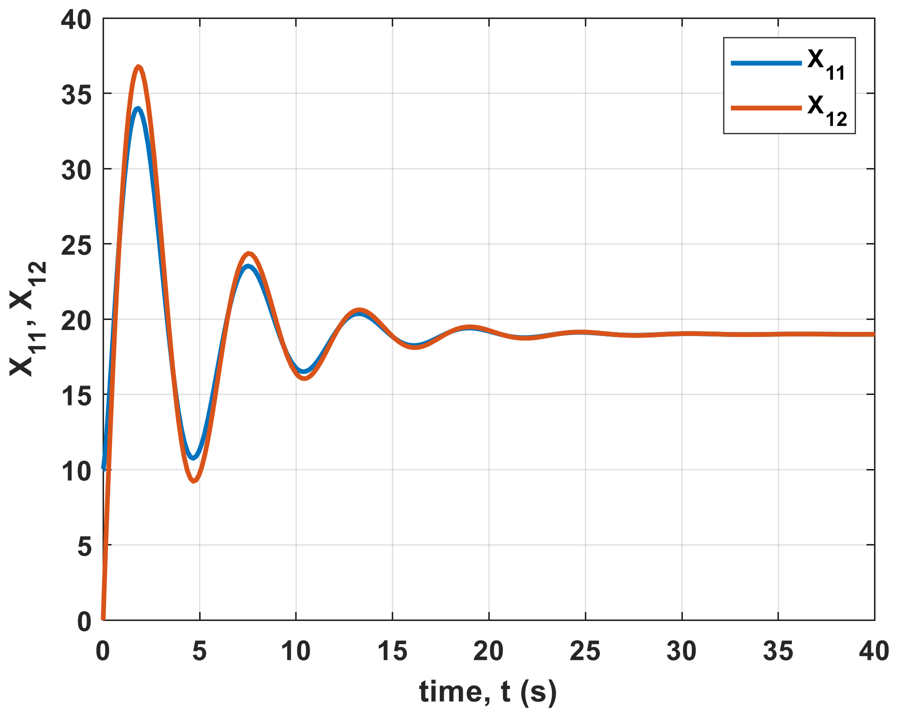

Figure 9 (c). The first states of the agents

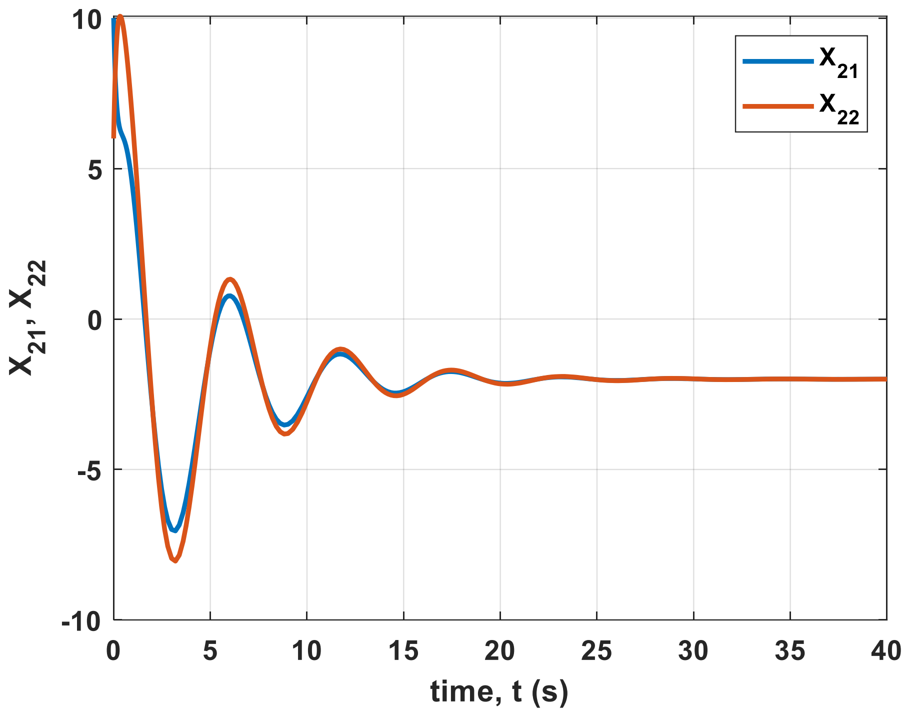

Figure 9 (d). The second states of the agents

Example 3.2. Lateral motion of a group of X-29A

As an engineering application, we borrow an example from [36] and [37]. A group of four identical Grumman X-29A aircrafts is needed to reach consensus. The LTI dynamics of each agent can be described as below:

$$\dot{\boldsymbol{X}} = \boldsymbol{A}_{X-29A}\boldsymbol{X}_i + \boldsymbol{B}_{X-29A}\boldsymbol{U}_i \tag{130}$$

where the two matrices of state and input are:

$$\boldsymbol{A}_{X-29A} = \begin{bmatrix} -2.59 & 0.997 & -16.55 & 0 \\ -0.1023 & -0.0679 & 6.779 & 0 \\ -0.0603 & -0.9928 & -0.1645 & 0.04413 \\ 1 & 0.07168 & 0 & 0 \end{bmatrix}$$
$$\boldsymbol{B}_{X-29A} = \begin{bmatrix} 1.347 & 0.2365 \\ 0.09194 & -0.07056 \\ -0.0006141 & 0.0006866 \\ 0 & 0 \end{bmatrix} \tag{131}$$

The state of each agent is defined as $\boldsymbol{X}_i = \begin{bmatrix} p_i & r_i & \beta_i & \phi_i \end{bmatrix}^T$ where p, r, β and ϕ are the roll rate, yaw rate, sideslip angle and bank angle respectively. These parameters are shown in Figure 10.

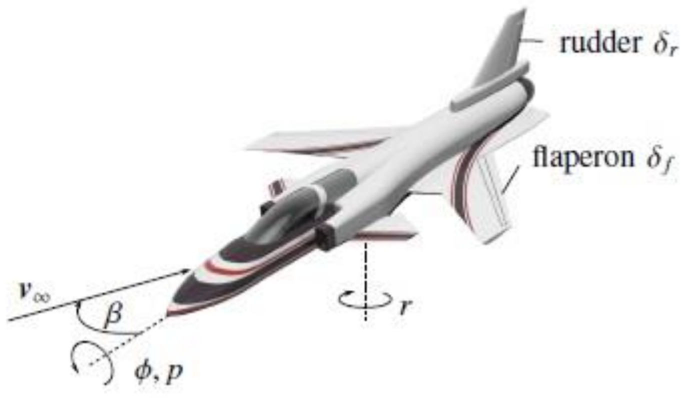

Figure 10. Grumman X-29A aircraft

Two different communication topologies of the agents will be considered. For both topologies the parameters have been chosen as $w_1 = 1,\ w_2 = 2,\ w_3 = 3,\ w_4 = 2,\ w_c = 0.5,\ w_{d_1} = 1,\ w_{d_2} = 20,\ w_{d_3} = 10$ and $w_{d_4} = 5$. The vector of initial condition has been set to be: $[0.5\ \ -1\ \ -1.5\ \ 1\ \ -0.14\ \ 0.5\ \ 0.24\ \ 0.5\ \ -0.2\ \ -0.6\ \ 0.3\ \ -0.2\ \ -0.1\ \ 0.2\ \ 0.9\ \ -0.5]^T$. In order to design an appropriate desired trajectory, the equation (119) is made first and then the system of differential equations (124) should be constructed and solved. In this example, it is supposed that the objective is to reach periodic functions. For keeping the amplitude reasonable, they are multiplied by constants between 0 and 1. It is still assumed that only the first aircraft has access to the desired trajectory. We set the last state of the first agent ϕ_1 to be $-0.76\sin t$. By solving (124), the rest of the states are determined as below:

$$\begin{aligned} p_1 &= -0.23\cos t - 0.76\sin t \\ r_1 &= -0.027\cos t + 0.72\sin t \\ \beta_1 &= -0.014\sin t - 0.0099\cos t \end{aligned} \tag{132}$$

i) For the first case the configuration is as what is shown in Figure 11.

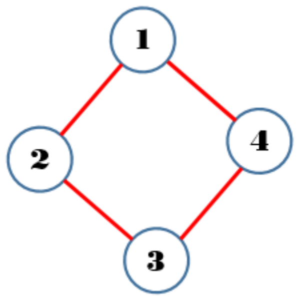

Figure 11. Topology of the aircrafts for the first case

The associated Laplacian matrix will be as below:

$$\boldsymbol{L} = \begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix} \tag{133}$$

the control input can be found by using the algorithm described in section 3.3.1 and equation (124). The results are shown in Figures 12 (a)-(d).

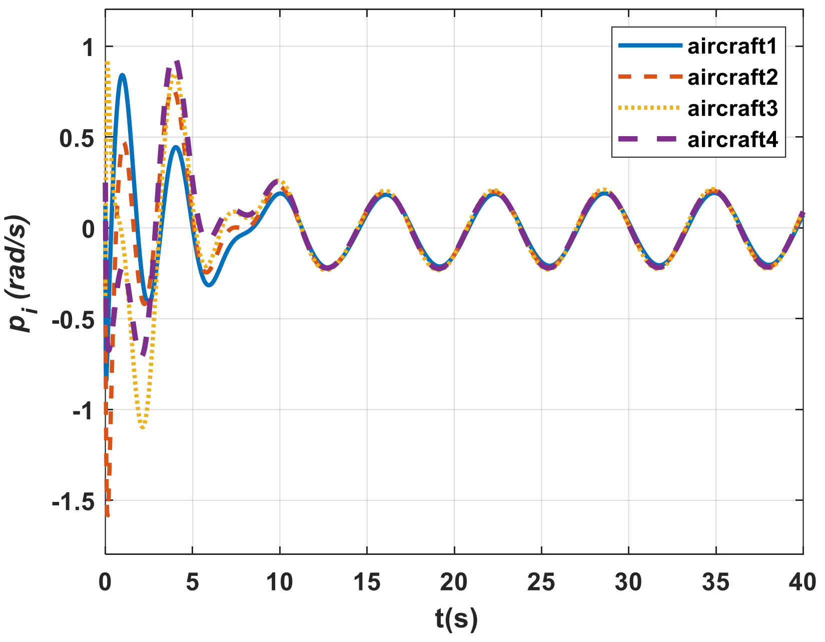

Figure 12 (a). Roll rates for the first case

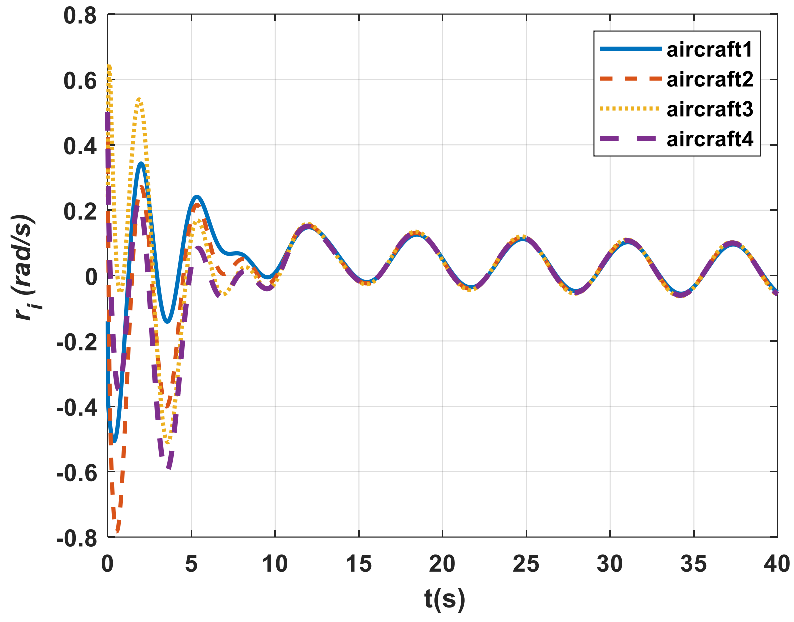

Figure 12. (b) Yaw rates for the first case

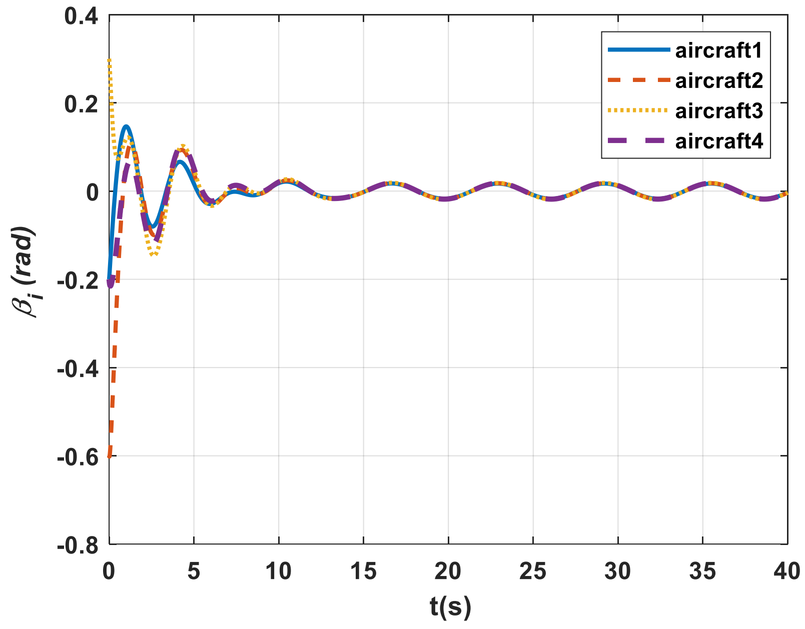

Figure 12. (c) Sideslip angles for the first case

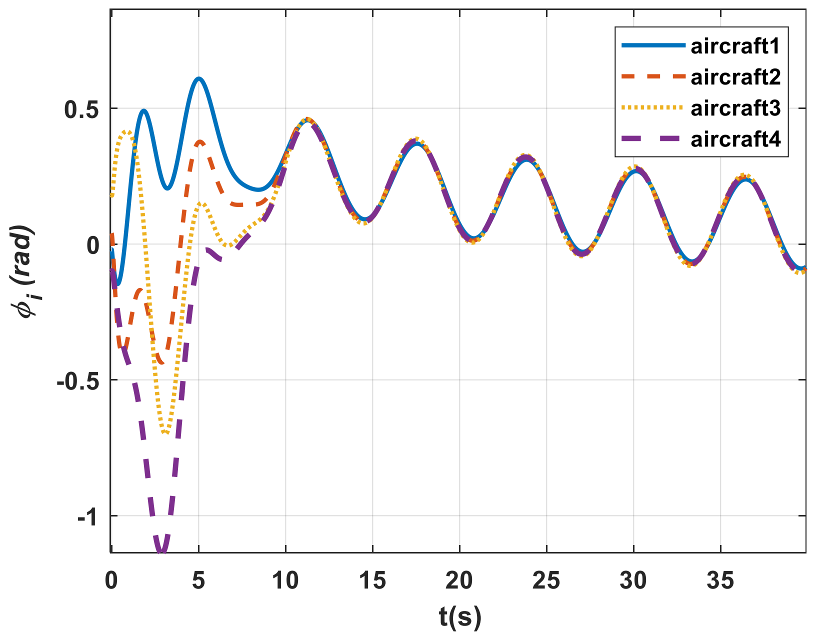

Figure 12 (d) Bank (roll) angles for the first case

It is seen that all the states reach the desired periodic functions in half a minute. Next another topology is investigated in which the agents 2,3 and 4 do not have direct connection to each other.

ii) The topology for the second case is shown in Figure 13.

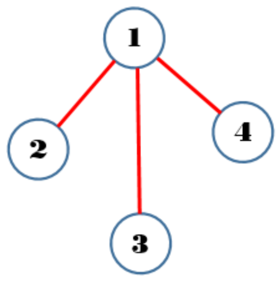

Figure 13. Topology of the aircrafts for the second case

In this case, the agent 1 exchange information with all the other three agents while they do not have direct access to each other. The process for finding the control input is still the same but the new Laplacian matrix is as below:

$$\boldsymbol{L} = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \tag{134}$$

The results are shown in Figures 14. (a)-(d).

