NATIONAL TECHNICAL UNIVERSITY OF ATHENS

SCHOOL OF MECHANICAL ENGINEERS

MASTER OF SCIENCE
"AUTOMATION SYSTEMS"

Distributed Robust Synchronization for Nonlinear Multi-agent Systems with Prescribed Transient and Steady State Performance Under Switching Topologies

Giamarelos Nikolaos

Three-member Examination Committee

K.Kyriakopoulos (Supervisor) E. Papadopoulos G. Papalambrou

September 30, 2018

Abstract

In this thesis, we consider the synchronization control problem for uncertain high-order nonlinear multi-agent systems in a leader-follower scheme, under directed switching communication topology. A robust decentralized control protocol of minimal complexity is used that achieves prescribed, arbitrarily fast and accurate synchronization of the following agents and the leader. The control protocol is distributed in the sense that the control signal of each agent is calculated based solely on local relative state information from its neighborhood set. Additionally, no information regarding the agents' dynamic model is employed in the design procedure. Moreover, provided that the switching communication graphs are always connected and contrary to the relative works on multi-agent systems under switching topology, the controller-imposed transient and steady state performance bounds are fully decoupled from: i) the underlying switching communication topology, ii) the control gains selection and iii) the agents' model uncertainties, and are solely prescribed by certain designer-specified performance functions. Finally, simulation examples are included to illustrate and verify the approach.

Index terms— Multi-agent systems, decentralized control, prescribed performance, directed graph, switching topology, leader-follower

Περίληψη

Στην παρούσα διπλωματική εργασία, εξετάζουμε το πρόβλημα ελέγχου συγχρονισμού για αβέβαια μη γραμμικά συστήματα πολαπλών πρακτόρων υψηλής τάξης σε σχηματισμό αρχηγού-ακόλουθου υπό κατευθυνόμενες εναλλασσόμενες τοπολογίες επικοινωνίας. Ένα εύρωστο αποκεντρωμένο πρωτόκολλο ελέγχου ελάχιστης πολυπλοκότητας χρησιμοποιείται, το οποίο επιτυγχάνει προκαθορισμένο, αυθαίρετα γρήγορο και ακριβή συγχρονισμό μεταξύ των ακόλουθων πρακτόρων και του αρχηγού. Το πρωτόκολλο ελέγχου είναι διανεμημένο υπό την έννοια ότι το σήμα ελέγχου του κάθε πράκτορα υπολογίζεται αποκλειστικά και μόνο με βάση την τοπική πληροφορία σχετικής κατάστασης από το γειτονικό του σύνολο. Επιπλέον, καμία πληροφορία σχετικά με το δυναμικό μοντέλο των πρακτόρων δεν χρησιμοποιείται κατά τη διαδικασία σχεδιασμού. Επιπλέον, δεδομένου ότι οι εναλλασσόμενοι γράφοι επικοινωνίας είναι πάντα συνεδεμένοι και σε αντίθεση με σχετικές εργασίες σε συστήματα πολαπλών πρακτόρων υπό εναλλασσόμενη τοπολογία, τα επιβαλλόμενα από τον ελεγκτή όρια απόδοσης μεταβατικής και σταθερής κατάστασης είναι πλήρως αποσυζευγμένα από: i) την υποκείμενη εναλλασσόμενη τοπολογία επικοινωνίας, ii) την επιλογή των κερδών ελέγχου και iii) τις αβεβαιότητες του μοντέλου των πρακτόρων, και προκαθορίζονται μόνο από συγκεκριμένες συναρτήσεις απόδοσης που ορίζονται από το σχεδιαστή. Τέλος, περιλαμβάνονται παραδείγματα προσομοίωσης που απεικονίζουν και επαληθεύουν την προσέγγιση.

Λέξεις Κλειδιά— Συστήματα πολαπλών πρακτόρων, αποκεντρωμένος έλεγχος, προκαθορισμένη απόδοση, κατευθυνόμενος γράφος, εναλασσόμενη τοπολογία, αρχηγός-ακόλουθος

Contents

Chapter 1

Introduction

Multi-agent systems have recently emerged as an inexpensive and robust way of addressing a wide variety of tasks ranging from exploration, surveillance, and reconnaissance to cooperative construction and manipulation. The success of these systems relies on efficient information exchange and coordination between team members. More specifically, their intriguing feature consists on the fact that each agent makes decisions solely on the basis of its local perception of the environment, which has also been observed in many biological systems [1]. Thus, a challenging task is to design the decentralized control approach for certain global goals in the presence of limited information exchanges. In this direction, drawing some enlightenments from biological observations, distributed cooperative control of multi-agent systems has received considerable attention during the last two decades (see the seminal works [2], [3], [4] and [5] for example). In particular, the leader-follower scheme, according to which the following agents aim at reaching a consensus with the leader' s state, employing only locally available information, has become very popular, since in the absence of any central control system and without global coordinate information, following a leader is an accountable motivation.

Although the majority of the works on distributed cooperative control under switching topologies consider known and simple dynamic models, many practical engineering systems exist that fail to satisfy that assumption and which moreover are constantly subject to environmental disturbances. Thus, taking into account the inherent model uncertainties when designing robust distributed control schemes is of paramount importance. The problem of distributed consensus problem with external disturbances is investigated using H_∞ control for general linear dynamics under connected undirected graphs or strongly connected and balanced digraphs in [6], for first order uncertain linear systems under directed graphs containing a spanning tree in [7] and for discrete-time first order linear dynamics where the switching topology is subject to a Markov chain in [8], for second order linear systems under undirected and connected graphs in [9] and for high-order linear systems under directed graphs containing a spanning tree in [10]. $L_2 - L_\infty$ control has also been employed for the distributed consensus of high-order linear MAS under directed graphs containing a spanning tree in [11] and for the containment control of a single integrator under directed and balanced switching topologies modeled by a continuous-time Markov chain in [12], both of them subject to external disturbances. Another popular way to deal with

disturbances is the Disturbance Observers-Based Control (DOBC). Such control schemes are developed to estimate the disturbances for the tracking control of a single integrator in [13] and for the consensus problem of second order linear dynamics under undirected and directed strongly connected graphs in [14] and [15] respectively. This control strategy is also used for the consensus of high-order nonlinear MAS in [16]. All the aforementioned techniques are leading to the increasing desing complexity, owing to the interacting system dynamics as reflected by the local intercourse specifications. Thus, the proposed protocol appears to be more easy to derive and implement.

Many attempts have been made to address different multi-agent control problems for systems with unknown nonlinear dynamics and disturbances employing neural networks to approximate the nonlinear functions. Specifically, in [17] for high-order MAS under jointly connected or uniformly jointly quasi-strongly connected respectively for undirected or directed graphs and in [18] for first order heterogeneous non-affine pure-feedback MAS. Similarly, a distributed adaptive neural controller for first order non-affine nonlinear dynamics is proposed in [19], whereas the synchronization of nonlinear, non-identical high-order multi-agent systems with external disturbances under switching directed strongly connected and undirected connected graphs is investigated in [20] and [21] respectively. Unfortunately, the aforementioned schemes as well as that in [22], where fuzzy logic systems are utilized for the distributed containment control problem of first order non-affine pure-feedback nonlinear MAS under directed graphs, inherently introduce certain issues affecting closed loop stability and robustness.

Another important issue associated with decentralized cooperative control schemes of multi-agent systems under model uncertainties, concerns the transient and steady state response of the closed loop system. As we know, transient behavior (i.e., the convergence rate) is difficult to establish analytically as it is affected heavily by the agents' model dynamics and the status of the overall underlying interaction topology, both of which are considered unknown. According to the relative works to this direction, the transient time in [23] and the convergence rate in [24] depend on the switching conditions and furthermore on the agents' dynamics in [25]. The upper bound of convergence time in [26] and the lower bound of consensus speed in [27] are related to a protocol parameter and communication topology. The convergence rate in [28] is adjusted by the feedback gain matrix. As regards to nonlinear multi-agent systems, according to [29] the upper bound of the transient time depends on Lipschitz constants and order and control parameters. The compact set that the tracking errors in [17] converge to is adjusted by a design parameter. From the distributed control schemes introduced in [20] and [22] it appears that the synchronization or containment errors respectively can be reduced at will by increasing the value of the control gain or by choosing appropriate design parameters via trial-and-error. The same laborious and time consuming procedure of intutive tunning have to be done in [18] where the transient and steady state tracking errors can be made smaller by suitable choice of design parameters. In contrast with these works, the transient and steady state error perforance bounds imposed by the proposed controller are determined by certain performance functions the parameters of which are specified by the designer and don' t involve the agents' dynamics, the switching topology or the control gains.

In [30], a generic class of high-order nonlinear multi-agent systems, under a directed fixed communication protocol is considered. A robust, decentralized and approximation-free synchronization control scheme is designed in the sense that each agent utilizes only local relative state information from its neighborhood set to calculate its own control signal. Many efforts have been made to investigate how to achieve distributed synchronization in multi-agent systems under dynamically changing environments. Motivated from [30], the design methodology and the distributed protocol in it, are extended in this thesis, for the case of switching communication topologies. The robust synchronization with prescribed transient and steady state performance under switching topology is achieved provided that all possible switching graphs are directed and connected over the switching intervals. Compared wit the previous work, the main contributions of this thesis summarized in the following three aspects. First, the robust synchronization for uncertain high-order nonlinear multi-agent systems with external disturbances is considered. However, no prior knowledge of model nonlinearities/disturbances is employed and no approximation structures are used to acquire such knowledge. Second, the proposed distributed protocol is of low complexity while very few calculations are required to obtain the control signal. Third, the imposed transient and steady state response bounds are solely pre-determined by certain designer-specified performance functions and are fully decoupled from the agents' dynamic model, the underlying switching graph topologies and the control gains selection.

The rest of the thesis is organized as follows. In Chapter 2, we provide some theoritical klowledge about multi-agent systems and graph theory. In Chapter 3, we introduce the nonlinear dynamic model of the multi-agent system under consideration and then we define the disagreement error of the states. Subsequently, we define the synchronization control problem and give the conditions which the switching communication topology of the system must respect. The distributed control protocol that solves the robust synchronization problem with prescribed performance under switching communication topology is designed in Chapter 4. Chapter 5 presents the simulation results and their interpretation. Some conclusions are drawn in Chapter 6 and we explain their significance. Finally, in the Appendix the reader can find all the MATLAB codes used to export all the results throughout this thesis accompanied by the necessary comments.

Chapter 2

Preliminaries

This chapter includes all the necessary knowledge about multi-agent systems and graph theory (i.e. the way the agents are connected and communicate) that will be needed in the following chapters.

2.1 Multi-agent Systems

A multi-agent system (MAS) is a computerized system composed of multiple interacting intelligent agents. Multi-agent systems can solve problems that are difficult or impossible for an individual agent or a monolithic system to solve. Intelligence may include methodic, functional, procedural approaches, algorithmic search or reinforcement learning. Despite considerable overlap, a multi-agent system is not always the same as an agent-based model (ABM). The goal of an ABM is to search for explanatory insight into the collective behavior of agents (which don't necessarily need to be "intelligent") obeying simple rules, typically in natural systems, rather than in solving specific practical or engineering problems. The terminology of ABM tends to be used more often in the sciences, and MAS in engineering and technology. Applications where multi-agent systems research may deliver an appropriate approach include online trading, disaster response and social structure modelling. Typically multi-agent systems research refers to software agents. However, the agents in a multi-agent system could equally well be robots, humans or human teams. A multi-agent system may contain combined human-agent teams. The agent is capable of indipendent action on behalf of its user/owner, meaning that it can decide what to do to meet the goals for which it was designed, without requiring commands at any time.

A multi-agent system is a society composed of many agents that interact by exchanging messages over a network. In the general case each agent represents different interests and to interact successfully the agents must negotiate, cooperate and coordinate as people do in their societies.

Today, MAS technology is used for a wide range of control applications, such as programming and design, diagnostics, status monitoring, distributed control, system recovery, market simulation, network control and automation. In addition, this technology is spreading to a level where the first multi-agent systems are transported from the laboratory to use, allowing the industry to gain experience in the use of MAS and also to evaluate their effectiveness.

2.1.1 Leader follower scheme

A system of multiple agents in leader-follower scheme consists of a leader, who provides the appropriate information (the desired trajectory of the system for example), and its followers, who update their states using local feedback. This control strategy has a variety of applications including the formation, cooperative control, synchronization, consensus, containment control of the network agents.

2.2 Graph Theory

In a multi-agent system, each agent must be able to interact (co-operate, negotiate, coordinate) with other agents to do the tasks we assign. The way agents interact with each other is based on the theory of graphs.

Graph theory is a cognitive field of discrete mathematics, with applications in computer science, engineering science, chemistry, sociology, and so on. Although the origins of the theory were founded in the 18th century, it developed postwar as a separate field of applied mathematics.

In greek terminology the term graph is also used for the graphic representation of a function but they should not be confused. Among the various definitions that are encountered, one relatively complete states that graph theory is the study of graphs and their relationships. Mathematical computations on graphs are implemented with specific algorithms. Several different physical or technological structures can be modeled with graphs, such as e.g. computer networks, where a network diagram is modeled as a simple, directed graph.

2.2.1 Graph

In mathematics, and more specifically in graph theory, a graph is a structure amounting to a set of objects in which some pairs of the objects are in some sense "related". The objects correspond to mathematical abstractions called vertices (also called nodes or points) and each of the related pairs of vertices is called an edge (also called an arc or line). Typically, a graph is depicted in diagrammatic form as a set of dots for the vertices, joined by lines or curves for the edges. Graphs are one of the objects of study in discrete mathematics.

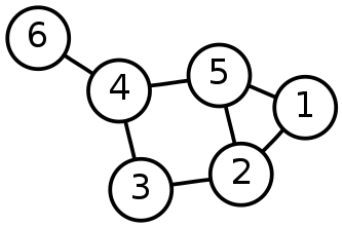


Figure 2.1: A graph with 6 nodes

Definition of graph

In one very common sense of the term, a graph is an ordered pair $G = (V, E)$ comprising a set V of vertices, nodes or points together with a set E of edges, arcs or lines, which are 2-element subsets of V (i.e., an edge is associated with two vertices, and the association takes the form of the unordered pair of the vertices). The vertices belonging to an edge are called the ends or end vertices of the edge. A vertex may exist in a graph and not belong to an edge. V and E are usually taken to be finite, and many of the well-known results are not true (or are rather different) for infinite graphs because many of the arguments fail in the infinite case. Moreover, V is often assumed to be non-empty, but E is allowed to be the empty set. The order of a graph is $|V|$, its number of vertices. The size of a graph is $|E|$, its number of edges. The degree or valency of a vertex is the number of edges that connect to it, where an edge that connects to the vertex at both ends (a loop) is counted twice.

Finite graph

A finite graph is a graph in which the vertex set and the edge set are finite sets. Otherwise, it is called an infinite graph. Most commonly in graph theory it is implied that the graphs discussed are finite. If the graphs are infinite, that is usually specifically stated.

Directed graph

A directed graph or digraph is a graph in which edges have orientations. It is written as an ordered pair $G = (V, E)$ with:

- V a set whose elements are called vertices, nodes, or points,
- E a set of ordered pairs of vertices, called arrows, directed edges, directed arcs, or directed lines.

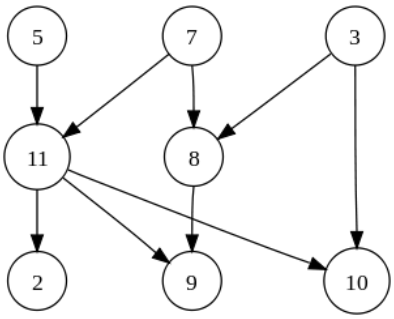


Figure 2.2: A directed graph

When an arrow (x, y) is considered to be directed from x to y, then y is called the *head* and x is called the *tail* of the arrow.

A directed graph G is called *symmetric* if, for every arrow in G, the corresponding inverted arrow also belongs to G. A symmetric loopless directed graph $G = (V, E)$ is equivalent to a simple undirected graph $G' = (V, A)$, where the

pairs of inverse arrows in E correspond one-to-one with the edges in A; thus the number of edges in G' is $|A| = |E|/2$, that is half the number of arrows in G.

For a vertex, the number of head ends adjacent to a vertex is called the *indegree* of the vertex and the number of tail ends adjacent to a vertex is its *outdegree*. Let $G = (V, E)$ and $v \in V$. The indegree of v is denoted $deg^-(v)$ and its outdegree is denoted $deg^+(v)$. A vertex with $deg^-(v) = 0$ is called a *source*, as it is the origin of each of its outcoming arrows. Similarly, a vertex with $deg^+(v) = 0$ is called a *sink*, since it is the end of each of its incoming arrows. If a vertex is neither a source nor a sink, it is called an *internal*. If for every vertex $v \in V$, $deg^+(v) = deg^-(v)$, the graph is called a *balanced* directed graph.

Undirected graph

An undirected graph is a graph in which edges have no orientation. The edge (x, y) is identical to the edge (y, x). That is, they are not ordered pairs, but unordered pairs - i.e., sets of two vertices (x, y) (or 2-multisets in the case of loops). The maximum number of edges in an undirected graph without a loop is $n(n-1)/2$, where n is the number of vertices.

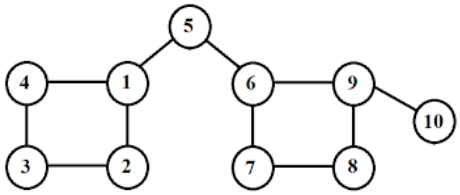


Figure 2.3: An undirected graph

2.2.2 Connectivity

In mathematics and computer science, connectivity is one of the basic concepts of graph theory: it asks for the minimum number of elements (nodes or edges) that need to be removed to disconnect the remaining nodes from each other. It is closely related to the theory of network flow problems. The connectivity of a graph is an important measure of its resilience as a network.

A graph is *connected* when there is a path between every pair of vertices. In a connected graph, there are no unreachable vertices. A graph that is not connected is *disconnected*. A graph G is said to be disconnected if there exist two nodes in G such that no path in G has those nodes as endpoints. A graph with just one vertex is connected. An edgeless graph with two or more vertices is disconnected. In an undirected graph, an unordered pair of vertices (x, y) is called connected if a path leads from x to y. Otherwise, the unordered pair is called disconnected. A connected graph is an undirected graph in which every unordered pair of vertices in the graph is connected. Otherwise, it is called a disconnected graph.

In a directed graph, an ordered pair of vertices (x, y) is called *strongly connected* if a directed path leads from x to y. Otherwise, the ordered pair is called *weakly connected* if an undirected path leads from x to y after replacing all of its directed edges with undirected edges. Otherwise, the ordered pair is called disconnected. A

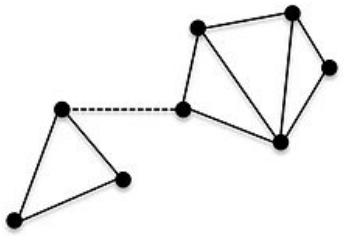


Figure 2.4: This graph becomes disconnected when the dashed edge is removed

strongly connected graph is a directed graph in which every ordered pair of vertices in the graph is strongly connected. Otherwise, it is called a weakly connected graph if every ordered pair of vertices in the graph is weakly connected. Otherwise it is called a disconnected graph.

A *k-vertex-connected graph* or *k-edge-connected graph* is a graph in which no set of $k-1$ vertices (respectively, edges) exists that, when removed, disconnects the graph. A k-vertex-connected graph is often called simply a *k-connected graph*.

Communication Topology

A multi-agent system can be depicted by a labeled graph, where each node represents an agent of the network. In order for the MAS to complete the task assigned to it, its agents must communicate with each other and exchange information. The communication topology that governs the graph dictates how this information exchange takes place.

When the communication topology i.e., the way the agents of a MAS interact, remains constant throughout the work being performed, then it is called a *fixed communication topology*. In this case, the interaction between agents can be modeled by a single graph wich indicates the flow of information from and to them.

