University of Alberta

Library Release Form

Name of Author: Shuning Li

Title of Thesis: Computational Properties of Unconstrained Linear Distributed Model Predictive Control

Degree: Master of Science

Year this Degree Granted: 2013

. . . .

Shuning Li
CME 536
University of Alberta
Edmonton, AB
Canada, T6G 2G6

Date: . . . .

"To see a world in a grain of sand,
And a heaven in a wild flower,
Hold infinity in the palm of your hand,
And eternity in an hour."
–William Blake

"Science is organized knowledge. Wisdom is organized life."
–Immanuel Kant

University of Alberta

Computational Properties of Unconstrained Linear Distributed Model Predictive Control

by

Shuning Li

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

in

Process Control

Department of Department of Chemical and Materials Engineering

Edmonton, Alberta
Fall 2013

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Computational Properties of Unconstrained Linear Distributed Model Predictive Control** submitted by Shuning Li in partial fulfillment of the requirements for the degree of **Master of Science** in *Process Control*.

. . . .

J. Fraser Forbes

. . . .

Jinfeng Liu

. . . .

Zukui Li

Date: . . . .

To Those I Have Loved and Lost. R. I. P.

Acknowledgements

When my 2-year program is about to finish, I would like to express my gratitude to those who have helped and supported me during these unforgettable years in my graduate school.

First of all, I owe my thanks to my dearest supervisors, Dr. J. Fraser Forbes and Dr. Jinfeng Liu. They guided me through the difficulties in research and encouraged me, for things I did well as well as not so perfectly. Although Fraser is super busy, he is always there when I need his advices. I treasure the weekly meeting with Fraser, in which everyone is free-minded and willing to share something new, no matter it is a journal paper, a software or a tourism destination. Jinfeng is more like an elder brother rather than a supervisor, with whom I can discuss almost everything. He would ease me based on his own experience when I have concern about my research or when I am not confident enough. He is also a role model for me to be more efficient and diligent.

Dr. Padideh Ghafoor Mohseni and Bardia Hassanzadeh, who came to the distributed optimization group prior to me, helped me to get familar with this field and offered many valuable discussions, for which I am very grateful. I give my sincerest congratulations to Padideh, for passing the defence with such a high-quality thesis.

I would also like to thank my friends here in CPC group as well as in the department, who shared many happy moments with me: Jing Zhang, Yaojie Lu, Ming Ma, Kangkang Zhang, Lei Sun, Lei Chen, Da Zheng, Xiongtan Yang, Ruben Gonzalez, Liang Chen, Xi Huang, Yinan Wang, Keren Jiang, Sheng Tian, etc.. Without them my UA life would not be so colourful and the winter here would seem too long to endure.

Last but not least, my deepest love goes to my parents, my family and my boyfriend Tianbo. I do not know how to show my gratitude to them other than trying to make every day in my life happy and worthwhile.

Abstract

Typical chemical plants are large-scale systems composed of a number of processing units, which are integrated with each other via material, energy and information flows. To achieve optimal plant operation, various control strategies have been developed. Although centralized control provides the best performance, its fragile fault tolerance makes it impractical to implement. In industrial practice, such large-scale systems are operated by decentralized controllers, which do not have the implementation problems of centralized controllers; however, the decentralized controllers, in general, most often give suboptimal performance and may lead to loss of closed-loop stability. These concerns motivate the recent interest in the development of distributed control schemes, particularly distributed model predictive control (DMPC).

DMPC methods are applied to existing decentralized control networks and aim to bring their control performance closer to the centralized performance. Often iteratively, the local controllers in a distributed control network communicate with each other or a coordinator to adjust their actions. Since interactions between subsystems can be taken into consideration via information exchange, DMPCs are able to improve decentralized performance or even reproduce the centralized performance. Nevertheless, iterative communication also implies that DMPCs may have high computational and communication costs, which are seldom studied in the literature.

The focus of this thesis is the computational properties, mainly convergence and computational complexity, of two linear coordinated DMPCs: prediction-driven coordinated DMPC and price-driven coordinated DMPC. First, by restricting the study to linear unconstrained systems, the DMPC algorithms are transformed into iterative forms. Subsequently, explicit expressions for their convergence accuracy, convergence rates and the computational complexities are derived. A series of numerical experiments were also conducted to study the two DMPC methods' empirical computational complexity. It was discovered that DMPC methods' computational load is closely related to the local MPC design as well as factors like size and number of subsystems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Typical chemical plants are large-scale systems with a number of operation units, or subsystems, that are connected to each other via material, energy and information flows. The increasing requirement in process safety and environmental regulations, as well as the pursuit of profits and productivity have led to more complex and integrated plant operations, which challenges control engineers to create new methods to improve plant performance.

Over the past forty years, various advanced control strategies have been developed to improve plant operation. Based on their different control structures, these optimal control strategies can be broadly classified into three categories, namely: centralized, decentralized, and distributed control schemes. Among them, distributed control, in the context of model predictive control (MPC), has recently received intensive attention from both academia and industry, because it has the potential to achieve the centralized control performance while maintaining the flexibility of a decentralized control scheme.

Thanks to the inter-unit communication in a distributed control system, interactions between subsystems are taken into account, which can iteratively improve the distributed control performance to the optimal; however, this also implies that a distributed control scheme may have high communication and computation costs, which may reduce the applicability of the scheme. Therefore, it is desirable to investigate the interplay between the performance improvement and the increased communication and computational costs of distributed control schemes, which is seldom studied in the literature. A clear understanding of the communication and computational properties can help us to design practical plant-wide control systems. This thesis is aimed to investigate these properties in two different distributed control schemes.

1.2 Different Control Structures

This section will introduce each of the three different control structures, with their advantages and drawbacks. Moreover, since it is distributed model predictive control (DMPC) that is addressed in this thesis, the section also provides a short review of the existing DMPC literature.

Centralized Control Scheme

In the centralized control scheme, the entire plant is modelled as a whole and the centralized controller solves a monolithic control problem. An illustration of centralized control, where MPC is used to operate a plant with two subsystems, is presented in Figure 1.1. The model

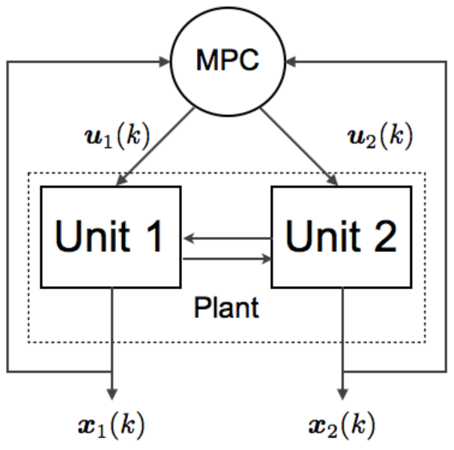


Figure 1.1: Illustration of centralized control scheme, when two subsystems are present in the plant.

built in the centralized control contains all of the interaction information between subsystems. Although centralized control gives the best possible control performance, there are difficulties when it comes to industrial implementation. According to Lu, 2003, centralized control lacks of flexibility and exhibits fragile fault tolerance to equipment failure. If for any reason the monolithic controller is shut down, the entire plant will be running without automatic control. It is also a difficult task to tune, maintain and improve the performance of local processing units under a centralized controller.