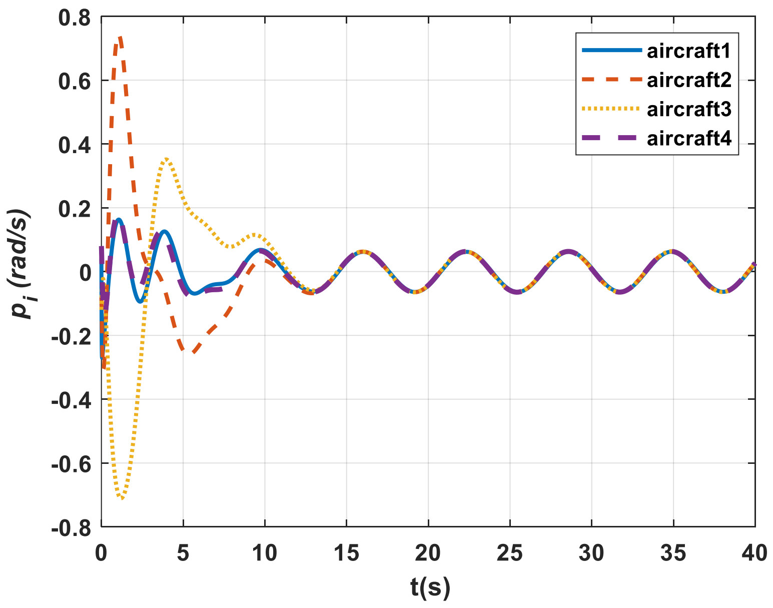

Figure 14 (a) Roll rates for the second case

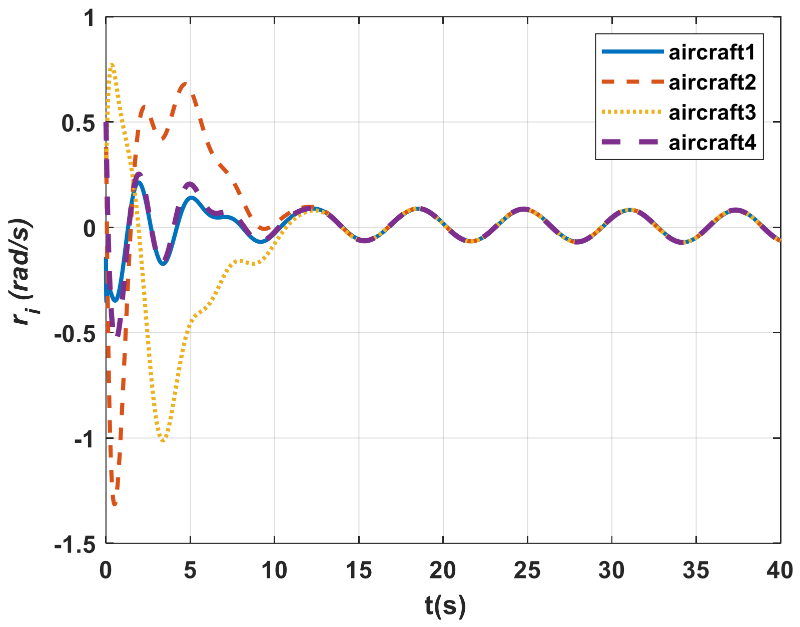

Figure 14 (b) Yaw rates for the second case

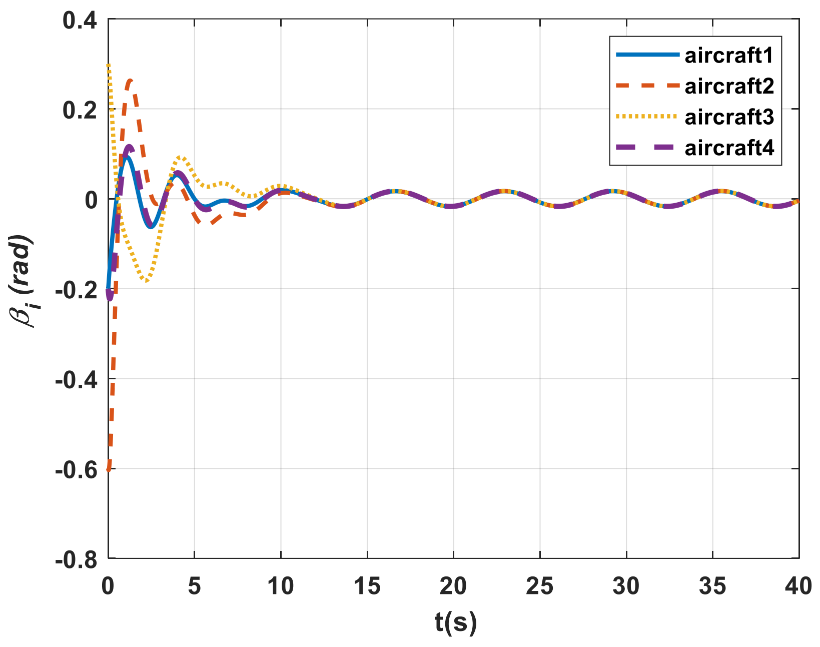

Figure 14 (c) Sideslip angles for the second case

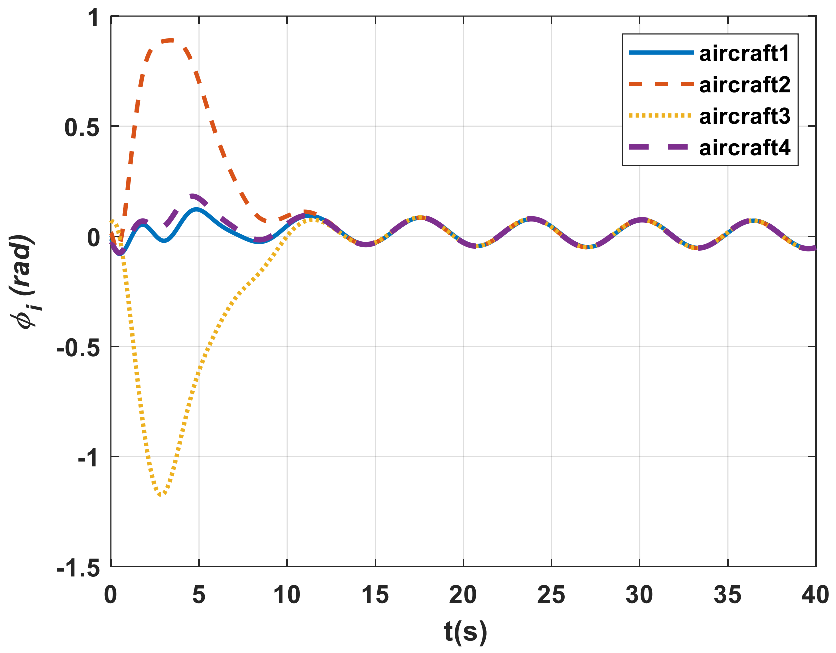

Figure 14 (d) Bank angles for the second case

It is seen that in the second case the convergence rate decreases but the states finally reach the desired trajectory. It is worth noting that none of the topologies is "fully connected". However, since the agents 2, 3 and 4 do not exchange information directly from each other in the second case, it takes longer for the agents to achieve consensus.

Robustness of the Algorithm

In many practical real situations the state matrix could get disturbed due to imperfection in communication topologies or limited bandwidth. In these situations, we need to make sure that the algorithm is robust enough to overcome the disturbances, stay stable and make the system achieve consensus even with a disturbed state matrix. In order to test the robustness of the control law in this example, I run the algorithm in way that the control law is made by the original state matrix while during integration step the disturbed state matrix has a error of 30% (i.e. $0.7\boldsymbol{A} \le \boldsymbol{A}_{Disturbed} \le 1.3\boldsymbol{A}$). The results for the maximum and minimum error have been shown in figures 15 and 16 (a)-(d). Figure 15 shows the states of the aircrafts when the disturbed state matrix is 70% of the original state matrix. It is seen that although it takes longer for the state to converge to the desired trajectory, the algorithm is still able to make the system achieve consensus.

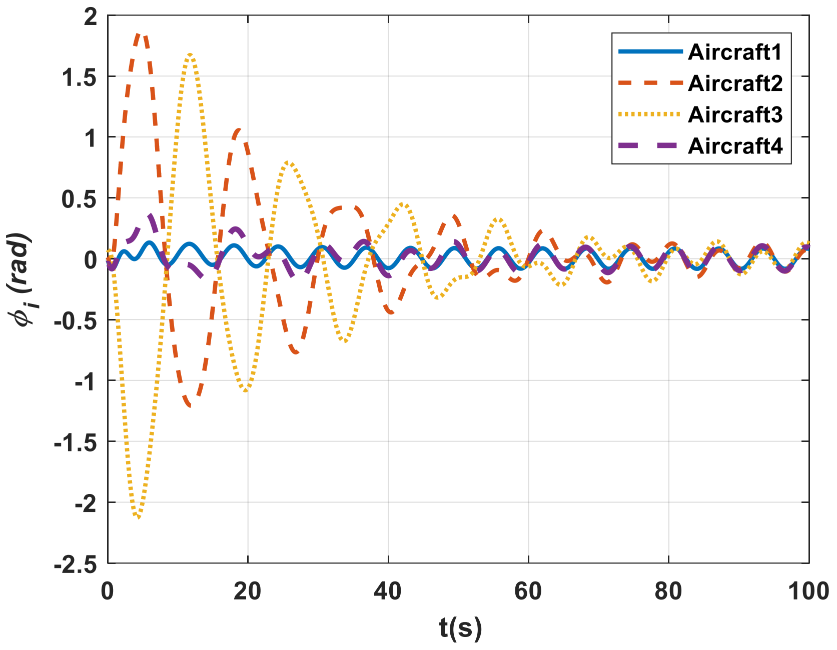

Figure 15 (a). Bank angles with a disturbance of -0.3A

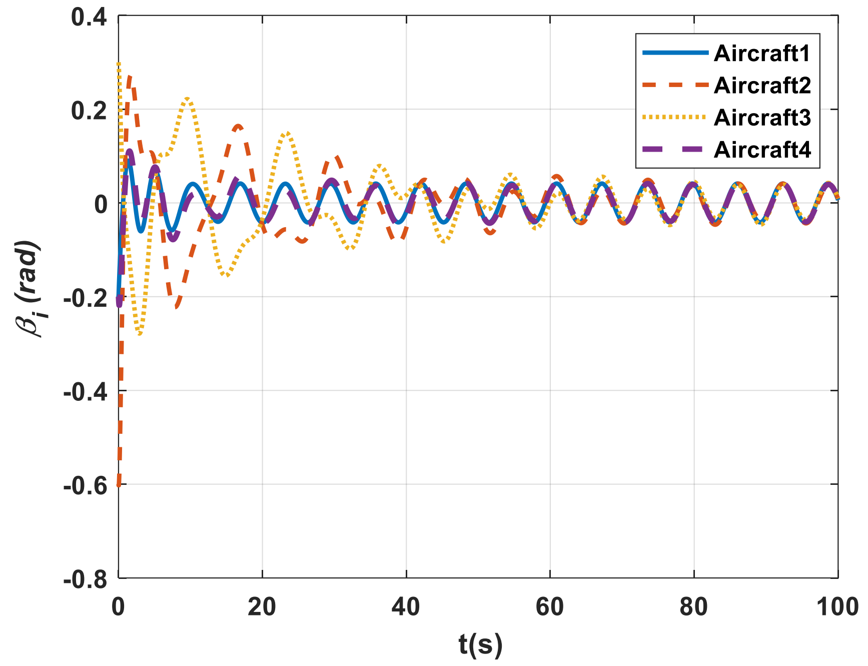

Figure 15 (b). Slideslip angles with a disturbance of -0.3A

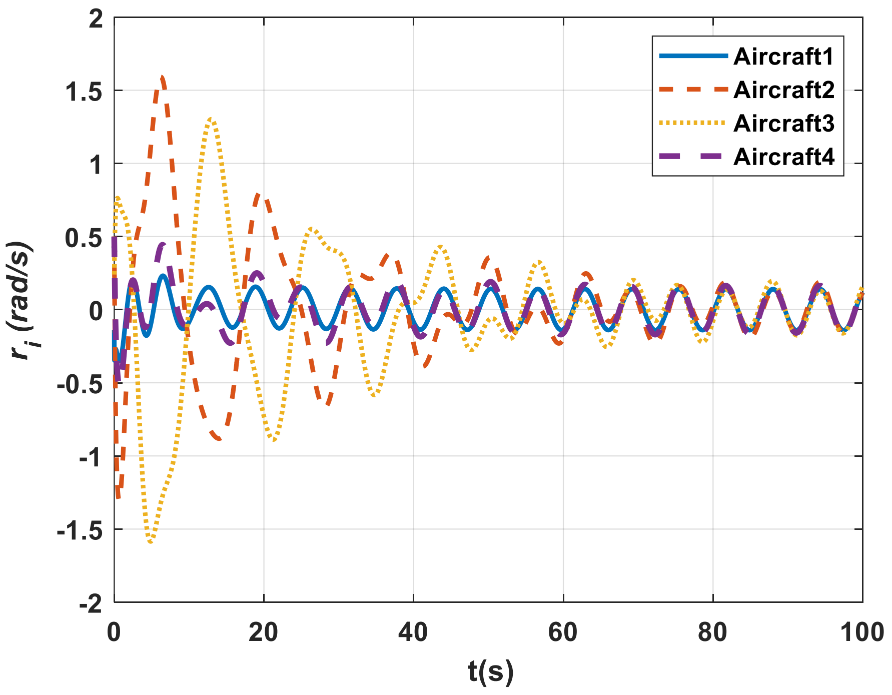

Figure 15 (c). Yaw rates with a disturbance of -0.3A

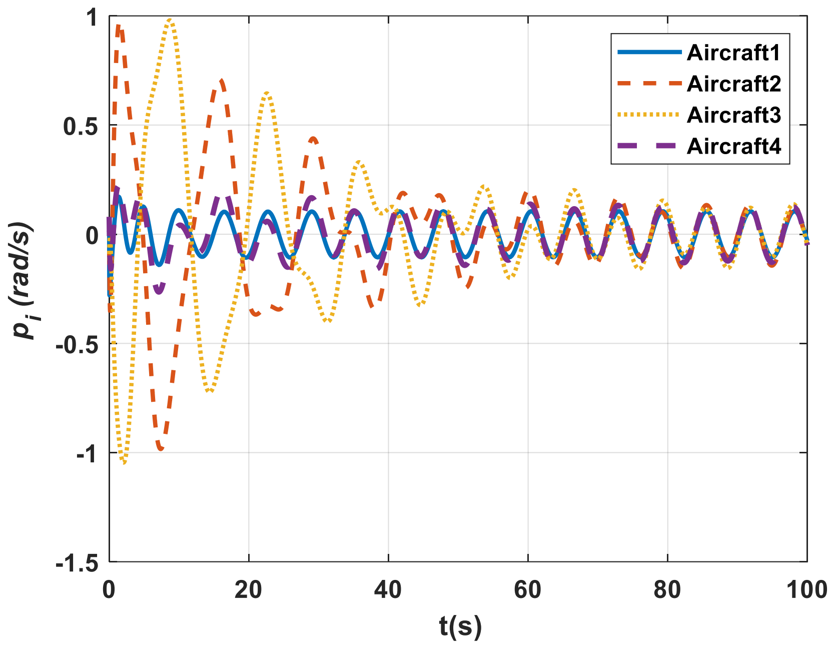

Figure 15 (d). Roll rates with a disturbance of -0.3A

In figure 16, the disturbed state matrix is 130% of the original state matrix. It is seen that this disturbance does not affect the effectiveness of the algorithm which means the control law is quite robust.

