UNIVERSITY OF SOUTHAMPTON

Distributed Iterative Learning Control for Networked Dynamical Systems

by

Bin Chen

A thesis submitted in for the
degree of Doctor of Philosophy

in the
Faculty of Engineering and Physical Science
School of Electronics and Computer Science

May 2022

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCE
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Bin Chen

Networked dynamical systems have found increasingly more applications during the last few decades, thanks to the significant reduction in the cost of sensing, computing and actuating technologies. Among them, there exists a class of networked dynamical systems working in a repetitive manner and requiring high control performance. As an example, next generation advanced manufacturing contains a large number of subsystems working together to perform a variety of manufacturing tasks repeatedly with high performance requirements. For such systems, traditional control methods have significant difficulties meeting the high performance requirements: centralised design does not scale well, while distributed methods mainly focus on asymptotic behaviour. In addition, they all require a highly accurate model which can be difficult/expensive to obtain in practice.

Recently, iterative learning control (ILC), which 'learns' from the input and error information of the previous attempts of the same task without requiring an accurate model to generate the input, has been proposed as an alternative solution. However, most of the existing ILC design for networked dynamical systems have poor scalability, limited convergence performance, and also lack of the ability to deal with system constraints and more general task, e.g., point-to-point (P2P) task. To address these limitations, this thesis proposes novel distributed/decentralised optimisation-based ILC design methods.

This thesis considers three design problems for networked dynamical systems, i.e., consensus tracking, formation control and collaborative tracking. We propose optimisation based ILC design methods using the idea of norm optimal ILC, and the proposed ILC methods show appealing convergence properties and certain degree of robustness to model uncertainties. Using the alternating direction method of multipliers (ADMM), all the designs can be implemented in the distributed/decentralised manner such that only local information is needed, allowing the proposed algorithms to be applied to large scale networked dynamical systems and have great scalability for dynamically growing network. These algorithms can also be extended to solve two unexplored problems in ILC design for networked dynamical systems, namely, constraint handling and P2P task. Numerical examples are given to illustrate the performance of the proposed algorithms.

Contents

List of Figures

Nomenclature

ADMM	The alternating direction method of multiplies
ILC	Iterative learning control
LQT	Linear quadratic tracking
LTI	Linear time invariant
MIMO	Multiple input multiple output
NOILC	Norm optimal iterative learning control
P2P	Point-to-point
QP	Quadratic programming
QCQP	Quadratically constrained quadratic program
SOCP	Second order cone program
SISO	Single input single output
$d(t)$	Response for initial state condition
$e(t)$	Error signal
i	Index of subsystems
k	ILC trial number
k^*	Relative degree of subsystem
p	Amount of subsystem in the networked dynamical systems
q	ADMM iteration number
$r(t)$	Reference signal
t	Time index
$x(t)$	State
x_0	Initial condition
$u(t)$	Input signal
$y(t)$	Output signal
z	Global variable
γ	Dual variable
μ	Scale dual variable
ξ	Feedforward gain matrix
ρ	Penalty parameter
ϕ	ADMM convergence factor
A, B, C, D	State space matrix
G	System matrix

G^*	Hilbert-adjoint operator
I_n	Identity matrix with $n \times n$ dimension
J	Performance index
$K(t)$	Riccati feedback gain matrix
L	Laplacian matrix
N	Time length
Q	Weighting matrix of output space
R	Weighting matrix of input space
T_s	Sampling time
U	Multiplicative uncertainty
Ω	Constraint set
$\mathcal{R}(\mathbb{A})$	Range space of a matrix $\mathbb{A}$
$\mathcal{N}(\mathbb{A})$	Null space of a matrix $\mathbb{A}$
$\mathcal{U}$	Input vector space
$\mathcal{Y}$	Output vector space
$\lvert \cdot \rvert$	Modulus
$\lVert \cdot \rVert$	Norm
$\langle x,\ y \rangle$	Inner product of x and y in Hilbert space
$\{\cdot\}_{k=\cdots}$	Sequence
$d(u, v)$	Distance from u to v
$Ker\{G\}$	Kernal of G
$Range\{G\}$	Range of G
$\mathcal{G}$	Graph
$\mathcal{E}$	Edge in an graph
$\mathcal{V}$	Nodes in an graph
$\mathcal{N}_i$	Neighbour's set
$\mathbb{A}^T$	Transpose of a matrix $\mathbb{A}$
$\mathbb{R}^n$	Set of n dimensional real vectors
$\mathbb{R}^{n \times m}$	Set of $n \times m$ real matrices
$\mathbb{X} \otimes \mathbb{Y}$	Kronecker product of two matrices $\mathbb{X}$ and $\mathbb{Y}$

Declaration of Authorship

I, Bin Chen , declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- Parts of this work have been published as: Chen and Chu (2019c,a,b, 2021, 2022); Chen et al. (2020).

Signed: ....

Date: ....

Acknowledgements

My deepest gratitude goes first and foremost to my first supervisor, Dr. Bing Chu. Many thanks for his pertinent suggestions in weekly meeting, which inspires me a lot in the research. Many thanks to his meticulous guidance and constant encouragement throughout my PhD. study. Also, I would like to show my appreciation to my second supervisor, Dr. Srinandan Dasmahapatra, for his suggestions in that period.

I extend my gratitude to my colleagues and friends in the University of Southampton for their continuous help and friendship. Special gratitude go to every members in the control reading group. The weekly learning seminar let me learn a lot and have a meaningful PhD. journey.

Finally, I am most indebted to my beloved parents and dear brother in China. Without their supports and encouragements, I cannot pursue my PhD. study abroad.

To my dear family

Chapter 1

Introduction

This chapter gives an overview of this thesis. In Section 1.1, we review the background of the networked dynamical systems working in a repetitive manner and requiring high control performance. Then, Section 1.2 gives the motivation of this thesis and finally, Section 1.3 summaries contributions and the organisation of this thesis.

1.1 Background

Networked dynamical systems or multi agents systems refer to a group of subsystems (agents) working together to perform a variety of tasks in a network, which was first introduced by Minski (1986). Networked dynamical systems were unappreciated in the early days because of the high cost in sensing, computing and actuating technologies. However, as the price of the component has dropped dramatically over time and the demand for network dynamical systems has also increased, the concept of networked dynamical systems attracts increasingly more attention. Till now, it has found many applications in various areas, e.g., collaborative micro-robots, unmanned aircraft vehicles, traffic networks, smart grids (Knorn et al., 2016; Olfati-Saber et al., 2007).

Among the above applications, there exists a class of networked dynamical systems working in a repetitive manner with high control performance requirements. Figure 1.1 shows an example of Amazon warehouse mobile robot systems. In the figure, each mobile robot is a subsystem and the group of them constitutes the warehouse mobile robot system. For such system, it requires all the subsystems moving in the desired formation and transporting the goods repeatedly. During the control process, each subsystem communicates with its neighbouring subsystems and negotiates a common control strategy to achieve the desired formation. Note that, the high control performance requirement is crucial in such system, as even a minor mistake caused by a single robot (subsystem) may lead to trouble for the whole warehouse mobile robot system, e.g., if one robot can-

Figure 1.1: Example of Amazon warehouse mobile robot (Wurman et al., 2008)

not achieve the speed consensus with the neighbours, the formation will deviate from the desired objective.

In such applications, most of the traditional control methods have significant difficulties in meeting the high performance requirements (Knorn et al., 2016; Olfati-Saber et al., 2007): the traditional centralised structure has poor scaling proprieties and cannot be applied to large scale networked dynamical systems; distributed control design often focuses on asymptotic performance than dynamic tracking; in addition, all of them require highly accurate model information to achieve the high performance requirement, which is often expensive and/or difficult to obtain in practice. High performance control design algorithms without the need of accurate model information have been waiting.

1.2 Research Motivation

Iterative learning control (ILC) is a well-known control design method that especially suitable for the system executing the same task repetitively and requiring high control performance. ILC can 'learn' from the input and error information of the previous attempts of the same task to generate its control input. Since the information have actually contained the model information, ILC does not require a highly accurate model to achieve the high performance requirements (Ahn et al., 2007; Bristow et al., 2006; Owens and Hatonen, 2005). This outstanding feature makes ILC increasingly more to be used on high performance networked dynamical systems working in a repetitive manner.

As stated in Section 1.1, conventional control algorithms are limited by several limitations, and hence they are not the best choice for high performance networked dynamical systems working in a repetitive manner. Benefiting from the learning process, ILC provides a more rational control structure for the aforementioned systems. A number of ILC algorithms have been proposed for different control tasks in networked dynamical systems, e.g., collaborative tracking (Chen et al., 2018a; Chu, 2019; Devasia, 2016), consensus tracking (Devasia, 2017; Jin, 2016; Li and Li, 2014a; Li et al., 2018; Li and Li, 2014b; Meng et al., 2012, 2015a; Shen et al., 2019; Yang and Xu, 2016; Yang et al., 2016, 2017), formation control (Li and Li, 2014a; Li et al., 2018; Li and Li, 2014b; Liu and Jia, 2012, 2015; Meng and Jia, 2014; Meng and Moore, 2016, 2017; Meng et al., 2012, 2014a, 2015b; Xu et al., 2011; Yang et al., 2017). However, there are some limitations in existing methods:

- **Poor scalability:** Scalability indicates the algorithm's capacity to handle large scale network and dynamically growing network:

  (A) The majority ILC algorithms have difficulties to control large scale networked dynamical systems, which causes the scalability problem. In particular, existing distributed PID type and adaptive type ILC (which currently occupy a dominant number in ILC design for networked dynamical systems) need to calculate the condition to guarantee the convergence performance and this convergence condition normally involves the Laplacian matrix of the graph (which is used to define the network topology and will be carefully defined in the next section) in the calculation. The dimension of Laplacian matrix is depending on the network size, and hence the computational complexity will increase as the increasing of the size of the network. When the size of the network increases to a very large number (e.g., millions), then it is extremely challenging to design a controller using most of these existing algorithms.

  (B) Another important issue of the scalability is about how to deal with a dynamically growing network, i.e., the algorithms do not need to be redesigned and/or the parameters do not need to be re-tuned even when the size of the network increases during the control process. For the dynamically growing network, none of the existing distributed/decentralised ILC algorithms for networked dynamical systems can deal with this situation. Currently, all the existing algorithms' convergence parameters are pre-calculated for certain network topologies and they have to use the new Laplacian matrix when the size of the network growing, hence they do not have the ability to handle the changing network. By contrast, the proposed algorithms in this thesis are designed in a fully distributed/decentralised manner and no parameters are required to be tuned even when the network change, hence it can deal with the dynamically growing network problem.

- **Limited convergence performance:** In practice, most designs require the tracking error norm to converge monotonically, since asymptotic convergence may cause the tracking error norm to grow to a large value before it eventually converges. The sudden increase in the error norm may cause some practical problems, e.g., product quality problem, time-consuming problem, and equipment depreciation problem (Owens and Hatonen, 2005). However, most of the existing designs focus on asymptotic convergence, while monotonic convergence of the tracking error norm, is either not guaranteed, or is only achievable when the system dynamics satisfy certain conditions.

- **Lack of the ability to deal with system constraints:** System constraints are widely existing in the practical design. As an example, within a group of unmanned aerial vehicles (UAVs) working together, each UAV has an allowable input (voltage) range. If one agent's applied input exceeds its acceptable voltage range, it may damage that UAV and further affect the mission of the whole networked dynamical system. Unfortunately, none of the existing ILC algorithms has the ability to handle the control constraints in the networked dynamical systems.

- **Lack of the ability to control general point to point (P2P) task:** P2P task focuses on the tacking performance at specific time instants. It has found a great number of applications in practice, e.g., robotic 'pick' and 'place' task (which only consider the tracking performance on the 'pick' and 'place' time instantsE). Despite its importance in practical applications, most of the existing ILC algorithms for networked dynamical systems have not considered the P2P tasks with exceptions for Meng and Jia (2011, 2012); Meng et al. (2013, 2014b, 2015c), where a special case, terminal ILC (i.e., the tracking at the final point), has been considered. However, terminal ILC can only achieve the tracking on the final point, while the general P2P task in networked dynamical systems cannot be solved.

The above problems limit the algorithm's control performance for practical applications. Hence, the following question arises naturally: could we design distributed/decentralised ILC algorithms to address all of the above limitations? This question motivates the research for this thesis.

1.3 Contributions and Organisation

1.3.1 Contributions

Motivated by the aforementioned limitations of existing ILC algorithms for networked dynamical systems, this thesis introduces novel distributed/decentralised optimization-based ILC designs for the networked dynamical systems working in a repetitive manner and requiring high control performance. The main contributions of this thesis are described as follows:

- The development of optimisation-based ILC frameworks for different control problems (e.g., consensus tracking, formation control, collaborative tracking) in networked dynamical systems. All the resulting frameworks can achieve the desired control objective and have superior convergence properties (e.g., most of them can guarantee the monotonic convergence of the tracking error norm). Moreover, they have certain degree of robustness against the model uncertainty, which are desirable in practice.

- The derivation of distributed/decentralised algorithms to implement the proposed designs using the alternating direction method of multipliers (ADMM). ADMM is a powerful tool for solving distributed/decentralised optimization problems, and has found applications in a range of areas, e.g., in machine learning, applied statistics and privacy preservation (Boyd et al., 2011; Cheng et al., 2017; Deng and Yin, 2016; Zhang et al., 2019a). Using ADMM, the resulting algorithms can be implemented in a distributed/decentralised manner which only requires local information, and therefore guarantees the algorithm's scalability and can be applied to large scale networked dynamical systems.

- The exploitation of novel ILC algorithms to deal with the constraint handling problem in networked dynamical systems. For this type of problem, the proposed ILC design methods guarantee not only the satisfaction of the system constraints, but also the convergence of the tracking error norm to a 'best fit' solution (in particular, zero tracking error when the desired objective is achievable), which is appealing in practice.

- The extension of the proposed ILC algorithms to deal with unexplored problem in ILC design for networked dynamical systems, namely, P2P control task. For P2P control problem, the proposed design can not only guarantee the achievement of the desired objective, but also converge to the minimum control energy solution for a particular choice of initial control input.

Note that, the proposed ILC algorithms have the capacity to control the networked dynamical systems with either homogeneous (i.e., all the subsystems have the same system dynamics) or heterogeneous (i.e., the system dynamics can be different) networks. Furthermore, most of the proposed algorithms have the ability to handle the non-minimum phase systems, since the design does not involve the calculation of the inverse of subsystem's dynamics.

1.3.2 Organisation

The organisation of the thesis is shown in Figure 1.2. The detailed explanation of each chapter is given as follows:

Chapter 2 reviews the concept of networked dynamical systems and defines the consensus tracking problem, formation control problem in networked dynamical systems. We

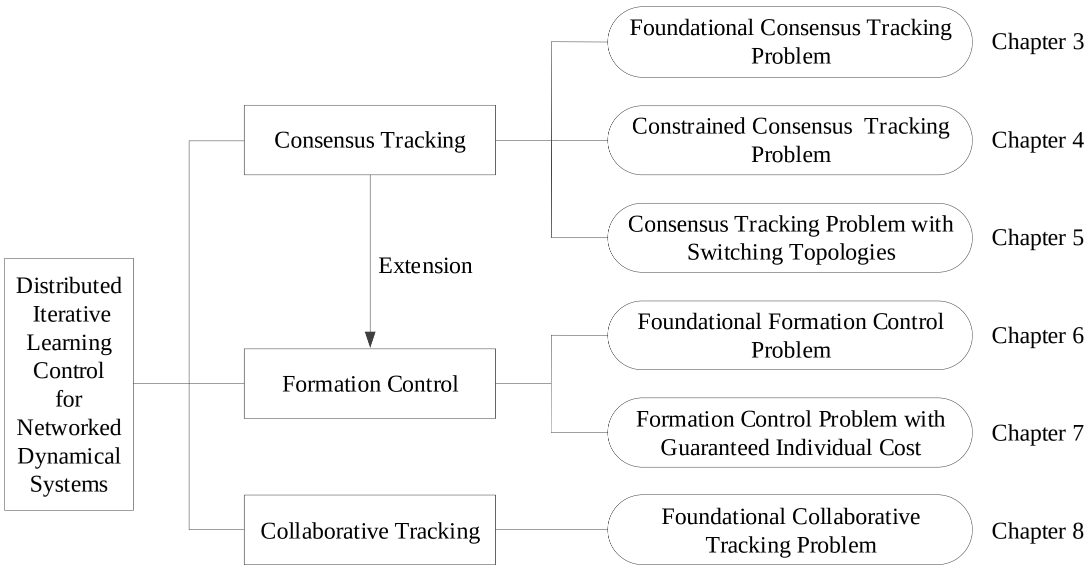
Distributed Iterative Learning Control for Networked Dynamical Systems
Consensus Tracking
Extension
Formation Control
Collaborative Tracking
Foundational Consensus Tracking Problem
Chapter 3
Constrained Consensus Tracking Problem
Chapter 4
Consensus Tracking Problem with Switching Topologies
Chapter 5
Foundational Formation Control Problem
Chapter 6
Formation Control Problem with Guaranteed Individual Cost
Chapter 7
Foundational Collaborative Tracking Problem
Chapter 8

Figure 1.2: The structure flow chart of this thesis

describe the details of concepts, background, mission of ILC and the basic algorithms in ILC. A general review of all the existing ILC algorithms for networked dynamical systems working in a repetitive manner is given to describe the current research landscape. Finally, we provide an explicitly description of ADMM and then review the general idea of ADMM for 'consensus' and 'sharing' problem.

Chapter 3 considers the high performance consensus tracking of networked dynamical systems. We formulate the high performance consensus tracking problem into the norm optimal ILC (NOILC) framework and propose a distributed implementation method using ADMM. The resulting ILC algorithm not only guarantees the tracking error norm converges monotonically to zero, but also has certain degree of robustness against the model uncertainty. In particular, the proposed algorithm can be extended to solve P2P consensus tracking problem, and applied to both homogeneous and heterogeneous networks, as well as non-minimum phase systems, which is of great practical relevance. Rigorous analyses of algorithms' (convergence and robustness) properties are given and simulation results are presented to demonstrate the effectiveness of the proposed distributed NOILC algorithm.

Chapter 4 considers the constrained consensus tracking problem and proposes constrained ILC algorithms using the well-known successive projection framework. The proposed algorithms guarantee the satisfaction of the system constraints and have appealing convergence performance: when perfect consensus tracking is achievable, the consensus tracking error norm converges monotonically to zero; otherwise, it converges monotonically to a minimum (possible) tracking error norm. Furthermore, we provide distributed implementations for the proposed ILC algorithms using the idea of ADMM, allowing the proposed algorithms to be applied to large scale networked dynamical systems using only local information. Convergence properties of the algorithms are analysed rigorously and numerical examples are given to show their performance.

Chapter 5 considers the high performance consensus tracking problem with switching network topologies and proposes a novel ILC framework to deal with this type of problem. The design of a novel performance index guarantees monotonic convergence of the tracking error norm to zero even with the switching network topologies. Furthermore, the proposed algorithm is suitable for homogeneous and heterogeneous networked dynamical systems, which is appealing in practice. A distributed implementation using the idea of ADMM for the proposed ILC framework is provided, allowing the algorithm to be applied to large scale networked dynamical systems. Convergence properties of the algorithm are analysed rigorously and numerical examples are presented to verify the algorithm's effectiveness.

Chapter 6 considers the high performance formation control of networked dynamical systems. We formulate the formation control problem into the NOILC framework and provide the ADMM distributed implementation of the proposed framework (as well as the method to find the best parameter of ADMM). The algorithm guarantees monotonic convergence of the formation error norm to zero, and can handle both homogeneous and heterogeneous networks, as well as non-minimum phase systems. In addition, compared to most existing algorithms, the proposed algorithm has a distinguished character that it converges to the minimum control energy solution for a particular choice of initial control input. An analysis of the algorithm's properties is given and numerical simulations are given to demonstrate the effectiveness of the proposed algorithm.

Chapter 7 considers an unexplored design problem in ILC design for networked dynamical systems, which aims at minimising the individual input energy cost in the system. Two novel distributed ILC algorithms are proposed: the first algorithm can guarantee the tracking error norm converge monotonically to a minimum value, while ensuring the individual input energy consumption does not exceed a prescribed level in the whole control process; the second algorithm can generate a proper input, which not only achieves the desired objective, but also produces minimum individual energy cost. Furthermore, the proposed algorithms are suitable for both homogeneous and heterogeneous networks, as well as non-minimum phase systems, which is appealing in practice. Convergence properties of the algorithms are analysed rigorously and numerical examples are presented to demonstrate the algorithms' effectiveness.

Chapter 8 considers the constrained collaborative tracking problem and proposes decentralised optimisation based ILC algorithm to solve it. The proposed algorithm can guarantee not only the satisfaction of the system constraints, but also the monotonic convergence of the tracking error norm to a minimum (possible) solution. Furthermore, the proposed algorithms are suitable for both homogeneous and heterogeneous networks, as well as non-minimum phase systems. To verify the effectiveness of this proposed algorithm, we exhaustively analyze the corresponding convergence properties and provide numerical examples.

Chapter 9 concludes the works have been done in this thesis, and then points out the future directions of the research.

Chapter 2

Literature Review

This chapter is divided into five parts. In Section 2.1, the concept and network topologies of networked dynamical systems are given. In addition, the ideas of consensus and formation control problems are presented, with the plant modelling for networked dynamical systems introduced. Section 2.2 provides a review of the concepts of iterative learning control (ILC). Some well-known ILC algorithms and important issues (e.g., robustness, non-minimum phase systems) are introduced. In Section 2.3, we review the idea of ILC for networked dynamical systems and give examples to illustrate the current situations of ILC for high performance networked dynamical systems. Section 2.4 reviews the idea of the alternating direction method of multipliers (ADMM) and introduces two important problem formulations (i.e., consensus and sharing) in ADMM. Finally, we summarize the context and highlight the motivations in Section 2.5.

2.1 Networked Dynamical Systems

This section introduces the concept of networked dynamical systems and then provides the modelling details for systems. Based on the formulation, we review some important control design problems in networked dynamical systems and introduce an important class of networked systems, which works repetitively and requires high performance.

2.1.1 Introduction

Networked dynamical systems are systems that contain more than one subsystem (agent) working together and achieving the desired objective. In networked dynamical systems, each subsystem needs to communicate, negotiate, and collaborate with neighbouring subsystems to reach a global objective, e.g., cooperatively track a desired output (Mirza et al., 2015), reach a common state (Anderson et al., 2008). In contrast to a single agent system that has limited capacity to observe the environment, networked dynamical

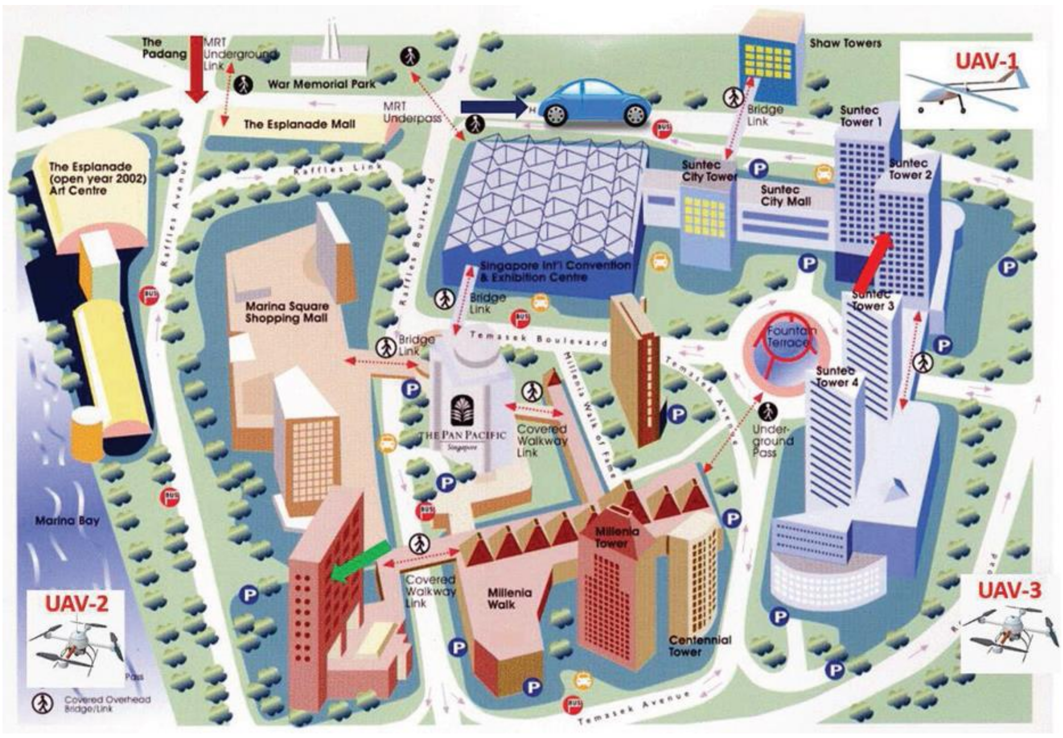

Figure 2.1: Example of networked dynamical systems (Yang et al., 2017)

systems have many advantages, e.g., wider mission areas, higher efficiency, improved performance, stronger robustness, parallel computing ability (Schurr et al., 2005).

Figure 2.1 illustrates an example of networked dynamical system in the civil domain. In the figure, the networked dynamical system consists of three heterogeneous unmanned aerial vehicles (UAVs), and the mission of the whole system is to accomplish exploration, searching, and surveillance of the urban region. For a single UAV, its control area is limited and it cannot cover the whole area at the same time. By adding two more subsystems into the network, the networked dynamical system's control area becomes large enough to cover the entire urban area, which ensures the feasibility of the mission.

Note that, the network topology could be homogeneous (i.e., all the subsystems have the same system dynamics) and heterogeneous (i.e., the subsystem's dynamics can be different from each other) in the system, hence it creates possibilities to describe different real-world systems. Up to now, networked dynamical systems have found numerous applications in varied areas, e.g., unmanned aerial vehicles, autonomous surface vessels, batch processes, traffic networks, smart grids, satellites (Olfati-Saber et al., 2007).

2.1.2 Modelling for Networked Dynamical Systems

The system model is important in analysing & designing the networked dynamical systems. Hence, we first introduce the general formulation of the system dynamics and the network topologies before formally defining the control problem in networked dynamical systems.

2.1.2.1 System Dynamics

Consider a networked dynamical system including p subsystems (agents), with the dynamics of i^{th} $(1 \le i \le p)$ subsystem is assumed to be a discrete-time, linear time invariant (LTI), single-input-single-output (SISO) system, represented using the following state-space model

$$\begin{aligned} x_i(t+1) &= A_i x_i(t) + B_i u_i(t), \qquad & x_{i,0} &= x_i(0) \\ y_i(t) &= C_i x_i(t), & i &= 1, 2, \cdots, p \end{aligned} \tag{2.1}$$

where $t \in [0, N]$ is the time index; N is the trial length; i is the subsystem index; $x_i(t) \in \mathbb{R}^{n_i}$ (n_i is the order of i^{th} subsystem), $u_i(t) \in \mathbb{R}$, $y_i(t) \in \mathbb{R}$ are the state, input and output of subsystem i; A_i, B_i, C_i are system matrices with proper dimensions.

2.1.2.2 Network Topology

To describe the relationship between different subsystems, we use the graph theory which is commonly used in analysing the network topology. As a branch of mathematics, graph theory provides a practical mathematics tool for analysing the property of networked dynamical systems. It is widely believed that Euler (1741) first introduced the concept of graph theory for 'the Seven Bridges of Königsber' problem. After that, this theory was generalized and derived to an important branch of mathematics, namely, topology.

In this thesis, we consider the undirected graph as the network topology since all the proposed algorithms are established on the undirected graph structure. The presentation in the following is mainly summarised from Bullo (2018).

Assuming a networked dynamical system is formed by p subsystems, and then the network topology can be represented using an undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V} = \{1, 2, ..., p\}$ is the vertex (node) set and $\mathscr{E}$ is the set of unordered pairs of vertexes called edges. For vertexes $i, j \in \mathscr{V}$ $(i \neq j)$, the set $(i, j) \in \mathscr{E}$ implies an undirected edge between vertexes i and j.

Figure 2.2 gives an example of an undirected graph in a networked dynamical system. In the figure, there exist three different vertexes and each vertex has certain relationship with other vertexes. For vertexes 1 and 2, they have an unordered edge between them, and hence set $(1, 2)$ define the edge $\mathscr{E}$, which means they can transfer the information between each other. On the other hand, vertexes 1 and 3 do not have any unordered edge between them, meaning they cannot transfer any data directly. However, vertex 1 can still indirectly transfer the information to 1 with the help of vertex 3.

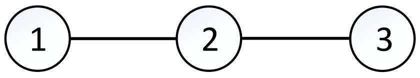

Figure 2.2: Example of undirected graph

To further describe the relationship between vertexes and edges, we introduce the adjacency matrix $\mathscr{A} = [a_{ij}]$. Its element a_{ij} is defined as

$$a_{ij} = \begin{cases} W_{ij} & if\ (i,j) \in \mathscr{E} \\ 0 & otherwise \end{cases} \tag{2.2}$$

where weight W_{ij} is often considered as the connection strength of the edge , i.e., subsystem i will put more emphasis on the information transformation with subsystem j. Given an example, in the transportation system, there exist some subsystems have direct access to the reference trajectory (i.e., 'leader' subsystems), their neighbours (without access to the reference) know the leaders have information about the desired objective and hence they will put more emphasize on communicating information with the 'leader' subsystems to improve its tracking efficiency. Note that, for an undirected graph, the adjacency matrix $\mathscr{A} = [a_{ij}]$ is symmetric.

For each subsystem, the range of communication is limited, and its next moment state is mainly calculated using the state of itself and its 'neighbours'. Usually, $\mathscr{N}_i$ is used to denote the 'neighbours' of subsystem i, and it is defined as

$$\mathscr{N}_i = \{j : (i,j) \in \mathscr{E}\} \tag{2.3}$$

In fact, the relationship between two subsystems may change along with different factors, e.g., time. Based on the changing relationship between two different subsystems, graphs can also be divided into two categories:

- **Fixed Graph:** The relationship between subsystems would not change for any factor. In this case, each subsystem has fixed 'neighbours' and exchanges information at all times.
- **Switching Graph:** The relationship between subsystems would change for different factors, e.g., time, experiment number. Under this situation, there exists a rule indicates how the 'neighbour' change. As an example, graph structure $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ becomes $\mathscr{G}(t) = (\mathscr{V}, \mathscr{E}(t))$ if the relationship between two subsystems change along with time.

Based on the i^{th} node's neighbours set, the concept of degree is introduced, which denoted as $d(i)$. In particular, the degree of a node i is the weighted number of edges of the form $(i, j) \in \mathscr{E}$, i.e.,

$$d(i) = \sum_{j=1}^{p} a_{ij} \tag{2.4}$$

and then, the global degree matrix is denoted as $\mathscr{D} = diag(d(1), d(2), \cdots d(p))$.

In addition to the adjacency matrix $\mathscr{A}$, there exists a more commonly used matrix called Laplacian matrix $L = \{l_{ij}\} := \mathscr{D} - \mathscr{A}$. It is a real symmetric matrix with element l_{ij} defined as below

$$l_{ij} = \begin{cases} \sum_{j=1}^{p} a_{ij} & if \quad j = i \\ -a_{ij} & otherwise \end{cases}, \tag{2.5}$$

or alternatively, in the following form

$$l_{ij} = \begin{cases} -W_{ij} & if \quad j \in \mathscr{N}_i \\ \sum_{j \in \mathscr{N}_i} W_{ij} & if \quad j = i \\ 0 & otherwise \end{cases} \tag{2.6}$$

Note that, the Laplacian matrix is commonly used when analysing networked dynamical systems, since it combines the degree matrix with the adjacency matrix and provides more detail when describing a network. Hence, we will design the algorithm mainly based on the Laplacian matrix.

Another important property of a graph is the connectivity and it can be divided into two different types in an undirected graph

- **Connected:** Each vertex can be connected with other vertexes through an undirected path.
- **Unconnected:** None of the vertexes can be connected to all the vertexes through undirected paths.

Normally, the undirected graph in the design is required to be connected to make sure the control information can flow through all the vertexes. However, using the switching topologies can release this requirement in some cases. As an example, the algorithm proposed in Jiang and Jiang (2020) only requires the union of the graph, over the time intervals, to have a spanning tree.

2.1.3 Control Design of Networked Dynamical Systems

Networked dynamical systems have found a great number of control designs in different fields, e.g., consensus, flocking, formation control (Olfati-Saber et al., 2007). Among these designs, consensus problem and formation control are two important classes, hence both of them will be introduced in the following.

2.1.3.1 Consensus Problem

The earliest studies of consensus problem appeared in the area of management science, statistics (DeGroot, 1974). Then its concept began to be used in different fields (Benediktsson and Swain, 1992; Olfati-Saber et al., 2007; Weller and Mann, 1997). In the area of networked dynamical systems, consensus problem requires all the subsystems to reach an agreement state that is negotiated by all of them (Olfati-Saber et al., 2007).

Consider a networked dynamical system where the dynamics of i^{th} $(1 \leq i \leq p)$ subsystem is assumed to be a first-order integrator, i.e.

$$x_i(t+1) = u_i(t). \tag{2.7}$$

The network topology is denoted using an undirected graph and the control design objective is to find a proper input $u_i(t)$ that guarantees all the subsystems reach a common state, i.e.

$$\lim_{t\to\infty} x_1(t) = \lim_{t\to\infty} x_2(t) = \cdots = \lim_{t\to\infty} x_p(t) \tag{2.8}$$

According to Olfati-Saber and Murray (2004), when the network topology is either a fixed graph, or a switching graph with zero transmission time-delay, the following control law can be used to achieve the distributed consensus design objective:

$$u_i(t) = \sum_{j\in\mathscr{N}_i} W_{ij}(x_j(t) - x_i(t)) \tag{2.9}$$

Input updating law (2.9) indicates that subsystem i keeps sharing its state x_i with all its neighbours $j \in \mathscr{N}_i$ during the control and its input is a collective decision by all the subsystems. Eventually, the collective decision of all the subsystems is equivalence to the average of all the subsystems' initial state, i.e.

$$\lim_{t\to\infty} x_i(t) = \frac{1}{p}\sum_{i=0}^{p} x_i(0) \tag{2.10}$$

which is called as average consensus. Besides the average consensus, there are other types of consensus designs: cluster consensus (Chen et al., 2011; Han et al., 2013; Liu and Chen, 2011), leader-follower consensus (Defoort et al., 2015; Ni et al., 2017), high performance consensus tracking within a finite time interval (Devasia, 2017; Jin, 2016; Yang et al., 2017).

2.1.3.2 Formation Control

Formation control is another important topic in the coordination of networked dynamical systems. It requires a group of subsystems working together to maintain a desired state-

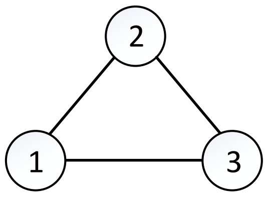

Figure 2.3: Triangle formation

difference between each other, which requires the collaboration of every subsystem at all times (Oh et al., 2015). For example, if the networked dynamical systems are required to form a triangle (as shown in Figure 2.3), we can define the state difference $\mathscr{R}_{ij}$ as the desired state-difference between subsystems i and j.

Consider a networked dynamical system where the dynamics of i^{th} $(1 \leq i \leq p)$ subsystem is assumed to be a first-order integrator, i.e.

$$x_i(t+1) = u_i(t), \tag{2.11}$$

and then the control design objective of formation control is to find a proper input $u_i(t)$ that can reach a relative state consensus, i.e.

$$\lim_{t\to\infty} x_i(t) = \lim_{t\to\infty} x_j(t) - \mathscr{R}_{ij}, \qquad j \in \mathscr{N}_i. \tag{2.12}$$

When the network topology is either a fixed graph, or switching graph with zero transmission time-delay, the following control law can be used to achieve the distributed formation (Olfati-Saber et al., 2007):

$$u_i(t) = \sum_{j\in\mathscr{N}_i} W_{ij}(x_j(t) - x_i(t) - \mathscr{R}_{ij}) = \sum_{j\in\mathscr{N}_i} W_{ij}(x_j(t) - x_i(t)) - \sum_{j\in\mathscr{N}_i} W_{ij}\mathscr{R}_{ij}. \tag{2.13}$$

From input updating law (2.13), it shows that the formation control problem requires the state difference between neighbouring subsystems equal to a nonzero bias term $\sum_{j\in\mathscr{N}_i} W_{ij}\mathscr{R}_{ij}$, which can be seen as a variant form of the consensus problem.

2.1.4 High Performance Networked Dynamical Systems

As described in the previous session, networked dynamical systems have found increasingly more applications during the last few decades, thanks to the significant reduction in the cost of sensing, computing and actuating technologies. Among these applications, there exists a class of networked dynamical systems working in a repetitive manner (i.e., executing the same task repetitively within a finite time interval) and requiring high performance during the control process. This type of networked dynamical system is called as high performance networked dynamical system and it has broad applications in different fields including physics, automobiles, biology and robotics (Liu and Jia, 2012).

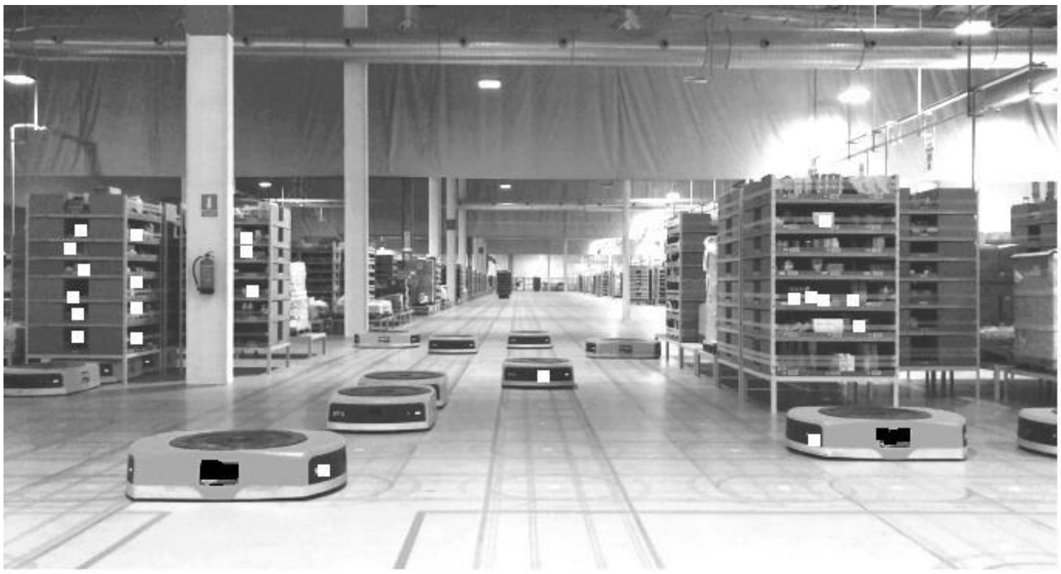

Figure 2.4: Example of high performance networked dynamical systems working in a repetitive manner: Amazon warehouse mobile (Farah et al., 2019)

In the following, we present two applications of high performance networked dynamical systems in real world:

1. Amazon Warehouse Mobile

Amazon warehouse mobile is a typical example of high performance networked dynamical systems, which is illustrated in Figure 2.4. In the figure, a group of warehouse mobiles (subsystems) form a networked dynamical system and the mission of the whole system is to transport the 'goods'. Note that, the warehouse system is working in a repetitive manner to guarantee the 'goods' can be precisely transferred to the right destination.

In each repetitive process, the Amazon warehouse robot will leave the charging station at the beginning (i.e., the initial condition remains the same) and continuously transport the 'goods' within a finite time interval (until it almost runs out of battery and returns to the charging station). For such application, all the Amazon robots are working in the same environment and performing their own mission in the network. The repetition of initial conditions and the control process is useful for such application: on the one hand, the repetition guarantees all the data from previous experiences are available and the controller can use all the data (stored in the memory) within the repetitive process to greatly save the computational complexity; on the other hand, the repetition process guarantees the causality when using all the past data in the whole time range $t \in [0, T]$ and hence improve the performance (by learning from the future time information in the past process).

2. Freeway Traffic System

From a macro point of view, the traffic flow pattern in the freeway traffic system works

Figure 2.5: Example of high performance networked dynamical systems working in a repetitive manner: Freeway traffic system (Yang et al., 2016)

in a repetitive manner that requires high control performance every day. Taking one day as a unit, the traffic flow starts from a high level in the morning rush time (around $7-9$ AM), then fluctuates until reaches the second peak in the afternoon (around $5-7$ PM). After reaching the afternoon peak, the traffic flow will decrease to a low level at night and finally return to the morning peak on the second day. Clearly, the traffic system is operating in a repetitive manner and the macroscopic traffic model is invariant along the week/month axis.

To illustrate the high performance requirement in the freeway traffic system, we take Figure 2.5 as an example. In the figure, each vehicle (subsystem) is required to coordinate its speed, and location with neighbouring vehicles because it needs to keep a safe distance from its neighbouring vehicles. The requirement on the speed consensus is extremely important, since any mistake caused by one subsystem (e.g, engine stalls) may lead to huge trouble (e.g., vehicle collision). It is worth mentioning that this application is often coupled with constraint handling problem. For example, the allowable amount of subsystems on the freeway will change along the hour axis because of the traffic rule and the limited capacity. This constrained design, requiring the traffic system to control the number of subsystems on the freeway to move in a desired formation, is non-trivial.

As shown in the above examples, this class of networked dynamical systems has enormous potential and board applications, which attracts increasingly more interest in practice. However, to achieve the high performance objective, most traditional control design methods will meet difficulties as they often require a highly accurate model which can be expensive and/or difficult to obtain in practice (Knorn et al., 2016). To avoid the strict requirement of the highly accurate model, the next section will introduce ILC which does not require a highly accurate model to achieve the high performance requirement.

2.2 Iterative Learning Control

This section gives a review of the ILC design. Concept, classical algorithms (including model-free and model-based design) and important issues of ILC are introduced. The presentation in the following is based on the work in Bristow et al. (2006); Owens (2016); Owens and Hatonen (2005).

2.2.1 Overview of Iterative Learning Control

2.2.1.1 Plane Modelling for ILC

ILC is a control design method extremely suitable for the system that executes the same task repeatedly and requires high control performance. In the control community, the heuristic article of ILC research is Arimoto et al. (1984). After that, a great number of ILC approaches have been proposed and the idea of ILC has become well-known in the control community.

In this section, we formulate the system using the following discrete time, linear time invariant, single-input-single-output state space model

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t), \qquad x_k(0) = x_0 \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{2.14}$$

where $t \in [0, N]$ is the time index; N is the trial length; k is the trial number; $x_k(\cdot) \in \mathbb{R}$, $u_k(\cdot) \in \mathbb{R}$, $y_k(\cdot) \in \mathbb{R}$ are the state, input and output at trial k, respectively; A, B, C are system matrices with proper dimensions; the initial condition is the same for all the trials. The system is working in a repetitive manner to track a reference over a finite time interval: at $t = N + 1$, the time t is reset to 0, the subsystem's state is reset to x_0, and the system is required to track the same reference again. Note that, the reset of initial condition is commonly required in ILC design, since it guarantees the causality when updating the control law.

By defining $r(t)$ as the reference signal, the mission of the system can be stated as generating the output $y_k(t)$ that tracks $r(t)$ precisely. Then, we can define the tracking error $e_k(t)$ as

$$e_k(t) = r(t) - y_k(t) \tag{2.15}$$

For traditional non-learning control algorithms, all the parameters are already set up once the controller is designed. Theoretically, if the model of a control system is precisely

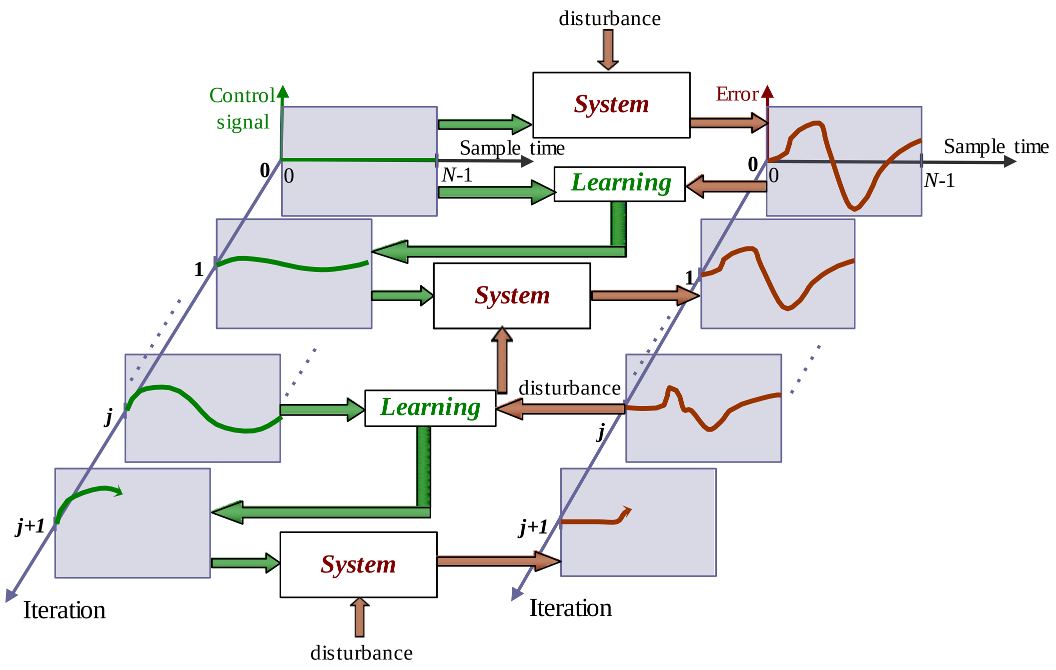

Figure 2.6: The block diagram of iterative learning control (reproduce from Figure 1 in (Bristow et al., 2006))

known, then the perfect tracking of reference trajectory is achievable. However, model uncertainty is commonly existing in practice and this reason makes most of the traditional non-learning control algorithms could not achieve the perfect output tracking of $r(t)$. By contrast, ILC can use the information (i.e., input and error) from the previous experiments to update its new trial's input (which is illustrated in Figure 2.6). Since the error information has contained the plant's model information, ILC can eventually achieve the perfect output tracking (by learning previous trials' information).

To facilitate later design for ILC, a 'lifted matrix form' representation of the system dynamics is first introduced (Hatonen et al., 2004). Assuming the relative degree of each subsystem is unity, i.e., $CB \neq 0$. Introduce u_k, y_k, r, e_k as the 'lifted form' of input $u_k(t)$, output $y_k(t)$, reference $r(t)$, and error $e_k(t)$, which are defined as follows

$$u_k = \begin{bmatrix} u_k(0) \\ u_k(1) \\ \vdots \\ u_k(N-1) \end{bmatrix} \quad y_k = \begin{bmatrix} y_k(1) \\ y_k(2) \\ \vdots \\ y_k(N) \end{bmatrix} \quad r = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(N) \end{bmatrix} \quad e_k = \begin{bmatrix} e_k(1) \\ e_k(2) \\ \vdots \\ e_k(N) \end{bmatrix} \tag{2.16}$$

Then, system model (2.14) can be represented in an equivalent 'lifted system' form

$$y_k = Gu_k + d \tag{2.17}$$

where $u_k \in \mathcal{U}$, $y_k \in \mathcal{Y}$, and $\mathcal{U}$, $\mathcal{Y}$ are real Hilbert space. The Hilbert space $\mathcal{H}$ is a complete inner product space and it is defined as

$$\mathcal{H} = \mathbb{R}^N \times \mathbb{R}^N \tag{2.18}$$

with inner product and induced norm

$$\begin{aligned} \langle u_1, u_2 \rangle_{\mathcal{U}} &= \sum_{t=0}^{N} u_1^T(t) R u_2(t), \qquad \|u\|_{\mathcal{U}} = \sqrt{\langle u, u \rangle_{\mathcal{U}}} \\ \langle y_1, y_2 \rangle_{\mathcal{Y}} &= \sum_{t=0}^{N} y_1^T(t) Q y_2(t), \qquad \|y\|_{\mathcal{Y}} = \sqrt{\langle y, y \rangle_{\mathcal{Y}}} \end{aligned} \tag{2.19}$$

and Q, R are positive definite matrices. The system matrix G is represented as

$$G = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \cdots & CB \end{bmatrix} \tag{2.20}$$

and d is the response of initial condition, which is defined as

$$d = \begin{bmatrix} CAx_0 & CA^2x_0 & \cdots & CA^Nx_0 \end{bmatrix}^T \tag{2.21}$$

Without loss of generality, we assume $d = 0$ by using $r - d$ to replace r, then the system model (2.17) can be simplified as

$$y_k = Gu_k \tag{2.22}$$

and the error vector e_k can now be rewritten as

$$e_k = r - Gu_k \tag{2.23}$$

In ILC, the control input is generated using the previous trials' input and error information, as shown in the following form

$$u_{k+1} = f(u_k, \ldots, u_{k-b}, e_k, \ldots, e_{k-c}) \tag{2.24}$$

where $0 < b \leq k$ and $0 < c \leq k$ (b and c are integer). By contrast with traditional non-learning algorithms, ILC can learn from the previous trials' data (e.g., inputs and errors) to update its input, and hence achieves the perfect tracking even when the model is inaccurate. An example is given below to further demonstrate the difference between the traditional feedback control algorithm and ILC.

Example 2.1. *Consider the following system*

$$G(s) = \frac{s+1}{s^2 + 5s + 3} \tag{2.25}$$

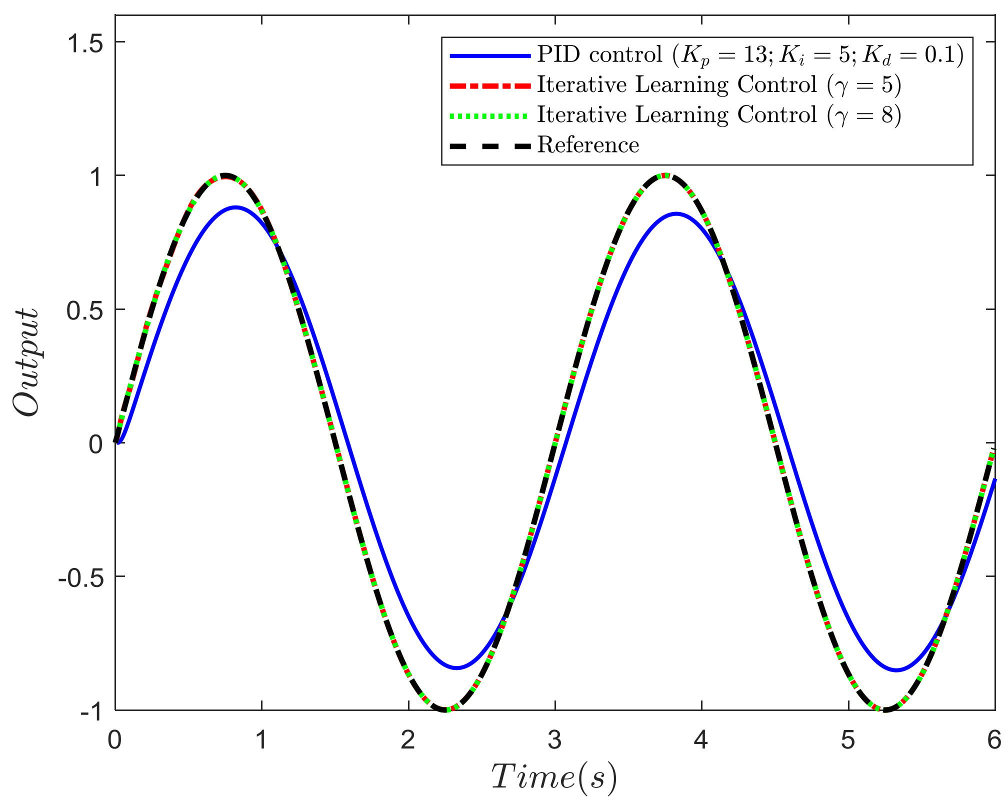

Figure 2.7: Output response of PID method and ILC algorithm at the 30th trial

where $t \in [0, 6]$. *This system is sampled using a zero-order hold, with the sampling time* $T_s = 0.01s$. *Zero initial condition is assumed and the reference trajectory* $r(t)$ *is defined as*

$$r(t) = sin(\frac{2}{3}\pi t) \tag{2.26}$$

To track the reference trajectory, we first use the Proportional-Integral-Derivative (PID) control algorithm (which is the most commonly used feedback control algorithm)

$$u(t) = K_p e(t) + K_i \sum_{l=0}^{t} e(l) + K_d[e(t+1) - e(t)] \tag{2.27}$$

to control the system for 30 trials (with the parameters $K_p = 13$, $K_i = 5$, $K_d = 0.1$*). Furthermore, to make comparison with the traditional PID control method, we use the Arimoto-type ILC algorithm (Arimoto et al., 1985)*

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1) \tag{2.28}$$

where $\gamma = 5, 8$, *to perform the same task again for* 30 *trials.*

Figure 2.7 shows the output for both two methods at 30th trial and it suggests that the output generated by ILC could track the reference trajectory perfectly, for both cases

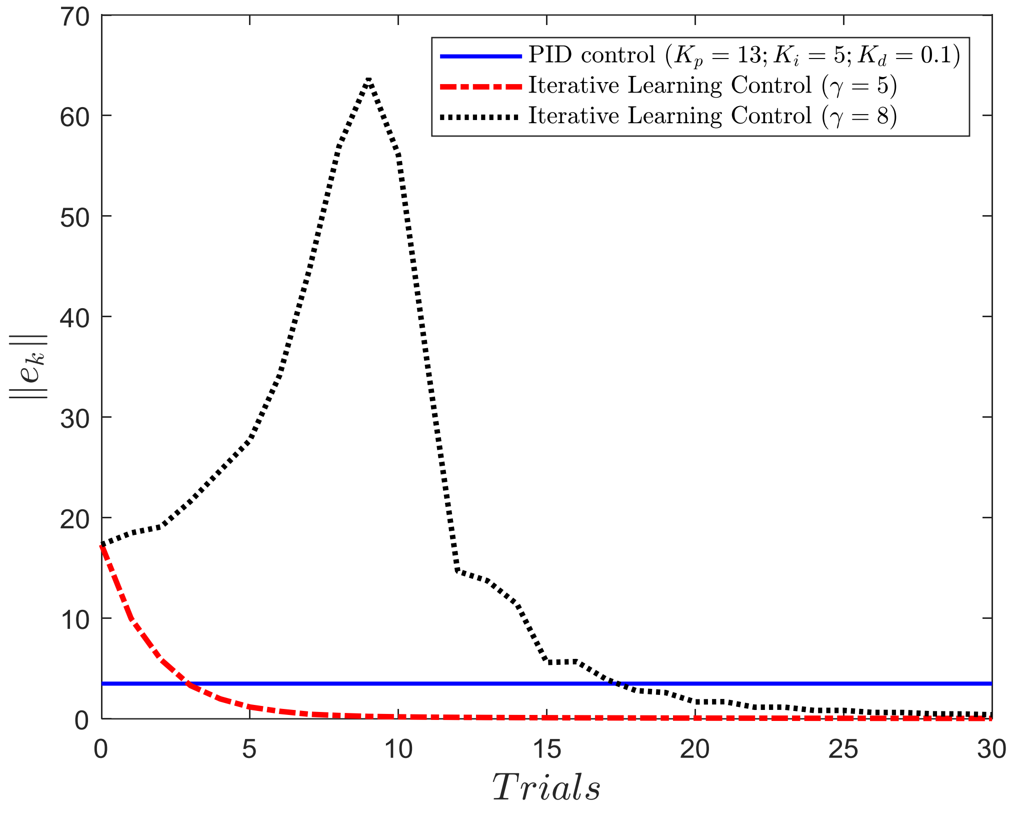

Figure 2.8: Convergence behaviour of PID method and ILC algorithm

$\gamma = 5$ and $\gamma = 8$. By contrast, the PID algorithm cannot achieve the perfect tracking of the reference signal for the choice of parameters. Figure 2.8 shows the convergence behaviour of the tracking error norm e_k over 30 trials. For PID control law, the tracking error norm $\|e_k\|$ remains the same along the trial length because of the non-learning property. In comparison, the tracking error norm of ILC design will converge to zero after 30^{th} trial, demonstrating that the perfect tracking of the reference trajectory can be achieved for both cases $\gamma = 5$ and $\gamma = 8$.

The fundamental idea of most traditional control methods is to design a model-based controller which accurately describes the real system, however, this controller is difficult and expensive to obtain in practice. ILC relaxes this restriction because its control input is generated by 'learning' the previous attempts' input and error. By constantly learning from previous information to modify its control strategy, ILC gradually improves the tracking performance. This outstanding advantage makes ILC explicitly suitable for high performance dynamical systems working in a repetitive manner.

Note that, in Figure 2.8, the ILC convergence behaviours are depending on the choices of learning gain γ. These two convergence behaviours represent two classes of convergence properties in terms of tracking error $e_k(t)$, which are summarised as follows:

- **Asymptotic convergence:** Asymptotic convergence only cares about the convergent result but not the convergence behaviour during the control process, i.e.

$$\lim_{k\to\infty} e_k = 0 \tag{2.29}$$

- **Monotonic convergence:** Monotonic convergence requires not only the tracking error $e_k = 0$ as $k \to \infty$, but also the tracking error norm in next trial is not greater than the tracking error norm in current trial, i.e.

$$\lim_{k\to\infty} e_k = 0 \qquad \& \qquad \|e_{k+1}\|_Q \leq \|e_k\|_Q \tag{2.30}$$

It is worth mentioning that, asymptotic convergence is often not enough since the tracking error may grow to a large value (as shown in Figure 2.8) before it ultimately converges. This may cause some problems in practice, e.g., product quality problem, wasting time or permanent damage of the system (Owens and Hatonen, 2005). Hence, monotonic convergence is more desirable when designing an ILC algorithm.

2.2.1.2 Adjoint Operator in ILC design

The adjoint operator G^* of system matrix G plays an important role in ILC design, hence we will give a discussion in the following. For the adjoint operator, it can be defined using the following equation (Luenberger, 1997)

$$\langle u, Gv\rangle_{\mathcal{Y}} = \langle G^* v, u\rangle_{\mathcal{U}} \tag{2.31}$$

for an arbitrary $u \in \mathcal{U}$ and $u \in \mathcal{Y}$, it can be shown that the adjoint operator G^* exists and is bounded and linear. When the system is discrete time, and the input and output space are defined as in (2.19), the following theorem holds:

Theorem 2.1. *With the above definitions, the adjoint operator of the linear map $y = Gu$ corresponding to the linear, time-invariant state space model $S(A, B, C, D)$*

$$x(t+1) = Ax(t) + Bu(t), \qquad y(t) = Cx_k(t) + Du(t), \qquad x(0) = 0 \tag{2.32}$$

is the map $v = G^* z$ corresponding to the linear system

$$p(t) = A^T p(t+1) + C^T Qz(t+1), \qquad v(t) = R^{-1}(B^T p(t) + D^T z(t)) \tag{2.33}$$

with a terminal boundary condition $p(N) = 0$. Or, in matrix form representation

$$G^* = \vec{R}^{-1} G^T \vec{Q} \tag{2.34}$$

where $\vec{R} = diag(R, R, \cdots, R)$ and $\vec{R} = diag(R, R, \cdots, R)$.

Now, after introducing the basic concept and idea of ILC, we will review some basic algorithms in ILC design in the following section. Note that, all the algorithms can be divided into model-free design and model-based design, and we first review the model-free ILC design (since it is much more straightforward).

2.2.2 Model Free ILC Algorithm

Model free ILC algorithms are easy to implement since they only require the information of tracking errors (without requiring any model information) when updating the input. In some cases, they are easy to implement and can provide acceptable performance, which is useful in some cases that do not require fast convergence speed and/or monotonic convergence of the tracking error norm. Next, we review some of the basic algorithms to illustrate the idea.

2.2.2.1 Proportional-Integral-Derivative ILC

In the field of ILC, the first algorithm was proposed by Arimoto et al. (1984) and it was called as the Derivative-type ILC. Its input updating law is shown as

$$u_{k+1}(t) = u_k(t) + K_d\big[e_k(t+1) - e_k(t)\big]. \tag{2.35}$$

Then, the Proportional-type ILC was presented in Arimoto et al. (1985), with the input updating law given as

$$u_{k+1}(t) = u_k(t) + K_p e_k(t+1) \tag{2.36}$$

However, the error term has not been fully utilized in both Proportional-type ILC and Derivative-type ILC. Hence, Arimoto and his co-author blended the idea of Proportional-Integral-Derivative (PID) algorithm into ILC in Arimoto (1986) and derived the PID type ILC algorithm

$$u_{k+1}(t) = u_k(t) + K_p e_k(t+1) + K_i \sum_{l=1}^{t} e_k(l) + K_d\big[e_k(t+1) - e_k(t)\big] \tag{2.37}$$

where K_p represents the proportional parameter; K_i denotes the integral parameter and K_d is the derivative parameter.

For PID type ILC, it records the error value and then uses the value to generate its input. It is robust to model uncertainty when the parameters are chosen appropriately. Hence, PID type ILC algorithm is still commonly used in ILC area.

2.2.2.2 Phase-Lead ILC

Phase-lead ILC is a feedforward algorithm and it was first introduced by Park et al. (1998). The general update law is shown as

$$u_{k+1}(t) = u_k(t) + Le_k(t+\tau) \tag{2.38}$$

where $\tau > 0$ is the phase-lead parameter of the error signal and L is an iteration gain.

The form of phase-lead ILC is similar to that of Proportional-type ILC and it is suitable for the dynamical system with time delay. For the phase-lead ILC design, it is useful when controlling a system with non-minimum phase dynamics. It can maximize convergence and minimize the tracking error when compared with P-Type, D-Type ILC (Freeman et al., 2005).

2.2.2.3 Data-Driven ILC

Data-driven ILC is a kind of model-free method that utilizes the data information to control the system. One common idea is to estimate the system model (parameters or system matrices) based on the input and output obtained in previous execution of the same task. After obtaining the estimated system model, the model-based ILC algorithms (which will be introduced in the later session) can be applied to the system. An example of this idea is shown below.

Janssens et al. (2013) proposed a data-driven method that especially uses the historical input, output to estimate the system matrix. In the paper, it first defines the u_{lc} and y_{lc} as the linear combinations of previous ILC trials' input and output signal, i.e.

$$\begin{aligned} u_{lc} &= l_0u_0 + l_1u_1 + \cdots + l_ju_j \\ y_{lc} &= l_0y_0 + l_1y_1 + \cdots + l_jy_j \end{aligned} \tag{2.39}$$

where l_j is weighting parameter for trial j.

Then, noting the fact that $y = Gu$, the estimated model can be defined as

$$\hat{G} = U_{lc}^{-1}Y_{lc} \tag{2.40}$$

in which U_{lc} and Y_{lc} are lower-triangular Toeplitz matrices of u_{lc} and y_{lc}. It has been proved in the paper that the estimated system matrix $\hat{G}$ is equal to G when the following conditions hold: (1) U_{lc} is full rank; (2) the whole system is LTI; (3) no measurement noise or other disturbances in the system.

However, noise and disturbance are commonly existing in practice, and the above requirement (3) is difficult to realize in practice. To relax this strict requirement, Janssens et al. (2013) further improved the estimation of the model. By assuming the measured output $y_j^m = y_j + n_j$ (with n_j a zero-mean Gaussian noise with standard deviation σ_n), Equation (2.40) can be further rewritten as

$$\hat{G} = U_{lc}^{-1} Y_{lc}^m = U_{lc}^{-1}(Y_{lc} + N_{lc}) \tag{2.41}$$

in which N_{lc} is the lower-triangular Toeplitz matrix of linear combinations of previous ILC trials' measurement noise N_j.

After using (2.41) to estimate the system model, model-based algorithms are applied to achieve the desired mission and result in a great performance. However, this data-driven method highly relies on the choice of l (e.g., how to choose a proper l to make sure U_{lc} is full rank), and it may be difficult to be used in practice.

In addition to the above algorithms, there is another class of model-free ILC suitable for nonlinear systems, namely, adaptive ILC. This ILC design is established on different approaches, e.g., contraction mapping (Hui et al., 2020; Xu, 1997; Xu and Qu, 1998; Yang et al., 2016), Lyapunov function (Jin, 2016; Qu and Xu, 2002; Shen et al., 2019; Tayebi, 2004), and can guarantee the achievement of the desired objective. Please refer to the above reference for more details.

2.2.3 Model Based ILC

In this section, we will introduce ILC algorithms that require both the tracking error and model information. The algorithms' convergence properties are analysed rigorously.

Consider the system model (2.14), the most widely used ILC updating law is shown as (Bristow et al., 2006)

$$u_{k+1} = Q(u_k + Le_k) \tag{2.42}$$

where Q denotes the Q-filter and L is the learning function. Normally, most of the ILC algorithm fix Q as an identity matrix and then choose different L. In the following, we set $Q = I$, and then from (2.42), we can have the following error evolution

$$e_{k+1} = (I - GL)e_k \tag{2.43}$$

To guarantee the convergence properties of the model based ILC algorithms, an intuitive way is to guarantee the spectral radius of matrix $I - GL$ smaller than 1, i.e.

$$\rho(I - GL) < 1 \tag{2.44}$$

where spectral radius $\rho(I-GL)$ is the modulus of the largest eigenvalue in matrix $I-GL$.

Note that, the condition in Equation (2.44) can only guarantee the asymptotic convergence of tracking error norm $\|e_k\|_Q$. As mentioned previously, monotonic convergence is more desirable in practice, and to achieve the monotonic convergence of $\|e_k\|_Q$, the convergence condition is further updated as

$$\|I - GL\|_Q < 1 \tag{2.45}$$

since

$$\|e_{k+1}\|_Q = \|(I - GL)e_k\|_Q \leq \|(I - GL)\| \, \|e_k\|_Q < \|e_k\|_Q \tag{2.46}$$

Note that the model G is fixed in (2.44) and (2.45), and hence, choosing a proper learning gain L is the main idea in model based ILC algorithms. In the following, we introduce three methods to choose the learning gain L.

2.2.3.1 Inverse Based ILC

An obvious way to guarantee the condition (2.45) is to choose the learning gain L as the inverse of system model G. This method is proposed by Harte et al. (2005) and the updating law is shown as follows

$$u_{k+1} = u_k + \beta G^{-1} e_k \tag{2.47}$$

with the convergence condition becomes

$$|1 - \beta| < 1. \tag{2.48}$$

Clearly, the choice of $\beta \in (0, 2)$ guarantees the convergence of tracking error norm. In the ideal case, the choice of $\beta = 1$ guarantees the zero tracking error in one trial, since

$$e_1 = r - (Gu_0 + GG^{-1}e_0) = 0. \tag{2.49}$$

However, we normally choose a small β because the system may contain the model uncertainty. In addition, inverse based ILC has difficulties to be applied to non-minimum phase systems because of the right half plane zeros.

2.2.3.2 Gradient Based ILC

To avoid the directly use of the model inverse, we introduce an alternative method called gradient based ILC (Owens et al., 2009). For any ILC algorithm, the ultimate goal is to find an proper input such that the tracking error $e_k = 0$ and this is a process of solving the following optimization problem

$$\min_{u_k \in \mathcal{U}} \|e_k\|_Q^2 \quad \& \quad e_k = r - Gu_k. \tag{2.50}$$

Note that, the steepest descent direction is the derivative of $\|e_k\|_Q^2$, i.e., the gradient of $\|e_k\|_Q^2$. Based on this idea, we can have the following control updating law

$$u_{k+1} = u_k + \beta G^* e_k \tag{2.51}$$

where $G^* = R^{-1}G^TQ$ is the adjoint of G. By choosing $L = \beta G^*$, the convergence condition in (2.45) becomes

$$\|I - \beta GG^*\| < 1 \tag{2.52}$$

In discrete time systems, since GG^* is a positive definite matrix, the convergence condition (2.52) can be further rewritten as

$$0 < \beta < \frac{2}{\|GG^*\|} = \frac{2}{\|G\|^2} \tag{2.53}$$

For gradient based ILC, it does not involve the inverse model in the design and hence it is much more robust to the model uncertainty, however, the strong robustness is at the expense of the convergence speed. For a gradient based ILC, its convergence speed can be slow even when the model is highly accurate. In the next section, we will introduce a moderate algorithm that can make the trade-off between robustness and convergence speed (by selecting different weighting matrices).

2.2.3.3 Norm Optimal ILC

Norm optimal ILC (NOILC) is a well-known algorithm which guarantees the monotonic convergence of the tracking error norm without any restraints on the controller. The idea of NOILC was first proposed by Amann et al. (1996a,b) in 1996 and from then on, the NOILC algorithm has been widely used in the ILC field, e.g. Barton and Alleyne (2011); Chu and Owens (2011, 2009); Owens (2016); Owens and Hatonen (2005). The presentation below is mainly summarised from Owens (2016).

Consider the system model (2.14), the general updating law of NOILC can be shown as

$$u_{k+1} = \underset{u_{k+1}\in\mathcal{U}}{\arg\min} \left\{ J_{k+1}(u_{k+1}) : \|e_{k+1}\|_{\mathcal{Y}}^2 + \|u_{k+1} - u_k\|_{\mathcal{U}}^2 \right\}. \tag{2.54}$$

Equation (2.54) shows that not only the error norm is required to be small in each trial, but also the difference value between the next trial's input and the current trial's input should not be large. By adding the second term, NOILC shows better performance than normal inverse-based algorithm both in stability and robustness (at the expense of convergence speed).

By directly applying partial derivative of J_{k+1} with respect to u_{k+1} and finding the stationary point (i.e., set $\frac{\partial J_{k+1}}{\partial u_{k+1}} = 0$), we can get the optimal solution

$$u_{k+1} = u_k + G^*(I + GG^*)^{-1}e_k \tag{2.55}$$

By multiplying G and subtracting from r in both side, we have

$$y_{k+1} = y_k + GG^*(I + GG^*)^{-1}e_k \tag{2.56}$$

and then the error evolution is given as

$$e_{k+1} = (I + GG^*)^{-1}e_k \tag{2.57}$$

Using the input updating law (2.55) to track a reference trajectory, NOILC has appealing convergence properties as shown in the following propositions:

Proposition 2.1. *(Amann et al., 1996b) Assuming that reference $r \in Rg(G)$, in which $Rg(G)$ denotes the range of G. The input sequence $\{u_k\}$ satisfies*

$$\lim_{k\to\infty} u_{k+1} - u_k = 0 \tag{2.58}$$

and the error sequence $\{e_k\}$ satisfies

$$\lim_{k\to\infty} e_k = 0 \tag{2.59}$$

Note that, for a continuous time system, the convergence in tracking error norm is slightly different (due to the infinite dimensions). Please refer to Amann et al. (1996a) for more details.

Proposition 2.2. *(Amann et al., 1996b) If the operator G^* has an inverse with norm $1/\sigma$ for some $\sigma > 0$, then the error sequence satisfies the following relationship:*

$$\|e_{k+1}\| \leq \frac{1}{1+\sigma^2}\|e_k\| \tag{2.60}$$

Note that, Amann et al. (1996a) and Amann et al. (1998) have shown that $\sigma^2 = 0$ (in Proposition 2.2) when the plane is a continuous-time system; otherwise, $\sigma^2 \neq 0$ when the plane is a discrete-time system. This implies that NOILC guarantees monotonic convergence in continuous-time systems and guarantees geometric convergence in discrete-time systems. Both Propositions 2.1 and 2.2 show that discrete-time NOILC achieves the perfect tracking of the reference trajectory, but also guarantees that the tracking error norm converges geometrically to 0, which is appealing in the practical design.

The NOILC input updating law (2.55) provides a simple way to obtain the control input, however, it requires the inverse calculation of the system matrix G and the computation will significantly increase with the dimensions of G. To address this limitation, a feedback plus feedforward implementation (Amann et al., 1996b) can be used to enhance the calculation speed and robustness. The introduction of Riccati equation $K(t)$, predictive term $\xi_k(t)$ can save the time used when calculating the system matrix G, at the same time, the feedback structure enhances the robustness when it is applied to the real world plant (because of the real-time feedback). The idea of the feedback plus feedforward implementation can be illustrated as follows:

Feedback plus feedforward implementation:

For the updating law (2.55), it can be further arranged as

$$u_{k+1}(t) = u_k(t) - R^{-1}B^T \left\{ K(t) \left[I + BR^{-1}B^T K(t) \right]^{-1} A \left[x_{k+1}(t) - x_k(t) \right] - \xi_{k+1}(t) \right\} \tag{2.61}$$

where $K(t)$ is the Riccati equation

$$\begin{aligned} K(t) &= A^T K(t+1)A + C^T QC - A^T K(t+1)B[B^T K(t+1)B + R]^{-1} B^T K(t+1)A \\ K(N) &= 0 \end{aligned} \tag{2.62}$$

and $\xi_{k+1}(t)$ is the predictive feedforward term

$$\begin{aligned} \xi_{k+1}(t) &= \left[I + K(t)BR^{-1}B^T \right]^{-1} \left[A^T \xi_{k+1}(t+1) + C^T Q e_k(t+1) \right] \\ \xi_{k+1}(N) &= 0 \end{aligned} \tag{2.63}$$

The main steps of this casual implementation could be described in the below steps:

1. Select suitable weighting Q, R and calculate $K(\cdot)$ using (2.62) in reverse time;
2. Choose an initial u_0 to control the system. Record $u_0(\cdot)$, $e_0(\cdot)$, $x_0(\cdot)$ and set $k = 1$.
3. Calculate $\xi_{k+1}(t)$ using (2.63) in reverse time;
4. Update the input using (2.61), and record $u_k(\cdot)$, $e_k(\cdot)$, $x_k(\cdot)$. Set $k = k + 1$ and go to step 3.