However, due to the needs of the work being performed, it may be necessary to have more than one graph to describe the underlying communication topology. This means that the interaction between agents is modeled by a finite set of possible graphs that alternate with each other during the project. Then, unlike the fixed one, this is called *switching communication topology*.

Chapter 3

Problem Formulation

In this chapter we will deal with the formulation of the problem, wich includes the presentation of the dynamic model of the multi-agent system to be studied, the definition of the control problem that we will try to solve as the main part of this thesis and finally the assumptions concerning the underlying communication topology in order to solve the problem.

3.1 Dynamic Model

Consider a multi-agent group comprised of a leader and N followers, with the leading agent acting as an exosystem that generates a desired command/reference trajectory for the multi-agent group. The followers, which have to be controlled, obey an m-th order nonlinear dynamic model in canonical form, described as follows:

$$\begin{matrix} \dot{x}_{i,j} = x_{i,j+1}, j = 1, ...m-1 \\ \dot{x}_{i,m} = f_i(x_i) + g_i u_i + d_i(t) \end{matrix} \quad , i = 1, ..., N \tag{3.1}$$

where $\mathbf{x}_i = [x_{i,1}, ..., x_{i,m}]^T \in \mathbb{R}^m, i = 1, ..., N$ denote the state vector of each agent, $f_i : \mathbb{R}^m \to \mathbb{R}, i = 1, ..., N$ are unknown locally Lipschitz functions, $g_i \in \mathbb{R}, i = 1, ..., N$ are unknown constant parameters, $u_i \in \mathbb{R}, i = 1, ..., N$ are the control inputs and $d_i : \mathbb{R}_+ \to \mathbb{R}, i = 1, ..., N$ represent piecewise continuous and bounded external disturbance terms. Apart from a sufficient controllability condition, i.e., the parameters $g_i, i = 1, ..., N$ are considered strictly positive or strictly negative, no further assumption is made on the stability of the aformentioned open loop nonlinear dynamics. Moreover, with no loss of generality, it is assumed that $g_i > 0, i = 1, ..., N$. Additionally, defining the j-th order state vector of the multi-agent system as $\bar{x}_j = [x_{1,j}, ..., x_{N,j}]^T \in \mathbb{R}^N$, the multi-agent dynamic model in vector form is written as follows:

$$\begin{matrix} \dot{\bar{x}}_j = \bar{x}_{j+1}, j = 1, ...m-1 \\ \dot{\bar{x}}_m = F(\bar{x}) + G(u) + d(t) \end{matrix} \quad , i = 1, ..., N \tag{3.2}$$

where $\bar{x} = [\bar{x}_1^T, ..., \bar{x}_m^T]^T \in \mathbb{R}^{mN}$ is the overall state vector with $F(\bar{x}) = [f_1(\mathbf{x}_1), ..., f_N(\mathbf{x}_N)]^T$, $G = diag([g_1, ..., g_N])$, $u = [u_1, ...u_N]^T$ and $d(t) = [d_1(t), ..., d_N(t)]^T$. Furthermore, the state/command variables of the leading agent are given by

$\mathbf{x}_0(t) := [x_0(t), \dot{x}_0(t), ..., x_0^{m-1}(t)]^T \in \mathbb{R}^m$, where $x_0 : \mathbb{R}_+ \to \mathbb{R}$ and its derivatives up to m-th order are assumed to be continuous and bounded.

3.2 Disagreement error

In the sequel, the robust synchronization control problem with prescribed performance to be confronted in this work is formulated for the multi-agent system 3.2. More specifically, the target is to design a distributed control protocol, of low computational complexity for all the following agents, considering relative state feedback, unknown model nonlinearities and external disturbances, such that the j-th order disagreement error vectors:

$$\bar{\delta}_j(t) = \bar{x}_j(t) - \bar{x}_{0,j}(t) \in \mathbb{R}^N, j = 1, ..., m \tag{3.3}$$

with $\bar{x}_{0,j}(t) := [x_0^{j-1}(t), ..., x_0^{j-1}(t)]^T \in \mathbb{R}^N, j = 1, ..., m$ are driven with a minimum convergence rate within prescribed and arbitary small neighborhoods of the origin, keeping all closed loop signals bounded.

3.3 Underlying Communication Topology

A set G of directed graphs (digraphs) $G^{\sigma(t)} = (V, E^{\sigma(t)})$ is used to model the switching communication topologies among the followers, where $\sigma(t) : [0, +\infty] \to S$ is a piecewise constant switching signal with switching times $t_0, t_1, ...t_{k\in\mathbb{Z}}$, $G^{\sigma(t)}$ denotes the communication graph at time t and S is the index set associated with the elements of G. G represents the set of all possible graphs $G^{\sigma(t)}$ with the N agents, $V = \{\upsilon_1, ..., \upsilon_N\}$ denotes the set of vertices that represent the followers and $E^{\sigma(t)} \subseteq V \times V$ denotes the set of edges at time t. The graph is assumed to be simple, i.e., $(\upsilon_i, \upsilon_i) \notin E^{\sigma(t)}$ (there exist no self loops). The adjacency matrix associated with the digraph $G^{\sigma(t)}$ is denoted as $A^{\sigma(t)} = [\alpha_{i,j}(t)] \in \mathbb{R}^{N\times N}$ with $\alpha_{i,j}(t) \in \{0, 1\}, i, j = 1, ..., N$. If $\alpha_{i,j}(t) = 1$ then the agent i obtains information regarding the state of the j-th agent $\big($i.e., $(\upsilon_i, \upsilon_j) \in E^{\sigma(t)}\big)$, whereas if $\alpha_{i,j}(t) = 0$ then there is no state-information flow from agent j to agent i $\big($i.e., $(\upsilon_i, \upsilon_j) \notin E^{\sigma(t)}\big)$. The set of neighbors of a vertex υ_i at time t is denoted by $N_i^{\sigma(t)} = \{\upsilon_j : (\upsilon_i, \upsilon_j) \in E^{\sigma(t)}\}$ and the degree matrix is defined as $D^{\sigma(t)} = diag([D_i(t)]) \in \mathbb{R}^{N\times N}$ with $D_i(t) = \sum_{j\in N_i^{\sigma(t)}} \alpha_{i,j}(t)$. In this respect, the graph Laplacian $L^{\sigma(t)} = D^{\sigma(t)} - A^{\sigma(t)} \in \mathbb{R}^{N\times N}$. The state information of the leader node (labeled υ_0) is only provided to a subgroup of the N agents. The access of the following agents to the leader' s state at time t is modeled by a diagonal matrix $B^{\sigma(t)} = diag([b_1(t), b_2(t), ..., b_N(t)]) \in \mathbb{R}^{N\times N}$. If $b_i(t)$, $i \in \{1, 2, ..., N\}$ is equal to 1, then the i-th agent obtains state-information from the leader node; otherwise $b_i(t)$, $i \in \{1, 2, ..., N\}$ is equal to 0. The augmented set $\bar{G}$ contains all possible augmented diagraphs defined as $\bar{G}^{\sigma(t)} = (\bar{V}, \bar{E}^{\sigma(t)})$, where $\bar{V} = \{\upsilon_0, \upsilon_1, ..., \upsilon_N\}$ and $\bar{E}^{\sigma(t)} = E^{\sigma(t)} \cup \{(\upsilon_i, \upsilon_0) : b_i(t) = 1\} \subseteq \bar{V} \times \bar{V}$ while $\bar{N}_i^{\sigma(t)} = \{\upsilon_j : (\upsilon_i, \upsilon_j) \in \bar{E}^{\sigma(t)}\}$, $i = 1, ..., N$ denotes the augmented set of neighbors at time t.

To solve the aforementioned multi-agent control problem in section 3.2 the following assumption regarding the underlying communication graph topology is required.

Assumption A1: Each augmented graph $\bar{G}^{\sigma(t)}$ of set $\bar{G}$ always contains a spanning tree with the root being the leader node.

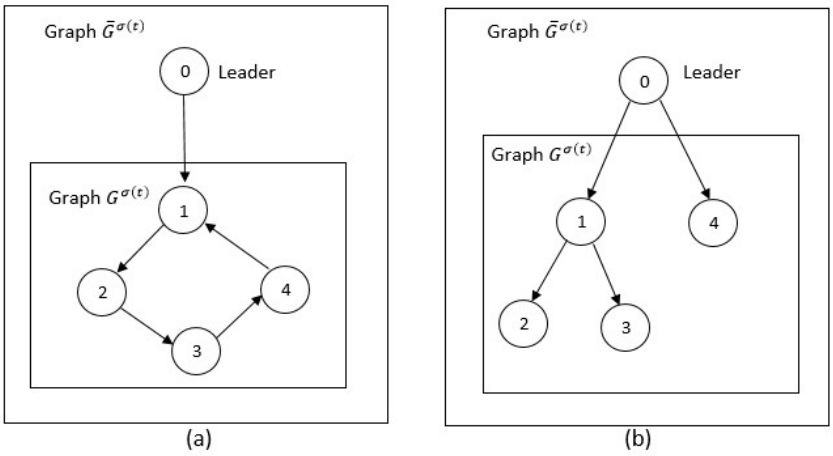


Figure 3.1: Two communication graphs at a specific time t that obey Assumption A1 and satisfy either the condition of items 1 or 2 of Remark 1, but not both simultaneously.

Remark 1: *The following non-equivalent common graph topologies satisfy Assumption A1:*

1. *The graph $G^{\sigma(t)}$ contains a spanning tree and at least one root node can get access to the leader node.*

2. *The augmented graph $\bar{G}^{\sigma(t)}$ has a hierarchical structure*[1].

Notice that in Fig. 3.1(a), the graph $\bar{G}^{\sigma(t)}$ does not have a hierarchical structure, i.e., the condition described in item 2 is not satisfied, although there exists a spanning tree in $G^{\sigma(t)}$ with its root getting access to the leader node. Similarly, in Fig. 3.1(b), the graph $\bar{G}^{\sigma(t)}$ that has a hierarchical structure, has a disconnected $G^{\sigma(t)}$ graph; thus there is no spanning tree within $G^{\sigma(t)}$. In addition, Assumption A1 dictates that $L^{\sigma(t)} + B^{\sigma(t)}$ is a nonsingular M-matrix[2] *[32].*

Finally, the following technical lemma regarding nonsingular M-matrices will be employed to derive the main results of this thesis.

Lemma 1. *([33] in pp. 168) Consider a nonsingular M-matrix $W \in \mathbb{R}^{N\times N}$. There exists a diagonal positive definite matrix $P = (diag(\bar{q}))^{-1}$, with $\bar{q} = W^{-1}\overline{\mathbf{1}}$ and $\overline{\mathbf{1}} := [1, ..., 1]^T \in \mathbb{R}^N$, such that $PW + W^TP$ is also positive definite.*

[1]The augmented graph $\bar{G}^{\sigma(t)}$ has a hierarchical structure when every node, except the one that denotes the leader, is subordinated to a single node [31].

[2]An M-matrix is a square matrix having its off-diagonal entries nonpositive and all principal minors nonnegative.

Chapter 4

Main Results

In this chapter will be presented the main theoretical results of this thesis. Specifically, the neighborhood errors are defined as well as the appropriate performance functions that achieve the convergence at a minimum rate of the disagreement errors in predetermined and arbitrarily small neighborhoods of the origin. Then follows the design of the distributed control protocol that solves the robust synchronization problem with prescribed performance under switching topologies, along with some theoretical remarks on it, such as the selection of the control parameters and the increase of dimensionality.

4.1 Sufficient Conditions

Owing to considering distributed control protocols with relative state information, the control law of each agent $i \in \{1, ..., N\}$ will be based on its neighborhood error feedback:

$$e_{i,j}(t) = \sum_{l \in N_i^{\sigma(t)}} \alpha_{i,l}(t)(x_{i,j} - x_{l,j}) + b_i(t)(x_{i,j} - x_0^{j-1}) \tag{4.1}$$

for $j = 1, ..., m$. Let us also define the j-th order neighborhood error vectors as $\bar{e}_j(t) = [e_{1,j}(t), ..., e_{N,j}(t)]^T$, $j = 1, ..., m$ which, employing the graph topology and after some trivial algebraic manipulations, become:

$$\bar{e}_j(t) = (L^{\sigma(t)} + B^{\sigma(t)})\bar{\delta}_j(t), \quad j = 1., ..., m \tag{4.2}$$

where the disagreement error vectors $\bar{\delta}_j(t)$, $j = 1, ..., m$ are defined in (3.3). Apparently, the j-th order neighborhood errors $\bar{e}_j(t)$, $j = 1, ..., m$ are expressed with respect to the leader state via the corresponding j-th order disagreement error variables $\bar{\delta}_j(t)$, $j = 1, ..., m$, which according to the problem statement are required to enter arbitrarily fast into arbitrarily small neighborhoods of the origin. However, the disagreement variables $\bar{\delta}_j(t)$, $j = 1, ..., N$ are global quantities and thus cannot be measured distributively based on the local intercourse specifications, as they involve information directly from the leader. Nevertheless, utilizing the nonsingularity of $L^{\sigma(t)} + B^{\sigma(t)}$, owing to Assumption A1, (4.2) yields:

$$\left\|\bar{\delta}_j(t)\right\| \leq \frac{\left\|\bar{e}_j(t)\right\|}{\sigma_{min}(L^{\sigma(t)}+B^{\sigma(t)})} \tag{4.3}$$

where $\sigma_{min}(L^{\sigma(t)}+B^{\sigma(t)})$ denotes the minimum singular value of $\left(L^{\sigma(t)}+B^{\sigma(t)}\right)$.

Remark 2. *It is concluded from (4.3) that the j-th order neighborhood errors $\bar{e}_j(t)$ represent a valid metric of the synchronization quality that is described by the disagreement errors $\bar{\delta}_j(t)$, $j=1,...,m$. In this respect, transient and steady state bounds imposed on the neighborhood errors $\bar{e}_j(t)$, $j=1,...,m$ can be directly translated into actual performance bounds on the disagreement variables $\bar{\delta}_j(t)$, $j=1,...,m$. However, $\sigma_{min}\left(L^{\sigma(t)}+B^{\sigma(t)}\right)$ is a global topology variable and thus cannot be employed in distributed control schemes to impose explicit bounds on $\bar{\delta}_j(t)$, $j=1,...,m$ via (4.3). To alleviate this issue, the conservative lower bound $\frac{\left(\frac{N-1}{N}\right)^{\frac{N-1}{2}}}{N^2+N-1} \leq \sigma_{min}\left(L^{\sigma(t)}+B^{\sigma(t)}\right)$ [34], that depends solely on the number of agents N and not on the graph topology can be utilized yielding:*

$$\left\|\bar{\delta}_j(t)\right\| \leq \frac{\left\|\bar{e}_j(t)\right\|}{\sigma_{min}(L^{\sigma(t)}+B^{\sigma(t)})} \leq \frac{N^2+N-1}{\left(\frac{N-1}{N}\right)^{\frac{N-1}{2}}}\left\|\bar{e}_j(t)\right\|. \tag{4.4}$$

Alternatively, a distributed estimation algorithm (power iteration or spectral analysis), similarly to [35], [36] for undirected or [37] for directed graphs, could be initially applied to estimate $\sigma_{min}\left(L^{\sigma(t)}+B^{\sigma(t)}\right)$.

To proceed, employing (3.2) and after some straightforward manipulations, the m-th order neighborhood error dynamics is obtained as follows:

$$\begin{aligned} \dot{\bar{e}}_j(t) &= \bar{e}_{j+1}(t), j=1,...m-1 \\ \dot{\bar{e}}_m &= (L^{\sigma(t)}+B^{\sigma(t)})(F(\bar{x})+G(u)+d(t)-\bar{x}_{0,m+1}(t)). \end{aligned} \tag{4.5}$$

To deal with the high order dynamics (4.5), certain time-varying surfaces over the neighborhood error space $\mathbb{R}^m$ are defined via the scalar equations $s_i(e_{i,1},...,e_{i,m})=0$, $i=1,...,N$, where:

$$s_i(e_{i,1},...,e_{i,m}) = \left(\frac{d}{dt}+\lambda\right)^{m-1} e_{i,1} = \sum_{z=0}^{m-1}\binom{m-1}{z}\lambda^z e_{i,m-z} \tag{4.6}$$

for $i=1,...,N$ with λ being a strictly positive constant[1]. Notice that all $s_i(e_{i,1},...,e_{i,m})=0, i=1,...,N$ can be calculated distributively, since only local relative state information from the neighborhood set is required (see (4.1) and (4.6)). Moreover, (4.6) can be considered as a set of stable linear filters with s_i and $e_{i,1}$, $i=1,...,N$ denoting their inputs and outputs respectively (i.e., $e_{i,1}(p)=\frac{s_i(p)}{(p+\lambda)^{m-1}}$, $i=1,...,N$ in the Laplace formulation with p denoting the Laplace frequency variable). Hence, the cooperative synchronization control problem (i.e., the problem of driving the disagreement variables $\bar{\delta}_j(t)$, $j=1,...,m$ close to the origin, or equivalently owing

[1]Denoting the derivative operand $\frac{d}{dt}$ in (4.6) by q, a polynomial with respect to q is formed that has $m-1$ identical real roots at $-\lambda$ and hence is Hurwitz.

to (4.3) driving the neighborhood error vectors $\bar{e}_j(t)$, $j = 1, ..., m$ close to the origin), can be reduced to the problem of driving $s_i(e_{i,1}, ..., e_{i,m})$, $i = 1, ..., N$ to zero, since the aforementioned stable linear filters have a unique equilibrium point $e_{i,j}(t) = 0$, $i = 1, ..., N$ and $j = 1, ..., m$, in case of zero input. Furthermore, bounds on $s_i(e_{i,1}, ..., e_{i,m}; t)$, $i = 1, ..., N$ can be directly translated into bounds on the neighborhood errors $e_{i,j}(t)$, $i = 1, ..., N$ and $j = 1, ..., m$ and consequently via (4.3) on the disagreement variables $\bar{\delta}_j(t)$, $j = 1, ..., m$. Hence, the scalars $s_i(e_{i,1}, ..., e_{i,m})$, $i = 1, ..., N$ represent valid performance metrics of the synchronization control problem. Henceforth, with a slight abuse of notation, we shall adopt $s_i^{\sigma(t)}(t)$, $i = 1, ..., N$ to denote the scalar error signals $s_i(e_{i,1}(t), ..., e_{i,m}(t))$, $i = 1, ..., N$ where $\sigma(t)$ is the aforementioned switching signal.

Assuming that $|s_i^{\sigma(t)}(t)| < \rho_i^{\sigma(t)}(t)$, $i = 1, ..., N$ for all $t \geq t_{l0}$, where $t_{l0}, l = 1, ..., k$ is l-th switching time, k is the number of switches and $\rho_i^{\sigma(t)}(t) = (\rho_{i0}^{\sigma(t)} - \rho_\infty)e^{-l^{\sigma(t)}t} + \rho_\infty$ are exponential performance functions, the following proposition dictates that the appropriate selection of the design parameters $\lambda, \rho_\infty, l^{\sigma(t)}$ and $\rho_{i0}^{\sigma(t)}$, $i = 1, ..., N$ guarantees the a priori specified exponential convergence of the disagreement variables $\bar{\delta}_j(t)$, $j = 1, ..., m$ arbitrarily close to the origin.

Proposition 1. *Consider the metrics $s_i^{\sigma(t)}(t)$, $i = 1, ..., N$ as defined in (4.6) and the performance functions $\rho_i^{\sigma(t)}(t) = (\rho_{i0}^{\sigma(t)} - \rho_\infty)e^{-l^{\sigma(t)}t} + \rho_\infty$, $i = 1, ..., N$ with $0 < l^{\sigma(t)} < \lambda, \rho_\infty > 0$ and $\rho_{i0}^{\sigma(t)} \equiv \rho_i^{\sigma(t)}(0) > |s_i^{\sigma(t)}(0)|, i = 1, ..., N$. If $|s_i^{\sigma(t)}(0)| < \rho_i^{\sigma(t)}(t), \forall t \geq t_{l0}, i = 1, ..., N$ and $l = 1, ..., k$ then the j-th order disagreement variables $\bar{\delta}_j(t)$, $j = 1, ..., m$ converge at least $e^{-l^{\sigma(t)}t}$ exponentially fast to the corresponding sets:*

$$\bar{\Delta}_j^{\sigma(t)} = \left\{ \delta \in \mathbb{R}^N : \|\delta\| \leq \frac{2^{j-1}\rho_\infty\sqrt{N}}{\sigma_{min}\left(L^{\sigma(t)} + B^{\sigma(t)}\right)\lambda^{m-j}} \right\} \quad for \;\; j = 1, ..., m. \tag{4.7}$$

Proof. See the Appendix in [30].