Decentralized Control Scheme

Traditionally, and in practice, decentralized control schemes are used to operate the large-scale, integrated systems, where the operation units are modelled and controlled separately.

Figure 1.2 shows an illustration of a decentralized control scheme, where both of the two operating units in the plant are operated by MPCs. Since each operating unit has its

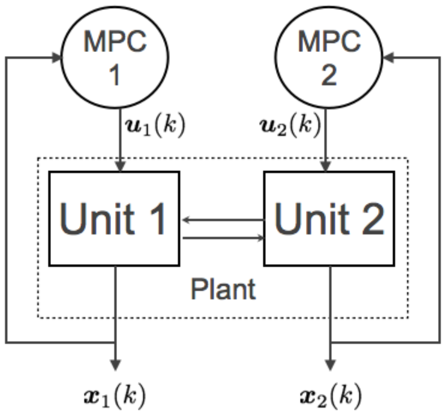
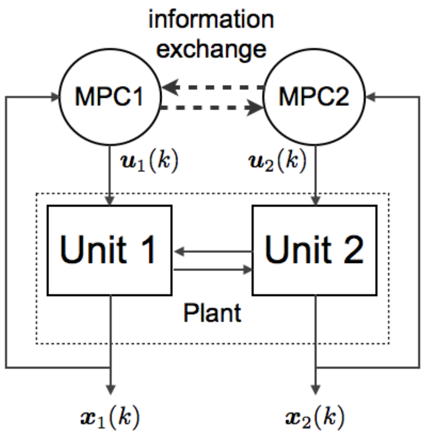


Figure 1.2: Illustration of decentralized control scheme, when two subsystems are present in the plant.

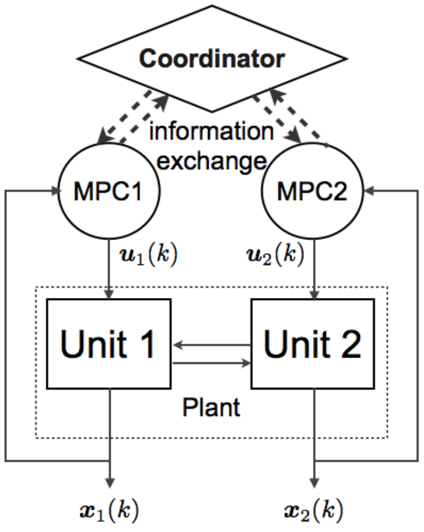
own controller, decentralized control is easy to implement and flexible in terms of system operation, controller tuning, equipment maintenance, etc.; however, the interaction between units are usually not included in the subsystems' models. For this reason, the decentralized control performance is often suboptimal. Only truly decoupled subsystems can achieve the centralized, optimal performance with a decentralized control network. In addition, when strong interactions exist between local processes, the entire plant is at risk of losing closed-loop stability (Lu, 2003; Sun and El-Farra, 2008; Rawlings and Stewart, 2008*b*; Marcos, 2011).

Distributed Control Scheme

The above concerns for decentralized and centralized control motivate recent interest on the distributed control, which is an alternative to the two approaches. Distributed control is applied on an existing decentralized control network, aiming to bring the plant performance closer to the centralized performance. This ensures that flexibility of the decentralized control is inherited by the distributed control system. Although existing distributed control schemes vary in control formulations and communication structures, they share the characteristic that information exchange exists between local controllers (Marcos, 2011). From the exchanged information, the local controllers are aware of other subsystems' intended control actions so that they can negotiate until a final 'agreement' is made. Appropriately designed communication content and structures will lead to improvement in distributed control performance, and the communication is often iterative. Among all the distributed



(a) The cooperative DMPC, when two subsystems are present in the plant. The local controllers operate their units, and also communicate with each other.



(b) The coordinated DMPC, when two subsystems are present in the plant. The local controllers operate their units, and also communicate with the coordinator.

Figure 1.3: Illustrations of cooperative DMPC and coordinated DMPC

control schemes proposed in recent years, the majority are distributed model predictive control (DMPC), which is also the focus of our research. For this reason, the following discussion on two classes of distributed control schemes will focus on DMPC.

There are two categories of DMPC schemes that will be emphasized in this work among different DMPC methods: cooperative DMPC and coordinated DMPC. In a cooperative DMPC, the local controllers communicate with each other iteratively until consensus is reached. A coordinated DMPC has a two-layer hierarchical structure, where an extra computing node called the 'coordinator' exists. The exchanged information flows between the coordinator and the local controllers to aid the local controllers in deciding their next move. The information exchange continues until the coordinator terminates the process. Figures 1.3a and 1.3b show the structures of the cooperative and coordinated DMPC, respectively.

Cooperative DMPC

Reviews of cooperative DMPC can be found in Camponogara *et al.*, 2002; Rawlings and Stewart, 2008*a*; Scattolini, 2009; Christofides *et al.*, 2013; Christofides *et al.*, 2011. The goal of this DMPC approach is to bring the decentralized control performance closer to the plant-wide optimal performance. To achieve this goal, a plant-wide control objective function is used in all the local controllers and a complete plant model is contained in each of the local controllers. Cooperative DMPC was first introduced by Venkat *et al.*, 2005 and then extended by Stewart *et al.*, 2010, where linear systems were considered. If a sufficient number of iterations are allowed at each sampling time, their methods will give the corresponding centralized solution. In Stewart *et al.*, 2011, the cooperative method was extended to nonlinear systems but without guaranteeing convergence to the optimal solution. The methods all have proved closed-loop stability for any intermediate termination.

The work in Liu *et al.*, 2009 and Liu *et al.*, 2010*b* proposes cooperative DMPC schemes for general nonlinear systems, wherein the local MPC controllers were developed via Lyapunov techniques. These cooperative methods feature in stable and feasible solutions, as well as the capability to handle asynchronous and delayed measurements (Liu *et al.*, 2010*a*; Liu *et al.*, 2012). Although general convergence to the optimal cannot be guaranteed, these DMPC approaches have shown convergence for linear systems (Christofides *et al.*, 2011).