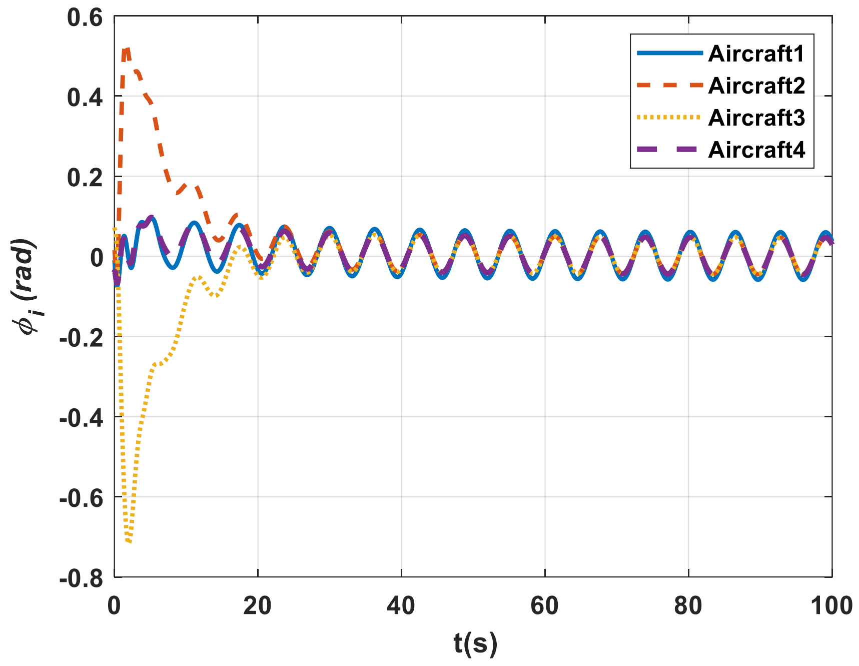

Figure 16 (a) Bank angles with a disturbance of +0.3A

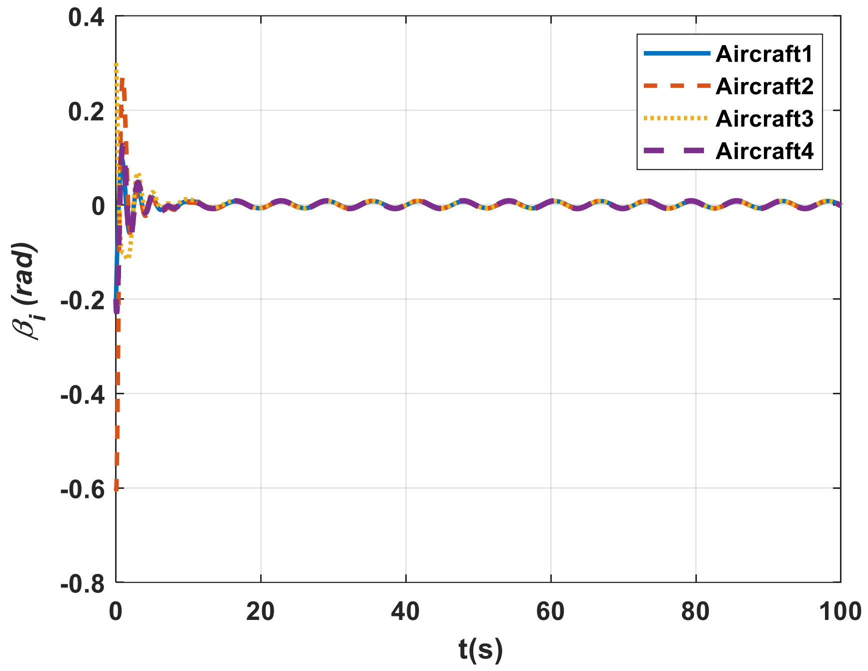

Figure 16 (b) Slideslip angles with a disturbance of +0.3A

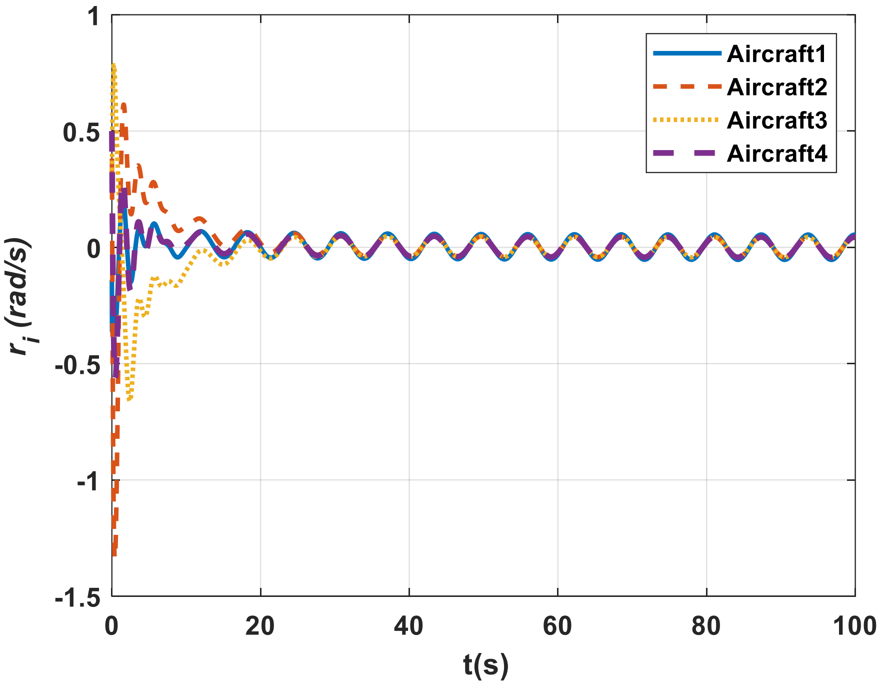

Figure 16 (c) Yaw rates with a disturbance of +0.3A

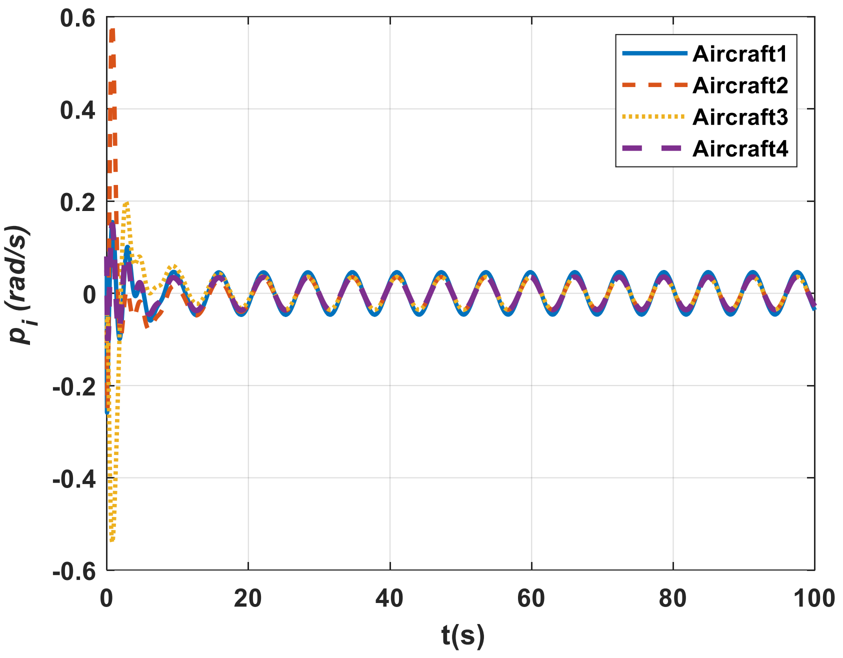

Figure 16 (d) Roll rates with a disturbance of +0.3A

4. Path planning of UAV fire fighters via Partially Observable Markov Decision Process (POMDP)

4.1 Overview

In this chapter, a path planning approach is designed in order to guide a group of UAVs that are assigned to track active wildfire fronts. The algorithm is developed based on the theory of Partially Observable Markov Decision Process (POMDP)[88, 89]. The UAVs are equipped with sensors, which measure the position of the fire fronts, and then the tracks are obtained by using a Kalman filter. Finally, the process completes by calculating the control variables for the UAVs, which are forward accelerations and bank angles. We investigate the problem of imposing practical constraints for the controls and collision avoidance between the UAVs. Since wildfires are characterized by fast and randomly evolving process, the major challenge of this study is that the UAVs must make the tracking decisions autonomously and be able to track the fire fronts, which are evolving over time in any random directions.

Problem specification is given in section 4.2 POMDP formulation and ingredients are explained in section 4.3 Nominal Belief-state Optimization (NBO) method is developed for our model in section 4.4 State-transition laws will be described in sections 4.5 and 4.7. The model for finding the coordinates of the fire fronts has been briefly described in section

4.6. In this study we assume this data is given. Three different scenarios are defined and investigated in section 4.7 as the main results in order to show the performance of the designed algorithm.

4.2 Problem specification

We assume that the fire fronts spread on the ground in 2-D. Therefore, we simplify the motion model of the UAVs by considering the altitude as a constant. The position of the UAVs are determined by the control parameters, which are assumed to be forward acceleration and bank angle. The speed of the UAVs is controlled by the first one and the second one controls the heading angles. Both control variables are restricted to be chosen within certain limits. The fire fronts spread randomly over time and can be seen as moving targets maneuvering in random directions. UAVs are equipped with sensors, which measure the positions of the active fire fronts (the flames that are evolving over time). There would always be random errors depending on the locations of sensors (UAVs) and the fire fronts. The path-planning problem of the UAVs is solved when the mean-squared error between the tracks and the fire fronts are minimized. We apply a Partially Observable Markov Decision Process (POMDP) in order to solve the path-planning problem. The process and ingredients of a POMDP problem are introduced and defined in details in the next section.

4.3 POMDP: Formulation and Ingredients

Markov Decision Processes (MDPs) are discrete-time control processes that model "decision making" in situations where the results are partially under the control of the decision maker and partially random. In case of a wildfire, the situations (so-called "states") of the fire fronts are not fully observable but only partially observable, which means the decision at each step must be made only based on an observation. According to the POMDP algorithms, five elements ("states", "actions", "observation law", "state-transition law," and "belief state") are defined and then a cost function is constructed which needs to be minimized. The set of the actions which minimizes the cost function is called the optimal policy. Three subsystems of states, targets, and trackers are introduced, and for each one we develop a state and a state-transition model. These elements are described in more details here:

States: In a POMDP problem, states are the parameters, that possibly evolve over time. For our path-planning problem it is better to define three subsystem of sensors, targets (fire fronts) and trackers. Therefore, the state at time k can be shown as a set of $x_k = (s_k, \chi_k, \xi_k, \boldsymbol{P}_k)$ where s_k and χ_k are the states of the sensors and fire fronts respectively and $(\xi_k, \boldsymbol{P}_k)$ is the tracker state. Sensor states include the locations, the speed and heading angles of the UAVs. The discrete points of the fire fronts is required for defining the front state. The states of the fire fronts only include the 2-D positions of the active flames. In section 6 we explain how to define the momentary model of the fire fronts for each time-step. The trackers' state are standard Kalman filters where ξ_k denote the posterior mean vector and $\boldsymbol{P}_k$ is the posterior covariance matrix.

Actions: The second elements of a POMDP problem is called "actions" which includes all the controllable variables of the model. For our problem, it is assumed that the two variables of forward accelerations and bank angles of the UAVs can be controlled. Therefore, the action vector is represented by $u_k = \left(a_k, \phi_k\right)$ where a_k denotes the forward acceleration and ϕ_k denotes the bank angle.

Observation and Observation law: In a POMDP problem, the actions are taken according to an observation at each time step because the states are not fully observable. If χ_k^{pos} and s_k^{pos} are the position vectors of a fire front and a UAV respectively, the observation law of the fire fronts is defined as:

$$z_k^{\chi} = \begin{cases} \chi_k^{pos} + w_k & \text{if the fire front is visible} \\ \text{no measurement} & \text{otherwise} \end{cases} \tag{135}$$

w_k is a random measurement error with a distribution depending on the location of the sensor (UAV) and the target (fire front). In case of tracking fire fronts, the sensor states and tracker states are assumed to be fully observable.

State-Transition law: State-Transition laws are the functions which give the next states according to the current states being affected by taking actions at time k . Since there are three subsystems, three state-transition laws are required to describe how the system evolves over time. For the sensors, the state-transition law is represented by $s_{k+1} = \psi\left(s_k, u_k\right)$ where the function ψ will be defined later in section 5. State-Transition law for a fire front is generally defined as $\chi_{k+1} = f\left(\chi_k\right) + v_k$ where v_k is an independent and identically distributed (IID) random noise. In our study, it is assumed that the data of the fire fronts is given but we still need a model to run the Kalman filter at each time-step. Therefore, f is a linear momentary model of fire front that will be derived later in section 6. The tracker States-Transition laws are determined by standard Kalman filter equations but the only difference is when the fire fronts cannot be observed the update equation will not be performed and only the prediction step is taken.

Cost Function: As it was said in section 2, for solving the path-planning problem, the mean squared error between the tracks and the fire fronts must be minimized. Therefore, the cost function is defined as $C\left(x_k, u_k\right) = E_{v_k, w_{k+1}}\left[\left\|\chi_{k+1} - \xi_{k+1}\right\|^2 | x_k, u_k\right]$ at the time k.

Belief State: In POMDP problems, belief states are defined as probability distributions over the states. They are updated at each time step after taking actions according to the Bayes rule and by using the observations. At each time-step the belief state is defined as $b_k = \left(b_k^s, b_k^\chi, b_k^\xi, b_k^{\boldsymbol{P}}\right)$ and since all the states other than fire front states are assumed to be fully observable we have: $b_k^s = \delta\left(s - s_k\right)$, $b_k^\xi = \delta\left(\xi - \xi_k\right)$ and $b_k^{\boldsymbol{P}} = \delta\left(\boldsymbol{P} - \boldsymbol{P}_k\right)$. The belief state for the fire fronts will be derived in section 4.

Policy: A sequence of actions is called a policy. Obviously, a policy is considered to be optimal if it minimizes the expected cumulative cost $J_H = E\left[\sum_{k=0}^{H-1} C\left(x_k, u_k\right)\right]$ over a time horizon such as H ($k = 0,\ 1,\ ...,\ H-1$). It is assumed that the action taken at time k could depend on the history of the observations until time step $k-1$. If an optimal action exists, it can be shown that the optimal sequence of optimal actions also exists depending on “belief-state feedback” [89]. This means that the expected cumulative cost function can be rewritten as $J_H = E\left[\sum_{k=0}^{H-1} c\left(b_k, u_k\right) | b_0\right]$ where $c\left(b_k, u_k\right) = \int C\left(x, u_k\right) b_k(x) dx$. Given a belief state b_0, the optimal cost function is defined as $J_H^*(b_0) = \min_u \left\{c(b_0, u) + E\left[J_{H-1}^*(b_1) | b_0, u\right]\right\}$ where b_1 is the new random belief state distribution for the next time step and J_{H-1}^* is the optimal cumulative cost over the horizon $H-1$. $E\left[\bullet | b_0, u\right]$ shows that the expectation is given by taking action u when belief state distribution is b_0 [90]. The terms in the curly brackets are defined as the $Q-$values with initial belief state distribution of b_0 and taking action u (i.e. $Q_H(b_0,u) = c(b_0,u) + E\left[J^*_{H-1}(b_1)\middle|b_0,u\right]$), then the optimal policy at $k=0$ is written as $\pi^*_0(b_0) = \arg\min_u Q_H(b_0,u)$ and at time k, we can write $\pi^*_k(b_k) = \arg\min_u Q_{H-k}(b_k,u)$. In other words, the optimal policy is the sequence of actions, which minimizes the Q values at each time step. Calculating Q-values is not always easy since the term $E\left[J^*_{H-1}(b_1)\middle|b_0,u\right]$ is hard to be approximated. Therefore, approximation methods must be implemented to obtain Q-values. Here we use Nominal Belief-states Optimization (NBO) to specify the cost function. We will see that the method considers the trace of the error covariance matrix as an approximation of the cumulative cost function [74, 88]. The algorithm is explained in the next section.

4.4 NBO Approximation method

It is assumed that there are N_{ff} fire fronts and the states at time k is represented by $\chi_k = \left(\chi^1_k, \chi^2_k, \ldots, \chi^{N_{ff}}_k\right)$ where χ^i_k is the state of the ith fire front. The track state can be conveyed as $\xi_k = \left(\xi^1_k, \xi^2_k, \ldots, \xi^{N_{ff}}_k\right)$ and $\boldsymbol{P}_k = \left(\boldsymbol{P}^1_k, \boldsymbol{P}^2_k, \ldots, \boldsymbol{P}^{N_{ff}}_k\right)$, where $\left(\xi^i_k, \boldsymbol{P}^i_k\right)$ is the track state associated with the ith fire front. For modeling the fire front dynamics at each time-step, we define a linear motion model for the fire front with zero-mean noise with covariance $\boldsymbol{Q}_k$ as:

$$\chi^i_{k+1} = \boldsymbol{F}_k \chi^i_k + v^i_k \qquad v^i_k \sim N(0, \boldsymbol{Q}_k) \tag{136}$$

The observation (135) is defined as $z_k^{\chi^i} = \boldsymbol{H}_k \chi_k^i + w_k^i$ if the fire front is visible where $w_k^i \sim N(0, \boldsymbol{R}_k(\chi_k^i, s_k))$ while there would be no measurement when the fire front is not visible. The motion model describing how the fire fronts evolve over time is derived according to a complicated model, which we will not go through here and we assume that the data of the fire fronts is given. The only data required is the location of the active flame (border of the fire front). Once we have this data, we define the state of the front as $\chi_k^i = \left[x_k^i, y_k^i, 1\right]^T$ where (x_k^i, y_k^i) is the 2-D position of the ith fire front. In section 6, we explain why we should make an augmented state vector like this. For each time-step, we construct a linear model matrix $\boldsymbol{F}_k$, which gives the next position. Since the position of the fire fronts are the only observed parameters, the observation model $\boldsymbol{H}_k$ is defined as $\boldsymbol{H}_k = \left[\boldsymbol{I}_{2\times 2}, \boldsymbol{0}_{2\times 1}\right]$. It is assumed that all the distributions are Gaussian, and the belief state for the ith fire front is written as $b_k^{\chi^i}(\chi) = N\left(\chi - \xi_k^i, \boldsymbol{P}_k^i\right)$.

In Nominal Belief-states Optimization (NBO) approximation method, a nominal belief state sequence is introduced over a time horizon such as H (i.e $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_{H-1}$) and then the objective cost function is approximated by $J_H(b_0) \approx \sum_{k=0}^{H-1} c(\hat{b}_k, u_k)$. The optimization is accomplished through taking a sequence of actions $u_1, u_2, \ldots, u_{H-1}$. The nominal belief state for the ith fire front is obtained by the nominal tracks $\left(\hat{\xi}_k^i, \hat{\boldsymbol{P}}_k^i\right)$ as $\hat{b}_k^{\chi^i}(\chi) = N(\chi - \hat{\xi}_k^i, \hat{\boldsymbol{P}}_k^i)$. The nominal tracks evolve according to the Kalman filter equation as follows:

$$\hat{\xi}_{k+1}^i = \boldsymbol{F}_k \hat{\xi}_k^i \quad \text{(zero-noise)}$$

$$\hat{\boldsymbol{P}}_{k+1}^i = \begin{cases} \left[\left[\hat{\boldsymbol{P}}_{k+1|k}^i\right]^{-1} + \boldsymbol{S}_{k+1}^i\right]^{-1} & \text{if measurement available} \\ \hat{\boldsymbol{P}}_{k+1|k}^i & \text{otherwise} \end{cases} \tag{137}$$

where $\hat{\boldsymbol{P}}^i_{k+1|k} = \boldsymbol{F}_k \hat{\boldsymbol{P}}^i_k \boldsymbol{F}_k^T + \boldsymbol{Q}_k$, $\boldsymbol{S}^i_{k+1} = \boldsymbol{H}^T_{k+1} \left[\boldsymbol{R}_{k+1}(\hat{\xi}^i_{k+1}, s_{k+1}) \right]^{-1} \boldsymbol{H}_{k+1}$ and s_{k+1} is the state of the sensors at time $k+1$ which is given by the model $\psi(s_k, u_k)$. Since $\hat{\boldsymbol{P}}^i_{k+1}$ requires the observation at time $k+1$ while it is not certainly known, we use the position estimate of the fire fronts and the corresponding sensor state at time $k+1$ (i.e. $\hat{\xi}^{i,pos}_{k+1}$ and s^{pos}_{k+1}). Now the cost function is approximated as the sum of the trace of the nominal error covariance matrix for all the targets or:

$$c(\hat{b}_k, u_k) = \sum_{i=1}^{N_{ff}} \mathrm{Tr}(\hat{\boldsymbol{P}}^i_{k+1}) \tag{138}$$

Obviously, the objective cost function which needs to be minimized over a time horizon H can be written as:

$$J_H(b_0) = \sum_{k=0}^{H-1} \sum_{i=1}^{N_{ff}} \mathrm{Tr}(\hat{\boldsymbol{P}}^i_{k+1}) \tag{139}$$

in case of having multiple UAVs, the covariance error matrix for the ith fire front is computed as:

$$\hat{\boldsymbol{P}}^i_{k+1} = \left[\sum_{j=1}^{N_{sens}} (\hat{\boldsymbol{P}}^{i,j}_{k+1})^{-1} \right]^{-1} \tag{140}$$

where N_{sens} denotes the number of the UAVs (sensors) and $\hat{\boldsymbol{P}}^{i,j}_{k+1}$ represents the nominal error covariance matrix of the ith fire front calculated by the jth sensor.