In the above, the standard NOILC algorithm for discrete time systems has been introduced and it shows great performance on convergence properties. In the control law, as the weighting matrix R decreases (with a fixed matrix Q), the convergence speed will increase. However, this increase is at the expense of the robustness of the system, which will cause problems when the model uncertainty is large. Amann et al. (1998) proposed a predictive extension of the standard NOILC algorithm, providing another way to improve the convergence speed. The idea of this predictive NOILC can be illustrated as follows:

Predictive Norm Optimal ILC:

The idea of predictive NOILC is to predict future input differences and tracking errors in the cost function. By looking into the future, it will have faster convergence than the standard NOILC algorithm. The performance index in predictive NOILC is replaced by:

$$J_{k+1,n}(\vec{u}_{k+1}) = \sum_{j=1}^{n} \lambda^{j-1} \left(\|e_{k+1,j}\|_Q^2 + \|u_{k+1,j} - u_{k+1,j-1}\|_R^2 \right) \tag{2.64}$$

in which $\vec{u}_{k+1} = [u_{k+1,1}^T \quad u_{k+1,2}^T \quad \cdots \quad u_{k+1,n}^T]^T$, $\lambda > 0$ is the weighting parameter (used to clarify the importance of future information) and $e_{k+1,j}$, $u_{k+1,j}$ are the (predicted) error, input for ILC iteration $k+j$ calculated during ILC iteration $k+1$. In Amann et al. (1998), it has been rigorously proved that the predictive cost function (2.64) result in faster convergence than the non-predictive performance index $J_{k+1}(u_{k+1})$ in Equation (2.54). For more details, please refer to Amann et al. (1998).

In addition to all of the above algorithms, there are other model-based ILC algorithms, for example, two dimensional ILC (Bolder and Oomen, 2016; Hladowski et al., 2010, 2011), H_∞ ILC (Amann et al., 1996c; Paszke et al., 2006). For more detail review of different model-based ILC algorithms, please refer to Bristow et al. (2006) and the reference therein.

2.2.4 Other Aspects in ILC

In this section, four important concepts of ILC are discussed: point-to-point task, constraint handling problem, robustness and non-minimum phase system.

2.2.4.1 Point-to-Point Task

In most of the literature, the reference $r(t)$ is considered as a signal defined on every single point in the whole trial length and the system is required to track this signal perfectly during the finite time. However, there exist some tasks in networked dynamical systems that only focus on the tracking of several intermediation points, e.g., Amazon warehouse multi robot 'pick and place' task. The Amazon warehouse multi robot 'pick and place' task contains a number of intelligent robots to perform the P2P task within a finite time interval. Starting from the initial position (e.g., the charging battery), all the robotic arms will travel to the 'pick' position (i.e., the storage unit) at time instant t_1; then move the objectives from the 'pick' position to the 'place' position (carrier car) at time instant t_2. This task lies emphasis on the consensus tracking at time instants t_1, t_2, and tracking at other time instants are of less interest. In this case, the normal reference signal r is changed into r^P and it can be written as

$$r^P = [r(t_1) \quad r(t_2)]^T \tag{2.65}$$

where $0 \leq t_1 < t_2 \leq t_N$. This type of task is called as point-to-point (P2P) task and it allows an infinite number of input choices. This freedom creates more design flexibility, on the other hand, it causes significant design difficulties. For more details about the P2P task in single-agent ILC design, please refer to Chen et al. (2018b); Freeman and Tan (2013); Jin (2018); Shen et al. (2017).

2.2.4.2 Constraint Handling Problem

System constraints are widely existing in practice, since they are often related to practical limitations (e.g., constraints of total input energy cost) or performance requirements (e.g., individual input saturation). As an example, the actuator in the system has a maximum input limitation. During the control process, if the actuator's applied input violates the maximum input limitation, it may damage the actuator and further affect the system. Hence, considering the ability to handle system constraints is important when designing an ILC algorithm.

To deal with the system constraint problem, a successive projection framework proposed in Chu and Owens (2010) can be used. Combining it into the ILC design, the resulting ILC algorithm has very nice convergence properties: the algorithm guarantees the monotonic convergence of the tracking error norm to a minimum (possible) solution (zero tracking error norm when perfect tracking is possible). For rigorous proof of the convergence properties, please refer to Chu and Owens (2010).

2.2.4.3 Robustness

Robustness discusses the tracking performance of control systems with various disturbances. For a robust ILC control algorithm, it not only guarantees the output converges to the reference trajectory under an ideal environment, but also generates the output signal to converge to the neighbouring area of the reference trajectory under bounded disturbances. For ILC, its robustness analysis can be separated into three categories:

- **Modelling error:** Modelling the dynamics of a real plant is one of the most important steps in analysing & designing a system, and it is difficult to obtain an accurate model in most cases. For a non-robust algorithm, an inaccurate model affects the convergence properties and causes the instabilities in control systems, which is undesirable in practice. To analyse the ILC algorithm's robustness against the model uncertainty, a number of methods have been proposed, e.g., Ge et al. (2018); Owens (2016); Son et al. (2016).

- **Effect of noise/disturbances:** In a real control environment, there exists a variety of random noise which affect the performance of the control system. Although adding a filter could eliminate most of the noise, there still exist some noise that are stubborn to be removed from the environment. In this situation, robustness to the effect of noise is required. To eliminate the noise, adding online observers or filters is an effective approach (Lee et al., 2000; Mandra et al., 2021).

- **Initial state error:** For most of the ILC algorithms, the initial state is assumed to be the same which means the control system can perfectly reset the initial state condition in each trial. However, perfect initial state setting is an ideal situation, and the initial state may change in practice. The non-coincident in initial state may cause the unstable in the control system when the robustness of the algorithm is not strong enough. Hence, a robust ILC algorithm is able to notice the changing of initial conditions in each trial, and then take some actions to eliminate the adverse effect. Some recent researches have considered the effect of the initial state error and designed robust ILC algorithms to eliminate this effect, e.g., Bi et al. (2018); Lan and Cui (2018).

2.2.4.4 Non-Minimum Phase Systems

Non-minimum phase systems are stable and causal systems, however, the inverses of their dynamics are causal and unstable (because the inverse model contain poles on the right half plane for continuous systems). Due to the poles on the right half plane, inverse-based ILC has difficulty to control the non-minimum phase systems. To illustrate this problem, consider the following inverse-based algorithm

$$u_{k+1}(t) = u_k(t) + G^{-1}e_k(t) \tag{2.66}$$

and the optimal input $u^*(t)$ that achieves perfect tracking is

$$u^*(t) = G^{-1}r(t). \tag{2.67}$$

When the system is a non-minimum phase system, the system inverse G^{-1} is unstable, and therefore the noise/disturbances will cause divergence of the tracking error in practice. On the other hand, since $r(t)$ is normally selected independent from system G, the optimal input $u^*(t)$ obtained from (2.67) contains the unstable models and grows to a large value, which is dangerous in many cases. Even for some algorithms which seem successfully been applied to non-minimum phase systems, although the tracking error norm converges fast at the beginning, the convergence speed will slow down after several trials (Owens et al., 2014). For more results of ILC for non-minimum phase systems, please refer to Noueili et al. (2017); Owens et al. (2014); Yoo et al. (2016).

2.3 ILC for High Performance Networked Systems

As mentioned in Section 2.1.4, the high performance requirements are crucially important for networked dynamical systems operating in a repetitive manner. In contrast to the traditional control methods which have significant difficulties to meet the requirements, ILC provides an alternative solution since its input is mainly updated by learning from the data (e.g. inputs and tracking errors) collected from previous executions of the task, without requiring accurate model information. Next, we will review several algorithms to illustrate the current situations of ILC for high performance networked dynamical systems.

2.3.1 Consensus Tracking

This section considers the consensus tracking problem of networked dynamical systems, i.e., all the subsystems in the networked dynamical system are required to track the same desired reference (and thus achieve consensus tracking) with high accuracy requirements. The difficulty here is that the reference information is only available to a subset of the network's subsystems. To give an example of the existing ILC algorithms for consensus tracking problem, the algorithm proposed in Devasia (2017) will be reviewed in the following.

Consider a networked dynamical system with p subsystems, where the dynamics of each subsystem is assumed to be a linear SISO system. The relationship between subsystem's individual input $u_{i,k}$ and subsystem's individual output $y_{i,k}$ in Laplace domain is described as

$$y_{i,k}(w) = G_i(w)u_{i,k}(w) \tag{2.68}$$

where k is the trial number; w is the frequency; i is the subsystem's index; G_i is the system matrix for i^{th} subsystem. The mission of the high performance consensus tracking problem is to achieve the following objective:

$$y_{i,k}(w) = r(w). \tag{2.69}$$

Note that the reference $r(w)$ is not generally known to all the subsystems, which makes the design non-trivial. For the subsystems know the reference information, they are called as 'leader' subsystems; otherwise, they are called as 'follower' subsystems.

For the 'follower' subsystems, Devasia (2017) design the updating law as follow:

$$u_{i,k+1}(w) = u_{i,k}(w) + \rho(w)G_i(w)\sum_{j\in\mathscr{N}_i}[y_{j,k}(w) - y_{i,k}(w)] \tag{2.70}$$

where $\rho(w)$ is a real-valued scalar. For the 'leader' subsystem, the updating law is defined as:

$$u_{i,k+1}(w) = u_{i,k}(w) + \rho(w)G_i(w)^{-1}\left\{\sum_{j\in\mathcal{N}_i}[y_{j,k}(w) - y_{i,k}(w)] + [r(w) - y_{i,k}(w)]\right\} \tag{2.71}$$

In updating laws (2.71), it shows that the 'leader' subsystems need to achieve the tracking mission of the desired reference and share the reference information with the neighbouring subsystems. For the 'follower' subsystems, the only mission is to keep consensus with their neighbouring subsystems (as shown in updating law (2.70)). By constantly communicating via the network, all the subsystems will have the information of the reference trajectory and then achieve the consensus tracking of the desired reference.

The above method uses the idea of inverse based ILC to solve the consensus tracking problem, which achieves the desired objective. Other types of ILC algorithms have been proposed for consensus tracking problem in networked dynamical systems: Proportional-Integral-Derivative type ILC algorithms which do not require model information, are proposed in Meng and Moore (2016); Meng et al. (2012, 2015a) for linear networked systems; Contraction mapping method based ILC algorithms are introduced in Yang et al. (2014, 2016) for continuous nonlinear systems with fixed graph and switching graphs, respectively; The Lyapunov function or composite energy function based adaptive ILC algorithms (which adaptively updating controller parameters) are introduced in Jin (2016); Li and Li (2013, 2015, 2016); Shen et al. (2019); Yang et al. (2015); Terminal ILC, focusing on the tracking performance at the final point, have been proposed in Bu et al. (2021); Meng and Jia (2011, 2012); Meng et al. (2013, 2014b, 2015c).

2.3.2 Formation Control

Formation control of networked dynamical systems is a control design problem that contains a group of subsystems working together to form a prescribed formation. Normally, there is no reference information in the formation control task, which results in infinite input solutions. To illustrate the current situation of ILC for high performance formation control, we will review the algorithm proposed in Meng and Jia (2014).

Consider a networked dynamical system with p subsystems, where i^{th} $(1 \le i \le p)$ subsystem's dynamics of each subsystem a linear multi-input-multi-output system

$$\begin{aligned} x_{i,k}(t+1) &= Ax_{i,k}(t) + Bu_{i,k}(t) \\ y_{i,k}(t) &= Cx_{i,k}, \qquad\qquad x_{i,k}(0) = x_{i0}, \end{aligned} \tag{2.72}$$

where k is the trial number; $t \in [0, N]$ represents the time; $u_{i,k}(\cdot) \in R^{n_u}$, $x_{i,k}(\cdot) \in R^{n_x}$, $y_{i,k}(\cdot) \in R^{n_y}$ are the input, state and output of subsystem i at trial k, respectively. Note that, for i^{th} subsystem, the initial condition remains the same for all the trials.

The control design objective of formation control problem is defined as

$$\lim_{k\to\infty} y_{i,k}(t) = r(t) + d_i(t) \tag{2.73}$$

where $d_i(t)$ is the desired trajectory deviation of subsystem i defining the desired formation, $r(t)$ is the desired reference trajectory that can be arbitrarily prescribed on $t \in [0, N]$. Note that, the reference $r(t)$ is not generally accessed to all the subsystems and the author define a nonnegative scalar w_i to represent this accessibility. When $r(t)$ is known by subsystem i, $w_i > 0$; otherwise, $w_i = 0$.

The input updating law is then given as

$$u_{i,k+1}(t) = u_{i,k}(t)+\Gamma q\left\{\sum_{j\in\mathscr{N}_i} \phi_{ij}a_{ij}\big[y_{j,k}(t)-y_{i,k}(t)+d_{ij}(t)\big]+\psi_i w_i\big[r(t)-y_{i,k}(t)+d_i(t))\big]\right\} \tag{2.74}$$

in which q is a forward time shift operation (e.g., $qx(t) = x(t+1)$), Γ is an $n_u \times n_y$ gain matrix, a_{ij} is the adjacency value, $d_{ij} = d_i - d_j$ is the desired formation difference between subsystems i and j. ψ_i and ϕ_{ij} are nonnegative learning gain, defined as

$$\psi_i = \begin{cases} > 0 & if\ j \in \mathscr{N}_i \\ = 0 & if\ j \notin \mathscr{N}_i \end{cases} \quad \& \quad \phi_{ij} = \begin{cases} > 0 & if\ w_i > 0 \\ = 0 & if\ w_i = 0 \end{cases} \tag{2.75}$$

The algorithm uses a modified Proportional type ILC to solve the formation control problem, and the simulation result in the paper shows that the desired objective can be achieved. There also exist other types of ILC algorithms proposed for formation control of networked dynamical systems: Proportional-Integral-Derivative (PID) type ILC control laws have been proposed for nonlinear networked dynamical systems in Liu and Jia (2012, 2015); Meng and Moore (2017); Meng et al. (2014a, 2015b); adaptive ILC control laws are proposed in Li and Li (2014a); Li et al. (2018); Li and Li (2014b) for nonlinear networked dynamical systems; a high-order internal model based ILC is developed in Xu et al. (2011); Yang et al. (2017).

2.3.3 Collaborative Tracking

Collaborative tracking aims to generate input to co-learn the desired reference trajectory, and the design objective in frequency domain can be defined as $\sum_{i=1}^{p} G_i(w)u_i(w) = r(w)$. Next, we will review the algorithm in Devasia (2016) to give a brief review of ILC algorithm for the collaborative tracking problem.

Consider a networked dynamical system with p subsystems, where the dynamics of each subsystem is assumed to be a linear SISO system. The relationship between subsystem's individual input $u_{i,k}$ and individual output $y_{i,k}$ in Laplace domain is defined as

$$y_{i,k}(w) = G_i(w)u_{i,k}(w). \tag{2.76}$$

Then, the overall output is defined as

$$y_k(w) = \sum_{i=1}^{p} y_{i,k}(w) = \sum_{i=1}^{p} G_i(w)u_{i,k}(w) \tag{2.77}$$

and the control design objective of cooperative tracking is described by

$$\sum_{i=1}^{p} G_i(w)u_{i,k}(w) = y_k(w) = r(w) \tag{2.78}$$

Based on the inverse-based framework, Devasia (2016) designed an ILC algorithm with time-partitioned update for collaborative tracking problem. In detail, the input updating law in frequency domain is proposed as

$$u_{i,k+1}(w) = u_{i,k}(w) + \rho_{i,k}(w)\hat{G}_i^{-1}(w)[r(w) - y_k(w)] \tag{2.79}$$

where $\rho_{i,k}(w)$ is a real-valued scalar and $\hat{G}_i(w)$ represents the nominal model of subsystem i.

By using the updating law (2.79) to achieve the collaborative tracking task, it guarantees the convergence performance for any unity partition (i.e., $\sum_{i=1}^{p} \rho_{i,k}(w) = 1$). For detailed proof of the convergence, please refer to Devasia (2016).

This inversed-based algorithm has been extended in Realmuto et al. (2018) for human-robot collaborative output tracking. In addition, Chen and Freeman (2020) propose a decentralised ILC framework with three model-based implementations for collaborative tracking problem and verify the framework on a wearable stroke rehabilitation technology. For more details, please see Chen and Freeman (2020); Realmuto et al. (2018).

2.3.4 Summary

From the above, we can see that most of the existing research has not fully investigated the potential of ILC for high performance networked dynamical systems. All the existing ILC algorithms have very poor scalability to control large scale networks and handle the dynamically growing network. For the monotonic convergence of the tracking/formation error norm (which is desirable in practice), most of the existing ILC algorithms cannot guarantee this properties, with the exception of a few algorithms that guarantee the monotonic convergence only when the system dynamics satisfy certain conditions.

Inverse based ILC methods explicitly utilise the model inverse to design the algorithm, and thus have difficulties to be applied to non-minimum phase networked dynamical systems. Besides, P2P task and constraint handling problem, representing a great number of practical applications, have not been fully investigated in existing ILC design for high performance networked dynamical systems.

As mentioned previously, an optimisation-based ILC algorithm (e.g., NOILC) can achieve various appealing system performances (e.g., monotonic convergence of the tracking error norm) by appropriately designing the performance index. Hence, designing novel optimisation-based ILC frameworks and implementing them in distributed/decentralised manner would be a feasible solution to solve the existing limitations. Among all the distributed implementation methods, there exists a widely used method called the alternating direction method of multipliers which has super convergence properties. The next section will give a review of this method.

2.4 The Alternating Direction Method of Multipliers

The alternating direction method of multipliers (ADMM) is a well-known distributed/decentralised optimisation algorithm and it was first presented in the 1970s (Gabay and Mercier, 1976). Boyd et al. (2011) review the ADMM systematically and apply its ideas to solve the lasso, group lasso, and other machine learning problems. In the following, we will review the basic idea, and convergence properties of ADMM based on Boyd et al. (2011).

2.4.1 Basic Idea of ADMM

ADMM is a method that blends the convergence properties of the method of multipliers with the decomposability of dual decomposition method, which has found many important applications in various areas, e.g., in machine learning, applied statistics and privacy preservation (Boyd et al., 2011; Cheng et al., 2017; Deng and Yin, 2016; Zhang et al., 2019a). To illustrate the general idea of ADMM, consider the following optimization problem

$$\begin{aligned} \text{minimize} \quad & f(x) + g(z) \\ \text{subject to} \quad & \mathbb{A}x + \mathbb{B}z = \mathbb{C} \end{aligned} \tag{2.80}$$

The augmented Lagrangian of the above problem is shown as

$$L_\rho(x, z, \gamma) = f(x) + g(z) + \gamma^T(\mathbb{A}x + \mathbb{B}z - \mathbb{C}) + \frac{\rho}{2}\|\mathbb{A}x + \mathbb{B}z - \mathbb{C}\|_2^2 \tag{2.81}$$

where γ is the dual variable and ρ is the penalty parameter. The ADMM updating steps can be shown as

$$x^{q+1} = \arg\min L_\rho(x, z^q, \gamma^q) \tag{2.82}$$

$$z^{q+1} = \arg\min L_\rho(x^{q+1}, z, \gamma^q) \tag{2.83}$$

$$\gamma^{q+1} = \gamma^q + \rho(\mathbb{A}x^{q+1} + \mathbb{B}z^{q+1} - \mathbb{C}) \tag{2.84}$$

For ADMM, it can guarantee the convergence of the residual, objective function and dual variable (under mild conditions). An analysis of its convergence properties will be given in the next section.

2.4.2 Convergence Properties of ADMM

To analyse the convergence properties of ADMM, the following two assumptions are required:

Assumption 2.1. *(Boyd et al., 2011) The (extended-real-valued) functions J: $R^n \to R \cup \{+\infty\}$ and $g : R^m \to R \cup \{+\infty\}$ are closed, proper, and convex.*

Assumption 2.2. *(Boyd et al., 2011) The unaugmented Lagrangian $\mathbb{L}_0$ has a saddle point.*

Assumption 2.1 guarantees that the x-updating law (2.82) and z-updating law (2.83) are solvable. Also, convex cost function guarantees the local optimization is the global optimization. Combining Assumptions 2.1 and 2.2, it shows that $\mathbb{L}_0(x^*, z^*, \gamma^*)$ is finite for any saddle point (x^*, z^*, γ^*), which indicates that (x^*, z^*) is a solution of the optimization problem (2.80). Under this situation, γ^* is dual optimal, and it follows that the primal optimal values equal to the dual optimal values, i.e., the strong duality holds.

Now, under Assumptions 2.1 and 2.2, ADMM has following convergence properties (Boyd et al., 2011):

- Residual convergence: $r^k \to 0$ as $k \to \infty$, i.e., the iterates approach feasibility.
- Objective convergence: $f(x^k) + g(z^k) \to p^*$ as $k \to \infty$, i.e., the objective function of the iterates approaches the optimal value.
- Dual variable convergence: $\gamma^k \to \gamma^*$ as $k \to \infty$, where γ^* is a dual optimal point.

It should be noted that, ADMM can guarantee the convergence for any positive penalty parameter ρ and this outstanding property makes ADMM widely used to solve the distributed/decentralised optimization problems. Rigorous analysis and proof of the convergence can be found in Boyd et al. (2011).

2.4.3 ADMM General Consensus Optimization with Regularization

For ADMM steps (2.82) – (2.84), it is the basic pattern mainly designed for centralised implementation. To deal with distributed problems (mainly for the consensus problem), we need to use the ADMM 'General Form Consensus Optimization with Regularization' pattern.

Consider the following distributed optimisation problem

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{p} f_i(x_i) + g(z) \\ &\text{subject to} \quad x_i - \widetilde{E}_i z = 0, \quad i = 1, \cdots, p \end{aligned} \tag{2.85}$$

where $x_i \in \mathbb{R}^{p_i}$ is the local input decision variable consisting of the component in the global variable $z \in \mathbb{R}^p$ and $\widetilde{E}_i$ is a matrix that selects the components from z that match the local variable x_i.

The augmented Lagrangian for (2.85) is shown as

$$L_\rho(x, z, \gamma) = \sum_{i=1}^{p} L_{\rho i}(x_i, z, \gamma_i) = g(z) + \sum_{i=1}^{p} \left[f_i(x_i) + \gamma_i^T (x_i - \widetilde{E}_i z) + \frac{\rho}{2} \|x_i - \widetilde{E}_i z\|^2 \right] \tag{2.86}$$

and ADMM solves the optimisation problem (2.85) by iteratively performing the following three steps

$$x_i^{q+1} = \arg\min L_{\rho i}(x_i, z^q, \gamma_i^q) \tag{2.87}$$

$$z^{q+1} = \arg\min L_\rho(x^{q+1}, z, \gamma^q) \tag{2.88}$$

$$\gamma_i^{q+1} = \gamma_i^q + \rho(x_i^{q+1} - \widetilde{E}_i z^{q+1}) \tag{2.89}$$

This 'General Form Consensus Optimization with Regularization' pattern is specifically suitable for the distributed problem (even with regularization on the global value), and thus, it will be used in later Chapters for the consensus tracking and formation control problem (with or without system constraints).

2.4.4 ADMM General Sharing Problem with Regularization

Besides the 'consensus' pattern mentioned in the last section, there exists another canonical problem in ADMM especially suitable for decentralised optimisation. The so called 'sharing' problem in ADMM has many similarities with the collaborative tracking problem, which we will review in the following.

To describe its general idea, we consider the following optimisation problem (Boyd et al., 2011)

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{p} f_i(x_i) + g\left(\sum_{i=1}^{p} z_i\right) \\ &\text{subject to} && \widetilde{E}_i x_i - z_i = 0, \quad i = 1, \cdots, p \end{aligned} \tag{2.90}$$

where $x_i \in \mathbb{R}^N$ denotes the local variable, $\widetilde{E}_i \in \mathbb{R}^{N\times N}$ is the corresponding matrix, $z_i \in \mathbb{R}^N$ denotes the related element in the global variable $z = [z_1^T \quad z_2^T \quad \cdots \quad z_p^T]^T$.

For problem (2.90), its augmented Lagrangian is defined as

$$L_\rho(x_i, z, \mu_i) = \sum_{i=1}^{p} f_i(x_i) + g\left(\sum_{i=1}^{p} z_i\right) + \frac{\rho}{2}\sum_{i=1}^{p} \|\widetilde{E}_i x_i - z_i + \mu_i\|^2 \tag{2.91}$$

where ρ is the penalty parameter, and $\mu_i = \gamma_i/\rho$ is the scaled dual value. Then, ADMM will perform the following three steps iteratively to solve problem (2.90)

$$x_i^{q+1} = \arg\min_{x_i} \left[f_i(x_i) + \frac{\rho}{2}\|\widetilde{E}_i x_i - z_i^q + \mu_i^q\|^2 \right] \tag{2.92}$$

$$z^{q+1} = \arg\min_{z} \left[g\left(\sum_{i=1}^{p} z_i\right) + \frac{\rho}{2}\sum_{i=1}^{p} \|\widetilde{E}_i x_i^{q+1} - z_i + \mu_i^q\|^2 \right] \tag{2.93}$$

$$\mu_i^{q+1} = \mu_i^q + \widetilde{E}_i x_i^{q+1} - z_i^{q+1} \tag{2.94}$$

Note that, z-step requires the controller to solve a problem in pN variables, which creates significant computational complexity. To save the computational complexity, we introduce auxiliary variables $\overline{Ex} = \frac{1}{p}\sum_{i=1}^{p} \widetilde{E}_i x_i$ and $\bar{z} = \frac{1}{p}\sum_{i=1}^{p} z_i$ to the above equations. Then, the ADMM steps can be simplified as (more details can be found in Boyd et al. (2011))

$$x_i^{q+1} = \arg\min_{x_i} \left[f_i(x_i) + \frac{\rho}{2}\|\widetilde{E}_i x_i - \widetilde{E}_i x_i^q + \overline{Ex}^q - \bar{z}^q + \mu^q\|^2 \right] \tag{2.95}$$

$$\bar{z}^{q+1} = \arg\min_{\bar{z}} \left[g(p\bar{z}) + \frac{\rho}{2}p\|\overline{Ex}^{q+1} - \bar{z} + \mu^q\|^2 \right] \tag{2.96}$$

$$\mu^{q+1} = \mu^q + \overline{Ex}^{q+1} - \bar{z}^{q+1}. \tag{2.97}$$

For these ADMM steps, the z-update step only requires solving a problem in N variables and each subsystem's scale dual variables μ_i are the same (and hence they are replaced by a single variable μ), which greatly saves the computational complexity.

2.4.5 Discussion about ADMM in Networked ILC Design

In this thesis, we use ADMM as the distributed implementation method because it has several outstanding advantages for networked ILC design:

- First and foremost, ADMM is extremely suitable for networked ILC design. As shown in Sections 2.4.3 and 2.4.4, ADMM can be used to solve 'consensus' and 'sharing' optimisation problem in a distributed manner if the performance index is separable. In the later chapters, it can be seen that due to ILC problem have finite horizon, we can formulate the networked ILC control problem into the form of a optimisation problem with separable global performance index. By separating the global performance index into several local performance indexes (for different subsystems) and using the corresponding matrix $\widetilde{E}_i$ (which denotes the network structure) to link the global decision variable with the local input plan, the global ILC design problem can be implemented in a fully distributed manner, which solves large-scale problems surprisingly fast. From the computational point of view, ILC is an off-line method and it can use all the data from previous trials to update the input between two continuous ILC trials, which would not meet the computational problem during the control process.

- ADMM can guarantee the convergence properties under very mild conditions, contrasting with most distributed algorithms, e.g., dual decomposition (Boyd et al., 2011). While the dual decomposition method implements an optimisation problem distributively, the convergence to the optimal solution is only guaranteed for proper choice of step sizes and under rather strong assumptions on the original problem. By contrast, ADMM can guarantee the convergence for any positive penalty parameters in a convex problem, and even for non-convex and non-smooth performance index (Wang et al., 2019).

- Benefited from the regularization term, ADMM has the capacity to handle the system constraints. Especially, if the global constraint set can be separated into smaller local constraint set, ADMM is able to update the global value in parallel, which greatly solves the computation complexity. Moreover, ADMM is robust to computation errors and noise (Simonetto and Leus, 2014) and therefore it has strong industrial practicability.

2.5 Summary

This chapter provides a general literature review of networked dynamical systems and ILC. Section 2.1 gives a general background of the networked dynamical systems, and discusses the consensus and formation control problem in networked dynamical systems. Then in Section 2.2, the concepts/approaches of ILC are reviewed and some important issues in ILC are introduced. Section 2.3 reviews the existing ILC algorithms for high performance networked dynamical systems and discusses the limitations of existing approaches. In Section 2.4, the basic concept of ADMM is reviewed, with the 'consensus' and 'sharing' problem introduced.

From the literature that have been reviewed, we could see that high performance networked dynamical systems, which work in a repetitive manner and require high control performance, have found a number of applications in different fields. However, these types of networked dynamical systems have not been fully studied in the ILC design and most of the existing ILC algorithms have the following limitations:

- They have very poor scalability, the majority of the distributed/decentralised ILC designs have difficulty controlling very large scale networked dynamical systems. In addition, none of the ILC algorithms can deal with the dynamically growing network, i.e., When the size of network growing during the control process, the algorithms need to be redesigned and/or the parameters need to be re-tuned.
- Most of the existing ILC designs for networked dynamical systems have limited convergence performance, the monotonic convergence of the tracking error norm (that is desirable in practice) is either not guaranteed, or only guaranteed when the controller satisfies certain conditions.
- None of the existing ILC algorithms consider the system constraint (that is widely existing in practice) in the design.
- All of the existing ILC design lack of the ability to control the general P2P task.

Hence, novel distributed/decentralised ILC algorithms that address these limitations have still been waiting to explore. In the next six chapters, we will introduce novel distributed/decentralised ILC algorithms to solve different control design problems in networked dynamical systems.

Chapter 3

Distributed NOILC for Consensus Tracking Problem

Consensus tracking is a design problem that requires all the subsystems perfectly track the same desired time varying reference with a high accuracy requirement. The difficulty here is that this reference information is only available to a subset of the network's subsystems. A number of ILC algorithms have been proposed for consensus tracking problem: distributed Proportional–Integral–Derivative (PID) type ILC control laws are proposed in Meng et al. (2012, 2015a); Yang et al. (2017); an adaptive ILC is introduced in Jin (2016) for a class of nonlinear networked dynamical systems; a model-inverse based ILC algorithm is proposed in Devasia (2017); terminal ILC, focusing on the tracking performance at the final point, have been proposed in Bu et al. (2021); Meng and Jia (2011, 2012); Meng et al. (2013, 2014b, 2015c). These algorithms have shown great advantages to utilise ILC for high performance consensus tracking problem.

However, most of the existing articles for consensus tracking problem have the following limitations: (1) the majority of the existing ILC algorithms have difficulties to be applied to large scale networked dynamical systems, since their computational complexity of the convergence condition depends on the size of the network; (2) all the existing algorithms lack of the capacity to deal with a dynamically growing network, i.e., when the size of network increased during the control process, the algorithms need to be redesigned or the parameters need to be re-tuned; (3) monotonic convergence of the tracking error norm (which is desired in practice) is either not guaranteed or guaranteed only when the controller satisfies certain conditions; (4) general point-to-point (P2P) problems, which focus on the tracking performance on the intermediate time instants, have not been considered in literature. Moreover, the robustness issue and heterogeneous network, which are of great practice relevance, have only been considered in few articles.

In this chapter, we address the aforementioned limitations by proposing a novel distributed norm optimal ILC (NOILC) algorithm for consensus tracking of networked dynamical systems working in a repetitive manner. The proposed algorithm has a number of advantages: it can guarantee the tracking error norm converge monotonically to zero (without any requirement on the controller), and has certain resistance to model uncertainties; it can be extended to solve the P2P consensus tracking problem and maintain the good convergence, robustness properties; it has nice scalability and can be applied to large scale networked systems; it can be applied to both homogeneous and heterogeneous networks, as well as non-minimum phase systems. This chapter is based on the work from Chen and Chu (2019c,a).

The chapter is organised as follows: Section 3.1 introduces the system dynamics, network topology and then defines the ILC design problem based on these formulations. In Section 3.2, a NOILC framework for consensus tracking is proposed and its convergence properties are analysed rigorously. Section 3.3 develops a distributed implementation of the proposed algorithm using ADMM. Moreover, we give a feedback plus feedforward implementation for the distributed algorithm and show an efficient way to find the best penalty parameters in ADMM. In Section 3.4, we analyse the algorithm's robustness properties and characterise the tolerated model uncertainty's range analytically both in time & frequency domain. Section 3.5 extend the proposed algorithm to solve the P2P task and then analyse the P2P algorithm's convergence and robustness properties. In Section 3.6, numerical simulations are presented to demonstrate the effectiveness of the algorithm, and finally summaries are given in Section 3.7.

3.1 Problem Formulation

In this section, the subsystem's dynamics is formulated using an abstract Hilbert space representation and the formulation is presented for discrete time, linear time invariant (LTI), single input single output (SISO) system for simplicity. In addition, we use graph structure to represent the network topology and then provide the ILC design problem.

3.1.1 System Dynamics

Consider a networked dynamical system (either homogeneous or heterogeneous) including p subsystems, with i^{th} $(i = 1, \cdots, p)$ subsystem's dynamics described as a discrete time, LTI, SISO system. Then, i^{th} subsystem's dynamics can be represented using the following state space model

$$\begin{aligned} x_{i,k}(t+1) &= A_i x_{i,k}(t) + B_i u_{i,k}(t), \qquad x_{i,k}(0) = x_{i,0} \\ y_{i,k}(t) &= C_i x_{i,k}(t) \end{aligned} \tag{3.1}$$

where $t \in [0, N]$ is the time index; k is the ILC trial index; $y_{i,k} \in \mathbb{R}$, $u_{i,k} \in \mathbb{R}$, $x_{i,k} \in \mathbb{R}^{n_i}$ (n_i is the order of subsystem i) are output, input, state of i^{th} subsystem at trial k; A_i,

B_i, C_i are state space matrices with appropriate dimensions. The networked dynamical system is working in a repetitive manner performing the same task within time interval $[0, N]$. At $t = N + 1$, the system resets the time $t = 0$ and the state $x_{i,k}$ to the initial state $x_{i,0}$, and executes the same task again.

The networked system is required to achieve the high performance consensus tracking, i,e., all the subsystems are required to track a given reference $r(t)$ with high accuracy. Note that, the reference trajectory is only known by part of the subsystems, as an example, in a group of UAVs, while the front subsystems have access to the reference information, the rear subsystems have to wait for the information transited from the front subsystems.

To facilitate later ILC design, we use a 'lifted matrix form' to describe the subsystem's dynamics (Hatonen et al., 2004). Assume the relative degree of each subsystem is unity (i.e., $C_iB_i \neq 0$) and define $u_{i,k}$, $y_{i,k}$, r as the 'lifted form' of i^{th} subsystem's input $u_{i,k}(t)$, output $y_{i,k}(t)$, reference signal $r(t)$, i.e.

$$\begin{aligned} u_{i,k} &= [u_{i,k}(0) \quad u_{i,k}(1) \quad \cdots \quad u_{i,k}(N-1)]^T \in \mathcal{U}_i \\ y_{i,k} &= [y_{i,k}(1) \quad y_{i,k}(2) \quad \cdots \quad y_{i,k}(N)]^T \in \mathcal{Y}_i \\ r &= [r(1) \quad r(2) \quad \cdots \quad r(N)]^T \in \mathcal{Y}_i \end{aligned} \tag{3.2}$$

where i^{th} subsystem's input space $\mathcal{U}_i = \mathbb{R}^N$ and output space $\mathcal{Y}_i = \mathbb{R}^N$ are defined with inner products and induced norms

$$\begin{aligned} \langle u, v\rangle_R &= u^T R v, \qquad \|u\|_R = \sqrt{\langle u, u\rangle_R} \\ \langle x, y\rangle_Q &= x^T Q y, \qquad \|y\|_Q = \sqrt{\langle y, y\rangle_Q} \end{aligned} \tag{3.3}$$

and R, Q are symmetric positive definite matrices. In 'lifted matrix form', the system model (3.1) can be written as follows

$$y_{i,k} = G_i u_{i,k} + d_i \tag{3.4}$$

where the system matrix G_i is denoted as

$$G_i = \begin{bmatrix} C_iB_i & 0 & \cdots & 0 \\ C_iA_iB_i & C_iB_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_iA_i^{N-1}B_i & C_iA_i^{N-2}B_i & \cdots & C_iB_i \end{bmatrix} \tag{3.5}$$

and $d_i \in \mathbb{R}^N$ represents the response of initial conditions

$$d_i = \begin{bmatrix} C_iA_ix_{i,0} & C_iA_i^2x_{i,0} & \cdots & C_iA_i^Nx_{i,0} \end{bmatrix}^T \tag{3.6}$$

Introduce $\vec{u}_k$, $\vec{y}_k$, $\vec{d}$, $\vec{r}$ as the global form of $u_{i,k}$, $y_{i,k}$, d_i, and r, i.e.

$$\begin{aligned}\vec{u}_k &= \begin{bmatrix} u_{1,k}^T & u_{2,k}^T & \cdots & u_{p,k}^T \end{bmatrix}^T \in \mathcal{U} \\ \vec{y}_k &= \begin{bmatrix} y_{1,k}^T & y_{2,k}^T & \cdots & y_{p,k}^T \end{bmatrix}^T \in \mathcal{Y} \\ \vec{d} &= \begin{bmatrix} d_1^T & d_2^T & \cdots & d_p^T \end{bmatrix}^T \in \mathcal{Y} \\ \vec{r} &= \begin{bmatrix} r^T & r^T & \cdots & r^T \end{bmatrix}^T \in \mathcal{Y}\end{aligned} \tag{3.7}$$

where the global input space $\mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2 \times \cdots \times \mathcal{U}_p$ and the global output space $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_p$ are defined with inner products and induced norms

$$\begin{aligned}\langle \vec{u}, \vec{v} \rangle_{\vec{R}} &= \sum_{i=1}^{p} u_i^T R v_i, \qquad \|\vec{u}\|_{\vec{R}} = \sqrt{\langle \vec{u}, \vec{u} \rangle_{\vec{R}}} \\ \langle \vec{x}, \vec{y} \rangle_{\vec{Q}} &= \sum_{i=1}^{p} x_i^T Q y_i, \qquad \|\vec{y}\|_{\vec{Q}} = \sqrt{\langle \vec{y}, \vec{y} \rangle_{\vec{Q}}}\end{aligned} \tag{3.8}$$

in which $\vec{Q} = \text{diag}(Q, Q, \cdots, Q)$, $\vec{R} = \text{diag}(R, R, \cdots, R)$ are symmetric positive definite matrices, and $\times$ represents the Cartesian product.

Then, the global system model can be written as

$$\vec{y}_k = \mathbb{G}\vec{u}_k + \vec{d} \tag{3.9}$$

where $\mathbb{G} = \text{diag}\,(G_1, G_2, \cdots, G_p)$.

The high performance consensus tracking problem can be stated as finding a proper input $\vec{u}_k$ such that the output $\vec{y}_k$ tracks the reference $\vec{r}$ precisely. For a single-agent system, the reference is fully accessed by the system, and hence it is straightforward to find the solution. However, in the networked dynamical system, the limited accessibility makes the control design non-trivial.

3.1.2 Network Topology

In this chapter, the network topology is represented using an undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V} = \{1, 2, ..., p\}$ is the vertex set, and $\mathscr{E} \subset \mathscr{V} \times \mathscr{V}$ is the set of pairs of vertexes called edges. If two vertexes have an edge between them, they are called as neighbours. The i^{th} subsystem's neighbours set is denote as $\mathscr{N}_i := \{j : (i, j) \in \mathscr{E}\}$.

To represent the topology relationship between different subsystems, we introduce the adjacency matrix $\mathscr{A} = [a_{ij}]$, with its element a_{ij} defined as

$$a_{ij} = \begin{cases} W_{ij} & if\ (i, j) \in \mathscr{E} \\ 0 & otherwise \end{cases} \tag{3.10}$$

where weight W_{ij} is often considered as the connection strength of the edge , i.e., subsystem i will put more emphasis on the information transformation with subsystem j. Given an example, in the transportation system, there exist some subsystems have direct access to the reference trajectory (i.e., 'leader' subsystems), their neighbours (without access to the reference) know the leaders have information about the desired objective and hence they will put more emphasize on communicating information with the 'leader' subsystems to improve its tracking efficiency. Based on the i^{th} node's neighbours set, the degree of a node i is defined as $d(i) = \sum_{j=1}^{p} a_{ij}$ and it follow by the degree matrix $\mathscr{D} = diag(d(1), d(2), \cdots d(p))$. Using the definition of adjacency matrix and degree matrix, the Laplacian matrix is defined as $L = \{l_{ij}\} := \mathscr{D} - \mathscr{A}$, which is a real symmetric matrix with element l_{ij} defined as below

$$l_{ij} = \begin{cases} -W_{ij} & if \quad j \in \mathscr{N}_i \\ \sum_{j\in\mathscr{N}_i} W_{ij} & if \quad j = i \\ 0 & otherwise \end{cases} \tag{3.11}$$

For the high performance consensus tracking problem, only few subsystems have access to the reference signal, and hence we introduce a reference-accessibility matrix $D = \text{diag}\{\mathcal{D}_{ii}\}$ to represent this relationship. The diagonal element $\mathcal{D}_{ii}$ is denoted as

$$\mathcal{D}_{ii} = \begin{cases} 1 & if \text{ subsystem } i \text{ has access} \\ 0 & if \text{ subsystem } i \text{ does not have access} \end{cases} \tag{3.12}$$

Remark 3.1. *In the design, matrices $\mathscr{A}$ and L indicate the topology relationship between different subsystems, while the reference-accessibility matrix represent the relationship between subsystems and reference trajectory, which form the basic structure to describe networked dynamical systems. For consensus tracking problem, the reference information will first transfer to subsystems with $\mathcal{D}_{ii} = 1$ (which called as 'leader' subsystems). After that, the 'leader' subsystem will transfer the reference information to other UAVs through the network (follows the topology relationship indicated on the Laplacian matrix).*

In this chapter, the following assumptions are required to achieve the desired objective:

Assumption 3.1. *At least one subsystem has access to the reference r, i.e., $\exists \mathcal{D}_{\rangle\rangle} \neq 0$.*

Assumption 3.2. *The discussed undirected graph $\mathscr{G}$ is connected, i.e., there exists at least one path from one node to another nodes (Bullo, 2018).*

Remark 3.2. *Assumption 3.1 guarantees the reference information is available in the system. However, even if not reference in the system, the proposed algorithm can still be applied to achieve the consensus (without tracking any reference signal).*

Remark 3.3. *Assumption 3.2 is commonly used for the consensus tracking task of the networked dynamical systems. If there exists a subsystem not connected with any other subsystems, it is impossible for it to achieve consensus with other subsystems.*

Under the above assumptions, the Laplacian matrix has the following properties, which will be used later in analysing the algorithm's convergence properties.

Lemma 3.1. *(Devasia, 2017) Under Assumption 3.2*

(1) The undirected graph has a real symmetric Laplacian matrix L. Correspondingly, $\{\lambda_i\}_{i=1}^p$, the eigenvalues of L are real, along with a set of orthonormal eigenvector $\{V_i\}_{i=1}^p$

$$V_i^T V_j = \delta_{ij} \tag{3.13}$$

where δ_{ij} represents Kronecker delta:

$$\delta_{ij} = \begin{cases} 1 & if\ i = j \\ 0 & otherwise \end{cases} \tag{3.14}$$

(2) For a Laplacian matrix L of any connected graph $\mathscr{G}$, its rank equals to $p-1$;

(3) The vector $\mathbb{1} = \{1, \cdots, 1\}^T$ is both a right and left eigenvector of the Laplacian L with eigenvalue 0, i.e.,

$$\mathbb{1}^T L = 0\mathbb{1}^T, \qquad L\mathbb{1} = 0\mathbb{1} \tag{3.15}$$

(4) Every non-zero eigenvalues of L should be positive and bounded, i.e.,

$$0 = \lambda_1 < \lambda_2 \le \cdots \le \lambda_p \le 2d_L \tag{3.16}$$

where d_L represents the maximum degree of $\mathscr{G}$

$$d_L = \max_{i \le i \le p} |l_{ii}| \tag{3.17}$$

Using Lemma 3.1 yields the following Lemma:

Lemma 3.2. *(Devasia, 2017) The augmented Laplacian $L_D := [L + D]$ is a real symmetric and positive-definite matrix, whose eigenvalues $\{\lambda_{D,i}\}_{i=1}^p$ are real and positive, corresponding to a set of orthonormal eigenvectors $\{V_{D,i}\}_{i=1}^p$*

$$V_{D,i}^T V_{D,j} = \delta_{ij} \tag{3.18}$$

where δ_{ij} is the Kronecker delta.

3.1.3 ILC for High Performance Consensus Tracking

The ILC design problem in this chapter can be stated as finding an appropriate input updating law in the form

$$\vec{u}_{k+1} = f(\vec{u}_k, \vec{e}_k) \tag{3.19}$$

such that i^{th} subsystem achieves the perfect tracking of the reference trajectory and the resulting input converges to the optimal solution as $k \to \infty$, i.e.

$$\lim_{k\to\infty} y_{i,k} = r, \quad \lim_{k\to\infty} u_{i,k} = u_i^*, \quad i = 1, 2, \cdots, p \tag{3.20}$$

where $\vec{e}_k = \vec{r} - \vec{y}_k$ is the 'virtual' tracking error and only part of it is available in ILC design (since not every subsystems have access to the reference information), u_i^* is i^{th} subsystem's optimal solution for the high performance consensus tracking problem.

For traditional ILC algorithm, the difficulty to achieve (3.20) is that they require full information of $\vec{e}_k$, which is generally unavailable in practice. To solve this problem, we will design a distributed ILC algorithm in the following sections.

3.2 NOILC Framework for Consensus Tracking Problem

In this section, we propose a novel NOILC framework for the high performance consensus tracking problem, and analyse the framework's convergence properties rigorously.

3.2.1 Algorithm Description

In principle, the NOILC framework aims at minimising a particular performance index (formed by the input difference norm and tracking error norm) to achieve the monotonic convergence property (the details are given in Section 2.2.3.3). However, since the full information of $\vec{e}_k$ is difficult to be obtained in the networked systems, we need to extend the idea of classical NOILC by properly defining the consensus tracking error as

$$\hat{e}_{k+1} := (\boldsymbol{L} + \boldsymbol{D})\vec{e}_{k+1} \tag{3.21}$$

in which $\boldsymbol{D} = D \otimes I_N$, $\boldsymbol{L} = L \otimes I_N$, $\otimes$ is the Kronecker product, I_N denotes a N by N identity matrix. In (3.21), it involves the calculation of $\boldsymbol{D}e_{k+1}$ and $\boldsymbol{L}e_{k+1}$. Minimising $\boldsymbol{D}e_{k+1}$ guarantees the subsystem with access to the reference (i.e., $\mathcal{D}_{ii} \neq 0$) can perfectly track the reference trajectory. At the same time, the minimisation of $\boldsymbol{L}e_{k+1}$ guarantees the reference information is transferring within the network, and hence achieve the desired consensus tracking objective. It is worth pointing out that solving $\boldsymbol{L}\vec{e}_{k+1}$ only requires the output difference between neighbouring subsystems (since $\boldsymbol{L}\vec{r} = 0$), and hence avoiding the requirement of full information of $\vec{e}_k$.

Now, using NOILC structure, we have the following algorithms:

Algorithm 3.1. *For any initial choice of global input $\vec{u}_0$ and associated consensus tracking error $\hat{e}_0$, the input sequence $\{\vec{u}_{k+1}\}_{k\geq 0}$ generated as follows*

$$\vec{u}_{k+1} = \arg\min\{J_{k+1}(\vec{u}_{k+1})\} \tag{3.22}$$

with $J_{k+1}(\vec{u}_{k+1})$ defined as

$$J_{k+1}(\vec{u}_{k+1}) = \|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}} + \|(\boldsymbol{L}+\boldsymbol{D})\vec{e}_{k+1}\|^2_{\vec{Q}} \tag{3.23}$$

in which $\vec{e}_{k+1} = \vec{e}_k - \mathbb{G}(\vec{u}_{k+1} - \vec{u}_k)$, provides a solution for the high performance consensus tracking problem, i.e.

$$\lim_{k\to\infty} y_{i,k} = r, \quad \lim_{k\to\infty} u_{i,k} = u_i^*, \quad i = 1, 2, \cdots, p. \tag{3.24}$$

Note that, when the number of subsystems is small, Algorithm 3.1 can be implemented in a centralised manner by directly finding the stationary point of the cost function (3.23). The input updating law is shown as follows

$$\vec{u}_{k+1} = \vec{u}_k + \left[\vec{R}^{-1}\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G} + I_{pN}\right]^{-1}\vec{R}^{-1}\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\vec{e}_k. \tag{3.25}$$

However, centralised implementation requires significant computational complexity for large scale networked systems, which can be difficult and expensive to implement in practice. Later, we will develop a distributed implementation in section 3.3.

3.2.2 Convergence Properties of the Algorithm 3.1

By designing a novel performance index, Algorithm 3.1 has nice convergence properties as shown in the following:

Theorem 3.1. *Given any initial input $\vec{u}_0$ with initial tracking error $\vec{e}_0$, the consensus tracking error norm $\|\hat{e}_k\|_{\vec{Q}} := \|(\boldsymbol{L}+\boldsymbol{D})\vec{e}_k\|_{\vec{Q}}$ converges monotonically to zero, i.e.*

$$\|\hat{e}_{k+1}\|_{\vec{Q}} \leq \|\hat{e}_k\|_{\vec{Q}}, \qquad \lim_{k\to\infty} \vec{e}_k = 0. \tag{3.26}$$

Consequently, the objective is achieved and each subsystem's input converges to optimal solution as $k \to \infty$, i.e.

$$\lim_{k\to\infty} y_{i,k} = r, \quad \lim_{k\to\infty} u_{i,k} = u_i^*, \quad i = 1, 2, \cdots, p \tag{3.27}$$

Proof. At trial $k+1$, the input of the proposed algorithm is generated by solving the following optimization problem

$$\vec{u}_{k+1} = \arg\min\{J_{k+1}(\vec{u}_{k+1})\} \tag{3.28}$$

For the above problem, the choice of $\vec{u}_{k+1} = \vec{u}_k$ gives a potentially non-optimal solution

$$J_{k+1}(\vec{u}_k) = \|\hat{e}_k\|^2_{\vec{Q}} \tag{3.29}$$

from which it follows that for optimal $\vec{u}_{k+1}$, we have

$$J_{k+1}(\vec{u}_{k+1}) \leq J_{k+1}(\vec{u}_k), \tag{3.30}$$

that is

$$\|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}} + \|\hat{e}_{k+1}\|^2_{\vec{Q}} \leq \|\hat{e}_k\|^2_{\vec{Q}} \tag{3.31}$$

As the value of $\|\vec{u}_{k+1} - \vec{u}_k\|^2$ is non-negative, we have

$$\|\hat{e}_{k+1}\|^2_{\vec{Q}} \leq \|\hat{e}_k\|^2_{\vec{Q}}, \quad k = 0, 1, \cdots, \infty \tag{3.32}$$

From (3.30), we also have (by summing up both sides from $k = 0$ to k)

$$\sum_{k=0}^{k} \|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}} \leq \|\hat{e}_0\|^2_{\vec{Q}} - \|\hat{e}_{k+1}\|^2_{\vec{Q}} \leq \|\hat{e}_0\|^2_{\vec{Q}} \tag{3.33}$$

Note that, the sequence $S_k := \sum_{k=0}^{k} \|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}}$ is a bounded non-decreasing sequence and use Monotone Convergence Theorem gives

$$\lim_{k\to\infty} S_{k+1} = \|\hat{e}_0\|^2_{\vec{Q}}, \tag{3.34}$$

and hence

$$\lim_{k\to\infty} S_{k+1} - S_k = 0 \tag{3.35}$$

It follows that

$$\lim_{k\to\infty} \|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}} = 0, \tag{3.36}$$

hence

$$\lim_{k\to\infty} \vec{u}_{k+1} - \vec{u}_k = 0 \tag{3.37}$$

Solving the optimization problem (3.22) gives

$$\begin{aligned}\vec{u}_{k+1} - \vec{u}_k &= \left[\vec{R}^{-1}\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G} + I_{pN}\right]^{-1} \vec{R}^{-1}\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\vec{e}_k \\ &= \left[\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G} + \vec{R}\right]^{-1} \mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\vec{e}_k\end{aligned}$$

Note that matrices $\mathbb{G}^T$ and $\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G} + \vec{R}$ are invertible, we have

$$\lim_{k\to\infty} (\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\vec{e}_k = 0 \tag{3.38}$$

Note that matrix $L + D$ is positive definite (as shown in Lemmas 3.1 & 3.2). Also, matrix $\vec{Q}$ is positive definite, hence $\vec{e}_k = 0$ as $k \to \infty$. As a result, $\lim\limits_{k\to\infty} u_{i,k} = u_i^*$. That completes the proof. □

Theorem 3.1 shows that $\|\hat{e}_{k+1}\|_{\vec{Q}}$ converges monotonically to zero and each subsystem's input $u_{i,k}$ converges to the optimal solution as $k \to \infty$, which is appealing in practice. For centralised implementation, ILC can use a central controller to generate the control input in once and it is easy to be implemented for a small scale networked dynamical system. However, real networked dynamical systems normally contain a great number of subsystems, and hence it is impractical to design a central controller for large scale networked dynamical systems. Next, we will introduce distributed methods to implement Algorithm 3.1 in a more reasonable manner for networked dynamical systems with great number of subsystems.

3.3 Distributed ILC Algorithm

This section develops a distributed implementation of the Algorithm 3.1 using the 'consensus' formulation in ADMM. In the distributed algorithm, a centralised controller (requiring the global information) is not required any more, and instead, each subsystem only needs to optimise its local cost function and exchange information with other subsystems via the network (which eventually obtain the same solution as the centralised result). Since each local input update can be done in parallel using multiple processors, Algorithm 3.1 is applicable to large scale networked dynamical systems. In the next section, we will first give a brief review of the 'consensus' formulation in ADMM.

3.3.1 The Alternating Direction Method of Multipliers

ADMM is a well-known optimisation method and its general form consensus optimization pattern can solve the following distributed optimisation problem

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{p} J_i(x_i) \\ &\text{subject to} \quad x_i - \widetilde{E}_i z = 0, \quad i = 1, \cdots, p \end{aligned} \tag{3.39}$$

where $x_i \in \mathbb{R}^{p_i}$ represents the local variable (p_i is the amount of corresponding element) and $\widetilde{E}_i$ is the corresponding matrix that map the global value $z \in \mathbb{R}^p$ to x_i.

Define the augmented Lagrangian of problem (3.39) as

$$L_\rho(x, z, \gamma) = \sum_{i=1}^{p} L_{\rho i}(x_i, z, \gamma_i) = \sum_{i=1}^{p} \left[J_i(x_i) + \gamma_i^T (x_i - \widetilde{E}_i z) + \frac{\rho}{2} \|x_i - \widetilde{E}_i z\|^2 \right] \tag{3.40}$$

Then, ADMM solve the problem (3.39) by iteratively performing the following steps

$$x_i^{q+1} = \arg\min L_{\rho i}(x_i, z^q, \gamma_i^q) \tag{3.41}$$
$$z^{q+1} = \arg\min L_\rho(x^{q+1}, z, \gamma^q) \tag{3.42}$$
$$\gamma_i^{q+1} = \gamma_i^q + \rho(x_i^{q+1} - \widetilde{E}_i z^{q+1}) \tag{3.43}$$

where q is ADMM iteration index, ρ is the penalty parameter and γ_i is the local dual variable.

ADMM has good convergence properties: it can guarantee the convergence to the optimal solution for any $\rho > 0$, which contrasts with other distributed optimisation methods. For rigorous proof of the convergence, please refers to Boyd et al. (2011).

3.3.2 Distributed Implementation of Algorithm 3.1

Note that, the cost function $J_{k+1}(\vec{u}_{k+1})$ in (3.23) can be written into the following separate form

$$J_{k+1}(\vec{u}_{k+1}) = \sum_{i=1}^{p} J_{i,k+1}(\vec{u}_{i,k+1}) \tag{3.44}$$

where i^{th} subsystem's local cost function $J_{i,k+1}(\vec{u}_{i,k+1})$ is defined as

$$J_{i,k+1}(\vec{u}_{i,k+1}) = \|(\boldsymbol{L}_i + \boldsymbol{D}_i)(\vec{r}_i - \vec{G}_i\vec{u}_{i,k+1} - \vec{d}_i)\|^2_{\boldsymbol{Q}_i} + \|\boldsymbol{S}_i(\vec{u}_{i,k+1} - \vec{u}_{i,k})\|^2_{\boldsymbol{R}_i} \tag{3.45}$$

in which $\boldsymbol{Q}_i = Q \otimes I_{p_i}$, $\boldsymbol{R}_i = R \otimes I_{p_i}$ and $\boldsymbol{L}_i, \boldsymbol{D}_i, \boldsymbol{S}_i, \vec{G}_i$ are local Laplacian matrix, local reference-accessibility matrix, local selected matrix, local system matrix, respectively. As an example, if $\mathscr{N}_i = \{l, m\}$, then

$$\boldsymbol{L}_i = \begin{bmatrix} \sum_{j\in\mathscr{N}_i} W_{ij} & -W_{il} & -W_{im} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \otimes I_N$$

$$\boldsymbol{D}_i = \begin{bmatrix} \mathcal{D}_{ii} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \otimes I_N \qquad \boldsymbol{S}_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \otimes I_N$$

$$\vec{G}_i = \text{diag}\ (G_i, G_l, G_m).$$

Correspondingly, $\vec{r}_i$, $\vec{d}_i$, $\vec{u}_{i,k}$ are local reference, local initial state respond and local input vector, which defined as

$$\begin{aligned} \vec{r}_i &= [r^T \quad r^T \quad r^T]^T \\ \vec{d}_i &= [d_i^T \quad d_l^T \quad d_m^T]^T \\ \vec{u}_{i,k} &= [u_{i,k}^T \quad u_{l,k}^T \quad u_{m,k}^T]^T \end{aligned}$$

Remark 3.4. *Note that, all the above local parameters are obtained by extracting the corresponding elements in the global matrices/vectors defined in (3.23). When each* *subsystem updates its control law, the computation only these local parameters and hence can be done in a fully distributed manner.*

Using ADMM to implement Algorithm 3.1 distributively, yields the following distributed implementation algorithm:

Algorithm 3.2.0. *At $k+1^{th}$ ILC trial, the input $\vec{u}_{k+1}$ generated distributively by the following ADMM steps*

$$\vec{u}^{q+1}_{i,k+1} = \arg\min\Big[J_{i,k+1}(\vec{u}^{q+1}_{i,k+1})+\frac{\rho}{2}||\vec{u}^{q+1}_{i,k+1}-\widetilde{E}_i z^q_{k+1}||^2+\gamma^q_{i,k+1}{}^T(\vec{u}^{q+1}_{i,k+1}-\widetilde{E}_i z^q_{k+1})\Big] \tag{3.46}$$

$$z^{q+1}_{i,k+1} = \frac{1}{1+|\mathscr{N}_i|}\sum_{o\in(\mathscr{N}_i\bigcup i)}(\vec{u}^{q+1}_{o,k+1})_i \tag{3.47}$$

$$\gamma^{q+1}_{i,k+1} = \gamma^q_{i,k+1}+\rho(\vec{u}^{q+1}_{i,k+1}-\widetilde{E}_i z^{q+1}_{k+1}) \tag{3.48}$$

provides a distributed implementation to solve the problem (3.22), i.e.

$$\lim_{q\to\infty} z^q_{k+1} = \arg\min\{J_{k+1}(\vec{u}_{k+1})\} \tag{3.49}$$

where $z_{i,k+1}$ is the i^{th} global component at $k+1^{th}$ ILC trials, $(\vec{u}_{o,k+1})_i$ is the element in $\vec{u}_{o,k+1}$ that related to $z_{i,k+1}$ and $|\mathscr{N}_i|$ is the i^{th} subsystem's neighbour amount.

Note that for the input updating law (3.46), it can be computed in a matrix form, or in a feedback plus feedforward form, as shown in the following two propositions:

Proposition 3.1. *The input law (3.46) can be implemented in a matrix form as follows:*

$$\begin{aligned}\vec{u}^{q+1}_{i,k+1} = &\Big[\vec{G}_i^T(\boldsymbol{L}_i+\boldsymbol{D}_i)^T\boldsymbol{Q}_i(\boldsymbol{L}_i+\boldsymbol{D}_i)\vec{G}_i+\frac{\rho}{2}I+\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i\Big]^{-1}\\ &\times\Big[\vec{G}_i^T(\boldsymbol{L}_i+\boldsymbol{D}_i)^T\boldsymbol{Q}_i(\boldsymbol{L}_i+\boldsymbol{D}_i)(\vec{r}_i-\vec{d}_i)+\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i\vec{u}_{i,k}+\frac{\rho}{2}\widetilde{E}_i z^q_{k+1}-\frac{1}{2}\gamma^q_{i,k+1}\Big]\end{aligned} \tag{3.50}$$

Proof. Finding the stationary point of Equation (3.46), we have

$$\begin{aligned}&\Big[\vec{G}_i^T(\boldsymbol{L}_i+\boldsymbol{D}_i)^T\boldsymbol{Q}_i(\boldsymbol{L}_i+\boldsymbol{D}_i)\vec{G}_i+\frac{\rho}{2}I+\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i\Big]\vec{u}^{q+1}_{i,k+1} =\\ &\quad\vec{G}_i^T(\boldsymbol{L}_i+\boldsymbol{D}_i)^T\boldsymbol{Q}_i(\boldsymbol{L}_i+\boldsymbol{D}_i)(\vec{r}_i-\vec{d}_i)+\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i\vec{u}_{i,k}+\frac{\rho}{2}\widetilde{E}_i z^q_{k+1}-\frac{1}{2}\gamma^q_{i,k+1}.\end{aligned} \tag{3.51}$$

Note that the term $\vec{G}_i^T(\boldsymbol{L}_i+\boldsymbol{D}_i)^T\boldsymbol{Q}_i(\boldsymbol{L}_i+\boldsymbol{D}_i)\vec{G}_i+\frac{\rho}{2}I+\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i$ is invertible, it yields the input updating law (3.50). □

Proposition 3.2. *The input law (3.46) can be calculated via a feedback plus feedforward structure*

$$\vec{u}^{q+1}_{i,k+1}(t) = \eta^q_{i,k+1}(t)+\boldsymbol{\Psi}_i^{-1}(t)\vec{B}_i^T\psi^{q+1}_{i,k+1}(t) \tag{3.52}$$

with costate $\psi_{i,k+1}^{q+1}(t)$ defined as

$$\psi_{i,k+1}^{q+1}(t) = \left\{ -K_i(t) \left[I_{p_i} + \vec{B}_i \boldsymbol{\Psi}_i^{-1}(t) \vec{B}_i^T K_i(t) \right]^{-1} \vec{A}_i \vec{\chi}_{i,k+1}^{q+1}(t) \right\} + \xi_{i,k+1}^{q+1}(t) \tag{3.53}$$

where $\vec{\chi}_{i,k+1}^{q+1}(t)$ is denoted as

$$\begin{aligned} \vec{\chi}_{i,k+1}^{q+1}(0) &= 0 \\ \vec{\chi}_{i,k+1}^{q+1}(t+1) &= \vec{A}_i \vec{\chi}_{i,k+1}^{q+1}(t) + \vec{B}_i \left[\vec{u}_{i,k+1}^{q+1}(t) - \eta_{i,k+1}^{q}(t) \right] \end{aligned} \tag{3.54}$$

and $\vec{A}_i$, $\vec{B}_i$, $\vec{C}_i$ are local state space matrices of subsystem i and $K_i(\cdot)$ is the local discrete Riccati Equation on the time interval $t \in [0, N-1]$ with terminal condition $K_i(N) = 0$:

$$\begin{aligned} K_i(t) = &\ \vec{A}_i^T K_i(t+1) \vec{A}_i + \vec{C}_i^T \boldsymbol{\Xi}_i(t+1) \vec{C}_i \\ &- \vec{A}_i^T K_i(t+1) \vec{B}_i \left[\vec{B}_i^T K_i(t+1) \vec{B}_i + \boldsymbol{\Psi}_i(t+1) \right]^{-1} B_i^T K_i(t+1) \vec{A}_i \end{aligned} \tag{3.55}$$

$\xi_{i,k+1}^{q+1}(t)$ denotes the feedforward term as

$$\xi_{i,k+1}^{q+1}(t) = \left[I_{p_i} + K_i(t) \vec{B}_i \boldsymbol{\zeta}_i^{-1} \vec{B}_i^T \right]^{-1} \left[\vec{A}_i^T \xi_{i,k+1}^{q+1}(t+1) + \vec{C}_i^T \boldsymbol{\Xi}_i(t+1) \theta_{i,k+1}^{q}(t+1) \right] \tag{3.56}$$

with terminal condition $\xi_{i,k+1}^{q+1}(N) = 0$; $\eta_{i,k+1}^{q}$ is defined as

$$\eta_{i,k+1}^{q} = \boldsymbol{\zeta}_i^{-2} \left(\frac{\rho}{2} \widetilde{E}_i z_{k+1}^{q} - \frac{1}{2} \gamma_{i,k+1}^{q} + \boldsymbol{S}_i{}^T \boldsymbol{R}_i \boldsymbol{S}_i \vec{u}_{i,k} \right) \tag{3.57}$$

$\boldsymbol{\zeta}_i$, $\boldsymbol{\varsigma}_i$ are defined as

$$\boldsymbol{\zeta}_i = \left(\boldsymbol{S}_i{}^T \boldsymbol{R}_i \boldsymbol{S}_i + \frac{\rho}{2} I_{p_i N} \right)^{\frac{1}{2}}, \qquad \boldsymbol{\varsigma}_i = \boldsymbol{Q}_i{}^{\frac{1}{2}} (\boldsymbol{L}_i + \boldsymbol{D}_i) \tag{3.58}$$

$\boldsymbol{\Psi}_i$, $\boldsymbol{\Xi}_i$ are defined as

$$\boldsymbol{\Psi}_i = \boldsymbol{\zeta}_i^T \boldsymbol{\zeta}_i, \qquad \boldsymbol{\Xi}_i = \boldsymbol{\varsigma}_i^T \boldsymbol{\varsigma}_i \tag{3.59}$$

$\theta_{i,k+1}^{q}$ is defined as

$$\theta_{i,k+1}^{q} = \vec{r}_i - \vec{G}_i \eta_{i,k+1}^{q} - \vec{d}_i \tag{3.60}$$

and $(\cdot)(t)$ denotes the t^{th} element in the vector or matrix.

Proof. At iteration $q+1$, the input of the ADMM algorithm is generated by solving the following optimization problem

$$\begin{aligned} \vec{u}_{i,k+1}^{q+1} = \arg\min\{ &\|(\boldsymbol{D}_i + \boldsymbol{L}_i)(\vec{r}_i - \vec{G}_i \vec{u}_{i,k+1}^{q+1} - \vec{d}_i)\|_{\boldsymbol{Q}_i}^2 + \|\boldsymbol{S}_i(\vec{u}_{i,k+1}^{q+1} - \vec{u}_{i,k})\|_{\boldsymbol{R}_i}^2 \\ &+ \frac{\rho}{2} \|\vec{u}_{i,k+1}^{q+1} - \widetilde{E}_i z_{k+1}^{q}\|^2 + \gamma_{i,k+1}^{q}{}^T (\vec{u}_{i,k+1}^{q+1} - \widetilde{E}_i z_{k+1}^{q}) \} \end{aligned} \tag{3.61}$$

For the above optimization problem, we define the last 3 terms as $\varpi_{i,k+1}^{q+1}$, which could be written as:

$$\begin{aligned}\varpi_{i,k+1}^{q+1} = (\vec{u}_{i,k+1}^{q+1})^T \left(\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i + \frac{\rho}{2}I_{p_iN}\right)\vec{u}_{i,k+1}^{q+1} \\ - 2(\vec{u}_{i,k+1}^{q+1})^T\left[\frac{\rho}{2}\widetilde{E}_i z_{k+1}^q - \frac{1}{2}\gamma_{i,k+1}^q + \boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i\vec{u}_{i,k}\right]\end{aligned} \tag{3.62}$$

that is

$$\varpi_{i,k+1}^{q+1} = \|\boldsymbol{\zeta}_i(\vec{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^q)\|^2 \tag{3.63}$$

where $\boldsymbol{\zeta}_i = \left(\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i + \frac{\rho}{2}I_{p_iN}\right)^{\frac{1}{2}}$, $\eta_{i,k+1}^q = \boldsymbol{\zeta}_i^{-2}\left(\frac{\rho}{2}\widetilde{E}_i z_{k+1}^q - \frac{1}{2}\gamma_{i,k+1}^q + \boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i\vec{u}_{i,k}\right)$. It follows that

$$\varpi_{i,k+1}^{q+1} = \|\vec{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^q\|^2_{\boldsymbol{\Psi}_i = \boldsymbol{\zeta}_i{}^T\boldsymbol{\zeta}_i} \tag{3.64}$$

Note that, the first term in (3.61) can be written as:

$$\varrho_{i,k+1}^{q+1} = \left\|\varsigma_i\left[\theta_{i,k+1}^q - \vec{G}_i(\vec{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^q)\right]\right\|^2 \tag{3.65}$$

where

$$\begin{aligned}\varsigma_i &= \boldsymbol{Q}_i{}^{\frac{1}{2}}(\boldsymbol{L}_i + \boldsymbol{D}_i) \\ \theta_{i,k+1}^q &= \vec{r}_i - \vec{G}_i\eta_{i,k+1}^q - \vec{d}_i\end{aligned} \tag{3.66}$$

then, we have

$$\varrho_{i,k+1}^{q+1} = \|\theta_{i,k+1}^q - \vec{G}_i(\vec{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^q)\|^2_{\boldsymbol{\Xi}_i = \varsigma_i{}^T\varsigma_i}. \tag{3.67}$$

Now, the input law (3.46) can be written into a Linear Quadratic Tracking (LQT) problem:

$$\vec{u}_{i,k+1}^{q+1} = \operatorname{argmin}(\|\theta_{i,k+1}^q - \vec{G}_i(\vec{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^q)\|^2_{\boldsymbol{\Xi}_i} + \|\vec{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^q\|^2_{\boldsymbol{\Psi}_i}) \tag{3.68}$$

and then substituting the term in (3.68) into standard LQT problem, yields (3.52). That completes the proof. □

Implementing Equation (3.46) in matrix form is easy to follow. However, it requires huge computational load to calculate the system matrix inverse when the trial length is large. By contrast, the feedback plus feedforward implementation has less calculation requirement (since it does not involve the system matrix inverse calculation), and stronger robustness (benefiting from the real-time state feedback).

3.3.3 Distributed NOILC Algorithm

Using Algorithm 3.2.0 to implement Algorithm 3.1, a distributed NOILC Algorithm 3.2 for high performance consensus tracking problem is obtained. For Algorithm 3.2, the

Algorithm 3.2. *Distributed NOILC Algorithm for Consensus Tracking Problem*

```
Input: State space matrix $A_i$, $B_i$, $C_i$, reference trajectory $r$, reference-accessibility
    matrix $D$, Laplacian matrix $L$, maximum-trial $k_{max}$, maximum-iteration $q_{max}$,
    penalty parameter $\rho$, weighting $Q$ and weighting $R$
Output: Optimal input $u_{i,k_{max}}$ for each subsystem
1:  Initialization: Set the trial number $k = 0$
2:  For: $k = 0$ to $k_{max}$
3:      For: $q = 0$ to $q_{max}$
4:          For: $i = 1$ to $p$
5:              Receive information from neighbours
6:              $\vec{u}_{i,k+1}^{q+1}$ updating law either in (3.50) or in (3.52)
7:              $z_{i,k+1}^{q+1}$ updating law in Equation (3.47)
8:              $\gamma_{i,k+1}^{q+1}$ updating law in Equation (3.48)
9:              Send information to neighbours
10:         End for
11:     End for
12: Transform $\vec{u}_{i,k+1}^{q_{max}}$ into $u_{i,k+1}$
13: End for
14: Return: Optimal input $u_{i,k_{max}}$ for each subsystem
```

mission of distributed implementation is to find the optimal input solution in each ILC experiment. In each ADMM iteration, all the subsystems are required to update the local variables in parallel (and hence save the computational complexity): in Step 5, each subsystem collects useful information from its neighbours and uses these information to generate the local input plan (either in matrix form using (3.50) or feedback plus feedforward structure using (3.52)) in Step 6; in Step 7, each subsystem performs a local averaging of all the corresponding local input plan to conclude the global element $z_{i,k+1}^{q+1}$ and then the local dual variable $\gamma_{i,k+1}^{q+1}$ is generated by calculating the difference between local input plan and global element in Step 8; finally, each subsystem sends its local data to neighbouring subsystems in Step 9 and waits for next ADMM iteration. By repetitively performing Step 4 – 11 for q_{max} times, the resulting input solution is considered as the optimal input choice in current ILC trial and the result will be used for next trial's input generation.

Note that, the choice of maximum-iteration q_{max} influence the convergence speed of Algorithm 3.2. In theory, it requires infinite ADMM iterations to approach the centralised result, which seems unrealisable. In reality, ADMM is efficient in most cases and hence few iteration numbers are enough to approach the centralised result, which is demonstrated in later simulations. In addition, the choice of penalty parameter also affect the convergence performance. A proper choice of the penalty parameter resulting in fast convergence is given in the next section.

3.3.4 Penalty Parameter Selection

Although ADMM can guarantee the convergence for any positive penalty parameter ρ, its convergence speed is extremely sensitive to the choice of ρ. In some cases, ADMM may converge slower than other distributed algorithm if the choice of ρ is not proper. To address this problem, some research propose different rule of thumb for choosing the penalty parameter ρ, e.g., Deng and Yin (2016); Hong and Luo (2017); Joshi et al. (2013). However, empirically tuning the parameter ρ cannot return the best control result, which is undesirable in practice. Fortunately, Ghadimi et al. (2015); Teixeira et al. (2013, 2016) provide a mathematics way to calculate ρ. Next, we will use the result in the above paper to find the best penalty parameter ρ.