Coordinated DMPC

Compared to cooperative DMPC, coordinated DMPC has a smaller literature. Recently, a comprehensive study of coordinated DMPC was made in Mohseni, 2013, which provided more insight into this area. Coordinated DMPC is designed based on the mathematical foundation of hierarchical multilevel system theories, which can be found in Mesarovic *et al.*, 1970. The purpose of the coordinated DMPC is to reproduce the optimal performance that would be achieved via centralized MPC. In the coordinated DMPC method, the coordinator uses intervention parameters (Mahmoud *et al.*, 1977), or equivalently coordinating variables (Mohseni, 2013) to influence the local controllers. It should be emphasized that the coordinator does not determine the local controller solutions; the role of the coordinator is more of advising the local controller about the next step.

The two-level hierarchical structure was applied in the area of control as early as the 1970s (Mahmoud, 1977 and Mahmoud *et al.*, 1977); however, not until recently in Cheng, 2007 and Cheng *et al.*, 2007 was the first attempt made to apply it on a distributed MPC network. In their work, the price-driven coordination method was applied to a decomposed, large-scale MPC. A price vector was used by the coordinator to adjust local controller behavior. Aske *et al.*, 2008 implemented the same coordinated MPC technique on the maximum throughput problem. Marcos, 2011 extended the price-driven coordination method for existing decentralized MPC networks, where the local processes are interconnected with each other. Inspired by Cohen, 1977, Marcos, 2011 also developed the prediction-driven coordination technique for Linear Quadratic (LQ) optimal control of large-scale, continuous-time, linear systems based on the Interaction Prediction Principle in Mesarovic *et al.*, 1970. Moreover, the prediction-driven coordinated method was applied to an unconstrained DMPC network in Marcos, 2011. Dual-rate coordinated DMPC problems were discussed, as well, in Marcos, 2011. Mohseni, 2013 reviewed three prevailing coordinated DMPCs: the Goal Coordination, the Interaction Prediction Coordination and the Modified Pseudo-Model Coordination methods. The aforementioned price-driven coordinated DMPC is a special case of the Goal Coordination method, while the prediction-driven method can be viewed as a variant of the Interaction Prediction Coordination method. Mohseni, 2013 also discussed the coordinated DMPC application where uncertainty and nonlinearity exist.

Other Non-coordinated DMPC

There are several other important existing DMPC studies. An analysis was made in Al-Gherwi *et al.*, 2011 of the DMPC to address the uncertainties in model, where a robust approach was developed. In Maestre *et al.*, 2011, the local systems were only coupled through inputs and the proposed DMPC technique was based on negotiations between agents. DMPCs of linear systems based on dissipativity conditions were proposed in Tippett and Bao, 2013. Scheu and Marquardt, 2011 discussed a sensitivity-based linear DMPC where convergence of the proposed algorithm was shown.

The majority of the above studies on DMPC focused on addressing closed-loop stability and performance. The computational properties of DMPC has received little attention.

1.3 Research Scope

This thesis is intended to provide some insight into the computational properties of DMPCs. Moreover, the research is restricted to linear and unconstrained Coordinated DMPC (CDMPC). With these restrictions, we are able to reach explicit iterative formulations so that the computational properties, mainly the convergence and complexity, can be analyzed theoretically.

For convenience of analysis, some assumptions are made regarding the large-scale system itself as well as modelling and control of the system. The assumptions are:

- the process has subsystems that are geographically distributed and interact with each other;

- the process has no noise and disturbance;

- the process and subsystems can be described using linear time-invariant discrete state-space systems with the same sampling time;
- the control instants of all units are the same as the sampling instants so that all control actions of the subsystems are taken at the same time;
- there is no plant-model mismatch in the modelling;
- all subsystems in the process have same prediction and control horizons;
- the control parameters do not change over time, the list of which include the sampling time, the setpoint, the prediction and control horizons of local MPCs;
- all the plant states are available for measurement and the outputs of the process are the same as the states;
- the objective of the plant-wide performance is the sum of all objectives of the local MPC controllers.

A CDMPC algorithm works iteratively during the interval between two sampling times, and it is the algorithm's convergence and complexity that is of our interest. In this thesis, the scope of time is the interval between two sampling times 'k' and '$k+1$'. Therefore, in the convergence and complexity analyses, the control step indicator '(k)' in the argument of the plant's predicted states $X(k)$, input changes $\Delta U(k)$, etc. is often omitted (especially in the two main chapters). An iteration for the coordinator and the local controllers to exchange information is called a **communication cycle**. It is also assumed that the time interval between k and $k+1$ is long enough so that sufficient communication cycles can take place and no premature termination occurs. Figure 1.4 illustrates the time scope of the thesis.

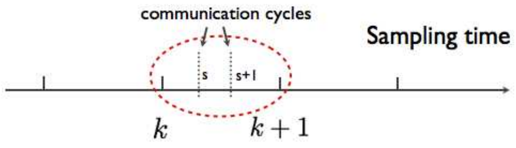


Figure 1.4: The scope of time of this research, which is marked by the dotted circle.

Since the performance of centralized MPC is the best possible, even though it is not physically practical, the centralized MPC is used as the benchmark of the DMPCs. Commonly in the literature, the comparison between DMPC performance and centralized MPC performance is for the control application as the sampling time k increases. 'Convergence to centralized MPC' in that context means that as the sampling time $k \to \infty$, the control action determined by DMPC is the same as that solved by the centralized MPC. In our research, 'convergence to centralized MPC' indicates that at the end of any k^{th} control interval, the DMPC algorithm will converge to the solution of centralized MPC at time k. Note that our definition of 'convergence' is stronger than the more common one, and if a DMPC is able to converge to the centralized MPC solution at the end of each sampling time, the convergence is guaranteed when the sampling time $k \to \infty$.

1.4 Thesis Outline and Contributions

Before the core content is presented, a preliminary chapter provides the terms, notation and conventions in the thesis. It also describes the plant model and the MPC formulation that will be used throughout the thesis. At the end of the chapter, the structure of CDMPC and the general formulation of the local MPCs are introduced.

There are two main chapters in the thesis, each discussing the computational properties of a certain type of CDMPC. Chapter 3 first presents the prediction-driven CDMPC as developed in Marcos, 2011, where the coordinator predicts all the process states and inputs within the plant. The algorithm is then written as an iterative function in the control action of the plant. Using theory for iterative methods, convergence condition and convergence rate of prediction-driven CDMPC algorithm are reached. The complexity of the CDMPC method is analyzed both theoretically and empirically to understand its computational load. Chapter 4 introduces the price-driven CDMPC, in which the price vector is used to coordinate local MPC controllers. Convergence conditions, convergence rate and computational complexity of the price-driven CDMPC are analyzed applying the same methodology as in Chapter 3.

The last chapter, Chapter 5, reviews the main results of the thesis and discusses the possible choices of future research work.