The measurement error distribution depends on the location of the fire fronts and sensors and is according to the Gaussian distribution, which means $w_k \sim N\left(0, \boldsymbol{R}_k(\chi_k, s_k)\right)$. Here $\boldsymbol{R}_k$ shows the uncertainties in range and angle between the fire front and the sensor. We assume that the range uncertainty is $p\%$ and angular uncertainty is q . The distance between the fire front and sensor at time k is denoted by r_k . Therefore, the standard deviations for the angle and range are calculated as:

$$\begin{aligned}\sigma_{range}(k) &= (p/100)r_k \\ \sigma_{angle}(k) &= qr_k\end{aligned} \tag{141}$$

If the UAV flies exactly above the fire front (active flame), $r_k = 0$ which makes the information matrix get very big due to the fact that the information matrix depends on the inverse of the measurement covariance matrix. This problem is solved by considering a positive number b and defining the effective distance as $r_{eff}(k) = \sqrt{r_k^2 + b^2}$. Now the standard deviations can be redefined as:

$$\begin{aligned}\sigma_{range}(k) &= (p/100)r_{eff}(k) \\ \sigma_{angle}(k) &= qr_{eff}(k)\end{aligned} \tag{142}$$

If it is assumed that the angle between the fire front and the UAV at time k , (the angle of the connecting line and the horizontal axis) is θ_k we have:

$$\begin{gathered}\boldsymbol{M}_k = \begin{bmatrix} \cos(\theta_k) & -\sin(\theta_k) \\ \sin(\theta_k) & \cos(\theta_k) \end{bmatrix} \\ \text{and} \\ \boldsymbol{R}_k = \boldsymbol{M}_k \begin{bmatrix} \sigma^2_{range}(k) & 0 \\ 0 & \sigma^2_{angle}(k) \end{bmatrix} \boldsymbol{M}_k^T\end{gathered} \tag{143}$$

and the eigenvalues of $\boldsymbol{R}_k$ are $\left\{\sigma^2_{range}(k), \sigma^2_{angle}(k)\right\}$. In our algorithm we implement the "receding horizon approach" which means that we do the optimization for H time-steps but apply the optimal actions for the current time-step. Then we optimize the actions for another H time-steps and apply the optimal actions for the next time-step and this process continues. This is the "look-ahead" quality of our algorithm we have mentioned in the introduction.

4.5 State-transition law for the sensors

The sensor states evolve over time according to a model such as $s_{k+1} = \psi(s_k, u_k)$ where the state of each UAV includes the 2-D position coordinates, speed of the UAV and its heading angle. We write the state of the jth sensor at time k as $s_k^j = \left(p_k^j, q_k^j, V_k^j, \theta_k^j\right)$ where (p_k^j, q_k^j) are the position coordinates, V_k^j denotes the speed of the UAV and θ_k^j is the heading angle. The actions are shown as the vector of forward acceleration and bank angle $u_k^j = \left(a_k^j, \phi_k^j\right)$. The relationship between the sensor state and the actions defines the mapping function ψ as follows:

$$\begin{aligned}
&V_{k+1}^j = \left[V_k^j + a_k^j T\right]_{V_{\min}}^{V_{\max}} \quad \text{where } \left[v\right]_{V_{\min}}^{V_{\max}} = \max\left\{V_{\min}, \min(V_{\max}, v)\right\} \\
&\theta_{k+1}^j = \theta_k^j + (gT\tan(\phi_k^j)/V_k^j) \\
&p_{k+1}^j = p_k^j + V_k^j T\cos(\theta_k^j) \\
&q_{k+1}^j = q_k^j + V_k^j T\sin(\theta_k^j)
\end{aligned} \tag{144}$$

where $V_{\max}$ and $V_{\min}$ shows the maximum and minimum speed limits for the UAVs. According to [91], the limits for the control variables is set to be as follows:

$$\begin{gathered}
-3.05\ \text{m/s}^2 \le a \le 3.05\ \text{m/s}^2 \\
-\frac{\pi}{6} \le \phi \le \frac{\pi}{6}
\end{gathered} \tag{145}$$

4.6 Modeling Fire Fronts

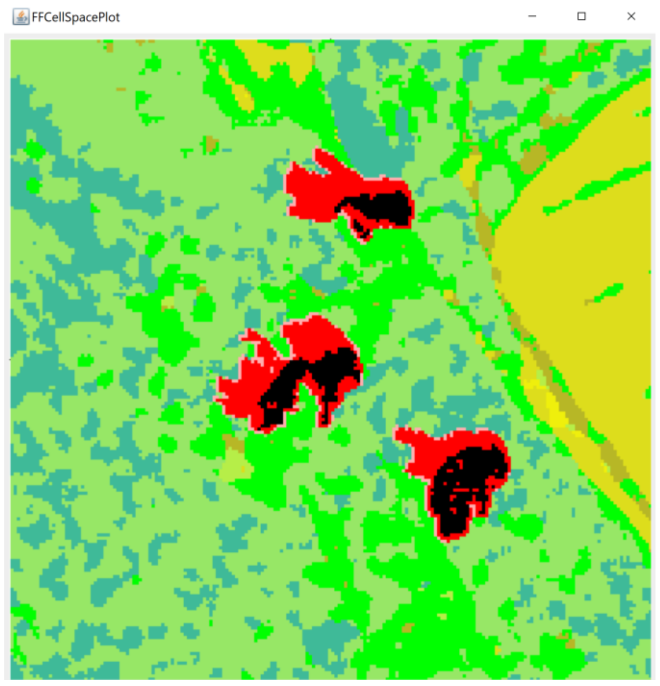

Figure 17 (a).Three fires at an early stage of fire spread

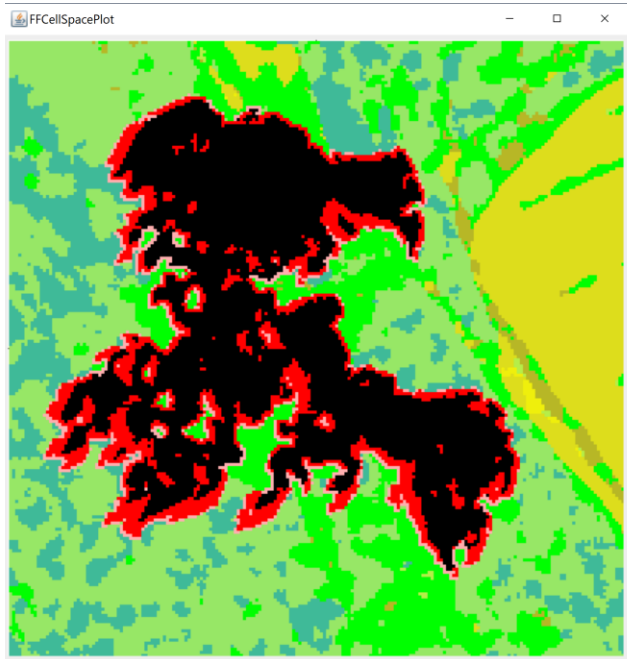

Figure 17 (b). The fire at the end of the simulation

The fire spread simulation was based on the DEVS-FIRE model [92, 93] which is a discrete event simulation model for surface fire spread simulation. DEVS-FIRE uses a cellular space to represent a wildland area, where each cell has its own terrain data and fuel (vegetation) data corresponding to the sub-regions in the area. All cells are coupled to a weather model to receive weather data (wind speed and wind direction) over time. Once a cell is ignited, it uses Rothermel's model [94] to compute the fire spread rate and direction within the cell. Fire spreading is modeled as a propagation process as burning cells ignite their unburned neighboring cells.

In this simulation, three fires are ignited at three different locations of a 200 by 200 cell space. Each cell represents a 30-meter by 30-meter area. The simulation is for 5 hours of fire spread. Figure 17(a) shows the three fires at an early stage of fire spread, where red cells are burning cells, black cells are burned cells, and other colors represent the different fuel types of the cells. Figure 17(b) shows the fire shape at the end of 5 hours of spread, when the three fires are combined into a larger fire with a perimeter of 34.11km and burned area of 1113.21 hectares.

During the simulation, we recorded the fire perimeter cells and burning & burned cells every 10 seconds. These data are then read and displayed by a program to visualize the fire spread process.

4.7 Main results

In this section the results for the path-planning problem will be provided. It is assumed that there are three separate fire fronts, which start from various locations and evolve over time to make a single unified big fire. The fire fronts are modeled for five hours and we investigate three different scenarios of tracking. In the first scenario, one UAV tracks the three fire fronts. We will see that the UAV must shifts from one front to another to keep the trace of the covariance error matrix minimized. Since the final fire front covers a large area, one UAV may seem insufficient to be able to do a proper coverage. In the second scenario, two UAVs track the three fire fronts. In this scenario the UAVs autonomously decide where to track so the coverage of all the three fronts is done the best. In the third scenario, three UAVs starts tracking the fire fronts but one of them drops off in the middle of its mission (this could occur due to an accident or battery discharge) and the two other UAVs try to make up for the dropped one and cover the fronts as good as they can. In each scenario, we let the fire fronts spread for a while and then the UAVs are assigned to start tracking them. The results for each scenario is shown by a few pictures to show the performance of the UAVs as time elapses.

Scenario 1: One UAV

The fire front data is supposed to have been calculated and given to us. We need the border points of the fire which specify the location of the active flame. In order for our algorithm to be working, a momentarily motion model for each time-step is generated regarding the two position of the fire front at time k . Sensors see two points of the fire as the positions for the current and the next time-step (i.e $\left(x_k^i, y_k^i\right)$ and $\left(x_{k+1}^i, y_{k+1}^i\right)$ respectively). A linear motion model matrix for each time-step $\boldsymbol{F}_k$ gives us the next state at time k as follows:

$$\boldsymbol{F}_k^i = \begin{bmatrix} 1 & 0 & \Delta x_k^i \\ 0 & 1 & \Delta y_k^i \\ 0 & 0 & 1 \end{bmatrix} \tag{146}$$

where $\Delta x_k^i = x_{k+1}^i - x_k^i$ and $\Delta y_k^i = y_{k+1}^i - y_k^i$ are obtained from the 2-D position coordinates for current and the next time-step. For each time-step the state is defined as $\chi_k^i = \left[x_k^i, y_k^i, 1\right]^T$ to make sure (146) can give us the next state. The tracking results are shown in Figures 18 (a)-(f). The UAV starts from a point far away from the region and immediately approaches the three evolving fire front. As time passes, the UAV flies over the three and tries to cover all in order to minimize the cost function. However, as fire fronts become bigger, one UAV would not be able to cover all the perimeter of the fire completely and some parts are skipped in trying to keep the tracking error minimized. Figures 19 (a)-(b) show the control variables (forward acceleration and bank angle) of the UAV at each 600 s for the whole period of 18000 s. It is seen that these variables remain within the limits of (145). The initial state of the UAV is assumed to be $s_0 = [3000\text{ m} \quad 0\text{ m} \quad 16\text{ m/s} \quad \pi/6\text{ rad}]^T$ and the initial action vector is $u_0 = \left[2\text{ m/s}^2 \quad \pi/12\text{ rad}\right]^T$

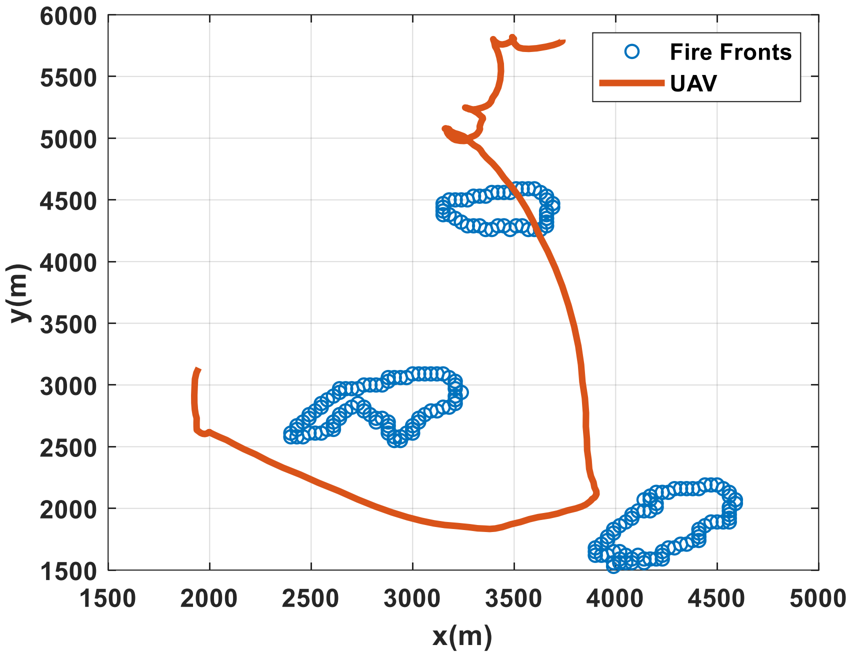

Figure 18 (a). The first scenario (t = 3000s)

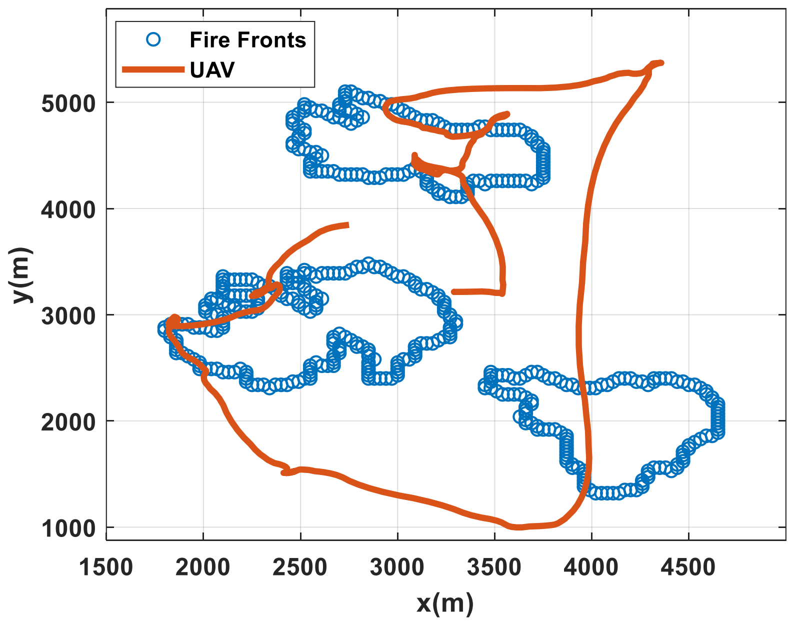

Figure 18 (b). The first scenario (t = 6000s)

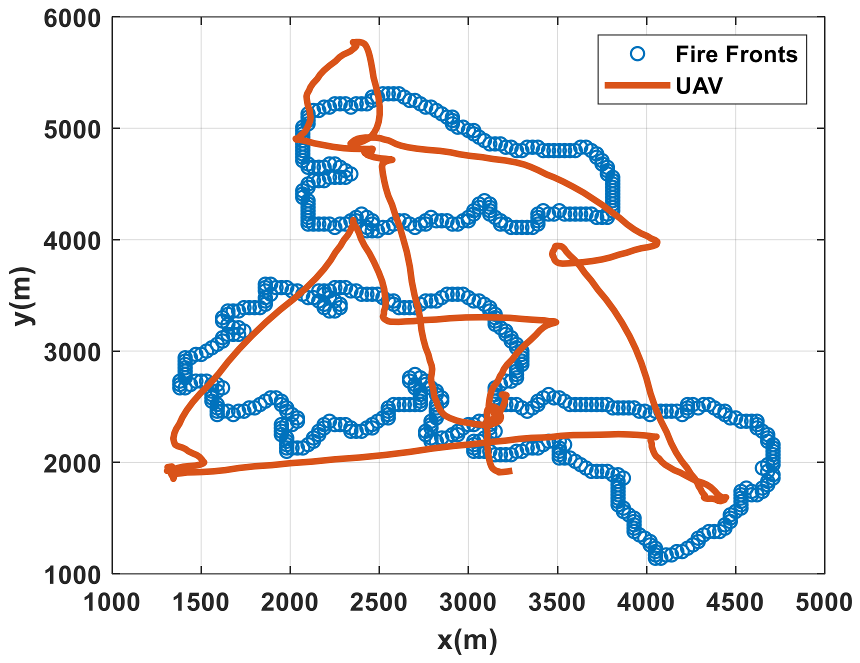

Figure 18 (c). The first scenario (t = 9000s)