To find the best ρ, we first transform the problem into the centralised formulation, i.e.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\vec{u}_{k+1}^T \mathcal{Q}\vec{u}_{k+1} + \mathfrak{q}^T\vec{u}_{k+1} \\ \text{subject to} \quad & M(\vec{u}_{k+1} - \widetilde{E}z_{k+1}) = 0 \end{aligned} \tag{3.69}$$

where $\mathcal{Q} = \vec{R}^{-1}\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G} + I_{pN}$ and $\mathfrak{q} = \mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\vec{r} + \vec{R}\vec{u}_k$, $\widetilde{E}$ is a matrix that maps the global variable z_{k+1} to the centralised input plan $\vec{u}_{k+1}$, and M is a scaling matrix. For simplicity, we define $\bar{F} = -M\widetilde{E}$ and the optimisation problem in (3.69) is rewritten as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\vec{u}_{k+1}^T \mathcal{Q}\vec{u}_{k+1} + \mathfrak{q}^T\vec{u}_{k+1} \\ \text{subject to} \quad & M\vec{u}_{k+1} + \bar{F}z_{k+1} = 0 \end{aligned} \tag{3.70}$$

Considering the input sequence $\{\vec{u}_{k+1}^q\}$ approaching to the optimal solution $\vec{u}_{k+1}^*$, and then we define the convergence factor as follow (Bertsekas and Tsitsiklis, 1997)

$$\phi \triangleq \sup_{\vec{u}_{k+1}^q \neq \vec{u}_{k+1}^*} \frac{\|\vec{u}_{k+1}^{q+1} - \vec{u}_{k+1}^*\|}{\|\vec{u}_{k+1}^q - \vec{u}_{k+1}^*\|} \tag{3.71}$$

and the mission of the penalty parameter selection can be stated as finding the best ρ that minimizes the convergence factor ϕ. By applying the result in Teixeira et al. (2016) to investigate the best choice of ρ, we have the following proposition:

Proposition 3.3. *In Algorithm 3.2, the optimal penalty parameter ρ^* is defined as*

$$\rho^* = \begin{cases} \dfrac{1-\sqrt{1-\lambda_{n-s}^2}}{\lambda_{n-s}^2 - 1 + \sqrt{1-\lambda_{n-s}^2}} & \lambda_{n-s} > 0 \\ 1 & \lambda_{n-s} \leq 0 \end{cases} \tag{3.72}$$

where λ_i is the i^{th} generalized eigenvalue of $(M^T\left[2\bar{F}(\bar{F}^T\bar{F})^{-1}\bar{F}^T - I_{pN}\right]M, M^TM)$, *order as* $\lambda_n \geq \cdots \geq \lambda_i \geq \lambda_1$, $M = [\vec{R}^{-1}\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}+I_{pN}]^{\frac{1}{2}}$, $\bar{F} = -M\widetilde{E}$, *and* $\mathcal{R}(A) \triangleq \{y \in \mathbb{R}^n | y = Ax, x \in \mathbb{R}^m\}$ *is the range-space of matrix* $A \in \mathbb{R}^{n\times m}$ *with* $s = dim(\mathcal{R}(\bar{F}))$*. In this situation, the corresponding optimal convergence factor is*

$$\phi^* = |\phi_{2n-s}| = \begin{cases} \frac{1}{2}\left(1 + \frac{\lambda_{n-s}}{1+\sqrt{1-\lambda_{n-s}^2}}\right) & \lambda_{n-s} > 0 \\ \frac{1}{2} & \lambda_{n-s} \leq 0 \end{cases} \tag{3.73}$$

Proof. By defining $\bar{F} = -M\widetilde{E}$ and $M^TM = \mathcal{Q}$ meet the Assumption 3 in Teixeira et al. (2016) , we can apply Theorem 3, Theorem 4, Corollary 1 in Teixeira et al. (2016) to obtain Proposition 3.3. That completes the proof. □

To sum up, the steps to find the optimal penalty parameter can be described as follows:

1. Set $M = \left[\vec{R}^{-1}\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}+I_{pN}\right]^{\frac{1}{2}}$ and calculate $\bar{F} = -M\widetilde{E}$;
2. Find the generalized eigenvalue of $(M^T\left[2\bar{F}(\bar{F}^T\bar{F})^{-1}\bar{F}^T - I_{pN}\right]M, M^TM)$;
3. Find $s = dim(\mathcal{R}(\bar{F}))$ and calculate λ_{n-s};
4. Use Proposition 3.3 to find the optimal penalty parameter ρ^* and corresponding optimal convergence factor ϕ^*.

Remark 3.5. *By introducing the scaling matrix* M*, the optimisation problem is reformulated, and the input and dual variable updating law are still separable since matrix* M*,* F *are block matrices and can be separated into* p *smaller local matrices. However, the global variable updating law can not be separated, it has a combined form defined as*

$$z_{k+1}^{q+1} = (\bar{F}^T\bar{F})^{-1}\bar{F}^T(M\vec{u}_{k+1}^{q+1} + \rho\gamma_{k+1}^{q+1}) \tag{3.74}$$

3.4 Robustness Analysis of the Proposed Algorithm

In practice, an accurate system model is not always possible, and therefore we need to analyse the robustness property of the proposed NOILC algorithm. In this section, the robustness is defined in terms of *Robust Monotone Convergence* (Owens et al., 2009):

Robust Monotone Convergence: An ILC algorithm has the property of robust monotone convergence with respect to a vector norm $\|\cdot\|^2$ in the presence of a defined set of model uncertainties if, and only if, for every choice of control on the first trial (and hence for the corresponding initial error) and for any choice of model uncertainty within the defined set, the resulting sequence of iteration error time signals converges to zero with a strictly monotonically decreasing norm.

In the following section, we consider $G_{0,i}(z)$ as the transfer function of i^{th} subsystem's plant, and then the relationship between plant and model can be represented as

$$G_{0,i}(z) = G_i(z)U_i(z), \qquad i = 1, \cdots, p \tag{3.75}$$

where $U_i(z)$ represents multiplicative uncertainty of i^{th} subsystem (assume it is proper and stable). Then, if $U_i(z)$ has a matrix representation U_i, it follows that

$$G_{0,i} = G_i U_i, \qquad i = 1, \cdots, p \tag{3.76}$$

Then, the global form of (3.76) can be written as

$$\mathbb{G}_0 = \mathbb{G}\mathbb{U} \tag{3.77}$$

where $\mathbb{G}_0 =$ diag $(G_{0,1}, \cdots, G_{0,p})$, $\mathbb{U} =$ diag $(U_1, \cdots, U_p)$.

Following a similar robustness analysis approach for NOILC in Owens (2016), we have the following results.

3.4.1 Time Domain Robustness Analysis

In the presence of model uncertainty, the error evolution of Algorithm 3.1 is shown as follows (by multiplying $\mathbb{G}\mathbb{U}$ in both side of Equation (3.25) and subtracting from $\vec{r} - \vec{d}$)

$$\vec{r} - \vec{d} - \mathbb{G}\mathbb{U}\vec{u}_{k+1} = \vec{r} - \vec{d} - \mathbb{G}\mathbb{U}\vec{u}_k - \mathbb{G}\mathbb{U}L_u\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\vec{e}_k \tag{3.78}$$

$$\vec{e}_{k+1} = \left[I_{pN} - \mathbb{G}\mathbb{U}L_u\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\right]\vec{e}_k \tag{3.79}$$

where $L_u = \left[\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G} + \vec{R}\right]^{-1}$ (introduced for notational simplicity). Note that, the inner product

$$\|\hat{e}_k\|^2_{\vec{Q}} = \left\langle \vec{Q}^{\frac{1}{2}}(\boldsymbol{L}+\boldsymbol{D})\vec{e}_k, \vec{Q}^{\frac{1}{2}}(\boldsymbol{L}+\boldsymbol{D})\vec{e}_k \right\rangle \tag{3.80}$$

and then use the error evolution (3.79), we have

$$\begin{aligned}\|\hat{e}_{k+1}\|^2_{\vec{Q}} &= \|\hat{e}_k\|^2_{\vec{Q}} + \vec{e}_k^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}L_u \\ &\times \left[\mathbb{U}^T\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}\mathbb{U} - \mathbb{U}^T L_m - L_m\mathbb{U}\right]L_u\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\vec{e}_k\end{aligned} \tag{3.81}$$

where $L_m = \mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G} + \vec{R}$ (introduced for notational simplicity).

Note that, achieving *Robust Monotone Convergence* is equivalent to ensure the monotonicity property $\|\hat{e}_{k+1}\|^2_{\vec{Q}} \leq \|\hat{e}_k\|^2_{\vec{Q}}$ for all $k \geq 0$. Based on Equation (3.81), we can derive the robustness condition as shown in the following theorem:

Theorem 3.2. *In the presence of the modelling error $U_i(z)$, Algorithm 3.1 is robust monotone convergent, if and only if the system satisfies the following condition*

$$L_m\mathbb{U} + \mathbb{U}^T L_m \geq \mathbb{U}^T\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}\mathbb{U} \tag{C1}$$

Proof. Note that, $L_u\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})$ is non-singular, and from the error evolution (3.81), it follows that the condition

$$\mathbb{U}^T\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}\mathbb{U} - L_m\mathbb{U} - \mathbb{U}^T L_m \leq 0$$

result in monotonicity of tracking error norm $\|\hat{e}_k\|^2_{\vec{Q}}$. □

Theorem 3.2 provides an explicit way to check the robustness of Algorithm 3.1 in the presence of the modelling error, however, the calculation is time-consuming. Alternatively, the following proposition simplifies the result by proving a conservative sufficient condition.

Proposition 3.4. *In the presence of the modelling error $U_i(z)$, Algorithm 3.1 is robust monotone convergent if (sufficient condition) the system satisfies the following condition*

$$L_m\mathbb{U} + \mathbb{U}^T L_m \geq \|(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}\|^2_{\vec{Q}}\,\mathbb{U}^T\mathbb{U} \tag{C2}$$

Proof. Note that, the term $\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}$ can be simplified as

$$\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G} \leq \|(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}\|^2_{\vec{Q}} \tag{3.82}$$

and follows from Condition C1, Condition C2 is obtained. □

Note that, we can obtain another conservative sufficient condition by finding the maximum eigenvalue of matrix $(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})$, as shown in the following:

Proposition 3.5. *In the presence of the modelling error $U_i(z)$, Algorithm 3.1 is robust monotone convergent if (sufficient condition) the system satisfies the following condition*

$$L_m\mathbb{U} + \mathbb{U}^T L_m \geq (2d_L + d_D)^2 d_Q \mathbb{U}^T\mathbb{G}^T\mathbb{G}\mathbb{U} \tag{C3}$$

where d_D and d_Q denote the maximum diagonal elements in the matrices D and Q.

Proof. Note that, adding the diagonal, non-negative matrix D to the Laplacian matrix L can only increase the diagonal elements of L. Using Lemma 3.2 and Gershgorin theorem, the eigenvalues of matrix $L + D$ are shown as

$$0 \leq \lambda_{L+D,i} \leq 2d_L + d_D, \qquad 1 \leq i \leq p. \tag{3.83}$$

Then, we have

$$(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D}) = \left[(L+D)^TQ(L+D)\right]\otimes I_N \leq (2d_L+d_D)^2 d_Q I_{pN} \tag{3.84}$$

and follows from Condition C1, Condition C3 is obtained. □

3.4.2 The Interpretation for Design

Some useful observations of the algorithm's robustness conditions can be obtained as:

(1) The first situation is when no modelling error and hence multiplicative uncertainty $\mathbb{U} = I_{pN}$. The condition in Theorem 3.2 reduces to

$$2L_m \geq \mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G} \tag{3.85}$$

and clearly, this (sufficient and necessary) condition is always hold. This implies that the proposed algorithm can guarantee the monotonic convergence of the consensus tracking error norm when no modelling error in the networked dynamical systems.

(2) When there exists modelling error in the system (i.e., $\mathbb{U} \neq I_{pN}$), different choices of weighting matrices Q and R affect the algorithm's robustness property. From the error evolution (3.79), we have the following relationship:

$$\hat{e}_{k+1} = \left[I_{pN} - \vec{Q}^{\frac{1}{2}}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}\mathbb{U}L_u\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}^{\frac{1}{2}}\right]\hat{e}_k \tag{3.86}$$

To guarantee the *Robust Monotone Convergence*, the sufficient and necessary condition for (3.86) is that

$$\|I_{pN} - \vec{Q}^{\frac{1}{2}}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}\mathbb{U}L_u\mathbb{G}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}^{\frac{1}{2}}\| < 1 \tag{3.87}$$

and the question becomes: how $\vec{Q}$ and $\vec{R}$ affect the range of modelling error $\mathbb{U}$?

Intuitively, a larger $\frac{\vec{Q}}{\vec{R}}$ can tolerate larger model uncertainty, and hence results in stronger robustness. Note that, analysing the effect of weighting matrices is difficult in theory, since all the terms are matrices and they are tight coupled. However, we can run a simple example to verify the phenomenon:

Example 3.1. *Consider a networked dynamical system has 3 subsystems and the time instant is set to be* 1 *for simplicity. We set the terms in Equation (3.87) as follows:*

$$\mathbb{G} = \begin{bmatrix} 0.15 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0.3 \end{bmatrix} \qquad \boldsymbol{L} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

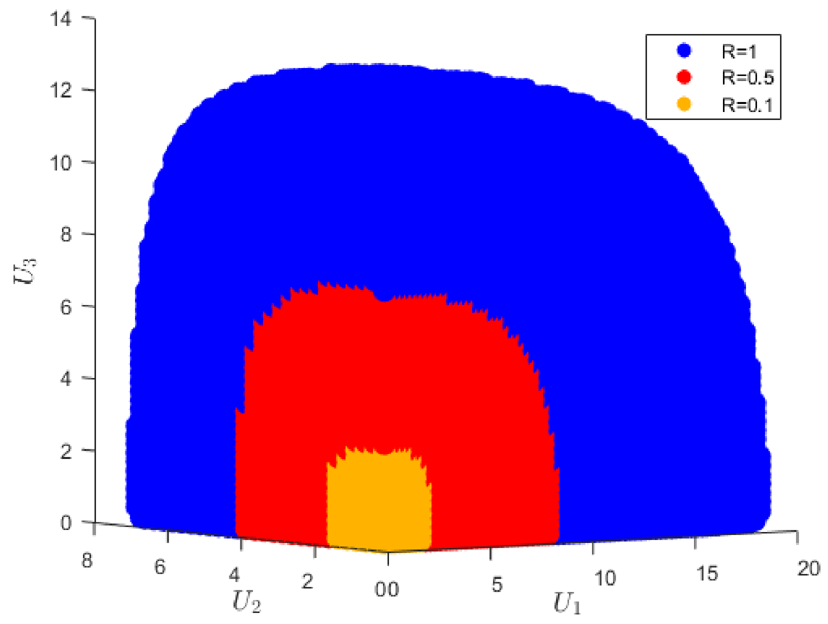

Figure 3.1: Robust region for different R

$$\mathbb{U} = \begin{bmatrix} U_1 & 0 & 0 \\ 0 & U_2 & 0 \\ 0 & 0 & U_3 \end{bmatrix} \qquad \boldsymbol{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

In the simulation, we use the Monte Carlo method to generate different $\mathbb{U}$. By fixing the scalar weighting $Q = 1$, we draw the region for different scalar weighting R that satisfies the condition (3.87), as shown in Figure 3.1. Clearly, when the scalar weighting R increases, the algorithm covers larger region (i.e., it tolerates more model uncertainties).

3.5 The Proposed Algorithm for Consensus Tracking of Point To Point Tasks

The P2P task has found a great number of application in practice and the exploration of the algorithm for P2P task is valuable. Given an example, multi robot ‘pick and place’ collaborative task requires all the robotic arms travel from the initial position to the ‘pick’ position at time instant t_1; then collaboratively move the objectives from the ‘pick’ position to the ‘place’ position at time instant t_2; finally return to the initial position and wait for next experiment. This task lies emphasis on the consensus tracking at time $t1$, $t2$, which allows infinite choices of the control input, and therefore creates greater design flexibility, but on the other hand , it leads to significant design difficulties. In this section, we further extend the proposed algorithm to solve P2P consensus tracking problem, with the convergence and robustness properties analysed rigorously.

3.5.1 Formulation of P2P Consensus Tracking Task

For the P2P consensus tracking task, it only requires tracking at M time instants t_m $(m = 1, 2, \cdots, M)$ rather than the tracking over the whole horizon $[0, N]$. Different from the general consensus tracking task, P2P consensus tracking task creates more freedoms for the control input choices because there exists infinity input solutions.

For P2P task, we define the time instant vector as

$$\Lambda = [t_1 \quad t_2 \quad \cdots \quad t_M]^T, \tag{3.88}$$

and naturally, the P2P reference vector is defined as

$$r^P = [r(t_1) \quad r(t_2) \quad \cdots \quad r(t_M)]^T. \tag{3.89}$$

Now, we define the i^{th} subsystem's P2P output as

$$y_{i,k}^P = [y_{i,k}(t_1) \quad y_{i,k}(t_2) \quad \cdots \quad y_{i,k}(t_M)]^T \tag{3.90}$$

where i^{th} subsystem's P2P output space $\mathcal{Y}_i^P = \mathbb{R}^M$ are defined with inner products and induced norms

$$\langle x, y \rangle_Q = x^T Q y, \qquad \|y\|_Q = \sqrt{\langle y, y \rangle_Q}. \tag{3.91}$$

Note that, with a slight abuse of notation, we still use the same notation Q to define the output weighting in P2P space, however, the dimension of Q will become $m \times m$.

Then, the P2P system model can be written as

$$y_{i,k}^P = G_i^P u_{i,k} + d_i^P \tag{3.92}$$

where G_i^P, d_i^P is obtained by extracting the related rows from G_i and d_i.

Using the 'lifted matrix form' representation, the global system model is written as

$$\vec{y}_k^P = \mathbb{G}^P \vec{u}_k + \vec{d}^P \tag{3.93}$$

where $\mathbb{G}^P$, $\vec{d}^P$, $\vec{r}^P$ and $\vec{y}_k^P$ are defined as

$$\begin{aligned}
\mathbb{G}^P &= \text{diag}\,(G_1^P, G_2^P, \cdots, G_p^P) \\
\vec{d}^P &= \begin{bmatrix} d_1^{P\,T} & d_2^{P\,T} & \cdots & d_p^{P\,T} \end{bmatrix}^T \in \mathcal{Y}^P \\
\vec{r}^P &= \begin{bmatrix} r^{P\,T} & r^{P\,T} & \cdots & r^{P\,T} \end{bmatrix}^T \in \mathcal{Y}^P \\
\vec{y}_k^P &= \begin{bmatrix} y_{1,k}^{P\ T} & y_{2,k}^{P\ T} & \cdots & y_{p,k}^{P\ T} \end{bmatrix}^T \in \mathcal{Y}^P
\end{aligned} \tag{3.94}$$

and the P2P output space $\mathcal{Y}^P = \mathcal{Y}_1^P \times \mathcal{Y}_2^P \times \cdots \times \mathcal{Y}_p^P$ are defined with inner products and induced norms

$$\langle \vec{x}, \vec{y} \rangle_{\vec{Q}} = \sum_{i=1}^{p} x_i^T Q y_i, \qquad \|\vec{y}\|_{\vec{Q}} = \sqrt{\langle \vec{y}, \vec{y} \rangle_{\vec{Q}}} \tag{3.95}$$

where matrix $\vec{Q} = \text{diag}(Q, Q, \cdots, Q)$ is positive definite.

Now, we are ready to define the ILC design problem for P2P consensus tracking tasks:

ILC for P2P Consensus Tracking Task: The ILC design problem can be stated as finding a control law

$$\vec{u}_{k+1} = f(\vec{u}_k, \vec{e}_k^P) \tag{3.96}$$

such that $\vec{y}_k^P$ tracks the desired P2P reference $\vec{r}^P$, i.e.

$$\lim_{k\to\infty} \vec{y}_k^P = \vec{r}^P \tag{3.97}$$

where $\vec{e}_k^P = \vec{r}^P - \hat{y}_k^P$ is the 'virtual' P2P tracking error (since the P2P reference is only available to part of the subsystems).

3.5.2 Algorithm Description

The proposed NOILC algorithm for P2P consensus tracking task is given as follows:

Algorithm 3.3. *For any initial input choice $\vec{u}_0$ and corresponding 'virtual' tracking error $\vec{e}_0^P$, the input sequence $\{\vec{u}_{k+1}\}_{k\geq 0}$ defined as*

$$\vec{u}_{k+1} = \arg\min\{\|(\boldsymbol{L} + \boldsymbol{D})\vec{e}_{k+1}^P\|_{\vec{Q}}^2 + \|\vec{u}_{k+1} - \vec{u}_k\|_{\vec{R}}^2\} \tag{3.98}$$

where $\vec{e}_{k+1}^P = \vec{e}_k^P - \mathbb{G}(\vec{u}_{k+1} - \vec{u}_k)$, $\boldsymbol{L} = L \otimes I_M$, $\boldsymbol{D} = D \otimes I_M$, provides a solution for P2P consensus tracking problem, i.e.

$$\lim_{k\to\infty} \vec{y}_{i,k}^P = r^P, \qquad i = 1, 2, \cdots, p \tag{3.99}$$

Remark 3.6. *The P2P distributed implementation follows exactly the same way as in Section 3.3, however, the form of feedback plus feedforward implementation will change (following a similar way as the traditional P2P feedback plus feedforward implementation), please refer to Owens et al. (2013) for more details.*

3.5.3 Convergence Properties of the P2P Algorithm

Algorithm 3.3 has appealing convergence properties shown in the following theorem:

Theorem 3.3. *For any initial input choice $\vec{u}_0$ and corresponding 'virtual' tracking error $\vec{e}_0^P$, Algorithm 3.3 achieves monotonic convergence of the P2P consensus tracking error norm $\|\hat{e}_k^P\|_{\vec{Q}} := \|(\boldsymbol{L}+\boldsymbol{D})\vec{e}_k^P\|_{\vec{Q}}$ to 0, i.e.*

$$\|\hat{e}_{k+1}^P\|_{\vec{Q}} \leq \|\hat{e}_k^P\|_{\vec{Q}}, \quad \lim_{k\to\infty} \hat{e}_k^P = 0. \tag{3.100}$$

Consequently, the perfect tracking is achieved as $k \to \infty$, i.e.

$$\lim_{k\to\infty} y_{i,k}^P = r^P, \qquad i = 1, 2, \cdots, p \tag{3.101}$$

Proof. Following the similar proof of Theorem 3.1, the P2P consensus tracking error norm converges monotonically to zero. Details are omitted here for brevity. □

In addition, the converged input has appealing properties shown in the next theorem.

Theorem 3.4. *For any initial input choice $\vec{u}_0$ and corresponding 'virtual' tracking error $\vec{e}_0^P$, Algorithm 3.3 guarantees the control input converge as follows*

$$\lim_{k\to\infty} \vec{u}_k = \vec{u}^* \tag{3.102}$$

where $\vec{u}^*$ is the solution for the following problem

$$\begin{aligned} &\text{minimize} && \|\vec{u}-\vec{u}_0\|_{\vec{R}}^2 \\ &\text{subject to} && \mathbb{G}^P\vec{u} + \vec{d}^P - \vec{r}^P = 0 \end{aligned} \tag{3.103}$$

Note that, if $\vec{u}_0$ is chosen to be 0, Algorithm 3.3 converges to the minimum input energy solution, i.e.

$$\begin{aligned} &\text{minimize} && \|\vec{u}\|_{\vec{R}}^2 \\ &\text{subject to} && \mathbb{G}^P\vec{u} + \vec{d}^P - \vec{r}^P = 0 \end{aligned} \tag{3.104}$$

Proof. For the optimisation problem

$$\begin{aligned} &\text{minimize} && \|\vec{u}-\vec{u}_0\|_{\vec{R}}^2 \\ &\text{subject to} && \mathbb{G}^P\vec{u} + \vec{d}^P - \vec{r}^P = 0 \end{aligned} \tag{3.105}$$

its Lagrangian is represented as

$$L(u,\lambda) = \|\vec{u}-\vec{u}_0\|_{\vec{R}}^2 + 2(\lambda, (\mathbb{G}^P\vec{u} + \vec{d}^P - \vec{r}^P)) \tag{3.106}$$

The control law for (3.98) is shown as

$$\begin{aligned} \vec{u}_{k+1} - \vec{u}_k &= \left[\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}^P + I_{pM}\right]^{-1}\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\vec{e}_k^P \\ &= \vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D}) + I_{pM}\right]^{-1}\vec{e}_k^P \end{aligned}$$

For matrix $\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})+I_{pM}\right]^{-1}$, it is positive definite with eigenvalues $0<\lambda_i\leq 1$. We then define the space spanned by the eigenvector corresponding to unitary eigenvalues as $\mathcal{E}^\perp$, yields

$$\begin{aligned}\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})+I_{pM}\right]^{-1}|_{\mathcal{E}}<I_{pM}\\ \left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})+I_{pM}\right]^{-1}|_{\mathcal{E}^\perp}=I_{pM}\end{aligned}\tag{3.107}$$

where $T|_B$ represents the restriction of operator T on the subspace B.

For any initial input choice $\vec{u}_0$, define the corresponding consensus error as $\vec{e}_0^P$, its decomposition on $\mathcal{E}$ and $\mathcal{E}^\perp$ is

$$\vec{e}_0^P=\vec{e}_0^{P^{\mathcal{E}}}+\vec{e}_0^{P^{\mathcal{E}^\perp}}\tag{3.108}$$

From (3.107), we have

$$\vec{e}_0^{P^{\mathcal{E}^\perp}}=\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})+I_{pM}\right]^{-1}\vec{e}_0^{P^{\mathcal{E}^\perp}}$$

therefore

$$\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\vec{e}_0^{P^{\mathcal{E}^\perp}}=0\tag{3.109}$$

Note that matrix $\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})$ is invertible, we have

$$\vec{e}_0^{P^{\mathcal{E}^\perp}}=0\tag{3.110}$$

Equation (3.107) becomes

$$\begin{aligned}\vec{u}_{k+1}-\vec{u}_0=&\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})+I_{pM}\right]^{-1}\\&\times\sum_{j=0}^{k}\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})+I_{pM}\right]^{-j}\vec{e}_0^P\end{aligned}$$

Note that $\vec{e}_0^{P^{\mathcal{E}}}\in\mathcal{E}$ and $\vec{e}_0^{P^{\mathcal{E}^\perp}}=0$, we have

$$\begin{aligned}\vec{u}_\infty-\vec{u}_0=&\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})+I_{pM}\right]^{-1}\\&\times\left\{I_{pM}-\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})+I_{pM}\right]^{-1}|_{\mathcal{E}}\right\}^{-1}\vec{e}_0^{P^{\mathcal{E}}}\end{aligned}$$

By setting $\lambda=-(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\left[I_{pM}+\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\right]^{-1}\Big\{I_{pM}$

$- \left[\mathbb{G}^P \vec{R}^{-1} \mathbb{G}^{P^T} (\boldsymbol{L}+\boldsymbol{D})^T \vec{Q} (\boldsymbol{L}+\boldsymbol{D}) + I_{pM}\right]^{-1} |_{\mathcal{E}} \Big\}^{-1} \vec{e}_0^{P^{\mathcal{E}}}$, we then have

$$\vec{u}_\infty - \vec{u}_0 = \vec{R}^{-1} \mathbb{G}^{P^T} \lambda \tag{3.111}$$

Equation (3.111) shows that $(\vec{u}_\infty, \lambda)$ is a stationary point of the Lagrangian. The problem (3.105) is strictly convex and hence $\vec{u}_\infty$ is the unique solution. That completes the proof. □

Remark 3.7. *Though the idea of NOILC framework has been successfully developed for P2P task in single-agent system (e.g., Chen et al. (2018b); Owens et al. (2013)), it has not been explored to networked dynamical systems. The extension of the result from the single-agent case to the multi-agent case is non-trivial, since the transformation in information relies on the network topology. For the P2P task in networked dynamical systems, each subsystem exist infinity choice of input solution and the whole system has infinity combination of these input solutions. All the subsystems can only use the network to communicate with each other and conclude the best input solution that guarantees the monotonic convergence (as shown in Theorem 3.3) and minimum energy cost (as shown in Theorem 3.4). It should be noticed that, the convergence result differs from the single-agent case, and it much more depends on the network topology (i.e., related to matrices* $\boldsymbol{L}$ *and* $\boldsymbol{D}$*).*

3.5.4 Robustness Properties of the P2P Algorithm

Following a similar analyse as in Section 3.4, we can have the following robustness conditions.

3.5.4.1 Time Domain Robustness Analysis

Theorem 3.5. *In the presence of the multiplicative modelling error* $U_i(z)$*, Algorithm 3.3 is robust monotone convergent if, and only if (sufficient and necessary condition), the system satisfies the following condition*

$$\mathbb{U}^T L_m^P + L_m^P \mathbb{U} \geq \mathbb{U}^T \mathbb{G}^{P^T} (\boldsymbol{L}+\boldsymbol{D})^T \vec{Q} (\boldsymbol{L}+\boldsymbol{D}) \mathbb{G}^P \mathbb{U} \tag{C4}$$

where $L_m^P = \mathbb{G}^{P^T} (\boldsymbol{L}+\boldsymbol{D})^T \vec{Q} (\boldsymbol{L}+\boldsymbol{D}) \mathbb{G}^P + \vec{R}$.

Proof. Following the similar proof of Theorem 3.2, Theorem 3.5 can be proved. □

We also provide the following conservative sufficient conditions for easier check.

Proposition 3.6. *In the presence of the modelling error $U_i(z)$, Algorithm 3.3 is robust monotone convergent if (sufficient condition) the system satisfies the following conditions*

$$\mathbb{U}^T L_m^P + L_m^P \mathbb{U} \geq \|(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}^P\|_{\vec{Q}}^2 \mathbb{U}^T\mathbb{U} \tag{C5}$$

Proof. Following the similar proof of Proposition 3.4, Proposition 3.6 can be proved. □

In addition, by finding the maximum eigenvalue of matrix $(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})$, we have the following Proposition:

Proposition 3.7. *In the presence of the modelling error $U_i(z)$, Algorithm 3.1 is robust monotone convergent if (sufficient condition) the system satisfies the following condition*

$$\mathbb{U}^T L_m^P + L_m^P \mathbb{U} \geq (2d_L + d_D)^2 d_Q \mathbb{U}^T {\mathbb{G}^P}^T \mathbb{G}^P \mathbb{U} \tag{C6}$$

where d_Q is the maximum eigenvalue of matrix Q.

Proof. Following the similar proof of Proposition 3.5, Proposition 3.7 can be proved. □

3.5.4.2 Input Energy Cost under Model Uncertainty

Without model uncertainty, Theorem 3.4 shows that Algorithm 3.3 converges to the minimum input energy solution when the initial input is chosen to be zero. Follows that, a natural question to be answered is 'in the presence of model uncertainty, what solution would the control input converge to?' This question can be answered by the following theorem.

Theorem 3.6. *For any initial input choice $\vec{u}_0$ and corresponding 'virtual' tracking error $\vec{e}_0^P$, if the model uncertainty $\mathbb{U}$ satisfying the following condition*

$$\|I_{pM} - \mathbb{G}^P\mathbb{U}L_u^P{\mathbb{G}^P}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\| \leq 1 \tag{3.112}$$

then Algorithm 3.3 guarantees the control input converge to a solution $(\vec{u}_\infty, \lambda_\mathbb{U})$ of the equations

$$\vec{u}_\infty - \vec{u}_0 = \vec{R}^{-1}{\mathbb{G}^P}^T\lambda_\mathbb{U} \tag{3.113}$$

and

$$\mathbb{G}^P\mathbb{U}\vec{u}_\infty + \vec{d}^P - \vec{r}^P = 0 \tag{3.114}$$

Proof. Note that, the error evolution of Algorithm 3.3 is shown as

$$\vec{e}_{k+1}^P = \left[I_{pM} - \mathbb{G}^P\mathbb{U}L_u^P{\mathbb{G}^P}^T(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\right]\vec{e}_k^P \tag{3.115}$$

where $L_u^P = \left[\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\mathbb{G}^P + \vec{R}\right]^{-1}$.

Combining the input law (3.107) with the error evolution (3.115) in the presence of model uncertainty, we have

$$\begin{aligned}\vec{u}_\infty - \vec{u}_0 =& \vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D}) + I_{pM}\right]^{-1} \\ &\times \sum_{j=0}^{\infty}\left[I_{pM} - \mathbb{G}^P\mathbb{U}L_{\mathbb{U}}^P\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\right]^j \vec{e}_0^P\end{aligned}$$

From error evolution (3.115), to guarantee the convergence of the tracking error norm, the model uncertainty must satisfy the following condition

$$\|I_{pM} - \mathbb{G}^P\mathbb{U}L_u^P\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\| \leq 1 \tag{3.116}$$

Under condition (3.116), $I_{pM} - \mathbb{G}^P\mathbb{U}L_u^P\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}\ (\boldsymbol{L}+\boldsymbol{D})$ is a positive definite matrix, with eigenvalues $0 < \lambda_i \leq 1$. We then define the space spanned by the eigenvector corresponding to unitary eigenvalues as $\mathcal{E}_{\mathbb{U}}^{\perp}$, yields

$$\begin{aligned}I_{pM} - \mathbb{G}^P\mathbb{U}L_u^P\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})|_{\mathcal{E}_{\mathbb{U}}} < I_{pM} \\ I_{pM} - \mathbb{G}^P\mathbb{U}L_u^P\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})|_{\mathcal{E}_{\mathbb{U}}^{\perp}} = I_{pM}\end{aligned} \tag{3.117}$$

For any initial input choice $\vec{u}_0$, define the corresponding consensus error as $\vec{e}_0^P$, its decomposition on $\mathcal{E}_{\mathbb{U}}$ and $\mathcal{E}_{\mathbb{U}}^{\perp}$ is

$$\vec{e}_0^P = \vec{e}_0^{P^{\mathcal{E}_{\mathbb{U}}}} + \vec{e}_0^{P^{\mathcal{E}_{\mathbb{U}}^{\perp}}} \tag{3.118}$$

and from (3.117), we have

$$\mathbb{G}^P\mathbb{U}L_u^P\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})|_{\mathcal{E}_{\mathbb{U}}^{\perp}}\vec{e}_0^{P^{\mathcal{E}_{\mathbb{U}}^{\perp}}} = 0 \tag{3.119}$$

Equation (3.116) becomes

$$\begin{aligned}\vec{u}_\infty - \vec{u}_0 =& \vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D}) + I_{pM}\right]^{-1} \\ &\times \sum_{j=0}^{\infty}\left[I_{pM} - \mathbb{G}^P\mathbb{U}L_u^P\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})|_{\mathcal{E}_{\mathbb{U}}}\right]^j \vec{e}_0^{P^{\mathcal{E}_{\mathbb{U}}^{\perp}}}\end{aligned}$$

and it follows that

$$\begin{aligned}\vec{u}_\infty - \vec{u}_0 =& \vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D}) + I_{pM}\right]^{-1} \\ &\times [\mathbb{G}^P\mathbb{U}L_u^P\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})]^{-1}\vec{e}_0^{P^{\mathcal{E}_{\mathbb{U}}^{\perp}}}\end{aligned}$$

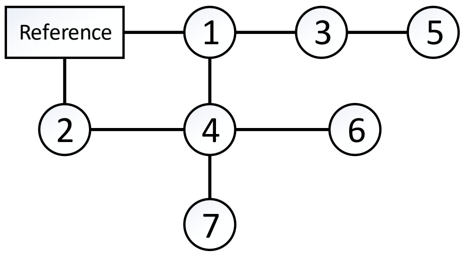

Figure 3.2: The graph structure of numerical examples

Setting $\lambda_{\mathbb{U}} = (\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\left[\mathbb{G}^P\vec{R}^{-1}\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})+I_{pM}\right]^{-1}\left[\mathbb{G}^P\mathbb{U}L_u^P\right.$ $\left.\mathbb{G}^{P^T}(\boldsymbol{L}+\boldsymbol{D})^T\vec{Q}(\boldsymbol{L}+\boldsymbol{D})\right]^{-1}\vec{e}_0^{P\mathcal{E}_{\mathbb{U}}^{\perp}}$, we then have

$$\vec{u}_\infty - \vec{u}_0 = \vec{R}^{-1}\mathbb{G}^{P^T}\lambda_{\mathbb{U}} \tag{3.120}$$

That completes the proof. □

Theorem 3.6 shows that Algorithm 3.3 converge to a perturbed solution when the uncertainty satisfies (3.112), and it returns good approximation for small model uncertainty.

3.6 Numerical Examples

In this section, two examples would be used to verify our proposed algorithms. All the following examples are constructed by a seven-agent networked system ($p = 7$) and the network topology is shown in Figure 3.2 (with unity weighting $W_{ij} = 1$). The desired reference is chosen as

$$r(t) = \left\{ \begin{array}{cc} sin(2\pi(t-1)) & 0 \le t < 2 \\ 0 & otherwise \end{array} \right. , \tag{3.121}$$

and then we assume the reference trajectory is defined on the time interval $[0, 3]$ with sampling time $T_s = 0.05s$ (sampled using a zero order hold). Furthermore, i^{th} subsystem's initial state condition $x_{i,0}$ and the first trial's input $u_{i,0}$ are assumed to be zero.

3.6.1 ILC for High Performance Consensus Tracking Task

We first consider a heterogeneous networked dynamical system, with i^{th} subsystem's dynamics is represented using the following discrete-time, minimum phase, low damping

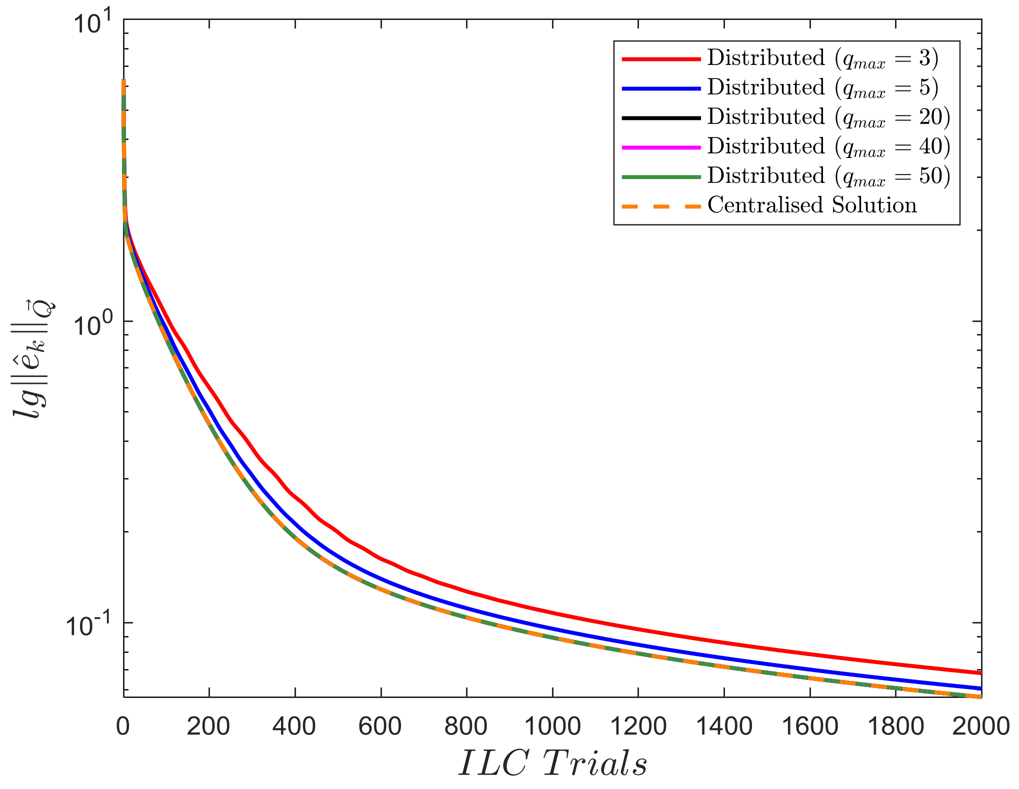

Figure 3.3: Effect of different q_{max} on convergence performance over 2000 trial

transfer function

$$G_i(s) = \frac{s + 10}{0.6 * (s^2 + s + \tau_i)} \tag{3.122}$$

where $\tau_i = i$ and $i = 1, 2, 3, \cdots, 7$.

For Algorithm 3.2, its convergence behaviour is influenced by different parameters, e.g., maximum-iteration q_{max}, weighting matrices Q and R, penalty parameter ρ. In the following, we will investigate these parameters individually.

To investigate the effect of q_{max}, the penalty parameter ρ is set to be 1, the scalar weighting $Q = R = 1$. Figure 3.3 shows how the convergence performance evolves for centralised and distributed implementation over 2000 ILC trials. For both implementations, they achieve monotonic convergence of the consensus tracking error norm $\|\hat{e}_k\|_{\vec{Q}}$ to zero, which verifies Theorem 3.1. Moreover, for distributed implementation, its convergence speed increases with the maximum-iteration q_{max}, however, the distributed result is almost coincide with the centralised result after $q_{max} \geq 20$. This implies that in this case, a small iteration number is enough for ADMM to approach the centralised result. However, it is still not clear how the the system parameters affect the ADMM iteration number, a more rigorously investigation will be done in the future.

Figure 3.4 demonstrates the tracking behaviour of Algorithm 3.2 at 2000^{th} ILC trial. It shows that for the subsystems with or without access to reference signal, they both

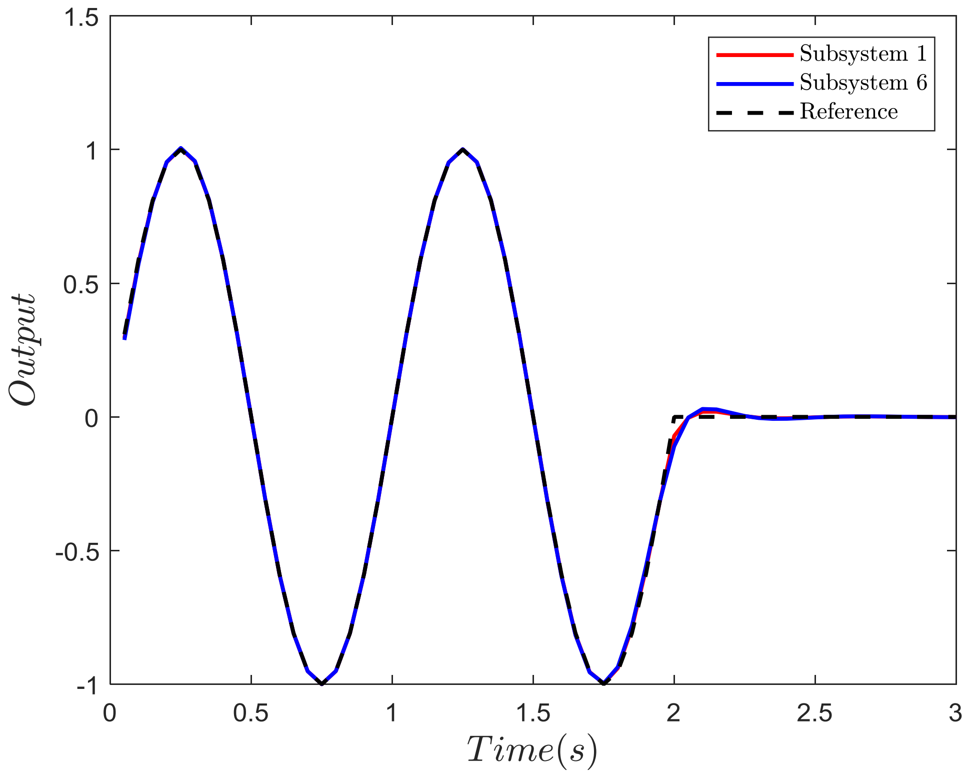

Figure 3.4: Output comparison for subsystems 1 and 6 at 2000th trial

achieve the perfect tracking of the desired reference. Figure 3.5 shows the input of seven subsystems at 2000th ILC trial. These input signal are different from each other because of the heterogeneous topology.

To investigate the effect of weighting matrices Q, R, the penalty parameter ρ is set to be 1, the maximum-iteration q_{max} is set to be 20 and the scalar weighting $Q = 1$. Figure 3.6 shows how the tracking error norm $\|\hat{e}_k\|_{\bar{Q}}$ evolves for different weighting R over the 2000 ILC trials. It can be seen from the figure that for different choice of R, the tracking error norm converges monotonically to zero. In addition, as R increases, the convergence speed decreases accordingly, i.e., a larger R results in slower convergence speed.

To investigate the effect of penalty parameter ρ, the scalar weighting $Q = R = 1$. By using Proposition 3.3 to calculate the optimal penalty parameter, we have the optimal result $\rho^* = 1$ (with the optimal corresponding convergence factor $\phi^* = 0.5$). Figure 3.7 shows the input accuracy evolution for different ρ in each ADMM iteration: beginning from a small ρ, the convergence speed of the proposed algorithm increases with the larger of penalty parameter ρ, however, the convergence speed will slow down after reaching the optimal penalty parameter $\rho^* = 1$, which is consistent with our expectations in Proposition 3.3.

To check whether $\rho = 1$ is the best choice in each ILC trial, we set the maximum ADMM iteration $q_{max} = 20$ and then operate the simulation with scalar weighting $Q = R = 1$.

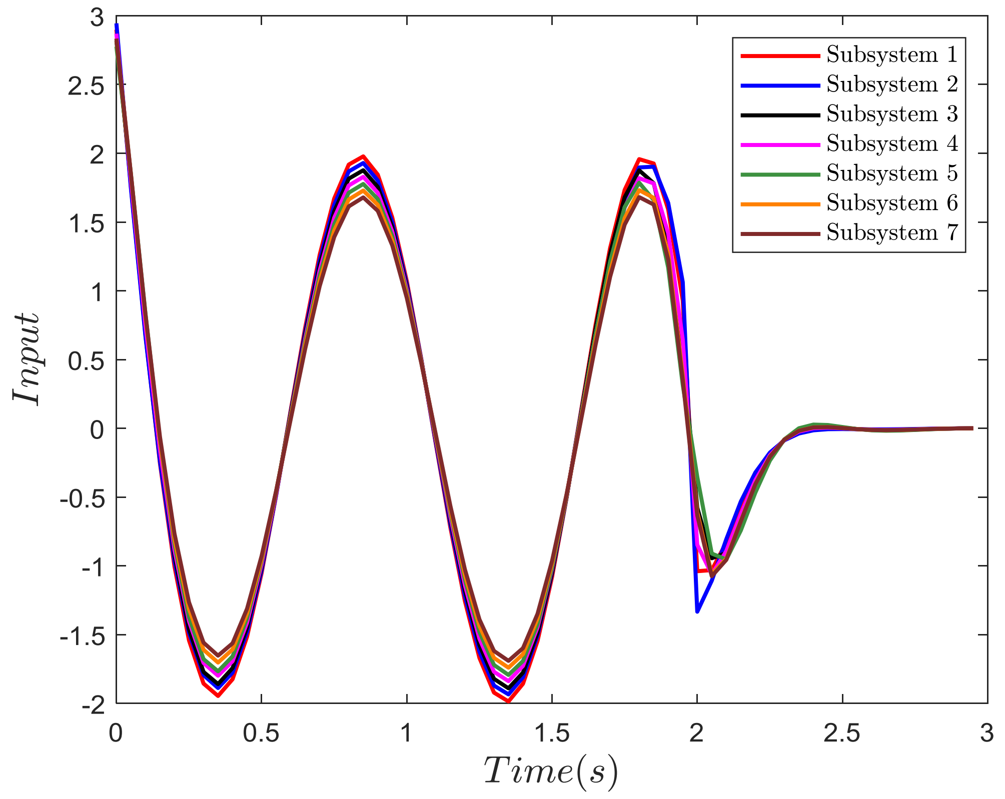

Figure 3.5: Input comparison between different subsystems at 2000th trial

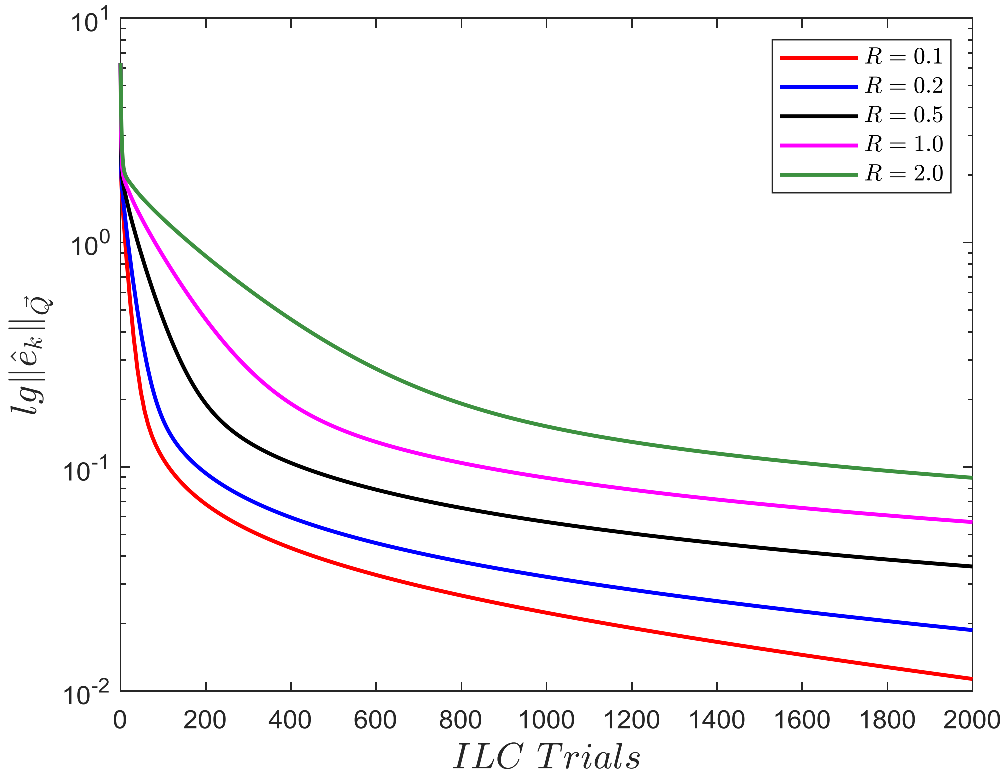

Figure 3.6: Effect of different R on convergence performance

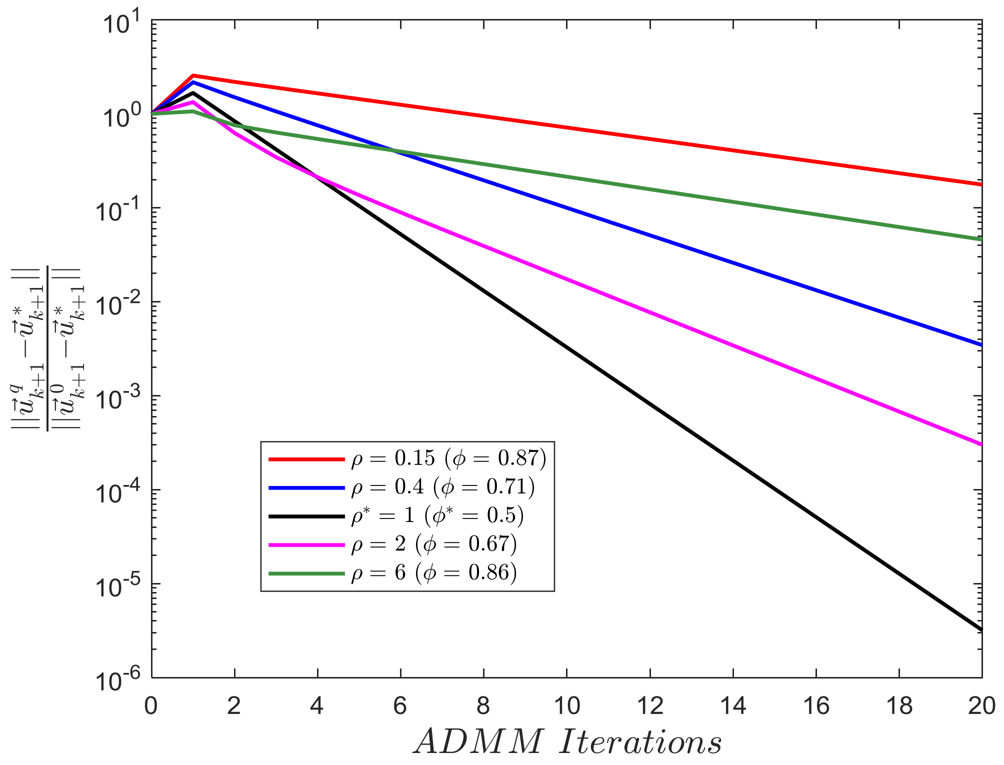

Figure 3.7: Convergence comparison for different ρ in ADMM

Figure 3.8 shows how the tracking error norm evolves for different ρ over the first ILC trial. The convergence speed of tracking error norm consists with the phenomenon summarised in Figure 3.7, which further verifies Proposition 3.3.

3.6.2 The Proposed Algorithms for Non-Minimum Phase Networked Dynamical Systems

As mentioned previously, the proposed algorithms are suitable to control non-minimum phase networked dynamical systems, and this session will investigate the algorithms' effectiveness for non-minimum phase dynamics. To make compression with the convergence result in Figure 3.3, we assume all the parameters remain unchanged except the minimum phase zero converts into non-minimum phase zero, i.e., the i^{th} subsystem's transfer function becomes

$$G_i(s) = \frac{s-10}{0.6*(s^2+s+\tau_i)} \tag{3.123}$$

where $\tau_i = i$ and $i = 1, 2, 3, \cdots, 7$.

Figure 3.9 shows how the convergence performance evolves for centralised and distributed implementation over 2000 ILC trials. For both implementations, they can still guarantee the monotonic convergence of the consensus tracking error norm $\|\hat{e}_k\|_{\vec{Q}}$. However, after a rapidly decreasing in the tracking error norm on the first few ILC trials, both algorithms will move along a plateau with little change on the tracking error norm, i.e., the

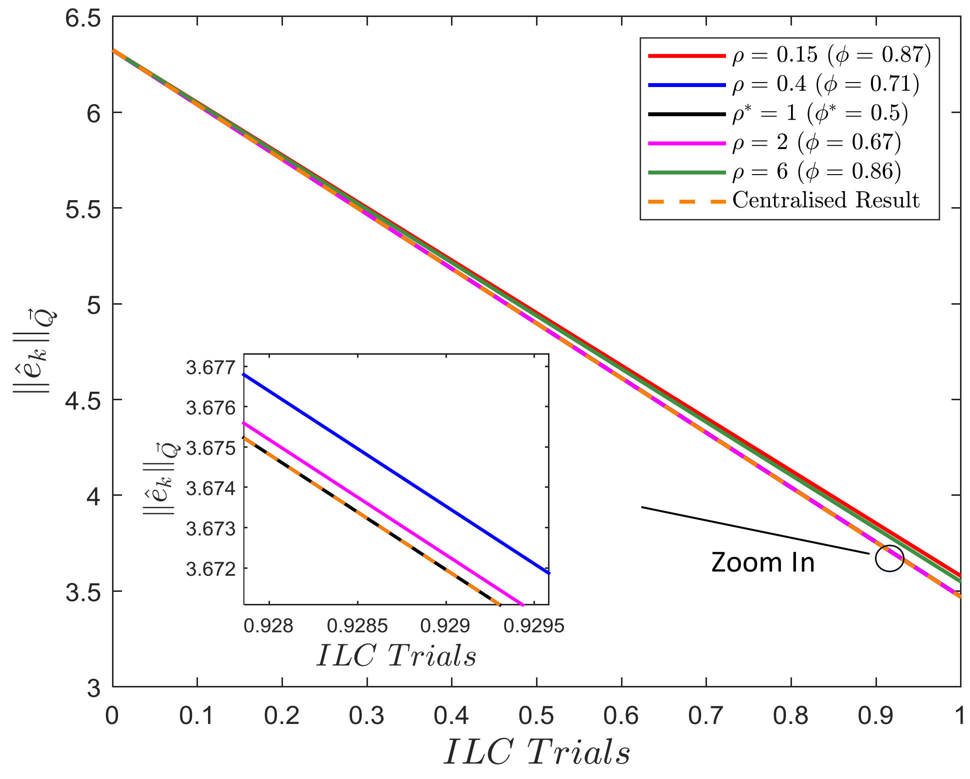

Figure 3.8: Convergence comparison for different ρ in ILC

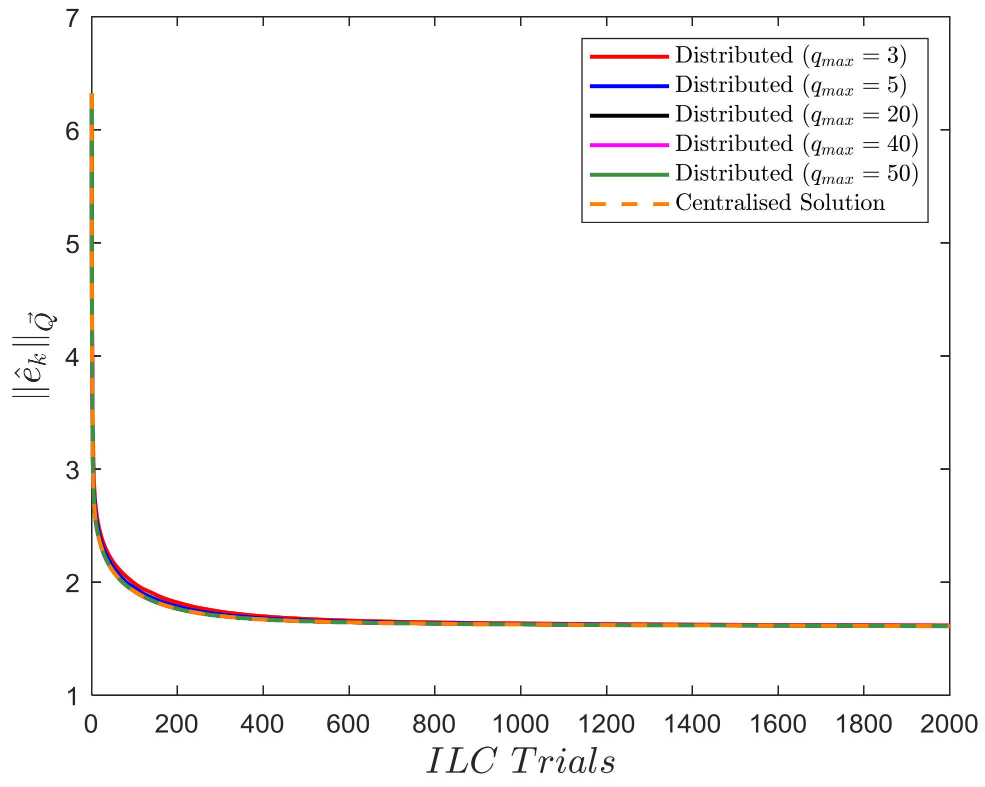

Figure 3.9: Effect of different q_{max} on convergence performance over 2000 trial

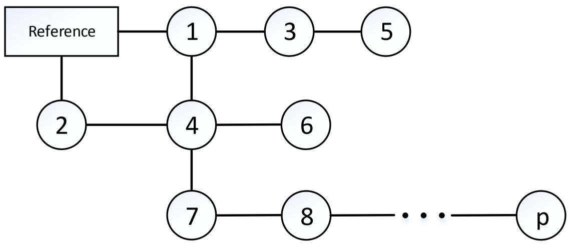

Figure 3.10: The extended graph structure for scalability case

convergence speed is almost zero. This is due to the existence of non-minimum phase zero, and the plateau value $\|\hat{e}_\infty\|_{\vec{Q}}$ is only an improvement of a factor of ~ 10 on the initial error $\hat{e}_0$. Please refer to Owens et al. (2014) for more details.

3.6.3 Scalability of the Distributed Algorithm 3.2

As mentioned previously, the proposed distributed Algorithm 3.2 has good scalability, i.e., it is applied to large scale network and it can deal with the dynamically growing network. We will investigate this property using the following simulation. Assuming all the new subsystems are directly connected to its previous index's subsystem, as shown in Figure 3.10. The new subsystems' transfer function are still in the same form as in (3.122), i.e.

$$G_i(s) = \frac{s+10}{0.6 * (s^2 + s + \tau_i)} \tag{3.124}$$

where $\tau_i = i$ and $i = 8, 9, \cdots, p$. For comparison, the other parameters remain the same. In the following section, we will first investigate the scalability of the distributed Algorithm 3.2 for large scale network.

3.6.3.1 Scalability of Algorithm 3.2 for Large Scale Network

To investigate the algorithm's scalability for large scale network, we assume the subsystem's number changes from 8 to 150, and accordingly record the computation time of both centralised and distributed implementation (with $q_{max} = 50$) for different subsystem's amount in the first trial, as shown in Table 3.1 and Figure 3.11. For the centralised Algorithm 3.1, its computation time will increase (in an exponential way) as the increasing of subsystem's number. This is due to the reason that Algorithm 3.1 requires a central controller to compute the input solution, and the controller's computation time will increase exponentially as the matrix dimension (that is related to the subsystem's amount) increases. Clearly, it is impossible to use Algorithm 3.1 for million-level networked dynamical systems in practice.

In contrast to Algorithm 3.1, it can be seen that distributed Algorithm 3.2 use the same computation time for different subsystem's amount, which shows great scalability for

Table 3.1: Subsystem Amounts V.S. Computation Time

Subsystem amounts	8	20	40	70	110	150
Time for centralised (s)	0.0239	0.2126	1.2532	5.7653	25.9331	62.3407
Time for distributed (s)	0.0338	0.0338	0.0309	0.0380	0.0381	0.0385

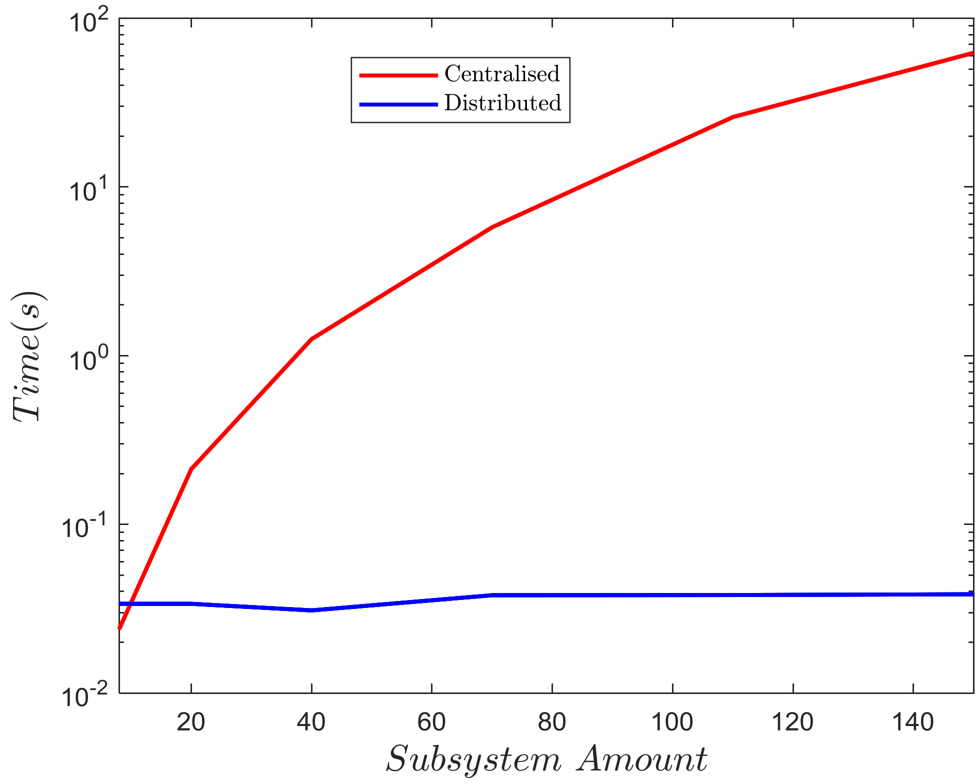

Figure 3.11: The consumption time against subsystem's amount

large scale network. For distributed algorithm, the global cost function has actually been separated into smaller local cost functions. Even for million-level networked dynamical systems, the dimension of each local cost function is relatively small. The calculation of each local cost function is done independently in different processors and hence the computation time for Algorithm 3.2 is almost not affected by the total subsystem's amount in the network.

3.6.3.2 Scalability of Algorithm 3.2 for Dynamically Growing Network

In this section, we will investigate the scalability of the distributed algorithm for the dynamically growing network. We assume a new subsystem 8 is directly connected to subsystem 7 at trial 60. By fixing the maximum-iteration number $q_{max} = 20$, the penalty parameter $\rho = 1$, scalar weighting $Q = R = 1$, we have the simulation result as shown in Figure 3.12. It can be concluded from the figure that although the consensus tracking

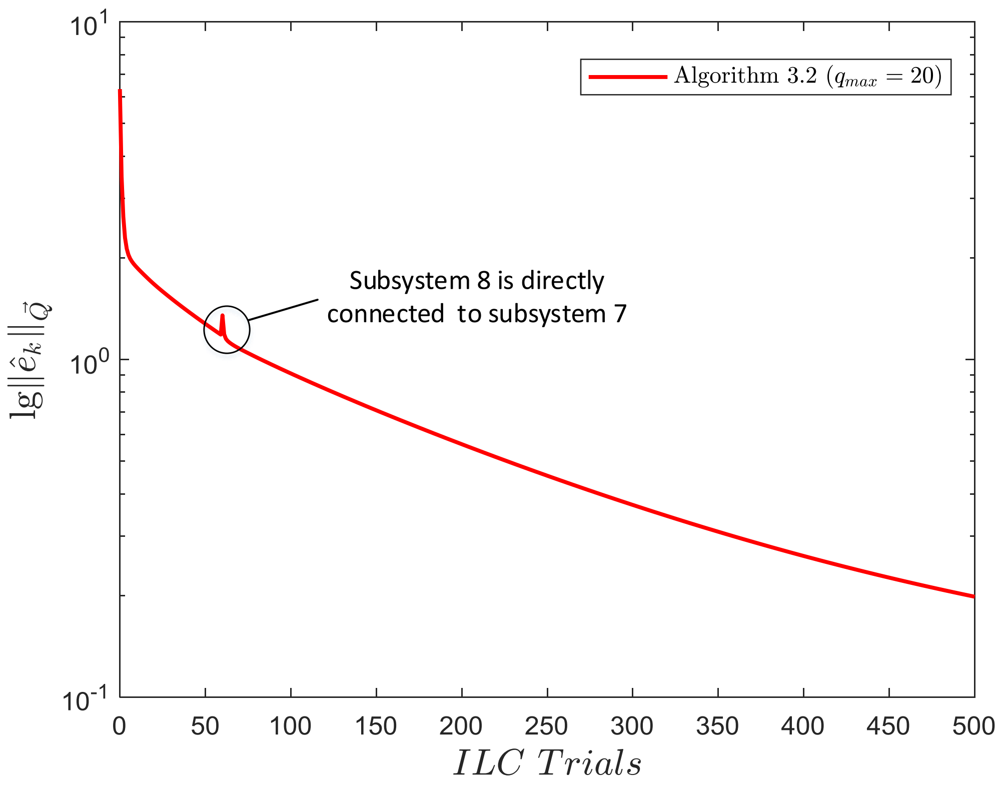

Figure 3.12: The evolution of convergence performance when a new subsystem for the dynamically growing network

error norm grows to a larger number when subsystem 8 is first added to the network, the tracking error norm converges monotonically after that. It should be noticed that, none of the parameters needed to be returned after adding the new subsystem, which demonstrates the scalability of Algorithm 3.2 for the dynamically growing network.

3.6.4 ILC for P2P Consensus Tracking Task

In the following simulation, we consider the same system as in Section 3.6.1, and define the time instant set as

$$\Lambda = [5 \quad 20 \quad 35]^T \tag{3.125}$$

with the P2P reference point $\vec{r}^P$ defined as

$$\vec{r}^P = [1 \quad 0 \quad -1]^T \tag{3.126}$$

We choose the best parameter have been found in Section 3.6.1 (i.e., maximum ADMM iteration $q_{max} = 20$, penalty parameter $\rho = 1$, scalar weighting $Q = R = 1$) and then investigate the performance of P2P Algorithm 3.3. Figure 3.13 shows all the subsystem's output curves on 2000^{th} ILC trial, demonstrating that the control target is achieved by all the subsystems (even when the reference signal is only known by subsystem 1 and 2).

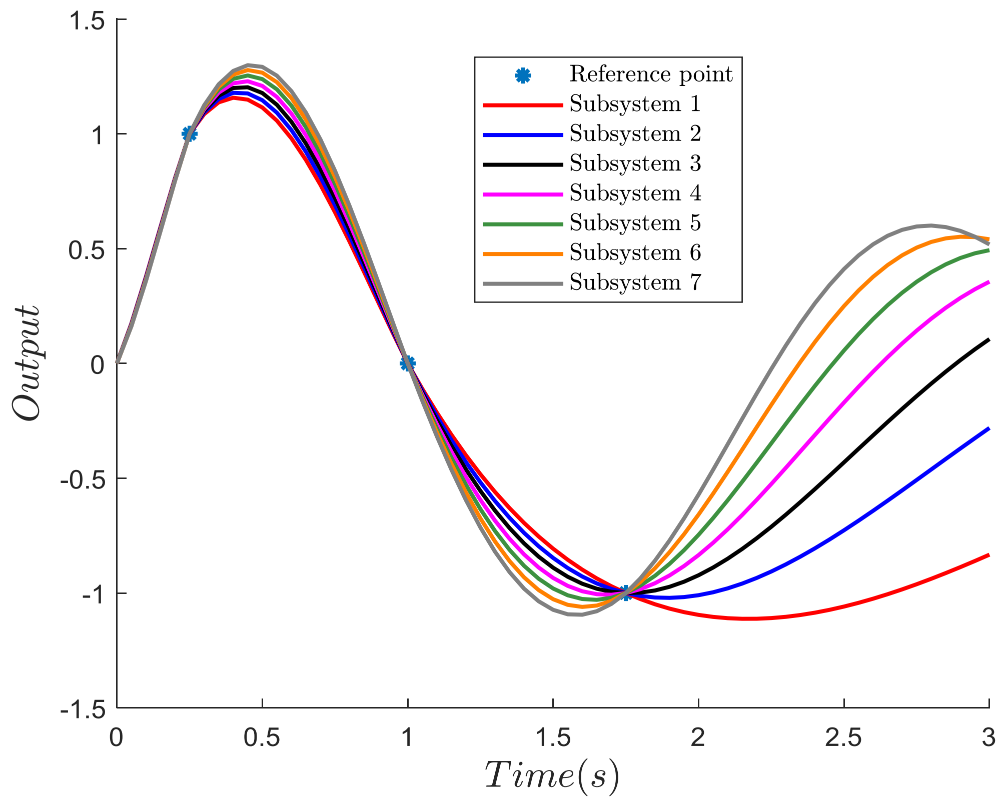

Figure 3.13: Output comparison for different subsystems at 2000^{th} trial

Figure 3.14 presents the evolution of system's input energy consumption over 2000 ILC trials, showing that the system's input energy cost does converge to the optimal input solution $\|\vec{u}^*\|^2_{\vec{R}} = 63.6020$, which verifies Theorem 3.4.

Next, we investigate the robustness performance of the proposed P2P algorithm. For simplicity, we define the multiplicative uncertainty $U_i(z)$ as a diagonal matrix, which is shown in the following structure

$$U_i(z) = \kappa I_N, \qquad i = 1, \cdots, p. \tag{3.127}$$

We set the scalar weighting $Q = R = 1$, and use the Condition C4 (in Theorem 3.5) to find the allowable range of κ that can guarantee the *Robust Monotone Convergence*. Condition C4 indicates that when $0 < \kappa \leq 2.0205$, the P2P algorithm will converge monotonically to zero. To further verify if this statement is correct or not, we choose different κ for numerical simulation. Figure 3.15 shows the convergence of $\|\hat{e}^P_k\|_{\vec{Q}}$ for different uncertainty over 50 ILC trials. The simulation results in Figure 3.15 conform to the conclusion obtained in Condition C4, which further verifies Theorem 3.5. Simulations with different reference trajectory, trial length, and damping ratio (including critically damped and over-damped cases) have also been investigated. The proposed algorithms work for all of the cases, hence they are omitted here for brevity.