□

Remark 3. *Proposition 1 dictates that the size of the sets $\bar{\Delta}_j^{\sigma(t)}, j = 1, ..., m$ where all disagreement errors converge, is directly controlled by ρ_∞ and λ. In fact, by reducing ρ_∞ and enlarging λ, the maximum allowable size of the disagreement errors at steady state can be reduced at will, to the point of meeting the desired problem specifications. In the same spirit, enlarging λ does not harm the convergence rate of the disagreement errors, as in this way the admissible values of $l^{\sigma(t)}$ can be also enlarged.*

4.2 Distributed Control Protocol

The following theorem summarizes the main results of this thesis. It proposes a distributed control protocol that solves the robust synchronization problem with prescribed performance under switching topology for the considered high-order multi-agent system class, by guaranteeing $|s_i^{\sigma(t)}(t)| < \rho_i^{\sigma(t)}(t), i = 1, ..., N$ for all $t \geq t_{l0}, l = 1, ..., k$. The solution is of low complexity and does not incorporate any

information regarding either the multi-agent system nonlinearities, the external disturbances or the underlying communication graph topology. In addition, no approximating structures (i.e., neural networks, fuzzy systems, etc.) are utilized to acquire such knowledge.

Theorem 1. *Consider the multi-agent system* (3.2) *with switching communication graph topology satisfying Assumption A1. Given the metrics* $s_i^{\sigma(t)}(t), i = 1, ..., N$ *defined in* (4.6) *and the exponential performance functions* $\rho_i^{\sigma(t)}(t) = (\rho_{i0}^{\sigma(t)} - \rho_\infty)e^{-l^{\sigma(t)}t} + \rho_\infty$, $i = 1, ..., N$ *appropriately selected to introduce the desired performance bounds, with* $0 < l^{\sigma(t)}, \rho_\infty > 0$ *and* $\rho_{i0}^{\sigma(t)} \equiv \rho_i^{\sigma(t)}(0) > |s_i^{\sigma(t)}(0)| \geq 0, i = 1, ..., N$*; the distributed control protocol:*

$$u_i^{\sigma(t)}(s_i^{\sigma(t)}, t) = -\frac{k_i}{2\rho_i^{\sigma(t)}(t)} \frac{ln\left(\frac{1+\frac{s_i^{\sigma(t)}}{\rho_i^{\sigma(t)}(t)}}{1-\frac{s_i^{\sigma(t)}}{\rho_i^{\sigma(t)}(t)}}\right)}{\left(1+\frac{s_i^{\sigma(t)}}{\rho_i^{\sigma(t)}(t)}\right)\left(1-\frac{s_i^{\sigma(t)}}{\rho_i^{\sigma(t)}(t)}\right)} \quad , i = 1, ..., N \tag{4.8}$$

with $k_i > 0, i = 1, ..., N$ *solves the robust synchronization problem with prescribed performance under switching topology.*

Proof. See the Appendix in [30].

□

In the sequel, several remarks are presented that provide intuitive explanations on the control design procedure and reveal its intriguing properties.

4.2.1 Design Philosophy

As stated in Proposition 1, a sufficient condition to guarantee prescribed transient and steady state performance bounds on the disagreement variables $\bar{\delta}_j(t), j = 1, ..., m$ is to enforce $s_i^{\sigma(t)}(t), i = 1, ..., N$ to evolve strictly within the envelope constructed with the aid of the exponential performance functions $\rho_i^{\sigma(t)}(t) = (\rho_{i0}^{\sigma(t)} - \rho_\infty)e^{-l^{\sigma(t)}t} + \rho_\infty$, $i = 1, ..., N$. Stated otherwise, $|s_i^{\sigma(t)}(t)| < \rho_i^{\sigma(t)}(t)$ for all $t \geq t_{l0}$, $i = 1, ..., N$ and $l = 1, ..., k$ or equivalently $-1 < \xi_{s_i}^{\sigma(t)}(t) \equiv \frac{s_i^{\sigma(t)}(t)}{\rho_i^{\sigma(t)}(t)} < 1$ for all $t \geq t_{l0}$, $i = 1, ..., N$ and $l = 1, ..., k$. Modulating $\xi_{s_i}^{\sigma(t)}(t)$ via the logarithmic function $\frac{1}{2}ln\left(\frac{1+\star}{1-\star}\right)$, which actually stands for the inverse hyperbolic tangent function $tanh^{-1}(\star)$, and selecting $\rho_i^{\sigma(t)}(0) > s_i^{\sigma(t)}(0)$, the signals $\varepsilon_i^{\sigma(t)}(t) = \frac{1}{2}ln\left(\frac{1+\xi_{s_i}^{\sigma(t)}(t)}{1-\xi_{s_i}^{\sigma(t)}(t)}\right)$, $i = 1, ..., N$ are initially well defined. It is not difficult to verify that maintaining the boundedness of $\varepsilon_i^{\sigma(t)}(t), i = 1, ..., N$ is equivalent to guaranteeing $|s_i^{\sigma(t)}(t)| < \rho_i^{\sigma(t)}(t), i = 1, ..., N$ for all $t \geq t_{l0}$, $l = 1, ..., k$. Therefore, the problem at hand can be visualized as minimizing the quadratic and positive definite objective function $V_\varepsilon^{\sigma(t)} = \frac{1}{2}\varepsilon^T(\xi_s^{\sigma(t)})P\varepsilon(\xi_s^{\sigma(t)})$ within the feasible region defined via $|\xi_{s_i}^{\sigma(t)}(t)| < 1$, $i = 1, ..., N$ for all $t \geq t_{l0}$, $l = 1, ..., k$. A careful inspection of the proposed controller (4.8) reveals that it actually operates similarly to barrier functions in constrained optimization, admitting high negative or positive values depending on whether $s_i^{\sigma(t)}(t) \to \rho_i^{\sigma(t)}(t)$ or $s_i^{\sigma(t)}(t) \to -\rho_i^{\sigma(t)}(t)$ respectively; eventually not

permitting $s_i^{\sigma(t)}(t)$ from reaching the performance boundaries. In order for the condition $|s_i^{\sigma(t)}(t)| < \rho_i^{\sigma(t)}(t), i = 1, ..., N$ to apply for all $t \geq t_{l0}$, $l = 1, ..., k$ to even when the communication graph topology switches, we can change the values of $l^{\sigma(t)}$ and $\rho_i^{\sigma(t)}(0), i = 1, ..., N$ in each switching time as follows: for $\rho_i^{\sigma(t)}(0), i = 1, ..., N$ we can easily choose each time its value so that $\rho_i^{\sigma(t)}(0) > |s_i^{\sigma(t)}(0)| \geq 0, i = 1, ..., N$. The value of $l^{\sigma(t)}$ in each switching time is calculated by the solution of the equation $\rho_i^{\sigma(t)}(t_f - t_{l0}) = (\rho_i^{\sigma(t)}(0) - \rho_\infty)e^{-l^{\sigma(t)}(t_f - t_{l0})} + \rho_\infty = \alpha\rho_\infty \Rightarrow ... \Rightarrow l^{\sigma(t)} = -\frac{ln\left(\frac{(\alpha-1)\rho_\infty}{\rho_i^{\sigma(t)}(0)-\rho_\infty}\right)}{t_f - t_{l0}}$, $i = 1, ..., N$, $l = 1, ..., k$, where $\alpha = 1 + \epsilon$ (i.e., slightly greater than 1) and t_f is the desired time in which the synchronization have to be ahieved.

4.2.2 Decentralization and structural complexity

The proposed control protocol for the generic class of multi-agent systems considered herein, is decentralized in the sense that each agent utilizes only local relative state information from its neighborhood set, expressed in a common frame, to calculate its own control signal. Moreover, it does not incorporate any prior knowledge of the model nonlinearities/disturbances or even of some corresponding upper/lower bounding functions, relaxing thus significantly the key assumptions made in the related literature. Additionally, no approximating structures (i.e., neural networks, fuzzy systems, etc.) have been employed to acquire such knowledge. Furthermore, no hard calculations (neither analytic nor numerical) are required to produce the control signal, thus making its distributed implementation straightforward. Therefore, the proposed synchronization protocol besides being decentralized, exhibits low structural complexity as well.

4.2.3 Robust Prescribed Performance

From the proof of Theorem 1 it can be deduced that the proposed control scheme achieves synchronization with prescribed transient and steady state performance under switching communication topology for the considered class of high-order nonlinear multi-agent systems, without residing in the need of rendering the uniform bound of $\|\varepsilon(t)\|$ (i.e., $\bar{\varepsilon}$ in (30) in the Appendix of [30]) arbitrarily small, by adopting extreme values of the control gains k_i, $i = 1, ..., N$. More specifically, (31) in the Appendix of [30] and consequently (32) in the Appendix of [30], which encapsulates the prescribed performance notion, hold no matter how large the finite bound $\bar{\varepsilon}$ is. In the same spirit, large model uncertainties can be compensated for, as they affect only the size of $\bar{\varepsilon}$ through $\bar{F}$, leaving unaltered the achieved stability properties. Hence, the performance bounds, which are solely determined by the performance functions $\rho_i^{\sigma(t)}(t)$, $i = 1, ..., N$ becomes isolated against model uncertainties, extending greatly the robustness of the proposed control scheme. Furthermore, and contrary to the standard distributed control schemes for switching graphs, whose convergence rate is dictated by the connectivity level (i.e., the smallest singular value of $L^{\sigma(t)} + B^{\sigma(t)}$), the transient response of the proposed scheme is independent of the underlying topology as long as Assumption A1 holds.

4.2.4 Control Parameters selection

Unlike what is common practice in the related literature, the prescribed performance bounds are explicitly and solely determined by appropriately selecting the parameters $l^{\sigma(t)}$, ρ_∞ of the performance functions $\rho_i^{\sigma(t)}(t)$, $i = 1, ..., N$. In particular, the decreasing rate $l^{\sigma(t)}$ of $\rho_i^{\sigma(t)}(t)$, $i = 1, ..., N$ introduces directly a lower bound on the speed of convergence of the disagreement variables $\bar{\delta}_j(t)$, $j = 1, ..., m$. Furthermore, $\rho_\infty = \lim_{t\to\infty} \rho_i^{\sigma(t)}(t)$, $i = 1, ..., N$ and $\lambda > l^{\sigma(t)} > 0$ regulate via (4.7) the maximum allowable error at steady state. In that respect, the attributes of the performance functions $\rho_i^{\sigma(t)}(t)$, $i = 1, ..., N$ are selected a priori, in accordance to the desired transient and steady state performance specifications, except from $l^{\sigma(t)}$ that is calculated as described in section 4.2.1. Additionally, an extra condition concerning the initial value of the performance functions has to be satisfied (i.e., $\rho_i^{\sigma(t)}(0) > |s_i^{\sigma(t)}(0)|$, $i = 1, ..., N$), which guarantees in Phase A of the proof in the Appendix of [30] of the Theorem 1 that $\xi(0) \in \Omega_\xi$. Nevertheless, it is stressed that the initial value $\rho_i^{\sigma(t)}(0)$, $i = 1, ..., N$ of the performance functions does not affect either their transient or their steady state properties, as mentioned earlier. Moreover, since $s_i^{\sigma(t)}(t)$, $i = 1, ..., N$ depend solely on the neighborhood errors, which are available to each member of the multi-agent group, the aforementioned condition can be easily satisfied by selecting the initial value of the corresponding performance function $\rho_{i0}^{\sigma(t)} = \rho_i^{\sigma(t)}(0)$ to be grater than $s_i^{\sigma(t)}(0)$, $i = 1, ..., N$. It is underlined however that the proposed controller does not guarantee: i) the quality of the evolution of the error metrics $s_i^{\sigma(t)}(t)$, $i = 1, ..., N$ inside the performance envelopes and consequently of the disagreement error variables and ii) the control input characteristics (magnitude and slew rate). In this direction extensive simulation studies have revealed that the selection of the control gains k_i, $i = 1, ..., N$ can have positive influence.

4.2.5 Increasing dimensionality

In case of M-dimensional agent states, where $x_{i,j} \in \mathbb{R}^M$, $u_i \in \mathbb{R}^M$, $f_i : \mathbb{R}^m M \to \mathbb{R}^M$, $g_i \in \mathbb{R}^{M\times M}$, $d_i : \mathbb{R}_+ \to \mathbb{R}^M$ for $i = 1, ..., N$, $j = 1, ..., m$ and $x_0 : \mathbb{R}_+ \to \mathbb{R}^M$, the robust synchronization control problem with prescribed performance under switching communication topology can be solved following a similar design procedure, under the controllability assumption that the input matrices g_i, $i = 1, ..., N$ are diagonal and positive (or negative) definite. In the same direction, let us define the neighborhood error feedback:

$$e_{i,j}^{\sigma(t)}(t) = \sum_{l\in N_i^{\sigma(t)}} \alpha_{il}(t)(x_{i,j} - x_{l,j}) + b_i(t)\left(x_{i,j} - xi_0^{(j-1)}\right) \in \mathbb{R}^M \tag{4.9}$$

for $i = 1, ..., N$ and $j = 1, ..., m$ as well as the errors:

$$s_i^{\sigma(t)}(e_{i,1}^{\sigma(t)}, ..., e_{i,m}^{\sigma(t)}) = [s_{i1}^{\sigma(t)}, ..., s_{iM}^{\sigma(t)}]^T = \sum_{\zeta=0}^{m-1} \binom{m-1}{\zeta} \lambda^\zeta e_{i,m-\zeta}^{\sigma(t)}(t) \in \mathbb{R}^M \tag{4.10}$$

for $i = 1, ..., N$. Thus, adopting element-wise for each error component s_{ik}, $i = 1, ..., N$ and $k = 1, ..., M$ the corresponding performance functions $\rho_{ik}^{\sigma(t)}(t) =$

$(\rho_{ik0}^{\sigma(t)} - \rho_\infty)e^{-l^{\sigma(t)}t} + \rho_\infty$, $i = 1, ..., N$ and $k = 1, ..., M$ such that $\rho_{ik0}^{\sigma(t)} > |s_{ik}^{\sigma(t)}(0)|$, $i = 1, ..., N$ and $k = 1, ..., M$ with $l^{\sigma(t)}$, ρ_∞ incorporating the desired transient and steady state specifications as presented in Proposition 1, the following distributed control scheme is designed:

$$u_i^{\sigma(t)}(s_i, t) = -\begin{bmatrix} \frac{k_{i1}}{2\rho_{i1}(t)} \frac{ln\left(\frac{1+\frac{s_{i1}^{\sigma(t)}}{\rho_{i1}^{\sigma(t)}(t)}}{1-\frac{s_{i1}^{\sigma(t)}}{\rho_{i1}^{\sigma(t)}(t)}}\right)}{\left(1+\frac{s_{i1}^{\sigma(t)}}{\rho_{i1}^{\sigma(t)}(t)}\right)\left(1-\frac{s_{i1}^{\sigma(t)}}{\rho_{i1}^{\sigma(t)}(t)}\right)} \\ \vdots \\ \frac{k_{iM}}{2\rho_{iM}(t)} \frac{ln\left(\frac{1+\frac{s_{iM}^{\sigma(t)}}{\rho_{iM}^{\sigma(t)}(t)}}{1-\frac{s_{iM}^{\sigma(t)}}{\rho_{iM}^{\sigma(t)}(t)}}\right)}{\left(1+\frac{s_{iM}^{\sigma(t)}}{\rho_{iM}^{\sigma(t)}(t)}\right)\left(1-\frac{s_{iM}^{\sigma(t)}}{\rho_{iM}^{\sigma(t)}(t)}\right)} \end{bmatrix} \tag{4.11}$$

with $k_{ik} > 0$, $i = 1, ..., N$ and $k = 1, ..., M$. Following the same line of proof as in the 1-D case, it can be easily verified that the aforementioned control protocol guarantees $|s_{ik}^{\sigma(t)}(t)| < \rho_{ik}^{\sigma(t)}(t)$, $\forall t \geq t_{l0}$, $i = 1, ..., N$, $k = 1, ..., M$ and $l = 1, ..., k$ as well as the boundedness of all closed loop signals, which consequently leads to the solution of the robust synchronization control problem with prescribed performance under switching communication topology for multidimensional agent states.

Chapter 5

Simulation Results

This chapter presents some simulation results in order to demonstrate the effectiveness of the proposed distributed synchronization prescribed performance control (PPC) protocol under switching communication topology and points out its intriguing performance properties. Initially, the multi-agent system to be studied is defined, as well as the switching communication topology to which the system is subjected. Synchronization application results are then follow to confirm and visualize the correct operation of the proposed control protocol. In addition, a comparative simulation was conduced with a linear control protocol, the results of which demonstrate the advantages of the proposed controller. The chapter ends with the discussion and interpretation of the results obtained from the simulations.

5.1 Second Order Multi-agent System (MAS)

A multi-agent system comprised of a group of $N = 5$ two degrees of freedom agents is considered. The 2nd order dynamics of the agents are expressed in the workspace variables $p_i = [x_i, y_i]^T \in \mathbb{R}^2$, $i = 1, ..., 5$, as follows:

$$\ddot{p}_i = \mathbf{F}_i(\dot{p}_i)\dot{p}_i + \mathbf{G}_i u_i + \mathbf{D}_i \tag{5.1}$$

where $u_i = [u_{ix}, u_{iy}]^T \in \mathbb{R}^2$, $i = 1, ..., 5$ denote the control inputs, $\mathbf{D}_i = [d_{ix}sin(3t), d_{iy}cos(4t)]^T$, $i = 1, ..., 5$ represent external disturbances and

$$\mathbf{F}_i(\dot{p}_i) = \begin{bmatrix} -f_{ixx}\dot{y}_i & -f_{ixy}(\dot{x}_i + \dot{y}_i) \\ f_{ixy}\dot{x}_i & 0 \end{bmatrix},$$

$$\mathbf{G}_i = diag([g_{ix}, g_{iy}]),\ i = 1, ..., 5$$

with $f_{ixx} = 1$, $f_{ixy} = 2$, $f_{iyy} = 1$, $g_{ix} = 1.2$, $g_{iy} = 0.3$, $d_{ix} = 0.1$, $d_{iy} = 0.1$, $i = 1, ..., 5$. As can be seen, the system is in canonical, controllable form as described in (3.2) and is a sub-class of triangular systems. Furthermore, the command/reference trajectory (leader node) is a smooth representation $p_0(t) := [x_d(t), y_d(t)]^T \in \mathbb{R}^2$ of the acronym CSL with constant linear velocity (i.e., $\sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)} = const$).

5.2 Set of Switching Graphs

The underlying switching communication topology is described via a set of strongly connected diagraphs $\Gamma = \{G_1, G_2, G_3\}$ with a finite natural number index set $I_\hbar = \{1, 2, 3\}$ given in Fig. 5.1. The communication topology for the nonlinear multi-agent system in (5.1) is switching as $G^{\sigma(t)} : G_1 \to G_2 \to G_1 \to G_3 \to G_2 \to G_2 \to G_1 \to G_3 \to G_2 \to G_1$ with the dwell time $T_d = \pi/5 = 0.6283$s and the number of switches $k = 10$.