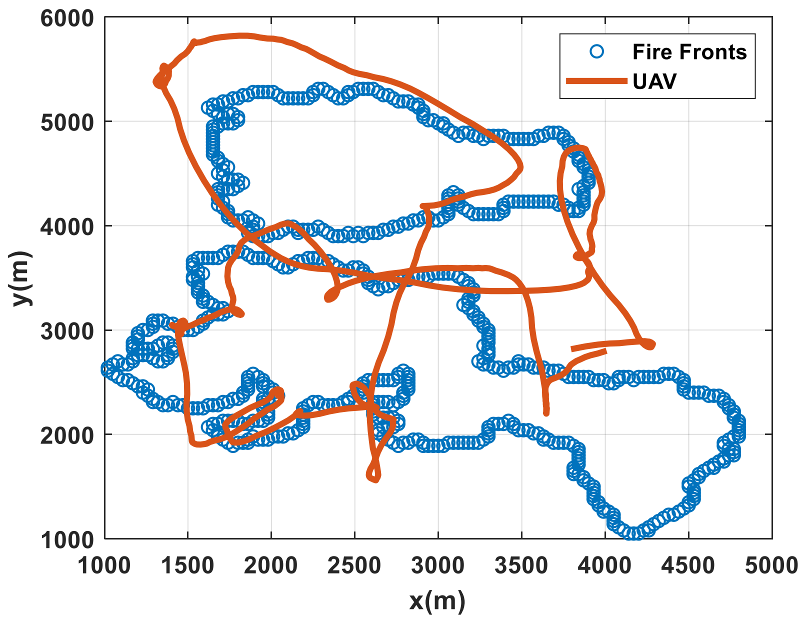

Figure 18(d). The first scenario (t = 12000s)

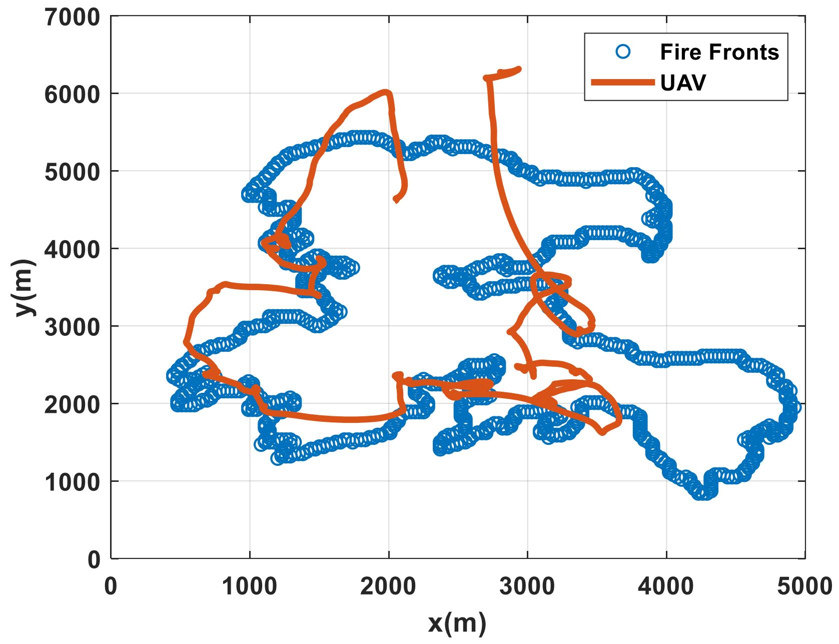

Figure 18 (e). The first scenario (t = 15000s)

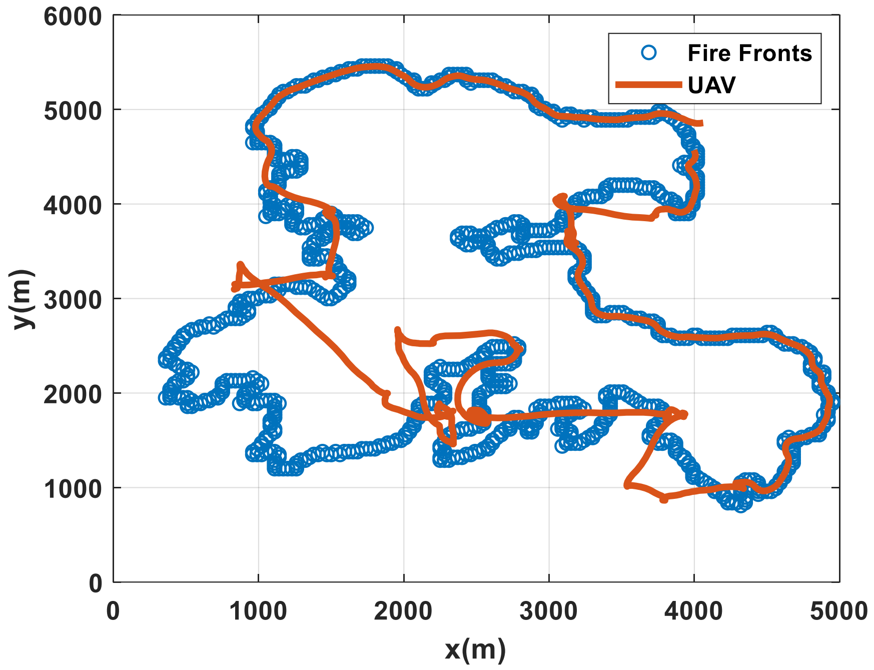

Figure 18 (f). The first scenario (t = 18000s)

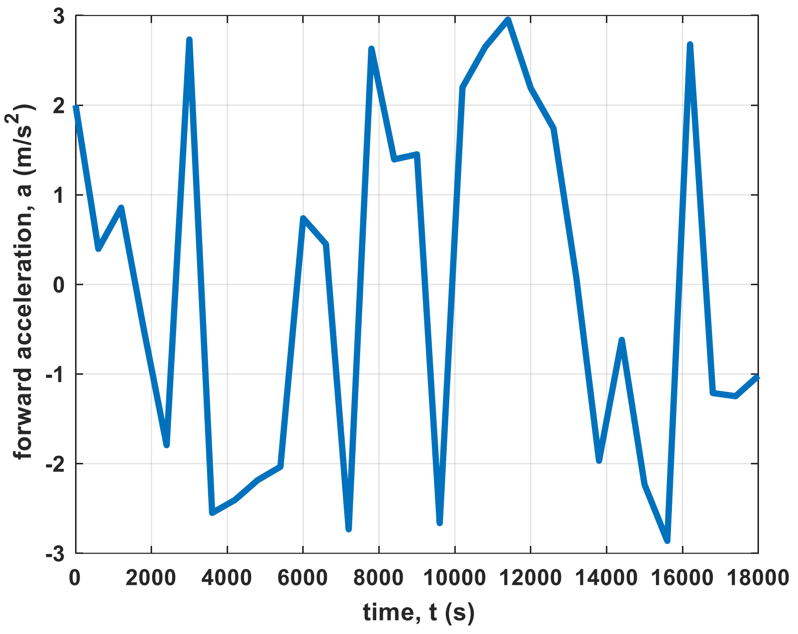

Figure 19 (a). Forward acceleration of the UAV

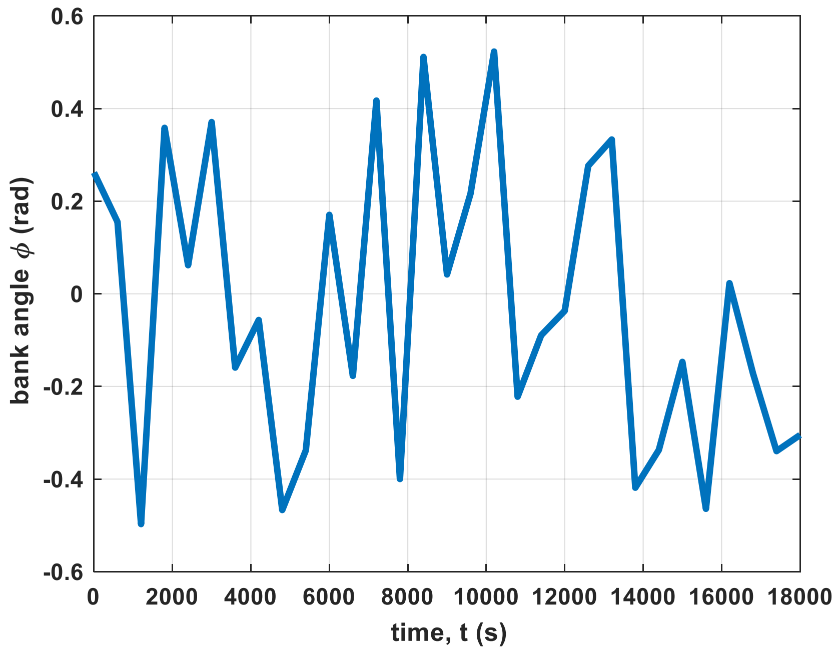

Figure 19 (b). Bank angle of the UAV

A way to check the validity of results could be making a comparison between true error and estimate error of the results. We define the true error as the Euclidean norm of the difference between fire front position and position of the nominal estimate (i.e. $\left\| \chi_{k+1}^{pos} - \hat{\xi}_{k+1}^{pos} \right\|$ where χ_{k+1}^{pos} is the position coordinates of the fire fronts and ξ_{k+1}^{pos} is the position coordinate in the nominal track estimate vector). The estimate error on the other hand is defined as $\sqrt{\hat{P}_{11} + \hat{P}_{22}}$ where $\hat{P}_{11}$ and $\hat{P}_{22}$ are the diagonal elements of the nominal error covariance matrix. Figure 20 shows these two errors during the process of tracking the fully developed fire ($t = 18000$ s) and it is seen that they both have the same pattern indicating that the algorithm works properly.

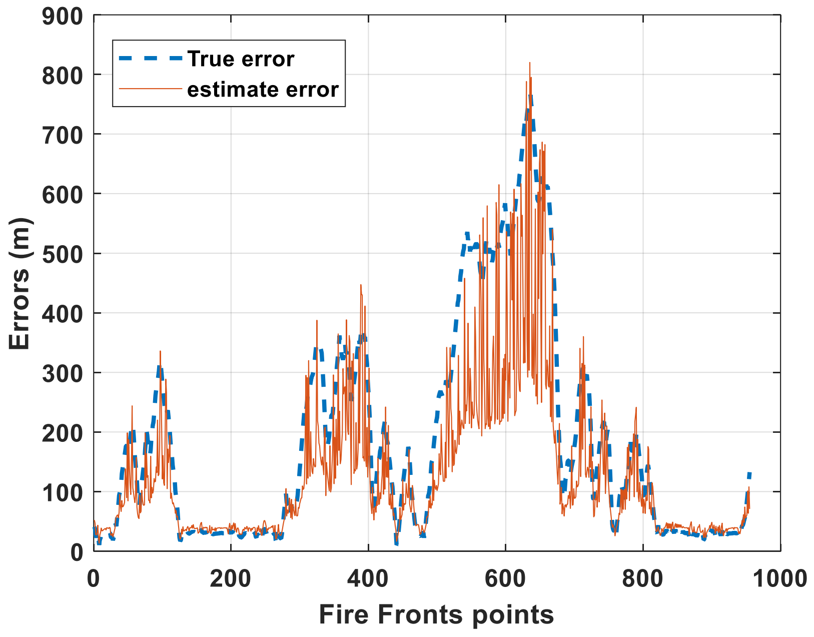

Figure 20. True and estimate error of the algorithm

Scenario 2: Two UAVs

In case of multiple UAVs, collision avoidance must be taken into consideration. A penalty function is introduced and added to the objective cost function in order to avoid collision among the UAVs. We show this penalty function by $\boldsymbol{P}_{k+1}^{coll,j}$ for the jth UAV and redefine the cost function to be:

$$J_H = \sum_{k=0}^{H-1}\left(\sum_{i=1}^{N_{ff}} \mathrm{Tr}\left(\hat{\boldsymbol{P}}_{k+1}^{i}\right) + \gamma \sum_{j=1}^{N_{sens}} \boldsymbol{P}_{k+1}^{coll,j}\right) \tag{147}$$

where γ is a positive scaling factor which must be chosen big enough to make the second term effective when the UAVs get close to each other. The penalty function is defined as follows. A constant distance such as D is chosen as a safe distance and the penalty function is calculated according to it as follows:

$$\boldsymbol{P}_{k+1}^{coll,j} = \begin{cases} D - d_{k+1}^{j} & \text{if } d_{k+1}^{j} < D \\ 0 & \text{otherwise} \end{cases} \tag{148}$$

where d_{k+1}^{j} is defined as $\min_{i,j\neq i} d_{k+1}^{ji}$ where d_{k+1}^{ji} denotes the distance between the ith and jth UAVs. Since the UAVs are assumed to be small D can be selected to be as small as 5 m and the scaling factor is set to be $\gamma = 10$. The advantage of this approach is that it considers the collision avoidance in POMDP context and does not need a separate algorithm to work. The results are shown in figures 21 (a)-(g). It is seen that at the beginning of the mission both UAVs may try to cover all the three fire fronts and switch between one to another but as time elapses they decide which part is the best for them to cover in order to minimize the trace of the covariance error matrix. This may happen due to the fact that the accuracies of the estimates may be lower when the burning regions are small but once the fire becomes large enough the UAVs make clear decisions to keep the distance and also simultaneously do the coverage the best they can do. It is assumed that the initial states for the two UAVs are $s_0^1 = \begin{bmatrix} 3600\text{ m} & 3300\text{ m} & 16\text{ m/s} & \pi/6\text{ rad} \end{bmatrix}^T$ and $s_0^2 = \begin{bmatrix} 4200\text{ m} & 4200\text{ m} & 16\text{ m/s} & \pi/6\text{ rad} \end{bmatrix}^T$. The initial action vectors are: $u_0^1 = \begin{bmatrix} 2\text{ m/s}^2 & \pi/12\text{ rad} \end{bmatrix}^T$ and $u_0^2 = \begin{bmatrix} 2\text{ m/s}^2 & \pi/12\text{ rad} \end{bmatrix}^T$.

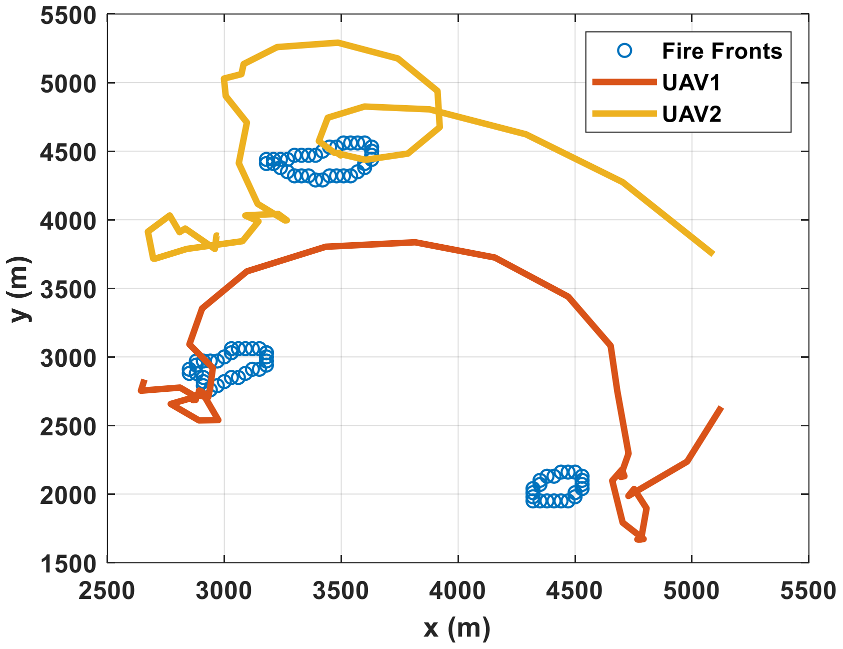

Figure 21(a). The second scenario (t = 1500s)

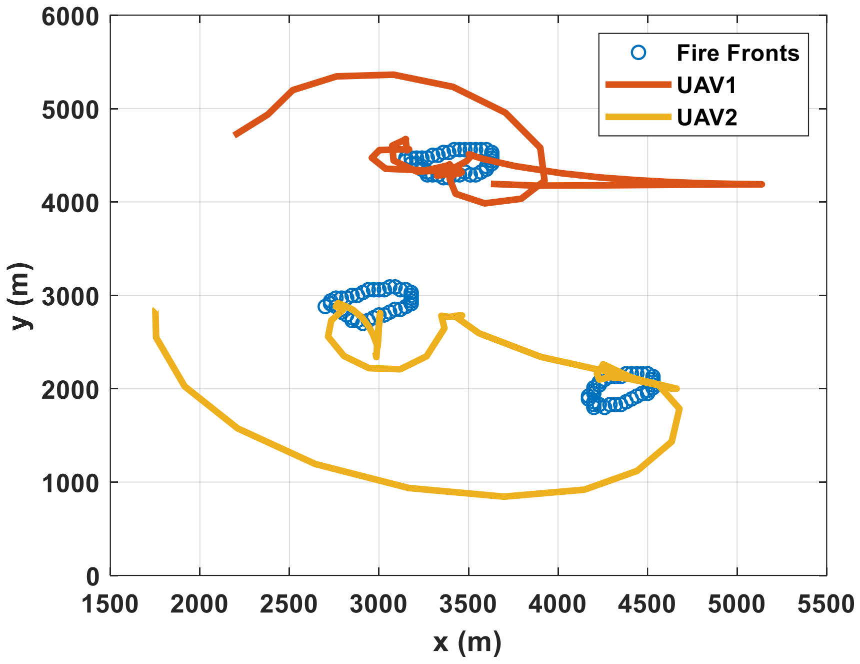

Figure 21 (b). The second scenario (t = 3000s)