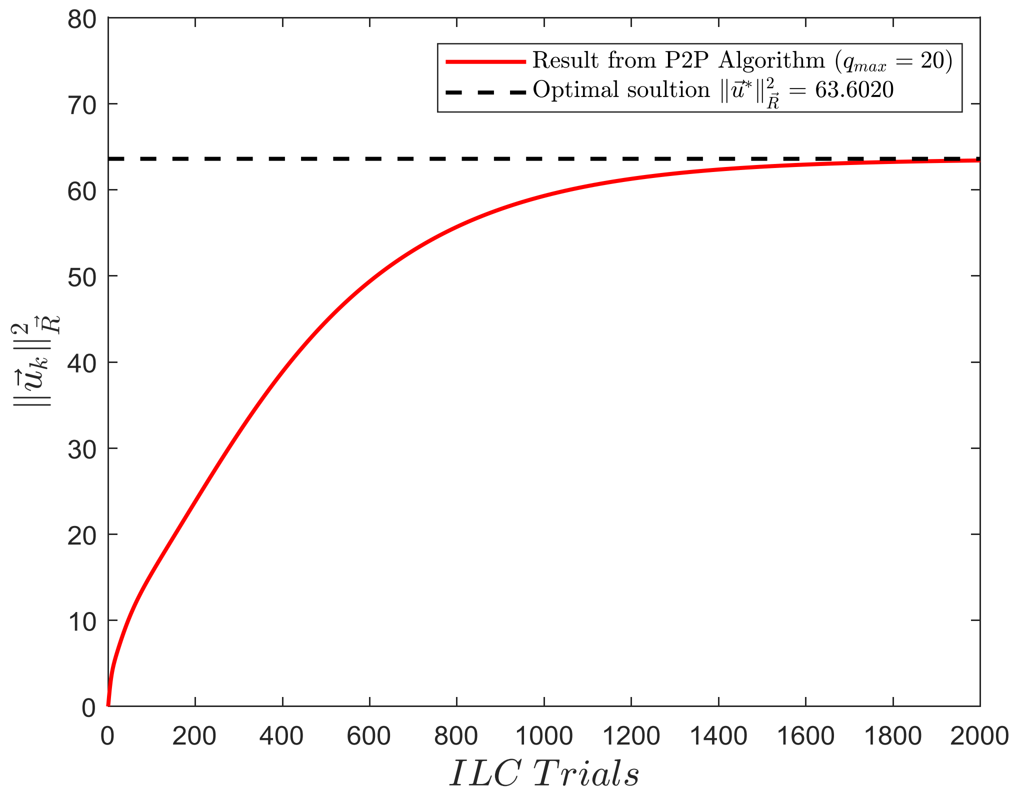

Figure 3.14: Input energy cost of the proposed P2P algorithm

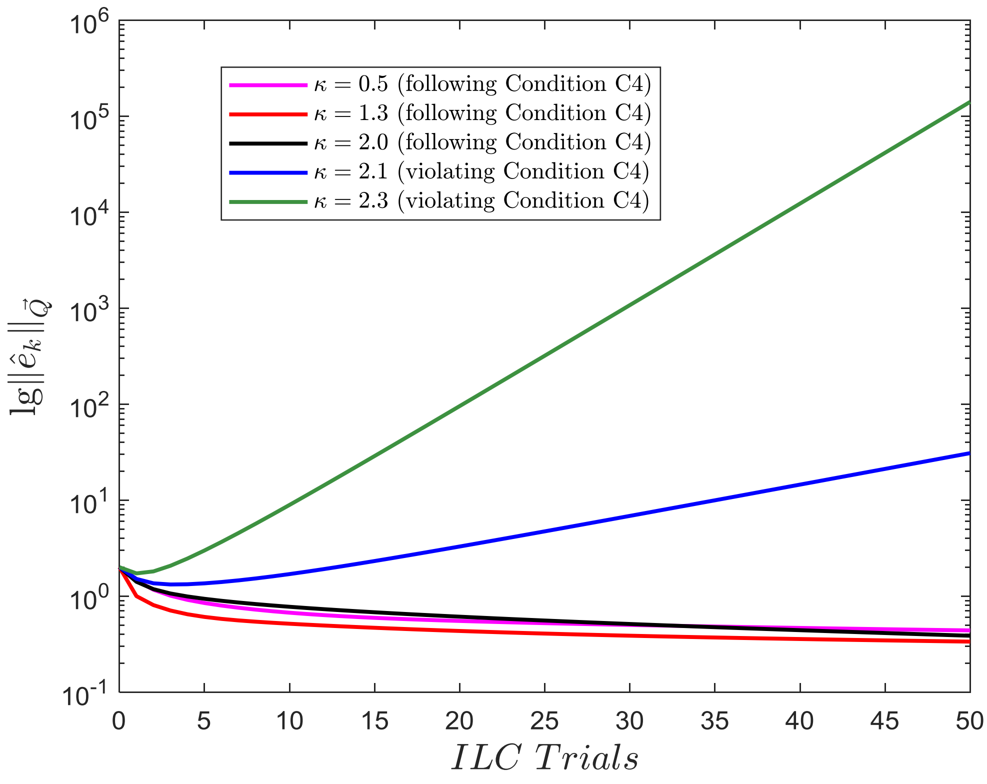

Figure 3.15: Convergence comparison between different κ

3.7 Summary

This chapter proposes a novel ILC framework for consensus tracking of networked dynamical systems working in a repetitive manner using the well-known norm optimal ILC framework. The resulting algorithm achieves perfect consensus tracking of the desired reference with tracking error norm reducing monotonically to zero and has certain degree of robustness against the model uncertainty. The algorithm can be applied to heterogeneous networked systems and non-minimum phase systems, which is appealing in practice. In addition, we extend the proposed ILC framework to solve the P2P tracking problem (that is an important task in networked dynamical systems), with the convergence and robustness properties remaining unchanged. Convergence and robustness properties of the proposed algorithms are analysed rigorously, and distributed implementations of the algorithms are derived using ADMM so they can be applied to very large scale networked dynamics (with great scalability). Feedback plus feedforward implementation and the method to choose the best penalty parameter ρ are also given. Numerical simulations are presented to illustrate the effectiveness of the proposed algorithms. However, we have not considered the system constraint (that is widely existing in practice) in the design. To address this limitation, we will propose a novel ILC algorithm to solve the constraint handling problem in the next chapter.

Chapter 4

Distributed NOILC for Constrained Consensus Tracking Problem

System constraints are widely existing in practice, often relating to performance requirements or physical limitations. As an example, within a group of unmanned aerial vehicles (UAVs) operating together, each UAV has an allowable input (voltage) range. If one agent's applied input exceeds its acceptable voltage range, it may damage that UAV and further affect the mission of the whole networked system. These constraints, however, are not considered by most existing research of ILC for consensus tracking problem. The only exceptions are the recent works in Shen and Xu (2017, 2018); Yang and Li (2019), which consider the constrained ILC design for nonlinear networked systems using adaptive or barrier/composite Lyapunov function methods that can only guarantee asymptotic convergence of tracking error norm (rather than monotonic convergence that is desirable in practice) under certain conditions.

To address the above limitations, this chapter uses a successive projection framework (that has been successfully applied to single system ILC design in Chu and Owens (2010)) to formulate the constrained consensus tracking problem and based on this formulation develops two novel ILC algorithms for constrained consensus tracking problem, further extending the work on ILC for unconstrained networked systems in Chapter 3. For both algorithms, they guarantee the consensus tracking error norm to converge to zero when perfect consensus tracking is possible, whereas the convergence speed of one algorithm is faster than the other at the cost of more computation complexity. When perfect consensus tracking is unachievable, the computational complex algorithm always converges to the best result that can be achieved, while the other converges to the result that is influenced by the input and output weighting matrices. Furthermore, all the algorithms can be applied to both homogeneous and heterogeneous networks, as well as nonminimum phase systems, which are desirable in practice. To avoid the computational difficulties for very large scale networked systems, we propose distributed implementations for the proposed ILC algorithm using the idea of the alternating direction method of multipliers (ADMM) that has been widely used in different areas (e.g., Boyd et al. (2011); Zhang et al. (2019a)), allowing the algorithm to be implemented distributively using only local information. This chapter is based on the work from Chen et al. (2020).

This chapter is organised as follows: Section 4.1 provides the formulation of the system dynamics, network topology, system constraints and defines the constrained consensus ILC problem; Section 4.2 reviews the idea of successive projection framework and then reformulates the constrained consensus ILC problem into successive projection framework; Sections 4.3 and 4.4 introduce two novel ILC algorithms (including distributed implementations) for the constrained consensus tracking problem, with the algorithms' convergence properties analysed rigorously; Section 4.5 provides numerical examples to demonstrate the algorithms' effectiveness and Section 4.6 summaries this chapter.

4.1 Problem Formulation

In this section, we provide the formulation of the system dynamics, network topology, system constraints, and define the constrained ILC design problem. For simplicity, we consider a single input, single output (SISO) system.

4.1.1 System Dynamics

The system description is the same as in Chapter 3, but for completeness, descried in the following again. Consider a networked system (either homogeneous or heterogeneous) consisting of p subsystems, with i^{th} $(1 \leq i \leq p)$ subsystem's dynamics a discrete-time, linear time invariant (LTI) system described as follows

$$\begin{aligned} x_{i,k}(t+1) &= A_i x_{i,k}(t) + B_i u_{i,k}(t), \quad x_{i,k}(0) = x_{i,0} \\ y_{i,k}(t) &= C_i x_{i,k}(t) \end{aligned} \tag{4.1}$$

where k is the trial index, $t \in [0, N]$ is the time; A_i, B_i, C_i are system matrices with appropriate dimensions; $x_{i,k}(\cdot) \in \mathbb{R}^{n_i}$ (n_i is i^{th} subsystem's order), $u_{i,k}(\cdot)$, $y_{i,k}(\cdot)$ are the state, input, output of subsystem i. For the consensus tracking problem, all the subsystems are required to track the same reference signal $r(t)$ within time interval $[0, N]$ repetitively. At time instance $t = N + 1$, the time t is reset to 0, the state of i^{th} subsystem is reset to the initial condition $x_{i,0}$, and the subsystems are required to track the same reference signal again. For such problem, all the subsystems need to conclude the best control strategy while only part of the subsystems know the reference, which makes the design non-trivial.

To facilitate later ILC design, a 'lifted form' representation is introduced (Hatonen et al., 2004). Assume each subsystem's relative degree is one (i.e., $C_iB_i \neq 0$), the input $u_{i,k}$, output $y_{i,k}$, reference r are denoted as

$$\begin{aligned} u_{i,k} &= [u_{i,k}(0) \quad u_{i,k}(1) \quad \cdots \quad u_{i,k}(N-1)]^T \in \mathbb{R}^N \\ y_{i,k} &= [y_{i,k}(1) \quad y_{i,k}(2) \quad \cdots \quad y_{i,k}(N)]^T \in \mathbb{R}^N \\ r &= [r(1) \quad r(2) \quad \cdots \quad r(N)]^T \in \mathbb{R}^N \end{aligned} \tag{4.2}$$

The system model (4.1) can then be rewritten as

$$y_{i,k} = G_i u_{i,k} + d_i \tag{4.3}$$

where system matrix G_i is defined as

$$G_i = \begin{bmatrix} C_iB_i & 0 & \cdots & 0 \\ C_iA_iB_i & C_iB_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_iA_i^{N-1}B_i & C_iA_i^{N-2}B_i & \cdots & C_iB_i \end{bmatrix} \tag{4.4}$$

and the initial condition's respond d_i is denoted as

$$d_i = \begin{bmatrix} C_iA_ix_{i,0} & C_iA_i^2x_{i,0} & \cdots & C_iA_i^Nx_{i,0} \end{bmatrix}^T \tag{4.5}$$

Introducing $\vec{u}_k$, $\vec{y}_k$, $\vec{d}$, $\vec{r}$ as the global vectors of $u_{i,k}$, $y_{i,k}$, d_i, r, i.e.

$$\begin{aligned} \vec{u}_k &= \begin{bmatrix} u_{1,k}{}^T & u_{2,k}{}^T & \cdots & u_{p,k}{}^T \end{bmatrix}^T \\ \vec{y}_k &= \begin{bmatrix} y_{1,k}{}^T & y_{2,k}{}^T & \cdots & y_{p,k}{}^T \end{bmatrix}^T \\ \vec{d} &= \begin{bmatrix} d_1{}^T & d_2{}^T & \cdots & d_p{}^T \end{bmatrix}^T \\ \vec{r} &= \begin{bmatrix} r^T & r^T & \cdots & r^T \end{bmatrix}^T, \end{aligned} \tag{4.6}$$

then, the global system model of (4.1) can be represented as

$$\vec{y}_k = \mathbb{G}\vec{u}_k + \vec{d} \tag{4.7}$$

where $\mathbb{G} = \text{diag}(G_1, G_2, \cdots, G_p)$.

Now, the high performance consensus tracking of networked dynamical systems can be stated as finding an appropriate global input $\vec{u}_k$ such that the global output $\vec{y}_k$ tracks the desired reference $\vec{r}$ perfectly.

4.1.2 Network Topology

For simplicity, the network topology in this chapter is represented using an undirected graph $\mathscr{G} = (\mathscr{V},\mathscr{E})$, in which the vertex set $\mathscr{V} = \{1, 2, \cdots, p\}$, and edge set $\mathscr{E} \subset \mathscr{V} \times \mathscr{V}$. When there exists an edge between vertexes i and j, the vertexes i and j are neighbours, which denoted as $\mathscr{N}_i := \{j : (i,j) \in \mathscr{E}\}$.

To represent the topology relationship between different subsystems, we introduce the adjacency matrix $\mathscr{A} = [a_{ij}]$, with its element a_{ij} defined as

$$a_{ij} = \begin{cases} W_{ij} & if\ (i,j) \in \mathscr{E} \\ 0 & otherwise \end{cases} \tag{4.8}$$

where weight W_{ij} is often considered as the connection strength of the edge. Based on the i^{th} node's neighbours set, the degree of a node i is defined as $d(i) = \sum_{j=1}^{p} a_{ij}$ and it follows by the degree matrix $\mathscr{D} = diag(d(1), d(2), \cdots d(p))$. Using the definition of adjacency matrix and degree matrix, the Laplacian matrix is defined as $L = \{l_{ij}\} := \mathscr{D} - \mathscr{A}$, which is a real symmetric matrix with element l_{ij} defined as below

$$l_{ij} = \begin{cases} -W_{ij} & if \quad j \in \mathscr{N}_i \\ \sum_{j\in\mathscr{N}_i} W_{ij} & if \quad j = i \\ 0 & otherwise \end{cases} \tag{4.9}$$

For the high performance consensus tracking problem, only few subsystems have access to the reference signal, and hence we introduce a reference-accessibility matrix $D = \text{diag}\{\mathcal{D}_{ii}\}$ to represent this relationship. The diagonal element $\mathcal{D}_{ii}$ is denoted as

$$\mathcal{D}_{ii} = \begin{cases} 1 & if \text{ subsystem } i \text{ has access} \\ 0 & if \text{ subsystem } i \text{ does not have access} \end{cases} \tag{4.10}$$

In this chapter, the following standard assumptions for high performance consensus tracking problem are required:

Assumption 4.1. *At least one Eulerian path from one vertex to other vertices, i.e., the graph $\mathscr{G}$ is connected.*

Assumption 4.2. *At least one of the subsystems has direct access to the reference trajectory, i.e., $\mathcal{D}_{ii} \neq 0$.*

Remark 4.1. *Assumption 4.1 is commonly required to achieve consensus tracking task. If there exists one subsystem has not relationship with any other subsystems (i.e., it cannot receive any information from others), then it is impossible for that subsystem to keep consensus with other subsystems.*

Remark 4.2. *Assumption 4.2 guarantees the reference information is available to at least one of the subsystems. Note that, when there is not reference in the system, the* *proposed algorithms can still achieve the consensus of the subsystems (without tracking any reference).*

4.1.3 System Constraints

System constraints often relate to performance requirements (e.g., input energy cost) or physical limitations (e.g., output saturation), and hence they are widely existing in practical applications. For simplicity, this chapter considers input constraints, however, all the results can be extended to other types of system constraints without any difficulties. Assuming the input u_i of i^{th} subsystem is constrained in a closed convex set Ω_i, then some examples of input constraints can be formulated as follows:

- Input sign constraints:

$$\Omega_i = \left\{ u_i \in \mathbb{R}^N : \quad 0 \le u_i(t) \right\} \tag{4.11}$$

- Input saturation constraints:

$$\Omega_i = \left\{ u_i \in \mathbb{R}^N : \quad |u_i(t)| \le M_i(t) \right\} \tag{4.12}$$

- Input energy constraints:

$$\Omega_i = \left\{ u_i \in \mathbb{R}^N : \quad \sum_{t=0}^{N-1} u_i^2(t) \le M_i \right\} \tag{4.13}$$

- Input amplitude constraints:

$$\Omega_i = \left\{ u_i \in \mathbb{R}^N : \quad \lambda_i(t) \le u_i(t) \le \mu_i(t) \right\} \tag{4.14}$$

The global input constraint set Ω can then be defined as

$$\Omega = \Omega_1 \times \Omega_2 \times \Omega_3 \times \cdots \times \Omega_p \tag{4.15}$$

in which $\times$ represents the Cartesian product.

4.1.4 Constrained Iterative Learning Control Design

In this chapter, the constrained ILC design problem can be stated as finding a proper input updating law in the following form

$$\vec{u}_{k+1} = f(\vec{u}_k, \vec{e}_k), \qquad \forall k \ge 0 \tag{4.16}$$

such that

$$\begin{aligned} u_{i,k} &\in \Omega_i, & \forall k \geq 0 \\ \lim_{k\to\infty} y_{i,k} &= r, & i = 1, 2, \cdots, p \end{aligned} \tag{4.17}$$

where $\vec{e}_k = \vec{r} - \vec{y_k}$ represents the 'virtual' consensus tracking error and it is not generally known by all the subsystems (since not all the subsystems have access to the reference information). It should be noted that the constraint requirement makes the design non-trivial, since the algorithm is required to guarantee the satisfaction of the constraints and to achieve the high performance consensus tracking simultaneously.

4.2 Constrained Consensus Tracking using Successive Projection Method

In this section, we first review the general idea of the successive projection framework and then reformulate the constrained consensus tracking problem using the successive projection framework.

4.2.1 Overview of Successive Projection Framework

The successive projection method described in Owens and Jones (1978) is an efficient technique for finding the intersection between two closed, convex sets K_1 and K_2 in some real Hilbert space H. By selecting an initial point k_0 in the Hilbert space H, the subsequent points are obtained by alternatively projecting the previous points onto one set and then the other set. The properties of the successive projection method are formally described in the following theorem.

Theorem 4.1. *(Owens and Jones, 1978) Let $K_1 \in H$, $K_2 \in H$ be two closed, convex sets in a real Hilbert space H with $K_1 \cap K_2$ non-empty. Define*

$$K_M = \begin{cases} K_1, & M \text{ odd} \\ K_2, & M \text{ even} \end{cases} \tag{4.18}$$

Then, given the initial guess $k_0 \in H$, the sequence $\{k_M\}_{M\geq 0}$ satisfying

$$\|k_M - k_{M-1}\| = \min_{k\in K_M} \|k - k_{M-1}\|, \quad M \geq 1 \tag{4.19}$$

with $k_M \in K_M, M \geq 1$, is uniquely defined for each $k_0 \in H$ and satisfies

$$\|k_{M+1} - k_M\| \leq \|k_M - k_{M-1}\|, \quad M \geq 2 \tag{4.20}$$

where $\|\cdot\|$ is the induced norm in the Hilbert space H.

Furthermore, for any $x \in K_1 \cap K_2$,

$$\|x - k_M\|^2 \geq \|x - k_{M+1}\|^2 + \|k_{M+1} - k_M\|^2, \tag{4.21}$$

so that the sequence $\{\|x - k_{M+1}\|^2\}_{M\geq 0}$ is monotonically decreasing and $\{k_M\}_{M\geq 0}$ continuously gets closer to every point in $K_1 \cap K_2$. In addition

$$\sum_{M=1}^{\infty} \|k_{M+1} - k_M\|^2 \leq \|x - k_1\|^2, \tag{4.22}$$

so that, for each $\epsilon \geq 0$, there exists an integer N such that for $M \geq N$

$$\inf_{k \in K_{M+1}} \|k - k_M\| < \epsilon. \tag{4.23}$$

that is, the iterates $k_M \in K_M$ become arbitrarily close to K_{M+1}.

Moreover, when $K_1 \cap K_2$ is empty, the algorithm converges in the sense that $\|k_{M+1} - k_M\| \to d(K_1, K_2)$ defining the minimum distance $d(K_1, K_2)$ between two sets K_1 and K_2.

4.2.2 Successive Projection Formulation for Constrained Consensus Tracking Problem

To illustrate the idea of constrained ILC design, we initially consider the ILC design problem without any constraints. For the general consensus tracking problem, it can be formulated into the successive projection framework by defining two closed, convex sets in the Hilbert space $H = \mathcal{U} \times \mathcal{Y}$ (where $\mathcal{U} \in \mathbb{R}^{pN}$ and $\mathcal{Y} \in \mathbb{R}^{pN}$) as follows

- $S_1 = \left\{(\vec{e}, \vec{u}) \in H : \vec{e} = \vec{r} - \mathbb{G}\vec{u} - \vec{d}\right\}$;
- $S_2 = \{(\vec{e}, \vec{u}) \in H : \vec{e} = 0\}$

with inner products, induced norms for $\mathcal{U}$, $\mathcal{Y}$ defined as

$$\begin{aligned} \langle \vec{u}, \vec{v} \rangle_{\mathcal{U}} &= \vec{u}^T \vec{R} \vec{v}, \qquad \|\vec{u}\|_{\mathcal{U}} = \sqrt{\langle \vec{u}, \vec{u} \rangle_{\mathcal{U}}} \\ \langle \vec{e}, \vec{z} \rangle_{\mathcal{Y}} &= \vec{e}^T \vec{Q} \vec{z}, \qquad \|\vec{e}\|_{\mathcal{Y}} = \sqrt{\langle \vec{e}, \vec{e} \rangle_{\mathcal{Y}}} \end{aligned} \tag{4.24}$$

in which $\vec{Q} = \boldsymbol{L_D^T} \bar{Q} \boldsymbol{L_D}$ with $\boldsymbol{L_D} = (L + D) \otimes I_N$, I_N is an N by N identity matrix, $\otimes$ is the Kronecker product and $\bar{Q} = \text{diag}(Q, Q, \cdots, Q)$, $\vec{R} = \text{diag}(R, R, \cdots, R)$ (where $Q \in \mathbb{R}^N$ and $R \in \mathbb{R}^N$ are positive definite matrices, therefore $\vec{Q}$ is also a positive definite matrix). The inner product and associated induced norm for the Hilbert space H are naturally defined as

$$\begin{aligned} \langle (\vec{e}, \vec{u}), (\vec{z}, \vec{v}) \rangle &= \vec{e}^T \vec{Q} \vec{z} + \vec{u}^T \vec{R} \vec{v} \\ \|(\vec{e}, \vec{u})\| &= \sqrt{\langle (\vec{e}, \vec{u}), (\vec{e}, \vec{u}) \rangle} \end{aligned} \tag{4.25}$$

Now, by introducing the constrained set as follows

- $S_3 = \{(\vec{e}, \vec{u}) \in H : \vec{u} \in \Omega\}$,

the constrained consensus tracking problem can be stated as finding the interaction of three different sets (i.e., $S_1 \cap S_2 \cap S_3$). Note that, perfect consensus tracking is not always possible under the constraint S_3, in particular, when $S_1 \cap S_2 \cap S_3 = \emptyset$, the perfect consensus tracking cannot be achieved.

It seems like there has three sets in the constraint handling problem and the successive projection method proposed in Theorem 4.1 cannot be applied. However, by incorporating set S_3 into either S_1 (i.e., $S_1 \cap S_3$) or S_2 (i.e., $S_2 \cap S_3$), the 3-set problem is formulated into a 2-set problem, which can be solved using the successive projection method. Based on different combinations of sets S_1, S_2 and S_3, we can generate two novel ILC frameworks for constrained consensus tracking problem, and then use ADMM to implement the ILC framework in a distributed manner (including the projection step computed using ADMM). Note that, the existence of the network topology makes the distributed design differ from the single-agent case, which will be described in the following sections.

4.3 Constrained Consensus ILC Algorithm 4.1

For this algorithm, we incorporate the input constraints with the system dynamics, and hence it requires more computational complexity than another formulation (which will be introduced later). In this cases, the set $K_1 = S_1 \cap S_3$ and $K_2 = S_2$, i.e.

- $K_1 = \left\{(\vec{e}, \vec{u}) \in H : \vec{e} = \vec{r} - \mathbb{G}\vec{u} - \vec{d}, \vec{u} \in \Omega\right\}$;
- $K_2 = \{(\vec{e}, \vec{u}) \in H : \vec{e} = 0\}$

Figure 4.1 illustrates that when $K_1 \cap K_2 \neq \emptyset$, perfect consensus tracking is possible; otherwise, perfect consensus tracking is impossible, as shown in Figure 4.2, e.g., the reference signal is designed inadequately and the subsystem's actuator could not simultaneously guarantee the tracking performance and the satisfaction of system constraints.

4.3.1 Algorithm Description

Algorithm 4.1. *Given any initial input $\vec{u}_0$ satisfying the input constraint set Ω, the input law defined as follows*

$$\vec{u}_{k+1} = \arg\min_{\vec{u} \in \Omega} \left\{ \|\vec{e}_{k+1}\|^2_{\vec{Q}} + \|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}} \right\} \tag{4.26}$$

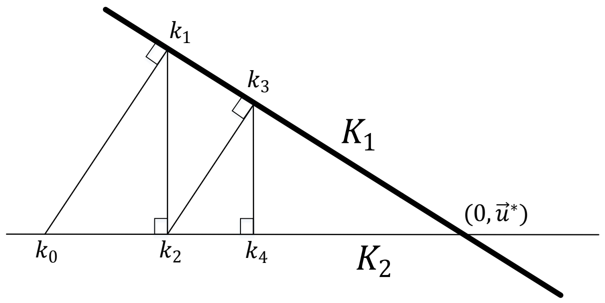

Figure 4.1: Algorithm 4.1 – perfect consensus tracking is possible

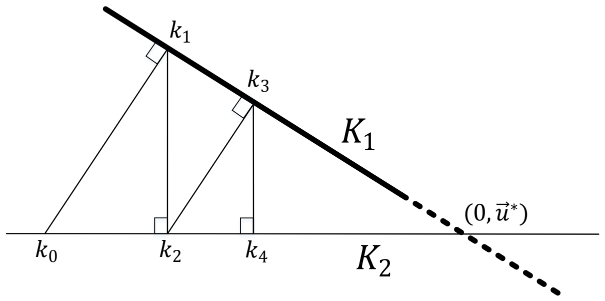

Figure 4.2: Algorithm 4.1 – perfect consensus tracking is impossible

where $\vec{e}_{k+1} = \vec{e}_k - \mathbb{G}(\vec{u}_{k+1} - \vec{u}_k)$, *iteratively solves the constrained ILC problem in the input constraints set* Ω, *i.e.*

$$\begin{aligned} u_{i,k} &\in \Omega_i, & \forall k \geq 0 \\ \lim_{k\to\infty} y_{i,k} &= r, & i = 1, 2, \cdots, p \end{aligned} \tag{4.27}$$

Remark 4.3. *By definition, the input updating law (4.26) can be rewritten as follow*

$$\vec{u}_{k+1} = \arg\min_{\vec{u}\in\Omega} \left\{ \|(\boldsymbol{L} + \boldsymbol{D})\vec{e}_{k+1}\|^2_{\bar{Q}} + \|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}} \right\}, \tag{4.28}$$

in which $\boldsymbol{L} = L \otimes I_N$, $\boldsymbol{D} = D \otimes I_N$, $\|(\boldsymbol{L} + \boldsymbol{D})\vec{e}_{k+1}\|^2_{\bar{Q}}$ *represents the quadratic form* $\vec{e}^T_{k+1}(\boldsymbol{L} + \boldsymbol{D})^T\bar{Q}(\boldsymbol{L} + \boldsymbol{D})\vec{e}_{k+1}$ *and similarly with* $\|\cdot\|^2_{\vec{R}}$. *It should be noted that, the existence of Laplacian matrix L makes the implementation of (4.28) only requires the output difference between neighbouring subsystems, and therefore avoiding the requirement of full information* $\vec{e}_k$ *that is normally unavailable in practice (since only part of the subsystems have access to reference trajectory).*

Remark 4.4. *For the input updating law (4.26), it can be implemented in a centralised manner by directly solving a constrained quadratic programming (QP) problem. However, it requires huge computation load when implementing the algorithm centrally for large scale networked dynamical systems, which is undesirable in practice.*

4.3.2 Convergence Properties of Algorithm 4.1

For the proposed constrained ILC Algorithm 4.1, its convergence analysis will be divided into two cases.

4.3.2.1 Perfect Consensus Tracking is Achievable

In this case, the point $(0, \vec{u}^*)$ in the intersection of K_1 and K_2 does exist, i.e., $K_1 \cap K_2 \neq \emptyset$. Algorithm 4.1 has appealing convergence properties, as shown in the following theorem:

Theorem 4.2. *Given any initial input $\vec{u}_0 \in \Omega$ and associated tracking error $\vec{e}_0$, when perfect consensus tracking is achievable, Algorithm 4.1 guarantees the generated input sequence satisfies the constraint requirements, i.e.*

$$\vec{u}_{k+1} \in \Omega, \qquad \forall k \geq 0, \tag{4.29}$$

and the consensus tracking error norm converges monotonically to zero, i.e.

$$\|\vec{e}_{k+1}\|_{\vec{Q}} \leq \|\vec{e}_k\|_{\vec{Q}}, \qquad \lim_{k\to\infty} \vec{e}_k = 0. \tag{4.30}$$

Consequently, each subsystem achieves perfect consensus tracking and the corresponding input converges to the optimal solution as $k \to \infty$, i.e.

$$\lim_{k\to\infty} y_{i,k} = r, \qquad \lim_{k\to\infty} u_{i,k} = u_i^* \tag{4.31}$$

Proof. The property of (4.29) is a direct consequence of solving the problem (4.26). For the optimisation problem (4.26), when perfect consensus tracking is achievable, the non-optimal input choice $\vec{u}_{k+1} = \vec{u}_k$ yields a suboptimal solution

$$\|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}} + \|\vec{e}_{k+1}\|^2_{\vec{Q}} \leq \|\vec{e}_k\|^2_{\vec{Q}} \tag{4.32}$$

Note that $\|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}}$ is non-negative, hence

$$\|\vec{e}_{k+1}\|^2_{\vec{Q}} \leq \|\vec{e}_k\|^2_{\vec{Q}}, \quad k = 0, 1, \cdots, \infty \tag{4.33}$$

Note that Algorithm 4.1 iteratively finds the intersection of sets K_1 and K_2, and when $K_1 \cap K_2 \neq \emptyset$, the crosspoint is $(0, \vec{u}^*)$. Hence

$$\lim_{k\to\infty} \|\vec{e}_k\|^2_{\vec{Q}} = 0, \quad \lim_{k\to\infty} u_k = u^* \tag{4.34}$$

which follows that

$$\lim_{k\to\infty} \vec{e}_k^T \vec{Q} \vec{e}_k = 0 \tag{4.35}$$

Note that matrix $L+D$ is positive definite (as shown in Lemma 3.1) and hence $\vec{Q}$ is a positive definite matrix, which follows that $\vec{e}_k = 0$ as $k \to \infty$. That completes the proof. □

Moreover, when perfect consensus tracking is possible, Algorithm 4.1 has the property that the input difference between the optimal input solution and the k^{th} input is decreasing monotonically, as shown in the following theorem:

Theorem 4.3. *Given any initial input $\vec{u}_0 \in \Omega$ and associated tracking error $\vec{e}_0$, Algorithm 4.1 guarantees the generated input approach the optimal solution monotonically in norm, i.e.*

$$\|\vec{u}_{k+1} - \vec{u}^*\|_{\vec{R}} \leq \|\vec{u}_k - \vec{u}^*\|_{\vec{R}}, \qquad \forall k \geq 0 \tag{4.36}$$

Proof. When the perfect consensus tracking is possible, the intersection $x \in K_1 \cap K_2 = (0, \vec{u}^*)$ and then based on Theorem 4.1, we have

$$\|k_{2k} - x\|^2 \geq \|k_{2k+1} - x\|^2 \geq \|k_{2(k+1)} - x\|^2. \tag{4.37}$$

Note that k_{2k} is the point $(0, \vec{u}_k)$ and $k_{2(k+1)}$ is the point $(0, \vec{u}_{k+1})$, and it follows that

$$\|\vec{u}_{k+1} - \vec{u}^*\|_{\vec{R}} \leq \|\vec{u}_k - \vec{u}^*\|_{\vec{R}}, \qquad \forall k \geq 0. \tag{4.38}$$

That completes the proof. □

4.3.2.2 Perfect Consensus Tracking is Unachievable

In this case, the point in the intersection of K_1 and K_2 does not exist, i.e., $K_1 \cap K_2 = \emptyset$. Algorithm 4.1 guarantees the monotonic convergence of the consensus tracking error norm to a minimum consensus tracking error solution, as shown in the following theorem:

Theorem 4.4. *Given any initial input $\vec{u}_0 \in \Omega$ and associated tracking error $\vec{e}_0$, Algorithm 4.1 guarantees the generated input sequence satisfies the constraint requirements, i.e.*

$$\vec{u}_{k+1} \in \Omega, \qquad \forall k \geq 0 \tag{4.39}$$

and the convergence of the consensus tracking error norm is monotonic, i.e.

$$\|\vec{e}_{k+1}\|_{\vec{Q}} \leq \|\vec{e}_k\|_{\vec{Q}}, \qquad k \geq 0 \tag{4.40}$$

Furthermore, the global input $\vec{u}$ converges to the optimal solution $\vec{u}_s^*$ for the following optimisation problem

$$\vec{u}_s^* = \arg\min_{\vec{u} \in \Omega} \|\vec{r} - \mathbb{G}\vec{u} - \vec{d}\|^2_{\vec{Q}} \tag{4.41}$$

Proof. The proof of the monotonically convergence of the consensus tracking error norm is similar to the proof in Theorem 4.2, hence it is omitted here.

According to Theorem 1, when perfect consensus tracking is unachievable, Algorithm 4.1 converges to $\vec{u}_s^*$, where $k_1 = (\vec{e}, \vec{u}) \in K_1$ and $k_2 = (0, \vec{u}_s^*) \in K_2$ defining the minimum distance of two sets, and it is the solution for the following optimisation problem

$$(k_1, k_2) = \arg \min_{k_1 \in K_1, k_2 \in K_2} \|k_1 - k_2\|^2 \tag{4.42}$$

From the definition of K_1 and K_2, solving problem (4.42) is equivalent to solve the following optimisation problem

$$(\vec{u}, \vec{u}_s^*) = \arg \min_{\vec{u} \in \Omega, \vec{u}_0} \left\{ \|\vec{r} - \mathbb{G}\vec{u} - \vec{d}\|_{\vec{Q}}^2 + \|\vec{u} - \vec{u}_0\|_{\vec{R}}^2 \right\} \tag{4.43}$$

and hence, Algorithm 4.1 converges to $\vec{u}_s^*$, which defined as

$$\begin{aligned} \vec{u}_s^* &= \arg \min_{\vec{u} \in \Omega, \vec{u}_0} \left\{ \|\vec{r} - \mathbb{G}\vec{u}_0 - \vec{d}\|_{\vec{Q}}^2 + \|\vec{u} - \vec{u}_0\|_{\vec{R}}^2 \right\} \\ &= \arg \min_{\vec{u} \in \Omega} \left\{ \min_{\vec{u}_0} \|\vec{r} - \mathbb{G}\vec{u}_0 - \vec{d}\|_{\vec{Q}}^2 + \|\vec{u} - \vec{u}_0\|_{\vec{R}}^2 \right\} \end{aligned} \tag{4.44}$$

Note that $\vec{u}_0 = \vec{u}$ is the solution of the inner minimization, and hence

$$\vec{u}_s^* = \arg \min_{\vec{u} \in \Omega} \|\vec{r} - \mathbb{G}\vec{u} - \vec{d}\|_{\vec{Q}}^2 \tag{4.45}$$

Since matrices $\vec{Q}$ and $\mathbb{G}$ are invertible, the performance index is strictly convex. It should be noted that, the constraint is also convex, hence (4.45) has the unique solution. That completes the proof. □

Remark 4.5. *It should be noted that, the convergence speed of Algorithm 4.1 will be affected by the weighting matrices Q and R (in a similar way as that of standard norm optimal ILC algorithm). Taking scalar weighting as an example: a smaller weighting R indicates greater input change between two ILC trials, and hence leads to faster convergence speed. For more information, please refer to Chu and Owens (2010).*

Theorems 4.2, 4.3 and 4.4 show that Algorithm 4.1 guarantees the satisfaction of the input constraints and the monotonic convergence of the consensus tracking error norm to a minimum (possible) solution, which is appealing in practice. As mentioned in Remark 4.4, the centralised implementation of Algorithm 4.1 requires huge computational complexity for large scale networked dynamical systems, and hence we will propose distributed implementations for Algorithm 4.1 in the following sections.

4.3.3 Distributed Implementations of Algorithm 4.1

In this section, the idea of 'consensus' formulation in ADMM is used to develop two distributed implementations for Algorithm 4.1. In the following, we first review the general idea of ADMM.

4.3.3.1 The Alternating Direction Method of Multipliers

ADMM has superior convergence properties: the convergence of the objective, residual and dual variable is guaranteed for any $\rho > 0$, which is contrast to most of the distributed methodologies, e.g., dual decomposition (Boyd et al., 2011). For rigorous proof of the convergence properties, please refer to Boyd et al. (2011). To describe the general idea of ADMM, consider the following optimisation problem

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{p} J_i(x_i) + g(z) \\ \text{subject to} \quad & x_i - \widetilde{E}_i z = 0, \quad i = 1, \cdots, p \end{aligned} \tag{4.46}$$

where $J_i(x_i)$ is the local cost function, $g(z)$ is the global regularization function, $x_i \in \mathbb{R}^{p_i}$ is the local variable, $z \in \mathbb{R}^p$ denotes the global variable, and $\widetilde{E}_i$ is the corresponding matrix that maps the local variable to the global component.

To solve problem (4.46), ADMM will perform the following three steps iteratively

$$x_i^{q+1} = \arg\min L_{\rho i}(x_i, z^q, \gamma_i^q) \tag{4.47}$$

$$z^{q+1} = \arg\min L_{\rho}(x^{q+1}, z, \gamma^q) \tag{4.48}$$

$$\gamma_i^{q+1} = \gamma_i^q + \rho(x_i^{q+1} - \widetilde{E}_i z^{q+1}) \tag{4.49}$$

where $\gamma_i \in \mathbb{R}^{p_i}$ denotes the dual variable, q is the ADMM iteration index, ρ is the penalty parameter and the augmented Lagrangian is defined as

$$\begin{aligned} L_\rho(x, z, \gamma) &= \sum_{i=1}^{p} L_{\rho i}(x_i, z, \gamma_i) + g(z) \\ L_{\rho i}(x_i, z, \gamma_i) &= J_i(x_i) + \gamma_i^T (x_i - \widetilde{E}_i z) + \frac{\rho}{2}\|x_i - \widetilde{E}_i z\|^2 \end{aligned} \tag{4.50}$$

4.3.3.2 Distributed Implementation Algorithm 4.2

The most intuitive method is to perform the restriction on each local input updating step. The optimal solution of input law (4.26) at trial $k+1$ can be found by solving the following optimisation problem

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{p} J_{i,k+1}(\vec{u}_{i,k+1}) \\ \text{subject to} \quad & \vec{u}_{i,k+1} - \widetilde{E}_i z_{k+1} = 0, \quad i = 1, \cdots, p \end{aligned} \tag{4.51}$$

in which $\vec{u}_{i,k+1}$ is i^{th} subsystem's local input plan for itself and its neighbours. As an example, if $\mathcal{N}_i = \{l, m\}$

$$\vec{u}_{i,k+1} = \left[u_{i,k+1}{}^T \quad u_{l,k+1}{}^T \quad u_{m,k+1}{}^T \right]^T \tag{4.52}$$

and $J_{i,k+1}(\vec{u}_{i,k+1})$ is defined as

$$\begin{aligned} J_{i,k+1}(\vec{u}_{i,k+1}) = \| \sum_{j \in \mathcal{N}_i} & W_{ij} \left[(G_j u_{j,k+1} + d_j) - (G_i u_{i,k+1} + d_i) \right] \\ & + \boldsymbol{\mathcal{D}}_{ii}(r - G_i u_{i,k+1} - d_i) \|_Q^2 + \| u_{i,k+1} - u_{i,k} \|_R^2 \end{aligned} \tag{4.53}$$

with the domain of $J_{i,k+1}$ defined as

$$\mathbf{dom}\, J_{i,k+1} = \left\{ \vec{u}_{i,k+1} | \ \vec{u}_{i,k+1} \in \vec{\Omega}_i \right\} \tag{4.54}$$

where $\boldsymbol{\mathcal{D}}_{ii} = \mathcal{D}_{ii} \otimes I_N$, $\vec{\Omega}_i = \Omega_i \times \Omega_l \times \Omega_m$ (if $\mathcal{N}_i = \{l, m\}$), $\| u_{i,k+1} - u_{i,k} \|_R^2 = (u_{i,k+1} - u_{i,k})^T R (u_{i,k+1} - u_{i,k})$ and similarly with $\| \cdot \|_Q^2$.

Now, by defining the global variable z_{k+1} in (4.46) as

$$z_{k+1} = \left[z_{1,k+1}^T \quad z_{2,k+1}^T \quad \cdots \quad z_{p,k+1}^T \right]^T \tag{4.55}$$

where $z_{i,k+1}$ is the i^{th} component of the global value, and using ADMM three steps (4.47) – (4.49) to solve (4.51), the following distributed implementation method can be obtained.

Algorithm 4.2. *At trial $k+1$, the input sequence $\{\vec{u}_{i,k+1}^{q+1}\}_{q \geq 0}$ obtained from the following three steps*

$$\begin{aligned} \vec{u}_{i,k+1}^{q+1} = \arg \min_{\vec{u}_{i,k+1} \in \vec{\Omega}_i} & \Big[\gamma_{i,k+1}^{q}{}^T (\vec{u}_{i,k+1}^{q+1} - \widetilde{E}_i z_{k+1}^q) \\ & + \frac{\rho}{2} \| \vec{u}_{i,k+1}^{q+1} - \widetilde{E}_i z_{k+1}^q \|^2 + J_{i,k+1}(\vec{u}_{i,k+1}^{q+1}) \Big] \end{aligned} \tag{4.56}$$

$$z_{i,k+1}^{q+1} = \frac{1}{1 + |\mathcal{N}_i|} \sum_{o \in (\mathcal{N}_i \bigcup i)} (\vec{u}_{o,k+1}^{q+1})_i \tag{4.57}$$

$$\gamma_{i,k+1}^{q+1} = \gamma_{i,k+1}^{q} + \rho (\vec{u}_{i,k+1}^{q+1} - \widetilde{E}_i z_{k+1}^{q+1}) \tag{4.58}$$

provides a solution for problem (4.26), i.e.

$$\lim_{q\to\infty} z_{k+1}^{q+1} = \arg\min_{\vec{u}\in\Omega} \left\{ \|\vec{e}_{k+1}\|_{\widetilde{Q}}^2 + \|\vec{u}_{k+1} - \vec{u}_k\|_{\widetilde{R}}^2 \right\} \tag{4.59}$$

where q is the iteration index, and $(\vec{u}_{o,k+1}^{q+1})_i$ denotes all the relevant elements in the local input plan $\vec{u}_{o,k+1}^{q+1}$ related to the i^{th} agent.

Implementing the proposed Algorithm 4.1 using Algorithm 4.2 is straightforward, however, its computational burden can be heavy since each input updating step needs to solve a constrained optimisation problem (without analytic solution) to obtain the optimal solution.

4.3.3.3 Distributed Implementation Algorithm 4.3

An alternative method is to perform the regularization on the global variable. The solution of input law (4.26) can be obtained by solving the following optimisation problem

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{p} J_{i,k+1}(\vec{u}_{i,k+1}) + g(z_{k+1}) \\ &\text{subject to} \quad \vec{u}_{i,k+1} - \widetilde{E}_i z_{k+1} = 0, \quad i = 1, \cdots, p \end{aligned} \tag{4.60}$$

where the local cost function $J_{i,k+1}(\vec{u}_{i,k+1})$ is defined as

$$\begin{aligned} J_{i,k+1}(\vec{u}_{i,k+1}) = \| &\sum_{j\in\mathcal{N}_i} W_{ij}[(G_j u_{j,k+1} + d_j) - (G_i u_{i,k+1} + d_i)] \\ &+ \boldsymbol{\mathcal{D}}_{ii}(r - G_i u_{i,k+1} - d_i)\|_Q^2 + \|u_{i,k+1} - u_{i,k}\|_R^2 \end{aligned} \tag{4.61}$$

with the domain of $J_{i,k+1}$ defined as

$$\textbf{dom}\, J_{i,k+1} = \left\{ \vec{u}_{i,k+1} |\ \vec{u}_{i,k+1} \in \mathbb{R}^{p_i N} \right\} \tag{4.62}$$

and $g(z_{k+1})$ is the indicator function of set Ω:

$$g(z_{k+1}) = \begin{cases} 0 & z_{k+1} \in \Omega \\ +\infty & z_{k+1} \notin \Omega \end{cases} \tag{4.63}$$

Then, using the idea of general form in ADMM, yields the following distributed implementation method:

Algorithm 4.3. *At trial $k+1$, the input sequence $\{\vec{u}_{i,k+1}^{q+1}\}_{q\geq 0}$ obtained from the following three steps*

$$\vec{u}_{i,k+1}^{q+1} = \text{argmin} \left[J_{i,k+1}(\vec{u}_{i,k+1}^{q+1}) + \frac{\rho}{2}\|\vec{u}_{i,k+1}^{q+1} - \widetilde{E}_i z_{k+1}^q\|^2 + \gamma_{i,k+1}^{q}{}^T (\vec{u}_{i,k+1}^{q+1} - \widetilde{E}_i z_{k+1}^q) \right] \tag{4.64}$$

Algorithm 4.4. *Distributed ILC Algorithm for Constraint Handling Problem*

```
Input: State space matrix $A_i$, $B_i$, $C_i$, reference trajectory $r$, reference-accessibility
    matrix $D$, Laplacian matrix $L$, maximum-trial $k_{max}$, maximum-iteration $q_{max}$,
    penalty parameter $\rho$, weighting $Q$ and weighting $R$
Output: Each subsystem's optimal input $u_{i,k_{max}}$
1:  Initialization: Set the ILC trial number $k = 0$
2:  For: $k = 0$ to $k_{max}$
3:    For: $q = 0$ to $q_{max}$
4:      For: $i = 1$ to $p$
5:        Receive data from neighbouring subsystems
6:        Perform $\vec{u}_{i,k+1}^{q+1}$ minimization (4.56) (or (4.64))
7:        Perform $z_{i,k+1}^{q+1}$ minimization (4.57) (or (4.65))
8:        Perform $\gamma_{i,k+1}^{q+1}$ minimization (4.58) (or (4.66))
9:        Send data to neighbouring subsystems
10:     End for
11:   End for
12: Transform local input plan $\vec{u}_{i,k+1}^{q_{max}}$ into $u_{i,k+1}$
13: End for
14: Return: Each subsystem's optimal input $u_{i,k_{max}}$
```

$$z_{i,k+1}^{q+1} = \Pi_{\Omega_i}\left[\frac{1}{1+|\mathcal{N}_i|}\sum_{o\in(\mathcal{N}_i\bigcup i)}[(\vec{u}_{o,k+1}^{q+1})_i+(\gamma_{o,k+1}^{q+1})_i]\right] \tag{4.65}$$

$$\gamma_{i,k+1}^{q+1} = \gamma_{i,k+1}^{q} + \rho(\vec{u}_{i,k+1}^{q+1} - \widetilde{E}_i z_{k+1}^{q+1}) \tag{4.66}$$

provides a solution for problem (4.26), i.e.

$$\lim_{q\to\infty} z_{k+1}^{q+1} = \arg\min_{\vec{u}\in\Omega}\left\{\|\vec{e}_{k+1}\|_{\vec{Q}}^2 + \|\vec{u}_{k+1} - \vec{u}_k\|_{\vec{R}}^2\right\} \tag{4.67}$$

where Π_{Ω_i} denotes the projection onto the input constraint set Ω_i and $(\gamma_{o,k+1}^{q+1})_i$ represents all the corresponding components of local dual variable $\gamma_{o,k+1}^{q+1}$ related to the i^{th} agent.

4.3.4 Distributed ILC Algorithm 4.4

Using distributed implementation Algorithm 4.2 (4.3) to implement Algorithm 4.1 distributively, yields the distributed ILC algorithm for constrained consensus tracking problem, as shown in the Algorithm 4.4. Algorithm 4.4 contains k_{max} ILC trials and each trial requires q_{max} ADMM iterations to approach the optimal input solution. In each ADMM iteration, each subsystem receives information from its neighbours in Step 5; in Step $6-8$, it uses the updating law (4.56) – (4.58) (or (4.64) – (4.66)) to update the local input plan $\vec{u}_{i,k+1}^{q+1}$, global value component $z_{i,k+1}^{q+1}$ and local dual value $\gamma_{i,k+1}^{q+1}$; in Step 9, each subsystem sends data to its neighbours and starts the next ADMM iteration (until reaching the maximum iteration number).

Remark 4.6. *Note that, for Algorithm 4.4, the convergence behaviour of ADMM is*

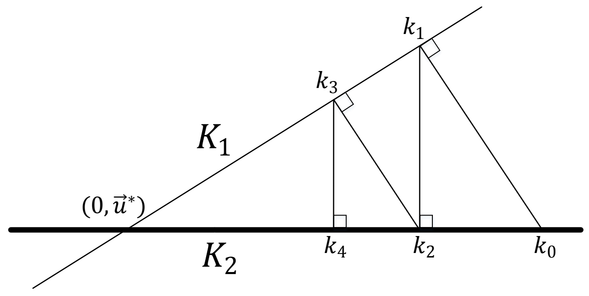

Figure 4.3: Algorithm 4.5 – perfect consensus tracking is possible

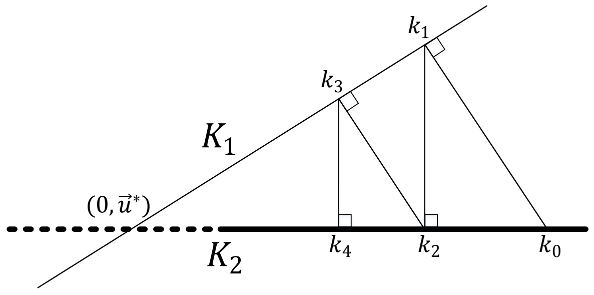

Figure 4.4: Algorithm 4.5 – perfect consensus tracking is impossible

affected by the maximum ADMM iteration number q_{max}. In theory, infinite iteration number is required for ADMM to approximate the centralised solution, however, a small number of ADMM iteration is usually sufficient to approach the centralised result in practice. This phenomenon will be verified in later simulation.

4.4 Constrained Consensus ILC Algorithm 4.5

In contrast to Algorithm 4.1 proposed in Section 4.3, the following algorithm will incorporate the input constraints with the zero error tracking. In this cases, the set $K_1 = S_1$ and $K_2 = S_2 \cap S_3$, i.e.

- $K_1 = \left\{(\vec{e},\vec{u}) \in H : \vec{e} = \vec{r} - \mathbb{G}\vec{u} - \vec{d}\right\}$;
- $K_2 = \{(\vec{e},\vec{u}) \in H : \vec{e} = 0, \vec{u} \in \Omega\}$.

The cases for $K_1 \cap K_2 \neq \emptyset$ and $K_1 \cap K_2 = \emptyset$ are illustrated in Figures 4.3 and 4.4, respectively.

4.4.1 Algorithm Description

Algorithm 4.5. *Given any initial input $\vec{u}_0$ satisfying the input constraint set Ω, the input sequence $\{\vec{u}_k\}_{k\geq 0}$ defined by the solution of the following unconstrained optimisation*

problem

$$\hat{u}_{k+1} = \arg\min_{\vec{u}} \left\{ \|\vec{e}_{k+1}\|^2_{\vec{Q}} + \|\vec{u} - \vec{u}_k\|^2_{\vec{R}} \right\} \tag{4.68}$$

followed by the input projection

$$\vec{u}_{k+1} = \arg\min_{\vec{u}\in\Omega} \|\vec{u} - \hat{u}_{k+1}\| \in \Omega \tag{4.69}$$

iteratively solves the constrained ILC problem in the input constraints set Ω*, i.e.*

$$\begin{aligned} u_{i,k} &\in \Omega_i, && \forall k \geq 0 \\ \lim_{k\to\infty} y_{i,k} &= r, && i = 1, 2, \cdots, p \end{aligned} \tag{4.70}$$

Remark 4.7. *For the first step (4.68) in Algorithm 4.5, it can be implemented in a centralised manner by directly finding the stationary point of the cost function and the input law is shown as*

$$\hat{u}_{k+1} = \vec{u}_k + (\mathbb{G}^T\vec{Q}\mathbb{G} + \vec{R})^{-1}\mathbb{G}^T\vec{Q}\vec{e}_k \tag{4.71}$$

However, it requires huge computation load when implementing the algorithm centrally for large scale networked dynamical systems, which is undesirable in practice.

Remark 4.8. *For the second step (4.69) in Algorithm 4.5, the solution seems not easy to be obtained. However, the individual input constraint set* Ω_i *is normally a point-wise constraint that can be calculated easily. As an example, if* $\Omega_i = \{u_i \in \mathbb{R}^N : 0 \leq u_i(t)\}$*, then the input solution is simply obtained as follows*

$$u_{i,k+1}(t) = \begin{cases} \hat{u}_{i,k+1}(t) & if \;\; \hat{u}_{i,k+1}(t) \geq 0 \\ 0 & if \;\; \hat{u}_{i,k+1}(t) < 0 \end{cases} \tag{4.72}$$

for $t \in [0, N-1]$*.*

4.4.2 Convergence Properties of Algorithm 4.5

For the proposed constrained ILC Algorithm 4.5, its convergence analysis will also be divided into two cases.

4.4.2.1 Perfect Consensus Tracking is Achievable

In this situation, the intersection of sets K_1 and K_2 is not empty (i.e., $K_1 \cap K_2 \neq \emptyset$) and Algorithm 4.5 has appealing convergence properties shown as follow:

Theorem 4.5. *Given any initial input $\vec{u}_0 \in \Omega$ and associated tracking error $\vec{e}_0$, Algorithm 4.5 guarantees the generated input sequence satisfies the constraint requirements, i.e.*

$$\vec{u}_{k+1} \in \Omega, \qquad \forall k \geq 0 \tag{4.73}$$

and the consensus error norm converges to zero, i.e.

$$\lim_{k\to\infty} \vec{e}_k = 0. \tag{4.74}$$

Consequently, each subsystem achieves the perfect consensus tracking and the corresponding input converges to the optimal solution as $k \to \infty$, i.e.

$$\lim_{k\to\infty} y_{i,k} = r, \qquad \lim_{k\to\infty} u_{i,k} = u_i^*. \tag{4.75}$$

Moreover, the monotonic convergence is hold for the following performance index:

$$J_k = \|E\vec{e}_k\|^2_{\vec{Q}} + \|F\vec{e}_k\|^2_{\vec{R}} \tag{4.76}$$

in which

$$E = I_{pN} - \mathbb{G}(\mathbb{G}^T\vec{Q}\mathbb{G} + \vec{R})^{-1}\mathbb{G}^T\vec{Q}, \tag{4.77}$$

$$F = (\mathbb{G}^T\vec{Q}\mathbb{G} + \vec{R})^{-1}\mathbb{G}^T\vec{Q} \tag{4.78}$$

Proof. When perfect consensus tracking is achievable (i.e., $S_1 \cap (S_2 \cap S_3) \neq \emptyset$), the successive projection framework iteratively finds the intersection $(0, \vec{u}^*)$, and hence Equations (4.74) and (4.75) are obtained.

According to Theorem 4.1, the distance between $\{k_0, k_1, k_2, k_3, \cdots\}$ is decreasing, i.e.,

$$\begin{aligned} \|k_{2k} - k_{2k+1}\| &\geq \|k_{2k+1} - k_{2(k+1)}\| \\ &\geq \|k_{2(k+1)} - k_{2(k+1)+1}\|. \end{aligned} \tag{4.79}$$

Note that, the norm in the left side is the projection of k_{2k} onto the set K_1, i.e.

$$\|k_{2k} - k_{2k+1}\| = \arg\min_{\vec{u}} \left\{ \|\vec{e}_{k+1}\|^2_{\vec{Q}} + \|\vec{u} - \vec{u}_k\|^2_{\vec{R}} \right\} \tag{4.80}$$

and it has an analytic solution, which shown as

$$\hat{u}_{k+1} = \vec{u}_k + (\mathbb{G}^T\vec{Q}\mathbb{G} + \vec{R})^{-1}\mathbb{G}^T\vec{Q}\vec{e}_k \tag{4.81}$$

Then, using the solution (4.81), Equation (4.80) is further rewritten as

$$\|k_{2k} - k_{2k+1}\| = \|E\vec{e}_k\|^2_{\vec{Q}} + \|F\vec{e}_k\|^2_{\vec{R}} \tag{4.82}$$

where matrix E and F are defined in (4.77) and (4.78). Note that, Equation (4.82) is the performance index J_k. Following the similar idea, the bottom term of (4.79) is J_{k+1} and then we have

$$J_{k+1} \leq J_k. \tag{4.83}$$

That completes the proof. □

In the above theorem, it shows that Algorithm 4.5 cannot guarantee the monotonic convergence of the consensus tracking error norm $\|\vec{e}_k\|_{\vec{Q}}$ even when perfect consensus tracking is possible, however, Algorithm 4.5 has the property that the input difference between the optimal input solution and the k^{th} trial's input is decreasing monotonically, as shown in the following theorem:

Theorem 4.6. *Given any initial input $\vec{u}_0 \in \Omega$ and associated tracking error $\vec{e}_0$, Algorithm 4.5 guarantees the generated input approach the optimal solution monotonically in norm, i.e.*

$$\|\vec{u}_{k+1} - \vec{u}^*\|_{\vec{R}} \leq \|\vec{u}_k - \vec{u}^*\|_{\vec{R}}, \qquad \forall k \geq 0 \tag{4.84}$$

Proof. The proof follows the similar procedure as the proof of monotonic convergence on control input in Theorem 4.3. Hence, it is omitted here for brevity. □

4.4.2.2 Perfect Consensus Tracking is Unachievable

In this situation, the intersection of sets K_1 and K_2 is empty (i.e., $K_1 \cap K_2 = \emptyset$) and Algorithm 4.5 has appealing convergence properties shown as follow.

Theorem 4.7. *Given any initial input $\vec{u}_0 \in \Omega$ and associated tracking error $\vec{e}_0$, Algorithm 4.5 guarantees the generated input sequence satisfies the constraint requirements, i.e.*

$$\vec{u}_{k+1} \in \Omega, \qquad \forall k \geq 0 \tag{4.85}$$

and the monotonic convergence is hold for the following performance index

$$J_k = \|E\vec{e}_k\|^2_{\vec{Q}} + \|F\vec{e}_k\|^2_{\vec{R}} \tag{4.86}$$

in which

$$E = I_{pN} - \mathbb{G}(\mathbb{G}^T\vec{Q}\mathbb{G} + \vec{R})^{-1}\mathbb{G}^T\vec{Q}, \tag{4.87}$$

$$F = (\mathbb{G}^T\vec{Q}\mathbb{G} + \vec{R})^{-1}\mathbb{G}^T\vec{Q} \tag{4.88}$$

Furthermore, the global input $\vec{u}$ converges to the optimal solution $\vec{u}^*_s$ for the following optimisation problem

$$\vec{u}^*_s = \arg\min_{\vec{u}\in\Omega} \left\{ \|E\vec{e}\|^2_{\vec{Q}} + \|F\vec{e}\|^2_{\vec{R}} \right\} \tag{4.89}$$

Proof. The proof of the monotonically convergence of the defined performance index J_k is similar to the proof in Theorem 4.5, hence it is omitted here.

According to Theorem 4.1, when perfect consensus tracking is unachievable, Algorithm 4.5 converges to $\vec{u}_s^*$, where $k_1 = (\vec{e}, \vec{u}) \in K_1$ and $k_2 = (0, \vec{u}_s^*) \in K_2$ defining the minimum distance of two sets, and it is the solution for the following optimisation problem

$$(k_1, k_2) = \arg \min_{k_1 \in K_1, k_2 \in K_2} \|k_1 - k_2\|^2 \tag{4.90}$$

From the definition of K_1 and K_2, solving problem (4.90) is equivalent to solve the following optimisation problem

$$(\vec{u}, \vec{u}_s^*) = \arg \min_{\vec{u} \in \Omega, \vec{u}_0} \left\{ \|\vec{r} - \mathbb{G}\vec{u}_0 - \vec{d}\|_{\vec{Q}}^2 + \|\vec{u}_0 - \vec{u}\|_{\vec{R}}^2 \right\} \tag{4.91}$$

and hence, Algorithm 4.5 converges to $\vec{u}_s^*$, which defined as

$$\begin{aligned} \vec{u}_s^* &= \arg \min_{\vec{u} \in \Omega, \vec{u}_0} \left\{ \|\vec{r} - \mathbb{G}\vec{u}_0 - \vec{d}\|_{\vec{Q}}^2 + \|\vec{u}_0 - \vec{u}\|_{\vec{R}}^2 \right\} \\ &= \arg \min_{\vec{u} \in \Omega} \left\{ \min_{\vec{u}_0} \|\vec{r} - \mathbb{G}\vec{u}_0 - \vec{d}\|_{\vec{Q}}^2 + \|\vec{u}_0 - \vec{u}\|_{\vec{R}}^2 \right\} \end{aligned} \tag{4.92}$$

Note that, the inner minimization has an analytic solution, which shown as

$$\vec{u}_0 = \vec{u} + (\mathbb{G}^T \vec{Q} \mathbb{G} + \vec{R})^{-1} \mathbb{G}^T \vec{Q} \vec{e} \tag{4.93}$$

Then, substitute (4.93) into (4.92), we have

$$\vec{u}_s^* = \arg \min_{\vec{u} \in \Omega} \left\{ \|E\vec{e}\|_{\vec{Q}}^2 + \|F\vec{e}\|_{\vec{R}}^2 \right\} \tag{4.94}$$

where matrix E and F are defined in (4.87) and (4.88).

Since matrices $\vec{Q}$, $\vec{R}$, E and F are invertible, the performance index is strictly convex. It should be noted that, the constraint is also convex, hence (4.92) has the unique solution. That completes the proof. $\square$

Remark 4.9. *Note that, when perfect consensus tracking is unachievable, smaller weighting R will still result in faster convergence speed, however, the convergent error value will change (in contrast to the case in Algorithm 4.1). As mentioned in Theorem 4.7, the input of Algorithm 4.5 converges as follow:*

$$\vec{u}_s^* = \arg \min_{\vec{u} \in \Omega} \left\{ \|E\vec{e}\|_{\vec{Q}}^2 + \|F\vec{e}\|_{\vec{R}}^2 \right\} \tag{4.95}$$

and when $\vec{R} \to \infty$, the first term is $\|\vec{e}\|_{\vec{Q}}^2$ and the second term is 0. Hence, Equation

(4.95) becomes

$$\vec{u}_s^* = \arg\min_{\vec{u}\in\Omega} \|\vec{e}\|^2_{\vec{Q}}. \tag{4.96}$$

This is the best result can be achieved when the perfect consensus tracking is unachievable, but on the other hand, the convergence speed of consensus error is extremely slow and it should take infinity ILC trials to achieve the optimal solution, which is undesired in practice.

Consider another extreme situation $\vec{R} = 0$, the first term becomes 0 and the second term is $\|\mathbb{G}^{-1}\vec{e}\|^2_{\vec{R}}$. Note that $\|\mathbb{G}^{-1}\vec{e}\|^2_{\vec{R}} = \|\vec{u} - \vec{u}^*\|^2_{\vec{R}}$ (where $\vec{u}^*$ is the optimal input when perfect consensus tracking is possible). Hence, Equation (4.95) becomes

$$\vec{u}_s^* = \arg\min_{\vec{u}\in\Omega} \|\vec{u} - \vec{u}^*\|^2_{\vec{R}}. \tag{4.97}$$

This is the projection of the optimal input $\|\vec{u}^*\|$ onto the constraint set Ω and clearly, the value of (4.97) is larger than (4.96). However, the convergence speed is extremely fast in this situation.

4.4.3 Distributed Implementations of (4.68) in Algorithm 4.5

For the first step (4.68) in Algorithm 4.5, its performance index can be fully separated into p local cost functions, with the local cost function $J_{i,k+1}(\hat{u}_{i,k+1})$ of i^{th} subsystem defined as

$$\begin{aligned} J_{i,k+1}(\hat{u}_{i,k+1}) = \| \sum_{j\in\mathscr{N}_i} & W_{ij}\left[(G_i u_{i,k+1} + d_i) - (G_j u_{j,k+1} + d_j)\right] \\ & + \boldsymbol{\mathcal{D}}_{ii}(r - G_i u_{i,k+1} - d_i)\|^2_Q + \|u_{i,k+1} - u_{i,k}\|^2_R \end{aligned} \tag{4.98}$$

in which $\hat{u}_{i,k+1}$ represents the local auxiliary input variable for subsystem i and its neighbours $j \in \mathscr{N}_i$. As an example, if $\mathscr{N}_i = \{l, m\}$, $\hat{u}_{i,k+1}$ is denoted as

$$\hat{u}_{i,k+1} = \begin{bmatrix} u^T_{i,k+1} & u^T_{l,k+1} & u^T_{m,k+1} \end{bmatrix}^T \tag{4.99}$$

Using the idea of ADMM, we can have the following implementation algorithm:

Algorithm 4.6.0. *At ILC trial $k+1$, the input sequence $\{\hat{u}^{q+1}_{i,k+1}\}_{q\geq 0}$ obtained from the following three steps*

$$\hat{u}^{q+1}_{i,k+1} = \arg\min \left[J_{i,k+1}(\hat{u}^{q+1}_{i,k+1}) + \frac{\rho}{2}\|\hat{u}^{q+1}_{i,k+1} - \widetilde{E}_i z^q_{k+1}\|^2 + {\gamma^q_{i,k+1}}^T(\hat{u}^{q+1}_{i,k+1} - \widetilde{E}_i z^q_{k+1})\right] \tag{4.100}$$

$$z^{q+1}_{i,k+1} = \frac{1}{1+|\mathscr{N}_i|} \sum_{o\in(\mathscr{N}_i \bigcup i)} (\hat{u}^{q+1}_{o,k+1})_i \tag{4.101}$$

$$\gamma^{q+1}_{i,k+1} = \gamma^q_{i,k+1} + \rho(\hat{u}^{q+1}_{i,k+1} - \widetilde{E}_i z^{q+1}_{k+1}) \tag{4.102}$$

provides a distributed implementation for problem (4.68), i.e.

$$\lim_{q\to\infty} z_{k+1}^{q} = \arg\min\left\{\|\vec{e}_{k+1}\|_{\vec{Q}}^{2} + \|\vec{u}-\vec{u}_k\|_{\vec{R}}^{2}\right\} \tag{4.103}$$

where $(\hat{u}_{o,k+1})_i$ *denotes the element in* $\hat{u}_{o,k+1}$ *that related to* $z_{i,k+1}$.

For the distributed input updating law (4.100), it is actually the form of unconstrained Linear Quadratic Tracking (LQT), and hence it can be implemented in a matrix form, or in a feedback plus feedforward form. We first consider the matrix form, as shown in the following proposition:

Proposition 4.1. *For the input law (4.100), it can be implemented in a matrix form as follows*

$$\begin{aligned}\hat{u}_{i,k+1}^{q+1} = &\left[\vec{G}_i^{T}(\boldsymbol{L}_i+\boldsymbol{D}_i)^{T}\boldsymbol{Q}_i(\boldsymbol{L}_i+\boldsymbol{D}_i)\vec{G}_i+\frac{\rho}{2}I+\boldsymbol{S}_i^{T}\boldsymbol{R}_i\boldsymbol{S}_i\right]^{-1}\\ &\times\left[\vec{G}_i^{T}(\boldsymbol{L}_i+\boldsymbol{D}_i)^{T}\boldsymbol{Q}_i(\boldsymbol{L}_i+\boldsymbol{D}_i)(\vec{r}_i-\vec{d}_i)+\boldsymbol{S}_i^{T}\boldsymbol{R}_i\boldsymbol{S}_i\vec{u}_{i,k}+\frac{\rho}{2}\widetilde{E}_i z_{k+1}^{q}-\frac{1}{2}\gamma_{i,k+1}^{q}\right]\end{aligned} \tag{4.104}$$

in which $\boldsymbol{Q}_i = Q\otimes I_{p_i}$, $\boldsymbol{R}_i = R\otimes I_{p_i}$ *and* $\boldsymbol{L}_i, \boldsymbol{D}_i, \boldsymbol{S}_i$, $\vec{G}_i$ *are local Laplacian matrix, local reference-accessibility matrix, local selected matrix and local system matrix. As an example, if* $\mathcal{N}_i = \{l, m\}$, *then*

$$\boldsymbol{L}_i = \begin{bmatrix} \sum_{j\in\mathcal{N}_i} W_{ij} & -W_{il} & -W_{im} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \otimes I_N$$

$$\boldsymbol{D}_i = \begin{bmatrix} d_{ii} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \otimes I_N \qquad \boldsymbol{S}_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \otimes I_N$$

$$\vec{G}_i = \operatorname{diag}(G_i, G_l, G_m).$$

Moreover, $\vec{r}_i$, $\vec{d}_i$, $\vec{u}_{i,k}$ *are local reference trajectory, local initial state respond and local input vector, which defined as*

$$\begin{aligned}\vec{r}_i &= \begin{bmatrix} r^T & r^T & r^T \end{bmatrix}^T\\ \vec{d}_i &= \begin{bmatrix} d_i^T & d_l^T & d_m^T \end{bmatrix}^T\\ \vec{u}_{i,k} &= \begin{bmatrix} u_{i,k}^T & u_{l,k}^T & u_{m,k}^T \end{bmatrix}^T\end{aligned}$$

Proof. All the above local parameters are obtained by extracting the corresponding elements in the global matrices/vectors defined in (4.68). Finding the stationary point of Equation (4.100), we have

$$\begin{aligned}&\left[\vec{G}_i^T(\boldsymbol{L}_i+\boldsymbol{D}_i)^T\boldsymbol{Q}_i(\boldsymbol{L}_i+\boldsymbol{D}_i)\vec{G}_i+\frac{\rho}{2}I+\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i\right]\hat{u}_{i,k+1}^{q+1} = \\ &\quad \vec{G}_i^T(\boldsymbol{L}_i+\boldsymbol{D}_i)^T\boldsymbol{Q}_i(\boldsymbol{L}_i+\boldsymbol{D}_i)(\vec{r}_i-\vec{d}_i)+\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i\vec{u}_{i,k}+\frac{\rho}{2}\widetilde{E}_i z_{k+1}^q-\frac{1}{2}\gamma_{i,k+1}^q.\end{aligned} \tag{4.105}$$

Note that the term $\vec{G}_i^T(\boldsymbol{L}_i+\boldsymbol{D}_i)^T\boldsymbol{Q}_i(\boldsymbol{L}_i+\boldsymbol{D}_i)\vec{G}_i+\frac{\rho}{2}I+\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i$ is invertible, it yields the input updating law (4.104). □

Note that, the matrix form implementation (4.104) is easy to follow, but on the other hand, it involve the calculation of the system matrix inverse and hence produces large computational load. Next, we will introduce a feedback plus feedforward implementation.