The contribution of the thesis can be summarized as:

- for unconstrained prediction-driven CDMPC, an explicit expression of convergence condition is reached, which is useful in judging whether or not a CDMPC design is applicable;

- unconstrained price-driven CDMPC is proved to converge to the centralized MPC for any system;

- the convergence rates of both CDMPCs are given, and, to the best of our knowledge, it is the first time that a convergence rate is derived for a DMPC algorithm;

- for unconstrained prediction-driven CDMPC, a reliable estimation of the required number of communication cycles is given; for the unconstrained price-driven CDMPC, the number of communication cycles is proved to be two;

- it has been found that the complexity of the prediction-driven CDMPC is not only determined by the size of the plant and the size of the maximum subsystem, but also the number of communication cycles, which is closely related to the algorithm's rate of convergence. Since this number is a very complicated mapping from the available information to the set of positive real numbers, and may be very large as plant size increases, it is the key contributor for the prediction-driven CDMPC's complexity;

- it is shown that the complexity of the unconstrained price-driven CDMPC is polynomial in the size of the plant and the size of the maximum subsystem. When the number of subsystem is large enough, this CDMPC algorithm is more efficient than centralized MPC.

Chapter 2

Preliminaries

2.1 Notation

In this thesis, all scalars, vectors and matrices and functions are restricted to the real space $\mathbb{R}$, if not specified. The blackboard bold font, (e.g., $\mathbb{R}$) generally denotes sets. For example, if a matrix is said to be in $\mathbb{R}^{n\times m}$, it has a dimension of $n \times m$ where all its elements are real. The superscript '$n \times m$' denotes the Cartesian product of the domain. $\mathbb{R}^{+}$ is the set of positive real numbers. In addition to $\mathbb{R}$, the complex space $\mathbb{C}$ also occurs in the thesis.

A bold $\mathbf{0}$ denotes a column vector with all entries being zero, and if its dimension is known, the information will be put at the subscript. For example, $\mathbf{0}_p$ means that it is a $p \times 1$ zero vector.

A zero matrix is either left blank or denoted by a bold, mathematical $\boldsymbol{O}$. Its dimensional information will be shown in the subscript when needed. For example, in the block-diagonal matrix

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{M}_1 & \\ & \boldsymbol{M}_2 \end{bmatrix}, \tag{2.1}$$

the off-diagonal matrices are zero matrices and are left blank. A matrix written as $\boldsymbol{O}_{l\times m}$ is a $l \times m$ zero matrix, and $\boldsymbol{O}_l$ represents a square, $l \times l$ matrix.

The convention of distinguishing scalars, vectors and matrices by different fonts are followed in the thesis, though some exceptions do exist. Scalars are denoted using the default math font. For example, the control interval k, the number of plant states n and the prediction horizon H_p are all scalars. Vectors are usually denoted by bold math font with lower case alphabet, such as the states in a system $\boldsymbol{x}$, the price vector $\boldsymbol{p}^{(s)}$ and the Lagrange multiplier $\boldsymbol{\lambda}$. The exceptions here, are the predicted state vector X, predicted input change vector ΔU, the predicted interaction V and the interaction error vector E, which are denoted by upper case but unbold font. Matrices are represented by bold font with upper case alphabet. Examples are the system state matrix $\boldsymbol{A}$, MPC weighting matrices $\boldsymbol{Q}$

and $\boldsymbol{R}$.

If the eigenvalues of a square matrix $\boldsymbol{M} \in \mathbb{C}^{m\times m}$ are: $\lambda_1, ..., \lambda_m$, then the **spectral radius** of $\boldsymbol{M}$ is defined as:

$$\rho(\boldsymbol{M}) = \max_{i=1,...,m}(|\lambda_i|). \tag{2.2}$$

In this work, we reserve the greek alphabet ρ to denote spectral radius, whether or not the associated matrix is specified.

The '*diag*' operator is used to build a block-diagonal matrix. For example, $\boldsymbol{M} = diag(\boldsymbol{M}_1, \boldsymbol{M}_2)$ is equivalent to the matrix presented in (2.1).

If something needs to be defined, usually the phrase 'define as' will be used. Sometimes for coherency or simplicity, the '$\triangleq$' symbol will be used in equations.

2.2 Terms and Definitions

Some of the key terms used in this thesis are listed and explained in this section to avoid possible misunderstandings.

A **large-scale system**, or a **plant**, refers to the system that consists of multiple subsystems that interact with each other, where **subsystems** are the geographically distributed operating units with their own controllers as described in §1.1. When the word **local** is used, the subsystems are addressed. Examples are **local controllers**, **local process models**, etc.

In this thesis, a **control network** is used to describe the set of controllers that operates the entire plant. It could be: a **decentralized control network**, which is the existing control network; a **centralized control network**, where the plant is controlled by an ideal, imaginary controller; or a **distributed control network**, which features the communication between network nodes and is considered to be a modification of the decentralized network. A **node** in the control network is able to carry out computation and communicate with other nodes in the network. In the CDMPC network, all local controllers and the coordinator are nodes.

A **centralized control problem** is used to refer the problem that optimizes the plant-wide objective function subject to all local process models and all of the interactions between subsystems. The **centralized solution**, or interchangeably, the **optimal solution**, or the **optimal operation** is the solution of the centralized control problem.

For a CDMPC network, the term **communication cycle** represents the iterative process of the coordinator and local controllers exchanging information, where one **cycle** includes a loop for the information flow. The iteration for the nodes in a DMPC network to communicate will be referred to as 'cycle' to distinguish it from the more general iteration in a numerical method, such as the iteration in the interior-point method, which is discussed in §3.3.1 and §4.3.1.

Since techniques drawn from computational science are applied to study the computational properties of DMPC in this thesis, in the analysis a **CDMPC method** is often treated as an algorithm. Therefore, **CDMPC algorithm** may be used interchangeably with CDMPC method, which means a particular CDMPC design that is implementable on a decentralized control network.

Some abbreviations that will be used in the thesis are listed here: **MPC** is an acronym for model predictive control; **DMPC** for distributed model predictive control; **CDMPC** for coordinated, distributed model predictive control; and **RHS** and **LHS** stand for the right hand side and the left hand side of an equation, respectively.