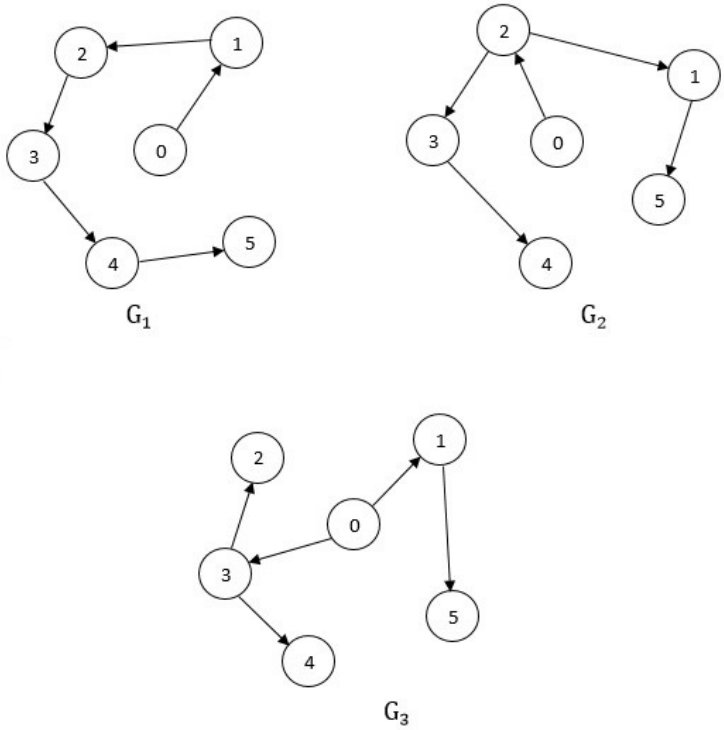


Figure 5.1: The interaction topology set Γ.

The switching signal is given in Fig. 5.2. The digraphs G_1, G_2, G_3 defined by the following augmented neighboring sets:

$$\bar{N}_1^{G_1} = \{0\},\ \bar{N}_2^{G_1} = \{1\},\ \bar{N}_3^{G_1} = \{2\},\ \bar{N}_4^{G_1} = \{3\},\ \bar{N}_5^{G_1} = \{4\}$$

$$\bar{N}_1^{G_2} = \{2\},\ \bar{N}_2^{G_2} = \{0\},\ \bar{N}_3^{G_2} = \{2\},\ \bar{N}_4^{G_2} = \{3\},\ \bar{N}_5^{G_2} = \{1\}$$

$$\bar{N}_1^{G_3} = \{0\},\ \bar{N}_2^{G_3} = \{3\},\ \bar{N}_3^{G_3} = \{0\},\ \bar{N}_4^{G_3} = \{3\},\ \bar{N}_5^{G_3} = \{1\}$$

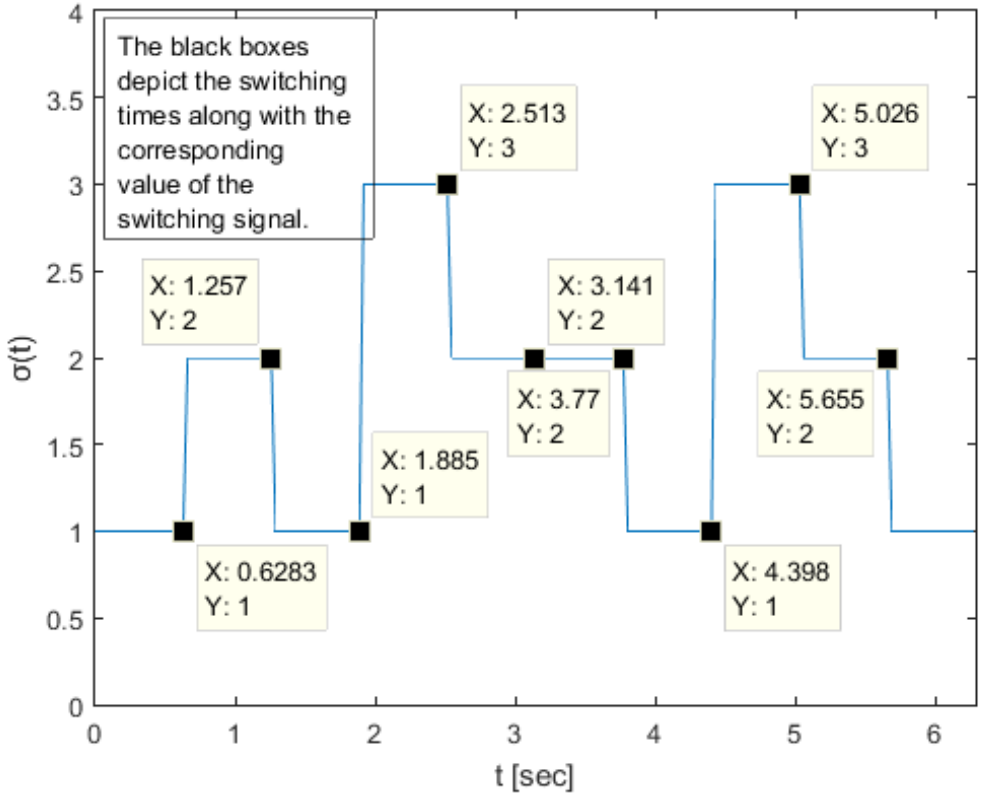


Figure 5.2: Switching signal $\sigma(t)$.

5.3 Synchronization Example

At this synchronization application, the command/reference trajectory of the leader is the parameterized with respect to time ellipse $x_d = 3\cos(t)$, $y_d = 1.5\sin(t)$. For the position and velocity disagreement variables $p_i - p_0$, $\dot{p}_i - \dot{p}_0$, $i = 1, ..., 5$ respectively, steady state errors 0.025 and 0.15 respectively are requested. In this direction, following section 4.2, the performance functions $\rho_{ik}^{\sigma(t)}(t) = (\rho_{ik0}^{\sigma(t)} - \rho_\infty)e^{-l^{\sigma(t)}t} + \rho_\infty$, $i = 1, ..., 5$ and $k \in \{x, y\}$ are selected with $\rho_{ik0}^{\sigma(t)} = 2|s_{ik}^{\sigma(t)}(0)| + 0.2$, $i = 1, ..., 5$ and $k \in \{x, y\}$. Moreover, employing Proposition 1 and adopting (4.4), the parameters λ, ρ_∞ are chosen as $\lambda = 3$ and $\rho_\infty = 0.0081$ such that the ultimate bounding sets, defined in (4.7), where the disagreement errors converge, meet the aforementioned steady state performance specifications. Finally, the control parameters k_{ik}, $i = 1, ..., 5$ and $k \in \{x, y\}$ are set to 0.5 to yield smooth state evolution and reasonable control effort. It was observed that decreasing the control gain values tends to intensify the oscillatory behavior of the agents states. The phenomenon is significantly smoothed out when increasing those values, at the expense of larger control action.

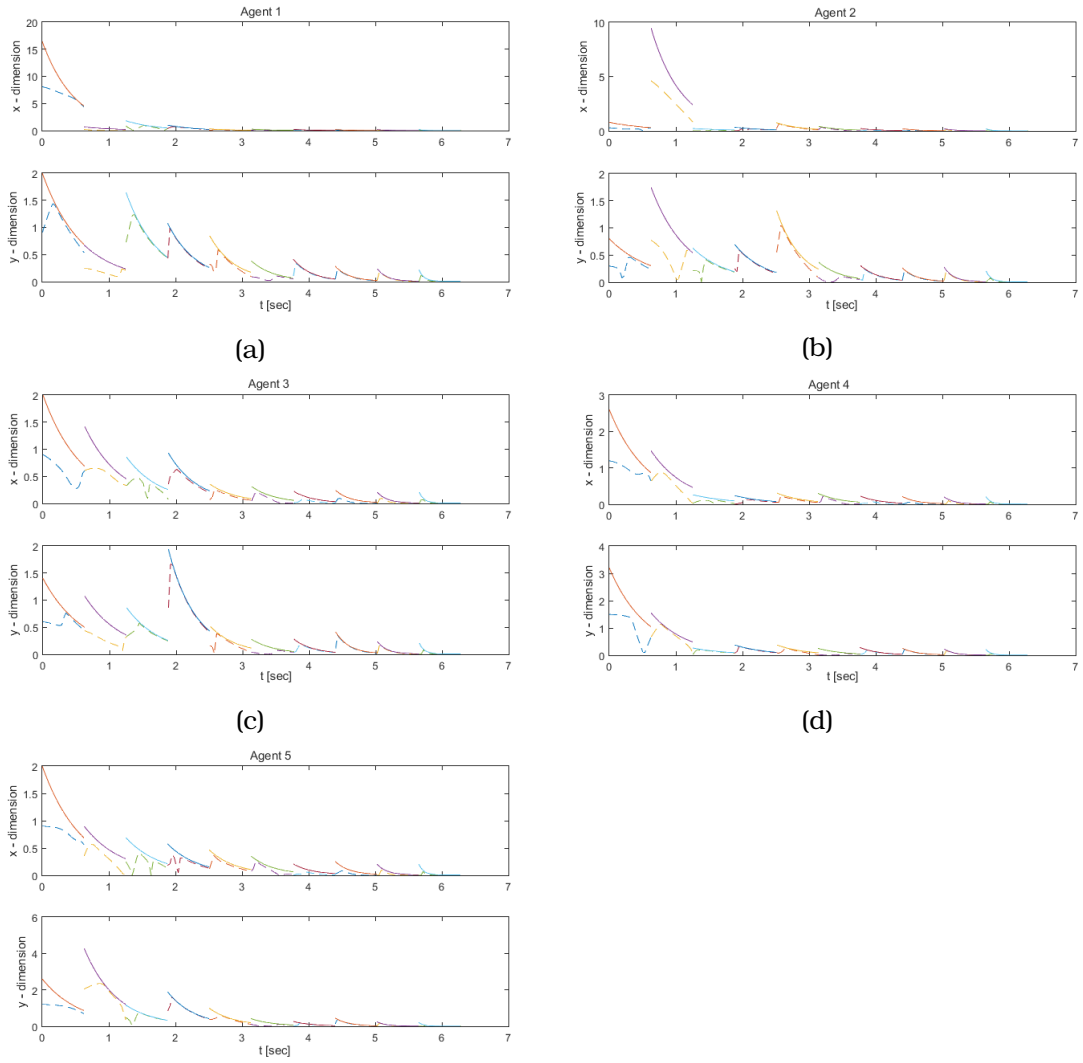


Figure 5.3: Evolution of the error metrics s_{ik}, $i = 1, ..., 5$, $k \in \{x, y\}$ (dashed lines) along with the imposed performance bounds (solid lines).

Simulation results with the proposed PPC protocol under the switching communication topology described above are illustrated in Figs. 5.3 - 5.11. More specifically, Fig. 5.3 depict the evolution of the errors $s_{ik}^{\sigma(t)}$, $i = 1, ..., 5$ and $k \in \{x, y\}$ along with the imposed performance bounds by the selected functions $\rho_{ik}^{\sigma(t)}$, $i = 1, ..., 5$ and $k \in \{x, y\}$. The trace of the agents in the workspace is pictured in Fig. 5.4. The trace of each agent begins from the corresponding numbered cycle. The evolution of the state variables (position and velocity) for each agent are provided in Figs. 5.5 - 5.9. Furthermore, the evolution of position and velocity disagreement errors is given in Fig. 5.10. Finally, the required control input signals are given in 5.11. In all these figures the curves associated with the agents comprised of ten different colorful parts as many as the switching time intervals. The corresponding MATLAB code that used for the simulation is in the Appendix A1.

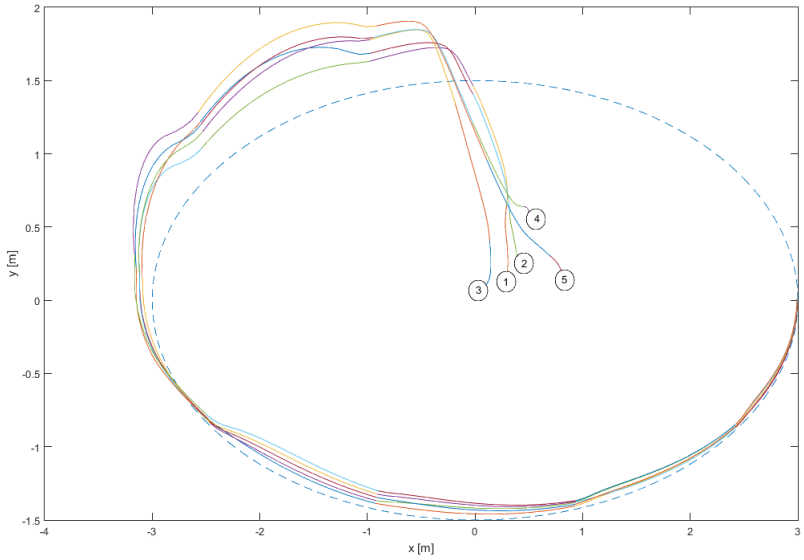


Figure 5.4: Trace of the agents in the workspace (solid lines) along with the ellipse (i.e., reference trajectory - dashed line).

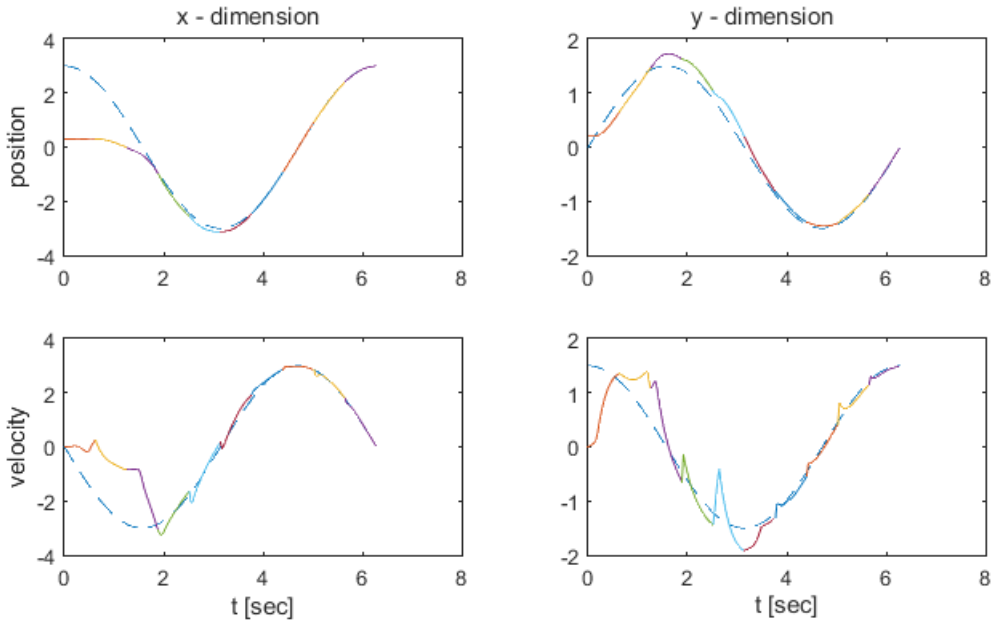


Figure 5.5: Evolution of the position and velocity states (solid lines) of agent 1 along with the reference position and velocity (dashed lines).

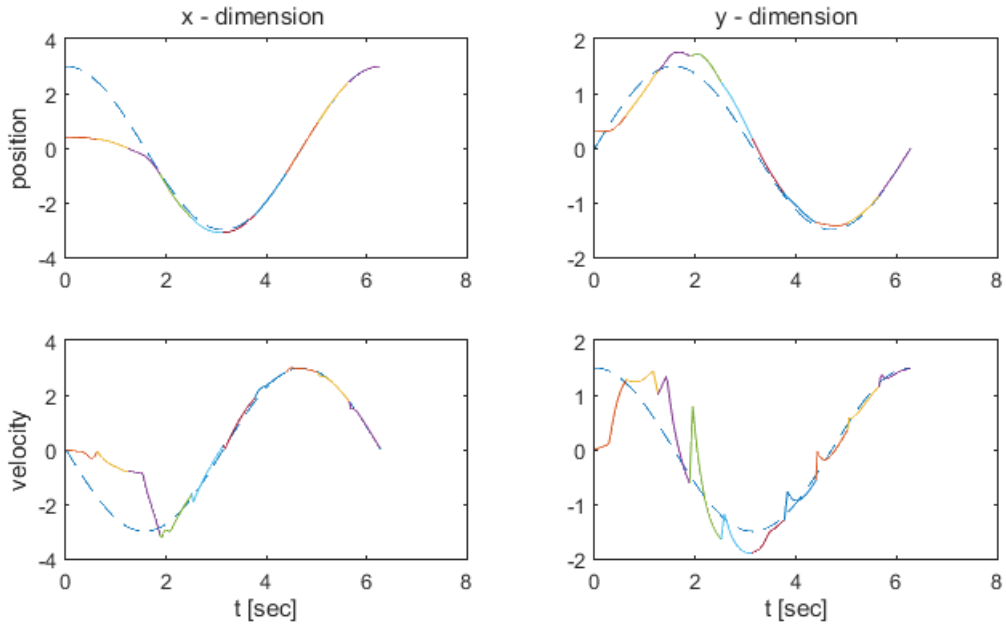


Figure 5.6: Evolution of the position and velocity states (solid lines) of agent 2 along with the reference position and velocity (dashed lines).

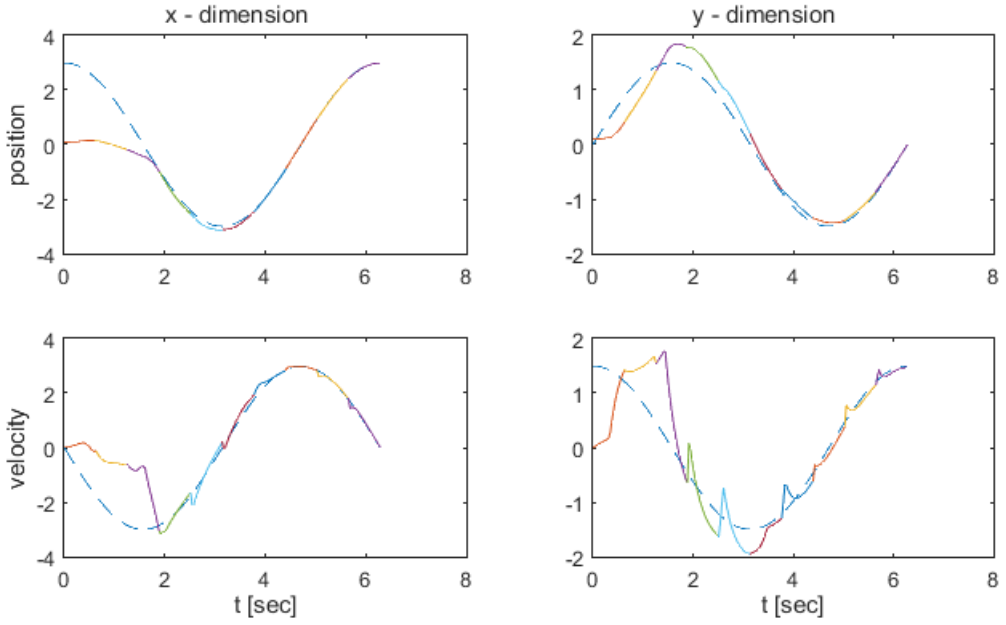


Figure 5.7: Evolution of the position and velocity states (solid lines) of agent 3 along with the reference position and velocity (dashed lines).