Proposition 4.2. *For the input law (4.100), it can be calculated using a feedback plus feedforward structure*

$$\hat{u}_{i,k+1}^{q+1}(t) = \eta_{i,k+1}^q(t)+\boldsymbol{\Psi}_i^{-1}(t)\vec{B}_i^T\psi_{i,k+1}^{q+1}(t) \tag{4.106}$$

with costate $\psi_{i,k+1}^{q+1}(t)$ is defined as

$$\psi_{i,k+1}^{q+1}(t) = \left[-K_i(t)\{I_{p_i}+\vec{B}_i\boldsymbol{\Psi}_i^{-1}(t)\vec{B}_i^TK_i(t)\}^{-1}\vec{A}_i\vec{\chi}_{i,k+1}^{q+1}(t)\right]+\xi_{i,k+1}^{q+1}(t) \tag{4.107}$$

where $\vec{\chi}_{i,k+1}^{q+1}(t)$ is denotes as

$$\begin{aligned}\vec{\chi}_{i,k+1}^{q+1}(0)&=0\\ \vec{\chi}_{i,k+1}^{q+1}(t+1)&=\vec{A}_i\vec{\chi}_{i,k+1}^{q+1}(t)+\vec{B}_i\left[\hat{u}_{i,k+1}^{q+1}(t)-\eta_{i,k+1}^q(t)\right]\end{aligned} \tag{4.108}$$

and $\vec{A}_i$, $\vec{B}_i$, $\vec{C}_i$ are local state-space matrices of subsystem i and $K_i(\cdot)$ represents the local discrete Riccati Equation on the time interval $t\in[0,N-1]$ with terminal condition $K_i(N)=0$:

$$\begin{aligned}K_i(t)=&-\vec{A}_i^TK_i(t+1)\vec{B}_i\{\vec{B}_i^TK_i(t+1)\vec{B}_i+\boldsymbol{\Psi}_i(t+1)\}^{-1}\\ &\times B_i^TK_i(t+1)\vec{A}_i+\vec{A}_i^TK_i(t+1)\vec{A}_i+\vec{C}_i^T\boldsymbol{\Xi}_i(t+1)\vec{C}_i\end{aligned} \tag{4.109}$$

and $\xi_{i,k+1}^{q+1}(t)$ denotes the feedforward term denotes as (with conditions $\xi_{i,k+1}^{q+1}(N)=0$)

$$\begin{aligned}\xi_{i,k+1}^{q+1}(t)=&\left\{I_{p_i}+K_i(t)\vec{B}_i\boldsymbol{\zeta}_i^{-1}(t)\vec{B}_i^T\right\}^{-1}\\ &\times\left\{\vec{A}_i^T\xi_{i,k+1}^{q+1}(t+1)+\vec{C}_i^T\boldsymbol{\Xi}_i(t+1)\theta_{i,k+1}^q(t+1)\right\}\end{aligned} \tag{4.110}$$

$\eta_{i,k+1}^{q}$ *is defined as*

$$\eta_{i,k+1}^{q} = \boldsymbol{\zeta}_i^{-2}\left(\frac{\rho}{2}\widetilde{E}_i z_{k+1}^{q} - \frac{1}{2}\gamma_{i,k+1}^{q} + \boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i\vec{u}_{i,k}\right), \tag{4.111}$$

$\boldsymbol{\zeta}_i$, $\boldsymbol{\varsigma}_i$ *are defined as*

$$\boldsymbol{\zeta}_i = (\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i + \frac{\rho}{2}I_{p_iN})^{\frac{1}{2}}, \tag{4.112}$$

$$\boldsymbol{\varsigma}_i = \boldsymbol{Q}_i{}^{\frac{1}{2}}(\boldsymbol{L}_i + \boldsymbol{D}_i), \tag{4.113}$$

$\boldsymbol{\Psi}_i$, $\boldsymbol{\Xi}_i$ *are defined as*

$$\boldsymbol{\Psi}_i = \boldsymbol{\zeta}_i^T\boldsymbol{\zeta}_i, \tag{4.114}$$

$$\boldsymbol{\Xi}_i = \boldsymbol{\varsigma}_i^T\boldsymbol{\varsigma}_i, \tag{4.115}$$

$\theta_{i,k+1}^{q}$ *is defined as*

$$\theta_{i,k+1}^{q} = \vec{r}_i - \vec{G}_i\eta_{i,k+1}^{q} - \vec{d}_i \tag{4.116}$$

and $\cdot(t)$ *denotes the* t^{th} *element in the corresponding vector or matrix.*

Proof. At iteration $q+1$, the input of the ADMM algorithm is generated by solving the following optimization problem

$$\begin{aligned}\hat{u}_{i,k+1}^{q+1} = \arg\min\big\{&\|(\boldsymbol{D}_i+\boldsymbol{L}_i)(\vec{r}_i - \vec{G}_i\hat{u}_{i,k+1}^{q+1} - \vec{d}_i))\|_{\boldsymbol{Q}_i}^2 + \|\boldsymbol{S}_i(\hat{u}_{i,k+1}^{q+1} - \vec{u}_{i,k}))\|_{\boldsymbol{R}_i}^2 \\ &+ \frac{\rho}{2}\|\hat{u}_{i,k+1}^{q+1} - \widetilde{E}_i z_{k+1}^{q}\|^2 + \gamma_{i,k+1}^{q}{}^T(\hat{u}_{i,k+1}^{q+1} - \widetilde{E}_i z_{k+1}^{q})\big\}\end{aligned} \tag{4.117}$$

For the above optimization problem, we define the last 3 terms as $\varpi_{i,k+1}^{q+1}$, which could be written as:

$$\begin{aligned}\varpi_{i,k+1}^{q+1} =& (\hat{u}_{i,k+1}^{q+1})^T(\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i + \frac{\rho}{2}I_{p_iN})\hat{u}_{i,k+1}^{q+1} \\ &- 2(\hat{u}_{i,k+1}^{q+1})^T\Big[\frac{\rho}{2}\widetilde{E}_i z_{k+1}^{q} - \frac{1}{2}\gamma_{i,k+1}^{q} + \boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i\vec{u}_{i,k}\Big]\end{aligned} \tag{4.118}$$

that is

$$\varpi_{i,k+1}^{q+1} = \|\boldsymbol{\zeta}_i(\hat{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^{q})\|^2 \tag{4.119}$$

where $\boldsymbol{\zeta}_i = (\boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i + \frac{\rho}{2}I_{p_iN})^{\frac{1}{2}}$, $\eta_{i,k+1}^{q} = \boldsymbol{\zeta}_i^{-2}(\frac{\rho}{2}\widetilde{E}_i z_{k+1}^{q} - \frac{1}{2}\gamma_{i,k+1}^{q} + \boldsymbol{S}_i{}^T\boldsymbol{R}_i\boldsymbol{S}_i\vec{u}_{i,k})$. It follows that

$$\varpi_{i,k+1}^{q+1} = \|\hat{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^{q}\|_{\boldsymbol{\Psi}_i = \boldsymbol{\zeta}_i{}^T\boldsymbol{\zeta}_i}^2 \tag{4.120}$$

Note that, the first term in (4.117) can be written into the following form:

$$\varrho_{i,k+1}^{q+1} = \|\boldsymbol{\varsigma}_i(\theta_{i,k+1}^{q} - \vec{G}_i(\hat{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^{q}))\|^2 \tag{4.121}$$

Algorithm 4.6. *Distributed ILC Algorithm for Constraint Handling Problem*

```
Input: State space matrices $A_i$, $B_i$, $C_i$, desired reference $r$, reference-accessibility
    matrix $D$, Laplacian matrix $L$, maximum-trial $k_{max}$, maximum-iteration $q_{max}$,
    penalty parameter $\rho$, weighting $Q$ and weighting $R$
Output: Each subsystem's optimal input $u_{i,k_{max}}$
1:  Initialization: Set the ILC trial number $k = 0$
2:  For: $k = 0$ to $k_{max}$
3:    For: $q = 0$ to $q_{max}$
4:      For: $i = 1$ to $p$
5:        Receive data from neighbouring subsystems
6:        Perform $\hat{u}_{i,k+1}^{q+1}$ minimization (4.104) (or (4.106))
7:        Perform $z_{i,k+1}^{q+1}$ minimization (4.101)
8:        Perform $\gamma_{i,k+1}^{q+1}$ minimization (4.102)
9:        Send data to neighbouring subsystems
10:     End for
11:   End for
12: Transform local input plan $\hat{u}_{i,k+1}^{q_{max}}$ into $\hat{u}_{i,k+1}$
13: Perform the input projection step (4.69)
14: End for
15: Return: Each subsystem's optimal input $u_{i,k_{max}}$
```

where

$$\begin{aligned} \varsigma_i &= \boldsymbol{Q}_i^{\frac{1}{2}}(\boldsymbol{L}_i + \boldsymbol{D}_i) \\ \theta_{i,k+1}^q &= \vec{r}_i - \vec{G}_i \eta_{i,k+1}^q - \vec{d}_i \end{aligned} \tag{4.122}$$

then, we have

$$\varrho_{i,k+1}^{q+1} = \|\theta_{i,k+1}^q - \vec{G}_i(\hat{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^q)\|^2_{\boldsymbol{\Xi}_i = \varsigma_i{}^T \varsigma_i} \tag{4.123}$$

Now, the input law (4.56) can be written into a Linear Quadratic Tracking (LQT) problem:

$$\hat{u}_{i,k+1}^{q+1} = \operatorname{argmin}(\|\theta_{i,k+1}^q - \vec{G}_i(\hat{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^q)\|^2_{\boldsymbol{\Xi}_i} + \|\hat{u}_{i,k+1}^{q+1} - \eta_{i,k+1}^q\|^2_{\boldsymbol{\Psi}_i}) \tag{4.124}$$

and then substituting the term in (4.124) into standard LQT problem, we can obtain (4.106). That completes the proof. □

By contrast to matrix form, the feedback plus feedforward implementation not only requires less calculation (since it does not involve the calculation of the system matrix inverse), but also has stronger robustness (benefiting from the real-time state feedback).

4.4.4 Distributed ILC Algorithm 4.6

Using distributed implementation Algorithm 4.6.0 to implement Algorithm 4.5 distributively, we have the distributed ILC Algorithm 4.6 for constrained consensus tracking problem. For Algorithm 4.6, it follows the similar idea as Algorithm 4.4: it repetitively perform ADMM three steps in the inner loop (either in matrix form or feedback plus feedforward form) to approximate the centralised solution. However, the main difference between two algorithms is that: Algorithm 4.4 directly solves the constrained ILC problem in the ADMM inner loop, while Algorithm 4.6 solves an unconstrained problem in the ADMM inner loop and then performs the projection step (4.69) in Step 13 to find the input solution.

Remark 4.10. *Note that, Algorithm 4.6 requires less computation than Algorithm 4.4 (since it has analytic solution in the ADMM inner loop). However, it cannot guarantee the monotonic convergence of the consensus tracking error norm, which may cause some problems in practice (e.g., product quality problem).*

4.5 Numerical Example

In this section, we provide numerical examples to verify the effectiveness of all the proposed algorithms. Consider a heterogeneous networked dynamical system including seven subsystems, where i^{th} subsystem's dynamics a discrete-time, over-damping, non-minimum phase system, which described as

$$G_i(s) = \frac{15s - 1}{s^2 + 6s + \tau_i} \tag{4.125}$$

where $\tau_i = i$, $i = 1, 2, \cdots, 7$.

Figure 4.5 shows the network topology of the networked dynamical system. Assuming the trial length is $3s$ (i.e., $t \in [0, 3]$), sampling time $T_s = 0.05s$ (sampled using a zero-order hold), i^{th} subsystem's initial condition $x_{i,0} = 0$. The reference is defined as

$$r(t) = \begin{cases} sin\left[\pi(T_s t - 1)\right] & 20 \leq t \leq 40 \\ 0 & otherwise \end{cases} \tag{4.126}$$

4.5.1 Example 1: Perfect Consensus Tracking is Possible

We consider input saturation constraint (Pakshin et al., 2019) in the following examples and the case when perfect consensus tracking is possible is first investigated. To make sure the perfect consensus tracking is achievable, we assume each subsystem's input

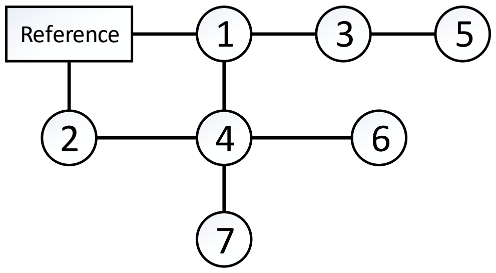

Figure 4.5: The graph structure of numerical example

constraint set is big enough to contain the optimal input solution:

$$\Omega_i = \left\{ u_i \in \mathbb{R}^N : -1 \leq u_i(t) \leq 1 \right\} \tag{4.127}$$

and in the following, we will investigate Algorithm 4.4 and Algorithm 4.6 separately to demonstrate their effectiveness.

4.5.1.1 Distributed Algorithm 4.4

As mentioned in Remark 4.6, the convergence speed of Algorithm 4.4 will be affected by the maximum-iteration q_{max}, and hence we investigate the effect of maximum-iteration q_{max} in the following. All the subsystems' initial input at the first trial are set to be 0, the weighting matrices Q and R are set to be I_N, and the penalty parameter ρ is set to 4. Figure 4.6 shows the convergence behaviour of distributed Algorithm 4.4 (using distributed implementation Algorithms 4.2 and 4.3) and centralised solution over 1000 ILC trials. For both distributed implementations, they guarantee the consensus tracking error norm converges monotonically in each ILC trial (with the same convergence behaviour) and eventually achieves perfect consensus tracking, which verifies Theorem 4.2.

Furthermore, as the increases of maximum-iteration number q_{max} in each ILC trial, the convergence speed of the proposed algorithm increases. However, this improvement on the convergence speed is marginal: after $q_{max} \geq 50$, the results of both distributed implementations are almost indistinguishable from the centralised result, showing that a small number of ADMM iterations is usually sufficient to approach the centralised solution in practice. The input of different subsystems on 1000^{th} ILC trial (with $q_{max} = 80$) are shown in Figure 4.7, and it shows that all the subsystem's input are constrained in the constrained set Ω_i, which achieves the desired objective (4.39).

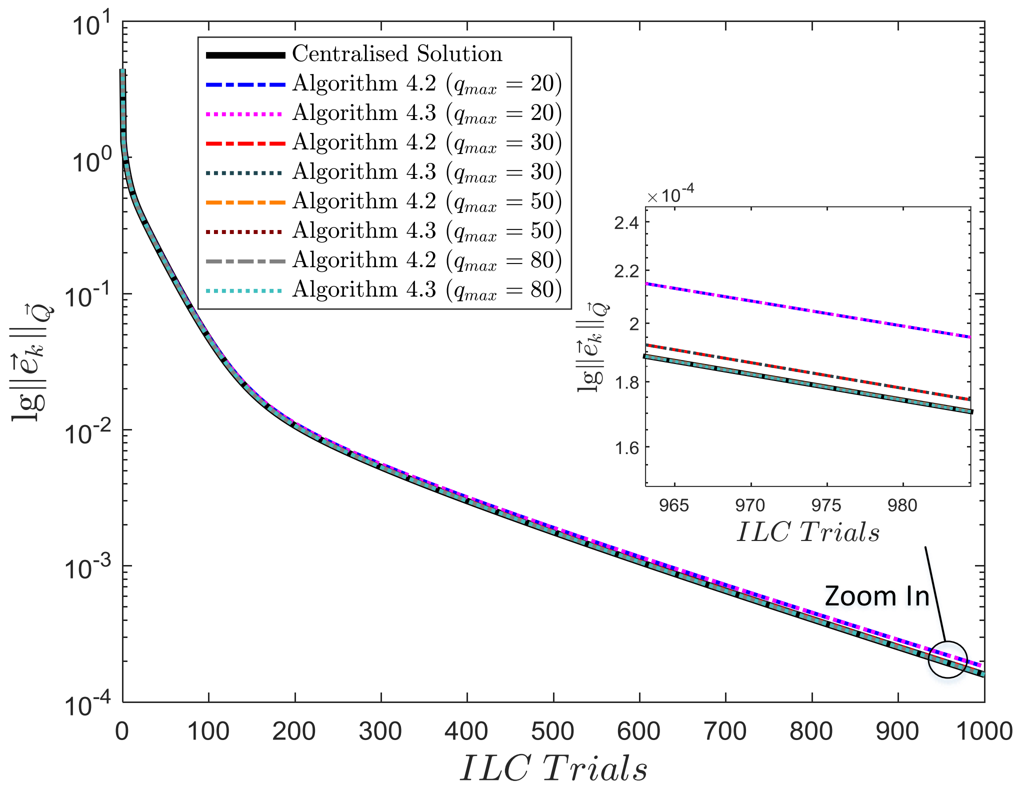

Figure 4.6: Algorithm 4.4 – Convergence of tracking error norm over 1000 trials (for different maximum-iteration q_{max})

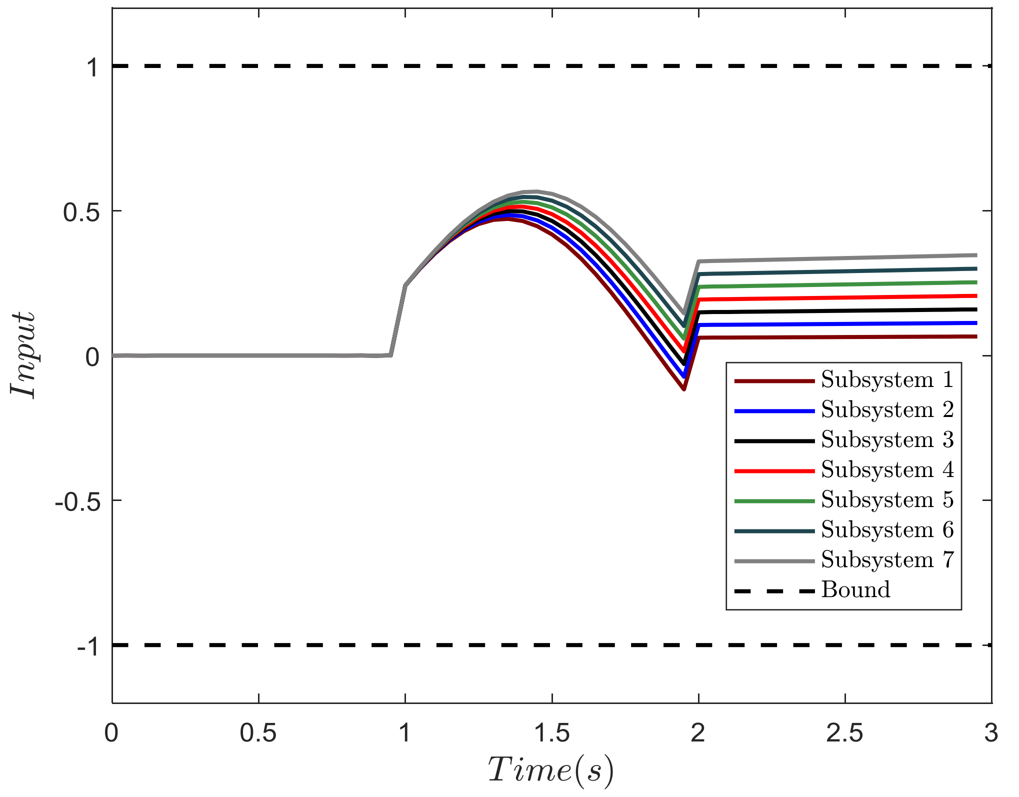

Figure 4.7: Algorithm 4.4 – Input signal of different subsystems on 1000th trial (when perfect consensus tracking is possible)

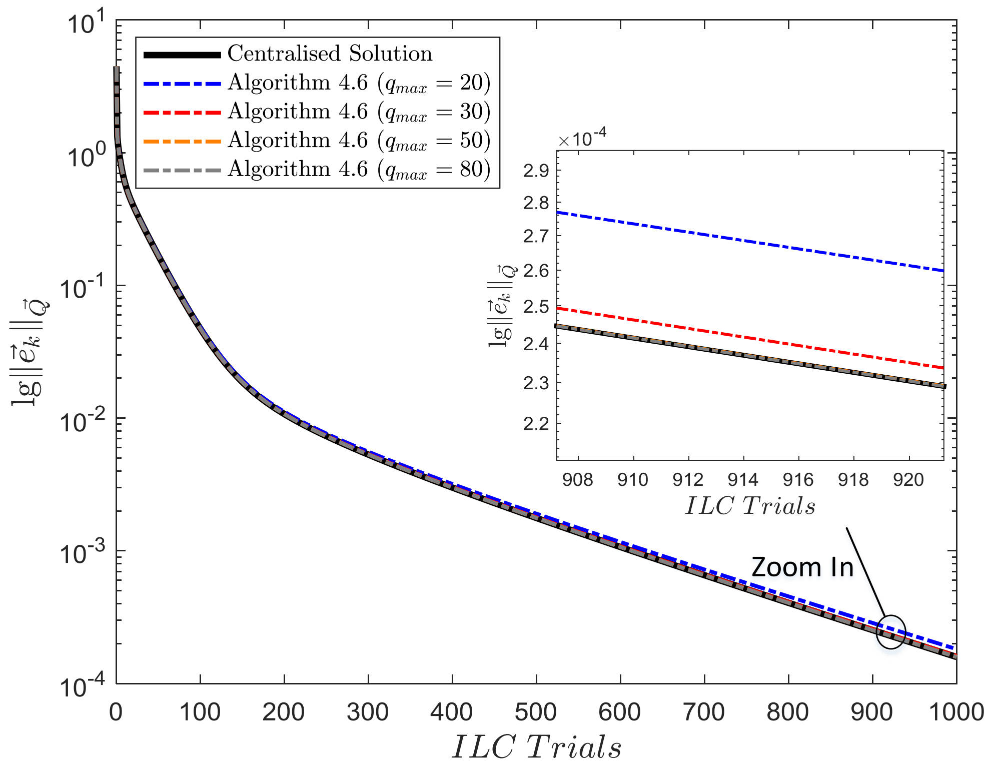

Figure 4.8: Algorithm 4.6 – Convergence of tracking error norm over 1000 trials (for different maximum-iteration q_{max})

4.5.1.2 Distributed Algorithm 4.6

To make comparison with Algorithm 4.4, we choose the same parameters as in last section, i.e., the weighting matrices $Q = R = I_N$, and the penalty parameter $\rho = 4$. Figure 4.8 shows the convergence behaviour of distributed Algorithm 4.6 over 1000 ILC trials. The figure shows that when perfect consensus tracking is possible, the proposed Algorithm 4.6 can guarantee the consensus tracking error norm converge to zero. Moreover, increasing ADMM maximum-iteration number q_{max} result in faster convergence before $q_{max} = 50$ and after that, the distributed results are equivalent to the centralised result, which demonstrate the high efficiency of ADMM in practice. By fixing the ADMM maximum-iteration $q_{max} = 80$, we have the input trajectory of seven subsystems as shown in Figure 4.9, demonstrating that each subsystem's input is constrained in the constraint set Ω_i.

4.5.2 Example 2: Perfect Consensus Tracking is Impossible

In this section, we assume that all the setting are the same as the last example except each subsystem's input constraint set is replaced by

$$\Omega_i = \{u_i \in \mathbb{R}^N : -0.35 \leq u_i(t) \leq 0.35\} \tag{4.128}$$

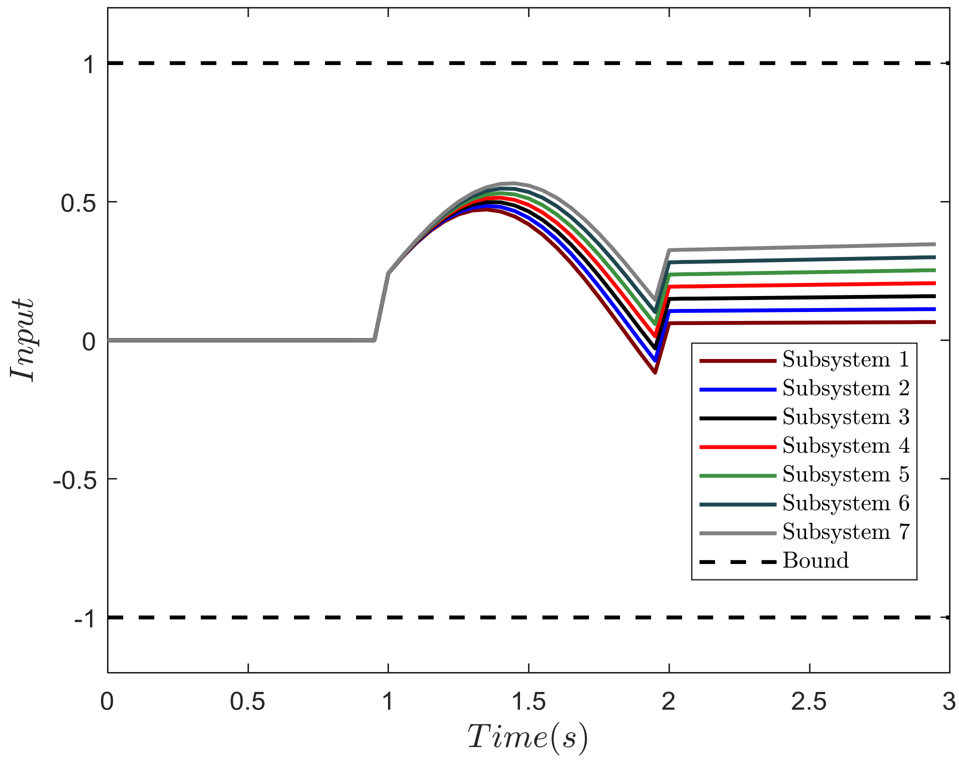

Figure 4.9: Algorithm 4.6 – Input signal of different subsystems on 1000th trial (when perfect consensus tracking is possible)

and hence the perfect consensus tracking is unachievable.

4.5.2.1 Distributed Algorithm 4.4

Figure 4.10 shows the convergence of the consensus consensus tracking error norm over 600 ILC trials. In this situation, both distributed algorithms (for different q_{max}) guarantee $\|\vec{e}_k\|_{\vec{Q}}$ converges monotonically to the 'best fit' solution, which verifies Theorem 4.4. All the agents' input on 600th trial are shown in Figure 4.11 and it demonstrates that each agent's resulting input is still constrained in the constraint set Ω_i , which confers to our expectations. Figure 4.12 shows the input energy consumption over 600 trials, and it demonstrates that Algorithm 4.4 guarantees the input converge to the optimal solution (4.41).

To investigate the effect of weighting matrices Q and R, the maximum-iteration number q_{max} is fixed to 50, with the penalty parameter $\rho = 4$ and the scale weighting $Q = 1$. Figure 4.13 shows the convergence performance of Algorithm 4.4 for different scalar weighting R. The figure shows that as the decreases of R, the convergence speed of consensus tracking error norm increases, which consist with our expectations. On the other hand, the improvement in convergence speed does not affect the convergent value and all the situations converge monotonically to the 'best fit' solution.

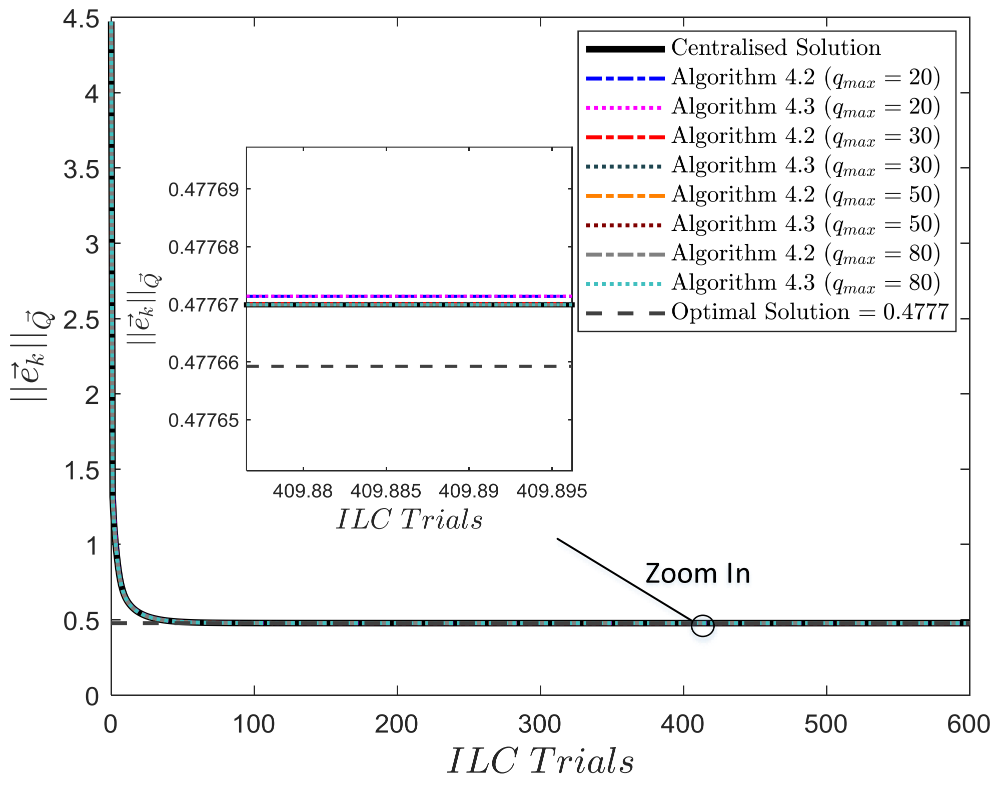

Figure 4.10: Algorithm 4.4 – Convergence of tracking error norm over 600 trials (for different maximum-iteration q_{max})

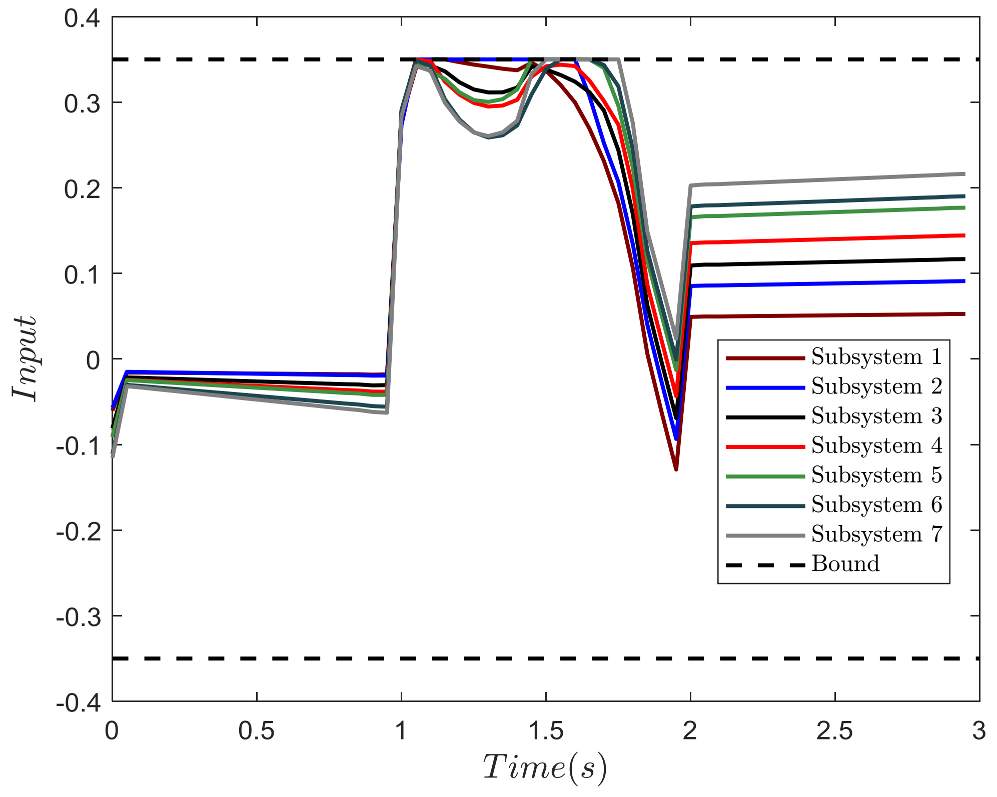

Figure 4.11: Algorithm 4.4 – Input signal of different subsystems on 600th trial (when perfect consensus tracking is impossible)

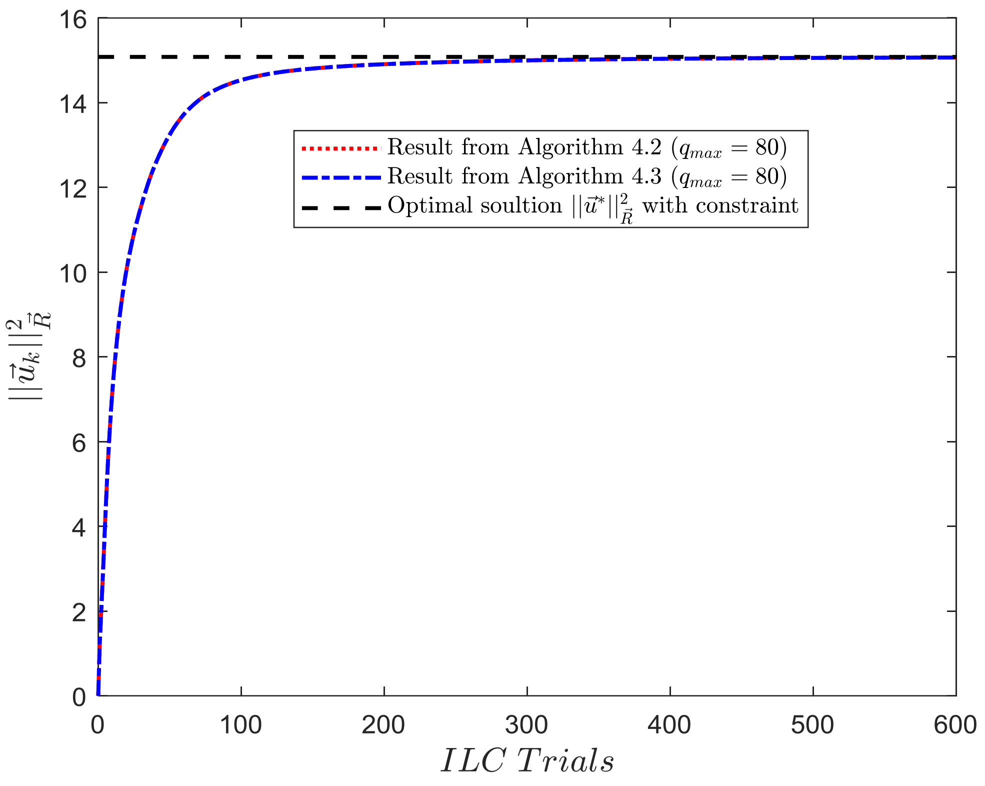

Figure 4.12: Algorithm 4.4 – Convergence of input energy consumption over 600 trials

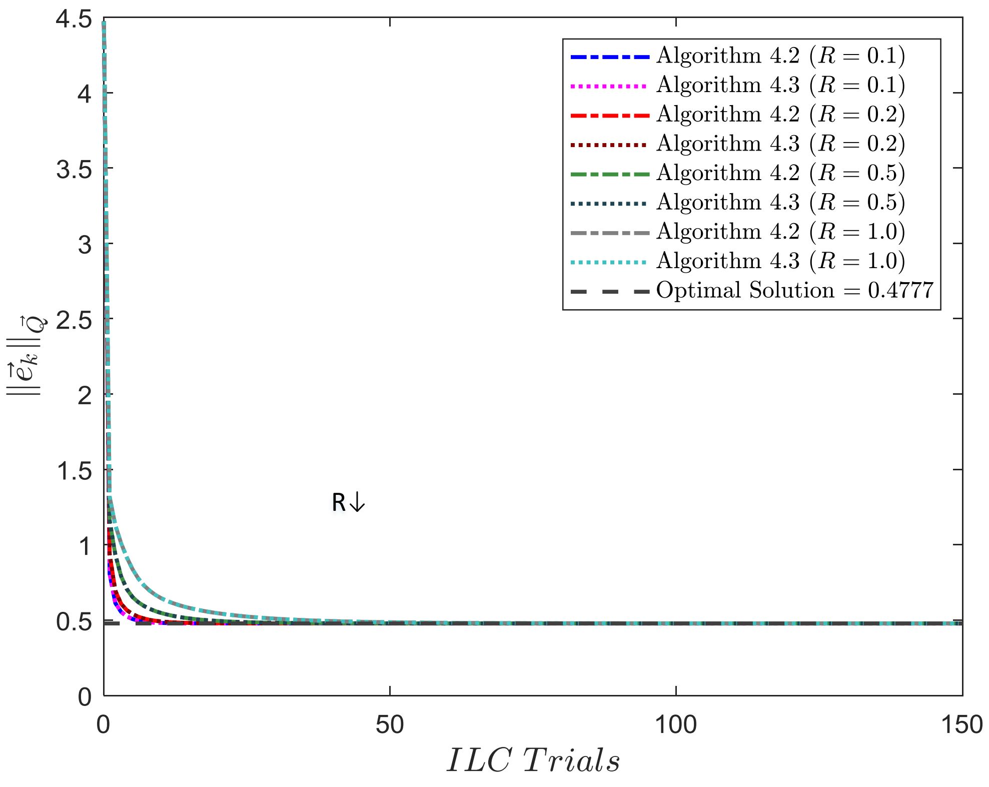

Figure 4.13: Algorithm 4.4 – Convergence of tracking error norm over 150 trials (for different scale weighting R)

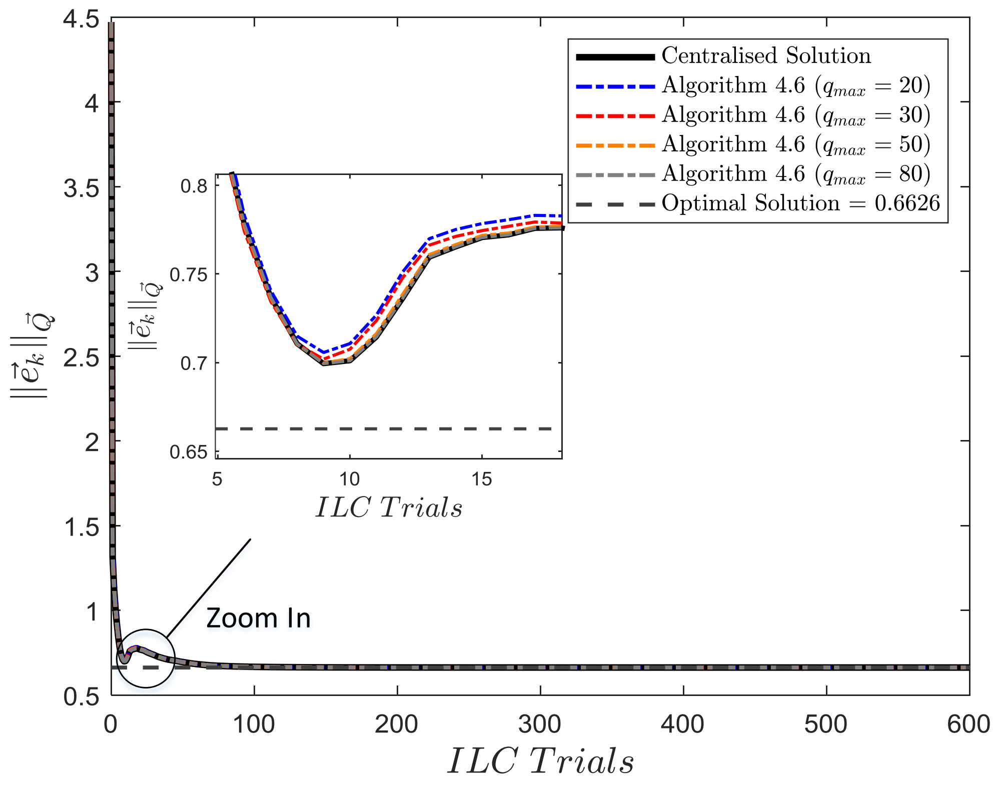

Figure 4.14: Algorithm 4.6 – Convergence of tracking error norm over 600 trials (for different maximum-iteration q_{max})

4.5.2.2 Distributed Algorithm 4.6

Figure 4.14 shows the convergence behaviour for different q_{max} over 600 ILC trials. It can be concluded that when perfect consensus tracking is impossible, the proposed algorithm guarantees the consensus tracking error norm converge asymptotically to a 'best approximation' solution (relating to the weighting matrices Q and R). For the effect of maximum-iteration q_{max} on the convergence speed, it confers to the phenomenon concluded in the previous simulations.

Figure 4.15 shows the input signal of different subsystems when perfect consensus tracking is impossible, which supports that the proposed Algorithm 4.6 can also guarantee the satisfaction of system constraints. For Figure 4.16, it shows that the total input energy consumption will converge to the desired solution (4.95), which verifies Theorem 4.7.

For the investigation of weighting matrices, we set the maximum-iteration number $q_{max} = 50$, with the penalty parameter $\rho = 4$ and scalar weighting $Q = 1$. Figure 4.17 shows the convergence performance of Algorithm 4.6 when scalar weighting R are set to be 0.1, 0.2, 0.5, 1, respectively. The figure shows that decreasing the scalar weighting R will increase the convergence speed, however, the changing of weighting matrix R also affect the convergent value (which different from Algorithm 4.4 that return the

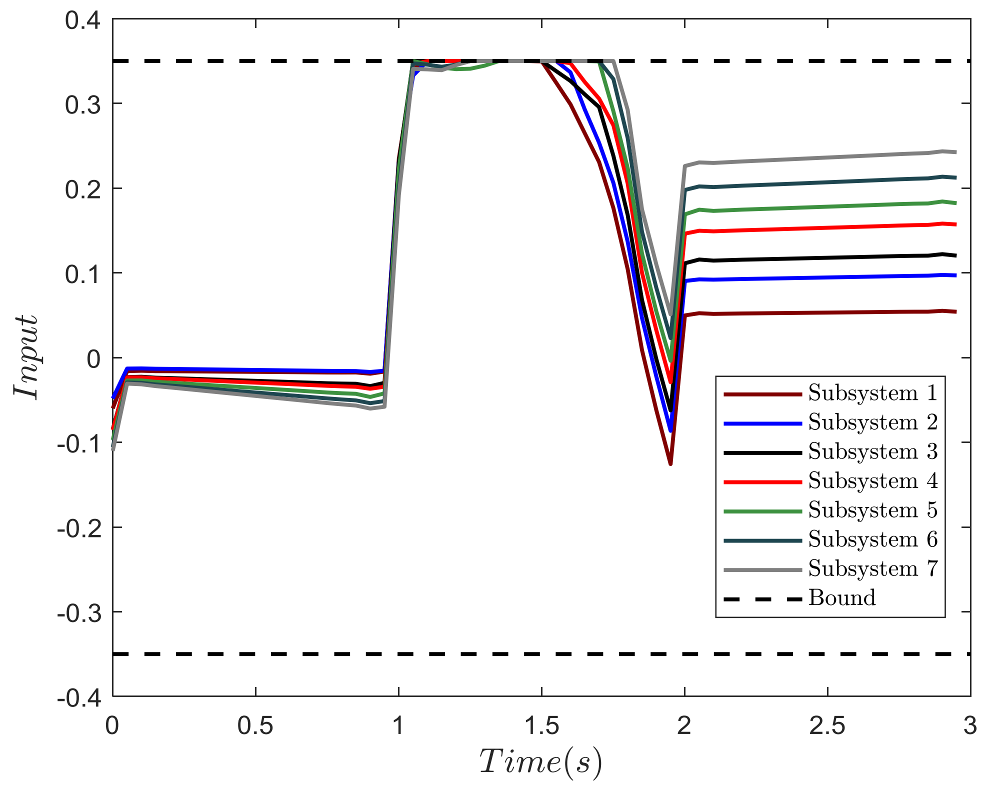

Figure 4.15: Algorithm 4.6 – Input signal of different subsystems on 600th trial (when perfect consensus tracking is impossible)

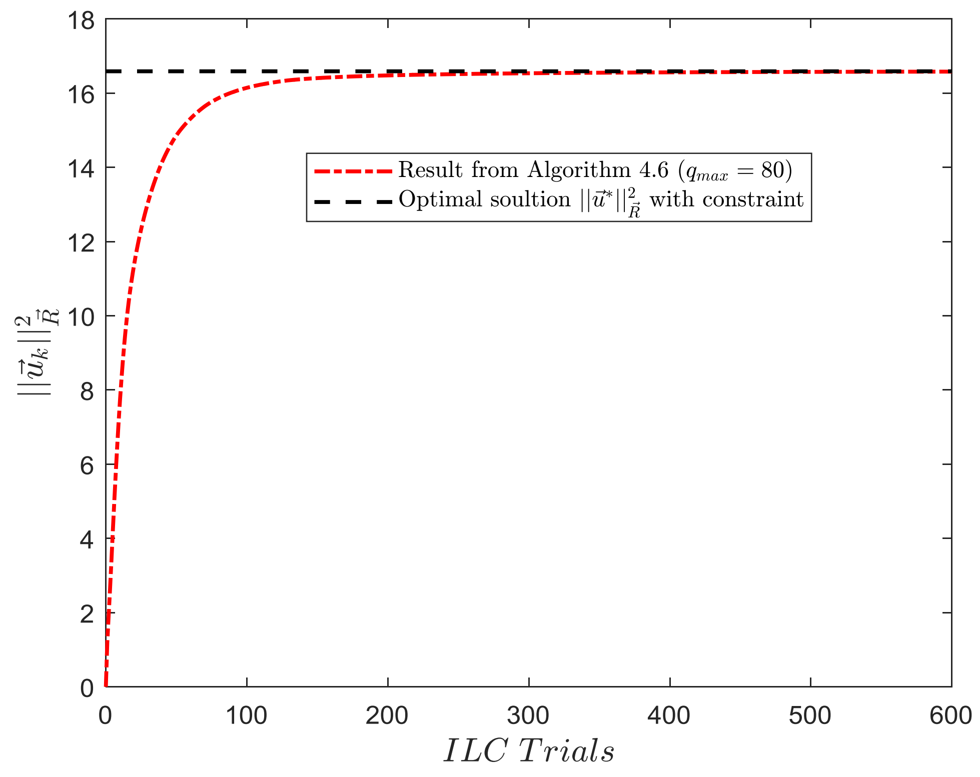

Figure 4.16: Algorithm 4.6 – Convergence of input energy consumption over 600 trials

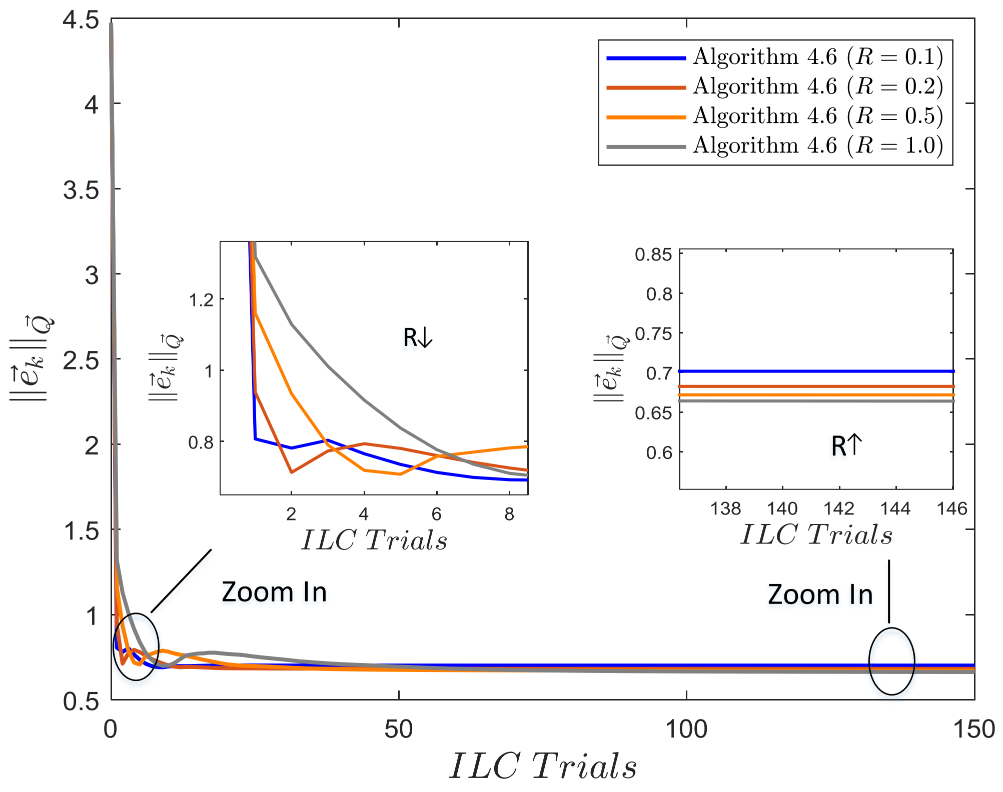

Figure 4.17: Algorithm 4.6 – Convergence of tracking error norm over 150 trials (for different scale weighting R)

same convergent value for different weighting matrices).

It should be noted that although Algorithm 4.4 has better convergence performance than Algorithm 4.6, that comes at the cost of a large calculation load. For some practical applications, they may require nearly optimal performance using simple computation, and then Algorithm 4.6 is much useful for these applications.

4.6 Summary

In this chapter, we consider an important design problem in consensus tracking of networked dynamical systems, which is called as the constraint handling problem. We use a well-known framework, namely successive projection framework, to reformulate the constrained consensus tracking problem and then propose two novel ILC algorithms. For both ILC algorithms, they have appealing convergence performance: when perfect consensus tracking is possible, they can guarantee the consensus tracking error norm converges to zero while satisfying the system constraints during the whole control process; when perfect consensus tracking is impossible, they can guarantee the sanctification of the system constraints and eventually converge to 'best fit' solutions. Note that, the main difference between the proposed algorithms is that one can always guarantee the monotonic convergence of consensus tracking error norm, while the other requires less computation. These observations need to be taken into consideration when selecting the algorithm in practice, which is a trade-off between the computation cost and convergence performance.

To avoid the huge computational complexity of the proposed algorithms for large scale networked dynamical systems, we use the idea of ADMM to design distributed methods (including different implementations) for the proposed ILC algorithms, allowing the input updating to be done in a distributed manner and have great scalability. Furthermore, the proposed distributed ILC algorithms can be applied to both homogeneous and heterogeneous networks, as well as non-minimum phase systems, which are appealing in real applications. The convergence properties of the algorithms are analysed rigorously and numerical examples are presented to demonstrate the algorithms' effectiveness. Though this chapter focus on consensus tracking on the whole trial length, the idea can also be extended to solve point-to-point (P2P) task without any difficulties.

Note that, all the existing results in this chapter are established on the fixed topology, while the switching topologies (that is another important class of network structure) have not been considered. To fill this gap, we will propose a novel distributed ILC algorithm to deal with the switching topologies in the next chapter.

Chapter 5

Distributed ILC for Networked Systems with Switching Topologies

Existing ILC algorithms for high performance consensus tracking problem mainly deal with networked dynamical systems with fixed network topology, i.e., the networked relationship between different subsystems is fixed during the control process. However, in some applications, the network topology may change for security reason or performance requirements. As an example, modern air transportation system contains a number of unmanned aerial vehicles (UAVs) that repetitively transport 'goods' from the origin to the destination, and normally one UAV can only talk to its neighbouring UAVs during the control process. Starting from the initial condition, the position of different UVAs may differ from the last trial and it leads to the change in network topology (i.e., the topology is switching). For such situation, the ability to deal with the switching topologies is crucially important. It is worth mentioning that there exist some ILC design algorithms especially working for high performance consensus tracking problem with switching topologies, e.g., Li et al. (2015); Liu et al. (2019); Meng et al. (2015a, 2016). However, the convergence performance of these algorithms are only guaranteed when the controller satisfies certain conditions.

Motivated by the work on ILC for networked dynamical systems with fixed topology in Chapter 3 , this chapter further relaxes the restriction on the network topology and proposes a novel ILC design to address the aforementioned limitations in the existing literature. By designing a novel performance index, the resulting ILC algorithm guarantees monotonic convergence of the tracking error norm to zero (without any restriction on the controller) even with switching topologies. Furthermore, since the proposed performance index can be written into a separable form, a distributed implementation method via the alternating direction method of multipliers (ADMM) is proposed, which allows the proposed algorithm to be applied to large scale networked dynamical systems. Finally, the proposed algorithm can be applied to both homogeneous and heterogeneous networked dynamical systems, which is appealing in practical applications where the dynamics of each subsystem can be different. This chapter is based on the work from Chen and Chu (2022).

The rest of this chapter is organised as follows: Section 5.1 provides the formulation of the system dynamics, network topology and control design objective; Section 5.2 introduces the proposed ILC algorithm and analyses its convergence properties rigorously; Section 5.3 provides a distributed implementation for the proposed ILC algorithm; Section 5.4 uses numerical examples to verify the effectiveness of the proposed algorithm and finally, Section 5.5 concludes this chapter.

5.1 Problem Formulation

In this section, the dynamics and network topology of the system are introduced, and the design objective of the high performance consensus tracking problem is provided. For simplicity, we consider a discrete time, single-input-single-output (SISO) system in this chapter.

5.1.1 System Dynamics

Consider a (homogeneous or heterogeneous) networked dynamical system with p subsystems, where i^{th} $(1 \leq i \leq p)$ subsystem's dynamics is represented using the following linear time invariant (LTI), SISO, discrete time state space model

$$\begin{aligned} x_{i,k}(t+1) &= A_i x_{i,k}(t) + B_i u_{i,k}(t) \qquad & x_{i,k}(0) &= x_{i,0} \\ y_{i,k}(t) &= C_i x_{i,k}(t) & t &\in [0, N] \end{aligned} \tag{5.1}$$

where t is the time index; k is the ILC trial index; N is the trial length; $x_{i,k}(\cdot) \in \mathbb{R}^{n_i}$ (n_i is the order of i^{th} subsystem), $u_{i,k}(\cdot) \in \mathbb{R}$, $y_{i,k}(\cdot) \in \mathbb{R}$ are state, input and output of subsystem i on k^{th} trial; A_i, B_i, C_i are system matrices with proper dimensions. The system is required to work in a repetitive manner, i.e., starting from the initial condition $x_{i,0}$, it performs the same task within the time interval $[0, N]$ and after $t = N + 1$, the time t is reset to 0, the state of i^{th} subsystem is reset to the initial condition $x_{i,0}$, and the system will execute the same design task again.

For high performance consensus tracking problem, it requires all the subsystems track the same reference $r(t)$ that is only known by part of subsystems, which makes the design non-trivial. To facilitate subsequent ILC design, a 'lifted form' representation proposed in Hatonen et al. (2004) is introduced. By assuming unity relative degree for all the subsystems (i.e., $C_iB_i \neq 0$), the ‘lifted’ input, output and reference are represented as

$$\begin{aligned} u_{i,k} &= [u_{i,k}(0), u_{i,k}(1), \cdots, u_{i,k}(N-1)]^T \in \mathcal{U}_i \\ y_{i,k} &= [y_{i,k}(1), y_{i,k}(2), \cdots, y_{i,k}(N)]^T \in \mathcal{Y}_i \\ r &= [r(1), r(2), \cdots, r(N)]^T \in \mathcal{Y}_i \end{aligned} \tag{5.2}$$

where the input space $\mathcal{U}_i = \mathbb{R}^N$ and output space $\mathcal{Y}_i = \mathbb{R}^N$ are defined with inner products and induced norms

$$\begin{aligned} \langle u, v\rangle_{R_i} &= u^T R_i v, \qquad \|u\|_{R_i} = \sqrt{\langle u, u\rangle_{R_i}} \\ \langle x, y\rangle_{Q_i} &= x^T Q_i y, \qquad \|y\|_{Q_i} = \sqrt{\langle y, y\rangle_{Q_i}} \end{aligned} \tag{5.3}$$

in which R_i, Q_i are symmetric positive definite matrices.

The system model (5.1) can then be rewritten as

$$y_{i,k} = G_i u_{i,k} + d_i \tag{5.4}$$

where the response of initial conditions $d_i \in \mathbb{R}^N$ is

$$d_i = \left[C_iA_ix_{i,0}, C_iA_i^2x_{i,0}, C_iA_i^3x_{i,0}, \cdots, C_iA_i^Nx_{i,0}\right]^T \tag{5.5}$$

and the system matrix G_i is defined as

$$G_i = \begin{bmatrix} C_iB_i & 0 & \cdots & 0 \\ C_iA_iB_i & C_iB_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_iA_i^{N-1}B_i & C_iA_i^{N-2}B_i & \cdots & C_iB_i \end{bmatrix} \tag{5.6}$$

Introduce $\vec{u}_k$, $\vec{y}_k$, $\vec{r}$, $\vec{d}$ as the global vector of ‘lifted’ input $u_{i,k}$, output $y_{i,k}$, reference r, and initial condition’s respond d_i, which are represented as

$$\begin{aligned} \vec{u}_k &= \left[{u_{1,k}}^T, {u_{2,k}}^T, \cdots, {u_{p,k}}^T\right]^T \in \mathcal{U} \qquad & \vec{y}_k &= \left[{y_{1,k}}^T, {y_{2,k}}^T, \cdots, {y_{p,k}}^T\right]^T \in \mathcal{Y} \\ \vec{r} &= \left[r^T, r^T, \cdots, r^T\right]^T \in \mathcal{Y} & \vec{d} &= \left[d_1^T, d_2^T, \cdots, d_p^T\right]^T \in \mathcal{Y} \end{aligned} \tag{5.7}$$

in which the global input space $\mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2 \times \cdots \times \mathcal{U}_p$ and global output space $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_p$ are defined with inner products and induced norms

$$\begin{aligned} \langle \vec{u}, \vec{v}\rangle_{\vec{R}} &= \sum_{i=1}^{p} u_i^T R_i v_i, \qquad \|\vec{u}\|_{\vec{R}} = \sqrt{\langle \vec{u}, \vec{u}\rangle_{\vec{R}}} \\ \langle \vec{x}, \vec{y}\rangle_{\vec{Q}} &= \sum_{i=1}^{p} x_i^T Q_i y_i, \qquad \|\vec{y}\|_{\vec{Q}} = \sqrt{\langle \vec{y}, \vec{y}\rangle_{\vec{Q}}} \end{aligned} \tag{5.8}$$

where $\times$ represents the Cartesian product, and matrices $\vec{Q} = \text{diag}(Q_1, Q_2, \cdots, Q_p)$, $\vec{R} = \text{diag}(R_1, R_2, \cdots, R_p)$.

The global system model can be then rewritten as

$$\vec{y}_k = \mathbb{G}\vec{u}_k + \vec{d} \tag{5.9}$$

where $\mathbb{G} = \text{diag}\ (G_1, G_2, \cdots, G_p)$.

The high performance consensus tracking problem can be stated as finding a proper global input $\vec{u}_k$ such that the global output $\vec{y}_k$ tracks the global reference $\vec{r}$ accurately even when $\vec{r}$ is not generally known by all the subsystems.

5.1.2 Network Topology

This chapter uses a switching undirected graph $\mathcal{G}_k = (\mathscr{V},\mathscr{E}_k)$ to represent the network topology on k^{th} trial, where $\mathscr{V} = \{1, 2, ..., p\}$ is the node set and $\mathscr{E}_k \subset \mathscr{V} \times \mathscr{V}$ is the set of pairs of nodes on k^{th} trial called edges. For i^{th} subsystem, its neighbours set on k^{th} trial is represented as $\mathscr{N}_{i,k} := \{j : (i,j) \in \mathscr{E}_k\}$. In addition, we introduce a set $\mathcal{G}_\sigma = \{\mathcal{G}_{\sigma 1}, \mathcal{G}_{\sigma 2}, \cdots\}$ to describe all the possible graph defined for the subsystems , i.e., $\mathcal{G}_k \in \mathcal{G}_\sigma$ for all k. It is worth pointing out that $\mathcal{G}_\sigma$ has finite elements.

The Laplacian matrix $L_k = \{l_{ij,k}\}$ indicates the relationship between different nodes, and it is a real positive semi-definite, symmetric matrix with element $l_{ij,k}$ defined as

$$l_{ij,k} = \begin{cases} -W_{ij,k} & if \ \ j \in \mathscr{N}_{i,k} \\ \sum_{j \in \mathscr{N}_{i,k}} W_{ij,k} & if \quad j = i \\ 0 & otherwise \end{cases} \tag{5.10}$$

where $W_{ij,k}$ is the weight used to denotes the edge's connection strength on trial k (Bullo, 2018). When nodes i and j are neighbours on trial k, $W_{ij,k} > 0$; otherwise, $W_{ij,k} = 0$. Note that, the larger the value $W_{ij,k}$ is, the stronger connection between subsystems i and j at trial k will be, i.e., subsystem i will put more emphasis on the information transformation with subsystem j. Given an example, in the transportation system, there exist some subsystems that have direct access to the reference trajectory (i.e., 'leader'), their neighbours (without access to the reference) know the leaders have information about the desired objective and hence they will put more emphasize on communicating information with the 'leader' subsystems to improve its tracking efficiency.

As mentioned above, the main difficulty of the high performance consensus tracking problem is the limited access of subsystems to the reference. To indicate the accessibility of different subsystems on different trials, we further define a reference-accessibility matrix $D_k = \text{diag}\{\mathcal{D}_{ii,k}\}$, with its element $\mathcal{D}_{ii,k}$ defined as

$$\mathcal{D}_{ii,k} = \begin{cases} 1 & \textit{if agent } i \textit{ has access on trial } k \\ 0 & \textit{otherwise} \end{cases} \tag{5.11}$$

To ensure the design objective is achievable, the following assumptions are necessary:

Assumption 5.1. *The switching graph $\mathcal{G}_\sigma$ is always connected, i.e., in each ILC trial, one node can be reached by another nodes through one path (Bullo, 2018).*

Assumption 5.2. *In each ILC trial, at least one subsystem has access to the reference r, i.e., $\exists i, \mathcal{D}_{ii,k} \neq 0$.*

5.1.3 Iterative Learning Control Design

The ILC design objective of high performance consensus tracking problem can be stated as finding a global input in the following form

$$\vec{u}_{k+1} = f(\vec{u}_k, \vec{e}_k) \tag{5.12}$$

such that all the subsystems' output track the reference trajectory as $k \to \infty$, i.e.

$$\lim_{k\to\infty} y_{i,k} = r, \quad i = 1, 2, \cdots, p \tag{5.13}$$

where $\vec{e}_k = \vec{r} - \vec{y}_k$ is the 'virtual' tracking error and only part of it is available in ILC design since the reference trajectory is only known by few of the subsystems in practice. Note that finding the optimal input for objective (5.13) is non-trivial because the network topology keeps changing during the control process.

5.2 Optimization-Based ILC Algorithm for High Performance Consensus Tracking Problem

In this section, a novel optimization-based ILC algorithm for high performance consensus tracking problem is provided and its convergence properties are analysed rigorously.

5.2.1 Algorithm Description

Algorithm 5.1. *For any initial choice of $\vec{u}_0$, the input series $\{\vec{u}_k\}_{k\geq 0}$ defined as follows*

$$\vec{u}_{k+1} = \arg\min \{J_{k+1}(\vec{u}_{k+1})\} \tag{5.14}$$

provides a solution for high performance consensus tracking problem, i.e.

$$\lim_{k\to\infty} y_{i,k} = r, \quad i = 1, 2, \cdots, p \tag{5.15}$$

in which $J_{k+1}(\vec{u}_{k+1})$ is defined as

$$J_{k+1}(\vec{u}_{k+1}) = \|(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{e}_{k+1}\|^2_{\vec{Q}} + \|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}} \tag{5.16}$$

where the weighting matrix $\vec{R}$ is selected as

$$\vec{R} = \gamma \mathbb{G}^T \mathbb{G}, \tag{5.17}$$

and $\boldsymbol{D}_{k+1} = D_{k+1} \otimes I_N$, $\boldsymbol{L}_{k+1} = L_{k+1} \otimes I_N$, $\vec{e}_{k+1} = \vec{r} - \vec{y}_{k+1} = \vec{e}_k - \mathbb{G}(\vec{u}_{k+1} - \vec{u}_k)$, I_N is an N by N identity matrix, $\otimes$ represents the Kronecker product and γ is a positive scaling factor.

Remark 5.1. *Algorithm 5.1 is an extended version of the basic consensus NOILC algorithm (that is proposed in Chapter 3) for networked dynamical systems with switching topologies. In the performance index (5.16), we introduce matrix $\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1}$ to ensure the reference information is always transforming within the network during the control process (based on Assumptions 5.1 and 5.2), which eventually achieve the perfect consensus tracking. However, as the network is changing during the control process, the monotonic convergence is not longer guaranteed for any weighting matrix $\vec{R}$. To recover the monotonic convergence property (that is desirable in practice) of the proposed algorithm, we observe that $\vec{R} = \gamma \mathbb{G}^T \mathbb{G}$ is a proper choice. A detailed convergence analysis will be introduced in the next section.*

5.2.2 Convergence Properties of the Proposed Algorithm

Using Algorithm 5.1 to solve the high performance consensus tracking problem, it has the desired convergence properties shown in the following theorem.

Theorem 5.1. *For any initial choice of $\vec{u}_0$ and $\vec{e}_0$, Algorithm 5.1 guarantees monotonic convergence of the 'virtual' tracking error norm to 0, i.e.,*

$$\|\vec{e}_{k+1}\|_{\vec{Q}} \le \|\vec{e}_k\|_{\vec{Q}}, \qquad \lim_{k\to\infty} \vec{e}_k = 0. \tag{5.18}$$

Consequently, all the subsystems track the reference trajectory perfectly as $k \to \infty$, i.e.

$$\lim_{k\to\infty} y_{i,k} = r, \quad i = 1, 2, \cdots, p \tag{5.19}$$

Proof. Solving the problem (5.14) on trial $k+1$, we have

$$\vec{u}_{k+1} = \vec{u}_k + \gamma^{-1}\mathbb{G}^{-1}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T \vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{e}_{k+1} \tag{5.20}$$

From (5.20), we have

$$\vec{e}_k = \left[I_{pN} + \gamma^{-1}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T \vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\right] \vec{e}_{k+1} \tag{5.21}$$

hence

$$\begin{aligned}\|\vec{e}_k\|^2_{\vec{Q}} &= \left[\vec{e}_{k+1} + \gamma^{-1}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T \vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{e}_{k+1}\right]^T \\ &\quad \times \left[\vec{Q}\vec{e}_{k+1} + \gamma^{-1}\vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T \vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{e}_{k+1}\right] \\ &= \|\vec{e}_{k+1}\|^2_{\vec{Q}} + 2\gamma^{-1}\vec{e}^{\,T}_{k+1}\left[(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{Q}\right]^2 \vec{e}_{k+1} \\ &\quad + \gamma^{-2}\vec{e}^{\,T}_{k+1}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T \vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T \vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{e}_{k+1}\end{aligned}$$

Note that

$$\begin{gathered}2\gamma^{-1}\vec{e}^{\,T}_{k+1}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{e}_{k+1} \geq 0 \\ \gamma^{-2}\vec{e}^{\,T}_{k+1}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T \vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T \vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{e}_{k+1} \geq 0\end{gathered}$$

hence

$$\|\vec{e}_k\|^2_{\vec{Q}} \geq \|\vec{e}_{k+1}\|^2_{\vec{Q}} \tag{5.22}$$

Therefore, $\|\vec{e}_k\|^2_{\vec{Q}}$ is converging (as is bounded below). Hence

$$\vec{e}^{\,T}_{k+1}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T \vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{e}_{k+1} \to 0 \tag{5.23}$$

It follows that

$$(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{e}_{k+1} \to 0 \tag{5.24}$$

Note that the possible graph set $\mathcal{G}_\sigma$ has finite element, matrix $\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1}$ is positive definite for all the k (based on Lemma 3.2 and Assumptions 5.1, 5.2). Hence, $\lim_{k\to\infty} \vec{e}_k = 0$. That completes the proof. □

Theorem 5.1 shows that the proposed algorithm achieves monotonic convergence of the tracking error norm to zero. As can be seen from Equation (5.21), the convergence speed of 'virtual' tracking error norm depends on the choice of scaling factor γ: a smaller scaling factor γ allows bigger input change and hence it leads to faster convergence speed. This will also be demonstrated later in numerical simulations.

Remark 5.2. *The centralised solution of Algorithm 5.1 can be found by directly calculating the stationary point of (5.16), and the input updating law is shown as follows*

$$\begin{aligned}\vec{u}_{k+1} = \vec{u}_k + \mathbb{G}^{-1}&\left[\gamma I_{pN} + (\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T \vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\right]^{-1} \\ &\times (\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})^T \vec{Q}(\boldsymbol{L}_{k+1} + \boldsymbol{D}_{k+1})\vec{e}_k\end{aligned} \tag{5.25}$$

However, centralised realization requires significant computational complexity when the subsystem's number is large. Hence, distributed implementation is more essential and applicable in practice.

5.3 Distributed Implementation of Algorithm 5.1

In this section, the idea of 'consensus' formulation in ADMM is used to implement Algorithm 5.1 in distributed manner. In the following, we first review the idea of 'consensus' formulation in ADMM.

5.3.1 The Alternating Direction Method of Multipliers

ADMM is a method that is well suited to distributed implementation and it has been widely used in different areas, e.g., in machine learning, applied statistics and privacy preservation (Boyd et al., 2011; Cheng et al., 2017; Deng and Yin, 2016; Zhang et al., 2019a). To introduce the general idea of ADMM, we consider the following optimisation problem

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{p} f_i(x_i) \\ &\text{subject to} \quad x_i - \widetilde{E}_i z = 0, \quad i = 1, \cdots, p \end{aligned} \tag{5.26}$$

where $x_i \in \mathbb{R}^{p_i}$, $z \in \mathbb{R}^p$ are the local variable and the global variable; $\widetilde{E}_i$ is a matrix that maps the local competent to the global element. For problem (5.26), ADMM can find the optimal solution by iteratively performing the following steps

$$x_i^{q+1} = \arg\min L_{\rho i}(x_i, z^q, \gamma_i^q) \tag{5.27}$$

$$z^{q+1} = \arg\min L_\rho(x^{q+1}, z, \gamma^q) \tag{5.28}$$

$$\gamma_i^{q+1} = \gamma_i^q + \rho(x_i^{q+1} - \widetilde{E}_i z^{q+1}) \tag{5.29}$$

where q is the ADMM iteration index, ρ is the penalty parameter, $\gamma_i \in \mathbb{R}^{p_i}$ is the dual variable and the augmented Lagrangian L_ρ is defined as

$$\begin{aligned} L_\rho(x, z, \gamma) &= \sum_{i=1}^{p} L_{\rho i}(x_i, z, \gamma_i) \\ &= \sum_{i=1}^{p} \left[f_i(x_i) + \gamma_i^T (x_i - \widetilde{E}_i z) + \frac{\rho}{2} \|x_i - \widetilde{E}_i z\|^2 \right] \end{aligned} \tag{5.30}$$

In obvious contrast to most of the distributed methodologies, ADMM can guarantee the objective convergence for any positive penalty parameter, which is appealing in practice. For detailed proof of its convergence properties, please refer to Boyd et al. (2011).

5.3.2 Distributed Implementation of Algorithm 5.1

On trial $k+1$, the optimal input solution of (5.14) can be found using (5.25). Note that $\mathbb{G}$ has a block diagonal structure. Therefore, if $\Delta\tilde{u}_{k+1}$ defined as follows

$$\Delta\tilde{u}_{k+1} = [\gamma I_{pN} + (\boldsymbol{L}_{k+1}+\boldsymbol{D}_{k+1})^T\vec{Q}(\boldsymbol{L}_{k+1}+\boldsymbol{D}_{k+1})]^{-1}(\boldsymbol{L}_{k+1}+\boldsymbol{D}_{k+1})^T\vec{Q}(\boldsymbol{L}_{k+1}+\boldsymbol{D}_{k+1})\vec{e}_k \tag{5.31}$$

is obtained, the algorithm can then be implemented as

$$\vec{u}_{k+1} := \vec{u}_k + \mathbb{G}^{-1}\Delta\tilde{u}_{k+1} \tag{5.32}$$

or in a local manner as

$$u_{i,k+1} := u_{i,k} + G_i^{-1}\Delta\tilde{u}_{i,k+1} \tag{5.33}$$

where $\Delta\tilde{u}_{k+1}$ is a global vector including the local $\Delta\tilde{u}_{i,k+1}$ for all the subsystems.

Computing (5.31) is equivalent to solving the following optimisation problem:

$$\Delta\tilde{u}_{k+1} = \arg\min_{\Delta\tilde{u}_{k+1}} \{J_{k+1}(\Delta\tilde{u}_{k+1})\} \tag{5.34}$$

where

$$J_{k+1}(\Delta\tilde{u}_{k+1}) = \|(\boldsymbol{L}_{k+1}+\boldsymbol{D}_{k+1})(\vec{r}-\vec{y}_k-\Delta\tilde{u}_{k+1})\|^2_{\vec{Q}} + \gamma\|\Delta\tilde{u}_{k+1}\|^2_{I_{pN}}$$

which can be rewritten into the following form

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{p} J_{i,k+1}(\Delta\tilde{\mathbf{u}}_{i,k+1}) \\ \text{subject to} \quad & \Delta\tilde{\mathbf{u}}_{i,k+1} - \widetilde{E}_{i,k+1}z_{k+1} = 0, \qquad 1 \le i \le p \end{aligned} \tag{5.35}$$

where i^{th} subsystem's local cost function is defined as

$$\begin{aligned} J_{i,k+1}(\Delta\tilde{\mathbf{u}}_{i,k+1}) = \| & \sum_{j\in\mathcal{N}_{i,k+1}} W_{ij,k}(y_{i,k} - y_{j,k} + \Delta\tilde{u}_{i,k+1} - \Delta\tilde{u}_{j,k+1}) \\ & + \boldsymbol{\mathcal{D}}_{ii,k+1}(r - y_{i,k} - \Delta\tilde{u}_{i,k+1})\|^2_{Q_i} + \gamma\|\Delta\tilde{u}_{i,k+1}\|^2_{I_N} \end{aligned}$$

in which $\boldsymbol{\mathcal{D}_{ii,k+1}} = \mathcal{D}_{ii,k+1} \otimes I_N$, z_{k+1} is a global vector contains all the element $z_{i,k+1}$, $\Delta\tilde{\mathbf{u}}_{i,k+1}$ is a local plan for i^{th} agent and its neighbours, and $\widetilde{E}_{i,k+1}$ is the local mapping matrix, e.g., if $\mathcal{N}_{i,k+1} = \{l, m\}$, $\Delta\tilde{\mathbf{u}}_{i,k+1}$ is defined as

$$\Delta\tilde{\mathbf{u}}_{i,k+1} = \begin{bmatrix}\Delta\tilde{u}^T_{i,k+1} & \Delta\tilde{u}^T_{l,k+1} & \Delta\tilde{u}^T_{m,k+1}\end{bmatrix}^T$$

$\widetilde{E}_{i,k+1}$ is defined as

$$\widetilde{E}_{i,k+1} = \widetilde{e}_{i,k+1} \otimes I_N$$

in which $\widetilde{e}_{i,k+1}$ is a 3 by p matrix with all elements zeros except for the $(1, i)$, $(2, l)$,

$(3, m)$ entries to be 1. Note that, the above formulation is in the format of (5.26) and therefore can be solved using ADMM.

After computing $\Delta\tilde{u}_{k+1}$ as above, the optimal input solution can be obtained as follows

$$\vec{u}_{k+1} = \vec{u}_k + \mathbb{G}^{-1}\Delta\tilde{u}_{k+1} \tag{5.36}$$

(or in a local manner). This leads to the following algorithm:

Algorithm 5.2. *On trial $k+1$, the series $\{\Delta\tilde{\mathbf{u}}_{i,k+1}^{q+1}\}_{q\geq 0}$ obtained by iteratively performing the following steps*

$$\begin{aligned}\Delta\tilde{\mathbf{u}}_{i,k+1}^{q+1} = \operatorname{argmin}\Big[J_{i,k+1}(\Delta\tilde{\mathbf{u}}_{i,k+1}^{q+1}) + {\gamma_{i,k+1}^{q}}^{T}(\Delta\tilde{\mathbf{u}}_{i,k+1}^{q+1} - \widetilde{E}_{i,k+1}z_{i,k+1}^{q})\\ + \frac{\rho}{2}\|\Delta\tilde{\mathbf{u}}_{i,k+1}^{q+1} - \widetilde{E}_{i,k+1}z_{i,k+1}^{q}\|_{I_{p_i N}}^{2}\Big]\end{aligned} \tag{5.37}$$

$$z_{i,k+1}^{q+1} = \frac{1}{1+|\mathscr{N}_{i,k+1}|}\sum_{o\in(\mathscr{N}_{i,k+1}\bigcup i)}(\Delta\tilde{\mathbf{u}}_{o,k+1}^{q+1})_i \tag{5.38}$$

$$\gamma_{i,k+1}^{q+1} = \gamma_{i,k+1}^{q} + \rho(\Delta\tilde{\mathbf{u}}_{i,k+1}^{q+1} - \widetilde{E}_{i,k+1}z_{i,k+1}^{q+1}) \tag{5.39}$$

provides a distributed implementation for problem (5.34), i.e.

$$\lim_{q\to\infty} z_{k+1}^{q} = \Delta\tilde{u}_{k+1} \tag{5.40}$$

where $(\Delta\tilde{\mathbf{u}}_{o,k+1})_i$ denotes the element in $\Delta\tilde{\mathbf{u}}_{o,k+1}$ corresponding to $z_{i,k+1}$, $|\mathscr{N}_{i,k+1}|$ is the neighbour's number of i^{th} subsystem on trial $k+1$.

The input updating law (5.25) can therefore be updated locally as follows

$$u_{i,k+1} = u_{i,k} + G_i^{-1}\Delta\tilde{u}_{i,k+1} \tag{5.41}$$

Note that no model information of neighbours is needed when solving problem (5.34) using Algorithm 5.2.

5.3.3 Distributed ILC Algorithm

Using Algorithm 5.2 to implement Algorithm 5.1, a distributed algorithm that solve the high performance consensus tracking problem with switching topologies is proposed, as shown in Algorithm 5.3. Algorithm 5.3 includes of k_{max} ILC trial, and each ILC trial contains q_{max} ADMM iteration as well as one experiment. In each ADMM iteration, each subsystem receives the local information from its neighbours in Step 5 and obtains local plan $\Delta\tilde{\mathbf{u}}_{i,k+1}^{q+1}$ (through calculating the local cost function $J_{i,k+1}(\Delta\tilde{\mathbf{u}}_{i,k+1})$) in Step 6. In Step 7, each subsystem averages corresponding local plan to conclude the global variable, and then uses its local plan and global variable to calculate the local dual

Algorithm 5.3. *Distributed ILC Algorithm for High Performance Consensus Tracking Problem with Switching Topologies*

```
Input: State space matrices $A_i$, $B_i$, $C_i$, desired reference $r$, reference-accessibility
    matrix $D_k$, Laplacian matrix $L_k$, maximum-trial $k_{max}$, maximum-iteration $q_{max}$,
    penalty parameter $\rho$
Output: Optimal input solution $\vec{u}_{k_{max}}$
1:  Initialization: Set $k = 0$
2:  For: $k = 0$ to $k_{max}$
3:    For: $q = 0$ to $q_{max}$
4:      For: $i = 1$ to $p$
5:        Receive data from neighbouring subsystems
6:        Perform $\Delta\tilde{\mathbf{u}}_{i,k+1}^{q+1}$ minimization step (5.37)
7:        Perform $z_{i,k+1}^{q+1}$ minimization step (5.38)
8:        Perform $\gamma_{i,k+1}^{q+1}$ minimization step (5.39)
9:        Send data to neighbouring subsystems
10:     End for
11:   End for
12: Set $u_{i,k+1} = u_{i,k} + G_i^{-1}\Delta\tilde{u}_{i,k+1}$
13: End for
14: Return: Optimal input solution $\vec{u}_{k_{max}}$
```

variable in Step 8. In Step 9, each subsystem sends its local information to neighbours and then waits for next iteration. By performing Step 4 − 11 continuously, all the subsystems 'codetermine' the optimal local plan $\Delta\tilde{u}_{i,k+1}$ and then it would be used in Step 12 when calculating the optimal input solution $\vec{u}_{i,k+1}$.