2.3 Plant Model

In this work, the entire plant will be described by the following discrete-time state-space model:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \tag{2.3}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^n$ is the plant state vector and $\boldsymbol{u}(k) \in \mathbb{R}^q$ is the plant input vector. Note that all process variables are in deviation form with respect to the steady-state value and are given at a specific time, e.g., k. It is considered that within the plant there are N interconnected subsystems, each with n_i states and q_i inputs, $i = 1, ..., N$. This implies that:

$$\boldsymbol{x}(k)^T = \left[\boldsymbol{x}_1(k)^T, \boldsymbol{x}_2(k)^T ..., \boldsymbol{x}_N(k)^T\right]^T, \tag{2.4}$$

$$\boldsymbol{u}(k)^T = \left[\boldsymbol{u}_1(k)^T, \boldsymbol{u}_2(k)^T ..., \boldsymbol{u}_N(k)^T\right]^T, \tag{2.5}$$

where $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$ and $\boldsymbol{u}_i \in \mathbb{R}^{q_i}$ are the state and input vectors of the i^{th} subsystem, respectively. The plant's state matrix $\boldsymbol{A}$ and input matrix $\boldsymbol{B}$ can be partitioned in the following block-wise fashion:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \cdots & \boldsymbol{A}_{1N} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \cdots & \boldsymbol{A}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{N1} & \boldsymbol{A}_{N2} & \cdots & \boldsymbol{A}_{NN} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} & \cdots & \boldsymbol{B}_{1N} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} & \cdots & \boldsymbol{B}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{B}_{N1} & \boldsymbol{B}_{N2} & \cdots & \boldsymbol{B}_{NN} \end{bmatrix}, \tag{2.6}$$

where $\boldsymbol{A}_{ij} \in \mathbb{R}^{n_i \times n_j}$, $\boldsymbol{B}_{ij} \in \mathbb{R}^{n_i \times q_j}$, $i, j = 1, ..., N$. The i^{th} subsystem can be represented by the state space model:

$$\boldsymbol{x}_i(k+1) = \boldsymbol{A}_{ii}\boldsymbol{x}_i(k) + \boldsymbol{B}_{ii}\boldsymbol{u}_i(k) + \sum_{i \neq j} (\boldsymbol{A}_{ij}\boldsymbol{x}_j(k) + \boldsymbol{B}_{ij}\boldsymbol{u}_j(k)). \tag{2.7}$$

The first part, $\boldsymbol{A}_{ii}\boldsymbol{x}_i(k)+\boldsymbol{B}_{ii}\boldsymbol{u}_i(k)$ in equation (2.7) denotes the local dynamics of subsystem i. $\boldsymbol{A}_{ij}\boldsymbol{x}_j(k)+\boldsymbol{B}_{ij}\boldsymbol{u}_j(k), j \neq i$ represents the impact of interaction on the i^{th} subsystem from another subsystem j.

2.4 MPC Formulation

Model predictive control is a model-based optimal control technique, which optimizes the predicted system performance over a finite horizon (called the 'prediction horizon') and solves for the best control action over a time period (called the 'control horizon') in each control interval[1]. The first control action of the input trajectory solution is applied to the process while the others are discarded. At the next control interval, the prediction and control horizons are shifted forward by one sampling time and the optimization is repeated based on the updated process measurements. MPC is widely accepted in industry as it can deal with multivariable control problems and is capable of handling constraints, etc.. In this work, the MPCs will be formulated as quadratic programming problems as described in Maciejowski, 2002, where the objective penalizes the state deviation from a given trajectory and the change of control input.

As stated in §1.2, it is assumed that each of the subsystems in the plant has been operated by a decentralized MPC controller. The DMPC schemes applied to these local controllers will modify the original decentralized design. For example, the coordinated DMPC schemes adds a modification term to the decentralized local MPC objective, so that the subsystem can exchange information with and be coordinated by the coordinator. The performance of DMPC control schemes are to be compared with the centralized MPC performance. The formulation of the centralized MPC and the decentralized local MPCs will be introduced in the following sections.

Also, since the scope of this work is restricted to unconstrained systems, in the quadratic optimization problems there are only equality constraints representing the predictions using the system models (2.3) and (2.7).

[1]Since the scope of the thesis is discretized systems, the MPC introduced here is discrete-time MPC. Those who are interested in continuous-time MPC can refer to Wang, 2009

Centralized MPC Formulation

The centralized MPC problem can be formulated as the following QP:

$$\begin{aligned}\min_{\tilde{X},\Delta\tilde{U}} \mathcal{J} = \frac{1}{2}\Bigg(&\sum_{l=1}^{H_p}(\hat{\boldsymbol{x}}(k+l|k)-\boldsymbol{r}(l))^T\boldsymbol{Q}(l)(\hat{\boldsymbol{x}}(k+l|k)-\boldsymbol{r}(l))+ \\ &\sum_{p=0}^{H_u-1}(\Delta\hat{\boldsymbol{u}}(k+p|k)^T\boldsymbol{R}(p)\Delta\hat{\boldsymbol{u}}(k+p|k))\Bigg)\end{aligned} \tag{2.8a}$$

subject to:

$$\hat{\boldsymbol{x}}(k+l+1|k) = \boldsymbol{A}\hat{\boldsymbol{x}}(k+l|k)+\boldsymbol{B}\left(\sum_{a=0}^{l}\Delta\hat{\boldsymbol{u}}(k+l|k)+\boldsymbol{u}(k-1)\right), \tag{2.8b}$$

$$\begin{cases} l=0,1,...,H_p-1, \\ \Delta\hat{\boldsymbol{u}}(k+l|k)=\hat{\boldsymbol{u}}(k+l|k)-\hat{\boldsymbol{u}}(k+l-1|k), \\ \hat{\boldsymbol{u}}(k-1|k)=\boldsymbol{u}(k-1), \\ \Delta\hat{\boldsymbol{u}}(k+b|k)=\boldsymbol{0}, H_u\le b\le H_p-1 \\ \hat{\boldsymbol{x}}(k|k)=\boldsymbol{x}(k), \end{cases} \tag{2.8c}$$

where: H_p and H_u are the prediction and control horizon, respectively: $\tilde{X} = [\hat{\boldsymbol{x}}(k+1|k)^T,...,\hat{\boldsymbol{x}}(k+H_p|k)^T]^T$ and $\Delta\tilde{U} = [\Delta\hat{\boldsymbol{u}}(k|k)^T,...,\Delta\hat{\boldsymbol{u}}(k+H_u-1|k)^T]^T$ are the predicted states and inputs, respectively; $\boldsymbol{r}(l)$ is the given reference trajectory at time $k+l$ for the predicted states $\hat{\boldsymbol{x}}(k+l|k)$, $l=1,...,H_p$; $\boldsymbol{Q}(l)$ is a positive definite matrix penalizing the deviation of $\hat{\boldsymbol{x}}(k+l|k)$ from $\boldsymbol{r}(l)$; and $\boldsymbol{R}(p)$ is the weighting matrix penalizing the predicted input changes $\Delta\hat{\boldsymbol{u}}(k+p|k), p=0,...,H_u-1$, and is positive definite.