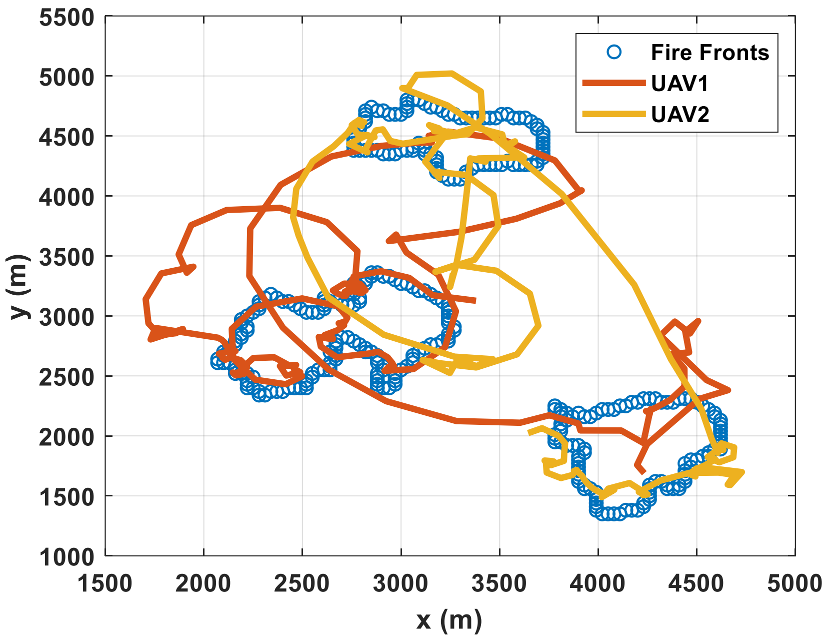

Figure 21 (c). The second scenario (t = 6000s)

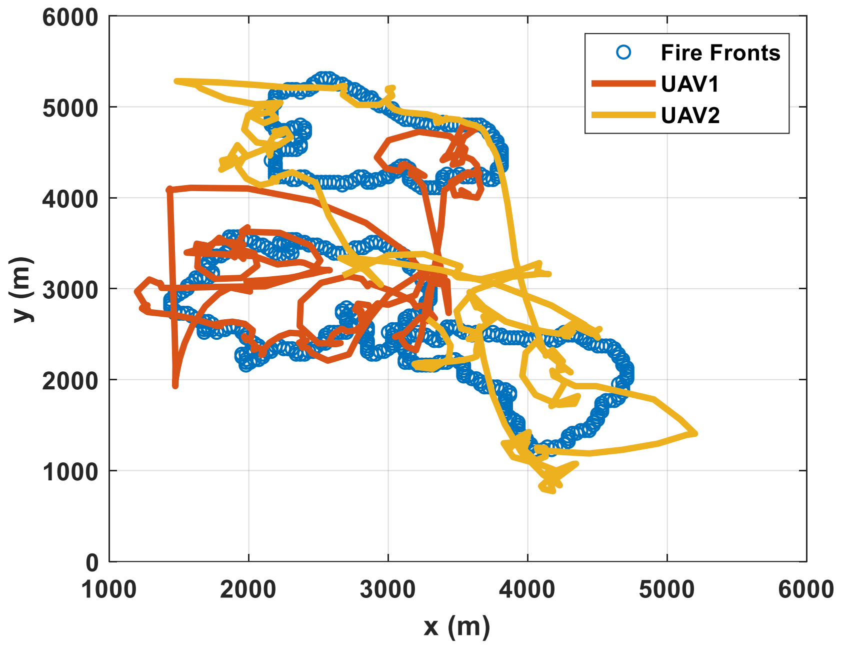

Figure 21 (d). The second scenario (t = 9000s)

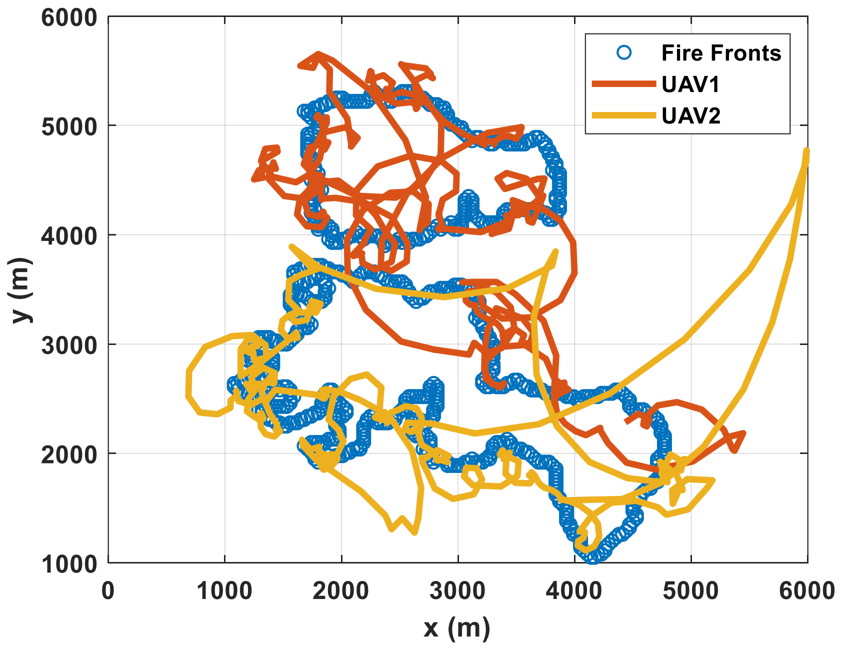

Figure 21 (e). The second scenario (t = 12000s)

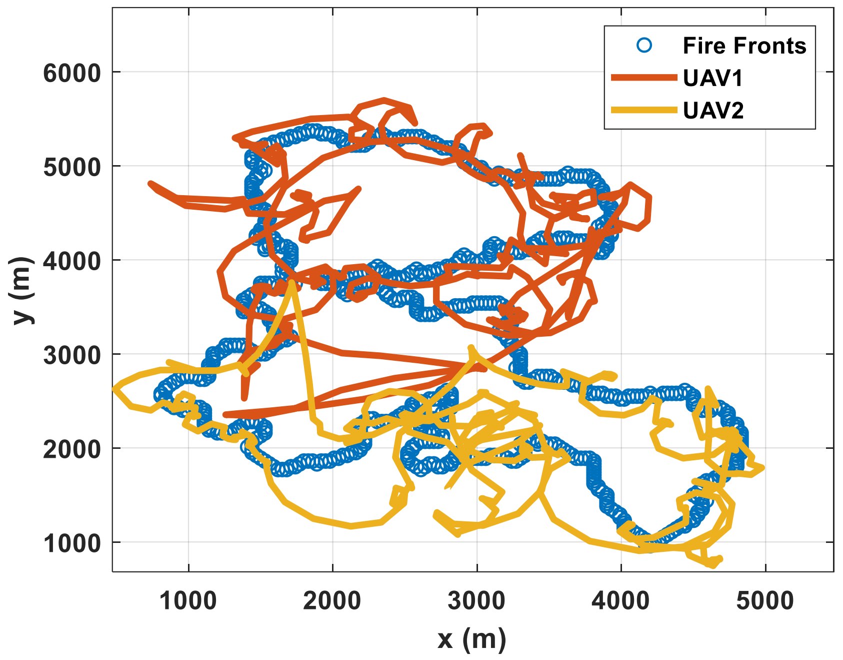

Figure 21 (f). The second scenario (t = 15000s)

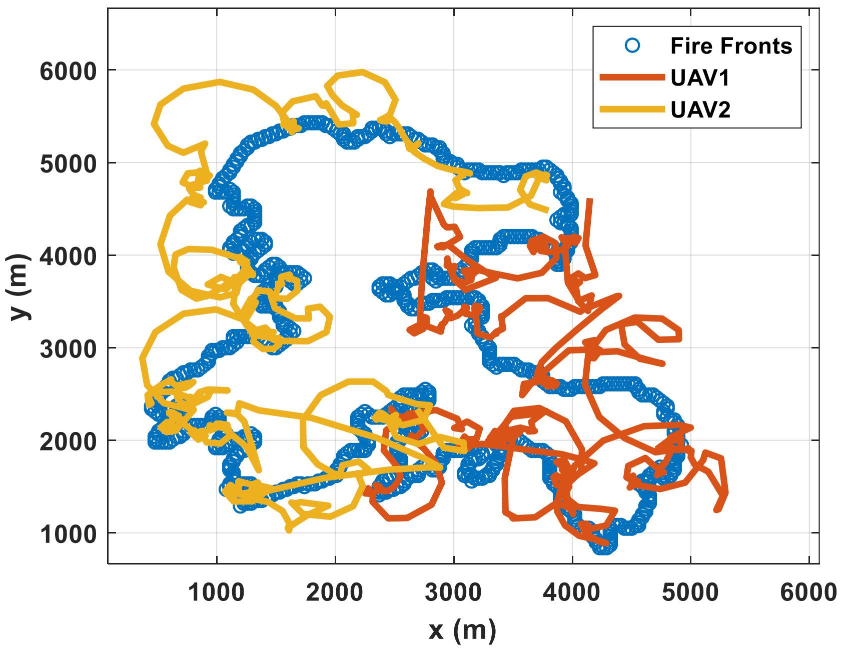

Figure 21 (g). The second scenario (t = 18000s)

Scenario 3: Three UAVs start tracking the fire fronts but one of them drops off at some point

Many incidents may cause the UAVs to stop tracking the targets among which battery discharge could be the most important one. Once the battery of a UAV is discharged, it has to fly back and descend on the ground to get it recharged. Meanwhile, the remainder of the UAVs must make up for the lost one in order to keep the coverage reliable or mathematically said keep the sum of the traces of the error covariance matrices minimized. The three UAVs cover the fronts and the third UAV stops tracking at time $t=10500 \text{ s}$ and after that, only UAVs 1 and 2 continue to cover the fire fronts. Results are shown in Figures 22(a)-(g). It is seen that after the sudden change of reducing the number of trackers to two, the UAVs experience a period of confusion and the coverage is done less perfectly compared to the time when all the UAVs were working (e.g. $t=12000 \text{ s}$) . However, as time elapses, the accuracy of the estimation steps rise up and the UAVs decide how to track the fronts such that the coverage becomes the best. We still use the same penalty function as (147) and (148) in order to avoid collision among the UAVs. The initial vectors for the UAVs are: $s_0^1=\left[3900 \text{ m} \quad 3900 \text{ m} \quad 16 \text{ m/s} \quad \pi/6 \text{ rad}\right]^T$, $s_0^2=\left[4800 \text{ m} \quad 2400 \text{ m} \quad 16 \text{ m/s} \quad \pi/6 \text{ rad}\right]^T$, $s_0^3=\left[4800 \text{ m} \quad 4500 \text{ m} \quad 16 \text{ m/s} \quad \pi/6 \text{ rad}\right]^T$ and the initial action vector for all the UAVs are the same and equal to $u_0=\left[2 \text{ m/s}^2 \quad \pi/12 \text{ rad}\right]^T$

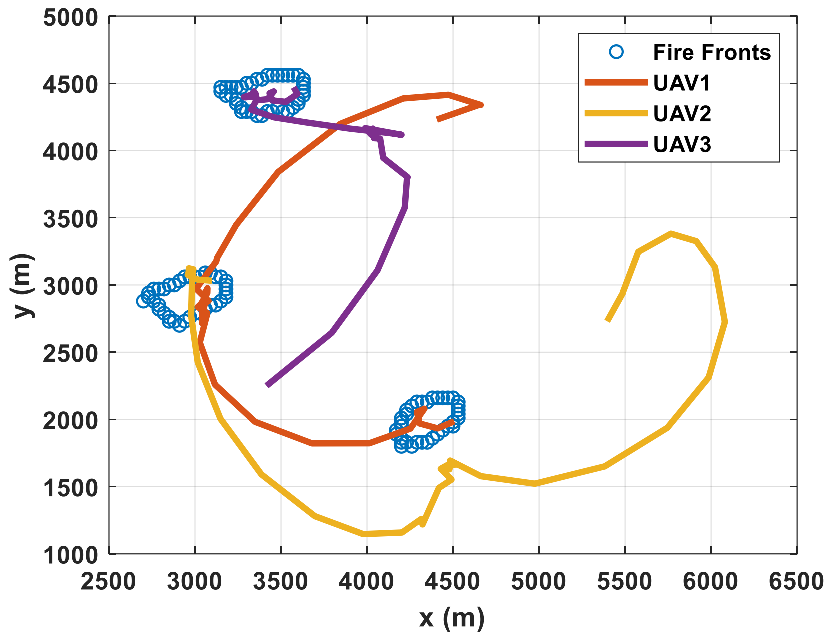

Figure 22 (a). The Third scenario (t = 3000s)

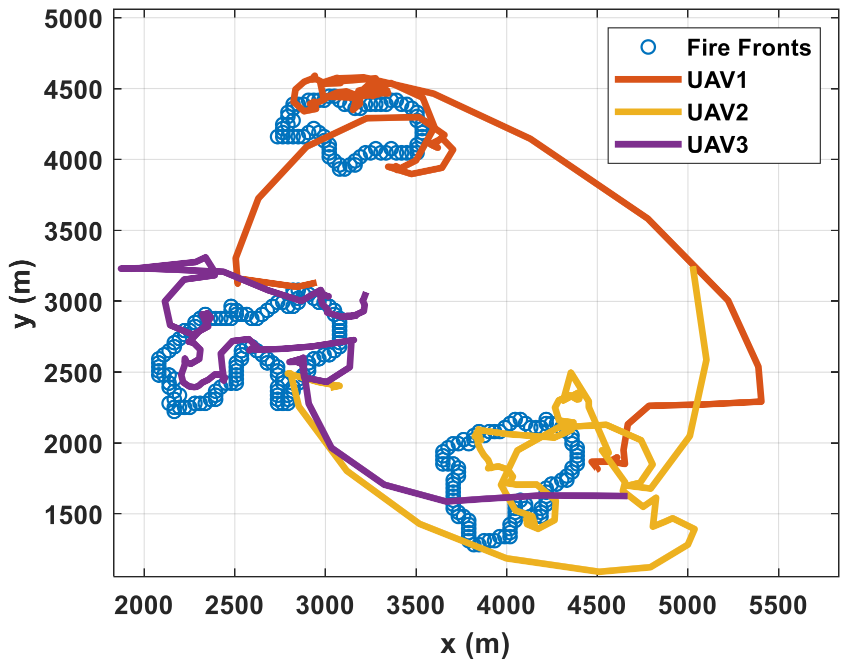

Figure 22 (b). The Third scenario (t = 6000s)

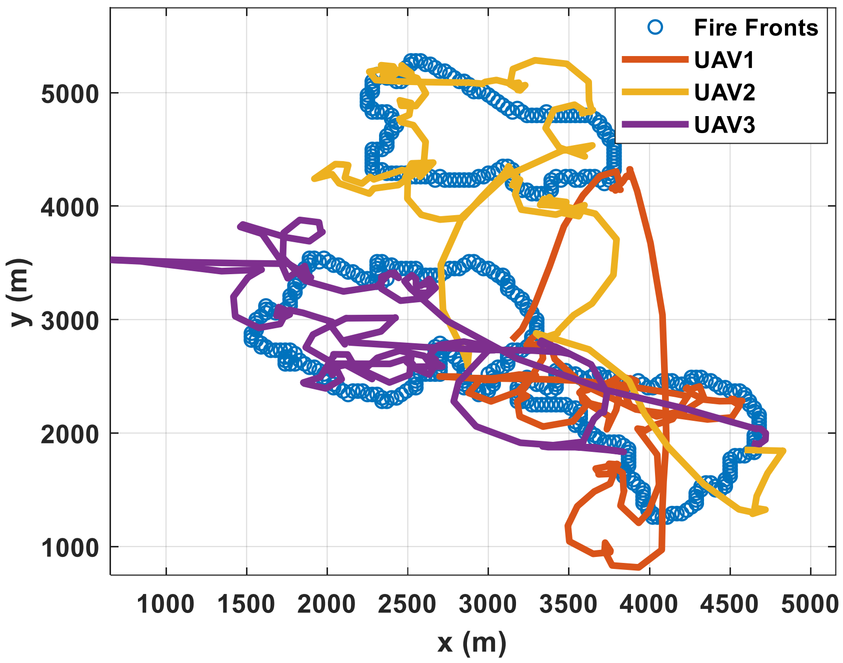

Figure 22 (c). The Third scenario (t = 9000s)

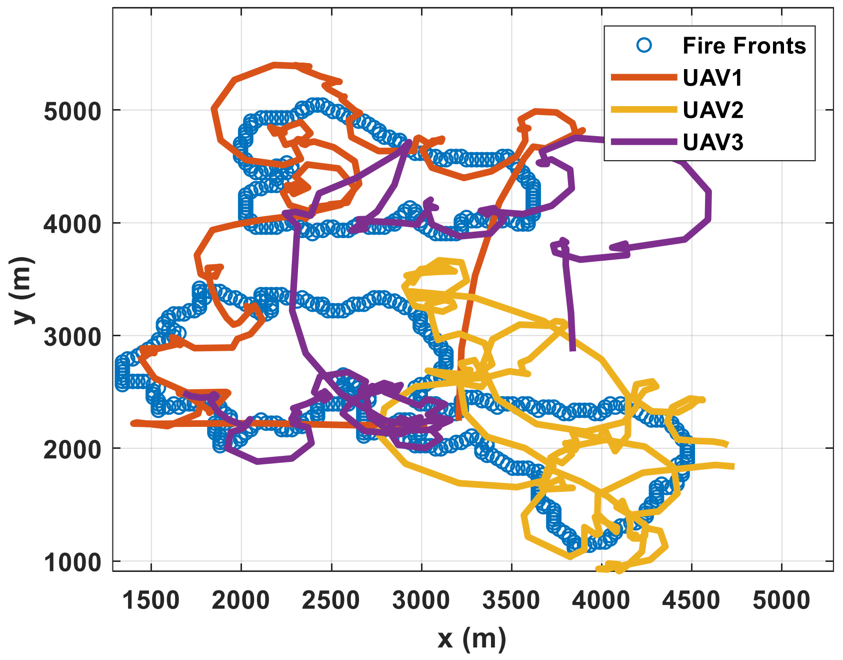

Figure 22 (d). The Third scenario (t = 10500s)