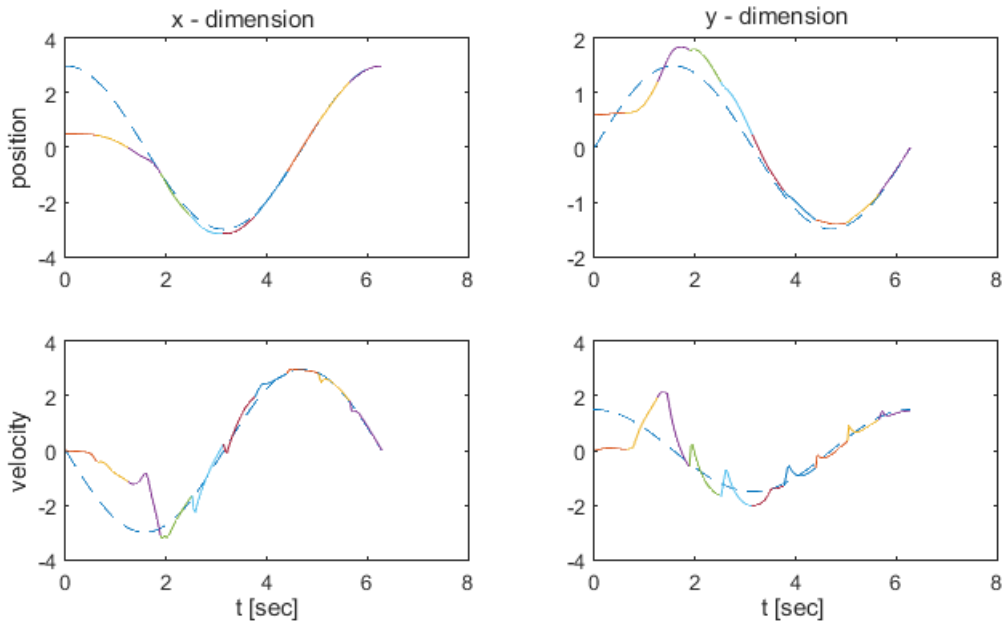


Figure 5.8: Evolution of the position and velocity states (solid lines) of agent 4 along with the reference position and velocity (dashed lines).

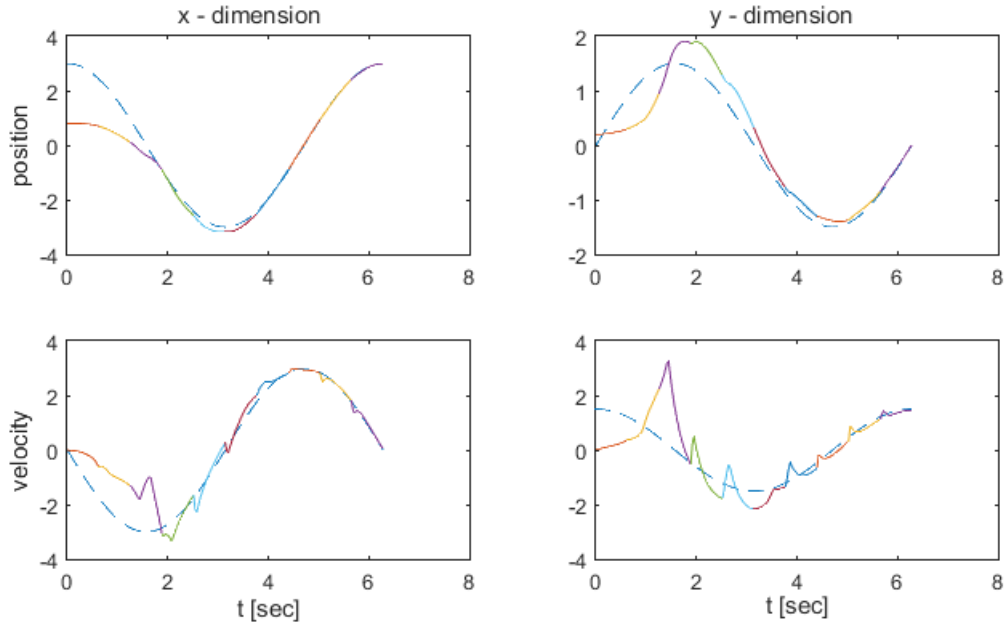
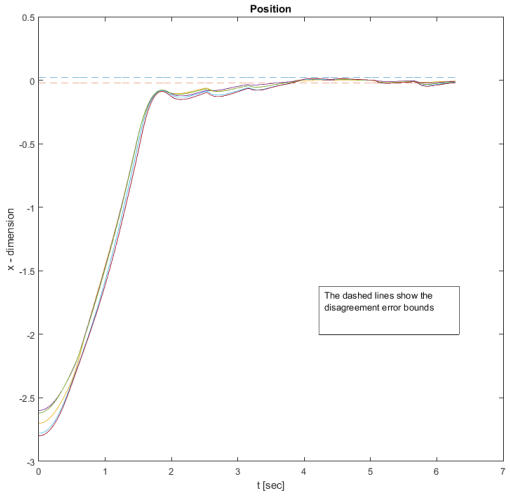
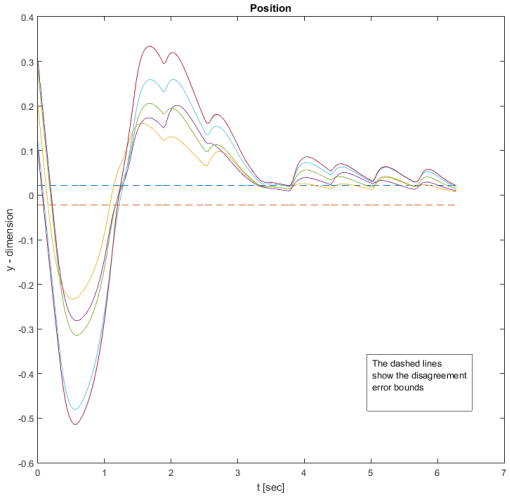


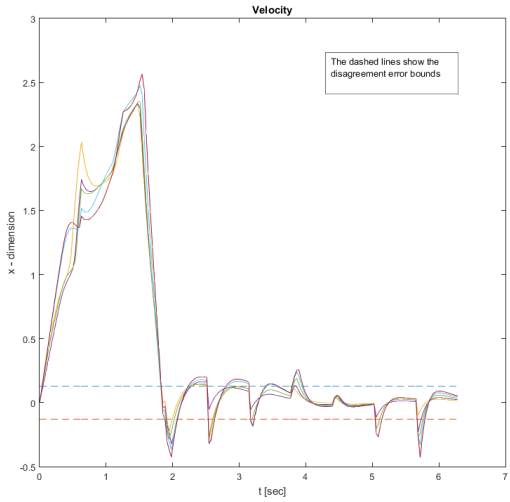
Figure 5.9: Evolution of the position and velocity states (solid lines) of agent 5 along with the reference position and velocity (dashed lines).



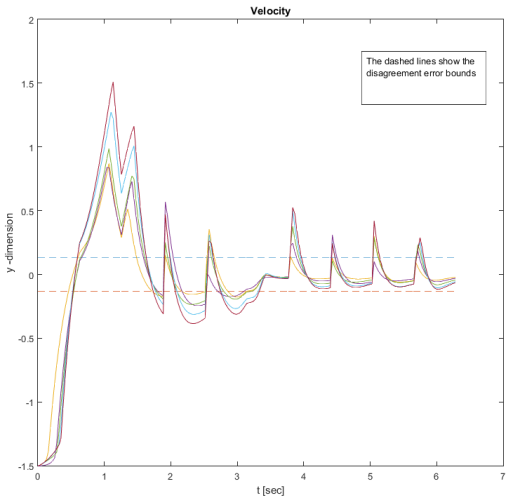
(a)



(b)



(c)



(d)

Figure 5.10: Evolution of the position and velocity diasagreement errors of the agents.

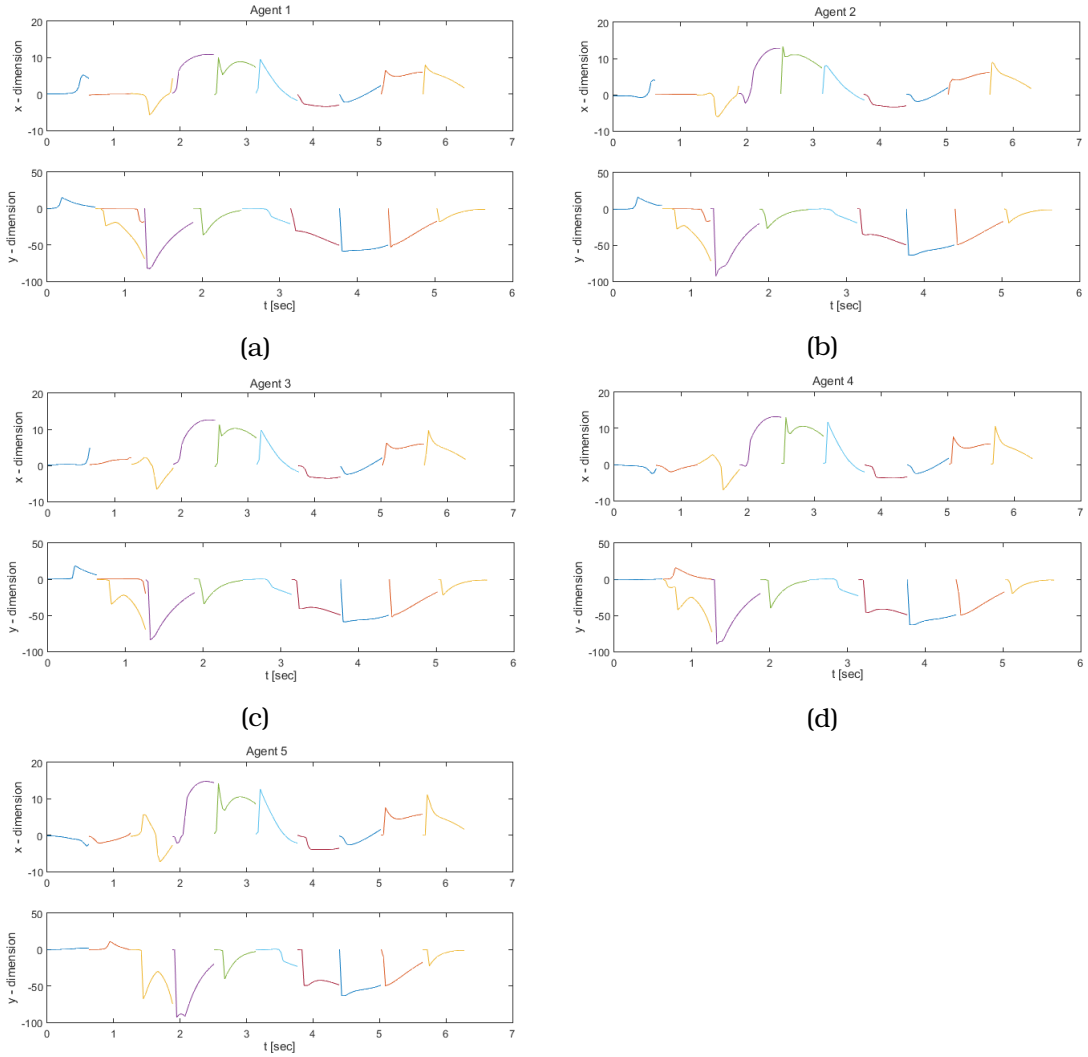


Figure 5.11: Required control input signal of the agents.

As it was predicted by the theoretical analysis, the synchronization control problem with prescribed performance under switching communication topology is solved despite the presence of external disturbances, the lack of knowledge of the agent's dynamics and the switches in the interaction of the agents with the leader. All the above figures were derived using the MATLAB code in the Appendix A1.

5.4 Comparative Simulation

In order to illustrate the advantages of the proposed controller, a comparative simulation was conducted with a consensus protocol for a multi-agent system with second-order dynamics [5]. The network system under consideration consists of n agents. Each agent updates its current state based upon the information received from its neighbors and the ith agent has the dynamics as follows:

$$\begin{aligned} \dot{x}_i &= v_i \\ \dot{v}_i &= u_i \end{aligned} \tag{5.2}$$

where x_i is the position state, v_i is the speed state and u_i is the control input. The consensus problem is solved by the following distributed control protocol:

$$u_i(t) = k_1 \sum_{s_j \in N_i} \alpha_{ij}(x_j(t) - x_i(t)) + k_2 \sum_{s_j \in N_i} \alpha_{ij}(v_j(t) - v_i(t)) \tag{5.3}$$

where $k_1, k_2 > 0$ are feedback gains and $\alpha_{ij} > 0$, $i = 1, ..., n$ are the elements of the adjacency matrix $A = [\alpha_{ij}]$. As regards the communication and connectivity of the agents, a directed network with switching topology $G^{\sigma(t)}$ that is kept strongly connected and balanced is considered.

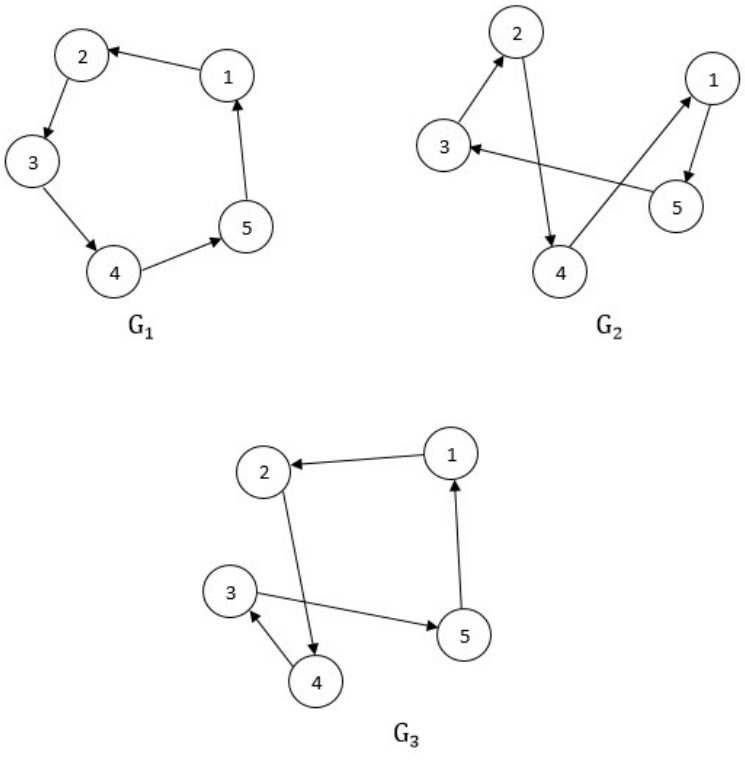


Figure 5.12: The interaction topology set Γ_1.

For the purposes of the simulation the nonlinear multi agent system in (5.1) had to be linearized as follows:

$$\ddot{p}_i = \mathbf{F}_i(\dot{p}_i)\dot{p}_i + \mathbf{G}_i v_i + \mathbf{D}_i \tag{5.4}$$

where $v_i = \mathbf{G}_i^{-1}(-\mathbf{F}_i(\dot{p}_i)\dot{p}_i + u_i)$ and u_i is the linear protocol in (5.3). The underlying switching communication topology is described via a set of strongly connected diagraphs $\Gamma_1 = \{G_1, G_2, G_3\}$ with a finite natural number index set $I_\hbar = \{1, 2, 3\}$ given in Fig. 5.12. The communication topology for the linear multi-agent system in (5.4) is switching as $G^{\sigma(t)} : G_1 \to G_2 \to G_1 \to G_3 \to G_2 \to G_2 \to G_1 \to G_3 \to G_2 \to G_1$ with the dwell time $T_d = \pi/5 = 0.6283$s and the number of switches $k = 10$. The feedback gains k_1 and k_2 are set to 5 to achieve consensus in the desired time. Figs. 5.13 - 5.14 depict the evolution of the position and velocity states of the agents with no disturbances (i.e., $d_{ix} = d_{iy} = 0$) and with disturbances and slightly modified dynamics respectively (i.e.,$d_{ix} = d_{iy} = 1$ and the parameters $f_{ixx}, f_{ixy}, f_{iyy}, g_{ix}, g_{iy}, i = 1, ..., 5$ were perturbed by 25% from their nominal values). The corresponding MATLAB code that used for the simulation is in the Appendix A1.

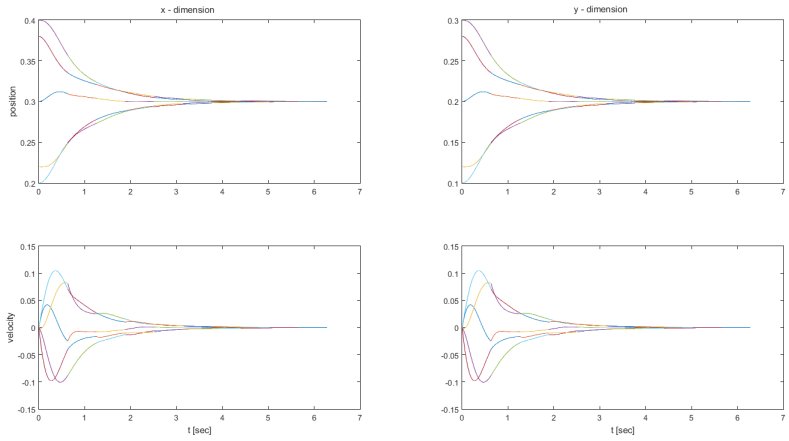


Figure 5.13: Evolution of the position and velocity states of the network with no disturbances with protocol in [5].

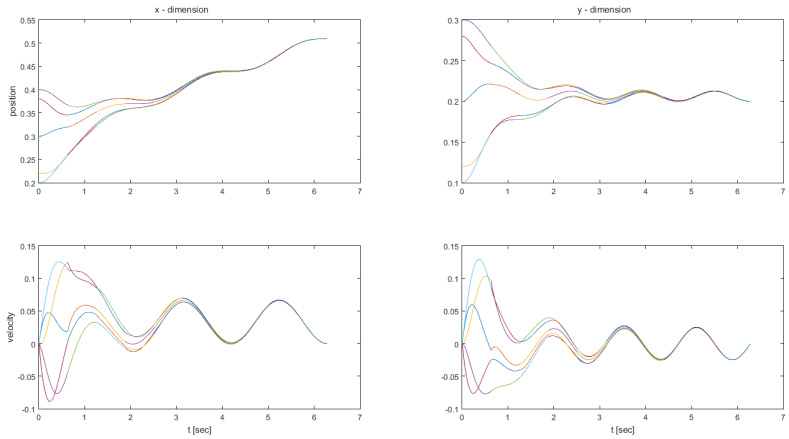


Figure 5.14: Evolution of the position and velocity states of the network with distrurbances and perturbed dynamics with protocol in [5].

The above figures were derived using the MATLAB code in the Appendix A1.

The respective simulation results of the system in (5.1) with the PPC protocol are given in Figs 5.15 - 5.16. The underlying switching communication topology Γ, the switching order $G^{\sigma(t)}$, the dwell time T_d, the number of switches k and the values of the control parameters $\rho_{ik0}^{\sigma(t)}$, λ, ρ_∞ k_{ik}, $i = 1, ..., 5$ and $k \in \{x, y\}$ are remain the same as in the above synchronization example. In order for the consensus to be achieved a slight modification had to be made in the disagreement error. More specifically, the new error is as follows:

$$\bar{\delta}_j(t) = \bar{x}_j(t) - s_{0,j} \in \mathbb{R}^N, j = 1, ..., m \tag{5.5}$$

with $s_{0,j} := \frac{1}{N} \sum_{i=1}^{N} \bar{x}_j(0), j = 1, ..., m$ (i.e., $s_{0,j}$, $j = 1, ..., m$ is the average of the initial values of the states).

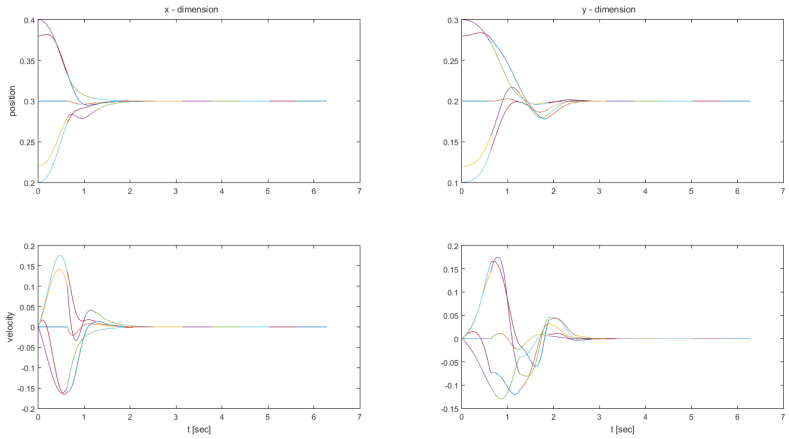


Figure 5.15: Evolution of the position and velocity states of the network with no disturbances with PPC protocol.