Since $\hat{\boldsymbol{x}}(k+l|k), l=1,...,H_p$ and $\Delta\hat{\boldsymbol{u}}(k+p|k), p=0,...,H_u-1$ can be written as:

$$\hat{\boldsymbol{x}}(k+l|k)^T = \left[\hat{\boldsymbol{x}}_1(k+l|k)^T, \hat{\boldsymbol{x}}_2(k+l|k)^T..., \hat{\boldsymbol{x}}_N(k+l|k)^T\right]^T, \tag{2.9}$$

$$\Delta\hat{\boldsymbol{u}}(k+p|k)^T = \left[\Delta\hat{\boldsymbol{u}}_1(k+p|k)^T, \Delta\hat{\boldsymbol{u}}_2(k+p|k)^T..., \Delta\hat{\boldsymbol{u}}_N(k+p|k)^T\right]^T, \tag{2.10}$$

then $\boldsymbol{r}(l) = [\boldsymbol{r}_1(l)^T, \boldsymbol{r}_2(l)^T,...,\boldsymbol{r}_N(l)^T]$, $\boldsymbol{Q}(l) = diag(\boldsymbol{Q}_1(l),\boldsymbol{Q}_2(l),...,\boldsymbol{Q}_N(l)), l=1,...,H_p$ and $\boldsymbol{R}(p) = diag(\boldsymbol{R}_1(p),\boldsymbol{R}_2(p),...,\boldsymbol{R}_N(p)), p=0,...,H_u-1$, with $\boldsymbol{Q}_i(l)$ and $\boldsymbol{R}_i(p)$ all positive definite matrices. Therefore, problem (2.8) can also be written as:

$$\min_{X,\Delta U} \mathcal{J} = \frac{1}{2}\sum_{i=1}^{N}\left(\sum_{l=1}^{H_p}(\hat{\boldsymbol{x}}_i(k+l|k)-\boldsymbol{r}_i(l))^T\boldsymbol{Q}_i(l)(\hat{\boldsymbol{x}}_i(k+l|k)-\boldsymbol{r}_i(l))+ \sum_{p=0}^{H_u-1}(\Delta\hat{\boldsymbol{u}}_i(k+p|k)^T\boldsymbol{R}_i(p)\Delta\hat{\boldsymbol{u}}_i(k+p|k))\right) \tag{2.11a}$$

subject to:

$$\hat{\boldsymbol{x}}_i(k+l+1|k) = \left(\boldsymbol{A}_{ii}\hat{\boldsymbol{x}}_i(k+l|k)+\boldsymbol{B}_{ii}(\sum_{a=0}^{l}\Delta\hat{\boldsymbol{u}}_i(k+l|k)+\boldsymbol{u}_i(k-1))\right)+ \left(\sum_{j=1,j\neq i}^{j=N}(\alpha\boldsymbol{A}_{ij}\boldsymbol{x}_j(k)+\boldsymbol{B}_{ij}\boldsymbol{u}_j(k-1))\right)+ \sum_{j=1,j\neq i}^{j=N}\left((1-\alpha)\boldsymbol{A}_{ij}\hat{\boldsymbol{x}}_j(k+l|k)+\boldsymbol{B}_{ij}\sum_{a=0}^{l}\Delta\hat{\boldsymbol{u}}_j(k+a|k)\right), \tag{2.11b}$$

$$\begin{cases} l=0,1,...,H_p-1, \\ \Delta\hat{\boldsymbol{u}}_i(k+l|k)=\hat{\boldsymbol{u}}_i(k+l|k)-\hat{\boldsymbol{u}}_i(k+l-1|k), \\ \hat{\boldsymbol{u}}_i(k-1|k)=\boldsymbol{u}_i(k-1), \\ \Delta\hat{\boldsymbol{u}}_i(k+b|k)=\boldsymbol{0}, H_u\le b\le H_p-1 \\ \hat{\boldsymbol{x}}_i(k|k)=\boldsymbol{x}_i(k), \\ \alpha=\begin{cases}1, \text{if } l=0, \\ 0, \text{if } l=1,...,H_p-1,\end{cases} \\ i=1,...,N, \end{cases} \tag{2.11c}$$

where $X=[X_1^T,...,X_N^T]^T$, $\Delta U=[\Delta U_1^T,...,\Delta U_N^T]^T$, and:

$$X_i=[\hat{\boldsymbol{x}}_i(k+1|k)^T,\hat{\boldsymbol{x}}_i(k+2|k)^T,...,\hat{\boldsymbol{x}}_i(k+H_p|k)^T]^T, \tag{2.12}$$

$$\Delta U_i=[\Delta\hat{\boldsymbol{u}}_i(k|k)^T,\Delta\hat{\boldsymbol{u}}_i(k+1|k)^T,...,\Delta\hat{\boldsymbol{u}}_i(k+H_u-1|k)^T]^T. \tag{2.13}$$

Problem (2.11) can be written in a compact matrix form as follows:

$$\min_{X,\Delta U}\mathcal{J}=\frac{1}{2}\left((X-\boldsymbol{r})^T\boldsymbol{Q}(X-\boldsymbol{r})+\Delta U^T\boldsymbol{R}\Delta U\right) \tag{2.14a}$$

subject to:

$$\boldsymbol{G}\begin{bmatrix} X \\ \Delta U\end{bmatrix}=\boldsymbol{g}, \tag{2.14b}$$

where $\boldsymbol{r}=[\boldsymbol{r}_1^T,...,\boldsymbol{r}_N^T]^T$, $\boldsymbol{Q}=diag(\boldsymbol{Q}_1,\boldsymbol{Q}_2,...,\boldsymbol{Q}_N)$, $\boldsymbol{R}=diag(\boldsymbol{R}_1,\boldsymbol{R}_2,...,\boldsymbol{R}_N)$ and:

$$\boldsymbol{r}_i=[\boldsymbol{r}_i(1)^T,...,\boldsymbol{r}_i(H_p)^T]^T, i=1,...N, \tag{2.15}$$

$$\boldsymbol{Q}_i=diag(\boldsymbol{Q}_i(1),...,\boldsymbol{Q}_i(H_p)), i=1,...N, \tag{2.16}$$

$$\boldsymbol{R}_i=diag(\boldsymbol{R}_i(0),...\boldsymbol{R}_i(H_u-1)), i=1,...N. \tag{2.17}$$