Remark 5.3. *In theory, ADMM requires infinity iterations to approach the centralised solution, which seems impossible to implement in reality. Fortunately, ADMM has been shown to be very efficient in practice, a small iteration number is sufficient to approach the centralised solution in most of the cases. Later simulation will illustrate this.*

5.4 Numerical Examples

In this section, numerical examples are provided to verify the effectiveness of the proposed ILC algorithm. Consider a heterogeneous over-damping, networked dynamical system contains seven subsystems, with i^{th} subsystem's transfer function defined as

$$G_i(s) = \frac{15s + 10}{s^2 + 6s + \tau_i} \tag{5.42}$$

where $\tau_i = i$, $i = 1, 2, \cdots, 7$.

We assume the trial length is $3s$ and the sampling time T_s is $0.05s$ (with a zero-order hold). The initial condition $x_{i,0}$ and the first trial input $u_{i,0}$ for i^{th} subsystem are set to

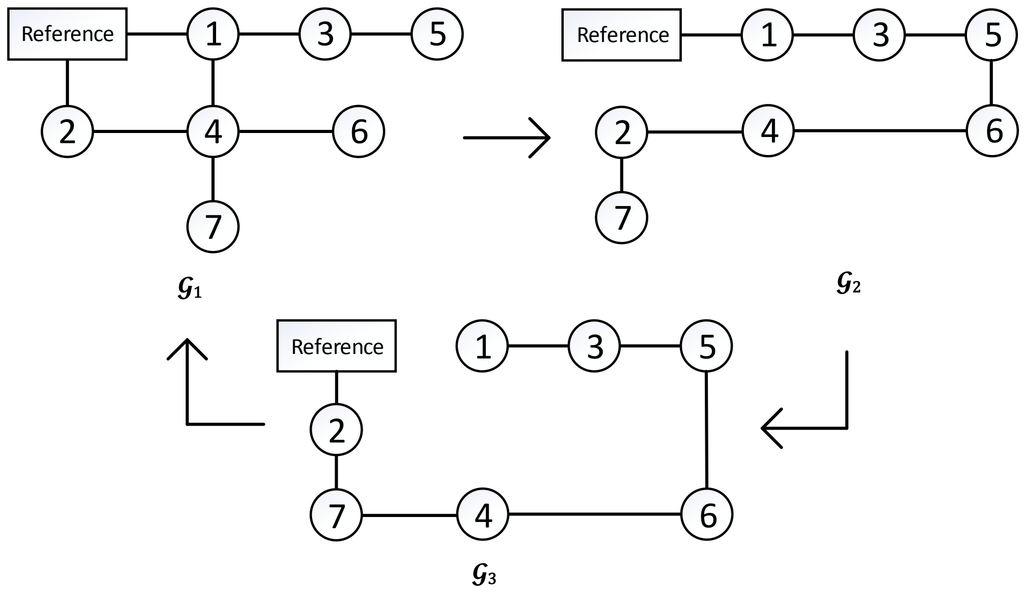

Figure 5.1: The graph structure of numerical example

be 0 and the reference trajectory is defined as

$$r(t) = \begin{cases} sin[\pi(T_s t - 1)] & 20 \leq t \leq 40 \\ 0 & otherwise \end{cases} \tag{5.43}$$

The discussed networked dynamical system's switching topologies are shown in Figure 5.1, with unity weight on each edge. We assume the network topology is $\mathcal{G}_1$ on the first trial, and it will switch follow the order: $\mathcal{G}_1 \rightarrow \mathcal{G}_2 \rightarrow \mathcal{G}_3 \rightarrow \mathcal{G}_1 \rightarrow \cdots$ as the trial number increases. In the following, we will first discuss the proposed algorithm's effectiveness without any model uncertainty.

5.4.1 Example without Model Uncertainty

The convergence speed of Algorithm 5.3 depends on the choice of maximum iteration number q_{max}, hence the effect of different q_{max} choices is first investigated and the convergence performance will be compared with the result generated by the centralised input solution (5.25). We set $\rho = 4$, $\gamma = 1$ and Figure 5.2 shows the evolution of the 'virtual' tracking error norm over 800 trials (i.e., $k_{max} = 800$ in Algorithm 5.3) with $q_{max} = 20, 30, 40, 60$, respectively. From Figure 5.2, it can be seen that both the distributed solution and centralised solution achieve monotonic convergence of the 'virtual' tracking error norm to zero even when most of the subsystems do not have access to the reference and the communication topologies are switching. Furthermore, the convergence speed of Algorithm 5.3 becomes faster as q_{max} increases (which means the distributed implementation provides more accurate solution to the optimisation problem), however, the improvement is marginal: after $q_{max} \geq 60$, the distributed algorithm's convergence

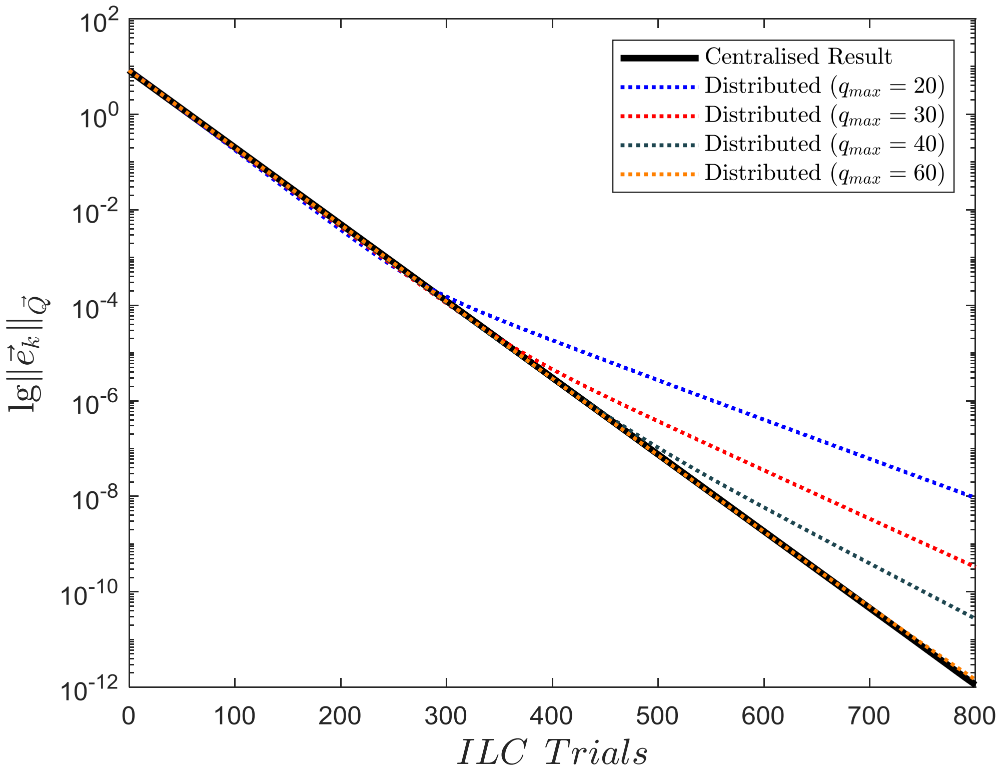

Figure 5.2: Convergence of the 'virtual' error norm over 800 trials (with different q_{max} choices)

performance is almost indistinguishable with the centralised result, which suggests that a small iteration number is sufficient for ADMM to approach the centralised result in practice. For $q_{max} = 60$, the final converging output of different subsystems are shown in Figure 5.3.

Next, the effect of scaling parameter γ is also investigated. We set $\rho = 4$, $q_{max} = 60$, and Figure 5.4 shows how the 'virtual' tracking error norm evolves for $\gamma = 0.3, 0.6, 1, 1.4, 1.7$, respectively. From the figure, it can be concluded that as γ decreases, the convergence performance of Algorithm 5.3 improves, which is consistent with our expectations.

5.4.2 Example with Model Uncertainty

In this section, the robustness of the proposed algorithm is demonstrated. An inaccurate model of i^{th} subsystem is represented using the following transfer function

$$G_{i,model}(s) = \frac{17s + 12}{1.05s^2 + 5.25s + 1.05\tau_i} \tag{5.44}$$

and it is used in the controller design. Figure 5.5 shows the convergence behaviour of the 'virtual' tracking error norm over 600 ILC trials with $\rho = 4$, $q_{max} = 60$ and $\gamma = 0.3, 0.6, 1, 1.4, 1.7$, respectively. It shows that without a highly accurate system

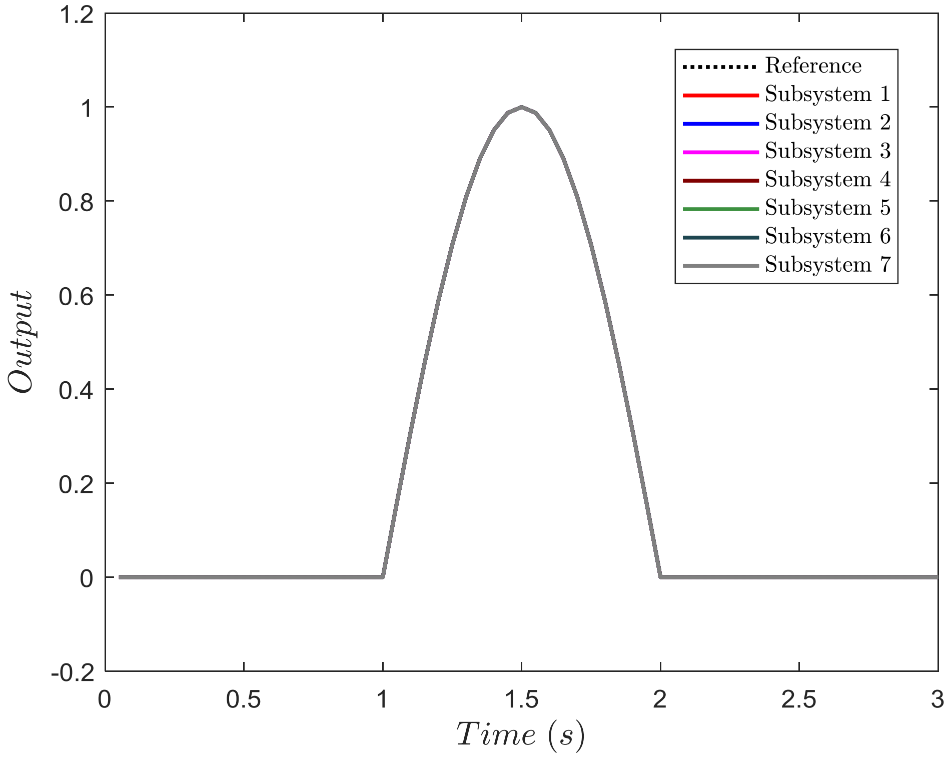

Figure 5.3: Output of different subsystems on 800th trial

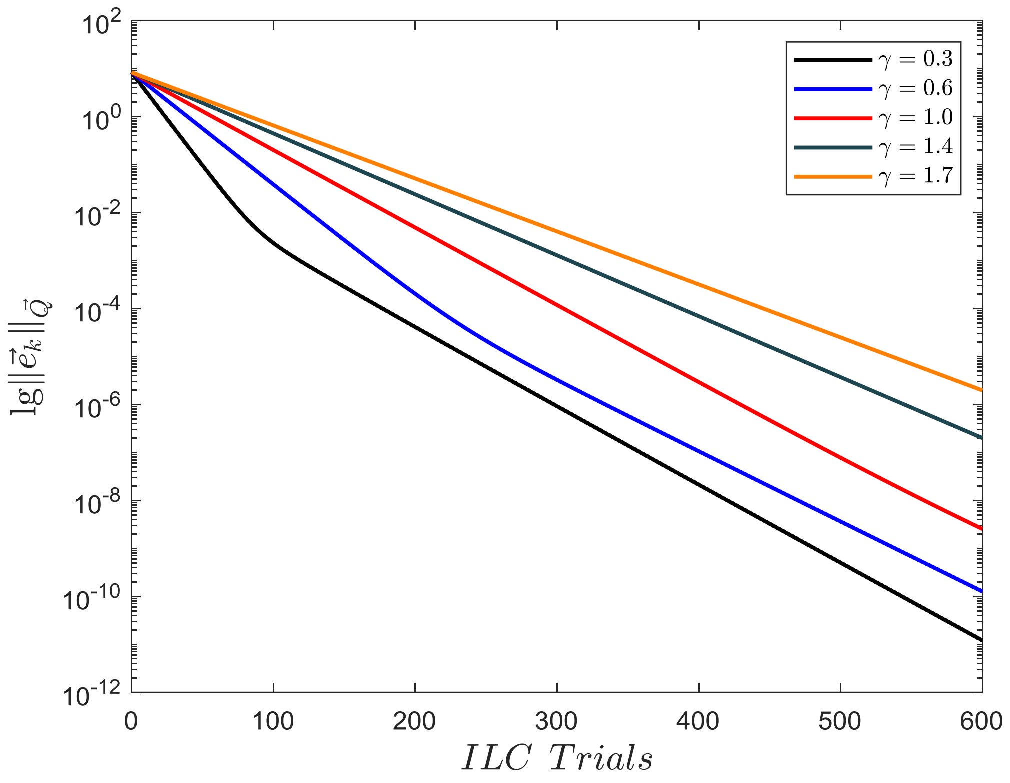

Figure 5.4: Convergence of the ‘virtual’ error norm over 600 trials (with different choices of γ when the model is accurate)

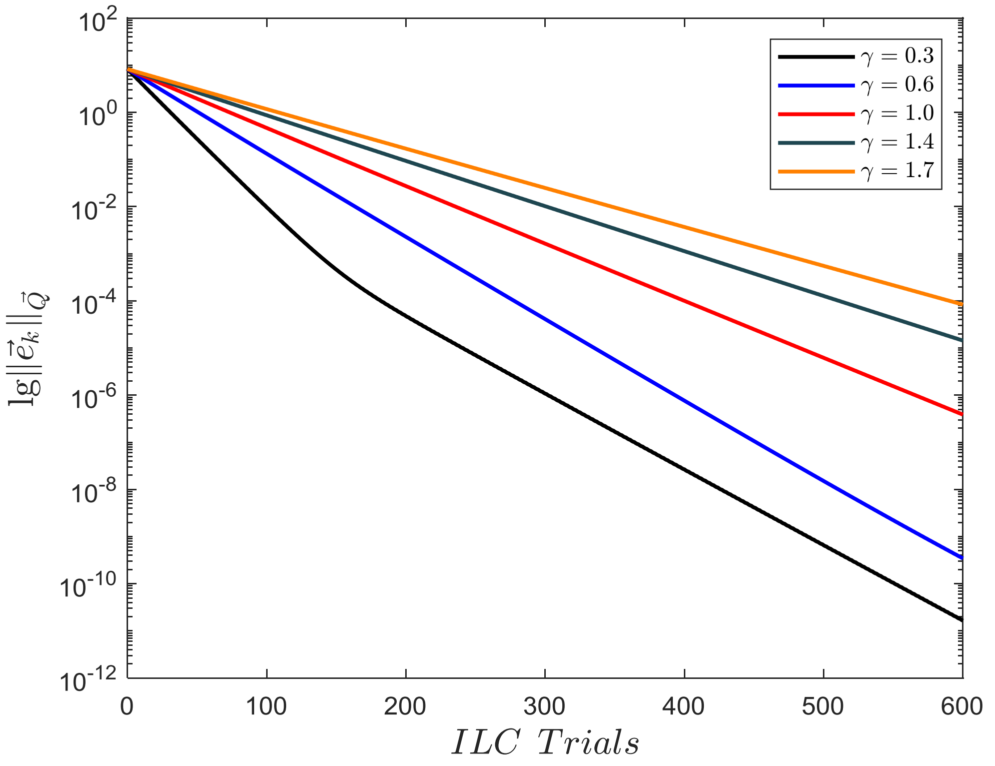

Figure 5.5: Convergence of the 'virtual' error norm over 600 trials (with different choices of γ when the model is inaccurate)

model, both the centralised implementation and distributed implementation can still guarantee monotonic convergence of the 'virtual' tracking error norm to zero even when the communication topologies are switching and only part of subsystems have access to the reference, demonstrating that the proposed ILC algorithm has certain degree of robustness against the model uncertainty. In all the above simulations, the effect of different ρ is also investigated and the result is consistent with the results above, hence the simulation is omitted here for simplicity.

5.5 Summary

In this chapter, we propose a distributed ILC algorithm for high performance consensus tracking problem with switching topologies. By designing a novel performance index, the proposed ILC algorithm has appealing convergence properties that it can achieve monotonic convergence of the 'virtual' tracking error norm to zero for any choice of controller's parameters, which is desired in practice. Considering the ubiquity of large scale networked dynamical systems in practice, we further design a distributed implementation method using the idea of ADMM, allowing the proposed ILC algorithm to be applied to networked dynamical systems with a large number of subsystems and have great scalability. Numerical examples with heterogeneous, switching topologies networked dynamical systems are given to verify the proposed algorithm's effectiveness. Although the robustness properties have not been rigorously proof in this chapter, a similar technology as in Chapter 3 can be applied to analyse the robustness properties. This would be part of future research.

However, the proposed design has an inevitable drawback, i.e., it will meet the high frequency issue and has difficulties to be applied to non-minimum phase networked dynamical systems (since the algorithms involve the inverse of system dynamics in the input updating law). By choosing the weighting matrix $\vec{R} = \gamma \mathbb{G}^T \mathbb{G}$, the proposed design shares some similarities with the inverse-based ILC and may cause the instability for non-minimum phase systems. This issue will be further investigated in the future.

In the next section, we will consider the formation control problem (which is another important design problem in networked systems coordination) and propose a novel distributed ILC algorithm to solve it.

Chapter 6

Distributed NOILC for Formation Control Problem

Formation control is a design problem that requires a group of subsystems work together and precisely form a prescribed formation within a finite time interval. Indeed, many practical applications is designed using the idea of formation control, e.g., cooperative transportation, sensor networks, formation flying (Cao et al., 2013). Benefiting from its broad applications, a number of ILC designs for formation control have been proposed: Proportional (P) type ILC control laws are proposed in Meng and Jia (2014); Meng et al. (2012) for linear networked dynamical systems and most recently for two dimensional (2D) switching graph in Meng and Moore (2016); Proportional-Integral-Derivative (PID) type ILC control laws have been proposed for nonlinear networked dynamical systems in Liu and Jia (2012, 2015); Meng and Moore (2017); Meng et al. (2014a, 2015b); adaptive ILC control laws are proposed in Li and Li (2014a); Li et al. (2018); Li and Li (2014b) for nonlinear networked dynamical systems; a high-order internal model based ILC is developed in Xu et al. (2011); Yang et al. (2017). While the aforementioned results provide some useful designs, there are certain limitations needed to be addressed. A particular aspect is that most existing designs focus on asymptotic formation convergence, while monotonic convergence of the formation error norm, which is desirable in practice, is either not guaranteed, or is only achievable when the system dynamics satisfy certain conditions.

In this chapter, we further develop the previous NOILC design (proposed in Chapter 3) to solve the formation control of networked dynamical systems working in a repetitive manner. By incorporating the formation control requirement explicitly into a performance index which is subsequently optimised, the resulting norm optimal design algorithm not only guarantees monotonic convergence of the formation error norm to zero but also minimizes the control input energy which is a distinct feature of the proposed design. The proposed design can be applied to both homogeneous and heterogeneous networked dynamical systems and can also handle the difficult case of non-minimum phase systems.

In addition, it can be shown that the performance index of the proposed ILC framework can be written into a separable form, and hence it can be implemented in a distributed manner via the alternating direction method of multipliers (ADMM), which has found extensive applications in distribution optimisations (e.g. Boyd et al. (2011); Cheng et al. (2017); Deng and Yin (2016); Zhang et al. (2019a)), allowing the proposed design to be applicable to networked systems with a large number of subsystems. This chapter is based on the work from Chen and Chu (2019b).

The chapter is organized as follows: Section 6.1 formulates the formation control problem of networked dynamical systems; Section 6.2 introduces a NOILC framework for formation control and analyses its convergence properties in detail; Section 6.3 develops a distributed implementation of the proposed NOILC framework using ADMM and gives a method to find the optimal penalty parameter ρ; Section 6.4 presents numerical simulations to demonstrate the performance of the proposed algorithm, and finally Section 6.5 gives a brief conclusion to summary what have been done in this chapter.

6.1 Problem Formulation

In this section, the dynamics of networked dynamical systems are described and the ILC design problem is formulated.

6.1.1 System Dynamics

Consider a networked dynamical system (which can be either homogeneous or heterogeneous) with p subsystems, where the dynamics of i^{th} subsystem ($1 \le i \le p$) is a m-input, ℓ-output discrete time, linear time-invariant (LTI) system described below

$$\begin{aligned} x_{i,k}(t+1) &= A_i x_{i,k}(t) + B_i u_{i,k}(t) \qquad x_{i,k}(0) = x_{i,0} \\ y_{i,k}(t) &= C_i x_{i,k}(t) \end{aligned} \tag{6.1}$$

where k is the trial number; $t \in [0, N]$ represents the time; A_i, B_i, C_i are system matrices with proper dimensions; $u_{i,k}(\cdot) \in \mathbb{R}^m$, $x_{i,k}(\cdot) \in \mathbb{R}^n$, $y_{i,k}(\cdot) \in \mathbb{R}^\ell$ are the input, state and output of subsystem i at trial k respectively, and for i^{th} subsystem, the initial condition remain the same for all the trials. The networked dynamical system is required to repetitively establish the desired formation over the finite interval $[0, N]$. At $t = N+1$, the time t is reset to 0, the subsystem's state to $x_{i,0}$, and the networked dynamical system is required to establish the desired formation from initial state condition again.

Given any subsystems i and j, let $y_{ij,k}(t) = y_{i,k}(t) - y_{j,k}(t)$ denotes the relative position between subsystems i and j. Then the formation control problem can be stated as designing input $u_{i,k}(t)$ to achieve the desired relative formation, i.e.

$$\lim_{k\to\infty} y_{ij,k}(t) = r^*_{ij}(t) \tag{6.2}$$

where $r^*_{ij}(t) = r^*_i(t) - r^*_i(t)$ denotes the desired relative formation between subsystems i and j, in which r^*_i is a virtual reference for agent i defining the desired formation. It is important to point out that r^*_i is neither available to nor is supposed to be tracked by subsystem i — it is the relative formation, i.e. r^*_{ij} that is of interest.

To facilitate subsequent design, we first introduce a 'lifted matrix form' representation of the system model (Hatonen et al., 2004). Note that there is no direct input feed through. The input $u_{i,k}(t)$, output $y_{i,k}(t)$, and virtual reference $r^*_i(t)$ of subsystem i can be defined using the 'lifted form' as follows

$$\begin{aligned} u_{i,k} &= \begin{bmatrix} u_{i,k}(0)^T & u_{i,k}(1)^T & \cdots & u_{i,k}(N-1)^T \end{bmatrix}^T \\ y_{i,k} &= \begin{bmatrix} y_{i,k}(1)^T & y_{i,k}(2)^T & \cdots & y_{i,k}(N)^T \end{bmatrix}^T \\ r^*_i &= \begin{bmatrix} r^*_i(1)^T & r^*_i(2)^T & \cdots & r^*_i(N)^T \end{bmatrix}^T \end{aligned} \tag{6.3}$$

where $u_{i,k} \in \mathcal{U}$, $y_k \in \mathcal{Y}$. $\mathcal{U}$, $\mathcal{Y}$ are real Hilbert space and their inner product and induced norm are defined as

$$\begin{aligned} \langle u_{1,k}, u_{2,k}\rangle_{\mathcal{U}} &= \sum_{t=0}^{N-1} u^T_{1,k}(t) R u_{2,k}(t), \qquad \|u_{i,k}\|_{\mathcal{U}} = \sqrt{\langle u_{i,k}, u_{i,k}\rangle_{\mathcal{U}}} \\ \langle y_{1,k}, y_{2,k}\rangle_{\mathcal{Y}} &= \sum_{t=0}^{N-1} u^T_{1,k}(t) Q u_{2,k}(t), \qquad \|y_{i,k}\|_{\mathcal{Y}} = \sqrt{\langle y_{i,k}, y_{i,k}\rangle_{\mathcal{Y}}} \end{aligned} \tag{6.4}$$

with weighting matrices Q, R are positive definite.

Then, we can rewrite the system model (6.1) into the following form

$$y_{i,k} = G_i u_{i,k} + d_i \tag{6.5}$$

where the system matrix G_i is

$$G_i = \begin{bmatrix} C_iB_i & 0 & \cdots & 0 & 0 \\ C_iA_iB_i & C_iB_i & \ddots & 0 & 0 \\ C_iA_i^2B_i & C_iA_iB_i & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & C_iB_i & 0 \\ C_iA_i^{N-1}B_i & \cdots & \cdots & C_iA_iB_i & C_iB_i \end{bmatrix} \tag{6.6}$$

and $d_i \in \mathbb{R}^N$ is the initial condition's response

$$d_i = \begin{bmatrix} C_iA_ix_{i,0} & C_iA_i^2x_{i,0} & C_iA_i^3x_{i,0} & \cdots & C_iA_i^Nx_{i,0} \end{bmatrix}^T \tag{6.7}$$

Introduce $\vec{u}_k$, $\vec{y}_k$, $\vec{r}^*$, d as the combined vector of all the subsystems' input $u_{i,k}$, output $y_{i,k}$, virtual reference r_i^* and initial condition's response d_i

$$\begin{aligned} \vec{u}_k &= \begin{bmatrix} u_{1,k}^T & u_{2,k}^T & \cdots & u_{p,k}^T \end{bmatrix}^T \\ \vec{y}_k &= \begin{bmatrix} y_{1,k}^T & y_{2,k}^T & \cdots & y_{p,k}^T \end{bmatrix}^T \\ \vec{r}^* &= \begin{bmatrix} r_1^{*T} & r_2^{*T} & \cdots & r_p^{*T} \end{bmatrix}^T \\ d &= \begin{bmatrix} d_1^T & d_2^T & \cdots & d_p^T \end{bmatrix}^T. \end{aligned} \tag{6.8}$$

Then, the 'lifted matrix form' representation of the whole networked dynamical system can be rewritten as

$$\vec{y}_k = \mathbb{G}\vec{u}_k + d \tag{6.9}$$

where $\mathbb{G} = \text{diag}\,(G_1, G_2, G_3, \cdots, G_{p-1}, G_p)$.

Using 'lifted matrix form' representation, the formation control requirement (6.2) can be written equivalently as

$$\lim_{k\to\infty} y_{i,k} - y_{j,k} = r_i^* - r_j^* \tag{6.10}$$

In this chapter, we assume that there exists a solution to the problem, i.e. there exists an input u_i^s such that

$$\lim_{k\to\infty} y_{i,k} - y_{j,k} = (G_iu_i^s + d_i) - (G_ju_j^s + d_j) = r_i^* - r_j^* \tag{6.11}$$

To guarantee the later results is applied to MIMO networked dynamical systems, the following assumption is required:

Assumption 6.1. *The global virtual vector $\vec{r}^* \in Rg(\boldsymbol{L}\mathbb{G})$, where $\boldsymbol{L} = L \otimes I_{N\ell}$, $\otimes$ is the Kronecker product, and $I_{N\ell}$ denotes $N\ell \times N\ell$ identity matrix.*

6.1.2 Network Topology

In this chapter, the topology of the networked dynamical system is represented using an undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V} = \{1, 2, 3, \cdots, p\}$ denotes the node set and $\mathscr{E} \subset \mathscr{V} \times \mathscr{V}$ is the edges set. The neighbour set of subsystem i is represented as $\mathcal{N}_i := \{j : (i, j) \in \mathscr{E}\}$.

Laplacian matrix $L = \{l_{ij}\}$ denotes the topology relationship between different subsystems, and it is a real positive semi-definite, symmetric matrix with element l_{ij}

$$l_{ij} = \begin{cases} -W_{ij} & if \quad j \in \mathscr{N}_i \\ \sum_{j\in\mathscr{N}_i} W_{ij} & if \quad j = i \\ 0 & otherwise \end{cases} \tag{6.12}$$

For the desired formation control to be achievable, the following assumption is standard:

Assumption 6.2. *The graph $\mathscr{G}$ is connected, i.e. there exists at least one path from one vertex to another vertex.*

6.1.3 Control Design Objective

The ILC design problem for formation control of networked dynamical systems can be stated as designing an input updating law in the following form

$$\vec{u}_{k+1} = f(\vec{u}_k, \vec{y}_k) \tag{6.13}$$

such that subsystem's output $y_{i,k}$ $(i = 1, 2, \cdots, p)$ can establish the desired formation as $k \to \infty$, i.e.

$$\lim_{k\to\infty} y_{i,k} - y_{j,k} = r_i^* - r_j^* \tag{6.14}$$

It should be noted that, the solution to the above problem is not unique. In fact as will be seen later, there are infinite number of solutions. Among these solutions, there exists one with the minimum control input energy. To the best of our knowledge, none of the existing ILC algorithms for formation control problem can guarantee the minimum control input energy solution. By contrast, the NOILC design proposed in next section will not only guarantee the achievement of the desired formation, but also converge to the minimum control energy solution.

6.2 Norm Optimal ILC Algorithm for Formation Control of Networked Dynamical Systems

In this section, we proposes a novel NOILC framework to achieve the high-precision formation control of networked dynamical systems. The algorithm's convergence properties are analysed rigorously.

6.2.1 Algorithm Description

Algorithm 6.1. *For any initial input choice $\vec{u}_0$, the control input sequence $\{\vec{u}_{k+1}\}_{k>0}$, defined as follows*

$$\vec{u}_{k+1} = \arg\min\{J_{k+1}(\vec{u}_{k+1})\} \tag{6.15}$$

iteratively solve the formation control problem, i.e.

$$\lim_{k\to\infty} y_{i,k} - y_{j,k} = r_i^* - r_j^* \tag{6.16}$$

where the cost function $J_{k+1}(\vec{u}_{k+1})$ is defined as

$$J_{k+1}(\vec{u}_{k+1}) = \|\vec{e}_{k+1}\|_Q^2 + \|\vec{u}_{k+1} - \vec{u}_k\|_R^2 \tag{6.17}$$

where $\vec{e}_{k+1} = \boldsymbol{L}(\vec{r}^* - \vec{y}_{k+1})$ is the formation error.

Remark 6.1. *For the formation control problem, the term $(\boldsymbol{L}+\boldsymbol{D})$ (appears in consensus tracking problem as in Chapter 3) is replaced by $\boldsymbol{L}$, since we consider the case that not reference trajectory is supported to be tracked by the subsystems. Instead, the subsystems need to discuss a strategy by themselves to form the desired formation.*

Remark 6.2. *By directly solving problem (6.15) (by calculating the stationary point of the performance index), Algorithm 6.1 can be implemented in a centralised manner as*

$$\vec{u}_{k+1} = \vec{u}_k + (\mathbb{G}^*\boldsymbol{L}^T\boldsymbol{L}\mathbb{G} + I_{pN\ell})^{-1}\mathbb{G}^*\boldsymbol{L}^T\boldsymbol{L}(\vec{r}^* - \vec{y}_k) \tag{6.18}$$

where $\mathbb{G}^* = \mathbb{R}^{-1}\mathbb{G}^T\mathbb{Q}$ is the adjoint operator of $\mathbb{G}$, $\mathbb{Q} = \text{diag}\,(Q, Q, \cdots, Q)$ and $\mathbb{R} = \text{diag}\,(R, R, \cdots, R)$.

However, solving Equation (6.18) requires huge computational load for large scale networked systems (i.e., when p is large), which is often unrealistic and impractical.

6.2.2 Convergence Properties

Algorithm 6.1 achieves monotonic convergence in the formation error norm as shown in following theorem.

Theorem 6.1. *For any initial input choice $\vec{u}_0$, Algorithm 6.1 guarantees that the formation error norm $\|\vec{e}_{k+1}\|_{\vec{Q}}$ converges monotonically to zero, that is*

$$\|\vec{e}_{k+1}\|_{\vec{Q}} \le \|\vec{e}_k\|_{\vec{Q}}, \quad \lim_{k\to\infty} \vec{e}_k = 0 \tag{6.19}$$

As a result, the control system achieve desired formation as $k \to \infty$, i.e.

$$\lim_{k\to\infty} y_{i,k} - y_{j,k} = r_i^* - r_j^* \tag{6.20}$$

Proof. For the optimisation problem (6.15), it is clear that choosing $\vec{u}_{k+1} = \vec{u}_k$ produces a suboptimal solution. Therefore

$$J_{k+1}(\vec{u}_{k+1}) \leq J_{k+1}(\vec{u}_k) \tag{6.21}$$

from which

$$\|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}} + \|\vec{e}_{k+1}\|^2_{\vec{Q}} \leq \|\vec{e}_k\|^2_{\vec{Q}} \tag{6.22}$$

hence

$$\|\vec{e}_{k+1}\|^2_{\vec{Q}} \leq \|\vec{e}_k\|^2_{\vec{Q}} \tag{6.23}$$

proving the monotonic convergence of the formation error norm.

To show that $\vec{e}_k \to 0$, note solving (6.15) gives

$$\vec{u}_{k+1} = \vec{u}_k + (\mathbb{G}^* \boldsymbol{L}^T \boldsymbol{L} \mathbb{G} + I_{pN\ell})^{-1} \mathbb{G}^* \boldsymbol{L}^T \vec{e}_k \tag{6.24}$$

from the proof of monotone convergence theorem. We have

$$\lim_{k \to \infty} \|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}} = 0 \tag{6.25}$$

hence

$$(\mathbb{G}^* \boldsymbol{L}^T \boldsymbol{L} \mathbb{G} + I_{pN\ell})^{-1} \mathbb{G}^* \boldsymbol{L}^T \vec{e}_k \to 0 \tag{6.26}$$

Note that matrix $\mathbb{G}^* \boldsymbol{L}^T \boldsymbol{L} \mathbb{G} + I_{pN\ell}$ is invertible, therefore

$$\mathbb{G}^* \boldsymbol{L}^T \vec{e}_k \to 0 \tag{6.27}$$

Note that $\vec{e}_k = \boldsymbol{L}(\vec{r}^* - \vec{y}_k)$. Denote a solution to the problem as $\vec{u}^s$ (see (6.11)). We then have

$$\boldsymbol{L}(\mathbb{G}\vec{u}^s + d) = \boldsymbol{L}\vec{r}^* \tag{6.28}$$

using which we have

$$\vec{e}_k = \boldsymbol{L}(\vec{r}^* - \vec{y}_k) = \boldsymbol{L}\mathbb{G}(\vec{u}^s - \vec{u}_k) \tag{6.29}$$

Substituting this into (6.27) gives

$$\mathbb{G}^* \boldsymbol{L}^T \boldsymbol{L} \mathbb{G}(\vec{u}^s - \vec{u}_k) \to 0 \tag{6.30}$$

therefore

$$(\vec{u}^s - \vec{u}_k)^T \mathbb{G}^* \boldsymbol{L}^T \boldsymbol{L} \mathbb{G}(\vec{u}^s - \vec{u}_k) \to 0 \tag{6.31}$$

It then follows that

$$\boldsymbol{L}\mathbb{G}(\vec{u}^s - \vec{u}_k) \to 0 \tag{6.32}$$

therefore

$$\boldsymbol{L}(\mathbb{G}\vec{u}^s - \mathbb{G}\vec{u}_k) = \boldsymbol{L}(\vec{r}^* - \vec{y}_k) = \vec{e}_k \to 0 \tag{6.33}$$

To show the formation control has been achieved, i.e.

$$\lim_{k\to\infty} y_{i,k} - y_{j,k} = r_i^* - r_j^* \tag{6.34}$$

Note that $\boldsymbol{L}(\vec{r}^* - \vec{y}_k) \to 0$ indicates $\vec{r}^* - \vec{y}_k \to \mathbb{1} \otimes c$ (as L is positive semi-definite with eigenvalues $\{0, \cdots\}$ and the eigenvector corresponding to 0 eigenvalue is $\mathbb{1}$, as shown in Lemma 3.1), where $c \in \mathbb{R}^{N\ell}$. We then have

$$y_{i,k} - y_{j,k} \to (r_i^* - c) - (r_j^* - c) = r_i^* - r_j^* \tag{6.35}$$

as desired in (6.34). That completes the proof. $\square$

The above theorem shows that Algorithm 6.1 can achieve the desired formation control of networked dynamical systems with the desirable property of monotonic reduction in formation error norm. Furthermore, the converged input has following property:

Theorem 6.2. *For any initial input choice $\vec{u}_0$, the input generated by Algorithm 6.1 converges as follows*

$$\lim_{k\to\infty} \vec{u}_k = \vec{u}^* \tag{6.36}$$

where $\vec{u}^*$ is the solution of the following optimisation problem

$$\begin{aligned} &\text{minimize} && \|\vec{u} - \vec{u}_0\|^2_{\vec{R}} \\ &\text{subject to} && \boldsymbol{L}(\mathbb{G}\vec{u} + d) = \boldsymbol{L}\vec{r}^* \end{aligned} \tag{6.37}$$

As a direct consequence, if the initial input is chosen as $\vec{u}_0 = 0$, Algorithm 6.1 converges to the minimum control energy solution that produces the desired formation. i.e.

$$\begin{aligned} &\text{minimize} && \|\vec{u}\|^2_{\vec{R}} \\ &\text{subject to} && \boldsymbol{L}(\mathbb{G}\vec{u} + d) = \boldsymbol{L}\vec{r}^* \end{aligned} \tag{6.38}$$

Proof. For the optimisation problem

$$\begin{aligned} &\text{minimize} && \|\vec{u} - \vec{u}_0\|^2_{\vec{R}} \\ &\text{subject to} && \boldsymbol{L}(\mathbb{G}\vec{u} + d) = \boldsymbol{L}\vec{r}^* \end{aligned} \tag{6.39}$$

the Lagrangian

$$L(u, \lambda) = \|\vec{u} - \vec{u}_0\|^2_{\vec{R}} + 2(\lambda, \boldsymbol{L}(\mathbb{G}\vec{u} + d - \vec{r}^*)) \tag{6.40}$$

Note that the control updating law for (6.17) is

$$\begin{aligned}\vec{u}_{k+1} &= \vec{u}_k + (\mathbb{G}^*\boldsymbol{L}^T\boldsymbol{L}\mathbb{G} + I_{pN\ell})^{-1}\mathbb{G}^*\boldsymbol{L}^T\vec{e}_k \\ &= \vec{u}_k + \mathbb{G}^*\boldsymbol{L}^T(\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1}\vec{e}_k\end{aligned} \tag{6.41}$$

Matrix $(\mathbb{G}^*\boldsymbol{L}^T\boldsymbol{L}\mathbb{G} + I_{pN\ell})^{-1}$ is positive definite with eigenvalues $0 < \lambda_i \leq 1$, define the space spanned by the eigenvector corresponding to unitary eigenvalues as $\mathcal{E}^\perp$, then we have

$$\begin{aligned}(\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1}|_{\mathcal{E}} < 1 \\ (\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1}|_{\mathcal{E}^\perp} = 1\end{aligned} \tag{6.42}$$

where $T|_B$ represents the restriction of operator T on the subspace B.

For any initial input choice $\vec{u}_0$, denote the corresponding formation error as $\vec{e}_0$, its decomposition on $\mathcal{E}$ and $\mathcal{E}^\perp$ is

$$\vec{e}_0 = \vec{e}_0^{\mathcal{E}} + \vec{e}_0^{\mathcal{E}^\perp} \tag{6.43}$$

Note that by definition

$$\vec{e}_0^{\mathcal{E}^\perp} = (\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1}\vec{e}_0^{\mathcal{E}^\perp} \tag{6.44}$$

therefore

$$\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T\vec{e}_0^{\mathcal{E}^\perp} = 0 \tag{6.45}$$

This further implies that

$$\mathbb{G}^*\boldsymbol{L}^T\vec{e}_0^{\mathcal{E}^\perp} = 0 \tag{6.46}$$

From the proof of Theorem 6.1 yields

$$\vec{e}_0^{\mathcal{E}^\perp} = 0 \tag{6.47}$$

then Equation (6.43) becomes

$$\vec{e}_0 = \vec{e}_0^{\mathcal{E}} \tag{6.48}$$

We then have

$$\vec{u}_{k+1} - \vec{u}_0 = \mathbb{G}^*\boldsymbol{L}^T(\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1}\sum_{j=0}^{k}(\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-j}\vec{e}_0 \tag{6.49}$$

therefore

$$\vec{u}_{k+1} - \vec{u}_0 = \mathbb{G}^*\boldsymbol{L}^T(\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1}\sum_{j=0}^{k}(\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-j}\vec{e}_0^{\mathcal{E}} \tag{6.50}$$

Note that $\vec{e}_0^{\mathcal{E}} \in \mathcal{E}$, we have

$$\vec{u}_\infty - \vec{u}_0 = \mathbb{G}^* \boldsymbol{L}^T (\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1} \left[I_{pN\ell} - (\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1}|_{\mathcal{E}} \right]^{-1} \vec{e}_0^{\mathcal{E}} \tag{6.51}$$

therefore

$$\vec{u}_\infty - \vec{u}_0 = \mathbb{G}^* \boldsymbol{L}^T (\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1} \left[I_{pN\ell} - (\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1}|_{\mathcal{E}} \right]^{-1} \vec{e}_0^{\mathcal{E}} \tag{6.52}$$

Set $\lambda = \left[I_{pN\ell} - (\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1} (\boldsymbol{L}\mathbb{G}\mathbb{G}^*\boldsymbol{L}^T + I_{pN\ell})^{-1}|_{\mathcal{E}} \right]^{-1} \vec{e}_0^{\mathcal{E}} + \vec{e}_0^{\mathcal{E}^\perp}$, then

$$\vec{u}_\infty - \vec{u}_0 = \mathbb{G}^* \boldsymbol{L}^T \lambda \tag{6.53}$$

Equation (6.53) shows that (u_∞,λ) is a stationary point of the Lagrangian. Also, note that the optimisation problem is strictly convex. This shows u_∞ is the unique solution to the optimisation problem. That completes the proof. □

Theorems 6.1 and 6.2 demonstrate that the proposed Algorithm 6.1 can not only guarantee monotonic convergence in the formation error norm to zero, but also obtain the minimum control input energy consumption, which is appealing in practice. Numerical simulations will be provided in later section to demonstrate these properties.

6.3 Distributed Implementation of the Proposed Design

As mentioned earlier, the centralised implementation (6.18) will face significant difficulties for large scale networked systems because of the huge computational complexity. To address this problem, this section introduces a distributed implementation of Algorithm 6.1 for large scale networked dynamical systems using ADMM. In the following, we will first review the idea of 'consensus' formulation in ADMM.

6.3.1 The Alternating Direction Method of Multipliers

ADMM is a powerful tool for solving distributed optimisation problems. To illustrate the method, we consider the following optimisation problem

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{p} f_i(x_i) \\ &\text{subject to} \quad x_i - \widetilde{E}_i z = 0, \quad i = 1, \cdots, p \end{aligned} \tag{6.54}$$

where $x_i \in \mathbb{R}^{p_i}$ is the local input decision variable consisting of the component in the global variable $z \in \mathbb{R}^p$ and $\widetilde{E}_i$ is a matrix that selects the components from z that match the local variable x_i.

The augmented Lagrangian for (6.54) is shown as

$$L_\rho(x,z,\gamma) = \sum_{i=1}^{p} L_{\rho i}(x_i,z,\gamma_i) = \sum_{i=1}^{p}\left[f_i(x_i) + \gamma_i^T(x_i - \widetilde{E}_i z) + \frac{\rho}{2}\|x_i - \widetilde{E}_i z\|^2\right] \tag{6.55}$$

and ADMM solves the optimisation problem (6.54) by iteratively performing the following three steps

$$x_i^{q+1} = \arg\min L_{\rho i}(x_i, z^q, \gamma_i^q) \tag{6.56}$$
$$z^{q+1} = \arg\min L_\rho(x^{q+1}, z, \gamma^q) \tag{6.57}$$
$$\gamma_i^{q+1} = \gamma_i^q + \rho(x_i^{q+1} - \widetilde{E}_i z^{q+1}) \tag{6.58}$$

ADMM has appealing convergence properties and it is widely used for solving the distributed optimisation problems in different areas, please refer to Boyd et al. (2011) for more details.

6.3.2 Distributed Implementation of Algorithm 6.1

To use ADMM to implement Algorithm 6.1, its cost function $J_{k+1}(\vec{u}_{k+1})$ at trial $k+1$ is rewritten into the following separable form

$$J_{k+1}(\vec{u}_{k+1}) = \sum_{i=1}^{p} J_{i,k+1}(\vec{u}_{i,k+1}) \tag{6.59}$$

in which the local cost function $J_{i,k+1}(\vec{u}_{i,k+1})$ of subsystem i is denoted as

$$J_{i,k+1}(\vec{u}_{i,k+1}) = \|\sum_{j\in\mathcal{N}_i} W_{ij}[(G_i u_{i,k+1}+d_i)-(G_j u_{j,k+1}+d_j)-(r_i^*-r_j^*)]\|_Q^2+\|u_{i,k+1}-u_{i,k}\|_R^2 \tag{6.60}$$

where $\vec{u}_{i,k+1}$ denotes the local input plan. As an example, for $\mathcal{N}_i = \{g,h\}$, $\vec{u}_{i,k+1}$ is represented as

$$\vec{u}_{i,k+1} = [u_{i,k+1}^T \quad u_{g,k+1}^T \quad u_{h,k+1}^T]^T \tag{6.61}$$

Using the 'consensus' formulation in ADMM to solve the above problem at trial $k+1$, we have the following algorithm:

Algorithm 6.2. *At trial $k+1$, the input sequence $\{\vec{u}_{i,k+1}^{q+1}\}_{q>0}$ generated by the following steps:*

$$\vec{u}_{i,k+1}^{q+1} = \arg\min\left[\frac{1}{2}J_{i,k+1}(\vec{u}_{i,k+1}^{q+1}) + \frac{\rho}{2}\|\vec{u}_{i,k+1}^{q+1} - \vec{E}_i z_{k+1}^q\|^2 + \gamma_{i,k+1}^{q\ T}(\vec{u}_{i,k+1}^{q+1} - \vec{E}_i z_{k+1}^q)\right] \tag{6.62}$$

$$z_{i,k+1}^{q+1} = \frac{1}{1+|\mathcal{N}_i|}\sum_{o\in(\mathcal{N}_i\bigcup i)} (\vec{u}_{o,k+1}^{q+1})_i \tag{6.63}$$

Algorithm 6.3. *Distributed NOILC Algorithm for Formation Control of Networked Dynamical Systems*

```
Input: Virtual reference $r_i^*$; state space matrices $A_i$, $B_i$, $C_i$; Laplacian matrix $L$;
    penalty parameter $\rho$; maximum ADMM iterations $q_{max}$; formation error norm
    tolerance $\sigma$
Output: Subsystem's optimal input $u_i^*$
1:  Initialization: Initialize the ILC trial index $k$
2:  Repeat:
3:      Set $k = k+1$, $q = 0$
4:      Repeat:
5:          Set $q = q+1$
6:          $\vec{u}_{i,k+1}^{q+1}$ minimization using updating law (6.62)
7:          $z_{i,k+1}^{q+1}$ minimization using updating law (6.63)
8:          $\gamma_{i,k+1}^{q+1}$ minimization using updating law (6.64)
9:      until  $q = q_{max}$
10: Convert local input plan $\vec{u}_{i,k+1}^{q_{max}}$ into $u_{i,k+1}$
11: Implement $u_{i,k+1}$ and record $y_{i,k+1}$ for all agents
11: until  $\|\vec{e}_k\| \leq \sigma$
12: Return: Subsystem's optimal input $u_i^*$
```

$$\gamma_{i,k+1}^{q+1} = \gamma_{i,k+1}^{q} + \rho(\vec{u}_{i,k+1}^{q+1} - \vec{E}_i z_{k+1}^{q+1}) \tag{6.64}$$

provides a distributed realization for (6.15), i.e.

$$\lim_{q\to\infty} z_{k+1}^{q} = \arg\min \{J_{k+1}(\vec{u}_{k+1})\} \tag{6.65}$$

where $z_{i,k+1}^{q+1}$ is the i^{th} element of global variable z_{k+1} at ADMM iteration $q+1$, $(\vec{u}_{o,k+1}^{q+1})_i$ denotes the corresponding element in each subsystem's local plan $\vec{u}_{o,k+1}$ related to subsystem i.

It is worth pointing out that, the input updating law (6.62) in distributed Algorithm 6.2 can be implemented in both feedback plus feedforward implementation and matrix form implementation. The details are similar with Propositions 3.1 and 3.2, hence they are omitted here.

6.3.3 Distributed NOILC Algorithm

Using the distributed realization Algorithm 6.2 to implement the norm optimal framework in Algorithm 6.1, we get the distributed NOILC algorithm as shown in Algorithm 6.3. The ADMM realization steps are embedded in step 4–9: for each iteration, each agent first uses the corresponding global element (stored locally) and the (local element of) dual variable to calculate the local input plan $\vec{u}_{i,k+1}^{q+1}$; then, new global value $z_{i,k+1}^{q+1}$ is generated by averaging the corresponding local input plan; finally, new dual variable

$\gamma_{i,k+1}^{q+1}$ is updated using the results in previous two steps. In theory, ADMM will converge to the optimal result as the iteration number q_{max} increases to infinity. However, ADMM is usually very efficient in practice, and a small q_{max} is sufficient to approximate the optimal solution in most of the cases. Later simulations will demonstrate this.

6.3.4 Penalty Parameter Selection

For ADMM, its convergence speed depends on the choice of penalty parameter ρ. Using the result in (Ghadimi et al., 2015; Teixeira et al., 2013, 2016) to Algorithm 6.3, a method to find the optimal penalty parameter is shown in the following proposition:

Proposition 6.1. *The optimal penalty parameter ρ^* for the proposed algorithm is*

$$\rho^* = \begin{cases} \dfrac{1-\sqrt{1-\lambda_{n-s}^2}}{\lambda_{n-s}^2 - 1 + \sqrt{1-\lambda_{n-s}^2}} & \lambda_{n-s} > 0 \\ 1 & \lambda_{n-s} \le 0 \end{cases} \tag{6.66}$$

and the corresponding convergence factor is

$$\phi^* = |\phi_{2n-s}| = \begin{cases} \dfrac{1}{2}\left(1 + \dfrac{\lambda_{n-s}}{1+\sqrt{1-\lambda_{n-s}^2}}\right) & \lambda_{n-s} > 0 \\ \dfrac{1}{2} & \lambda_{n-s} \le 0 \end{cases} \tag{6.67}$$

where λ_i is the i^{th} generalized eigenvalue of $(M^T[2\bar{F}(\bar{F}^T\bar{F})^{-1}\bar{F}^T - I_{pN\ell}]M, M^TM)$, order as $\lambda_n \ge \cdots \ge \lambda_i \ge \lambda_1$, matrix $M = (\mathbb{G}^*\boldsymbol{L}^T\boldsymbol{L}\mathbb{G} + I_{pN\ell})^{\frac{1}{2}}$ and $\bar{F} = -M\widetilde{E}$. $\mathcal{R}(A) \triangleq \{y \in \boldsymbol{R}^n | y = Ax, x \in \mathcal{R}^m\}$ is the range-space of matrix $A \in \mathcal{R}^{n\times m}$, $s = dim(\mathcal{R}(\bar{F}))$ and the convergence factor is defined as (Bertsekas and Tsitsiklis, 1997)

$$\phi^* \triangleq \sup_{\vec{u}_{k+1}^q \neq \vec{u}_{k+1}^*} \frac{\|\vec{u}_{k+1}^{q+1} - \vec{u}_{k+1}^*\|}{\|\vec{u}_{k+1}^q - \vec{u}_{k+1}^*\|} \tag{6.68}$$

Proof. Following a similar proof of Proposition 3.3, we can obtain Proposition 6.1. □

The main steps to find the optimal penalty parameter could be described in below steps

1 Set $M = (\mathbb{G}^*\boldsymbol{L}^T\boldsymbol{L}\mathbb{G} + I_{pN\ell})^{\frac{1}{2}}$ and calculate $\bar{F} = -M\widetilde{E}$;

2 Find the generalized eigenvalue of $(M^T[2\bar{F}(\bar{F}^T\bar{F})^{-1}\bar{F}^T - I_{pN\ell}]M, M^TM)$;

3 Find $s = dim(\mathcal{R}(\bar{F}))$ and calculate λ_{n-s};

4 Use Proposition 6.1 to find the optimal penalty parameter ρ^* and corresponding convergence factor ϕ^* ;

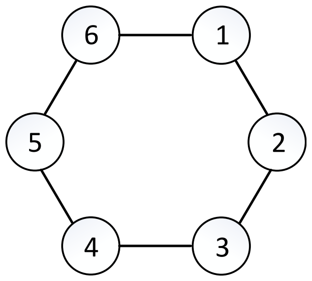

Figure 6.1: The network topology of numerical example

6.4 Numerical Example

In this section, a numerical example is provided to verify the proposed algorithm's performance. Consider a heterogeneous networked dynamical system with six ($p = 6$) subsystems, and the system dynamics of each agent is a non-minimum phase system with non-minimum phase zeros ($z_1 = 0.1$, $z_2 = 0.2$). System state space matrices are defined as follow

$$A_i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -5\tau_i - 5 & -1.5\tau_i - 3 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -5\tau_i - 10 & -1.5\tau_i - 1 \end{bmatrix}$$

$$B_i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T \qquad C_i = \begin{bmatrix} -1 & 10 & 0 & 0 \\ 0 & 0 & -2 & 10 \end{bmatrix}$$

where $\tau_i = i$, $i = 1, 2, \cdots, 6$.

The system is sampled using a sampling time of $0.1s$ and a zero order hold. The trial length is $1s$ and Figure 6.1 shows the network topology of the networked dynamical system. Assume the initial condition and first trial's input of all agents are zero. The desired formation is defined by the virtual reference as follows (i.e. forming a hexagon that both rotates counterclockwise and shrinks its size along the time from $t = 0$ to $t = 1$)

$$r_1^*(t) = R(t)\begin{bmatrix} 1 \\ 0 \end{bmatrix} e^{-0.5t} \qquad r_2^*(t) = R(t)\begin{bmatrix} \frac{1}{2} \\ -\frac{\sqrt{3}}{2} \end{bmatrix} e^{-0.5t}$$

$$r_3^*(t) = R(t)\begin{bmatrix} -\frac{1}{2} \\ -\frac{\sqrt{3}}{2} \end{bmatrix} e^{-0.5t} \qquad r_4^*(t) = R(t)\begin{bmatrix} -1 \\ 0 \end{bmatrix} e^{-0.5t}$$

$$r_5^*(t) = R(t)\begin{bmatrix} -\frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix} e^{-0.5t} \qquad r_6^*(t) = R(t)\begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix} e^{-0.5t}$$

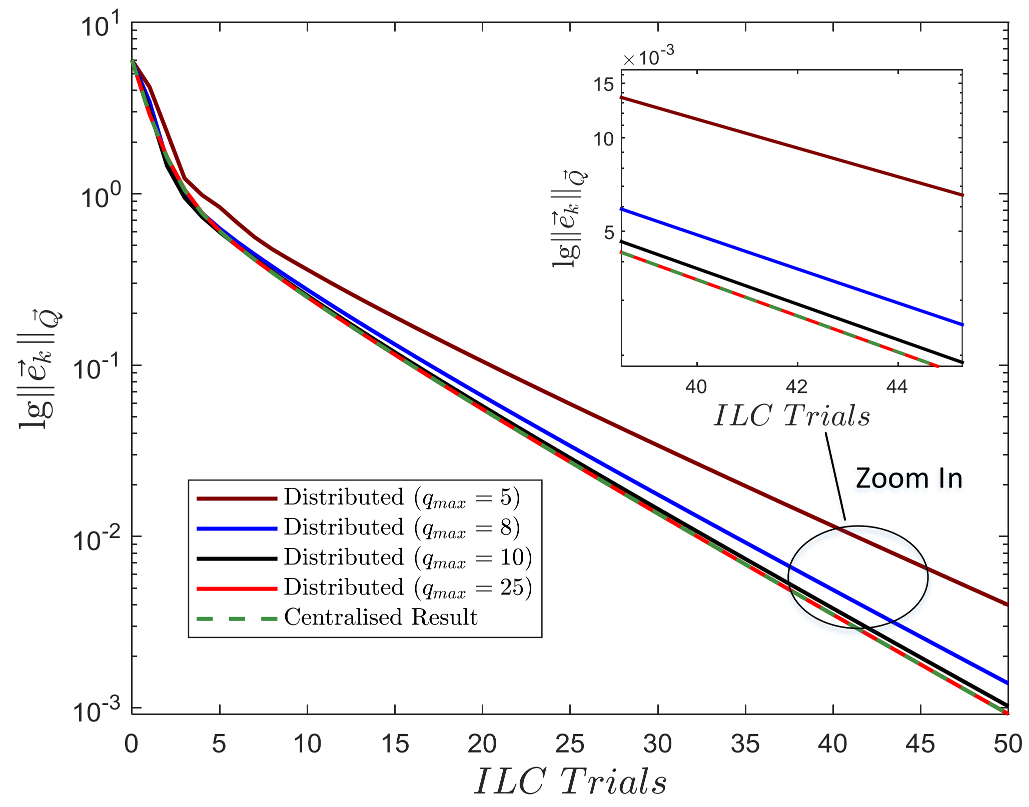

Figure 6.2: Convergence of the formation error norm (different q_{max})

where $R(t)$ is a rotation matrix

$$R(t) = \begin{bmatrix} cos(2\pi t) & -sin(2\pi t) \\ sin(2\pi t) & cos(2\pi t) \end{bmatrix}$$

We simulate the system performance using both centralised implementation (6.18) and the distributed implementation (Algorithm 6.3) over 50 trials with zero initial input choice for all agents. In Algorithm 6.3, the penalty parameter ρ is set to be 1; the scalar weighting $Q = R = 1$; the maximum ADMM iteration q_{max} is set to be $5, 8, 10, 25$, respectively. Figure 6.2 shows the formation error norm over 50 ILC trials. It can been seen that as the maximum ADMM iteration number q_{max} increases, the performance using distributed implementation improves and converges to the (optimal) centralised solution.However, the improvement is marginal – for $q_{max} \geq 25$, its performance is almost identical to the optimal centralised solution, suggesting when using the distributed Algorithm 6.3, a small q_{max} can be used in practice with much less computational demand. Furthermore, for both centralised and distributed implementations, the formation error norm monotonically converges to zero which verifies Theorem 6.1.

Figure 6.3 shows the output of all the subsystems at 50^{th} ILC trial, demonstrating that the desired formation has been achieved. Figure 6.4 shows the total input energy cost of the system. Clearly, the total input energy consumption converges to the minimum

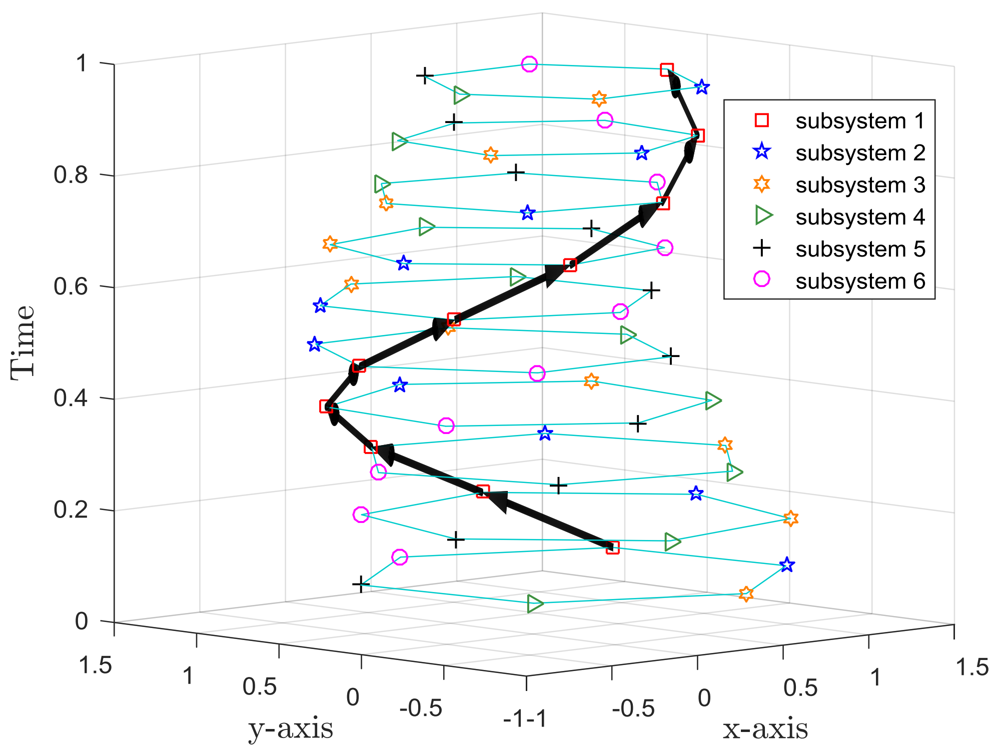

Figure 6.3: The control result at ILC trial 50

control energy solution (6.38) which is consistent with our expectations in Theorem 6.2.

To investigate the effect of penalty parameter ρ, we set the scalar weighting $Q = R = 1$. By using Theorem 6.1 to calculate the optimal penalty parameter, we obtain $\rho^* = 1$. Figure 6.5 shows how the input accuracy evolves for different ρ over 30 ADMM iteration. Before reaching $\rho^* = 1$, the convergence speed increases, however, the convergence speed decreases after $\rho > 1$. This phenomenon consist with Proposition 6.1.

To further investigate the convergence speed of optimal penalty parameter ρ in each ILC trial, we run the simulation for 20 ILC trial with the maximum ADMM iteration $q_{max} = 10$ and the scalar weighting $Q/R = 1$. Figure 6.6 shows how $\|\vec{e}_k\|_{\vec{Q}}$ evolves for different ρ. It can be seem from the figure that for $\rho = 1$, it has the fastest convergence speed, which consists with the centralised result. This phenomenon further verify the presentation in Proposition 6.1.

To investigate the effect of scalar weighting Q and R, the penalty parameter ρ is set to be 1, the maximum-iteration q_{max} is set to be 25 and the scalar weighting Q is set to be 1. Figure 6.7 shows how $\|\vec{e}_k\|_{\vec{Q}}$ evolves for different R over 20 ILC trials. The proposed algorithm can always guarantees the tracking error norms converge monotonically to zero for different choice of R. It should be noticed that, increases R results faster convergence speed, however, fast convergence speed is at the expense of robustness. In practice, the choices of scalar weighting R should accord to the requirements.

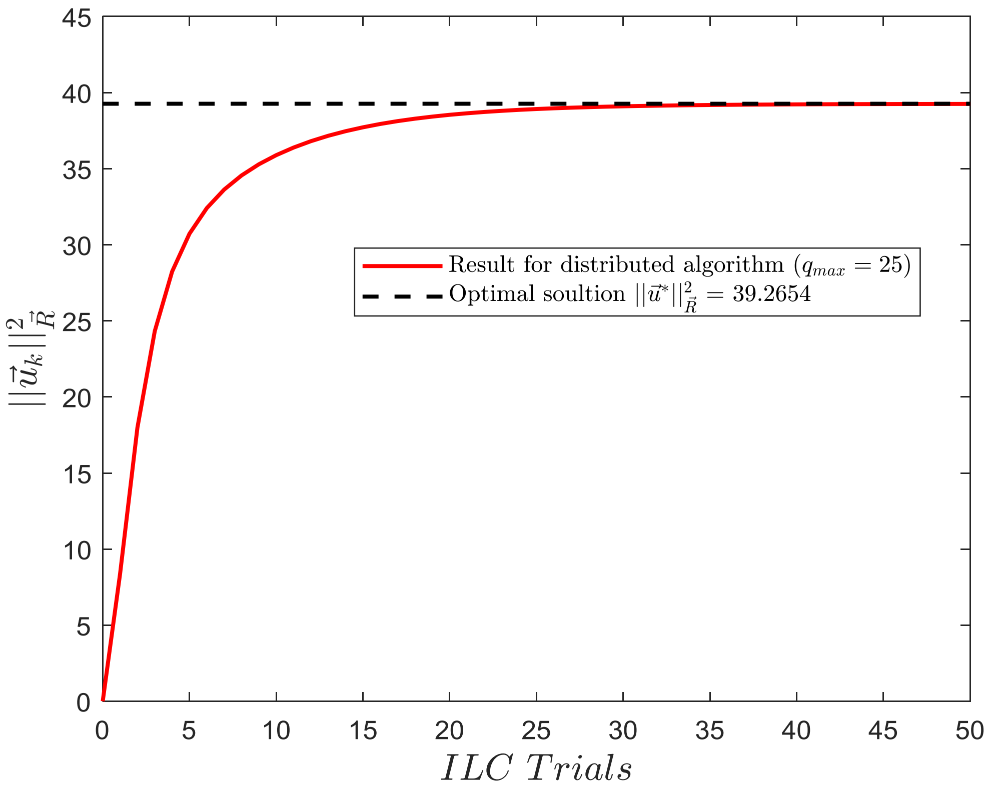

Figure 6.4: The control input energy consumption

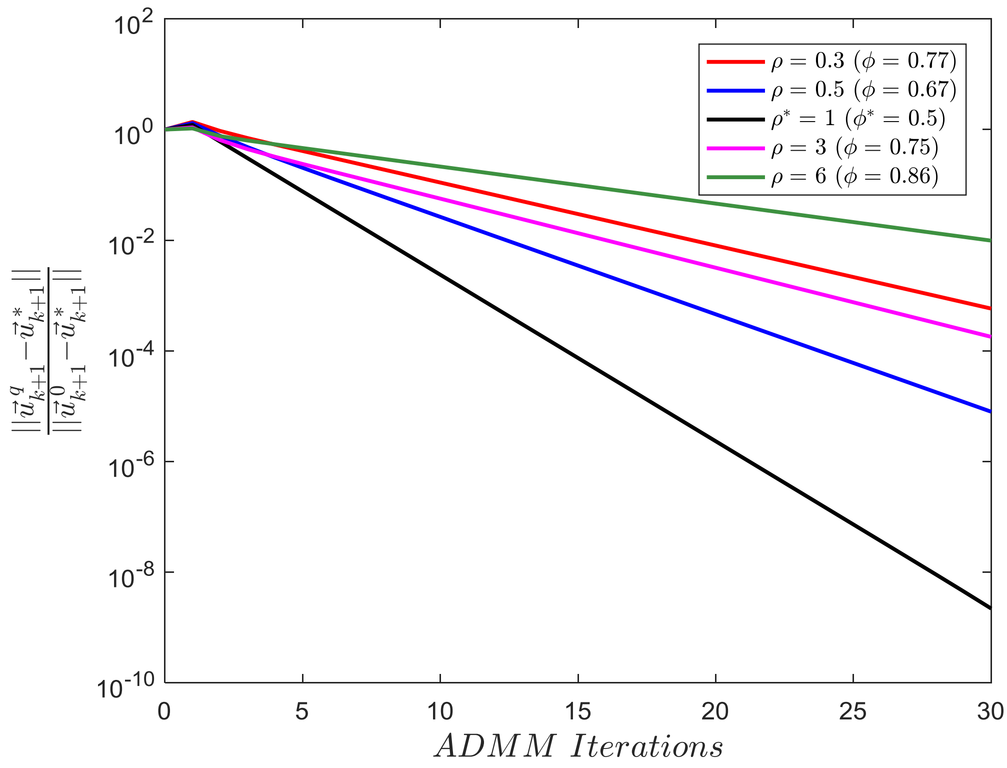

Figure 6.5: Convergence comparison for different penalty parameter ρ in ADMM

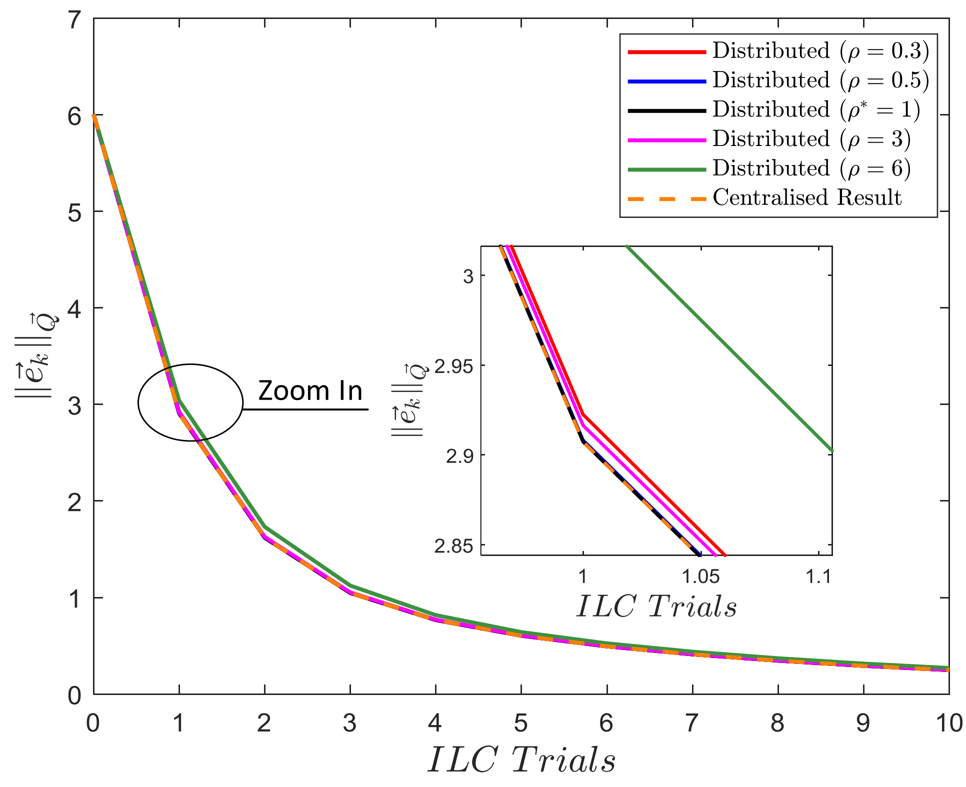

Figure 6.6: Convergence comparison for different penalty parameter ρ in ILC

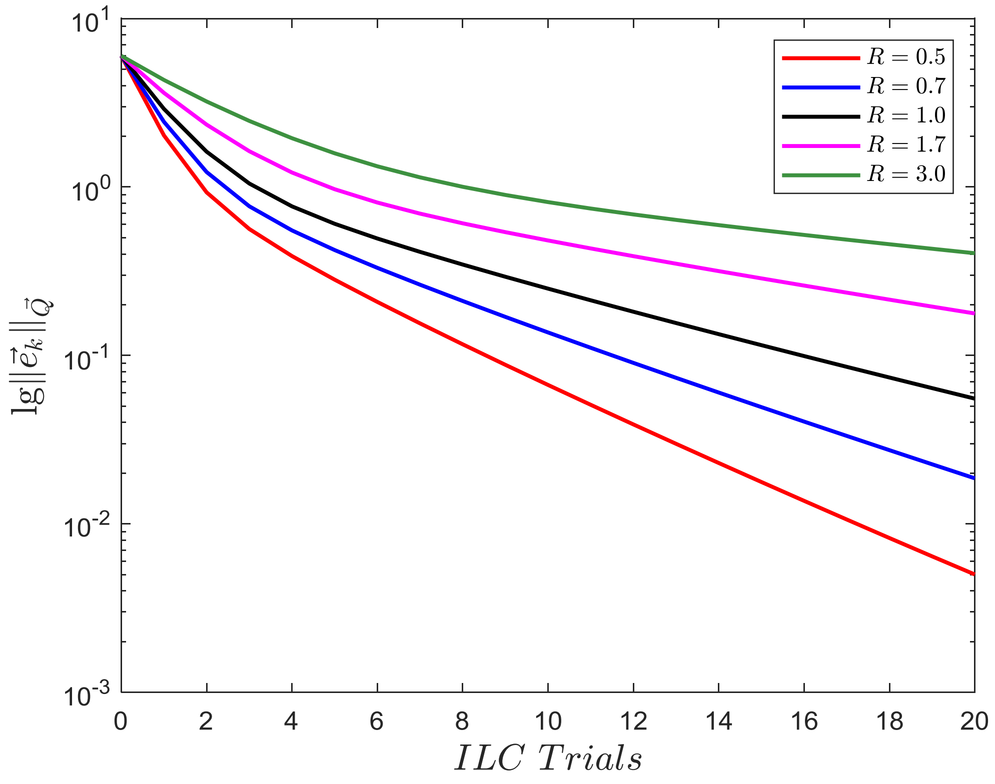

Figure 6.7: Convergence comparison for different R (with $Q = 1$)

6.5 Summary

This chapter proposes a novel distributed NOILC algorithm to solve the formation control problem of networked dynamical systems. The proposed NOILC framework guarantees the formation error norm monotonically converges to zero, and for a particular choice of initial input converges to the minimum control input energy solution, which is desirable in practice. Furthermore, we develop a distributed algorithm to implement the proposed NOILC framework using ADMM which greatly reduces the computational complexity so it can be applied to large scale networked dynamical systems. Also, a method to find the best penalty parameter ρ is given in Proposition 6.1. Numerical simulations using heterogeneous non-minimum phase networked dynamical systems are given to demonstrate the proposed algorithm's effectiveness.