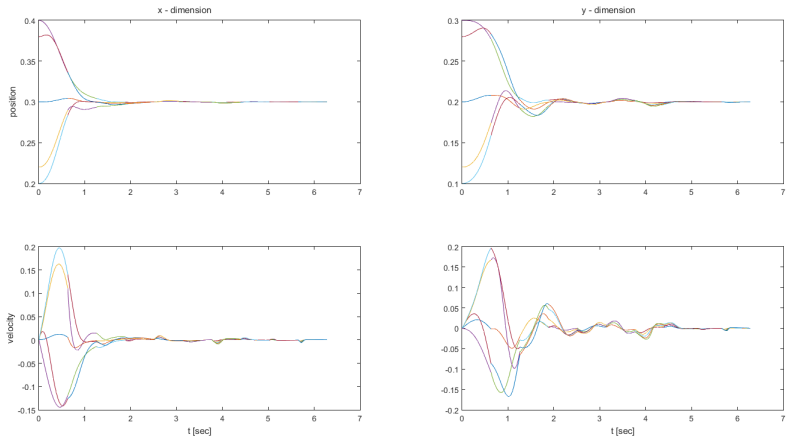


Figure 5.16: Evolution of the position and velocity states of the network with distrurbances and perturbed dynamics with PPC protocol.

5.5 Discussion of Results

As can be seen from the above simulation results, the proposed distributed communication protocol works well. More specifically, Fig. 5.4 shows that the robust synchronization of the agents with the leader is achieved at the desired time. Furthermore, Fig. 5.3 confirm that $|s_i^{\sigma(t)}(t)| < \rho_i^{\sigma(t)}(t)$, $\forall t \geq t_{l0}$, $i = 1, ..., 5$ and $l = 1, ..., k$ that is, that Proposition 1 is valid and therefore the control protocol of Theorem 1 solves the robust synchronization problem with prescribed performance under switching communication topology. Obviously, the considered switching graphs for the above application is consistent with Assumption A1. From Figs 5.5 - 5.9 we find that the position and velocity states of all agents converge with the reference position and velocity respectively at the prescribed time. Fig. 5.10 shows that the requested steady state errors are achieved and the prescribed transient and steady state performance is verified. In addition, we see that the control inputs of the agents in Fig 5.11 have the expected form (sinusoidal or co-sinusoidal). Finally, from Figs. 5.13 - 5.16 we can see that the consensus with the PPC protocol is maintained by the addition of disturbances and the slight change of dynamics while with the linear protocol in [5] not. This demonstrates the robustness of the proposed distributed controller.

Chapter 6

Conclusions

A distributed control protocol for uncertain nonlinear multi-agent systems in a leader-follower scheme, that achieves and maintains arbitrarily fast and accurate synchronization with the leader state under switching communication topology, was proposed. The developed scheme exhibits the following important characteristics. First, it is decentralized in the sense that the control signal of each agent is calculated on the basis of local relative state information from its neighborhood set, expressed in a common frame. Moreover, its complexity proves to be considerably low. Very few and simple calculations are required to output the control signal. Additionally, no prior knowledge of the agents' dynamic model is required and no approximating structures are employed to acquire such knowledge. Furthermore, the achieved prescribed transient and steady state performance, that is independent of the underlying graph topology and is explicitly imposed by designer-specified performance functions, simplifies significantly the control gains selection.

As a next step it would be the extension of the present work in the general case of random and arbitary large but finite number of communication switches. Our expectations for future work also focus on the collision avoidance and connectivity maintenance problems as well as the case where the agents do not share information expressed in a common frame, within a similar framework (i.e., robustness, prescribed transient and steady state performance), which would significantly increase its applicability in mobile multi-agent systems.

Acknowledgements

I would like to thank the Professor of School of Mechanical Engineering - NTUA, Mr. Kyriakopoulos K., who as my supervisor direct me wisely in my choice of subject and gave me the opportunity to elaborate this thesis. Equally important for me was the excellent collaboration with the Post - Doc Associate, Mr. Bechlioulis Ch. who, in a methodical way, helped me acquire a lot of knowledge about this subject of study and eventually to successfully complete this work. I would also like to thank my family for supporting me both mentally and financially throughout my studies, as well as my friends and fellow students who have always offered me help and cooperation.

References

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

[2] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

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

[4] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[5] P. Lin, Y. Jia, J. Du, and S. Yuan, "Distributed control of multi-agent systems with second-order agent dynamics and delay-dependent communications," *Asian Journal of Control*, vol. 10, no. 2, pp. 254–259, 2008.

[6] G. Wen, G. Hu, W. Yu, and G. Chen, "Distributed H_∞ consensus of higher order multiagent systems with switching topologies," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 5, pp. 359–363, 2014.

[7] P. Lin, Y. Jia, and L. Li, "Distributed robust H_∞ consensus control in directed networks of agents with time-delay," *Systems and Control Letters*, vol. 57, no. 8, pp. 643–653, 2008.

[8] Y. Liu and Y. Jia, "Consensus problem of high-order multi-agent systems with external disturbances: An H_∞ analysis approach," *International Journal of Robust and Nonlinear Control*, vol. 20, no. 14, pp. 1579–1593, 2010.

[9] C.-X. Zhang and H. Lin, "Agreement coordination for second-order multi-agent systems with disturbances," *Chinese Physics B*, vol. 17, no. 12, pp. 4458–4465, 2008.

[10] H. Zhang, R. Yang, H. Yan, and F. Yang, "H_∞ consensus of event-based multi-agent systems with switching topology," *Information Sciences*, vol. 370-371, pp. 623–635, 2016.

[11] Y. Cui and Y. Jia, "Robust $L_2 - L_\infty$ consensus control for uncertain high-order multi-agent systems with time-delay," *International Journal of Systems Science*, vol. 45, no. 3, pp. 427–438, 2014.

[12] X. Mu, B. Zheng, and K. Liu, "$L_2 - L_\infty$ containment control of multi-agent systems with markovian switching topologies and non-uniform time-varying delays," *IET Control Theory and Applications*, vol. 8, no. 10, pp. 863–872, 2014.

[13] H.-Y. Yang, H.-L. Zou, H.-X. Liu, F. Liu, M. Zhao, and F. Han, "Consensus tracking of multiagent systems with time-varying reference state and exogenous disturbances," *Mathematical Problems in Engineering*, vol. 2014, no. 213862, 2014.

[14] H. Yang, Z. Zhang, and S. Zhang, "Consensus of second-order multi-agent systems with exogenous disturbances," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 9, pp. 945–956, 2011.

[15] H.-Y. Y. andL. Guo and H.-L. Zou, "Robust consensus of multi-agent systems with time-delays and exogenous disturbances," *International Journal of Control, Automation and Systems*, vol. 10, no. 4, pp. 797–805, 2012.

[16] H.-Y. Yang, Z.-X. Zhang, and S. Zhang, "Consensus of mobile multiple agent systems with disturbance observer-based control," *Kongzhi Lilun Yu Yingyong/Control Theory and Applications*, vol. 27, no. 12, pp. 1787–1792, 2010.

[17] M. Rezaei, M. Kabiri, and M. Menhaj, "Adaptive consensus for high-order unknown nonlinear multi-agent systems with unknown control directions and switching topologies," *Information Sciences*, vol. 459, pp. 224–237, 2018.

[18] Y. Yang, D. Yue, and C. Dou, "Distributed adaptive output consensus control of a class of heterogeneous multi-agent systems under switching directed topologies," *Information Sciences*, vol. 345, pp. 294–312, 2016.

[19] M. Shahvali and K. Shojaei, "Distributed adaptive neural control of nonlinear multi-agent systems with unknown control directions," *Nonlinear Dynamics*, vol. 83, no. 4, pp. 2213–2228, 2016.

[20] S. Su and Z. Lin, "A multiple lyapunov function approach to distributed synchronization control of multi-agent systems with switching directed communication topologies and unknown nonlinearities," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 23–33, 2018.

[21] S. Su, Z. Lin, and A. Garcia, "Distributed synchronization control of multiagent systems with unknown nonlinearities," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 325–338, 2016.

[22] W. Wang, D. Wang, and Z. Peng, "Distributed containment control for uncertain nonlinear multi-agent systems in non-affine pure-feedback form under switching topologies," *Neurocomputing*, vol. 152, pp. 1–10, 2015.

[23] M. Franceschelli, A. Giua, A. Pisano, and E. Usai, "Finite-time consensus for switching network topologies with disturbances," *Nonlinear Analysis: Hybrid Systems*, vol. 10, no. 1, pp. 83–93, 2013.

[24] X. Li, C. Lei, L. Dong, and S.-K. Nguang, "Guaranteed convergence control for consensus of mobile sensor networks with dynamical topologies," *International Journal of Distributed Sensor Networks*, vol. 12, no. 11, 2016.

[25] J. Qin and C. Yu, "Exponential consensus of general linear multi-agent systems under directed dynamic topology," *Automatica*, vol. 50, no. 9, pp. 2327–2333, 2014.

[26] X. Wang, J. Li, J. Xing, R. Wang, L. Xie, and X. Zhang, "A novel finite-time average consensus protocol for multi-agent systems with switching topology," *Transactions of the Institute of Measurement and Control*, vol. 40, no. 2, pp. 606–614, 2018.

[27] D. Nguyen, "Distributed consensus design for a class of uncertain linear multi-agent systems under unbalanced randomly switching directed topologies," *Mathematical Problems in Engineering*, vol. 2018, no. 8081264, 2018.

[28] B. Cui, C. Zhao, T. Ma, and C. Feng, "Leader-following consensus of nonlinear multi-agent systems with switching topologies and unreliable communications," *Neural Computing and Applications*, vol. 27, no. 4, pp. 909–915, 2016.

[29] J. Davila and A. Pisano, "Fixed-time consensus for a class of nonlinear uncertain multi-agent systems," *Proceedings of the American Control Conference*, vol. 2016, no. 7525493, pp. 3728–3733, 2016.

[30] C. P. Bechlioulis and G. A. Rovithakis, "Decentralized robust synchronization of unknown high order nonlinear multi-agent systems with prescribed transient and steady state performance," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 123–134, 2017.

[31] H. Du, S. Li, and C. Qian, "Finite-time attitude tracking control of spacecraft with application to attitude synchronization," *IEEE Transactions on Automatic Control*, vol. 56, no. 11, pp. 2711–2717, 2011.

[32] P. N. Shivakumar and K. H. Chew, "A sufficient condition for nonvanishing of determinants," *Proceedings of the American Mathematical Society*, vol. 43, no. 1, pp. 63–66, 1974.

[33] Z. Qu, *TCooperative Control of Dynamical Systems: Applications to Autonomous Vehicles*. New York, USA: Springer, 2009.

[34] Y. P. Hong and C. T. Pan, "A lower bound for the smallest singular value," *Linear Algebra and its Applications*, vol. 172, pp. 27–32, 1992.

[35] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390–396, 2010.

[36] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized estimation of laplacian eigenvalues in multi-agent systems," *Automatica*, vol. 49, no. 4, pp. 1031–1036, 2013.

[37] L. Sabattini, C. Secchi, and N. Chopra, "Decentralized estimation and control for preserving the strong connectivity of directed graphs," *IEEE Transactions on Cybernetics*, 2014. article in Press, DOI:10.1109/TCYB.2014.2369572.

Appendix A

Appendix

A.1 MATLAB Code

Main m-file for system simulation - Synchronization with PPC:

```
p1=[0.30;0.20;0;0;0.4;0.3;0;0;0.1;0.1;0;0;0.5;0.6;0;0;0.8;0.2;0;0];
% initial values of state variables
options=odeset('reltol',1e-6,'abstol',1e-9);
ts=zeros([21 10]);
sol=zeros([21 200]);
s=zeros([21 100]);
r=zeros([21 100]);
gp=zeros([21 100]);
gv=zeros([21 100]);
for i=0:20:180; % 10 iterations, one for each switching topology
t1=(i/20)*pi/5:0.05*pi/5:((i+20)/20)*pi/5;
% the switching time intervals
tstart=t1(1); % switching times
[l,r0]=calc_r0_l(p1,tstart); % calculation of parameters r0
                             % and l for each agent
save('t_r0_l.mat','l','r0','tstart');
ls1=@mal1;
[ts1,sol1]=ode15s(ls1,t1,p1,options);
% solve the system with ode15s
[s1,r1,gp1,gv1]=calc_all(ts1,sol1,l,r0,tstart);
% calculation of the values of error signals 's1',
% performance functions 'r1'
% and neighborhood errors 'gp1' and 'gv1'
p1=sol1(21,:)';
ts(:,(i/20)+1)=ts1; % save the simulation time
sol(:,(i+1:i+20))=sol1;
% save the values of the solutions of the states
s(:,(i/2+1:i/2+10))=s1.';
% save the values of the error signals
r(:,(i/2+1:i/2+10))=r1.';
% save the values of the performance functions
```

```
gp(:,(i/2+1:i/2+10))=gp1.'; % save the values of the
gv(:,(i/2+1:i/2+10))=gv1.'; % neighborhood errors
end
```

Calculation of parameters $\rho_{i,k}^{\sigma(t)}(0)$ and $l_{i,k}^{\sigma(t)}$, $i = 1, ..., 5$, $k \in \{x, y\}$:

```
function [l,r0]=calc_r0_l(pstart,tstart)
if tstart<pi/5; % the adjancy matix
 Astart=[0 0 0 0 0; % is different at each
     1 0 0 0 0; % switching time
     0 1 0 0 0;
     0 0 1 0 0;
     0 0 0 1 0];
elseif tstart<2*pi/5;
   Astart=[0 1 0 0 0;
       0 0 0 0 0;
       0 1 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
elseif tstart<3*pi/5;
  Astart=[0 0 0 0 0;
      1 0 0 0 0;
      0 1 0 0 0;
      0 0 1 0 0;
      0 0 0 1 0];
elseif tstart<4*pi/5;
   Astart=[0 0 0 0 0;
       0 0 1 0 0;
       0 0 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
elseif tstart<5*pi/5;
    Astart=[0 1 0 0 0;
        0 0 0 0 0;
        0 1 0 0 0;
        0 0 1 0 0;
        1 0 0 0 0];
elseif tstart<6*pi/5;
   Astart=[0 1 0 0 0;
       0 0 0 0 0;
       0 1 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
elseif tstart<7*pi/5;
    Astart=[0 0 0 0 0;
```

```
40          1 0 0 0 0;
41          0 1 0 0 0;
42          0 0 1 0 0;
43          0 0 0 1 0];
44  elseif tstart<8*pi/5;
45     Astart=[0 0 0 0 0;
46          0 0 1 0 0;
47          0 0 0 0 0;
48          0 0 1 0 0;
49          1 0 0 0 0];
50  elseif tstart<9*pi/5;
51     Astart=[0 1 0 0 0;
52          0 0 0 0 0;
53          0 1 0 0 0;
54          0 0 1 0 0;
55          1 0 0 0 0];
56     %{
57  elseif tstart<10*pi/5;
58      Astart=[0 0 0 0 0;
59          1 0 0 0 0;
60          0 1 0 0 0;
61          0 0 1 0 0;
62          0 0 0 1 0];
63  elseif tstart<11*pi/5; % the adjancy matix
64   Astart=[0 0 0 0 0;
65          0 0 1 0 0;
66          0 0 0 0 0;
67          0 0 1 0 0;
68          1 0 0 0 0];
69  elseif tstart<12*pi/5;
70     Astart=[0 1 0 0 0;
71          0 0 0 0 0;
72          0 1 0 0 0;
73          0 0 1 0 0;
74          1 0 0 0 0];
75  elseif tstart<13*pi/5;
76    Astart=[0 0 0 0 0;
77         1 0 0 0 0;
78         0 1 0 0 0;
79         0 0 1 0 0;
80         0 0 0 1 0];
81  elseif tstart<14*pi/5; % the adjancy matix
82     Astart=[0 0 0 0 0;
83          0 0 1 0 0;
84          0 0 0 0 0;
85          0 0 1 0 0;
86          1 0 0 0 0];
87  elseif tstart<15*pi/5;
```

```
88      Astart=[0 1 0 0 0;
89          0 0 0 0 0;
90          0 1 0 0 0;
91          0 0 1 0 0;
92          1 0 0 0 0];
93  elseif tstart<16*pi/5;
94    Astart=[0 0 0 0 0;
95        1 0 0 0 0;
96        0 1 0 0 0;
97        0 0 1 0 0;
98        0 0 0 1 0];
99    elseif tstart<17*pi/5;
100      Astart=[0 0 0 0 0;
101       1 0 0 0 0;
102       0 1 0 0 0;
103       0 0 1 0 0;
104       0 0 0 1 0];
105   elseif tstart<18*pi/5; % the adjancy matix
106    Astart=[0 0 0 0 0;
107         0 0 1 0 0;
108         0 0 0 0 0;
109         0 0 1 0 0;
110         1 0 0 0 0];
111 elseif tstart<19*pi/5;
112    Astart=[0 1 0 0 0;
113         0 0 0 0 0;
114         0 1 0 0 0;
115         0 0 1 0 0;
116         1 0 0 0 0];
117         %}
118 else
119   Astart=[0 0 0 0 0;
120       1 0 0 0 0;
121       0 1 0 0 0;
122       0 0 1 0 0;
123       0 0 0 1 0];
124
125 end
126 %%%%%%%%%%%%%%%%%%
127 diagarraystart=zeros([5 1]);
128 for i=1:5;
129     for k=1:5;
130        diagarraystart(i)=diagarraystart(i)+Astart(i,k);
131     end
132 end
133 Dstart=diag(diagarraystart);
134 %%%%%%%%%%%%%%%%%%%%%%
135 Lstart=Dstart-Astart;
```

```
%%%%%%%%%%%%%%%%%%%%
Bstart=zeros([5 5]);
for i=1:5;
  if Dstart(i,i)==0;
      Bstart(i,i)=1;
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
LBstart=Lstart+Bstart;
if tstart==0;
dposx1=[pstart(1)-3*cos(tstart);pstart(5)-3*cos(tstart);pstart(9)
-3*cos(tstart);pstart(13)-3*cos(tstart);pstart(17)-3*cos(tstart)];
dposy1=[pstart(2)-1.5*sin(tstart);pstart(6)-1.5*sin(tstart);pstart(10)
-1.5*sin(tstart);pstart(14)-1.5*sin(tstart);pstart(18)-1.5*sin(tstart)]
dvelx1=[pstart(3)+3*sin(tstart);pstart(7)+3*sin(tstart);pstart(11)
+3*sin(tstart);pstart(15)+3*sin(tstart);pstart(19)+3*sin(tstart)];
dvely1=[pstart(4)-1.5*cos(tstart);pstart(8)-1.5*cos(tstart);pstart(12)
-1.5*cos(tstart);pstart(16)-1.5*cos(tstart);pstart(20)-1.5*cos(tstart)]
%dposx1=[pstart(1)-0.3;pstart(5)-0.3;pstart(9)-0.3;pstart(13)-0.3;pstar
%dposy1=[pstart(2)-0.2;pstart(6)-0.2;pstart(10)-0.2;pstart(14)-0.2;psta
%dvelx1=[pstart(3)-0;pstart(7)-0;pstart(11)-0;pstart(15)-0;pstart(19)-0
%dvely1=[pstart(4)-0;pstart(8)-0;pstart(12)-0;pstart(16)-0;pstart(20)-0
else
load('sfalmata.mat');
end
epos1start=[LBstart*dposx1;LBstart*dposy1];
evel1start=[LBstart*dvelx1;LBstart*dvely1];
geitpos1start=[epos1start(1);epos1start(6);epos1start(2);
epos1start(7);epos1start(3);epos1start(8);epos1start(4);
epos1start(9);epos1start(5);epos1start(10)];
geitvel1start=[evel1start(1);evel1start(6);evel1start(2);
evel1start(7);evel1start(3);evel1start(8);evel1start(4);
evel1start(9);evel1start(5);evel1start(10)];
lamda=3;
r_apeiro=0.0081;
sigma1=geitvel1start+lamda*geitpos1start;
r0=2*abs(sigma1)+0.2;
a=1.005;
l=-(log(((a-1)*r_apeiro)./(r0-r_apeiro))/(10*pi/5)-tstart);
end
```