Since all $\boldsymbol{Q}_i(l), l = 1, ..., H_p$ and $\boldsymbol{R}_i(p), p = 1, ..., H_u$ are positive definite matrices, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are positive definite as well. Consequently, they are both symmetric matrices, i.e., $\boldsymbol{Q} = \boldsymbol{Q}^T$ and $\boldsymbol{R} = \boldsymbol{R}^T$. $\boldsymbol{G}$ in the equality constraint (2.14b) is composed of two blocks, i.e., $\boldsymbol{G} = [\boldsymbol{G}_A, \boldsymbol{G}_B]$, where $\boldsymbol{G}_A$ and $\boldsymbol{G}_B$ have the following block-wise form:

$$\boldsymbol{G}_A = \begin{bmatrix} \boldsymbol{G}_{A_{11}} & \boldsymbol{G}_{A_{12}} & \cdots & \boldsymbol{G}_{A_{1N}} \\ \boldsymbol{G}_{A_{21}} & \boldsymbol{G}_{A_{22}} & \cdots & \boldsymbol{G}_{A_{2N}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{G}_{A_{1N}} & \boldsymbol{G}_{A_{2N}} & \cdots & \boldsymbol{G}_{A_{NN}} \end{bmatrix}, \boldsymbol{G}_B = \begin{bmatrix} \boldsymbol{G}_{B_{11}} & \boldsymbol{G}_{B_{12}} & \cdots & \boldsymbol{G}_{B_{1N}} \\ \boldsymbol{G}_{B_{21}} & \boldsymbol{G}_{B_{22}} & \cdots & \boldsymbol{G}_{B_{2N}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{G}_{B_{1N}} & \boldsymbol{G}_{B_{2N}} & \cdots & \boldsymbol{G}_{B_{NN}} \end{bmatrix}. \tag{2.18}$$

$\boldsymbol{G}_{A_{ii}}$ and $\boldsymbol{G}_{A_{ij}}, i \neq j$ can be written as:

$$\boldsymbol{G}_{A_{ii}} = \underbrace{\begin{bmatrix} \boldsymbol{I}_{n_i} & & & & \\ -\boldsymbol{A}_{ii} & \boldsymbol{I}_{n_i} & & & \\ & -\boldsymbol{A}_{ii} & \boldsymbol{I}_{n_i} & & \\ & & \ddots & \ddots & \\ & & & -\boldsymbol{A}_{ii} & \boldsymbol{I}_{n_i} \end{bmatrix}}_{H_p \times H_p \text{ blocks}}, \tag{2.19}$$

$$\boldsymbol{G}_{A_{ij}} = \underbrace{\begin{bmatrix} \boldsymbol{O}_{n_j \times n_i} & & & & \\ -\boldsymbol{A}_{ij} & \boldsymbol{O}_{n_j \times n_i} & & & \\ & -\boldsymbol{A}_{ij} & \boldsymbol{O}_{n_j \times n_i} & & \\ & & \ddots & \ddots & \\ & & & -\boldsymbol{A}_{ij} & \boldsymbol{O}_{n_j \times n_i} \end{bmatrix}}_{H_p \times H_p \text{ blocks}}, \tag{2.20}$$

while $\boldsymbol{G}_{B_{ij}}, i, j = 1, ..., N$ are uniformly defined as:

$$\boldsymbol{G}_{B_{ij}} = \underbrace{\begin{bmatrix} -\boldsymbol{B}_{ij} & & & \\ -\boldsymbol{B}_{ij} & -\boldsymbol{B}_{ij} & & \\ \vdots & \vdots & \ddots & \\ -\boldsymbol{B}_{ij} & -\boldsymbol{B}_{ij} & \cdots & -\boldsymbol{B}_{ij} \end{bmatrix}}_{H_p \times H_u \text{ blocks}}. \tag{2.21}$$

The dimensions of $\boldsymbol{G}_A$ and $\boldsymbol{G}_B$ are $H_p n \times H_p n$ and $H_p n \times H_u q$, respectively. $\boldsymbol{g}$ in (2.14b) is defined as:

$$\boldsymbol{g} = [\boldsymbol{g}_1^T, \boldsymbol{g}_2^T, ..., \boldsymbol{g}_N^T]^T, \tag{2.22}$$

$$\boldsymbol{g}_i = \sum_{i=1}^{N} \boldsymbol{g}_{ij}, \tag{2.23}$$

where $\boldsymbol{g}_{ij}, i, j = 1, ..., N$ can be built following (2.24):

$$\boldsymbol{g}_{ij} = \begin{bmatrix} \boldsymbol{A}_{ij}\boldsymbol{x}_j(k) + \boldsymbol{B}_{ij}\boldsymbol{u}_j(k-1) \\ \boldsymbol{B}_{ij}\boldsymbol{u}_i(k-1) \\ \vdots \\ \boldsymbol{B}_{ij}\boldsymbol{u}_i(k-1) \end{bmatrix} \quad \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} (H_p - 1) \text{ times.} \tag{2.24}$$

Existing Decentralized MPC Formulation

The existing i^{th} local MPC is formulated as the following optimization problem:

$$\min_{X_i,\Delta U_i} \mathcal{J}_i = \frac{1}{2}\left(\sum_{l=1}^{H_p}(\hat{\boldsymbol{x}}_i(k+l|k)-\boldsymbol{r}_i(l))^T\boldsymbol{Q}_i(l)(\hat{\boldsymbol{x}}_i(k+l|k)-\boldsymbol{r}_i(l))+\right.$$
$$\left.\sum_{p=0}^{H_u-1}(\Delta\hat{\boldsymbol{u}}_i(k+p|k)^T\boldsymbol{R}_i(p)\Delta\hat{\boldsymbol{u}}_i(k+p|k))\right) \tag{2.25a}$$

subject to:

$$\hat{\boldsymbol{x}}_i(k+l+1|k)=\boldsymbol{A}_{ii}\hat{\boldsymbol{x}}_i(k+l|k)+\boldsymbol{B}_{ii}\left(\sum_{a=0}^{l}\Delta\hat{\boldsymbol{u}}_i(k+l|k)+\boldsymbol{u}_i(k-1))\right), \tag{2.25b}$$

$$\begin{cases} l=0,1,...,H_p-1, \\ \Delta\hat{\boldsymbol{u}}_i(k+l|k)=\hat{\boldsymbol{u}}_i(k+l|k)-\hat{\boldsymbol{u}}_i(k+l-1|k), \\ \hat{\boldsymbol{u}}_i(k-1|k)=\boldsymbol{u}_i(k-1), \\ \Delta\hat{\boldsymbol{u}}_i(k+b|k)=\boldsymbol{0}, H_u\le b\le H_p-1, \\ \hat{\boldsymbol{x}}_i(k|k)=\boldsymbol{x}_i(k), \end{cases} \tag{2.25c}$$

where: X_i and ΔU_i are the predicted states and input for the i^{th} subsystem, and have been defined in (2.12) and (2.13), respectively; $\boldsymbol{r}_i(l)$ is the given reference trajectory for the predicted states $\hat{\boldsymbol{x}}_i(k+l|k)$, $l=1,...,H_p$; $\boldsymbol{Q}_i(l)$ penalizes the deviation of $\hat{\boldsymbol{x}}_i(k+l|k)$ from $\boldsymbol{r}_i(l)$; and $\boldsymbol{R}_i(p)$ penalizes the predicted input changes $\Delta\hat{\boldsymbol{u}}_i(k+p|k), p=0,...,H_u-1$.