For the proposed algorithm, it can guarantee the total input energy consumption is minimum, as shown in Theorem 6.2. However, there exist some applications that consider the individual input energy cost and none of the ILC existing algorithms have considered this problem. To bridge this gap, we will propose novel ILC algorithms to solve this unexplored problem in the next chapter.

Chapter 7

Distributed ILC for Networked Systems with Guaranteed Individual Energy Cost

High performance formation control problem where a group of subsystems work repetitively to form a desired formation (using only local information) within a finite time interval, has attracted significant interest in a range of areas, e.g., in transportation, robotics and satellites (Knorn et al., 2016; Olfati-Saber et al., 2007). Most of the existing ILC algorithms for high performance formation control problem focus on the tracking performance without considering constraints on the input energy cost. However, there exist some applications where the individual input energy consumption is of great concern. As an example, an autonomous air transportation system contains a number of unmanned aerial vehicles (UAVs) repetitively transporting the goods from the origin to the destination. During the control process, each UAV is required to maintain the formation with its neighbours, at the same time, minimising its input energy or ensuring its input energy consumption (i.e., individual energy cost) does not exceed the energy limit (of that stored in its battery).

For the high performance formation control problem with guaranteed individual energy cost, it contains infinity input solutions (since there is no direct reference in the control problem), which creates more freedom in the input selection. ILC algorithms that can achieve the desired formation control requirement with guaranteed individual energy cost have been waiting.

To address the above problem, this chapter proposes two novel ILC algorithms for high performance formation control problem with guaranteed individual energy cost. The first algorithm considers the scenario where a prescribed level on the energy consumption of each subsystem is known. The algorithm guarantees monotonic convergence of the formation error norm, and satisfaction of the energy level requirement. The second algorithm, on the other hand, aims to optimise the energy use of the subsystems (by minimising a common upper bound). However, monotonic convergence of the formation error norm is lost.

In addition, the proposed algorithms have some very nice properties. They can be applied to both homogeneous and heterogeneous networks, as well as non-minimum phase dynamics. Furthermore, we provide two distributed implementations using the alternating direction method of multipliers (ADMM) for the proposed algorithms, allowing them to be applied to large scale networked dynamical systems and have great scalability. This chapter is based on the work from Chen and Chu (2021).

The rest of this chapter is organised as follows: Section 7.1 formulates the system dynamics, network topology, and provides the control objectives; Section 7.2 introduces the first distributed ILC algorithm, with its convergence properties given; Section 7.3 proposes the second distributed ILC algorithm and analyses its convergence properties; Section 7.4 presents numerical examples to verify the effectiveness of the proposed algorithms; Section 7.5 concludes this chapter.

7.1 Problem Formulation

In this section, the dynamics and network topology of the system are introduced, and the design objectives are described. For simplicity, we consider a discrete time, single-input-single-output (SISO) system in this chapter.

7.1.1 System Dynamics

Consider a (homogeneous or heterogeneous) networked dynamical system with p subsystems, where i^{th} ($1 \leq i \leq p$) subsystem's dynamics is represented using the following linear time invariant (LTI), SISO, discrete time state space model

$$\begin{aligned} x_{i,k}(t+1) &= A_i x_{i,k}(t) + B_i u_{i,k}(t), \qquad && x_{i,k}(0) = x_{i,0} \\ y_{i,k}(t) &= C_i x_{i,k}(t), && t \in [0, N] \end{aligned} \tag{7.1}$$

where t is the time index; k is the ILC trial index; $x_{i,k}(\cdot) \in \mathbb{R}^{n_i}$ (n_i is the order of i^{th} subsystem), $u_{i,k}(\cdot) \in \mathbb{R}$, $y_{i,k}(\cdot) \in \mathbb{R}$ are state, input and output of subsystem i on k^{th} trial; A_i, B_i, C_i are system matrices with proper dimensions. The system is required to work in a repetitive manner, i.e., starting from the initial condition $x_{i,0}$, it performs the same task within the time interval $[0, N]$ and after $t = N + 1$, the time t is reset to zero, the state of i^{th} subsystem is reset to the initial condition $x_{i,0}$, and the system executes the same task again.

For high performance formation control problem, it requires all the subsystems work together to construct a desired formation using only local information, which makes the design problem non-trivial. Given any subsystems i and j, denote $y_{ij,k}(t) = y_{i,k}(t) - y_{j,k}(t)$ as their relative position, and then the objective can be stated as designing a proper input $u_{i,k}(t)$ $(1 \leq i \leq p)$ such that

$$\lim_{k\to\infty} y_{ij,k}(t) = r^*_{ij}(t) = r^*_i(t) - r^*_j(t) \tag{7.2}$$

where $r^*_{ij}(t)$ is the prescribed relative formation, and $r^*_i(t)$ denotes a virtual reference for subsystem i. It should be noted that $r^*_i(t)$ is neither supposed to be tracked by subsystem i nor known to subsystem i, since it is only used to define the formation (for notational simplicity).

To facilitate subsequent ILC design, a 'lifted form' representation proposed in Hatonen et al. (2004) is introduced. By assuming unity relative degree for all the subsystems (i.e., $C_iB_i \neq 0$), the 'lifted' input, output and virtual reference are represented as

$$\begin{aligned} u_{i,k} &= [u_{i,k}(0), u_{i,k}(1), \cdots, u_{i,k}(N-1)]^T \\ y_{i,k} &= [y_{i,k}(1), y_{i,k}(2), \cdots, y_{i,k}(N)]^T \\ r^*_i &= [r^*_i(1), r^*_i(2), \cdots, r^*_i(N)]^T \end{aligned} \tag{7.3}$$

For the formation control objective to be achievable, it is assumed that the initial condition between neighbouring subsystems should satisfy the condition $y_{ij,k}(0) = r^*_{ij}(0)$ (due to the relative degree is unity in the design, the current output at time instant $t = 0$ is not able to be controlled).

The system model (7.1) can then be rewritten as

$$y_{i,k} = G_i u_{i,k} + d_i \tag{7.4}$$

where d_i is the response of initial conditions given by

$$d_i = \left[C_iA_ix_{i,0}, C_iA_i^2x_{i,0}, C_iA_i^3x_{i,0}, \cdots, C_iA_i^Nx_{i,0}\right]^T \tag{7.5}$$

and the system matrix G_i is defined as

$$G_i = \begin{bmatrix} C_iB_i & 0 & \cdots & 0 \\ C_iA_iB_i & C_iB_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_iA_i^{N-1}B_i & C_iA_i^{N-2}B_i & \cdots & C_iB_i \end{bmatrix} \tag{7.6}$$

Now, the control design objective (7.2) can be rewritten in a 'lifted form' as

$$\lim_{k\to\infty} y_{ij,k} = r^*_i - r^*_j \tag{7.7}$$

For simplicity, we introduce $\vec{u}_k$, $\vec{y}_k$, $\vec{r}^*$, $\vec{d}$ as the global vectors of the 'lifted' input $u_{i,k}$, output $y_{i,k}$, desired formation $\vec{r}_i^*$, and initial condition d_i, represented as

$$\begin{aligned} \vec{u}_k &= \left[u_{1,k}^T, u_{2,k}^T, \cdots, u_{p,k}^T\right]^T \\ \vec{y}_k &= \left[y_{1,k}^T, y_{2,k}^T, \cdots, y_{p,k}^T\right]^T \\ \vec{r}^* &= \left[r_1^{*T}, r_2^{*T}, \cdots, r_p^{*T}\right]^T \\ \vec{d} &= \left[d_1^T, d_2^T, \cdots, d_p^T\right]^T \end{aligned} \tag{7.8}$$

and the global system model can then be written as

$$\vec{y}_k = \mathbb{G}\vec{u}_k + \vec{d} \tag{7.9}$$

where $\mathbb{G} = \text{diag}\,(G_1, G_2, \cdots, G_p)$.

7.1.2 Network Topology

In this chapter, we use an undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ to represent the network topology, where $\mathscr{V} = \{1, 2, ..., p\}$ is the node set and $\mathscr{E} \subset \mathscr{V} \times \mathscr{V}$ is the set of pairs of nodes called edges. For i^{th} subsystem, its neighbour's set is represented as $\mathscr{N}_i := \{j : (i,j) \in \mathscr{E}\}$.

To represent the topology relationship between different subsystems, we introduce the adjacency matrix $\mathscr{A} = [a_{ij}]$, with its element a_{ij} defined as

$$a_{ij} = \begin{cases} W_{ij} & if\ (i,j) \in \mathscr{E} \\ 0 & otherwise \end{cases} \tag{7.10}$$

where weight W_{ij} is often considered as the connection strength of the edge. Based on the i^{th} node's neighbours set, the degree of a node i is defined as $d(i) = \sum_{j=1}^{p} a_{ij}$ and it follow by the degree matrix $\mathscr{D} = diag(d(1), d(2), \cdots d(p))$. Using the definition of adjacency matrix and degree matrix, the Laplacian matrix is defined as $L = \{l_{ij}\} := \mathscr{D} - \mathscr{A}$, which is a real symmetric matrix with element l_{ij} defined as below

$$l_{ij} = \begin{cases} -W_{ij} & if \quad j \in \mathscr{N}_i \\ \sum_{j \in \mathscr{N}_i} W_{ij} & if \quad j = i \\ 0 & otherwise \end{cases} \tag{7.11}$$

To ensure the control design objectives are achievable, the following standard assumption is necessary:

Assumption 7.1. *The graph $\mathscr{G}$ is connected, i.e., one node can be reached by another node through one path.* (Bullo, 2018)

7.1.3 Iterative Learning Control Design

To achieve the desired formation, all the subsystems need to make efforts and work together, and hence the individual energy cost is of interest. We introduce $J_{i,k}$ as i^{th} subsystem's input energy cost on trial k, which is defined as

$$J_{i,k} = \|u_{i,k}\|^2_{R_i} = u^T_{i,k} R_i \, u_{i,k} \tag{7.12}$$

where R_i is a positive definite weighting matrix, representing the 'price' of i^{th} subsystem's energy cost. We consider the following two scenarios/problems:

P1: The networked system is required to achieve a desired formation, at the same time, the individual energy cost $J_{i,k}$ $(1 \leq i \leq p)$ does not exceed a prescribed level M.

P2: The networked dynamical system is required to achieve a desired formation, at the same time, an upper bound on the individual energy cost $J_{i,k}$ $(1 \leq i \leq p)$ is minimised.

The ILC design objectives can be stated as constructing input updating law in the following form

$$\vec{u}_{k+1} = f(\vec{u}_k, \vec{y}_k) \tag{7.13}$$

such that

$$\lim_{k \to \infty} y_{ij,k} = r^*_{ij}, \quad i = 1, \cdots, p, \quad j \in \mathscr{N}_i \tag{7.14}$$

and for problem P1

$$\|u_{i,k}\|^2_{R_i} \leq M, \quad i = 1, \cdots, p, \quad \forall k > 0 \tag{7.15}$$

where $M > 0$ is a prescribed energy consumption level, or for problem P2

$$M = \lim_{k \to \infty} \max_i \|u_{i,k}\|^2_{R_i} = \arg\min\{M | \boldsymbol{L}(\mathbb{G}\vec{u}_k + \vec{d}) = \boldsymbol{L}\vec{r}^*, \|u_{i,k}\|^2_{R_i} \leq M\} \tag{7.16}$$

is minimised.

Since there is no reference information available for subsystems in the problem, each subsystem needs to exchange its information with neighbouring subsystems and then uses the limited information to negotiate a best strategy to achieve the desired requirements, which makes the design problem non-trivial. To the best of our knowledge, none of the existing ILC algorithms for high performance formation control problem have considered the above practically very relevant design objectives. To bridge this gap, we will propose two novel distributed ILC algorithms that can guarantee the above objectives in the following two sections.

7.2 Distributed ILC Algorithm with Guaranteed Energy Consumption Level

In this section, a novel optimization-based ILC algorithm is provided to solve problem P1. We analyse the algorithm's convergence properties rigorously and provide a distributed implementation for the proposed algorithm.

7.2.1 Algorithm Description

Based on the constrained design for networked dynamical systems in Chapter 4 (Algorithm 4.1), we propose the following algorithm to solve problem P1:

Algorithm 7.1. *For any initial choice of $\vec{u}_0 \in \Omega$, the input sequence $\{\vec{u}_{k+1}\}_{k\geq 0}$ defined as follows*

$$\vec{u}_{k+1} = \arg\min_{\vec{u}_{k+1}\in\Omega} \left\{ \|\vec{e}_{k+1}\|^2_{\vec{Q}} + \|\vec{u}_{k+1} - \vec{u}_k\|^2_{\vec{R}} \right\} \tag{7.17}$$

where $\vec{e}_{k+1} = \boldsymbol{L}(\vec{r}^* - \vec{y}_{k+1}) = \vec{e}_k - \boldsymbol{L}(\mathbb{G}\vec{u}_{k+1} + d^* - \vec{y}_k)$ denotes the formation error, in which $\boldsymbol{L} = L \otimes I_N$, I_N is a $N \times N$ identity matrix, $\otimes$ denotes the Kronecker product, $\|\vec{e}_{k+1}\|^2_{\vec{Q}}$ denotes the quadratic form $\vec{e}^T_{k+1}\vec{Q}\vec{e}_{k+1}$ and similarly with $\|\cdot\|^2_{\vec{R}}$, $\vec{R} = \mathrm{diag}(R_1, R_2, \cdots, R_p)$, $\vec{Q} = \mathrm{diag}(Q_1, Q_2, \cdots, Q_p)$ (where $Q_i \in \mathbb{R}^N$ is a positive definite matrix), and the input constraint set Ω is defined as

$$\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_p \tag{7.18}$$

with

$$\Omega_i = \{u_{i,k+1} \in \mathbb{R}^N : \|u_{i,k+1}\|^2_{R_i} \leq M\}, \tag{7.19}$$

iteratively achieve the desired objectives (7.14) and (7.15).

Remark 7.1. *For the input law (7.17), the first term contains the error and input information from the last trial. Note that, the error information is obtained from the real experiment while the system matrix G is the nominal model. When the model uncertainty lay within a torrent range, Algorithm 1 can guarantee the convergence performance. For the torrent range of model uncertainty, it will be investigated in the future.*

Remark 7.2. *Note that the convergence speed of Algorithm 7.1 will be affected by different choices of weighting matrices $\vec{Q}$, $\vec{R}$ in a similar way as that of the standard norm optimal ILC algorithm, i.e., a smaller $\vec{R}$ leads to faster convergence speed. For more details, please refer to Chu and Owens (2010).*

Remark 7.3. *Note that the updating law (7.17) is a nonlinear optimisation problem with high order, however, it is in a very special form of the optimisation problem, called as Quadratically Constrained Quadratic Programming (QCQP) problem, that has efficient algorithms to solve it (please refer to Boyd and Vandenberghe (2004)). Hence,* *Algorithm 7.1 can be implemented in a centralised manner by solving the QCQP problem. However, solving (7.17) centrally requires significant computational complexity for large scale networked dynamical systems, and hence can be difficult to be implemented in practice. Later, we will develop a distributed implementation to make sure Algorithm 7.1 can be applied to large scale networked dynamical systems.*

7.2.2 Convergence Properties of Algorithm 7.1

The convergence analysis of Algorithm 7.1 will be discussed in two situations due to the existence of the constraint set Ω.

7.2.2.1 Perfect Formation Control is Achievable

When the prescribed input energy consumption level M is selected large enough to cover the possible solution, perfect formation control is possible. Algorithm 7.1 has appealing convergence properties, as shown in the following theorem.

Theorem 7.1. *Given any initial input $\vec{u}_0 \in \Omega$, Algorithm 7.1 guarantees the generated input sequence satisfies the input energy consumption requirements, i.e.*

$$\|u_{i,k}\|^2_{R_i} \leq M, \quad i = 1, \cdots, p, \quad \forall k \geq 0 \tag{7.20}$$

and the formation error norm converges monotonically to zero, i.e.

$$\|\vec{e}_{k+1}\|_{\vec{Q}} \leq \|\vec{e}_k\|_{\vec{Q}}, \qquad \lim_{k\to\infty} \vec{e}_k = 0. \tag{7.21}$$

Consequently, the networked dynamical system forms the desired formation as $k \to \infty$, i.e.

$$\lim_{k\to\infty} y_{ij,k} = r^*_{ij}, \quad i = 1, \cdots, p, \quad j \in \mathscr{N}_i, \tag{7.22}$$

and that the input energy consumption requirement is satisfied.

Proof. Following a similar proof of Theorem 4.2, we can obtain Theorem 7.1. □

7.2.2.2 Perfect Formation Control is Unachievable

When the selection of input energy cost level M is small and could not cover the possible solution, perfect formation control is impossible. Algorithm 7.1 guarantees the monotonic convergence of the formation error norm to a minimum formation error solution, as shown in the following theorem:

Theorem 7.2. *Given any initial input $\vec{u}_0 \in \Omega$, Algorithm 7.1 guarantees the generated input sequence satisfies the input bound constraint requirements, i.e.*

$$\|u_{i,k}\|_{R_i}^2 \leq M, \quad i = 1, \cdots, p, \quad \forall k \geq 0 \tag{7.23}$$

and the convergence of the formation error norm is monotonic, i.e.

$$\|\vec{e}_{k+1}\|_{\vec{Q}} \leq \|\vec{e}_k\|_{\vec{Q}}, \qquad k \geq 0, \tag{7.24}$$

to the minimum possible value, i.e., the solution $\vec{u}_s^*$ for the following optimisation problem

$$\vec{u}_s^* = \arg\min_{\vec{u}\in\Omega} \|\vec{e}^*\|_{\vec{Q}}^2 \tag{7.25}$$

Proof. Following a similar proof of Theorem 4.4, we can obtain Theorem 7.2. □

Both Theorems 7.1 and 7.2 show that Algorithm 7.1 guarantees the input constraint (7.15) is satisfied and the formation error norm converges monotonically to the solution that produces a minimum formation error norm, which is desirable in practice. However, as mentioned in Remark 7.3, implementing Algorithm 7.1 in a centralised manner requires significant computational load for large scale networked dynamical systems. Hence, the next subsection provides a distributed implementation for Algorithm 7.1.

7.2.3 Distributed Implementation of Algorithm 7.1

To be suitable for large scale networked dynamical systems, we provide a distributed implementation for Algorithm 7.1 using the idea of 'consensus' formulation in ADMM. In the following section, we will first introduce the general idea of ADMM.

7.2.3.1 The Alternating Direction Method of Multipliers

The alternating direction method of multipliers (ADMM) is a well-known distributed implementation algorithm, which was firstly presented in Gabay and Mercier (1976) and systematically reviewed in Boyd et al. (2011). ADMM can guarantee the convergence for any positive penalty parameter ρ, contrasting with other distributed algorithms that rely on the selection of step size (Boyd et al., 2011). Rigorous analysis and proof of the convergence can be found in Boyd et al. (2011).

Consider the following distributed optimisation problem to illustrate the basic idea of ADMM:

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{p} f_i(x_i) + g(z) \\ &\text{subject to} \quad x_i - \widetilde{E}_i z = 0, \quad i = 1, \cdots, p \end{aligned} \tag{7.26}$$

where the global variable $z = [z_1^T \quad z_2^T \quad \cdots \quad z_p^T]^T \in \mathbb{R}^p$, $x_i \in \mathbb{R}^{p_i}$ is the local input decision variable, and $\widetilde{E}_i$ is a matrix that selects the components from z that match the local variable x_i.

The augmented Lagrangian for (7.26) is shown as

$$\begin{aligned} L_\rho(x, z, \gamma) &= \sum_{i=1}^{p} L_{\rho i}(x_i, z, \gamma_i) + g(z) \\ L_{\rho i}(x_i, z, \gamma_i) &= J_i(x_i) + \gamma_i^T(x_i - \widetilde{E}_i z) + \frac{\rho}{2}\|x_i - \widetilde{E}_i z\|^2 \end{aligned} \tag{7.27}$$

and ADMM solves the optimisation problem (7.26) by iteratively performing the following three steps

$$x_i^{q+1} = \arg\min L_{\rho i}(x_i, z^q, \gamma_i^q) \tag{7.28}$$

$$z^{q+1} = \arg\min L_\rho(x^{q+1}, z, \gamma^q) \tag{7.29}$$

$$\gamma_i^{q+1} = \gamma_i^q + \rho(x_i^{q+1} - \widetilde{E}_i z^{q+1}) \tag{7.30}$$

where q is the ADMM iteration index and $\gamma_{i,k+1}^{q+1}$ is the local dual variable.

Remark 7.4. *Note that, if the global regularization function $g(\cdot)$ is separable, the global value update law can also be done in a distributed manner. For problem P1, the global constraint set is formed by several individual constraint set $\Omega_i (1 \le i \le p)$ and hence the z-update step can be written into p small update steps, which will be shown in the following sections.*

7.2.3.2 Distributed Algorithm 7.2

Note that, the optimal solution of input law (7.17) at trial $k+1$ can be found by solving the following (equivalent) problem

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{p} f_{i,k+1}(\vec{u}_{i,k+1}) \\ &\text{subject to} \quad \vec{u}_{i,k+1} - \widetilde{E}_i z_{k+1} = 0, \quad i = 1, \cdots, p \end{aligned} \tag{7.31}$$

where $z_{k+1} \in \mathbb{R}^{pN}$ is the global value on trial $k+1$

$$z_{k+1} = \begin{bmatrix} z_{1,k+1}^T & z_{2,k+1}^T & \cdots & z_{p,k+1}^T \end{bmatrix}^T \tag{7.32}$$

$\vec{u}_{i,k+1} \in \mathbb{R}^{p_i N}$ is the local input plan for subsystem i on trial $k+1$, e.g., if $\mathcal{N}_i = \{l, m\}$

$$\vec{u}_{i,k+1} = \begin{bmatrix} u_{i,k+1}^T & u_{l,k+1}^T & u_{m,k+1}^T \end{bmatrix}^T \tag{7.33}$$

and $\widetilde{E}_i$ is the corresponding matrix that maps the local input $\vec{u}_{i,k+1}$ to the global element z_{k+1}.

In (7.31), the local cost function for i^{th} subsystem $f_{i,k+1}(\vec{u}_{i,k+1})$ is defined as

$$f_{i,k+1}(\vec{u}_{i,k+1}) = \| \sum_{j\in\mathscr{N}_i} W_{ij}[(G_i u_{i,k+1} + d_i) - (G_j u_{j,k+1} + d_j) - r^*_{ij}]\|^2_{Q_i} + \|u_{i,k+1} - u_{i,k}\|^2_{R_i}$$

with the domain of $f_{i,k+1}$ defined as

$$\mathbf{dom}\, f_{i,k+1} = \left\{\vec{u}_{i,k+1} |\ \vec{u}_{i,k+1} \in \vec{\Omega}_i\right\} \tag{7.34}$$

where $\vec{\Omega}_i = \Omega_i \times \Omega_l \times \Omega_m$ (if $\mathcal{N}_i = \{l, m\}$).

Now, applying ADMM to solve (7.17) distributively, a distributed ILC algorithm is obtained as following:

Algorithm 7.2. *Given any initial input $\vec{u}_0 \in \Omega$, the input $\vec{u}_{k+1}$ generated distributively by the following ADMM steps*

$$\vec{u}^{q+1}_{i,k+1} = \arg \min_{\vec{u}_{i,k+1}\in\vec{\Omega}_i} \left[f_{i,k+1}(\vec{u}^{q+1}_{i,k+1}) + \frac{\rho}{2}\|\vec{u}^{q+1}_{i,k+1} - \widetilde{E}_i z^q_{k+1}\|^2 + {\gamma^q_{i,k+1}}^T(\vec{u}^{q+1}_{i,k+1} - \widetilde{E}_i z^q_{k+1})\right] \tag{7.35}$$

$$z^{q+1}_{i,k+1} = \frac{1}{1+|\mathcal{N}_i|} \sum_{o\in(\mathcal{N}_i \bigcup i)} \left[(\vec{u}^{q+1}_{o,k+1})_i + \frac{1}{\rho}(\gamma^{q+1}_{o,k+1})_i\right] \tag{7.36}$$

$$\gamma^{q+1}_{i,k+1} = \gamma^q_{i,k+1} + \rho(\vec{u}^{q+1}_{i,k+1} - \widetilde{E}_i z^{q+1}_{k+1}) \tag{7.37}$$

with

$$\vec{u}_{k+1} = \lim_{q\to\infty} z^{q+1}_{k+1} \tag{7.38}$$

iteratively solves the high performance formation control in the input bound constraints set Ω, i.e.

$$\|u_{i,k}\|^2_{R_i} \le M, \quad i = 1, \cdots, p, \quad \forall k \ge 0 \tag{7.39}$$

$$\lim_{k\to\infty} y_{ij,k} = r^*_{ij}, \quad i = 1, \cdots, p, \quad j \in \mathscr{N}_i, \tag{7.40}$$

where $(\vec{u}^{q+1}_{o,k+1})_i$ and $(\gamma^{q+1}_{o,k+1})_i$ denotes the element in $\vec{u}^{q+1}_{o,k+1}$ and $\gamma^{q+1}_{o,k+1}$ corresponding to $z_{i,k+1}$.

Remark 7.5. *In theory, ADMM requires infinite iterations to converge to the centralised solution. Fortunately, ADMM has been shown to be very efficient in practice, and a small number of iterations are usually sufficient in most of the cases. Later simulations will illustrate this.*

7.3 Distributed ILC Algorithm with Minimum Individual Input Energy Consumption

In this section, we propose a distributed ILC algorithm to solve problem P2, i.e., to find the minimum individual input energy consumption level M while achieving the desired formation. The algorithm's convergence properties are analysed, and a distributed implementation is provided.

7.3.1 Algorithm Description

To design the algorithm, we introduce an auxiliary vector $\hat{u}_{k+1}$ as following:

$$\hat{u}_{k+1} = [\vec{u}_{k+1}^T \quad M_{k+1}]^T \tag{7.41}$$

where $M_{k+1} \in \mathbb{R}$. By introducing the auxiliary vector, the problem P2 can now be stated as finding $\hat{u}_{k+1}$ that the input $\vec{u}_{k+1}$ achieves the desired formation requirement (7.14), and at the same time, the bound M_{k+1} defined in (7.16) is minimised.

Based on the above formulation, we can design the following algorithm for the problem P2:

Algorithm 7.3. *For any initial choice of $\hat{u}_0$, the sequence $\{\hat{u}_{k+1}\}_{k\geq 0}$ defined as follows*

$$\hat{u}_{k+1} = \underset{\hat{u}_{k+1}\in\Omega_{k+1}}{\text{argmin}} \{T\hat{u}_{k+1}\} \equiv \underset{\hat{u}_{k+1}\in\Omega_{k+1}}{\text{argmin}} \; M_{k+1} \tag{7.42}$$

where the matrix T is defined as

$$T = [\underbrace{0 \quad 0 \quad \cdots \quad 0}_{p} \quad 1]^T \tag{7.43}$$

and constraint set Ω_{k+1} is defined as

$$\Omega_{k+1} = \Psi_{1,k+1} \cap \Psi_{2,k+1} \cap \cdots \cap \Psi_{p,k+1} \cap \Phi_{k+1} \tag{7.44}$$

in which $\Psi_{i,k+1}$ $(1 \leq i \leq p)$ defined as

$$\Psi_{i,k+1} = \{\hat{u}_{k+1} \in \mathbb{R}^{pN+1} : \; \|u_{i,k+1}\|_{R_i}^2 \leq T\hat{u}_{k+1}\} \tag{7.45}$$

and Φ_{k+1} is defined as

$$\Phi_{k+1} = \{\hat{u}_{k+1} \in \mathbb{R}^{pN+1} : \vec{e}_k - \boldsymbol{L}\mathbb{G}(\vec{u}_{k+1} - \vec{u}_k) = 0\}, \tag{7.46}$$

iteratively achieve the objectives (7.14) and (7.16).

Note that, Algorithm 7.4 can be implemented in a centralised manner by solving the following second order cone programming (SOCP) problem:

$$\begin{aligned}
&\text{minimize} && T\hat{u}_{k+1} \\
&\text{subject to} && \boldsymbol{L}\mathbb{G}(\vec{u}_{k+1} - \vec{u}_k) = \vec{e}_k \\
& && \|u_{i,k+1}\|^2_{R_i} \leq T\hat{u}_{k+1}, \quad \forall 1 \leq i \leq p
\end{aligned} \tag{7.47}$$

Using (7.47) to solve Algorithm (7.42), yields the following algorithm for problem P2:

Algorithm 7.4. *For any initial choice of $\hat{u}_0$, the input sequence $\{\hat{u}_{k+1}\}_{k\geq 0}$ obtained by solving the following optimisation problem*

$$\begin{aligned}
&\text{minimize} && T\hat{u}_{k+1} \\
&\text{subject to} && \boldsymbol{L}\mathbb{G}(\vec{u}_{k+1} - \vec{u}_k) = \vec{e}_k \\
& && \|u_{i,k+1}\|^2_{R_i} \leq T\hat{u}_{k+1}, \quad \forall 1 \leq i \leq p
\end{aligned} \tag{7.48}$$

iteratively achieve the objectives (7.14) and (7.16).

However, same as in Algorithm 7.1 for Problem P1, solving (7.42) centrally requires significant computational complexity for large scale networked dynamical systems, and hence can be difficult to be implemented in practice. In the next section, we will design a distributed implementation for Algorithm 7.4 so it can be used to large scale networked dynamical systems.

7.3.2 Distributed Algorithm 7.4

To design the distributed ILC algorithm, we first introduce the following auxiliary matrices and vectors

$$\mathcal{A}_i = \begin{bmatrix} I_{p_iN+1} \\ \boldsymbol{L}_i\vec{G}_i \end{bmatrix} \qquad \mathcal{B}_i = \begin{bmatrix} \hat{E}_i \\ 0_{N\times(pN+1)} \end{bmatrix} \qquad \mathcal{C}_{i,k+1} = \begin{bmatrix} 0_{(p_iN+1)\times 1} \\ \vec{e}_{i,k} + \boldsymbol{L}_i\vec{G}_i\vec{u}_k \end{bmatrix}$$

$$\hat{u}_{i,k+1} = [\vec{u}^T_{i,k+1} \quad M_{i,k+1}]^T$$

$$\hat{z}_{k+1} = [z^T_{k+1} \quad M_{k+1}]^T$$

where $\hat{E}_i$ is a mapping matrix that links $\hat{u}_{i,k+1}$ with $\hat{z}_{k+1}$, and the local formation error $\vec{e}_{i,k}$ is defined in the form that

$$\vec{e}_{i,k} = \sum_{j\in\mathcal{N}_i} W_{ij}[(G_iu_{i,k} + d_i) - (G_ju_{j,k} + d_j) - r^*_{ij}] \tag{7.49}$$

and $\vec{G}_i$, $\boldsymbol{L}_i$ are local system matrix and local Laplacian matrix. As an example, if $\mathcal{N}_i = \{l, m\}$, then

$$\begin{aligned} \vec{G}_i &= \text{diag}(G_i, G_l, G_m) \\ \boldsymbol{L}_i &= \begin{bmatrix} \sum_{j\in\mathcal{N}_i} W_{ij} & -W_{il} & -W_{im} \end{bmatrix} \otimes I_N \end{aligned} \tag{7.50}$$

After introducing the above auxiliary matrices and vectors, the optimisation problem (7.42) can then be rewritten as the following (equivalent) optimisation problem

$$\begin{aligned} &\text{minimize} \quad T\hat{z}_{k+1} + \sum_{i=1}^{p} g_{i,k+1}(\hat{u}_{i,k+1}) \\ &\text{subject to} \quad \mathcal{A}_i\hat{u}_{i,k+1} - \mathcal{B}_i\hat{z}_{k+1} = \mathcal{C}_{i,k+1} \end{aligned} \tag{7.51}$$

where $g_{i,k+1}(\hat{u}_{i,k+1})$ is an indicator function defined as

$$g_{i,k+1}(\hat{u}_{i,k+1}) = \begin{cases} 0 & if \;\; \hat{u}_{i,k+1} \in \hat{\Psi}_{i,k+1} \\ +\infty & otherwise \end{cases} \tag{7.52}$$

in which $\hat{\Psi}_{i,k+1}$ is the local constraint set, as an example, if $\mathcal{N}_i = \{l, m\}$, then

$$\hat{\Psi}_{i,k+1} = \vec{\Psi}_{i,k+1} \cap \vec{\Psi}_{l,k+1} \cap \vec{\Psi}_{m,k+1} \tag{7.53}$$

with

$$\begin{aligned} \vec{\Psi}_{i,k+1} &= \{\hat{u}_{i,k+1} \in \mathbb{R}^{3N+3} : \|u_{i,k+1}\|^2_{R_i} \leq T_i\hat{u}_{i,k+1}\} \\ T_i &= [\underbrace{0 \quad 0 \quad \cdots \quad 0}_{p_i} \quad 1]^T \end{aligned} \tag{7.54}$$

Applying ADMM to solve problem (7.51) distributively, we have the following distributed ILC algorithm:

Algorithm 7.5. *Given any initial $\hat{u}_0$, the input $\vec{u}_{k+1}$ generated distributively by the following ADMM steps*

$$\begin{aligned} \hat{u}^{q+1}_{i,k+1} = \arg \min_{\hat{u}_{i,k+1}\in\hat{\Psi}_{i,k+1}} &\Big[\frac{\rho}{2}\|\mathcal{A}_i\hat{u}^{q+1}_{i,k+1} - \mathcal{B}_i\hat{z}^q_{k+1} - \mathcal{C}_{i,k+1}\|^2 \\ &+ {\gamma^q_{i,k+1}}^T(\mathcal{A}_i\hat{u}^{q+1}_{i,k+1} - \mathcal{B}_i\hat{z}^q_{k+1} - \mathcal{C}_{i,k+1})\Big] \end{aligned} \tag{7.55}$$

$$\hat{z}^{q+1}_{i,k+1} = \begin{cases} \frac{1}{1+|\mathcal{N}_i|} \sum\limits_{o\in(\mathcal{N}_i \bigcup i)} \left[(\vec{u}^{q+1}_{o,k+1})_i + \frac{1}{\rho}(\gamma^{q+1}_{o,k+1})_i\right] & if \;\; 1 \leq i \leq p \\ \frac{1}{p}\left\{-\frac{1}{\rho} + \sum\limits_{i=1}^{p}\left[(\hat{u}^{q+1}_{i,k+1})_{p+1} + \frac{1}{\rho}(\gamma^{q+1}_{i,k+1})_{p+1}\right]\right\} & if \;\; i = p+1 \end{cases} \tag{7.56}$$

$$\gamma^{q+1}_{i,k+1} = \gamma^q_{i,k+1} + \rho(\mathcal{A}_i\vec{u}^{q+1}_{i,k+1} - \mathcal{B}_i\hat{z}^q_{k+1} - \mathcal{C}_{i,k+1}) \tag{7.57}$$

with

$$\vec{u}_{k+1} = \lim_{q\to\infty} z^{q+1}_{k+1} \tag{7.58}$$

iteratively achieves the desired formation

$$\lim_{k\to\infty} y_{ij,k} = r^*_{ij}, \quad i = 1, \cdots, p, \quad j \in \mathscr{N}_i \tag{7.59}$$

and generates the minimum individual input energy, i.e.

$$\lim_{k\to\infty} \max_i \|u_{i,k+1}\|^2_{R_i} = M^* \tag{7.60}$$

where M^* is the solution of (7.63).

7.3.3 Convergence Properties of Algorithm 7.5

Applying Algorithm 7.5 for the Problem P2, it has nice convergence properties shown in the following:

Theorem 7.3. *For any initial choice of auxiliary vector $\hat{u}_0$, Algorithm 7.5 guarantees the achievement of the desired formation as $k \to \infty$, i.e.*

$$\lim_{k\to\infty} y_{ij,k} = r^*_{ij}, \quad i = 1, \cdots, p, \quad j \in \mathscr{N}_i \tag{7.61}$$

and the individual input energy consumption is minimised in the sense that

$$\lim_{k\to\infty} \max_i \|u_{i,k+1}\|^2_{R_i} = M^* \tag{7.62}$$

where M^* is the solution of the following problem

$$\begin{aligned} \text{minimize} \quad & M \\ \text{subject to} \quad & \|u_{i,k+1}\|^2_{R_i} \le M \\ & \boldsymbol{L}(\mathbb{G}\vec{u} + \vec{d}) = \boldsymbol{L}\vec{r}^* \end{aligned} \tag{7.63}$$

Proof. By reformulating the input updating law (7.42) into the form in (7.51), yields the distributed Algorithm 7.5 for the Problem P2. Note that, the objective function and equality constraints in (7.51) are convex. Based on the convergence properties of ADMM, we can have the optimal solution for the Problem P2, which yields Theorem 7.3. □

7.4 Numerical Examples

In this section, numerical examples are provided to verify the effectiveness of the proposed ILC algorithms. Consider a heterogeneous, non-minimum phase, over-damping networked dynamical system has four subsystems, with i^{th} subsystem's transfer function

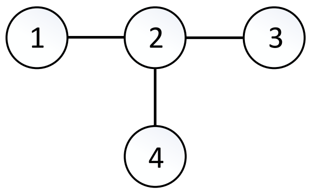

Figure 7.1: The graph structure of numerical examples

defined as

$$G_i(s) = \frac{40s - 1}{s^2 + 6s + \tau_i} \tag{7.64}$$

where $\tau_i = i$, $i = 1, 2, 3, 4$.

We assume the trial length is $1s$ and the sampling time $T_s = 0.05s$ (with a zero-order hold). Each subsystem's initial condition $x_{i,0}$ is set as 0, and the desired formation is

$$\begin{aligned} r_1^*(t) &= -1 - 0.5 * cos(\pi T_s \cdot t) \\ r_2^*(t) &= 1 + 0.5 * cos(\pi T_s \cdot t) \\ r_3^*(t) &= 2 + 1 * cos(\pi T_s \cdot t) \\ r_4^*(t) &= 2.8 + 2 * cos(\pi T_s \cdot t) \end{aligned} \tag{7.65}$$

and the network topology is shown in Figure 7.1. In the following, we will first investigate the performance of Algorithms 7.1 and 7.2 that proposed in Section 7.2.

7.4.1 Problem P1 with Prescribed Energy Consumption Level

We first consider the algorithm's performance for problem P1. We set the input energy bound $M = 8$, the weighting matrices $Q_i = R_i = I_N$, penalty parameter $\rho = 2$ and all the subsystems' initial condition $x_{i,0} = 0$ and first trial's initial input $u_{i,0} = 0$. As mentioned in Remark 7.3, the convergence speed of Algorithm 7.2 depends on the choice of maximum iteration number q_{max}, hence the effect of different q_{max} is also investigated. Figure 7.2 shows the evolution of the formation error norm for both centralised solution (7.17) and distributed solution over 14 ILC trials. It can be seen that the proposed Algorithms 7.1 & 7.2 guarantee the formation error norm converge monotonically to zero. Furthermore, the convergence speed of Algorithm 7.2 becomes faster as q_{max} increases, however, this improvement is marginal: after $q_{max} \geq 40$, the convergence performance of the distributed ILC Algorithm 7.2 is almost indistinguishable with the centralised result, which suggests that a small number of iterations is sufficient for ADMM to approach the centralised result in this case.

From Figure 7.2, it shows that the choice of $q_{max} = 40$ is sufficient to generate the optimal solution, hence, we set $q_{max} = 40$ for the following investigation. Figure 7.3 shows the individual input energy cost $J_{i,k}$ along the ILC trials, and it can be seen that

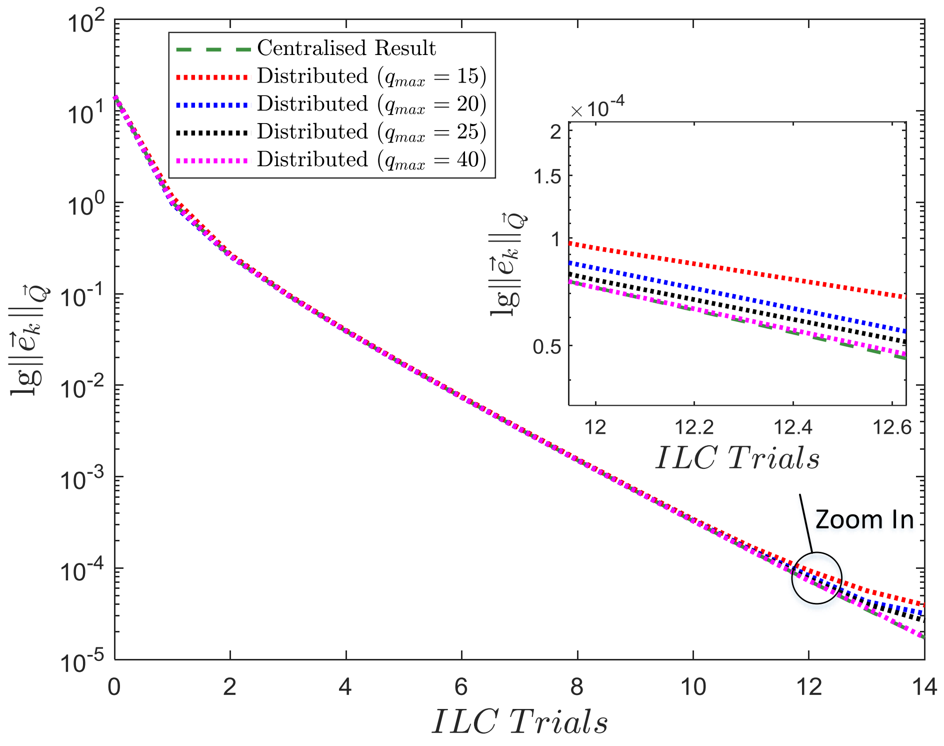

Figure 7.2: Algorithms 7.1&7.2 – Convergence of the formation error norm over 14 ILC trials

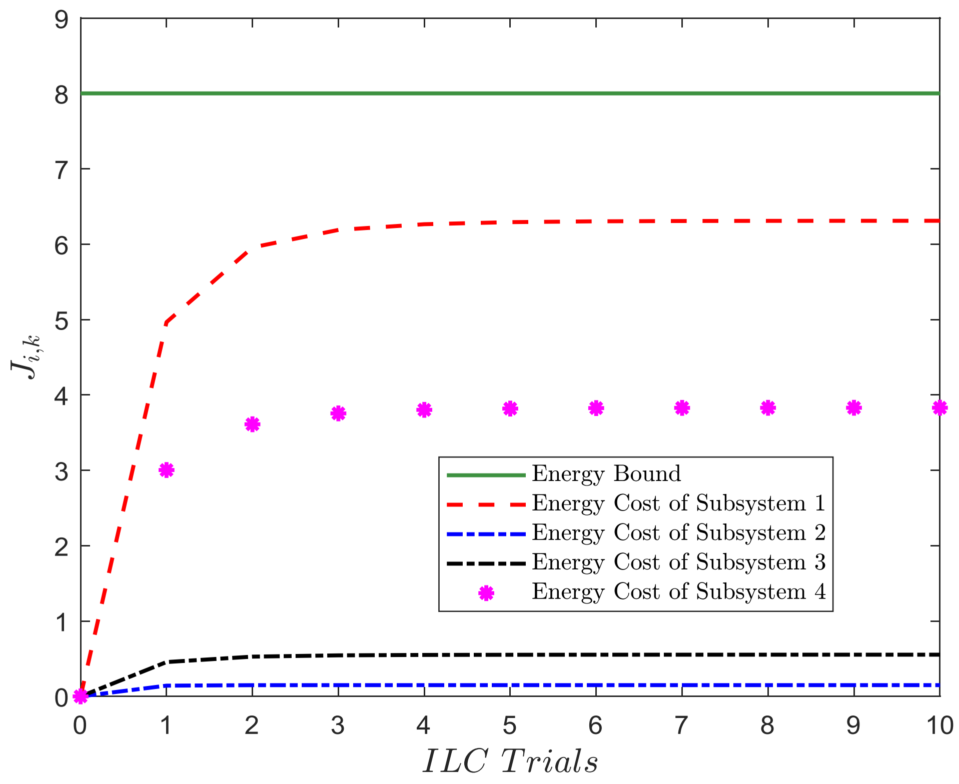

Figure 7.3: Algorithm 7.2 – Convergence of the individual energy cost of different subsystems over 10 ILC trials

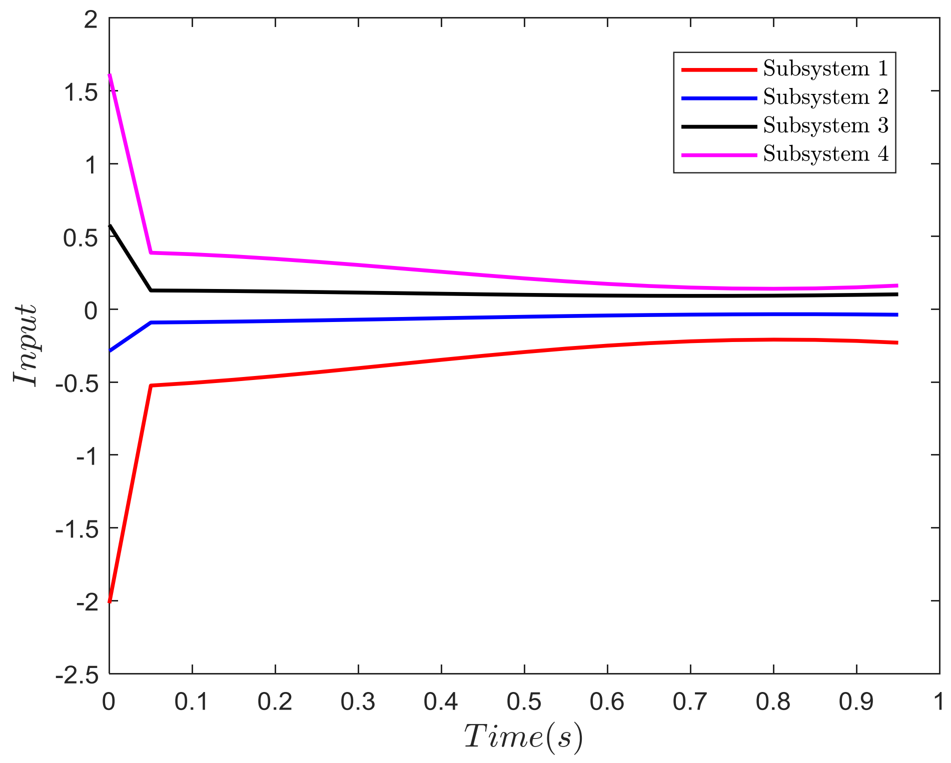

Figure 7.4: Algorithm 7.2 – Input of different subsystems on the 10th ILC trial

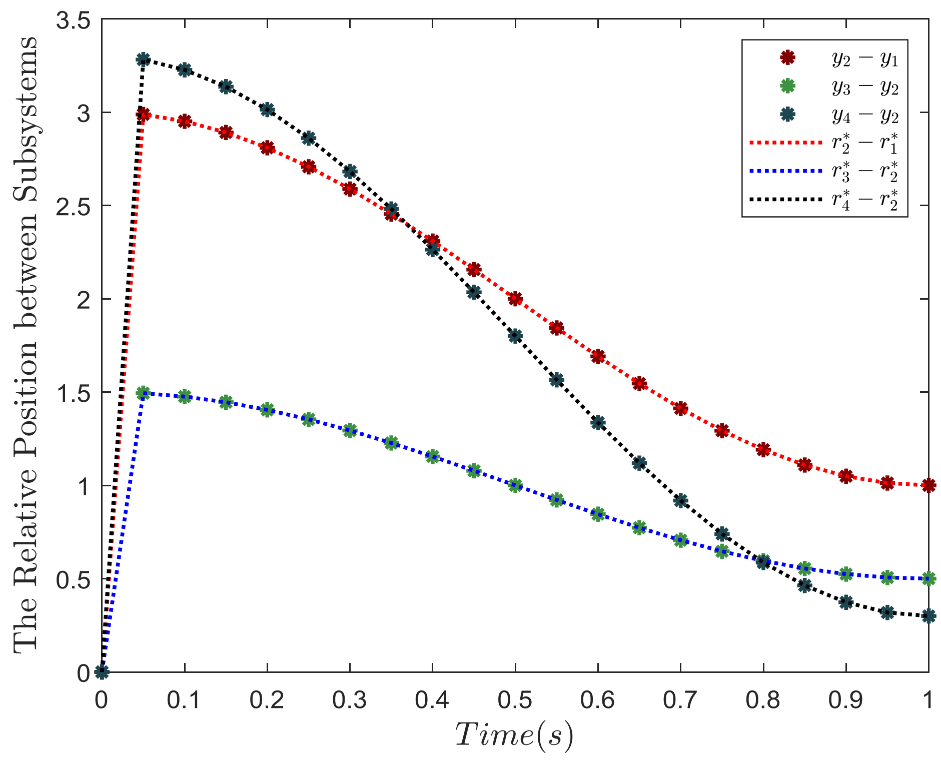

Figure 7.5: Algorithm 7.2 – Relative position between different subsystems on the 10th ILC trial

the prescribed input energy bound $M = 8$ is satisfied by each subsystem during the whole control process. Figure 7.4 presents the input of different subsystems on the 10^{th} ILC trial and Figure 7.5 shows the output difference between different subsystems on the 10^{th} ILC trial, demonstrating that all the subsystems achieve the desired formation, which is consistent with our expectations. Simulations with different penalty parameter ρ, weighting matrices Q_i and R_i are also investigated. The results are consistent with our expectations and are omitted here for brevity.

7.4.2 Problem P2 with Minimum Individual Energy Consumption

In the following simulation, we consider the same system as in Example 7.4.1 and set the weighting matrices $Q_i = R_i = I_N$, penalty parameter $\rho = 2$, and all the subsystems' first trial's input $u_{i,0} = 0$ to demonstrate the effectiveness of Algorithm 7.5.

For Algorithm 7.5, we set the maximum ADMM iteration number $q_{max} = 40$ and Figure 7.6 shows the evolution of individual input energy cost $J_{i,k}$ along the ILC trials. From the figure, it shows that Algorithm 7.5 can automatically find the minimum individual energy cost bound (i.e., $M^* = 5.0420$) for the discussed problem. Different from Algorithm 7.5 which can only guarantee the individual energy cost not exceed a given bound (as shown in Figure 7.3), Figure 7.6 has smaller individual energy cost, which is appealing in practice.

Figure 7.7 shows the evolution of the formation error norm over 20 ILC trials. It can be seen that Algorithm 7.5 can only guarantee the formation error norm converges asymptotically to zero, since the design of Algorithm 7.5 is mainly based on ADMM that cannot guarantee the monotonic convergence property. It should be noticed that, we can choose a much larger q_{max} to make the problem becomes a steepest optimisation problem (i.e., the optimal solution can be obtained in one trial), however, even a minor model uncertainty in the system dynamics will perturb the solution for steepest optimisation problem. To enhance the robustness property of Algorithm 7.5, we suggest that choose only dozens of ADMM iterations for each ILC trial in practice, and use more practical error value to update the input (which improves the robustness to the model uncertainty). Simulations with different system dynamics, formation, damping ratio, constraint set have also been investigated, and both algorithms work well in all the cases, hence omitted here for brevity.

7.5 Summary

In this chapter, we propose two ILC algorithms for high performance formation control problem with guaranteed individual input energy cost, and both algorithms have appealing properties. For the first ILC algorithm, it can not only guarantee the formation

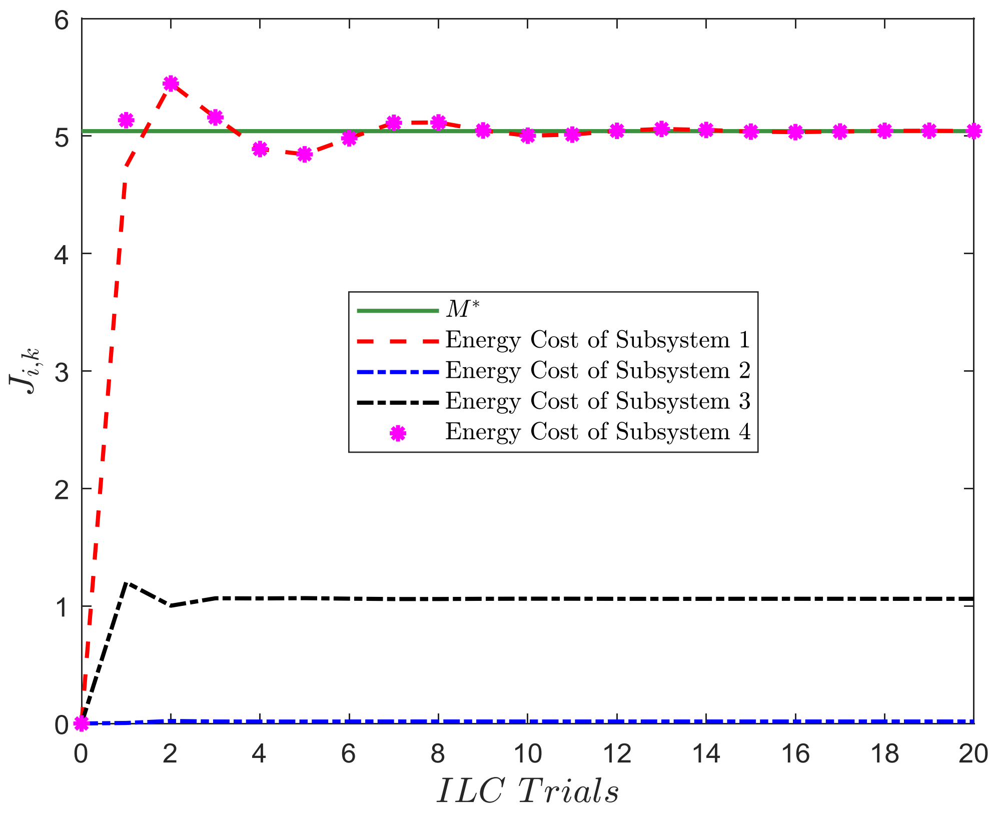

Figure 7.6: Algorithm 7.5 – Convergence of the individual energy consumption of different subsystems over 20 ILC trials

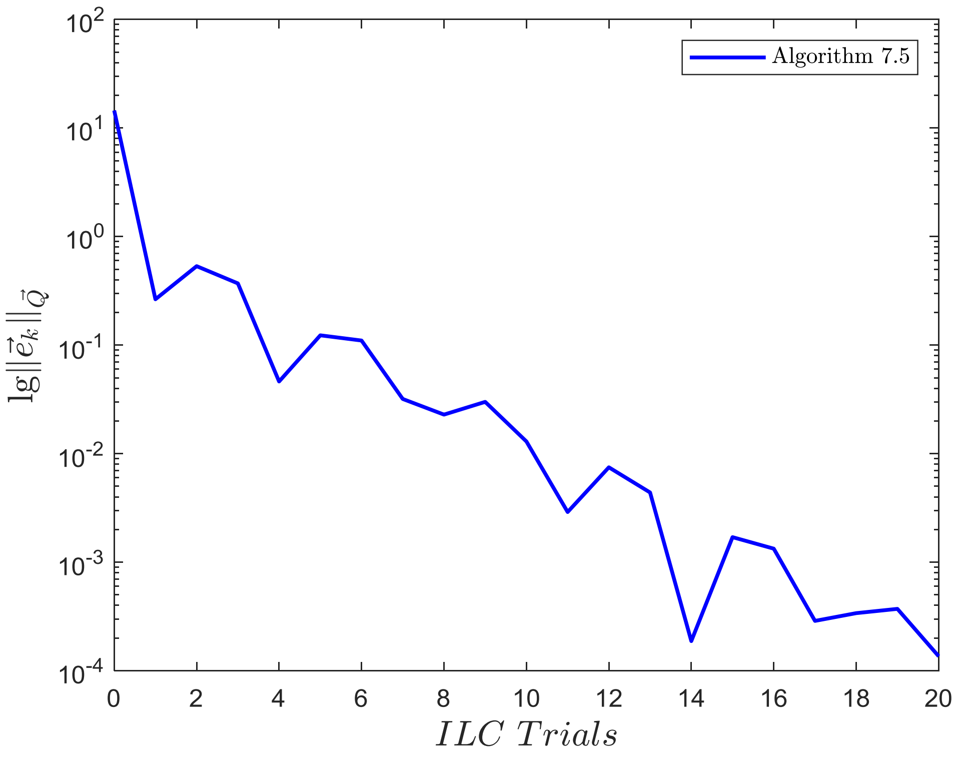

Figure 7.7: Algorithm 7.5 – Convergence of the formation error norm over 20 ILC trials

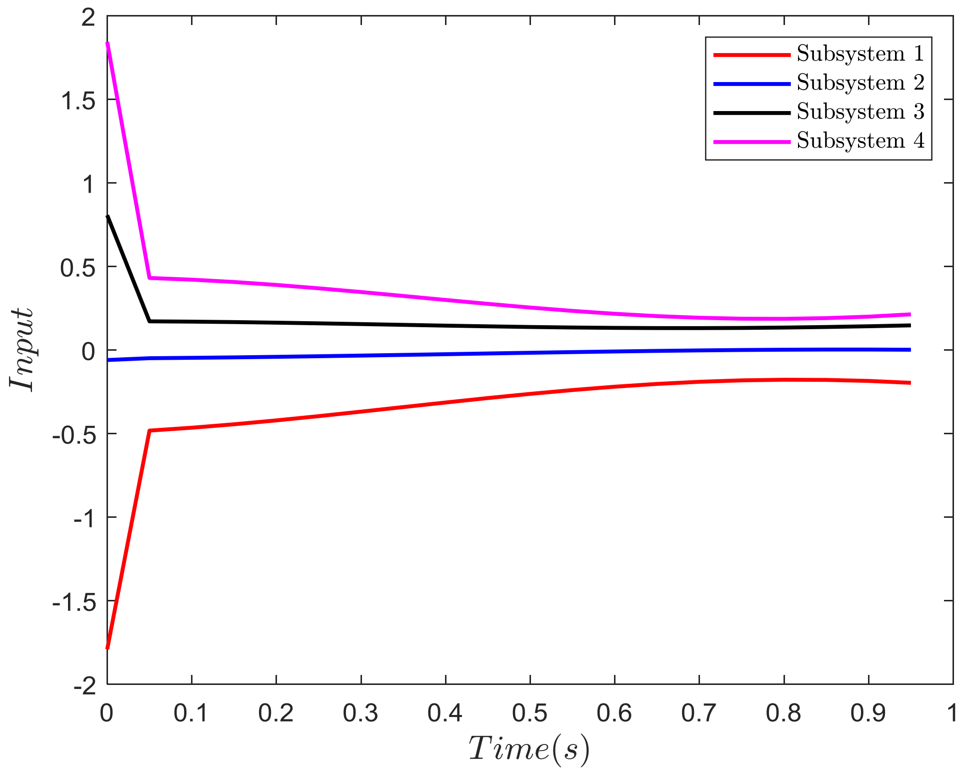

Figure 7.8: Algorithm 7.5 – Input of different subsystems on the 20th ILC trial

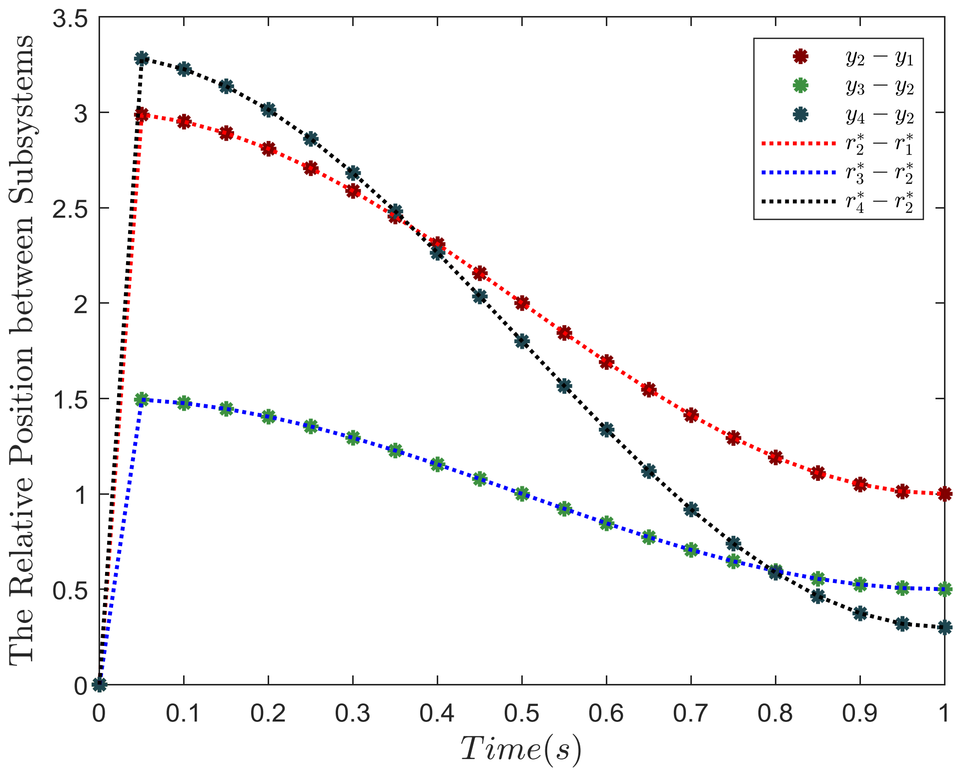

Figure 7.9: Algorithm 7.5 – Relative position between different subsystems on the 20th ILC trial

error norm converge monotonically to the minimum possible solution, but also constrain each subsystem's input energy cost within a given energy level. The second proposed ILC algorithm achieves the desired formation, at the same time, finds the minimum possible individual input energy consumption. Considering the ubiquity of large scale networked dynamical systems in practice, we further design distributed implementation methods for the proposed algorithms, allowing them to be applied to networked dynamical systems with large number of subsystems. Numerical examples with heterogeneous, non-minimum phase, discrete-time networked dynamical systems are given to verify the proposed algorithms' effectiveness. In next section, we will consider the collaborative tracking problem and propose novel decentralised ILC algorithm to solve it.

Chapter 8

Constrained ILC for Collaborative Tracking

High accuracy collaborative tracking of networked dynamical systems is an important control task that requires a group of subsystems (agents) working collaboratively to track a desired reference, and to execute the same tracking mission repetitively. Such control task has found a number of important applications in various areas. As an example, multiple actuator robots contain different actuators (subsystems) to perform a variety of collaborative positioning task repeatedly with high performance requirements (Chang and Kim, 2013; Wilcox and Devasia, 2015); human machine collaboration requires the repeated cooperation between human and subsystems to achieve a high precision operation (Warrier and Devasia, 2016, 2017).

Benefiting from the learning feature, a number of ILC based design have been proposed for collaborative tracking problem: Devasia (2016) introduces an inverse based ILC algorithm in frequency domain for heterogeneous linear networked dynamical systems and shows that any unity partitions of the updating law guarantee the achievement of the desired objective; an inverse based ILC algorithm is proposed in Warrier and Devasia (2016, 2017) for human-machine collaboration problem, which enables even a novice human operator to perform a task to high accuracy with the help of the machine controller; a gradient based ILC algorithm and a projection based ILC method are introduced in Chu (2019) for the point to point (P2P) tasks and the constraint handling problem, respectively; decentralised ILC framework with three model-based implementations are proposed and verified on a wearable stroke rehabilitation technology in Chen and Freeman (2020); collaborative tracking problem with the presence of noise is considered in Shen et al. (2020).

System constraints are widely existing in practice, since they are often related to practical limitations or performance requirements. For example, autonomous vehicle contains several actuators working collaboratively to move forward and each actuator (subsystem) has its own allowable voltage (input) range. During the control process, if one actuator's applied input violates its allowable voltage range, it may damage the actuator and further affect the performance of the autonomous vehicle. However, most existing decentralised ILC methods only consider unconstrained collaborative tracking problem, which lack the capability to handle system constraints. It is worth mentioning that Chu (2019) has considered the constraint handling problem, but the monotonic convergence of the tracking error norm is only guaranteed when the controller satisfies certain conditions.

Motivated by the constraint handling design for consensus tracking problem in Chapter 4, this chapter further considers the constrained collaborative tracking problem. We first propose a constrained ILC algorithm for collaborative tracking using the constrained norm optimal ILC (NOILC) framework. The algorithm has a simple structure but needs a central controller. Then, using the idea of the alternating direction method of multipliers (ADMM), a decentralised constrained ILC algorithm is developed. The resulting ILC algorithms not only guarantee the satisfaction of the system constraints, but also achieve the monotonic convergence of the collaborative tracking error norm to a minimum (possible) solution. Furthermore, the proposed ILC algorithms can be applied to both homogeneous and heterogeneous networks, as well as non-minimum phase systems, which are desirable in practice. Convergence properties of the proposed ILC algorithms are analysed rigorously and numerical examples are presented to verify the algorithms' effectiveness.

The rest of this chapter is organised as follows: Section 8.1 provides the formulation of the system dynamics and system constraints, and then defines the ILC design objective; Section 8.2 proposes a constrained ILC algorithm to solve the collaborative tracking problem; Section 8.3 provides a decentralised constrained collaborative tracking ILC algorithm; Section 8.4 provides numerical examples to demonstrate the effectiveness of the proposed algorithms, and finally, Section 8.5 summaries this chapter.

8.1 Problem Formulation

In this section, we first provide the general formulation of the system dynamics and system constraints, and then introduce the ILC design problem for constrained collaborative tracking of networked dynamical systems.

8.1.1 System Dynamics

Consider a (homogeneous or heterogeneous) networked dynamical system including p subsystems, where i^{th} $(1 \leq i \leq p)$ subsystem's dynamics is denoted using a linear time invariant (LTI), single input single output (SISO), discrete time state space model:

$$\begin{aligned} x_{i,k}(t+1) &= A_i x_{i,k}(t) + B_i u_{i,k}(t), \qquad & x_{i,k}(0) = x_{i,0} \\ y_{i,k}(t) &= C_i x_{i,k}(t), & t \in [0, N] \end{aligned} \tag{8.1}$$

where i is the index of the subsystem; t is the time index; k is the ILC trial index; N is the trial length; $x_{i,k}(\cdot) \in \mathbb{R}^{n_i}$ (n_i is the order of i^{th} subsystem), $u_{i,k}(\cdot) \in \mathbb{R}$, $y_{i,k}(\cdot) \in \mathbb{R}$ are state, input and output of subsystem i on the k^{th} ILC trial, respectively; A_i, B_i, C_i are system matrices with proper dimensions. The networked dynamical systems are required to execute the same task repetitively, i.e., each subsystem's state is reset to the initial condition $x_{i,0}$ at the end of each trial, and the subsystem is required to execute the same control task again.

For the collaborative tracking problem, the output of the networked dynamical system is defined as follows

$$y_k(t) = \sum_{i=1}^{p} y_{i,k}(t) \tag{8.2}$$

and the system is required to track a desired reference trajectory $r(t)$. Define the collaborative tracking error as

$$e_k(t) = r(t) - y_k(t). \tag{8.3}$$

The control design objective of collaborative tracking problem is that all the subsystems work collaboratively to make sure $e_k(t) = 0$. Note that, each subsystem only knows the overall system output $y_k(t)$ but no information of the other subsystems and it has to work independently, which makes the design non-trivial.

In this chapter, we introduce a 'lifted form' representation (see Hatonen et al. (2004) for more details) to facilitate later ILC design. Assuming at least one subsystem's relative degree is unity (i.e., $\exists i$ such that $C_i B_i \neq 0$), the 'lifted' input, output, error, desired reference can then be defined as

$$\begin{aligned} u_{i,k} &= [u_{i,k}(0) \quad u_{i,k}(1) \quad \cdots \quad u_{i,k}(N-1)]^T \in \mathcal{U}_i \\ y_{i,k} &= [y_{i,k}(1) \quad y_{i,k}(2) \quad \cdots \quad y_{i,k}(N)]^T \in \mathcal{Y} \\ e_k &= [e_k(1) \quad e_k(2) \quad \cdots \quad e_k(N)]^T \in \mathcal{Y} \\ r &= [r(1) \quad r(2) \quad \cdots \quad r(N)]^T \in \mathcal{Y} \end{aligned} \tag{8.4}$$

where the input space $\mathcal{U}_i = \mathbb{R}^N$ and the output space $\mathcal{Y} = \mathbb{R}^N$ are defined with inner products and induced norms

$$\begin{aligned} \langle u, v \rangle_{R_i} &= u^T R_i v, \qquad & \|u\|_{R_i} = \sqrt{\langle u, u \rangle_{R_i}} \\ \langle x, y \rangle_Q &= x^T Q y, & \|y\|_Q = \sqrt{\langle y, y \rangle_Q} \end{aligned} \tag{8.5}$$

in which R_i, Q are symmetric positive definite matrices.

Then, the system model (8.1) can be rewritten as

$$y_{i,k} = G_i u_{i,k} + d_i \tag{8.6}$$

where i^{th} subsystem's system matrix G_i is defined as

$$G_i = \begin{bmatrix} C_iB_i & 0 & \cdots & 0 \\ C_iA_iB_i & C_iB_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_iA_i^{N-1}B_i & C_iA_i^{N-2}B_i & \cdots & C_iB_i \end{bmatrix} \tag{8.7}$$

and the response of i^{th} subsystem's initial condition d_i is

$$d_i = \left[C_iA_ix_{i,0}, C_iA_i^2x_{i,0}, C_iA_i^3x_{i,0}, \cdots, C_iA_i^Nx_{i,0}\right]^T \tag{8.8}$$

For notational simplicity, we introduce d, y_k, $\mathbb{G}$, $\vec{u}_k$ as the global vector of d_i, $y_{i,k}$, G_i and $u_{i,k}$, which are defined as

$$\begin{aligned} d &= d_1 + d_2 + \cdots + d_p \\ y_k &= y_1 + y_2 + \cdots + y_p \\ \mathbb{G} &= [G_1 \quad G_2 \quad \cdots \quad G_p] \\ \vec{u}_k &= \left[u_{1,k}^T \quad u_{2,k}^T \quad \cdots \quad u_{p,k}^T\right]^T \in \mathcal{U} \end{aligned} \tag{8.9}$$

where the global input space $\mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2 \times \cdots \times \mathcal{U}_p$, with the inner products and induced norm defined as

$$\langle \vec{u}, \vec{v} \rangle_{\vec{R}} = \vec{u}^T \vec{R} \vec{v}, \qquad \|\vec{u}\|_{\vec{R}} = \sqrt{\langle \vec{u}, \vec{u} \rangle_{\vec{R}}} \tag{8.10}$$

in which $\vec{R} = \text{diag}(R_1, R_2, \cdots, R_p)$, and $\times$ represents the Cartesian product. Then, the overall system model (8.2) can be represented in 'lifted form' as

$$y_k = \mathbb{G}\vec{u}_k + d. \tag{8.11}$$

Based on the above 'lifted form' formulation, the desired objective of collaborative tracking problem can be stated as finding a set of proper individual input $u_{i,k}$ such that the overall system output y_k track the reference r precisely.

8.1.2 System Constraints

In practice, system constraints are widely existing, since they are often related to practical limitations (e.g., individual input saturation) or performance requirements (e.g., constraints of total input energy consumption). Hence, considering the ability to handle system constraints is important when designing a control algorithm. In this chapter, we only consider input constraints for simplicity. The results can be extended to handle other types of constraints (e.g., state constraints, output saturation) without any difficulties. Assuming the individual input u_i is lied in a closed and convex set Ω_i in the input space $\mathcal{U}_i$. Some commonly used examples are defined as follows:

- Input sign constraints:

$$\Omega_i = \big\{u_i \in \mathbb{R}^N : \quad 0 \leq u_i(t)\big\} \tag{8.12}$$

- Input saturation constraints:

$$\Omega_i = \big\{u_i \in \mathbb{R}^N : \quad |u_i(t)| \leq M_i(t)\big\} \tag{8.13}$$

- Input energy constraints:

$$\Omega_i = \left\{u_i \in \mathbb{R}^N : \quad \sum_{t=0}^{N-1} u_i^2(t) \leq M_i\right\} \tag{8.14}$$

- Input amplitude constraints:

$$\Omega_i = \big\{u_i \in \mathbb{R}^N : \quad \lambda_i(t) \leq u_i(t) \leq \mu_i(t)\big\} \tag{8.15}$$

where M_i, λ_i and μ_i are prescribed constraint bounds.

It should be noticed that, each subsystem's input constraint is not necessary to be the same, as commonly seen in practice. To facilitate subsequence ILC design, we define Ω as the global input constraint set, i.e.

$$\Omega = \Omega_1 \times \Omega_2 \times \Omega_3 \times \cdots \times \Omega_p. \tag{8.16}$$

8.1.3 Constrained Iterative Learning Control Design

In this chapter, the ILC design problem for constrained collaborative tracking problem is stated as finding an input updating law in the following form

$$\vec{u}_{k+1} = f(\vec{u}_k, e_k) \tag{8.17}$$

such that

$$\begin{aligned} \vec{u}_k &\in \Omega, && \forall k \geq 0 \\ \lim_{k\to\infty} y_k &= r, && i = 1, 2, \cdots, p \end{aligned} \tag{8.18}$$

Note that, during the whole control process, each subsystem needs to work independently (since it has not access to the other subsystem's information) without violating the input constraints, which makes the design problem non-trivial. To solve this challenging problem, we will develop a novel constrained ILC framework in the following section.

8.2 Constrained ILC Framework for Collaborative Tracking Problem

In this section, we propose a novel constrained ILC framework to solve the ILC design problem (8.18) and then analyse its convergence properties rigorously.

8.2.1 Algorithm Description

Inspired by the constrained design for networked dynamical systems (please refer to Algorithm 4.1 in Chapter 4 for more details), we propose the following NOILC framework to solve the constrained collaborative tracking problem:

Algorithm 8.1. *For any $\vec{u}_0 \in \Omega$, the input series $\{\vec{u}_{k+1}\}_{k\geq 0}$ defined as follows*

$$\vec{u}_{k+1} = \arg\min_{\vec{u}_{k+1}\in\Omega} \left\{ \|e_{k+1}\|_Q^2 + \|\vec{u}_{k+1} - \vec{u}_k\|_{\vec{R}}^2 \right\} \tag{8.19}$$

where $e_{k+1} = r - y_{k+1} = e_k - \mathbb{G}(\vec{u}_{k+1} - \vec{u}_k)$, iteratively solve the constrained collaborative tracking problem, i.e.

$$\begin{aligned} \vec{u}_k &\in \Omega, && \forall k \geq 0 \\ \lim_{k\to\infty} y_k &= r, && i = 1, 2, \cdots, p \end{aligned} \tag{8.20}$$

Remark 8.1. *Note that the convergence speed of Algorithm 8.1 will be affected by different choices of weighting matrices Q, $\vec{R}$ in a similar way as that of the standard norm optimal ILC algorithm, i.e., a smaller R leads to faster convergence speed. For more details, please refer to Amann et al. (1996a).*

8.2.2 Convergence Properties of Algorithm 8.1

The convergence analysis of Algorithm 8.1 will be discussed in two cases due to the existence of the constraint set Ω.