Calculation of requested quantities $s_{i,k}^{\sigma(t)}(t)$, $\rho_{i,k}^{\sigma(t)}(t)$ and $e_{i,k}^{\sigma(t)}(t)$, $i = 1, ..., 5$, $k \in \{x, y\}$:

```
function [sigma,rfun,dposx0,dposy0,dvelx0,dvely0]=calc_all(ts1,sol1,l,r
if tstart<pi/5; % the adjancy matix
 A0=[0 0 0 0 0; % is different at each
```

```
      1 0 0 0 0; % switching time
      0 1 0 0 0;
      0 0 1 0 0;
      0 0 0 1 0];
elseif tstart<2*pi/5;
   A0=[0 1 0 0 0;
       0 0 0 0 0;
       0 1 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
elseif tstart<3*pi/5;
  A0=[0 0 0 0 0;
      1 0 0 0 0;
      0 1 0 0 0;
      0 0 1 0 0;
      0 0 0 1 0];
elseif tstart<4*pi/5;
   A0=[0 0 0 0 0;
       0 0 1 0 0;
       0 0 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
elseif tstart<5*pi/5;
    A0=[0 1 0 0 0;
        0 0 0 0 0;
        0 1 0 0 0;
        0 0 1 0 0;
        1 0 0 0 0];
elseif tstart<6*pi/5;
   A0=[0 1 0 0 0;
       0 0 0 0 0;
       0 1 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
elseif tstart<7*pi/5;
    A0=[0 0 0 0 0;
        1 0 0 0 0;
        0 1 0 0 0;
        0 0 1 0 0;
        0 0 0 1 0];
elseif tstart<8*pi/5;
   A0=[0 0 0 0 0;
       0 0 1 0 0;
       0 0 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
elseif tstart<9*pi/5;
   A0=[0 1 0 0 0;
```

```
52          0 0 0 0 0;
53          0 1 0 0 0;
54          0 0 1 0 0;
55          1 0 0 0 0];
56      %{
57  elseif tstart<10*pi/5;
58      A0=[0 0 0 0 0;
59           1 0 0 0 0;
60           0 1 0 0 0;
61           0 0 1 0 0;
62           0 0 0 1 0];
63  elseif tstart<11*pi/5; % the adjancy matix
64   A0=[0 0 0 0 0;
65          0 0 1 0 0;
66          0 0 0 0 0;
67          0 0 1 0 0;
68          1 0 0 0 0];
69  elseif tstart<12*pi/5;
70      A0=[0 1 0 0 0;
71          0 0 0 0 0;
72          0 1 0 0 0;
73          0 0 1 0 0;
74          1 0 0 0 0];
75  elseif tstart<13*pi/5;
76    A0=[0 0 0 0 0;
77         1 0 0 0 0;
78         0 1 0 0 0;
79         0 0 1 0 0;
80         0 0 0 1 0];
81
82  elseif tstart<14*pi/5; % the adjancy matix
83      A0=[0 0 0 0 0;
84          0 0 1 0 0;
85          0 0 0 0 0;
86          0 0 1 0 0;
87          1 0 0 0 0];
88  elseif tstart<15*pi/5;
89      A0=[0 1 0 0 0;
90          0 0 0 0 0;
91          0 1 0 0 0;
92          0 0 1 0 0;
93          1 0 0 0 0];
94  elseif tstart<16*pi/5;
95    A0=[0 0 0 0 0;
96         1 0 0 0 0;
97         0 1 0 0 0;
98         0 0 1 0 0;
99         0 0 0 1 0];
```

```
100   elseif tstart<17*pi/5;
101      A0=[0 0 0 0 0;
102       1 0 0 0 0;
103       0 1 0 0 0;
104       0 0 1 0 0;
105       0 0 0 1 0];
106   elseif tstart<18*pi/5; % the adjancy matix
107    A0=[0 0 0 0 0;
108        0 0 1 0 0;
109        0 0 0 0 0;
110        0 0 1 0 0;
111        1 0 0 0 0];
112 elseif tstart<19*pi/5;
113    A0=[0 1 0 0 0;
114        0 0 0 0 0;
115        0 1 0 0 0;
116        0 0 1 0 0;
117        1 0 0 0 0];
118    %}
119 else
120   A0=[0 0 0 0 0;
121       1 0 0 0 0;
122       0 1 0 0 0;
123       0 0 1 0 0;
124       0 0 0 1 0];
125
126 end
127 %%%%%%%%%%%%%%%%%%
128 diagarray0=zeros([5 1]);
129 for i=1:5;
130     for k=1:5;
131        diagarray0(i)=diagarray0(i)+A0(i,k);
132     end
133 end
134 D0=diag(diagarray0);
135 %%%%%%%%%%%%%%%%%%%%%%
136 L0=D0-A0;
137 %%%%%%%%%%%%%%%%%%%%
138 B0=zeros([5 5]);
139 for i=1:5;
140   if D0(i,i)==0;
141       B0(i,i)=1;
142   end
143 end
144 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
145 LB0=L0+B0;
146 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
147 dposx0=[sol1(:,1)'-3*cos(ts1');sol1(:,5)'-3*cos(ts1');sol1(:,9)'
```

```
148 -3*cos(ts1');sol1(:,13)'-3*cos(ts1');sol1(:,17)'-3*cos(ts1')];
149 dposy0=[sol1(:,2)'-1.5*sin(ts1');sol1(:,6)'-1.5*sin(ts1');sol1(:,10)'
150 -1.5*sin(ts1');sol1(:,14)'-1.5*sin(ts1');sol1(:,18)'-1.5*sin(ts1')];
151 dvelx0=[sol1(:,3)'+3*sin(ts1');sol1(:,7)'+3*sin(ts1');sol1(:,11)'
152 +3*sin(ts1');sol1(:,15)'+3*sin(ts1');sol1(:,19)'+3*sin(ts1')];
153 dvely0=[sol1(:,4)'-1.5*cos(ts1');sol1(:,8)'-1.5*cos(ts1');sol1(:,12)'
154 -1.5*cos(ts1');sol1(:,16)'-1.5*cos(ts1');sol1(:,20)'-1.5*cos(ts1')];
155 %dposx0=[sol1(:,1)'-0.3;sol1(:,5)'-0.3;sol1(:,9)'-0.3;sol1(:,13)'-0.3;s
156 %dposy0=[sol1(:,2)'-0.2;sol1(:,6)'-0.2;sol1(:,10)'-0.2;sol1(:,14)'-0.2;
157 %dvelx0=[sol1(:,3)'-0;sol1(:,7)'-0;sol1(:,11)'-0;sol1(:,15)'-0;sol1(:,1
158 %dvely0=[sol1(:,4)'-0;sol1(:,8)'-0;sol1(:,12)'-0;sol1(:,16)'-0;sol1(:,2
159 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
160 epos0=[LB0*dposx0;LB0*dposy0];
161 evel0=[LB0*dvelx0;LB0*dvely0];
162 geitpos0=[epos0(1,:);epos0(6,:);epos0(2,:);epos0(7,:);epos0(3,:);
163 epos0(8,:);epos0(4,:);epos0(9,:);epos0(5,:);epos0(10,:)];
164 geitvel0=[evel0(1,:);evel0(6,:);evel0(2,:);evel0(7,:);evel0(3,:);
165 evel0(8,:);evel0(4,:);evel0(9,:);evel0(5,:);evel0(10,:)];
166 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
167 lamda=3;
168 sigma=geitvel0+lamda*geitpos0;
169 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
170 r_apeiro=0.0081;
171 rfun=zeros([10 21]);
172 for i=1:21;
173 rfun(:,i)=(r0-r_apeiro).*exp(-l*(ts1(i)-tstart))+r_apeiro;
174 end
175 end
```

Constructing the differential system:

```
1  function  syst=mal1(t1,p1) % function that
2                             % creates the
3                             % differential system
4  load('t_r0_l.mat');
5  A1=alpha2(tstart); % adjency matrix
6  D1=degree1(A1); % degree matrix
7  L1=laplacian1(A1,D1); % laplacian
8  B1=beta1(D1); % matrix that shows the access of the agents
9                % to the leader 's state
10 LB1=betalap1(L1,B1); % L + B
11 [dposx1,dposy1,dvelx1,dvely1]=deltaerror1(p1,t1); % disagreement errors
12 [geitpos1,geitvel1]=neiberror1(LB1,dposx1,dposy1,dvelx1,dvely1);
13                                  % neighborhood errors
14 sigma1=serror1(geitpos1,geitvel1); % error signals
15 rfun1=apodosh1(t1); % performance functions
16 u1=eisodos1(rfun1,sigma1); % control input
```

```
syst=systhma1(p1,u1,t1); % differential system in matrix form
end
```

Functions that "mal1" includes:

```
function A1=alpha2(tstart)
if tstart<pi/5; % the adjancy matix
 A1=[0 0 0 0 0; % is different at each
     1 0 0 0 0; % switching time
     0 1 0 0 0;
     0 0 1 0 0;
     0 0 0 1 0];
elseif tstart<2*pi/5;
   A1=[0 1 0 0 0;
       0 0 0 0 0;
       0 1 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
elseif tstart<3*pi/5;
  A1=[0 0 0 0 0;
      1 0 0 0 0;
      0 1 0 0 0;
      0 0 1 0 0;
      0 0 0 1 0];
elseif tstart<4*pi/5;
   A1=[0 0 0 0 0;
       0 0 1 0 0;
       0 0 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
elseif tstart<5*pi/5;
    A1=[0 1 0 0 0;
        0 0 0 0 0;
        0 1 0 0 0;
        0 0 1 0 0;
        1 0 0 0 0];
elseif tstart<6*pi/5;
   A1=[0 1 0 0 0;
       0 0 0 0 0;
       0 1 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
elseif tstart<7*pi/5;
    A1=[0 0 0 0 0;
        1 0 0 0 0;
        0 1 0 0 0;
        0 0 1 0 0;
```

```
         0 0 0 1 0];
elseif tstart<8*pi/5;
   A1=[0 0 0 0 0;
       0 0 1 0 0;
       0 0 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
elseif tstart<9*pi/5;
   A1=[0 1 0 0 0;
       0 0 0 0 0;
       0 1 0 0 0;
       0 0 1 0 0;
       1 0 0 0 0];
else
    A1=[0 0 0 0 0;
        1 0 0 0 0;
        0 1 0 0 0;
        0 0 1 0 0;
        0 0 0 1 0];
end
end
```

```
function D1=degree1(A1)
diagarray=zeros([5 1]);
for i=1:5;
    for k=1:5;
       diagarray(i)=diagarray(i)+A1(i,k);
    end
end
D1=diag(diagarray);
end
```

```
function L1=laplacian1(A1,D1)
L1=D1-A1;
end
```

```
function B1=beta1(D1)
B1=zeros([5 5]);
for i=1:5;
  if D1(i,i)==0;
      B1(i,i)=1;
  end
end
end
```

```
function LB1=betalap1(L1,B1)
LB1=L1+B1;
end
```

```
function [dposx1,dposy1,dvelx1,dvely1]=deltaerror1(p1,t1)
dposx1=[p1(1)-3*cos(t1);p1(5)-3*cos(t1);p1(9)-3*cos(t1);
p1(13)-3*cos(t1);p1(17)-3*cos(t1)];
dposy1=[p1(2)-1.5*sin(t1);p1(6)-1.5*sin(t1);p1(10)-1.5*sin(t1);
p1(14)-1.5*sin(t1);p1(18)-1.5*sin(t1)];
dvelx1=[p1(3)+3*sin(t1);p1(7)+3*sin(t1);p1(11)+3*sin(t1);
p1(15)+3*sin(t1);p1(19)+3*sin(t1)];
dvely1=[p1(4)-1.5*cos(t1);p1(8)-1.5*cos(t1);p1(12)-1.5*cos(t1);
p1(16)-1.5*cos(t1);p1(20)-1.5*cos(t1)];
%dposx1=[p1(1)-0.3;p1(5)-0.3;p1(9)-0.3;p1(13)-0.3;p1(17)-0.3];
%dposy1=[p1(2)-0.2;p1(6)-0.2;p1(10)-0.2;p1(14)-0.2;p1(18)-0.2];
%dvelx1=[p1(3)-0;p1(7)-0;p1(11)-0;p1(15)-0;p1(19)-0];
%dvely1=[p1(4)-0;p1(8)-0;p1(12)-0;p1(16)-0;p1(20)-0];
save('sfalmata.mat','dposx1','dposy1','dvelx1','dvely1');
end
```

```
function [geitpos1,geitvel1]=neiberror1(LB1,dposx1,dposy1,
dvelx1,dvely1)
epos1=[LB1*dposx1;LB1*dposy1];
evel1=[LB1*dvelx1;LB1*dvely1];
geitpos1=[epos1(1);epos1(6);epos1(2);epos1(7);
epos1(3);epos1(8);epos1(4);epos1(9);epos1(5);epos1(10)];
geitvel1=[evel1(1);evel1(6);evel1(2);evel1(7);
evel1(3);evel1(8);evel1(4);evel1(9);evel1(5);evel1(10)];
end
```

```
function sigma1=serror1(geitpos1,geitvel1)
lamda=3;
sigma1=geitvel1+lamda*geitpos1;
end
```

```
function rfun1=apodosh1(t1)
r_apeiro=0.0081;
load('t_r0_l.mat');
rfun1=(r0-r_apeiro).*exp(-l*(t1-tstart))+r_apeiro;
end
```

```
function u1=eisodos1(rfun1,sigma1)
```

```
kapa=0.5;
one=-kapa./(2.*rfun1);
two=(1+(sigma1./rfun1));
three=(1-sigma1./rfun1);
u=real(one.*(log(two./three))./((two).*(three)));
save('input.mat','u');
u1=[0;0;u(1);u(2);0;0;u(3);u(4);0;0;u(5);u(6);
    0;0;u(7);u(8);0;0;u(9);u(10)];
end
```

```
function syst=systhma1(p1,u1,t1)
z4x4=zeros([4 4]);
fixx=1;
fixy=2;
gix=1.2;
giy=0.3;
dix=0.1;
diy=0.1;
subf1=[0 0 1 0;0 0 0 1;0 0 -fixx*p1(4) -fixy*(p1(3)+p1(4));
    0 0 fixy*p1(3) 0];
subf2=[0 0 1 0;0 0 0 1;0 0 -fixx*p1(8) -fixy*(p1(7)+p1(8));
    0 0 fixy*p1(7) 0];
subf3=[0 0 1 0;0 0 0 1;0 0 -fixx*p1(12) -fixy*(p1(11)+p1(12));
    0 0 fixy*p1(11) 0];
subf4=[0 0 1 0;0 0 0 1;0 0 -fixx*p1(16) -fixy*(p1(15)+p1(16));
    0 0 fixy*p1(15) 0];
subf5=[0 0 1 0;0 0 0 1;0 0 -fixx*p1(20) -fixy*(p1(19)+p1(20));
    0 0 fixy*p1(19) 0];
F=[subf1 z4x4 z4x4 z4x4 z4x4;z4x4 subf2 z4x4 z4x4 z4x4;
    z4x4 z4x4 subf3 z4x4 z4x4;z4x4 z4x4 z4x4 subf4 z4x4;
    z4x4 z4x4 z4x4 z4x4 subf5];
subu=[0 0 0 0;0 0 0 0;0 0 gix 0;0 0 0 giy];
G=[subu z4x4 z4x4 z4x4 z4x4;z4x4 subu z4x4 z4x4 z4x4;
    z4x4 z4x4 subu z4x4 z4x4;z4x4 z4x4 z4x4 subu z4x4;
    z4x4 z4x4 z4x4 z4x4 subu];
D=[0;0;dix*sin(3*t1);diy*cos(4*t1);0;0;dix*sin(3*t1);
    diy*cos(4*t1);0;0;dix*sin(3*t1);diy*cos(4*t1);
    0;0;dix*sin(3*t1);diy*cos(4*t1);0;0;dix*sin(3*t1);diy*cos(4*t1)];
subs1=F*p1;
subs2=G*u1;
syst=real(subs1+subs2+D);
end
```

Reference trajectory (leader node):

```
function tr1=troxia(t1)
```

```
tr1=[3*cos(t1);1.5*sin(t1);-3*sin(t1);1.5*cos(t1)]; % ellipse
end
```

Plots of requested quantities: Trace of the agents in the workspace, error metrics along with the imposed performance bounds, evolution of the position and velocity states and required control input signals