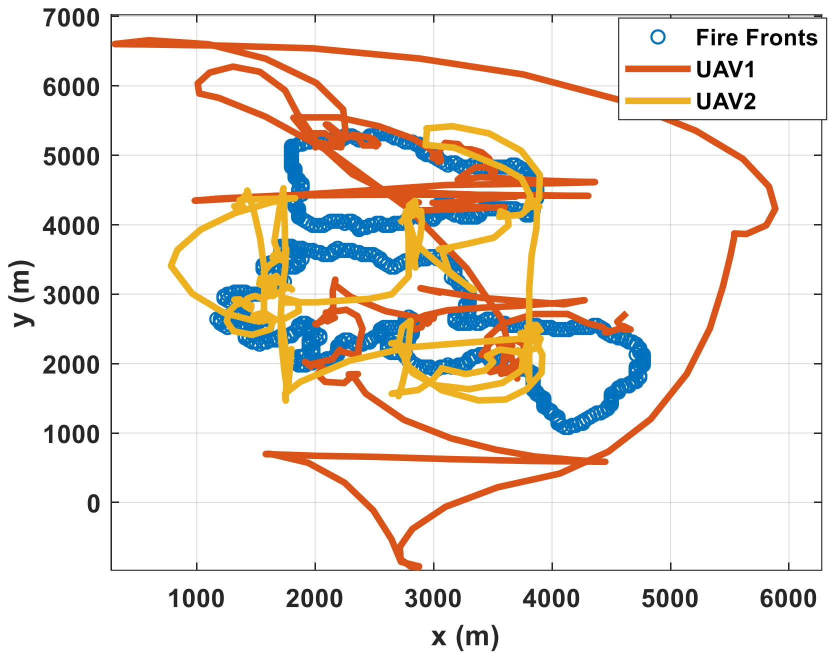

Figure 22 (e). The Third scenario (t = 12000s)

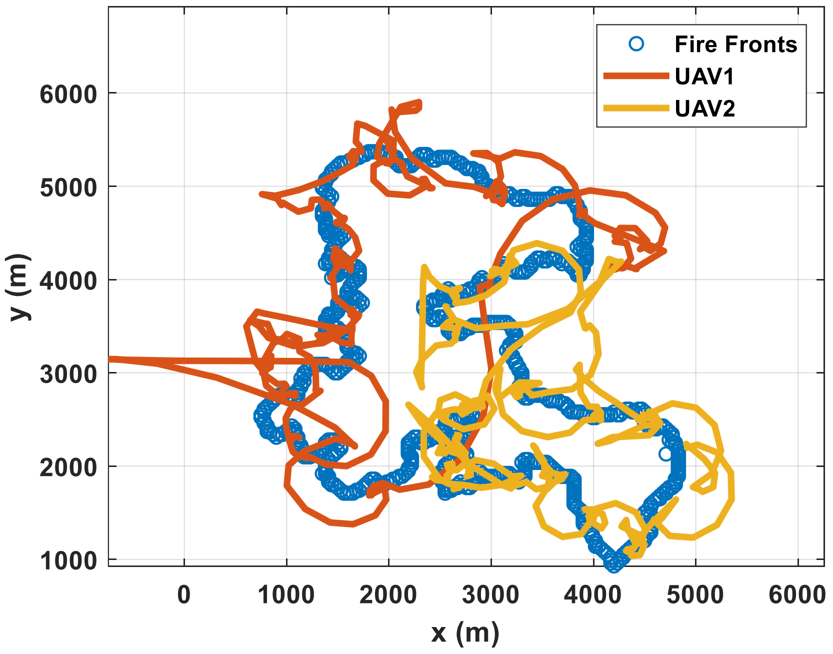

Figure 22 (f). The Third scenario (t = 15000s)

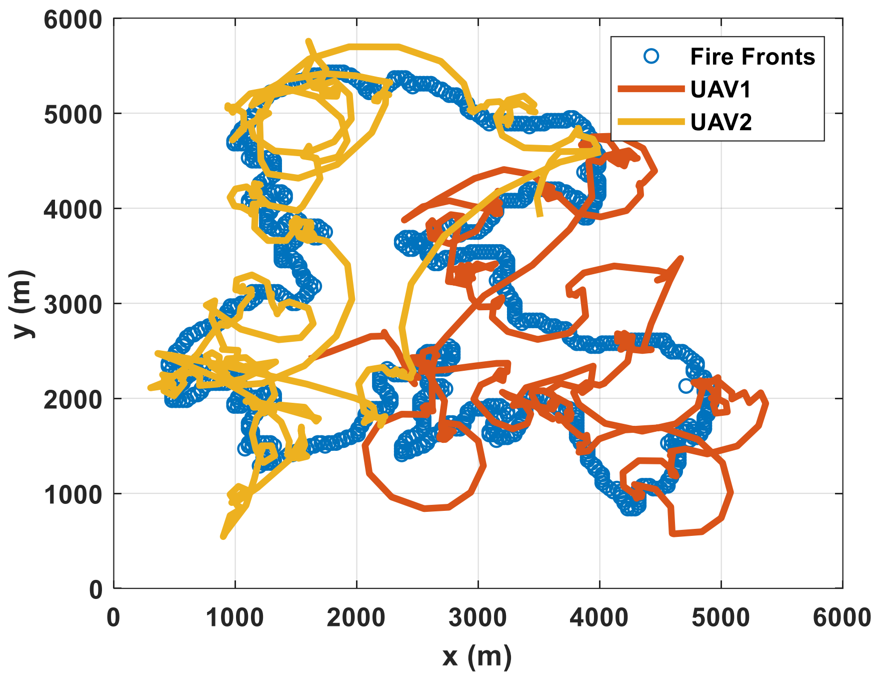

Figure 22 (g). The Third scenario (t = 18000s)

4.8 Robustness of the Algorithm

It practical application there would always be some disturbances and it needs for the algorithm to be able to overcome the unprecedented situations in order to be applicable. Here we check robustness of the designed algorithm by testing the performance of the UAV in the presence of a sudden wind gust. It is modeled by adding a certain acceleration in the sensor models while the rest of the state-transition functions are kept untouched. A good algorithm must be able to get to its normal condition of performance after being disturbed. We let the acceleration grow due to the wind be 50% of the maximum acceleration the UAV can achieve. In our case, this value would be equal to $a_d = a_{\max}/2 = 1.525\ \text{m/s}^2$ where a_d denotes the disturbance acceleration induced by the wind gust and $a_{\max}$ is the maximum acceleration of the UAV. We only consider the first scenario for the sake of clearness and simplicity and investigate two scenarios of wind disturbance. In both one all the other conditions are assumed to be the same as what we had in scenario one in section 4.7. In the first case, we let the UAV start tracking the fire fronts when the fire is fully developed ($t = 18000\text{ s}$) and assume that the wind starts blowing when the UAV is at its 100th time step ($k = 100\text{ s}$) when it is at $[3068\text{ m}, 3993\text{ m}]$. The blow lasts for 100 seconds and vanishes. Figures 23 (a)-(c) show the response of the UAV. For better comparison I have put the undisturbed situation beside it. It seems that at first the UAV gets deviated from its original track and after the wind stops, the UAV still needs some time to get back to its normal way but finally it manages to find its way back and continue tracking the fire fronts. In the second case, the fire starts at the time-step $k = 750\text{ s}$ at lasts for 100 seconds. Again, we see that the UAV gets disturbed any finally manages to find it way to accomplish its mission of tracking which means that the algorithm is reliable enough to be used in practical circumstances where several disturbance factors exist.

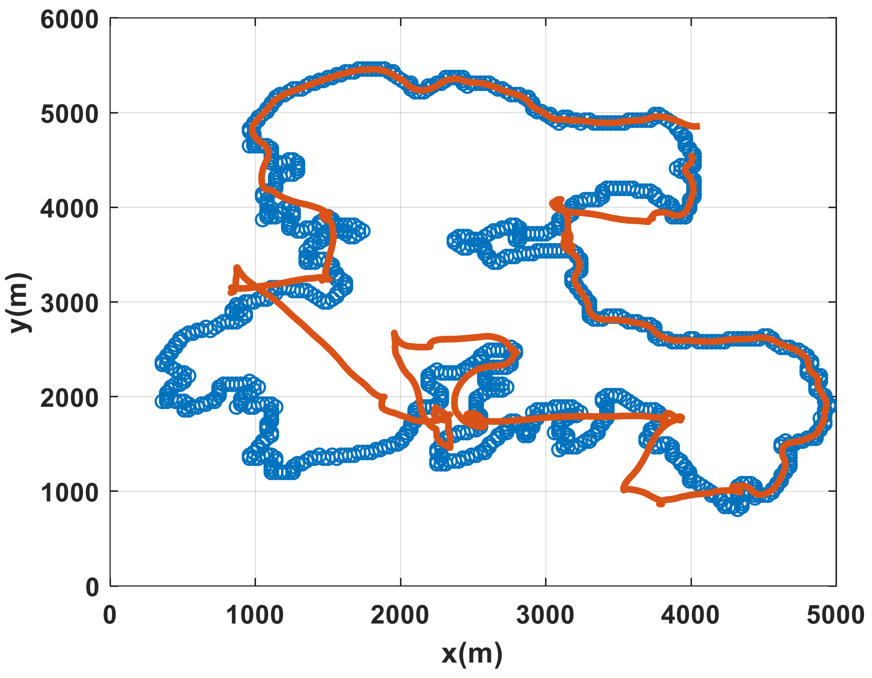

Figure 23 (a) undisturbed tracking

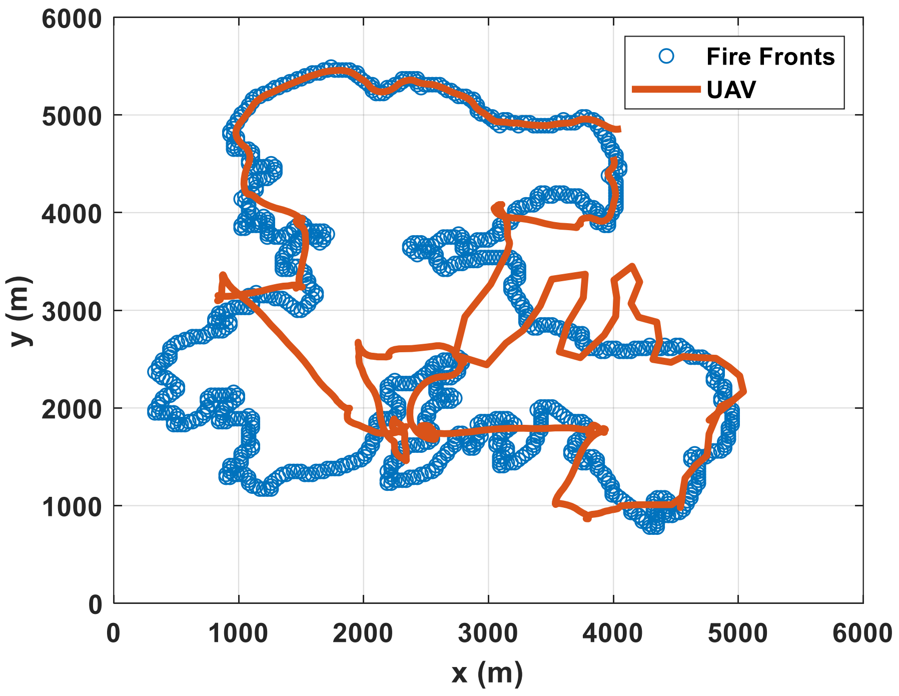

Figure 23 (b) The wind begins at k=100 s and ends at k=200 s

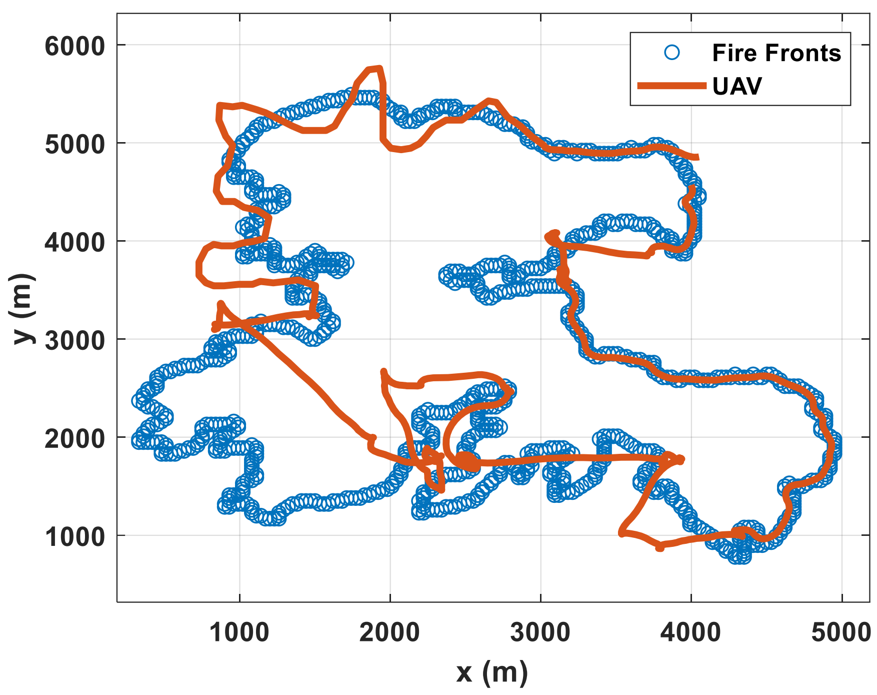

Figure 23 (c) The wind begins at k=750 s and ends at k=850 s

CONCLUSION

In this study, new optimal control approaches have been developed for solving consensus and path planning problems among multi-agent systems (MASs). It was seen that by defining proper cost functions and proper algorithms in order to minimize these cost functions, optimal control would be a very powerful tool in solving these two problems.

In chapters 2 and 3 the multi-agent consensus tracking problem for a class of general linear time-invariant systems was investigated in an optimal control framework. An inverse optimal control approach was employed to derive a proper cost function in order for the system to track a defined desired trajectory. The optimal control law was designed with an analytical solution and was a linear function of the Laplacian matrix such that the control implementation was distributed in that it only needed local information of the agent's own state and its neighbors with the communication links. Both optimality and stability of the control law are proved. The approach was then generalized to embrace general multi-input Linear Time-Invariant systems. For each type (single-input and multi-input systems), two examples were designed and solved to illustrate the algorithm.

In chapter 4, an algorithm was designed in order to solve the path-planning problem of a group of UAVs that are assigned to track the wildfire fronts. The approach was developed based on the theory of Partially Observable Markov Decision Process (POMDP) and it was seen that all important features such as collision avoidance and dynamic constraints can be considered in the context of POMDP. The approach has a "look-ahead" property in a sense that it calculates the control variables for each time step based on computation over a certain time horizon. This is done for more accuracy since the UAVs are supposed to track randomly evolving fire fronts. Dynamic constraints for the motion of the UAVs were taken into account and by using NBO approach, the cost function was defined according to tracking error. For each time-step a simple linear motion model for the fire fronts was defined which enabled the track state to have been constructed fast and efficiently. Three different scenarios were investigated and it was seen that the UAVs were able to make decisions about which fronts to track if they were given enough time. Furthermore, if a UAV, for any reasons, drops or stops working in the middle of the mission, the remainder of the UAVs keep tracking the fire fronts the way they make up for the lost one and keep the coverage as good as they can to maintain the defined cost function minimized.

REFERENCES:

[1] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 1859-1864.

[2] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics,* vol. 9, pp. 427-438, 2013.

[3] W. Ren and R. W. Beard, *Distributed consensus in multi-vehicle cooperative control*: Springer, 2008.

[4] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE,* vol. 95, pp. 215-233, 2007.

[5] W. Ren and Y. Cao, *Distributed coordination of multi-agent networks: emergent problems, models, and issues*: Springer Science & Business Media, 2010.

[6] W. Ren, K. Moore, and Y. Chen, "High-order consensus algorithms in cooperative vehicle systems," in *Networking, Sensing and Control, 2006. ICNSC'06. Proceedings of the 2006 IEEE International Conference on*, 2006, pp. 457-462.

[7] W. Yu, G. Chen, W. Ren, J. Kurths, and W. X. Zheng, "Distributed higher order consensus protocols in multiagent dynamical systems," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 58, pp. 1924-1932, 2011.

[8] F. Mirali, A. M. Gonzalez, and H. Werner, "First-Order Average Consensus for Cooperative Control Problems Using Novel Weighting Strategies," *IFAC-PapersOnLine,* vol. 50, pp. 14302-14307, 2017.

[9] J. Ni, L. Liu, C. Liu, X. Hu, and S. Li, "Further Improvement of Fixed-Time Protocol for Average Consensus of Multi-Agent Systems," *IFAC-PapersOnLine,* vol. 50, pp. 2523-2529, 2017.

[10] Y. Cao, D. Stuart, W. Ren, and Z. Meng, "Distributed containment control for multiple autonomous vehicles with double-integrator dynamics: algorithms and experiments," *IEEE Transactions on Control Systems Technology,* vol. 19, pp. 929-938, 2011.