8.2.2.1 Perfect Collaborative Tracking is Possible

When the constraint set Ω is large enough to cover the possible solution, perfect collaborative tracking is possible. Algorithm 8.1 has appealing convergence properties, as shown in the following theorem.

Theorem 8.1. *Given any $\vec{u}_0 \in \Omega$ and associated collaborative tracking error e_0, Algorithm 8.1 guarantees the generated input sequence satisfy the constraint requirements, i.e.*

$$\vec{u}_{k+1} \in \Omega, \qquad \forall k \geq 0 \tag{8.21}$$

and the collaborative tracking error norm converges monotonically to zero, i.e.

$$\|e_{k+1}\|_Q \leq \|e_k\|_Q, \qquad \lim_{k\to\infty} e_k = 0. \tag{8.22}$$

Consequently, the networked dynamical system achieves perfect collaborative tracking and the corresponding input converges as $k \to \infty$, i.e.

$$\begin{aligned} \lim_{k\to\infty} y_k &= r, \\ \lim_{k\to\infty} u_{i,k} &= u_i^*, \qquad i = 1, 2, \cdots, p \end{aligned} \tag{8.23}$$

Proof. The proof of convergence follows from the result in Chapter 4 by formulating the problem into a successive projection framework. In more details, we define two closed, convex sets K_1 and K_2 in $H = \mathcal{U} \times \mathcal{Y}$ as follows

- $K_1 = \{(e, \vec{u}) \in H : e = r - \mathbb{G}\vec{u} - d, \vec{u} \in \Omega\}$;
- $K_2 = \{(e, \vec{u}) \in H : e = 0\}$

with the inner product and the associated induced norm for H defined as

$$\begin{aligned} \langle (e, \vec{u}), (z, \vec{v}) \rangle &= e^T Q z + \vec{u}^T \vec{R} \vec{v} \\ \|(e, \vec{u})\| &= \sqrt{\langle (e, \vec{u}), (e, \vec{u}) \rangle} \end{aligned} \tag{8.24}$$

Then, the property of (8.21) is a direct consequence of solving the problem (8.19) (by using Theorem 4.1). For the optimisation problem (8.19), when perfect collaborative tracking is achievable, the non-optimal input choice $\vec{u}_{k+1} = \vec{u}_k$ yields a suboptimal solution

$$\|\vec{u}_{k+1} - \vec{u}_k\|_{\vec{R}}^2 + \|e_{k+1}\|_Q^2 \leq \|e_k\|_Q^2 \tag{8.25}$$

Note that $\|\vec{u}_{k+1} - \vec{u}_k\|_{\vec{R}}^2$ is non-negative, hence

$$\|e_{k+1}\|_Q^2 \leq \|e_k\|_Q^2, \quad k = 0, 1, \cdots, \infty \tag{8.26}$$

Note that Algorithm 8.1 iteratively finds a point of the intersection of sets K_1 and K_2. When $K_1 \cap K_2 \neq \emptyset$, the crosspoint is $(0, \vec{u}^*)$, hence

$$\lim_{k\to\infty} \|e_k\|_Q^2 = 0, \quad \lim_{k\to\infty} u_k = u^* \tag{8.27}$$

which follows that

$$\lim_{k\to\infty} e_k^T Q e_k = 0 \tag{8.28}$$

Note that matrix Q is positive definite, hence, $e_k = 0$ as $k \to \infty$. That completes the proof. □

8.2.2.2 Perfect Collaborative Tracking is Impossible

When the constraint set Ω does not cover the possible solution, perfect collaborative tracking is impossible. In this case, Algorithm 8.1 guarantees the monotonic convergence of the collaborative tracking error norm to a minimum (possible) solution, as shown in the following theorem.

Theorem 8.2. *Given any $\vec{u}_0 \in \Omega$ and associated collaborative tracking error e_0, Algorithm 8.1 guarantees the generated input sequence satisfy the constraint requirements, i.e.*

$$\vec{u}_{k+1} \in \Omega, \qquad \forall k \geq 0 \tag{8.29}$$

and the convergence of the collaborative tracking error norm is monotonic, i.e.

$$\|e_{k+1}\|_Q \leq \|e_k\|_Q, \qquad k \geq 0 \tag{8.30}$$

Furthermore, the global input $\vec{u}$ converges to the optimal solution $\vec{u}_s^*$ for the following optimisation problem

$$\vec{u}_s^* = \arg\min_{\vec{u}\in\Omega} \|e^*\|_Q \tag{8.31}$$

Proof. The proof of the monotonic convergence of the collaborative tracking error norm is similar to the proof in Theorem 8.1, hence it is omitted here.

According to Theorem 4.1, when perfect collaborative tracking is unachievable, Algorithm 8.1 converges to $\vec{u}_s^*$, where $k_1 = (e, \vec{u}) \in K_1$ and $k_2 = (0, \vec{u}_s^*) \in K_2$ defining the minimum distance of two sets, and it is the solution for the following optimisation problem

$$(k_1, k_2) = \arg \min_{k_1\in K_1, k_2\in K_2} \|k_1 - k_2\|^2 \tag{8.32}$$

From the definition of K_1 and K_2, solving problem (8.32) is equivalent to solving the following optimisation problem

$$(\vec{u}, \vec{u}_s^*) = \arg \min_{\vec{u}\in\Omega, \vec{u}_0} \left\{ \|r - \mathbb{G}\vec{u} - d\|_Q^2 + \|\vec{u} - \vec{u}_0\|_{\vec{R}}^2 \right\} \tag{8.33}$$

and hence, Algorithm 8.1 converges to $\vec{u}_s^*$, which is defined as

$$\begin{aligned}\vec{u}_s^* &= \arg\min_{\vec{u}\in\Omega,\vec{u}_0}\left\{\|r-\mathbb{G}\vec{u}_0-d\|_Q^2+\|\vec{u}-\vec{u}_0\|_{\vec{R}}^2\right\}\\ &= \arg\min_{\vec{u}\in\Omega}\left\{\min_{\vec{u}_0}\|r-\mathbb{G}\vec{u}_0-d\|_Q^2+\|\vec{u}-\vec{u}_0\|_{\vec{R}}^2\right\}\end{aligned} \tag{8.34}$$

Note that $\vec{u}_0=\vec{u}$ is the solution of the inner minimization, and hence

$$\vec{u}_s^* = \arg\min_{\vec{u}\in\Omega}\|r-\mathbb{G}\vec{u}^*-d\|_Q^2 \tag{8.35}$$

Since matrix Q is positive definite and matrix $\mathbb{G}$ has full rank, the performance index is convex. It should be noticed that, the constraint is also convex, hence (8.35) has an optimal solution. That completes the proof. □

Theorems 8.1 & 8.2 show that Algorithm 8.1 guarantees both the satisfaction of the constraints and the monotonic convergence of the tracking error norm to a minimum solution (in particular, zero tracking error when perfect collaborative tracking is possible), which is desirable in practice. However, using a central controller (to solve a constrained quadratic programming problem) may not be possible in practice (especially for large-scale networked dynamical system). Hence, we will propose a decentralised ILC algorithm in the next session to address this problem.

8.3 Decentralised Constrained Collaborative Tracking ILC Algorithm

In this section, we use the idea of 'sharing' formulation in ADMM to provide a decentralised constrained collaborative tracking ILC Algorithm. In the following, the general idea of 'sharing' formulation in ADMM will be reviewed.

8.3.1 The Alternating Direction Method of Multipliers

ADMM is a powerful distributed optimisation method with nice convergence properties: the convergence of the objective is guaranteed under very mild conditions, which in contrast to many other distributed methods, e.g., dual decomposition (Boyd et al., 2011). The so called 'sharing' problem in ADMM has many similarities with the collaborative tracking problem considered in this paper, which will be reviewed in the following.

For the 'sharing' problem, it is defined in the following form

$$\text{minimize} \quad \sum_{i=1}^{p} f_i(u_i) + g(\sum_{i=1}^{p} u_i) \tag{8.36}$$

where $u_i \in \mathbb{R}^N$ denotes the local variable.

Then, applying the idea of ADMM in Boyd et al. (2011), the 'sharing' problem (8.36) can be written into the following form

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{p} f_i(u_i) + g(\sum_{i=1}^{p} z_i) \\ \text{subject to} \quad & u_i - z_i = 0, \quad i = 1, \cdots, p \end{aligned} \tag{8.37}$$

where $z_i \in \mathbb{R}^N$ denotes the global variable component.

For problem (8.37), its augmented Lagrangian is defined as

$$L_\rho(u_i, z, \gamma_i) = \sum_{i=1}^{p} f_i(u_i) + g(\sum_{i=1}^{p} z_i) + \frac{\rho}{2} \sum_{i=1}^{p} \|u_i - z_i + \gamma_i\|^2 \tag{8.38}$$

where ρ is the penalty parameter, γ_i is the scaled dual value. For problem (8.37), ADMM iteratively performs the following steps to find the solution (more details can be found in Boyd et al. (2011))

$$u_i^{q+1} = \arg\min_{u_i} \left[f_i(u_i) + \frac{\rho}{2} \|u_i - u_i^q + \overline{u}^q - \bar{z}^q + \gamma^q\|^2 \right] \tag{8.39}$$

$$\bar{z}^{\,q+1} = \arg\min_{\bar{z}} \left[g(p\bar{z}) + \frac{\rho}{2} p \|\overline{u}^{q+1} - \bar{z} + \gamma^q\|^2 \right] \tag{8.40}$$

$$\gamma^{\,q+1} = \gamma^{\,q} + \overline{u}^{q+1} - \bar{z}^{q+1}. \tag{8.41}$$

where $\overline{u} = \frac{1}{p} \sum_{i=1}^{p} u_i$, $\bar{z} = \frac{1}{p} \sum_{i=1}^{p} z_i$ and q is the ADMM iteration number .

Note that, the z-update step only requires solving a problem in N variables (for $i = 1, 2, \ldots, p$) and each subsystem's scaled dual variables γ_i are the same (and hence they are replaced by a single variable γ), which greatly reduces the computational complexity. Note that, the above algorithm is guaranteed to converge to the optimal solution for any $\rho > 0$, which is appealing in practice.

8.3.2 Decentralised Implementation of Algorithm 8.1

Using the 'sharing' problem formulation, a decentralised implementation for Algorithm 8.1 can be developed in the following. At trial $k + 1$, the optimal solution of input updating law (8.19) can be found by solving the following optimisation problem:

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{p} f_{i,k+1}(u_{i,k+1}) + g_{k+1}\left(\sum_{i=1}^{p} z_{i,k+1}\right) \\ &\text{subject to} \quad G_i u_{i,k+1} - z_{i,k+1} = 0, \quad i = 1, \cdots, p \end{aligned} \tag{8.42}$$

where $f_{i,k+1}(u_{i,k+1})$ is defined as

$$\begin{aligned} f_{i,k+1}(u_{i,k+1}) &= \|u_{i,k+1} - u_{i,k}\|_{R_i}^2 \\ \mathbf{dom}\, f_{i,k+1} &= \{u_{i,k+1} |\ u_{i,k+1} \in \Omega_i\} \end{aligned} \tag{8.43}$$

and $g_{k+1}(\sum_{i=1}^{p} z_{i,k+1})$ is defined as

$$\begin{aligned} g_{k+1}(\sum_{i=1}^{p} z_{i,k+1}) &= \|e_k + y_k - \sum_{i=1}^{p} z_{i,k+1}\|_Q^2 \\ \mathbf{dom}\, z_{i,k+1} &= \mathbb{R}^N \end{aligned} \tag{8.44}$$

Now, applying ADMM steps to solve (8.42), yields the following decentralised implementation:

Algorithm 8.2. *At ILC trial $k+1$, the input $u_{i,k+1}$ obtained distributively using the following ADMM steps*

$$u_{i,k+1}^{q+1} = \arg \min_{u_{i,k+1} \in \Omega_i} \left[\frac{\rho}{2} \|G_i u_{i,k+1} - G_i u_{i,k+1}^q + \overline{Gu}^q - \overline{z_{k+1}}^q + \gamma_{k+1}^q\|^2 + \|u_{i,k+1} - u_{i,k}\|_{R_i}^2\right] \tag{8.45}$$

$$\bar{z}_{k+1}^{q+1} = (pQ + \frac{\rho}{2} I_N)^{-1} \left[\frac{\rho}{2}(\gamma_{k+1}^q + \overline{Gu}^{q+1}) + Q(e_k + y_k)\right] \tag{8.46}$$

$$\gamma_{k+1}^{q+1} = \gamma_{k+1}^q + \overline{Gu}^{q+1} - \overline{z_{k+1}}^{q+1} \tag{8.47}$$

where $\overline{Gu}^{q+1} = \frac{1}{p}\sum_{i=1}^{p} G_i u_{i,k+1}^{q+1}$, provides a decentralised implementation for optimisation problem (8.19).

8.3.3 Decentralised Constrained ILC Algorithm

Using distributed implementation Algorithms 8.2 to implement the centralised Algorithm 8.1, yields the decentralised constrained ILC Algorithm 8.3 for the high performance collaborative tracking problem. A flow chart is shown in Figure 8.1 to describe the information flow during the control process. In Algorithm 8.3, each subsystem updates its own input independently, which is desirable in practice. The only information needed, in addition to the collaborative tracking error, are $\overline{Gu}^{q+1}$, $\overline{z_{k+1}}^{q+1}$, γ_{k+1}^{q+1}. Note that, these information are easy to share, since they only require simple calculation of the measurement unit, which also provide the tracking error information. By applying

Algorithm 8.3. *Decentralised Constrained Algorithm for Collaborative Tracking*

```
Input: Desired reference trajectory $r$; system matrices $A_i$, $B_i$, $C_i$; maximum ILC
    trials $k_{max}$; maximum ADMM iterations $q_{max}$; penalty parameter $\rho$
Output: Optimal input solution $\vec{u}_{k_{max}}$
1:  Initialization: Set the ILC trial index $k = 0$
2:  For: $k = 0$ to $k_{max}$
3:    For: $q = 0$ to $q_{max}$
4:      For: $i = 1$ to $p$
5:        Each subsystem receives information
6:        Each subsystem updates $u_{i,k+1}^{q+1}$ using (8.45)
7:      End for
8:      Measurement unit updates $\bar{z}_{k+1}^{q+1}$, $\gamma_{k+1}^{q+1}$ using (8.46), (8.47), and broadcasts
        them to all subsystems
9:    End for
10: Each subsystem applies $u_{i,k+1}$ to the system and measurement unit
    records $y_{k+1}$, $e_{k+1}$
11: End for
12: Return: Optimal input solution $\vec{u}_{k_{max}}$
```

the above idea, Algorithm 8.3 can be calculated in a fully decentralised manner, which is desirable in practice.

Remark 8.2. *Note that, ADMM requires infinite iteration number to approach the centralised result in theory. However, ADMM is usually efficient in practice, i.e., a small iteration number is enough to approximate the centralised result in most cases. Later simulations will verify this phenomenon.*

8.4 Numerical Examples

In this section, we provide numerical examples to verify the proposed algorithms' effectiveness. Consider a heterogeneous, non-minimum phase, over-damping, seven-agent networked dynamical systems. Each subsystem's dynamics is defined as

$$A_i = \begin{bmatrix} 0 & 1 \\ -\tau_i & -6 \end{bmatrix} \qquad B_i = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad C_i = \begin{bmatrix} -1 \\ 5 \end{bmatrix}^T$$

in which $\tau_i = i$, $i = 1, 2, \cdots, 7$.

The trial length is assumed to be $3s$ with the sampling time $T_s = 0.05s$ (sampled using a zero-order hold), and for simplicity, each subsystem's initial condition $x_{i,0}$ is set to be 0. The desired reference trajectory is defined as

$$r(t) = sin(\pi(T_s t - 1)). \tag{8.48}$$

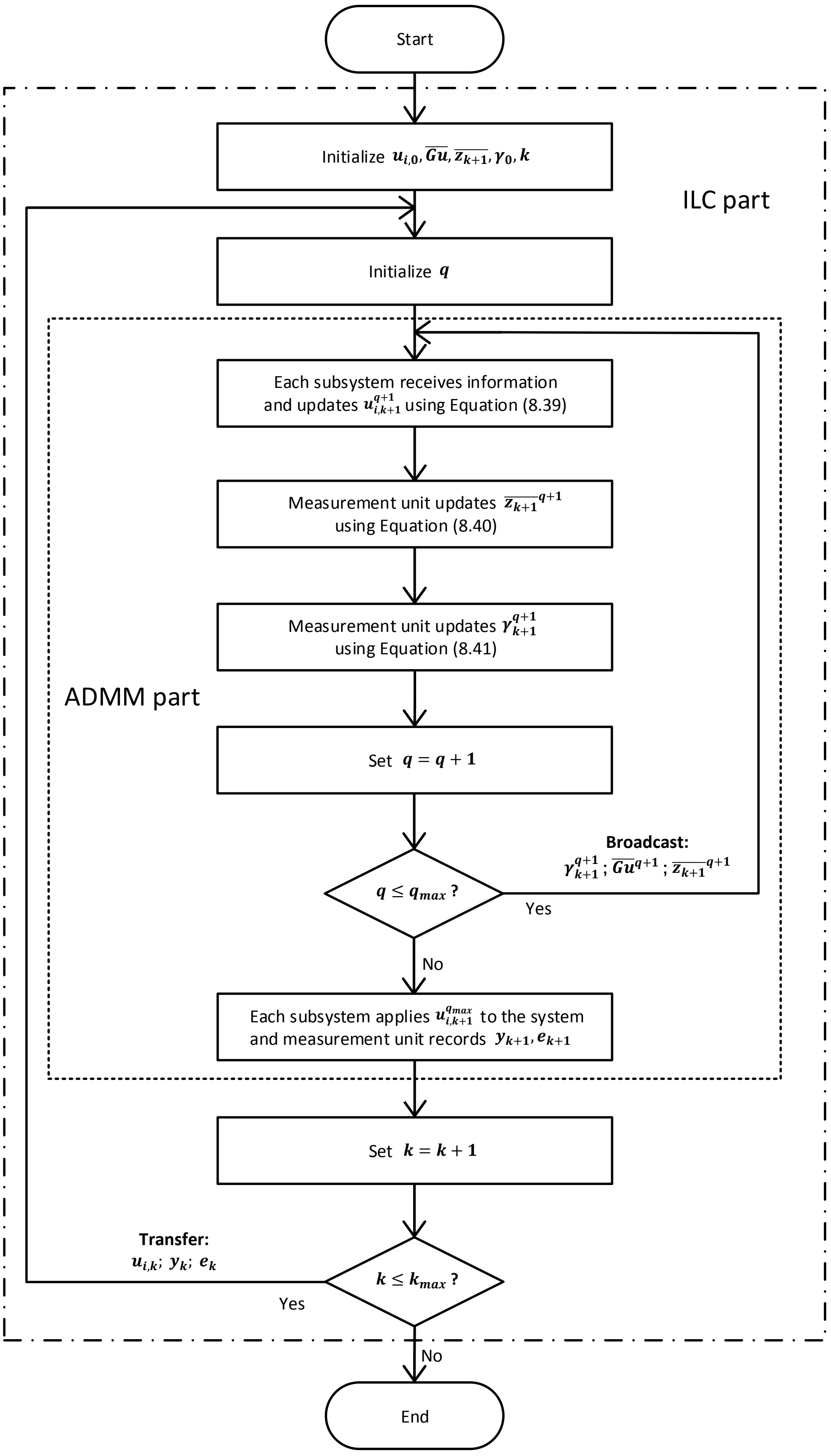
Start
Initialize $u_{i,0}, \overline{Gu}, \overline{z_{k+1}}, \gamma_0, k$
ILC part
Initialize q
Each subsystem receives information and updates $u_{i,k+1}^{q+1}$ using Equation (8.39)
Measurement unit updates $\overline{z_{k+1}}^{q+1}$ using Equation (8.40)
Measurement unit updates γ_{k+1}^{q+1} using Equation (8.41)
ADMM part
Set $q = q + 1$
Broadcast: γ_{k+1}^{q+1} ; $\overline{Gu}^{q+1}$; $\overline{z_{k+1}}^{q+1}$
$q \le q_{max}$?
Yes
No
Each subsystem applies $u_{i,k+1}^{q_{max}}$ to the system and measurement unit records y_{k+1}, e_{k+1}
Set $k = k + 1$
Transfer: $u_{i,k}$; y_k; e_k
$k \le k_{max}$?
Yes
No
End

Figure 8.1: The flow chart of distributed Algorithm 8.3

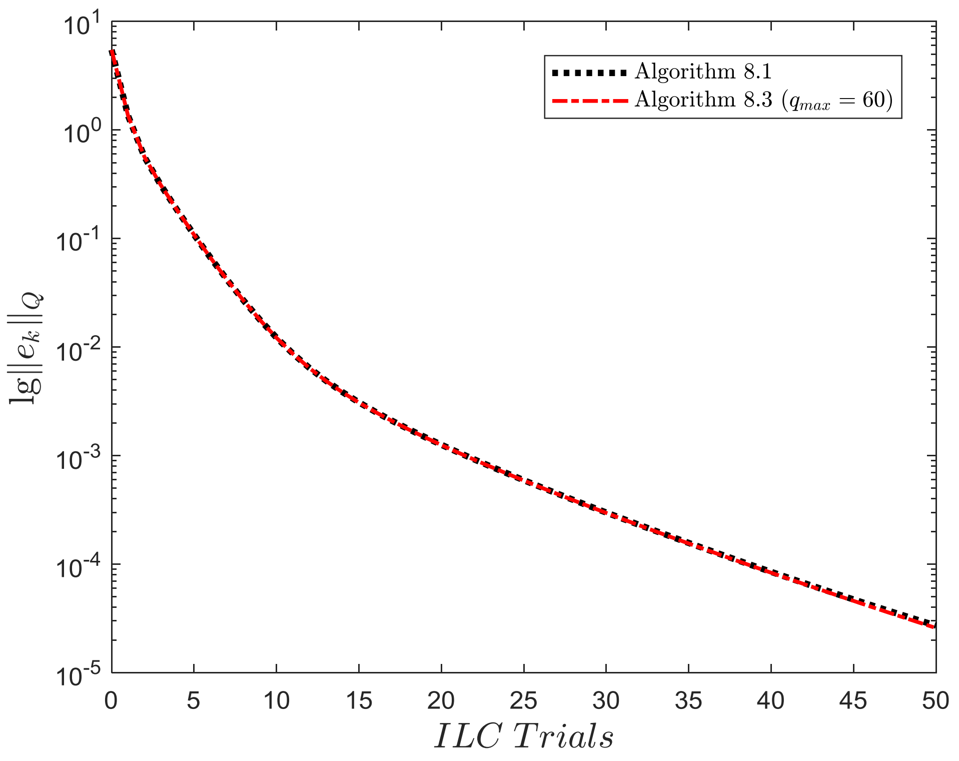

Figure 8.2: Convergence of collaborative tracking error norm over 50 trials (when perfect collaborative tracking is achievable)

In the following, we first consider the case that perfect collaborative tracking is achievable.

8.4.1 Perfect Collaborative Tracking is Achievable

To make sure the perfect collaborative tracking is achievable, each subsystem's individual constraint set is selected large enough as follows to cover the solution:

$$\Omega_i = \{u_i \in \mathbb{R}^N : -0.25 \leq u_i(t) \leq 0.25\} \tag{8.49}$$

In this case, we set the maximum ADMM iteration number $q_{max} = 60$, penalty parameter $\rho = 1$, and the weighting matrices $Q = R_i = I_N$ for simplicity. Figure 8.2 shows the convergence behaviour of both centralised and decentralised implementation over 50 ILC trials. As shown in the figure, the decentralised result is almost identical with the centralised solution, demonstrating that a small iteration number is usually sufficient for ADMM to approach the centralised result in most of the cases. It can also be seen from the figure that the collaborative tracking error norm converges monotonically to zero, which verifies Theorem 8.1.

Figures 8.3 and 8.4 show the input of seven subsystems and the output of the whole

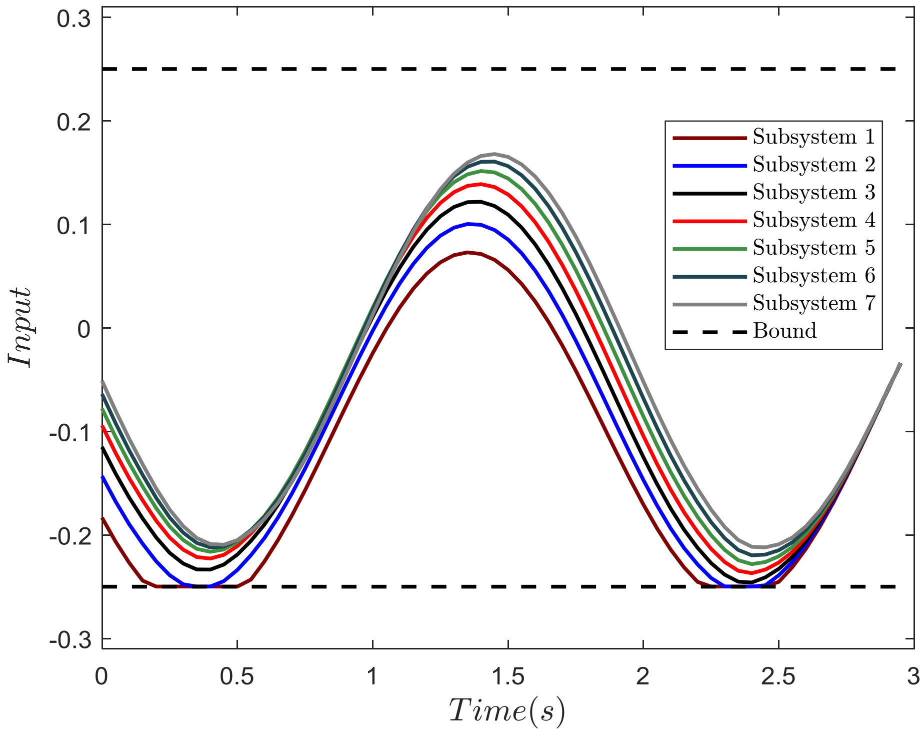

Figure 8.3: Input of different subsystems on the 50th trial (when perfect collaborative tracking is achievable)

networked dynamical systems on the 50th ILC trial. These figures demonstrate that when perfect collaborative tracking is possible, Algorithms 8.1 & 8.3 not only guarantee the satisfaction of the input constraint, but also achieve the desired collaborative tracking objective, which further verifies Theorem 8.1.

8.4.2 Perfect Collaborative Tracking is Unachievable

In this case, we change the input constraint set into a smaller one (and hence there is not possible solution in the set) as

$$\Omega_i = \{u_i \in \mathbb{R}^N : -0.20 \le u_i(t) \le 0.20\}. \tag{8.50}$$

To make comparison with the case that perfect collaborative tracking is possible, we consider the same parameter setting as the previous case. Figure 8.5 shows the convergence performance of the collaborative tracking error norm over 50 ILC trials. Again, the decentralised result and the centralised solution are almost indistinguishable in the figure, which conform to our exception. Although the collaborative tracking error norm cannot converge to zero in this case, it will converge monotonically to the minimum solution $\|e^*\|_{\vec{Q}} = 0.1824$ (that is obtained from (8.31)), which verifies Theorem 8.2.

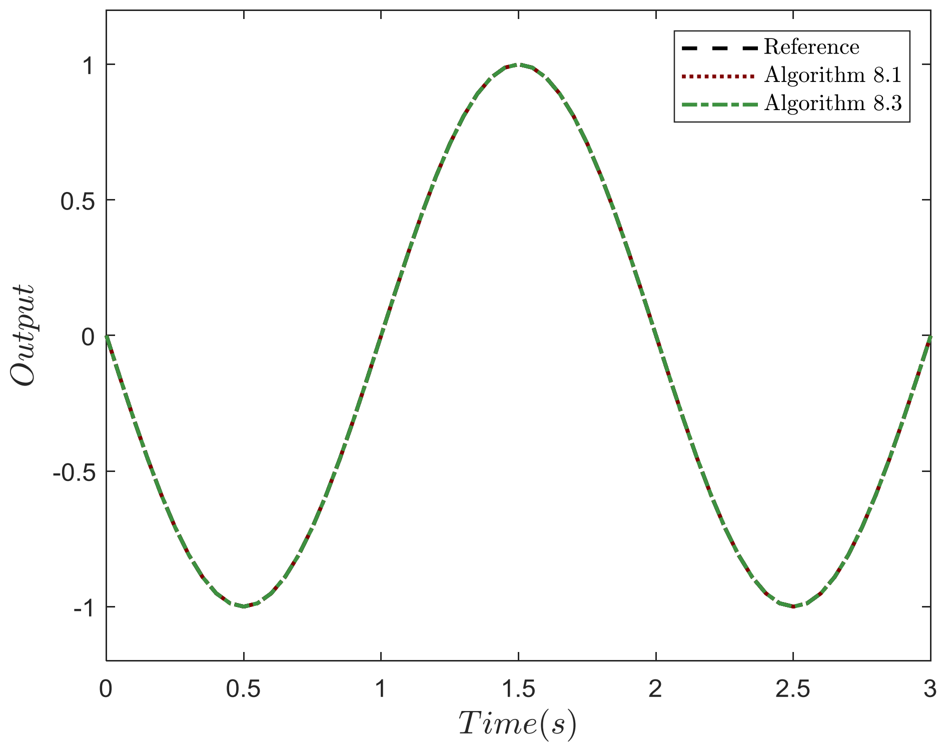

Figure 8.4: Output of the overall system on the 50th trial (when perfect collaborative tracking is achievable)

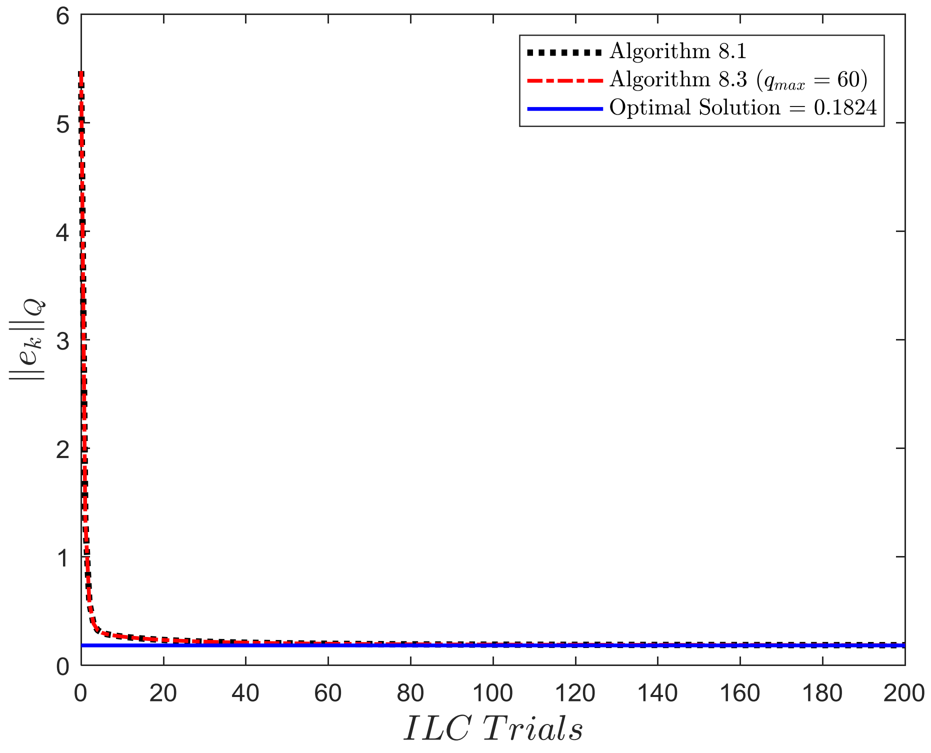

Figure 8.5: Convergence of collaborative tracking error norm over 200 trials (when perfect collaborative tracking is impossible)

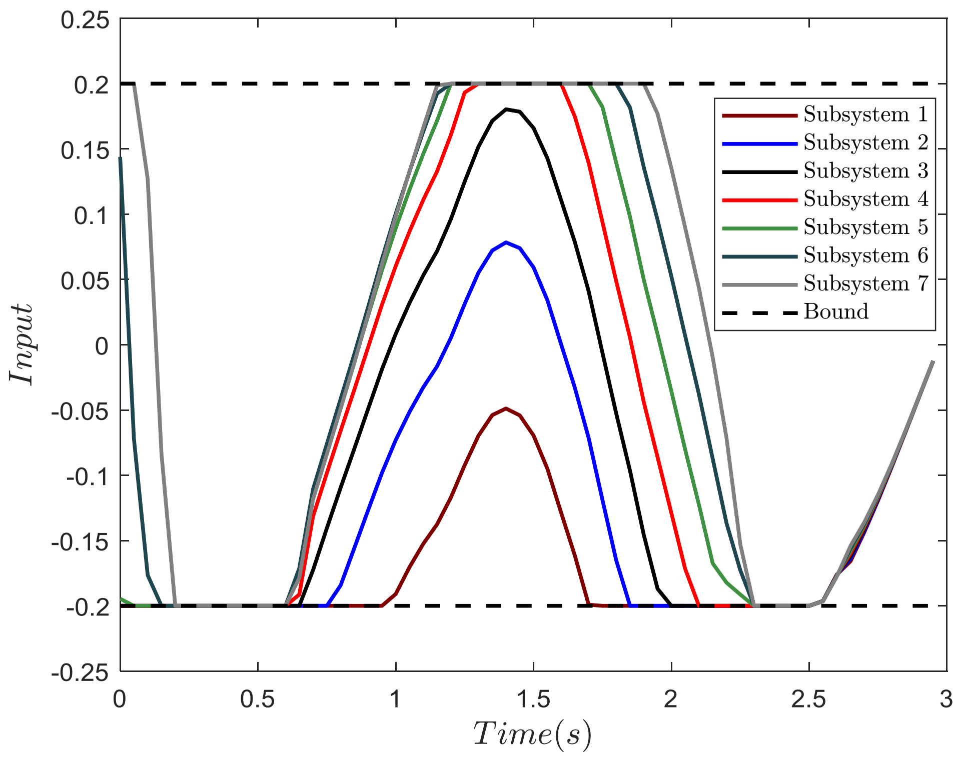

Figure 8.6: Input signal of different subsystems on the 200th trial (when perfect collaborative tracking is unachievable)

All of the subsystems' input are shown in Figure 8.6, which demonstrates that even when the perfect collaborative tracking is impossible, each subsystem can still generate its input within the constraint set. Figure 8.7 shows the output of the overall system on the 200th ILC trial, and it demonstrates that Algorithms 8.1 & 8.3 can return a good approximation to the reference trajectory. We also investigate the effect of weighting matrices Q and R_i, and the results conform to Remark 8.1.

8.5 Summary

In this chapter, we propose novel optimisation-based ILC algorithms for the constraint handling problem in collaborative tracking of networked dynamical systems. The resulting constrained ILC framework can handle the system constraint while maintaining the monotonic convergence of the collaborative tracking error norm to a minimum value (in particular, zero tracking error when perfect collaborative tracking is achievable). Moreover, the proposed framework can be applied to both homogeneous and heterogeneous networks, as well as non-minimum phase systems, which has strong applicability in practice. Two algorithms are proposed. The first one has a centralised structure. Noting the fact that the central controller is not always possible in practice, we develop a decentralised collaborative tracking ILC algorithm (using the 'sharing' problem formulation

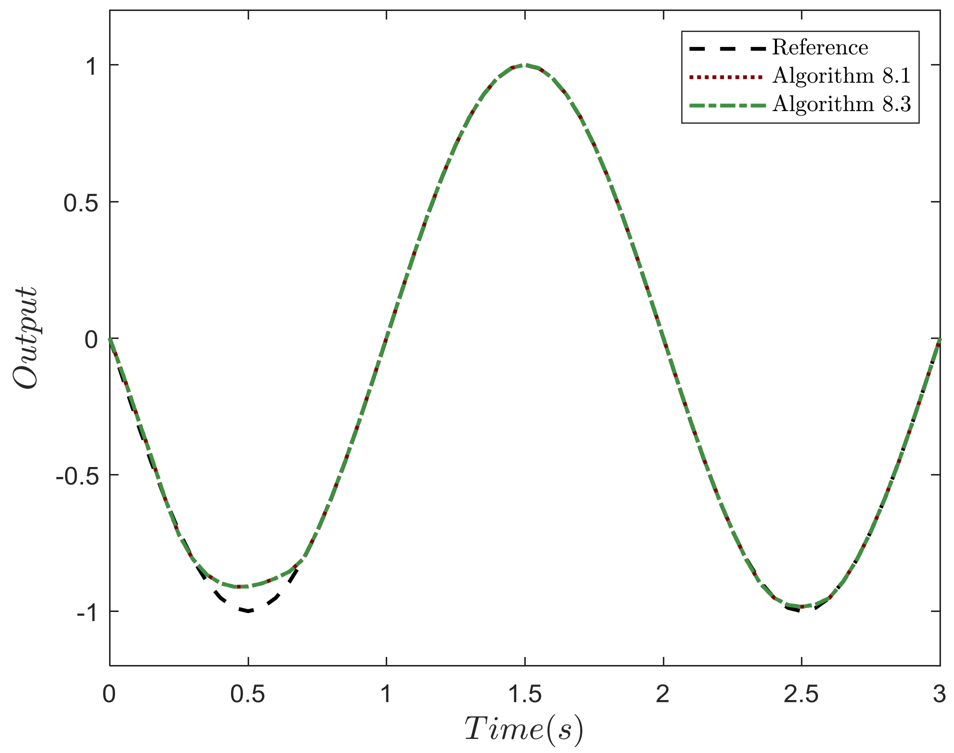

Figure 8.7: Output of the overall system on the 200^{th} trial (when perfect collaborative tracking is unachievable)

in ADMM), which allows the resulting algorithm to be applied to large-scale networked dynamical systems. In next chapter, we will summarises what have been done in this thesis and what could be done in the future.

Chapter 9

Conclusion and Future Work

This chapter summaries the thesis and points out the direction for future research.

9.1 Summary and Conclusion

Networked dynamical systems have attracted increasingly more attention during the last few decades since they can handle difficult/dangerous control tasks that are impossible to be achieved by the single agent system. Benefiting from the vast potential, networked dynamical systems have found many applications in different areas, e.g., traffic networks, collaborative micro-robots, unmanned aircraft vehicles and smart grids. Among all the applications of networked dynamical systems, there exists a class of networked dynamical systems performing repetitive tasks and requiring high control performance. However, most of the existing traditional control methods have difficulty controlling the above networked system: the traditional centralised algorithm has poor scaling properties, and it is usually computationally expensive when the size of networked system is large; the distributed design often focuses on asymptotic (steady state) behaviour, achieving high precision tracking for dynamic reference from the initial state largely remains open. Moreover, they all require highly accurate model information (which is difficult and expensive to obtain in practice) to achieve the high performance requirement.

To avoid the requirement of a highly accurate model, iterative learning control (ILC) has recently been used for high performance networked dynamical systems. ILC updates the input by learning from the data (e.g. input and tracking error) collected from previous executions of the same task. These data contain (explicitly) the system model information and therefore learning from the data avoids the use of an accurate system model. Benefiting from the learning feature, a number of ILC algorithms have been investigated for networked dynamical systems working in a repetitive manner. However, existing ILC algorithms have some limitations: (1) the majority ILC algorithms cannot be applied to large scale networked dynamical systems and none of the existing algorithms can deal with a dynamically growing of networks during the control process; (2) monotonic convergence of the tracking error norm (which is desired in practice) is either not guaranteed or guaranteed only when the controller satisfies certain conditions; (3) they have not considered the constraints handling problem (that is widely existing in practice) in networked dynamical systems; (4) general point-to-point (P2P) tasks, which focus on the tracking performance on the intermediate time instants, have not been considered in existing ILC article for networked dynamical systems.

This thesis addresses the above limitations by introducing novel optimization-based ILC design algorithms for high performance networked dynamical systems working in a repetitive manner. In particular, the main contributions of this thesis are summaries as follows:

- The design of optimisation-based ILC algorithms for three control problems (i.e., consensus tracking, formation control, and collaborative tracking) in networked dynamical systems. The resulting frameworks have nice convergence properties and certain degree of robustness against the model uncertainty.
- The derivation of distributed/decentralised implementations for the proposed ILC frameworks, which allows the proposed frameworks to be applied to large scale networked dynamical systems and have great scalability to a dynamically growing network.
- The exploitation of ILC design algorithms for constraints handling problem of networked dynamical systems. The resulting algorithms guarantee the system constraints are always satisfied during the control process and the monotonic convergence of the tracking error norm to a minimum (possible) solution.
- The extension of the proposed ILC design frameworks to solve P2P task, with the convergence and robustness properties remaining unchanged.

Although all the proposed algorithms have shown great performance for high performance networked dynamical systems working in a repetitive manner, the present work can be further developed in the future, which are summarised in the next section.

9.2 Future Works

Based on what has been done in this thesis, the following topics will be of interest for future research.

9.2.1 Robustness Analysis of the Proposed Algorithms (Estimated Time: 10 Months)

Due to the time limit, we have only analysed the robustness properties of some of the distributed ILC algorithms. However, the robustness analysis of other extended ILC algorithms remains open. For different assuming conditions (e.g., switching topologies, the existence of system constraints, and different model uncertainty type), it will lead to different robustness analyses in theory. To ensure the algorithms are applicable in practice, we will investigate the robustness properties of all the proposed algorithms in the future.

For all the ILC algorithms, their error evolution can be written into the following form

$$e_{k+1} = Le_k \tag{9.1}$$

and the convergence condition is that

$$\rho(L) < 1. \tag{9.2}$$

This condition should always hold for a robust ILC algorithm. Hence, this condition could be seen as a starting point when analysing the robustness for algorithms with different assumptions.

As shown in Chapter 3, we have successfully used the time domain method in Owens (2016) to derive the robustness condition in our situation. Hence, this robustness method will be used for further robustness analysis of the other control problems in networked dynamical systems. After robustness is rigorously analysed, we will test the robustness condition on the simulation platform (under model uncertainty generating using the Monte Carlo method).

9.2.2 ILC for Multi Input Multi Output, Continuous Time System (Estimated Time: 10 Months)

In this thesis, all the ILC frameworks we design are established based on the discrete-time, single input single output (SISO) networked dynamical system. It has been shown that the proposed algorithms show great performance both on convergence and robustness properties. However, the extension to continuous time, multi input multi output (MIMO) is not-trivial because of the following potential risks:

- Possibly $\rho(L) = 1$ in the design. For a continuous time system, the convergence condition $\rho(L) < 1$ is almost impossible to satisfy. The spectrum radius of the learning operator need to be carefully analysed in order to establish the convergence of tracking error.

- Additional computational complexities. Given an example, if we want to use the feedback and feedforward implementation in the ILC design, then it will involve the calculation of the differential Riccati Equation for a continuous time system, which creates more computational complexities.

- Mixed rank problem may arise in the MIMO setting. For a MIMO system with heterogeneous networks (i.e., each subsystem's dynamics is different), there may exist a situation which some of the subsystem's dynamics G_i is singular and leads to unachievable of the control target.

The above issue needs to be addressed when we extend the results to the continuous time, MIMO case. In Owens (2016), there have some answers to these questions and we will investigate them in the future.

9.2.3 ILC for Nonlinear Networked Dynamical System (Estimated Time: 10 Months)

In this thesis, the problem formations and results are established on the linear networked dynamical system. However, the dynamics of each subsystem is often nonlinear in practice, e.g., 6 degree-of-freedom Quadrotor (Najm and Ibraheem, 2019), satellite (Ahn et al., 2010). To make sure the proposed design is applicable to most practical applications, the investigation of ILC for nonlinear networked dynamical systems is required in the future.

For nonlinear ILC design, it is much more difficult to analysis the system behaviour. For heterogeneous network, each nonlinear subsystem's dynamics is different in its own way and it is non-trivial to analyse the system's convergence and robustness properties. In the literature, there exist some nonlinear ILC designs for networked dynamical systems using different approaches, e.g., contraction mapping (Hui et al., 2020; Yang et al., 2016), Newton method (Lin and Owens, 2007; Lin et al., 2006), Lyapunov function (Jin, 2016; Shen et al., 2019). The investigation of these methods can be seen as a starting point. We will verify these methods' effectiveness on the networked dynamical systems in the future.

In addition, all the distributed/decentralised ILC algorithms proposed in this thesis are established on the Hilbert space, allowing the algorithms and results to be applied to more widely cases (including nonlinear networked dynamical systems). The extension from linear system to nonlinear system is a very interesting research direction and it is worthwhile to further study.

9.2.4 Experiment Verification using a Quadrotor Platform (Estimated Time: 1 Year)

Currently, the proposed algorithms are verified based on Matlab simulation, experiment verification of the proposed algorithms is necessary in the future. Quadrotor platform is commonly used in the experiment verification of networked dynamical systems. However, most of the practical swarm Quadrotors are controlled manually (through remote-control), which cannot autonomously achieve the desired high performance requirement. Even considering the recent popularity of reinforcement learning (RL) Quadrotors (which only existing in theory), they will meet difficulties in parameter tuning and stability.

The proposed algorithms have shown great convergence and robustness properties both in theory and simulation, and they do not require a highly accurate model to achieve the desired objective, which is definitely very competitive in practice. Hence, we will build up a multiple Quadrotor test platform to verify the effectiveness of proposed algorithms and improve the algorithms to make sure they meet the practical requirements (e.g, constraint handling ability, energy efficiency).

The potential applications of our proposed algorithms include the high performance task with repetitive manner, e.g., air logistics or urban surveillance. These applications are all the important direction of future intelligence and hence the proposed algorithms will have high commercial value. This future work forms an important part of our future research, which provides evidence for the proposed algorithms to be applied to real world networked dynamical systems.

9.2.5 Model Free ILC Design (Estimated Time: 1 Year)

In this thesis, all the ILC frameworks we design are model based. Generally speaking, the explicit use of the (not necessarily accurate) model information shows appealing convergence properties. However, model information may be expensive and difficult to obtain in some practical applications.

Reinforcement learning (RL) is a completely model free method that learns an optimal action policy to maximise some performance (reward). By constantly interacting with the environment, applying policy, receiving and learning from the reward, RL can modify the action policy to improve the tracking performance without any model information. A preliminary study on the similarity between ILC and RL is proposed in Zhang et al. (2019b), which shows that RL is suitable for developing model free ILC algorithms. Model free actor-critic ILC algorithm is proposed in Poot et al. (2020) and it has been successfully verified on a printer setup.

For RL based method, it needs to try various input/parameters to achieve the perfect tracking. However, in some real application where safety critical issues is important (e.g., advanced robotic surgical system), the arbitrary choice of system parameters may cause serious consequence (since the possible output is unpredictable). To deal with this safety issue, there has been recent research on safety guaranteed model free learning, and this will be further investigated.

Data-driven ILC methods can utilize previous trials' input and output data to establish the input-output relationship (which does not reproduce the system model), and it is a powerful method when the model information is difficult to obtain. Some results that successfully applying data-driven ILC to high performance control problem are presented in Bolder and Oomen (2015); Bolder et al. (2014, 2018); Janssens et al. (2013), which could be a starting point of the investigation of data-driven ILC for high performance networked dynamical systems.

However, for existing data-driven approach, it will require extra more trials to perform the experiment in order to establish the input-output relationship, which increases the computational complexity. In networked dynamical systems, if the network is heterogeneous, it may produce great computational complexity when establishing each subsystem's input-output relationship. Furthermore, for the formation control problem without any reference trajectory, it is difficult to establish the input-output relationship (since each subsystem has infinite input choices and the tracking error is 'virtual'). These topics worth for future research and will be investigated.

Bibliography

H.-S. Ahn, Y. Chen, and K. L. Moore, "Iterative learning control: Brief survey and categorization," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1099–1121, 2007.

H.-S. Ahn, K. L. Moore, and Y. Chen, "Trajectory-keeping in satellite formation flying via robust periodic learning control," *International Journal of Robust and Nonlinear Control*, vol. 20, no. 14, pp. 1655–1666, 2010.

N. Amann, D. H. Owens, and E. Rogers, "Iterative learning control for discrete-time systems with exponential rate of convergence," *IEE Proceedings-Control Theory and Applications*, vol. 143, no. 2, pp. 217–224, 1996.

N. Amann, D. H. Owens, and E. Rogers, "Iterative learning control using optimal feedback and feedforward actions," *International Journal of Control*, vol. 65, no. 2, pp. 277–293, 1996.

N. Amann, D. H. Owens, E. Rogers, and A. Wahl, "An H_∞ approach to linear iterative learning control design," *International Journal of Adaptive Control and Signal Processing*, vol. 10, no. 6, pp. 767–781, 1996.

N. Amann, D. H. Owens, and E. Rogers, "Predictive optimal iterative learning control," *International Journal of Control*, vol. 69, no. 2, pp. 203–226, 1998.

B. D. Anderson, C. Yu, B. Fidan, and J. M. Hendrickx, "Rigid graph control architectures for autonomous formations," *IEEE Control Systems Magazine*, vol. 28, no. 6, pp. 48–63, 2008.

S. Arimoto, "Mathematical theory of learning with applications to robot control," in *Adaptive and Learning Systems.* Springer, 1986, pp. 379–388.

S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Field Robotics*, vol. 1, no. 2, pp. 123–140, 1984.

S. Arimoto, S. Kawamura, F. Miyazaki, and S. Tamaki, "Learning control theory for dynamical systems," in *1985 24th IEEE Conference on Decision and Control (CDC).* IEEE, 1985, pp. 1375–1380.

K. L. Barton and A. G. Alleyne, “A norm optimal approach to time-varying ILC with application to a multi-axis robotic testbed,” *IEEE Transactions on Control Systems Technology*, vol. 19, no. 1, pp. 166–180, 2011.

J. A. Benediktsson and P. H. Swain, “Consensus theoretic classification methods,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 4, pp. 688–704, 1992.

D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: Numerical methods*. New York: Athena Scientific, 1997.

H. Bi, M. Yang, and J. Chen, “Feedback-aided PID-type iterative learning control against initial state error,” in *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*. IEEE, 2018, pp. 899–902.

J. Bolder and T. Oomen, “Rational basis functions in iterative learning control—With experimental verification on a motion system,” *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 722–729, 2015.

J. Bolder and T. Oomen, “Inferential iterative learning control: A 2D-system approach,” *Automatica*, vol. 71, pp. 247–253, 2016.

J. Bolder, T. Oomen, S. Koekebakker, and M. Steinbuch, “Using iterative learning control with basis functions to compensate medium deformation in a wide-format inkjet printer,” *Mechatronics*, vol. 24, no. 8, pp. 944–953, 2014.

J. Bolder, S. Kleinendorst, and T. Oomen, “Data-driven multivariable ilc: enhanced performance by eliminating l and q filters,” *International Journal of Robust and Nonlinear Control*, vol. 28, no. 12, pp. 3728–3751, 2018.

S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

D. A. Bristow, M. Tharayil, and A. G. Alleyne, “A survey of iterative learning control,” *IEEE Control Systems*, vol. 26, no. 3, pp. 96–114, 2006.

X. Bu, J. Liang, Z. Hou, and R. Chi, “Data-driven terminal iterative learning consensus for nonlinear multiagent systems with output saturation,” *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

F. Bullo, *Lectures on Network Systems*, 1st ed. CreateSpace, 2018.

Y. Cao, W. Yu, W. Ren, and G. Chen, “An overview of recent progress in the study of distributed multi-agent coordination,” *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2013.

Y. C. Chang and W. J. Kim, “Aquatic ionic-polymer-metal-composite insectile robot with multi-DOF legs,” *IEEE/ASME Transactions On Mechatronics*, vol. 18, no. 2, pp. 547–555, 2013.

B. Chen and B. Chu, “Distributed norm optimal iterative learning control for point-to-point consensus tracking,” *IFAC-PapersOnLine*, vol. 52, no. 29, pp. 292–297, 2019.

B. Chen and B. Chu, “Distributed norm optimal iterative learning control for networked dynamical systems,” in *2019 American Control Conference (ACC).* IEEE, 2019, pp. 2879–2884.

B. Chen and B. Chu, “Distributed norm optimal iterative learning control for formation of networked dynamical systems,” in *2019 IEEE 58th Conference on Decision and Control (CDC).* IEEE, 2019, pp. 5574–5579.

B. Chen and B. Chu, “Distributed iterative learning control for networked dynamical systems with guaranteed individual energy cost,” in *2021 60th IEEE Conference on Decision and Control (CDC).* IEEE, 2021, pp. 6554–6559.

B. Chen and B. Chu, “Distributed iterative learning control for high performance consensus tracking problem with switching topologies,” in *2022 UKACC 13th International Conference on Control (CONTROL), to appear 2022*, 2022.

B. Chen, B. Chu, and H. Geng, “Distributed iterative learning control for constrained consensus tracking problem,” in *2020 59th IEEE Conference on Decision and Control (CDC).* IEEE, 2020, pp. 3745–3750.

S. Chen and C. T. Freeman, “A decentralised iterative learning control framework for collaborative tracking,” *Mechatronics*, vol. 72, p. 102465, 2020.

S. Chen, C. Freeman, and B. Chu, “Gradient-based iterative learning control for decentralised collaborative tracking,” in *2018 European Control Conference (ECC).* IEEE, 2018, pp. 721–726.

Y. Chen, J. Lü, F. Han, and X. Yu, “On the cluster consensus of discrete-time multi-agent systems,” *Systems & Control Letters*, vol. 60, no. 7, pp. 517–523, 2011.

Y. Chen, B. Chu, and C. T. Freeman, “Point-to-point iterative learning control with optimal tracking time allocation,” *IEEE Transactions on Control Systems Technology*, 2018.

Z. Cheng, Z. He, S. Zhang, and J. Li, “Constant modulus waveform design for MIMO radar transmit beampattern,” *IEEE Transactions on Signal Processing*, vol. 65, no. 18, pp. 4912–4923, 2017.

B. Chu and D. H. Owens, “Accelerated predictive norm-optimal iterative learning control,” *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 744–759, 2011.

B. Chu, "Iterative learning control for collaborative tracking: Point to point tasks and constraint handling," *IFAC-PapersOnLine*, vol. 52, no. 29, pp. 326–331, 2019.

B. Chu and D. H. Owens, "Accelerated norm-optimal iterative learning control algorithms using successive projection," *International Journal of Control*, vol. 82, no. 8, pp. 1469–1484, 2009.

B. Chu and D. H. Owens, "Iterative learning control for constrained linear systems," *International Journal of Control*, vol. 83, no. 7, pp. 1397–1413, 2010.

M. Defoort, A. Polyakov, G. Demesure, M. Djemai, and K. Veluvolu, "Leader-follower fixed-time consensus for multi-agent systems with unknown non-linear inherent dynamics," *IET Control Theory & Applications*, vol. 9, no. 14, pp. 2165–2170, 2015.

M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," *Journal of Scientific Computing*, vol. 66, no. 3, pp. 889–916, 2016.

S. Devasia, "Iterative learning control with time-partitioned update for collaborative output tracking," *Automatica*, vol. 69, pp. 258–264, 2016.

S. Devasia, "Iterative control for networked heterogeneous multi-agent systems with uncertainties," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 431–437, 2017.

L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii academiae scientiarum Petropolitanae*, pp. 128–140, 1741.

B. Farah, A. N. Bensaghir, and T. Iliass, "Minimization of delivery time in e-commerce and the impact on the global value chain," *Arabian J Bus Manag Review*, vol. 9, no. 376, p. 2, 2019.

C. T. Freeman and Y. Tan, "Iterative learning control with mixed constraints for point-to-point tracking," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 604–616, 2013.

C. T. Freeman, P. Lewin, and E. Rogers, "Experimental evaluation of iterative learning control algorithms for non-minimum phase plants," *International Journal of Control*, vol. 78, no. 11, pp. 826–846, 2005.

D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17–40, 1976.

X. Ge, J. L. Stein, and T. Ersal, “Frequency-domain analysis of robust monotonic convergence of norm-optimal iterative learning control,” *IEEE Transactions on Control Systems Technology*, vol. 26, no. 2, pp. 637–651, 2018.

E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, “Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems,” *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 644–658, 2015.

Y. Han, W. Lu, and T. Chen, “Cluster consensus in discrete-time networks of multiagents with inter-cluster nonidentical inputs,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 4, pp. 566–578, 2013.

T. J. Harte, J. Hatonen, and D. H. Owens, “Discrete-time inverse model-based iterative learning control: Stability, monotonicity and robustness,” *International Journal of Control*, vol. 78, no. 8, pp. 577–586, 2005.

J. Hatonen, D. H. Owens, and K. L. Moore, “An algebraic approach to iterative learning control,” *International Journal of Control*, vol. 77, no. 1, pp. 45–54, 2004.

L. Hladowski, K. Galkowski, Z. Cai, E. Rogers, C. T. Freeman, and P. L. Lewin, “Experimentally supported 2D systems based iterative learning control law design for error convergence and performance,” *Control Engineering Practice*, vol. 18, no. 4, pp. 339–348, 2010.

L. Hladowski, K. Galkowski, Z. Cai, E. Rogers, C. T. Freeman, and P. L. Lewin, “A 2D systems approach to iterative learning control for discrete linear processes with zero Markov parameters,” *International Journal of Control*, vol. 84, no. 7, pp. 1246–1262, 2011.

M. Hong and Z.-Q. Luo, “On the linear convergence of the alternating direction method of multipliers,” *Mathematical Programming*, vol. 162, no. 1-2, pp. 165–199, 2017.

Y. Hui, R. Chi, B. Huang, and Z. Hou, “3-D learning-enhanced adaptive ILC for iteration-varying formation tasks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 1, pp. 89–99, 2020.

P. Janssens, G. Pipeleers, and J. Swevers, “A data-driven constrained norm-optimal iterative learning control framework for LTI systems,” *IEEE Transactions on Control Systems Technology*, vol. 21, no. 2, pp. 546–551, 2013.

J. Jiang and Y. Jiang, “Leader-following consensus of linear time-varying multi-agent systems under fixed and switching topologies,” *Automatica*, vol. 113, p. 108804, 2020.

X. Jin, “Adaptive iterative learning control for high-order nonlinear multi-agent systems consensus tracking,” *Systems & Control Letters*, vol. 89, pp. 16–23, 2016.

X. Jin, “Iterative learning control for output-constrained nonlinear systems with input quantization and actuator faults,” *International Journal of Robust and Nonlinear Control*, vol. 28, no. 2, pp. 729–741, 2018.

S. Joshi, M. Codreanu, and M. Latva-aho, “Distributed SINR balancing for MISO downlink systems via the alternating direction method of multipliers,” in *2013 11th International Symposium and Workshops on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*. IEEE, 2013, pp. 318–325.

S. Knorn, Z. Chen, and R. H. Middleton, “Overview: Collective control of multiagent systems,” *IEEE Transactions on Control of Network Systems*, vol. 3, no. 4, pp. 334–347, 2016.

Y.-H. Lan and Z.-M. Cui, “ILC with initial state learning for fractional order linear distributed parameter systems,” *Algorithms*, vol. 11, no. 6, p. 85, 2018.

J. H. Lee, K. S. Lee, and W. C. Kim, “Model-based iterative learning control with a quadratic criterion for time-varying linear systems,” *Automatica*, vol. 36, no. 5, pp. 641–657, 2000.

J. Li and J. Li, “Adaptive iterative learning control for consensus of multi-agent systems,” *IET Control Theory & Applications*, vol. 7, no. 1, pp. 136–142, 2013.

J. Li and J. Li, “Adaptive iterative learning control for coordination of second-order multi-agent systems,” *International Journal of Robust and Nonlinear Control*, vol. 24, no. 18, pp. 3282–3299, 2014.

J. Li and J. Li, “Coordination control of multi-agent systems with second-order nonlinear dynamics using fully distributed adaptive iterative learning,” *Journal of the Franklin Institute*, vol. 352, no. 6, pp. 2441–2463, 2015.

J. Li and J. Li, “Distributed adaptive fuzzy iterative learning control of coordination problems for higher order multi-agent systems,” *International Journal of Systems Science*, vol. 47, no. 10, pp. 2318–2329, 2016.

J. Li, S. Liu, and J. Li, “Adaptive iterative learning protocol design for nonlinear multi-agent systems with unknown control direction,” *Journal of the Franklin Institute*, vol. 355, no. 10, pp. 4298–4314, 2018.

J. Li and J. Li, “Adaptive fuzzy iterative learning control with initial-state learning for coordination control of leader-following multi-agent systems,” *Fuzzy sets and Systems*, vol. 248, pp. 122–137, 2014.

X. Li, J.-X. Xu, and D. Huang, “Iterative learning control for nonlinear dynamic systems with randomly varying trial lengths,” *International Journal of Adaptive Control and Signal Processing*, vol. 29, no. 11, pp. 1341–1353, 2015.

T. Lin and D. H. Owens, "Monotonic Newton method based ILC with parameter optimization for non-linear systems," *International Journal of Control*, vol. 80, no. 8, pp. 1291–1298, 2007.

T. Lin, D. Owens, and J. Hatonen, "Newton method based iterative learning control for discrete non-linear systems," *International Journal of Control*, vol. 79, no. 10, pp. 1263–1276, 2006.

C. Liu, D. Shen, and J. Wang, "Iterative learning control of multi-agent systems with random noises and measurement range limitations," *International Journal of Systems Science*, vol. 50, no. 7, pp. 1465–1482, 2019.

X. Liu and T. Chen, "Cluster synchronization in directed networks via intermittent pinning control," *IEEE Transactions on Neural Networks*, vol. 22, no. 7, pp. 1009–1020, 2011.

Y. Liu and Y. Jia, "An iterative learning approach to formation control of multi-agent systems," *Systems & Control Letters*, vol. 61, no. 1, pp. 148–154, 2012.

Y. Liu and Y. Jia, "Robust formation control of discrete-time multi-agent systems by iterative learning approach," *International Journal of Systems Science*, vol. 46, no. 4, pp. 625–633, 2015.

D. G. Luenberger, *Optimization by vector space methods.* John Wiley & Sons, 1997.

S. Mandra, K. Galkowski, A. Rauh, H. Aschemann, and E. Rogers, "Iterative learning control for a class of multivariable distributed systems with experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 949–960, 2021.

D. Meng and Y. Jia, "Finite-time consensus for multi-agent systems via terminal feedback iterative learning," *IET control theory & applications*, vol. 5, no. 18, pp. 2098–2110, 2011.

D. Meng and Y. Jia, "Iterative learning approaches to design finite-time consensus protocols for multi-agent systems," *Systems & Control Letters*, vol. 61, no. 1, pp. 187–194, 2012.

D. Meng and Y. Jia, "Formation control for multi-agent systems through an iterative learning design approach," *International Journal of Robust and Nonlinear Control*, vol. 24, no. 2, pp. 340–361, 2014.

D. Meng and K. L. Moore, "Learning to cooperate: Networks of formation agents with switching topologies," *Automatica*, vol. 64, pp. 278–293, 2016.

D. Meng and K. L. Moore, "Robust cooperative learning control for directed networks with nonlinear dynamics," *Automatica*, vol. 75, pp. 172–181, 2017.

D. Meng, Y. Jia, J. Du, and F. Yu, "Tracking control over a finite interval for multi-agent systems with a time-varying reference trajectory," *Systems & Control Letters*, vol. 61, no. 7, pp. 807–818, 2012.

D. Meng, Y. Jia, and J. Du, "Multi-agent iterative learning control with communication topologies dynamically changing in two directions," *IET Control Theory & Applications*, vol. 7, no. 2, pp. 261–270, 2013.

D. Meng, Y. Jia, and J. Du, "Finite-time consensus protocols for networks of dynamic agents by terminal iterative learning," *International Journal of Systems Science*, vol. 45, no. 11, pp. 2435–2446, 2014.

D. Meng, Y. Jia, J. Du, and J. Zhang, "On iterative learning algorithms for the formation control of nonlinear multi-agent systems," *Automatica*, vol. 50, no. 1, pp. 291–295, 2014.

D. Meng, Y. Jia, and J. Du, "Robust consensus tracking control for multiagent systems with initial state shifts, disturbances, and switching topologies." *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 4, pp. 809–824, 2015.

D. Meng, Y. Jia, and J. Du, "Robust iterative learning protocols for finite-time consensus of multi-agent systems with interval uncertain topologies," *International Journal of Systems Science*, vol. 46, no. 5, pp. 857–871, 2015.

D. Meng, Y. Jia, J. Du, and J. Zhang, "High-precision formation control of nonlinear multi-agent systems with switching topologies: A learning approach," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 13, pp. 1993–2018, 2015.

D. Meng, Y. Jia, and J. Du, "Consensus seeking via iterative learning for multi-agent systems with switching topologies and communication time-delays," *International Journal of Robust and Nonlinear Control*, vol. 26, no. 17, pp. 3772–3790, 2016.

M. Minski, "The society of mind," *Columbia: Simon & Schuster*, 1986.

D. Mirza, P. Naughton, C. Schurgers, and R. Kastner, "Real-time collaborative tracking for underwater networked systems," *Ad Hoc Networks*, vol. 34, pp. 196–210, 2015.

A. A. Najm and I. K. Ibraheem, "Nonlinear PID controller design for a 6-DOF UAV quadrotor system," *Engineering Science and Technology, an International Journal*, vol. 22, no. 4, pp. 1087–1097, 2019.

J. Ni, L. Liu, C. Liu, and J. Liu, "Fixed-time leader-following consensus for second-order multiagent systems with input delay," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 11, pp. 8635–8646, 2017.

L. Noueili, W. Chagra, and M. L. Ksouri, "Optimal iterative learning control for a class of non-minimum phase systems," *International Journal of Modelling, Identification and Control*, vol. 28, no. 3, pp. 284–294, 2017.

K.-K. Oh, M.-C. Park, and H.-S. Ahn, “A survey of multi-agent formation control,” *Automatica*, vol. 53, pp. 424–440, 2015.

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.

R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

D. H. Owens, *Iterative learning control: An optimization paradigm.* Springer, 2016.

D. H. Owens and J. Hatonen, “Iterative learning control-An optimization paradigm,” *Annual Reviews in Control*, vol. 29, no. 1, pp. 57–70, 2005.

D. H. Owens and R. Jones, “Iterative solution of constrained differential/algebraic systems,” *International Journal of Control*, vol. 27, no. 6, pp. 957–974, 1978.

D. H. Owens, J. Hatonen, and S. Daley, “Robust monotone gradient-based discrete-time iterative learning control,” *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 19, no. 6, pp. 634–661, 2009.

D. H. Owens, C. T. Freeman, and T. Van Dinh, “Norm-optimal iterative learning control with intermediate point weighting: theory, algorithms, and experimental evaluation,” *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 999–1007, 2013.

D. H. Owens, B. Chu, E. Rogers, C. T. Freeman, and P. L. Lewin, “Influence of nonminimum phase zeros on the performance of optimal continuous-time iterative learning control,” *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1151–1158, 2014.

P. Pakshin, J. Emelianova, E. Rogers, and K. Galkowski, “Iterative Learning Control with Input Saturation,” *IFAC-PapersOnLine*, vol. 52, no. 29, pp. 338–343, 2019.

K.-H. Park, Z. Bien, and D.-H. Hwang, “Design of an iterative learning controller for a class of linear dynamic systems with time delay,” *IEE Proceedings-Control Theory and Applications*, vol. 145, no. 6, pp. 507–512, 1998.

W. Paszke, K. Gałkowski, E. Rogers, and D. H. Owens, “H_∞ and guaranteed cost control of discrete linear repetitive processes,” *Linear Algebra and its Applications*, vol. 412, no. 2-3, pp. 93–131, 2006.

M. Poot, J. Portegies, and T. Oomen, “On the role of models in learning control: Actor-critic iterative learning control,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1450–1455, 2020.

Z. Qu and J. Xu, “Asymptotic learning control for a class of cascaded nonlinear uncertain systems,” *IEEE Transactions on Automatic Control*, vol. 47, no. 8, pp. 1369–1376, 2002.

J. Realmuto, R. B. Warrier, and S. Devasia, “Data-inferred personalized human-robot models for iterative collaborative output tracking,” *Journal of Intelligent & Robotic Systems*, pp. 1–17, 2018.

N. Schurr, J. Marecki, M. Tambe, P. Scerri, N. Kasinadhuni, and J. P. Lewis, “The future of disaster response: Humans working with multiagent teams using defacto.” in *AAAI spring symposium: AI technologies for homeland security*, 2005, pp. 9–16.

D. Shen and J.-X. Xu, “Distributed adaptive iterative learning control for nonlinear multiagent systems with state constraints,” *International Journal of Adaptive Control and Signal Processing*, vol. 31, no. 12, pp. 1779–1807, 2017.

D. Shen and J.-X. Xu, “Distributed learning consensus for heterogenous high-order nonlinear multi-agent systems with output constraints,” *Automatica*, vol. 97, pp. 64–72, 2018.

D. Shen, J. Han, and Y. Wang, “Stochastic point-to-point iterative learning tracking without prior information on system matrices,” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 376–382, 2017.

D. Shen, C. Zhang, and J.-X. Xu, “Distributed learning consensus control based on neural networks for heterogeneous nonlinear multiagent systems,” *International Journal of Robust and Nonlinear Control*, vol. 29, no. 13, pp. 4328–4347, 2019.

D. Shen, Y. Liu, and Q. Song, “Collaborative tracking systems using decentralized iterative learning control,” in *2020 39th Chinese Control Conference (CCC)*. IEEE, 2020, pp. 1957–1962.

A. Simonetto and G. Leus, “Distributed maximum likelihood sensor network localization,” *IEEE Transactions on Signal Processing*, vol. 62, no. 6, pp. 1424–1437, 2014.

T. D. Son, G. Pipeleers, and J. Swevers, “Robust monotonic convergent iterative learning control,” *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1063–1068, 2016.

A. Tayebi, “Adaptive iterative learning control for robot manipulators,” *Automatica*, vol. 40, no. 7, pp. 1195–1203, 2004.

A. Teixeira, E. Ghadimi, I. Shames, H. Sandberg, and M. Johansson, “Optimal scaling of the ADMM algorithm for distributed quadratic programming,” in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 6868–6873.

A. Teixeira, E. Ghadimi, I. Shames, H. Sandberg, and M. Johansson, “The ADMM algorithm for distributed quadratic problems: Parameter selection and constraint preconditioning,” *IEEE Transactions on Signal Processing*, vol. 64, no. 2, pp. 290–305, 2016.

Y. Wang, W. Yin, and J. Zeng, “Global convergence of ADMM in nonconvex nonsmooth optimization,” *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, 2019.

R. B. Warrier and S. Devasia, “Iterative learning from novice human demonstrations for output tracking,” *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 4, pp. 510–521, 2016.

R. B. Warrier and S. Devasia, “Inferring intent for novice human-in-the-loop iterative learning control,” *IEEE Transactions on Control Systems Technology*, vol. 25, no. 5, pp. 1698–1710, 2017.

S. C. Weller and N. C. Mann, “Assessing rater performance without a “gold standard” using consensus theory,” *Medical Decision Making*, vol. 17, no. 1, pp. 71–79, 1997.

S. Wilcox and S. Devasia, “Stability of velocity control for a piezoelectric stepper,” *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 910–923, 2015.

P. R. Wurman, R. D’Andrea, and M. Mountz, “Coordinating hundreds of cooperative, autonomous vehicles in warehouses,” *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

J.-X. Xu, “Analysis of iterative learning control for a class of nonlinear discrete-time systems,” *Automatica*, vol. 33, no. 10, pp. 1905–1907, 1997.

J.-X. Xu and Z. Qu, “Robust iterative learning control for a class of nonlinear systems,” *Automatica*, vol. 34, no. 8, pp. 983–988, 1998.

J.-X. Xu, S. Zhang, and S. Yang, “A HOIM-based iterative learning control scheme for multi-agent formation,” in *2011 IEEE International Symposium on Intelligent Control*. IEEE, 2011, pp. 418–423.

N. Yang and J. Li, “Distributed iterative learning coordination control for leader–follower uncertain non-linear multi-agent systems with input saturation,” *IET Control Theory & Applications*, vol. 13, no. 14, pp. 2252–2260, 2019.

S. Yang and J.-X. Xu, “Leader–follower synchronisation for networked Lagrangian systems with uncertainties: A learning approach,” *International Journal of Systems Science*, vol. 47, no. 4, pp. 956–965, 2016.

S. Yang, J.-X. Xu, D. Huang, and Y. Tan, “Optimal iterative learning control design for multi-agent systems consensus tracking,” *Systems & Control Letters*, vol. 69, pp. 80–89, 2014.

S. Yang, J.-X. Xu, D. Huang, and Y. Tan, “Synchronization of heterogeneous multi-agent systems by adaptive iterative learning control,” *Asian Journal of Control*, vol. 17, no. 6, pp. 2091–2104, 2015.

S. Yang, J.-X. Xu, and X. Li, “Iterative learning control with input sharing for multi-agent consensus tracking,” *Systems & Control Letters*, vol. 94, pp. 97–106, 2016.

S. Yang, J.-X. Xu, X. Li, and D. Shen, *Iterative Learning Control for Multi-agent Systems Coordination*. John Wiley & Sons, 2017.

H. W. Yoo, S. Ito, and G. Schitter, "High speed laser scanning microscopy by iterative learning control of a galvanometer scanner," *Control Engineering Practice*, vol. 50, pp. 12–21, 2016.

C. Zhang, M. Ahmad, and Y. Wang, "ADMM based privacy-preserving decentralized optimization," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 565–580, 2019.

Y. Zhang, B. Chu, and Z. Shu, "A preliminary study on the relationship between iterative learning control and reinforcement learning," *IFAC-PapersOnLine*, vol. 52, no. 29, pp. 314–319, 2019.