---

```
traj1=troxia([0:0.01*pi/5:10*pi/5]); %
plot(traj1(1,:),traj1(2,:),'--')     %
hold on                              %  plot of the traces of all
for k=0:4:16;                        %  agents and plot of the
for i=1+k:20:181+k;                  %  reference trajectory
plot(sol(:,i),sol(:,i+1))            %
hold on                              %
end                                  %
end                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Evolution of the position and velocity states %%%%%%%
%%%%%%%% of the 1st agent%%%%%%%%
subplot(2,2,1)
plot([0:0.01*pi/5:10*pi/5],traj1(1,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+1))
hold on
end
subplot(2,2,2)
plot([0:0.01*pi/5:10*pi/5],traj1(2,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+2))
hold on
end
subplot(2,2,3)
plot([0:0.01*pi/5:10*pi/5],traj1(3,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+3))
hold on
end
subplot(2,2,4)
plot([0:0.01*pi/5:10*pi/5],traj1(4,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+4))
hold on
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Evolution of the position and velocity states %%%%%%%
%%%%%%% of the 2nd agent%%%%%%%%
subplot(2,2,1)
plot([0:0.01*pi/5:10*pi/5],traj1(1,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+5))
hold on
end
subplot(2,2,2)
plot([0:0.01*pi/5:10*pi/5],traj1(2,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+6))
hold on
end
subplot(2,2,3)
plot([0:0.01*pi/5:10*pi/5],traj1(3,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+7))
hold on
end
subplot(2,2,4)
plot([0:0.01*pi/5:10*pi/5],traj1(4,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+8))
hold on
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Evolution of the position and velocity states %%%%%%%%
%%%%%%%%%% of the 3rd agent%%%%%%%%
subplot(2,2,1)
plot([0:0.01*pi/5:10*pi/5],traj1(1,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+9))
hold on
end
subplot(2,2,2)
plot([0:0.01*pi/5:10*pi/5],traj1(2,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+10))
hold on
end
```

```
subplot(2,2,3)
plot([0:0.01*pi/5:10*pi/5],traj1(3,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+11))
hold on
end
subplot(2,2,4)
plot([0:0.01*pi/5:10*pi/5],traj1(4,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+12))
hold on
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Evolution of the position and velocity states %%%%%%
%%%%%% of the 4th agent%%%%%%%%
subplot(2,2,1)
plot([0:0.01*pi/5:10*pi/5],traj1(1,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+13))
hold on
end
subplot(2,2,2)
plot([0:0.01*pi/5:10*pi/5],traj1(2,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+14))
hold on
end
subplot(2,2,3)
plot([0:0.01*pi/5:10*pi/5],traj1(3,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+15))
hold on
end
subplot(2,2,4)
plot([0:0.01*pi/5:10*pi/5],traj1(4,:),'--')
hold on
for i=0:9;
plot(ts(:,i+1),sol(:,i*20+16))
hold on
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Evolution of the position and velocity states %%%%%%%%
%%%%%%% of the 5th agent%%%%%%%%
```

```
137 subplot(2,2,1)
138 plot([0:0.01*pi/5:10*pi/5],traj1(1,:),'--')
139 hold on
140 for i=0:9;
141 plot(ts(:,i+1),sol(:,i*20+17))
142 hold on
143 end
144 subplot(2,2,2)
145 plot([0:0.01*pi/5:10*pi/5],traj1(2,:),'--')
146 hold on
147 for i=0:9;
148 plot(ts(:,i+1),sol(:,i*20+18))
149 hold on
150 end
151 subplot(2,2,3)
152 plot([0:0.01*pi/5:10*pi/5],traj1(3,:),'--')
153 hold on
154 for i=0:9;
155 plot(ts(:,i+1),sol(:,i*20+19))
156 hold on
157 end
158 subplot(2,2,4)
159 plot([0:0.01*pi/5:10*pi/5],traj1(4,:),'--')
160 hold on
161 for i=0:9;
162 plot(ts(:,i+1),sol(:,i*20+20))
163 hold on
164 end
165 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
166 %%%%%%%%%%%%%%Evolution of the error metrics %%%%%%%
167 %%%%%%% of the 1st agent%%%%%%%%%%%%
168 subplot(2,1,1)
169 for i=0:9;
170 plot(ts(:,i+1),abs(s(:,i*10+1)),'--')
171 hold on
172 plot(ts(:,i+1),r(:,i*10+1))
173 hold on
174 end
175 subplot(2,1,2)
176 for i=0:9;
177 plot(ts(:,i+1),abs(s(:,i*10+2)),'--')
178 hold on
179 plot(ts(:,i+1),r(:,i*10+2))
180 hold on
181 end
182 %%%%%%%%%%%%%%Evolution of the error metrics %%%%%%%%%%
183 %%%%%% of the 2nd agent%%%%%%%%%%%%
184 subplot(2,1,1)
```

```
for i=0:9;
plot(ts(:,i+1),abs(s(:,i*10+3)),'--')
hold on
plot(ts(:,i+1),r(:,i*10+3))
hold on
end
subplot(2,1,2)
for i=0:9;
plot(ts(:,i+1),abs(s(:,i*10+4)),'--')
hold on
plot(ts(:,i+1),r(:,i*10+4))
hold on
end
%%%%%%%%%%%%%%Evolution of the error metrics %%%%%%%%%%%%
%%%%%%%% of the 3rd agent%%%%%%%%%%%%%
subplot(2,1,1)
for i=0:9;
plot(ts(:,i+1),abs(s(:,i*10+5)),'--')
hold on
plot(ts(:,i+1),r(:,i*10+5))
hold on
end
subplot(2,1,2)
for i=0:9;
plot(ts(:,i+1),abs(s(:,i*10+6)),'--')
hold on
plot(ts(:,i+1),r(:,i*10+6))
hold on
end
%%%%%%%%%%%%%%Evolution of the error metrics %%%%%%%%%
%%%%%%%%% of the 4th agent%%%%%%%%%%%%%
subplot(2,1,1)
for i=0:9;
plot(ts(:,i+1),abs(s(:,i*10+7)),'--')
hold on
plot(ts(:,i+1),r(:,i*10+7))
hold on
end
subplot(2,1,2)
for i=0:9;
plot(ts(:,i+1),abs(s(:,i*10+8)),'--')
hold on
plot(ts(:,i+1),r(:,i*10+8))
hold on
end
%%%%%%%%%%%%%%Evolution of the error metrics %%%%%%%%%%%%
%%%%%%%% of the 5th agent%%%%%%%%%%%%%
subplot(2,1,1)
```

```
for i=0:9;
plot(ts(:,i+1),abs(s(:,i*10+9)),'--')
hold on
plot(ts(:,i+1),r(:,i*10+9))
hold on
end
subplot(2,1,2)
for i=0:9;
plot(ts(:,i+1),abs(s(:,i*10+10)),'--')
hold on
plot(ts(:,i+1),r(:,i*10+10))
hold on
end
%%%%calculation of input signals%%%%%%%%%%
inp=zeros([21 100]);
for k=1:100
for i=1:21
inp(i,k)=-(0.5/(2*r(i,k)))*((log((1+(s(i,k)/r(i,k)))/
(1-(s(i,k)/r(i,k)))))/((1+(s(i,k))/(r(i,k)))*(1-((s(i,k))/(r(i,k))))));
end
end
%%%%%%%%%%%%%plot of the control input signal of the 1st_agent%%%%%%%%
subplot(2,1,1)
for i=0:9
plot(ts(:,i+1),inp(:,i*10+1))
hold on
end
subplot(2,1,2)
for i=0:9
plot(ts(:,i+1),inp(:,i*10+2))
hold on
end
%%%%%%%%%%%%%plot of the control input signal of the 2nd agent%%%%%%%%
subplot(2,1,1)
for i=0:9
plot(ts(:,i+1),inp(:,i*10+3))
hold on
end
subplot(2,1,2)
for i=0:9
plot(ts(:,i+1),inp(:,i*10+4))
hold on
end
%%%%%%%%%%%%%plot of the control input signal of the 3rd agent%%%%%%%%
subplot(2,1,1)
for i=0:9
plot(ts(:,i+1),inp(:,i*10+5))
hold on
```

```
281 end
282 subplot(2,1,2)
283 for i=0:9
284 plot(ts(:,i+1),inp(:,i*10+6))
285 hold on
286 end
287 %%%%%%%%%%%%plot of the control input signal of the 4th agent%%%%%%%%
288 subplot(2,1,1)
289 for i=0:9
290 plot(ts(:,i+1),inp(:,i*10+7))
291 hold on
292 end
293 subplot(2,1,2)
294 for i=0:9
295 plot(ts(:,i+1),inp(:,i*10+8))
296 hold on
297 end
298 %%%%%%%%%%%%plot of the control input signal of the 5th agent%%%%%%%%
299 subplot(2,1,1)
300 for i=0:9
301 plot(ts(:,i+1),inp(:,i*10+9))
302 hold on
303 end
304 subplot(2,1,2)
305 for i=0:9
306 plot(ts(:,i+1),inp(:,i*10+10))
307 hold on
308 end
309 %%%%%%%%Bounds of the disagreement position error %%%%%%%%%%%%%
310 up_bound_x=zeros([1 21]);
311 down_bound_x=zeros([1 21]);
312 for i=1:21
313     up_bound(i)=0.022;
314     down_bound(i)=-0.022;
315 end
316 %%%%%%%%%Bounds of the disagreement velocity error %%%%%%%%%
317 up_bound=zeros([1 21]);
318 down_bound=zeros([1 21]);
319 for i=1:21
320     up_bound(i)=0.13;
321     down_bound(i)=-0.13;
322 end
323 %%%%%Evolution of the disagreement position error in x dimension%%%%%%%
324 for k=0:5:45
325     plot(ts(:,k/5+1),up_bound,'--')
326     hold on
327     plot(ts(:,k/5+1),down_bound,'--')
328     hold on
```

```
329      plot(ts(:,k/5+1),dpx(:,k+1))
330      hold on
331      plot(ts(:,k/5+1),dpx(:,k+2))
332      hold on
333      plot(ts(:,k/5+1),dpx(:,k+3))
334      hold on
335      plot(ts(:,k/5+1),dpx(:,k+4))
336      hold on
337      plot(ts(:,k/5+1),dpx(:,k+5))
338  end
339  %%%%%Evolution of the disagreement position error in y dimension%%%%
340  for k=0:5:45
341      plot(ts(:,k/5+1),up_bound,'--')
342      hold on
343      plot(ts(:,k/5+1),down_bound,'--')
344      hold on
345      plot(ts(:,k/5+1),dpy(:,k+1))
346      hold on
347      plot(ts(:,k/5+1),dpy(:,k+2))
348      hold on
349      plot(ts(:,k/5+1),dpy(:,k+3))
350      hold on
351      plot(ts(:,k/5+1),dpy(:,k+4))
352      hold on
353      plot(ts(:,k/5+1),dpy(:,k+5))
354  end
355  %%%Evolution of the disagreement velocity error in x dimension%%%%
356  for k=0:5:45
357      plot(ts(:,k/5+1),up_bound,'--')
358      hold on
359      plot(ts(:,k/5+1),down_bound,'--')
360      hold on
361      plot(ts(:,k/5+1),dvx(:,k+1))
362      hold on
363      plot(ts(:,k/5+1),dvx(:,k+2))
364      hold on
365      plot(ts(:,k/5+1),dvx(:,k+3))
366      hold on
367      plot(ts(:,k/5+1),dvx(:,k+4))
368      hold on
369      plot(ts(:,k/5+1),dvx(:,k+5))
370  end
371  %%%%%Evolution of the disagreement velocity error in y dimension%%%%
372  for k=0:5:45
373      plot(ts(:,k/5+1),up_bound,'--')
374      hold on
375      plot(ts(:,k/5+1),down_bound,'--')
376      hold on
```

```
377     plot(ts(:,k/5+1),dvy(:,k+1))
378     hold on
379     plot(ts(:,k/5+1),dvy(:,k+2))
380     hold on
381     plot(ts(:,k/5+1),dvy(:,k+3))
382     hold on
383     plot(ts(:,k/5+1),dvy(:,k+4))
384     hold on
385     plot(ts(:,k/5+1),dvy(:,k+5))
386 end
```

Main m-file for system simulation - Consensus with protocol in [6]:

```
1  p=[0.30;0.20;0;0;0.4;0.3;0;0;0.38;0.28;0;0;0.22;0.12;0;0;0.2;0.1;0;0];
2  options=odeset('reltol',1e-6,'abstol',1e-9);
3  ts=zeros([21 10]);
4  sol=zeros([21 200]);
5  for i=0:20:180;
6  t1=(i/20)*pi/5:0.05*pi/5:((i+20)/20)*pi/5;
7  tstart=t1(1);
8  save('t_.mat','tstart');
9  ls1=@allfun;
10 [ts1,sol1]=ode15s(ls1,t1,p,options);
11 p=sol1(21,:)';
12 ts(:,(i/20)+1)=ts1;
13 sol(:,(i+1:i+20))=sol1;
14 end
```

Constructing the differential system:

```
1  function  syst=allfun(t1,p)
2  load('t_.mat');
3  A=alpha(tstart);
4  %D1=degree1(A1);
5  %L1=laplacian1(A1,D1);
6  %B1=beta1(D1);
7  [dposx,dposy,dvelx,dvely]=delta(A,p);
8  u=control_input(dposx,dposy,dvelx,dvely);
9  syst=systhma(p,u,t1);
10 end
```

Functions that "allfun" includes:

```
1  function A=alpha(tstart)
2  if tstart<pi/5;
```

```
3       A=[0 1 0 0 0;
4          0 0 1 0 0;
5          0 0 0 1 0;
6          0 0 0 0 1;
7          1 0 0 0 0];
8   elseif tstart<2*pi/5;
9       A=[0 0 0 1 0;
10         0 0 1 0 0;
11         0 0 0 0 1;
12         0 1 0 0 0;
13         1 0 0 0 0];
14    elseif tstart<3*pi/5;
15      A=[0 1 0 0 0;
16         0 0 1 0 0;
17         0 0 0 1 0;
18         0 0 0 0 1;
19         1 0 0 0 0];
20    elseif tstart<4*pi/5;
21      A=[0 0 0 0 1;
22         1 0 0 0 0;
23         0 0 0 1 0;
24         0 1 0 0 0;
25         0 0 1 0 0];
26    elseif tstart<5*pi/5;
27      A=[0 0 0 1 0;
28         0 0 1 0 0;
29         0 0 0 0 1;
30         0 1 0 0 0;
31         1 0 0 0 0];
32    elseif tstart<6*pi/5;
33      A=[0 0 0 1 0;
34         0 0 1 0 0;
35         0 0 0 0 1;
36         0 1 0 0 0;
37         1 0 0 0 0];
38    elseif tstart<7*pi/5;
39      A=[0 1 0 0 0;
40         0 0 1 0 0;
41         0 0 0 1 0;
42         0 0 0 0 1;
43         1 0 0 0 0];
44     elseif tstart<8*pi/5;
45      A=[0 0 0 0 1;
46         1 0 0 0 0;
47         0 0 0 1 0;
48         0 1 0 0 0;
49         0 0 1 0 0];
50    elseif tstart<9*pi/5;
```

```
    A=[0 1 0 0 1;
       0 0 1 0 0;
       1 0 0 1 0;
       1 0 0 0 0;
       0 0 1 0 0];
else
    A=[0 1 0 0 0;
       0 0 1 0 0;
       0 0 0 1 0;
       0 0 0 0 1;
       1 0 0 0 0];
end
end
```

```
function [dposx,dposy,dvelx,dvely]=deltaerror(A,p)
px1=[p(1)-p(1);p(1)-p(5);p(1)-p(9);p(1)-p(13);p(1)-p(17)];
px2=[p(5)-p(1);p(5)-p(5);p(5)-p(9);p(5)-p(13);p(5)-p(17)];
px3=[p(9)-p(1);p(9)-p(5);p(9)-p(9);p(9)-p(13);p(9)-p(17)];
px4=[p(13)-p(1);p(13)-p(5);p(13)-p(9);p(13)-p(13);p(13)-p(17)];
px5=[p(17)-p(1);p(17)-p(5);p(17)-p(9);p(17)-p(13);p(17)-p(17)];

py1=[p(2)-p(2);p(2)-p(6);p(2)-p(10);p(2)-p(14);p(2)-p(18)];
py2=[p(6)-p(2);p(6)-p(6);p(6)-p(10);p(6)-p(14);p(6)-p(18)];
py3=[p(10)-p(2);p(10)-p(6);p(10)-p(10);p(10)-p(14);p(10)-p(18)];
py4=[p(14)-p(2);p(14)-p(6);p(14)-p(10);p(14)-p(14);p(14)-p(18)];
py5=[p(18)-p(2);p(18)-p(6);p(18)-p(10);p(18)-p(14);p(18)-p(18)];

vx1=[p(3)-p(3);p(3)-p(7);p(3)-p(11);p(3)-p(15);p(3)-p(19)];
vx2=[p(7)-p(3);p(7)-p(7);p(7)-p(11);p(7)-p(15);p(7)-p(19)];
vx3=[p(11)-p(3);p(11)-p(7);p(11)-p(11);p(11)-p(15);p(11)-p(19)];
vx4=[p(15)-p(3);p(15)-p(7);p(15)-p(11);p(15)-p(15);p(15)-p(19)];
vx5=[p(19)-p(3);p(19)-p(7);p(19)-p(11);p(19)-p(15);p(19)-p(19)];

vy1=[p(4)-p(4);p(4)-p(8);p(4)-p(12);p(4)-p(16);p(4)-p(20)];
vy2=[p(8)-p(4);p(8)-p(8);p(8)-p(12);p(8)-p(16);p(8)-p(20)];
vy3=[p(12)-p(4);p(12)-p(8);p(12)-p(12);p(12)-p(16);p(12)-p(20)];
vy4=[p(16)-p(4);p(16)-p(8);p(16)-p(12);p(16)-p(16);p(16)-p(20)];
vy5=[p(20)-p(4);p(20)-p(8);p(20)-p(12);p(20)-p(16);p(20)-p(20)];

a1=[A(1,1) A(1,2) A(1,3) A(1,4) A(1,5)];
a2=[A(2,1) A(2,2) A(2,3) A(2,4) A(2,5)];
a3=[A(3,1) A(3,2) A(3,3) A(3,4) A(3,5)];
a4=[A(4,1) A(4,2) A(4,3) A(4,4) A(4,5)];
a5=[A(5,1) A(5,2) A(5,3) A(5,4) A(5,5)];

dposx=[a1*px1;a2*px2;a3*px3;a4*px4;a5*px5];
dposy=[a1*py1;a2*py2;a3*py3;a4*py4;a5*py5];
```

```
dvelx=[a1*vx1;a2*vx2;a3*vx3;a4*vx4;a5*vx5];
dvely=[a1*vy1;a2*vy2;a3*vy3;a4*vy4;a5*vy5];

end
```

```
function u=control_input(dposx,dposy,dvelx,dvely)
k1=5;
k2=5;
ux=([5 1]);
uy=([5 1]);
for i=1:5
ux(i)=-k1*dposx(i)-k2*dvelx(i);
uy(i)=-k1*dposy(i)-k2*dvely(i);
end
u=[ux(1);uy(1);ux(2);uy(2);ux(3);uy(3);ux(4);uy(4);ux(5);uy(5)];
end
```

```
function syst=systhma(p,u,t1)
z4x4=zeros([4 4]);
fixx=1;
fixy=2;
gix=1.2;
giy=0.3;
dix=0.1;
diy=0.1;
g=[gix 0;0 giy];
f1=[-fixx*p(4) -fixy*(p(3)+p(4));fixy*p(3) 0];
f2=[-fixx*p(8) -fixy*(p(7)+p(8));fixy*p(7) 0];
f3=[-fixx*p(12) -fixy*(p(11)+p(12));fixy*p(11) 0];
f4=[-fixx*p(16) -fixy*(p(15)+p(16));fixy*p(15) 0];
f5=[-fixx*p(20) -fixy*(p(19)+p(20));fixy*p(19) 0];
v1=inv(g)*(-f1*[p(3);p(4)]+[u(1);u(2)]);
v2=inv(g)*(-f2*[p(7);p(8)]+[u(3);u(4)]);
v3=inv(g)*(-f3*[p(11);p(12)]+[u(5);u(6)]);
v4=inv(g)*(-f4*[p(15);p(16)]+[u(7);u(8)]);
v5=inv(g)*(-f5*[p(19);p(20)]+[u(9);u(10)]);
v=[0;0;v1;0;0;v2;0;0;v3;0;0;v4;0;0;v5];
subf1=[0 0 1 0;0 0 0 1;0 0 -fixx*p(4) -fixy*(p(3)+p(4));
    0 0 fixy*p(3) 0];
subf2=[0 0 1 0;0 0 0 1;0 0 -fixx*p(8) -fixy*(p(7)+p(8));
    0 0 fixy*p(7) 0];
subf3=[0 0 1 0;0 0 0 1;0 0 -fixx*p(12) -fixy*(p(11)+p(12));
    0 0 fixy*p(11) 0];
subf4=[0 0 1 0;0 0 0 1;0 0 -fixx*p(16) -fixy*(p(15)+p(16));
    0 0 fixy*p(15) 0];
subf5=[0 0 1 0;0 0 0 1;0 0 -fixx*p(20) -fixy*(p(19)+p(20));
```

```
    0 0 fixy*p(19) 0];
F=[subf1 z4x4 z4x4 z4x4 z4x4;z4x4 subf2 z4x4 z4x4 z4x4;
    z4x4 z4x4 subf3 z4x4 z4x4;z4x4 z4x4 z4x4 subf4 z4x4;
    z4x4 z4x4 z4x4 z4x4 subf5];
subu=[0 0 0 0;0 0 0 0;0 0 gix 0;0 0 0 giy];
G=[subu z4x4 z4x4 z4x4 z4x4;z4x4 subu z4x4 z4x4 z4x4;
    z4x4 z4x4 subu z4x4 z4x4;z4x4 z4x4 z4x4 subu z4x4;
    z4x4 z4x4 z4x4 z4x4 subu];
D=[0;0;dix*sin(3*t1);diy*cos(4*t1);0;0;dix*sin(3*t1);diy*cos(4*t1);
    0;0;dix*sin(3*t1);diy*cos(4*t1);0;0;dix*sin(3*t1);diy*cos(4*t1);
    0;0;dix*sin(3*t1);diy*cos(4*t1)];
subs1=F*p;
subs2=G*v;
syst=real(subs1+subs2+1.*D);
end
```

Evolution of the position and velocity states plots of the network agents

```
%%consensus_x_position%%%%%%%%%%%%
subplot(2,2,1)
for i=1:4:17
    for j=0:9
        plot(ts(:,j+1),sol(:,20*j+i))
        hold on
    end
    hold on
end
%%consensus_y_position%%%%%%%%%%%%
subplot(2,2,2)
for i=2:4:18
    for j=0:9
        plot(ts(:,j+1),sol(:,20*j+i))
        hold on
    end
    hold on
end
%%consensus_x_velocity%%%%%%%%%%%%
subplot(2,2,3)
for i=3:4:19
    for j=0:9
        plot(ts(:,j+1),sol(:,20*j+i))
        hold on
    end
    hold on
end
%%%%consensus_y_velocity%%%%%
subplot(2,2,4)
```

```
30 for i=4:4:20
31     for j=0:9
32         plot(ts(:,j+1),sol(:,20*j+i))
33         hold on
34     end
35     hold on
36 end
```