[11] Y. Cao and W. Ren, "Distributed coordinated tracking with reduced interaction via a variable structure approach," *IEEE Transactions on Automatic Control,* vol. 57, pp. 33-48, 2012.

[12] Y. Cao, W. Ren, and Z. Meng, "Decentralized finite-time sliding mode estimators and their applications in decentralized finite-time formation tracking," *Systems & Control Letters,* vol. 59, pp. 522-529, 2010.

[13] Y. Hatano and M. Mesbahi, "Agreement over random networks," *IEEE Transactions on Automatic Control,* vol. 50, pp. 1867-1872, 2005.

[14] A. Tahbaz-Salehi and A. Jadbabaie, "A necessary and sufficient condition for consensus over random networks," *Departmental Papers (ESE),* p. 355, 2008.

[15] Y. Zhang and Y.-P. Tian, "Consentability and protocol design of multi-agent systems with stochastic switching topology," *Automatica,* vol. 45, pp. 1195-1201, 2009.

[16] G. Wen, G. Hu, W. Yu, J. Cao, and G. Chen, "Consensus tracking for higher-order multi-agent systems with switching directed topologies and occasionally missing control inputs," *Systems & Control Letters,* vol. 62, pp. 1151-1158, 2013.

[17] J. Zhu and L. Yuan, "Consensus of high-order multi-agent systems with switching topologies," *Linear Algebra and Its Applications,* vol. 443, pp. 105-119, 2014.

[18] C.-J. Li and G.-P. Liu, "Consensus for heterogeneous networked multi-agent systems with switching topology and time-varying delays," *Journal of the Franklin Institute,* vol. 355, pp. 4198-4217, 2018.

[19] Z. Meng, Z. Lin, and W. Ren, "Leader–follower swarm tracking for networked Lagrange systems," *Systems & Control Letters,* vol. 61, pp. 117-126, 2012.

[20] H. Min, F. Sun, S. Wang, and H. Li, "Distributed adaptive consensus algorithm for networked Euler–Lagrange systems," *IET control theory & applications,* vol. 5, pp. 145-154, 2011.

[21] L. Gao, J. Li, X. Zhu, and W. Chen, "Leader-following consensus of linear multi-agent systems with state-observer under switching topologies," in *Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on,* 2012, pp. 572-577.

[22] H. Du, G. Wen, G. Chen, J. Cao, and F. E. Alsaadi, "A distributed finite-time consensus algorithm for higher-order leaderless and leader-following multiagent systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 47, pp. 1625-1634, 2017.

[23] C. Yuan, "Distributed adaptive switching consensus control of heterogeneous multi-agent systems with switched leader dynamics," *Nonlinear Analysis: Hybrid Systems,* vol. 26, pp. 274-283, 2017.

[24] X. Xie and X. Mu, "Observer-based Intermittent Consensus Control of Nonlinear Singular Multi-agent Systems," *International Journal of Control, Automation and Systems,* vol. 17, pp. 2321-2330, 2019.

[25] X. Deng, X. Sun, and S. Liu, "Iterative Learning Control for Leader-following Consensus of Nonlinear Multi-agent Systems with Packet Dropout," *International Journal of Control, Automation and Systems,* vol. 17, pp. 2135-2144, 2019.

[26] T. Wang, H. Fu, J. Li, Y. Zhang, X. Zhou, and X. Chen, "Optimal Consensus Control for Heterogeneous Nonlinear Multiagent Systems with Partially Unknown Dynamics," *International Journal of Control, Automation and Systems,* vol. 17, pp. 2400-2413, 2019.

[27] W. Guo, W. Luo, and Z. Zheng, "Lag group consensus for the second-order nonlinear multi-agent systems via adaptive control approach," *International Journal of Control, Automation and Systems,* vol. 17, pp. 1971-1977, 2019.

[28] Z. Li, Z. Duan, G. Chen, and L. Huang, "Consensus of multiagent systems and synchronization of complex networks: A unified viewpoint," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 57, pp. 213-224, 2010.

[29] M. Wu, H. Zhang, H. Yan, and H. Ren, "Self-triggered output feedback control for consensus of multi-agent systems," *Neurocomputing,* vol. 190, pp. 179-187, 2016.

[30] J. H. Seo, H. Shim, and J. Back, "Consensus of high-order linear systems using dynamic output feedback compensator: Low gain approach," *Automatica,* vol. 45, pp. 2659-2664, 2009.

[31] M. S. Radenković and M. Krstić, "Distributed adaptive consensus and synchronization in complex networks of dynamical systems," *Automatica,* vol. 91, pp. 233-243, 2018.

[32] T.-C. Lee, W. Xia, Y. Su, and J. Huang, "Exponential consensus of discrete-time systems based on a novel Krasovskii–LaSalle theorem under directed switching networks," *Automatica,* vol. 97, pp. 189-199, 2018.

[33] Y. Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian," in *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 99-103.

[34] Y. Cao and W. Ren, "Optimal linear-consensus algorithms: an LQR perspective," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 40, pp. 819-830, 2010.

[35] D. Zhang, X. Wang, and L. Meng, "Consensus problems for high-order LTI systems: a decentralized static output feedback method," *International Journal of Innovative Computing, Information and Control,* vol. 9, pp. 2143-2154, 2013.

[36] K. D. Listmann, "Novel conditions for the synchronization of linear systems," in *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, 2015, pp. 5605-5612.

[37] L. D. Col, "Sur l'analyse et la conception des controles distribues pour les systemes multi-agents soumis a des informations limitee.," PhD, Universite de toulouse, Francais, 2016.

[38] S. E. Tuna, "LQR-based coupling gain for synchronization of linear systems," *arXiv preprint arXiv:0801.3390,* 2008.

[39] S. E. Tuna, "Conditions for synchronizability in arrays of coupled linear systems," *IEEE Transactions on Automatic Control*, vol. 54, pp. 2416-2420, 2009.

[40] J. Wang and M. Xin, "Multi-agent consensus algorithm with obstacle avoidance via optimal control approach," *International Journal of Control,* vol. 83, pp. 2606-2621, 2010.

[41] D. S. Bernstein, "Nonquadratic cost and nonlinear feedback control," *International Journal of Robust and Nonlinear Control,* vol. 3, pp. 211-229, 1993.

[42] J. Wang and M. Xin, "Distributed optimal cooperative tracking control of multiple autonomous robots," *Robotics and Autonomous systems,* vol. 60, pp. 572-583, 2012.

[43] J. Wang and M. Xin, "Flocking of Multi-Agent Systems Using a Unified Optimal Control Approach," *Journal of Dynamic Systems, Measurement, and Control,* vol. 135, p. 061005, 2013.

[44] Y. Xie and Z. Lin, "Global optimal consensus for higher-order multi-agent systems with bounded controls," *Automatica,* vol. 99, pp. 301-307, 2019.

[45] M. E. Dehshalie, M. B. Menhaj, and M. Karrari, "Fault tolerant cooperative control for affine multi-agent systems: An optimal control approach," *Journal of the Franklin Institute,* vol. 356, pp. 1360-1378, 2019.

[46] R. Bailo, M. Bongini, J. A. Carrillo, and D. Kalise, "Optimal consensus control of the Cucker-Smale model," *IFAC-PapersOnLine,* vol. 51, pp. 1-6, 2018.

[47] J. Wang, S. Gong, S. Peeta, and L. Lu, "A real-time deployable model predictive control-based cooperative platooning approach for connected and autonomous

vehicles," *Transportation Research Part B: Methodological,* vol. 128, pp. 271-301, 2019.

[48] J. Wang and M. Xin, "Integrated optimal formation control of multiple unmanned aerial vehicles," *IEEE Transactions on Control Systems Technology,* vol. 21, pp. 1731-1744, 2013.

[49] S. M. LaValle, *Planning algorithms*: Cambridge university press, 2006.

[50] X. Ma and D. A. Castanon, "Receding horizon planning for Dubins traveling salesman problems," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 5453-5458.

[51] Y. Lu, X. Huo, O. Arslan, and P. Tsiotras, "Incremental multi-scale search algorithm for dynamic path planning with low worst-case complexity," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 41, pp. 1556-1570, 2011.

[52] W. Ren, J.-S. Sun, R. W. Beard, and T. W. McLain, "Nonlinear tracking control for nonholonomic mobile robots with input constraints: An experimental study," in *Proceedings of the 2005, American Control Conference, 2005.*, 2005, pp. 4923-4928.

[53] W. Li and C. G. Cassandras, "A cooperative receding horizon controller for multivehicle uncertain environments," *IEEE Transactions on Automatic Control,* vol. 51, pp. 242-257, 2006.

[54] J. Kim, K. Jo, D. Kim, K. Chu, and M. Sunwoo, "Behavior and path planning algorithm of autonomous vehicle A1 in structured environments," *IFAC Proceedings Volumes,* vol. 46, pp. 36-41, 2013.

[55] Y. Zhuang, H. Huang, S. Sharma, D. Xu, and Q. Zhang, "Cooperative path planning of multiple autonomous underwater vehicles operating in dynamic ocean environment," *ISA transactions,* vol. 94, pp. 174-186, 2019.

[56] S. Ruan and Y. Ma, "Optimization of Acceleration Motion Trajectory of SHEV Based on Radau Pseudospectral Method," *IFAC-PapersOnLine,* vol. 52, pp. 48-53, 2019.

[57] C. Xiong, D. Chen, D. Lu, Z. Zeng, and L. Lian, "Path planning of multiple autonomous marine vehicles for adaptive sampling using Voronoi-based ant colony optimization," *Robotics and Autonomous Systems,* vol. 115, pp. 90-103, 2019.

[58] E. Taheri, M. H. Ferdowsi, and M. Danesh, "Closed-loop randomized kinodynamic path planning for an autonomous underwater vehicle," *Applied Ocean Research,* vol. 83, pp. 48-64, 2019.

[59] F. Hegedüs, T. Bécsi, S. Aradi, and P. Gápár, "Model based trajectory planning for highly automated road vehicles," *IFAC-PapersOnLine,* vol. 50, pp. 6958-6964, 2017.

[60] C. Geyer, "Active target search from UAVs in urban environments," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 2366-2371.

[61] J. Tisdale, H. Durrant-Whyte, and J. K. Hedrick, "Path planning for cooperative sensing using unmanned vehicles," in *ASME 2007 International Mechanical Engineering Congress and Exposition*, 2007, pp. 715-723.

[62] R. He, A. Bachrach, and N. Roy, "Efficient planning under uncertainty for a target-tracking micro-aerial vehicle," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 1-8.

[63] C. G. Cassandras and W. Li, "A receding horizon approach for dynamic UAV mission management," in *Enabling Technologies for Simulation Science VII*, 2003, pp. 284-293.

[64] P. W. Sarunic, R. J. Evans, and B. Moran, "Control of unmanned aerial vehicles for passive detection and tracking of multiple emitters," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1-7.

[65] S. Temizer, M. Kochenderfer, L. Kaelbling, T. Lozano-Pérez, and J. Kuchar, "Collision avoidance for unmanned aircraft using Markov decision processes," in *AIAA guidance, navigation, and control conference*, 2010, p. 8040.

[66] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, pp. 566-580, 1996.

[67] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 19, pp. 1179-1187, 1989.

[68] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras, "Evolutionary algorithm based offline/online path planner for UAV navigation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 33, pp. 898-912, 2003.

[69] B. A. Kumar and D. Ghose, "Radar-assisted collision avoidance/guidance strategy for planar flight," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, pp. 77-90, 2001.

[70] J. Moreau, P. Melchior, S. Victor, L. Cassany, M. Moze, F. Aioun, and F. Guillemard, "Reactive path planning in intersection for autonomous vehicle," *IFAC-PapersOnLine,* vol. 52, pp. 109-114, 2019.

[71] K. Kawabata, L. Ma, J. Xue, C. Zhu, and N. Zheng, "A path generation for automated vehicle based on Bezier curve and via-points," *Robotics and Autonomous Systems*, vol. 74, pp. 243-252, 2015.

[72] H. S. Lim, S. Fan, C. K. Chin, S. Chai, N. Bose, and E. Kim, "Constrained path planning of autonomous underwater vehicle using selectively-hybridized particle swarm optimization algorithms," *IFAC-PapersOnLine,* vol. 52, pp. 315-322, 2019.

[73] C. Wei, R. Romano, F. Hajiseyedjavadi, N. Merat, and E. Boer, "Driver-centred Autonomous Vehicle Motion Control within A Blended Corridor," *IFAC-PapersOnLine,* vol. 52, pp. 212-217, 2019.

[74] S. Ragi and E. K. Chong, "UAV path planning in a dynamic environment via partially observable Markov decision process," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 49, pp. 2397-2412, 2013.

[75] C. Kreucher, A. O. Hero, K. Kastella, and D. Chang, "Efficient methods of non-myopic sensor management for multitarget tracking," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, 2004, pp. 722-727.

[76] D. P. Bertsekas and D. A. Castanon, "Rollout algorithms for stochastic scheduling problems," *Journal of Heuristics*, vol. 5, pp. 89-108, 1999.

[77] E. K. Chong, R. L. Givan, and H. S. Chang, "A framework for simulation-based network control via hindsight optimization," in *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187),* 2000, pp. 1433-1438.

[78] G. Wu, E. K. Chong, and R. Givan, "Burst-level congestion control using hindsight optimization," *IEEE Transactions on Automatic Control*, vol. 47, pp. 979-991, 2002.

[79] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," in *Proceedings of 1995 34th IEEE Conference on Decision and Control*, 1995, pp. 560-564.

[80] D. P. Bertsekas, "Dynamic programming and optimal control 3rd edition, volume II," *Belmont, MA: Athena Scientific,* 2011.

[81] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning* vol. 135: MIT press Cambridge, 1998.

[82] K. Ogata and Y. Yang, *Modern control engineering* vol. 5: Prentice hall Upper Saddle River, NJ, 2010.

[83] K. CONRAD, "UNIVERSAL IDENTITIES, II:$\otimes$ AND$\wedge$," Technical report, Univ. of Connecticut, 2017. Expository paper in math. uconn. edu/kconrad/blurbs/linmultialg/univid2. pdf2017.

[84] F. W. Warner, *Foundations of differentiable manifolds and Lie groups* vol. 94: Springer Science & Business Media, 2013.

[85] J. Almeida, C. Silvestre, and A. M. S. Pascoal, "Self-Triggered Output Feedback Control of Linear Plants in the Presence of Unknown Disturbances," *IEEE Trans. Automat. Contr.*, vol. 59, pp. 3040-3045, 2014.

[86] R. Biernacki, H. Hwang, and S. Bhattacharyya, "Robust stability with structured real parameter perturbations," *IEEE Transactions on Automatic Control*, vol. 32, pp. 495-506, 1987.

[87] W. F. Arnold and A. J. Laub, "Generalized eigenproblem algorithms and software for algebraic Riccati equations," *Proceedings of the IEEE,* vol. 72, pp. 1746-1754, 1984.

[88] S. A. Miller, Z. A. Harris, and E. K. Chong, "A POMDP framework for coordinated guidance of autonomous UAVs for multitarget tracking," *EURASIP Journal on Advances in Signal Processing,* vol. 2009, p. 724597, 2009.

[89] E. K. Chong, C. M. Kreucher, and A. O. Hero, "Partially observable Markov decision process approximations for adaptive sensing," *Discrete Event Dynamic Systems,* vol. 19, pp. 377-422, 2009.

[90] R. Bellman, "Dynamic programming," *Science,* vol. 153, pp. 34-37, 1966.

[91] B. Geiger, J. Horn, A. DeLullo, A. Niessner, and L. Long, "Optimal path planning of UAVs using direct collocation with nonlinear programming," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006, p. 6199.

[92] L. Ntaimo, X. Hu, and Y. Sun, "DEVS-FIRE: Towards an integrated simulation environment for surface wildfire spread and containment," *Simulation,* vol. 84, pp. 137-155, 2008.

[93] X. Hu, Y. Sun, and L. Ntaimo, "DEVS-FIRE: design and application of formal discrete event wildfire spread and suppression models," *Simulation,* vol. 88, pp. 259-279, 2012.

[94] R. C. Rothermel, *A mathematical model for predicting fire spread in wildland fuels* vol. 115: Intermountain Forest and Range Experiment Station, Forest Service, United …, 1972.

VITA

Poorya Shobeiry was born in Tehran, Iran. After completing his schoolwork at National Organization for Development of Exceptional Talents (NODET) in 2003, Poorya entered Razi University in Kermanshah, Iran. He received a Bachelor of Science with a major in Mechanical Engineering from Razi University in July 2008. He then attended Sharif University of Technology and earned his master’s degree in Mechanical Engineering (Energy Conversion) in 2011. He started his PhD at University of Missouri in August 2014 with a major in Mechanical and Aerospace Engineering (MAE) and during the last six years, he has been working on developing innovative optimal control approaches in order to solve consensus and path planning problems in multi-agent systems (MASs). His results have been implemented successfully in modeling many practical applications. He has also been a TA for the course of “Engineering Graphics, Fundamentals” since 2015. During his years of teaching, he managed to make effective contribution in teaching AutoCAD to the freshmen by making video tutorials to walk the students through most of the common issues they face during the course. These tutorials proved to be so valuable, that he was provided with a $1,000 honorarium by the university and his permission was gained for other TAs to use the tutorials as well. In his doctoral work, Poorya gained a high level of proficiency for controlling multiple Unmanned Aerial Vehicles (UAVs) for fighting wildfires – a project sponsored jointly by the National Science Foundation and the US Department of Agriculture. The results of this research are believed to be impactful and improve ability to suppress wildfires in arid climates.