As in the centralized MPC, problem (2.25) can be rearranged into a compact matrix form:

$$\min_{X_i,\Delta U_i} \mathcal{J}_i = \frac{1}{2}\left((X_i-\boldsymbol{r}_i)^T\boldsymbol{Q}_i(X_i-\boldsymbol{r}_i)+\Delta U_i^T\boldsymbol{R}_i\Delta U_i\right) \tag{2.26a}$$

subject to:

$$\boldsymbol{G}_{ii}\begin{bmatrix} X_i \\ \Delta U_i \end{bmatrix}=\boldsymbol{g}_{ii}, \tag{2.26b}$$

where $\boldsymbol{r}_i$, $\boldsymbol{Q}_i$ and $\boldsymbol{R}_i$ have been defined in (2.15), (2.16) and (2.17) respectively. $\boldsymbol{G}_{ii}$ consists of two blocks, i.e.:

$$\boldsymbol{G}_{ii}=[\boldsymbol{G}_{A_{ii}},\boldsymbol{G}_{B_{ii}}], \tag{2.27}$$

where $\boldsymbol{G}_{A_{ii}}$, $\boldsymbol{G}_{B_{ii}}$ and $\boldsymbol{g}_{ii}$ can be constructed following equations (2.19), (2.21) and (2.24), respectively.

From equations (2.25b) and (2.25c), it can be seen that the constraints of decentralized local MPC problem only have the information for local states and inputs. This indicates that the i^{th} local MPC controller is not aware of other subsystems' existence and their actions.

This lack of awareness will cause plant-wide performance to degrade when interaction is present. Therefore, if the performance of centralized MPC (2.14) is to be achieved, the decentralized MPC has to be modified. This has been done by both DMPCs in this work, which will be introduced in the following chapters.

2.5 Coordinated, Distributed MPC

This section provides a short introduction to the basic structure of coordinated, distributed MPC and hopes to give the readers some general idea of the DMPC scheme. Since CDMPC is a two-level hierarchical control scheme, the design of a CDMPC also includes two parts: one for the coordinator and the other for the local controllers. While the design of local MPC controllers has a general formulation, the design of the coordinator varies in each specific CDMPC method. §2.5.1 introduces the formulation of the local controllers and §2.5.2 will provide a discussion of the general design of a CDMPC method.

2.5.1 General Formulation of Local Controllers in CDMPC

The general structure of the local controllers in a coordinated DMPC used in this thesis is given in Mohseni, 2013. Adapting it to unconstrained systems yields, for the i^{th} local controller:

$$\min_{X_i,\Delta U_i} \mathcal{J}_i = \frac{1}{2}\left((X_i - \boldsymbol{r}_i)^T \boldsymbol{Q}_i (X_i - \boldsymbol{r}_i) + \Delta U_i^T \boldsymbol{R}_i \Delta U_i\right) + \{CoorTm\}_i \tag{2.28a}$$

subject to:

$$\begin{aligned}\hat{\boldsymbol{x}}_i(k+l+1|k) = &\left(\boldsymbol{A}_{ii}\hat{\boldsymbol{x}}_i(k+l|k) + \boldsymbol{B}_{ii}(\sum_{a=0}^{l}\Delta\hat{\boldsymbol{u}}_i(k+l|k) + \boldsymbol{u}_i(k-1))\right) + \\ &\left(\sum_{j=1,j\neq i}^{j=N}(\alpha \boldsymbol{A}_{ij}\boldsymbol{x}_j(k) + \boldsymbol{B}_{ij}\boldsymbol{u}_j(k-1))\right) + \hat{\boldsymbol{v}}_i(k+l|k),\end{aligned} \tag{2.28b}$$

$$\begin{cases} l = 0, 1, ..., H_p - 1, \\ \Delta\hat{\boldsymbol{u}}_i(k+l|k) = \hat{\boldsymbol{u}}_i(k+l|k) - \hat{\boldsymbol{u}}_i(k+l-1|k), \\ \hat{\boldsymbol{u}}_i(k-1|k) = \boldsymbol{u}_i(k-1), \\ \Delta\hat{\boldsymbol{u}}_i(k+b|k) = \boldsymbol{0}, H_u \leq b \leq H_p - 1 \\ \hat{\boldsymbol{x}}_i(k|k) = \boldsymbol{x}_i(k) \\ \alpha = \begin{cases} 1, \text{if } l = 0, \\ 0, \text{if } l = 1, ..., H_p - 1, \end{cases} \end{cases} \tag{2.28c}$$

where $\{CoorTm\}_i$ is the so-called **coordinating term(s)** for the i^{th} subsystem. The purpose of adding this term to the original decentralized MPC objective (2.26a), is to create a connection between the local controller and the coordinator. $\{CoorTm\}_i$ contains **coordinating variables** that are temporarily fixed in the local optimization problems. The local controllers send their optimal solutions in the current communication cycle to the coordinator, who, based on this information, will update the coordinating variables and send them back to the local controllers. This process is repeated until a termination criterion is met.

The equality constraint (2.28b), by comparison with the decentralized MPC constraints (2.25b), has two additional terms: $\left(\sum_{j=1,j\neq i}^{j=N}(\alpha \boldsymbol{A}_{ij}\boldsymbol{x}_j(k)+\boldsymbol{B}_{ij}\boldsymbol{u}_j(k-1))\right)$ and $\hat{\boldsymbol{v}}_i(k+l|k)$. The first term, $\left(\sum_{j=1,j\neq i}^{j=N}(\alpha \boldsymbol{A}_{ij}\boldsymbol{x}_j(k)+\boldsymbol{B}_{ij}\boldsymbol{u}_j(k-1))\right)$, contains only measured information for other subsystems at time k, and is referred to as the **known interaction** in Mohseni, 2013. The second term $\hat{\boldsymbol{v}}_i(k+l|k)$ is the interaction variable of the i^{th} subsystem, which would be calculated either by the local controllers or the coordinator.

Note that, in (2.28b), if

$$\hat{\boldsymbol{v}}_i(k+l|k)=\sum_{j=1,j\neq i}^{j=N}\left((1-\alpha)\boldsymbol{A}_{ij}\hat{\boldsymbol{x}}_j(k+l|k)+\boldsymbol{B}_{ij}\sum_{a=0}^{l}\Delta\hat{\boldsymbol{u}}_j(k+a|k)\right),$$
$$\begin{cases} l=0,1,...,H_p-1,\\ \alpha=\begin{cases}1, \text{if } l=0,\\ 0, \text{if } l=1,...,H_p-1,\end{cases}\\ i=1,...,N,\end{cases} \tag{2.29}$$

the constraints (2.28b) and (2.28c) would be exactly the constraints (2.11b) and (2.11c) in the plant-wide optimization problem; however, in a decentralized network, the i^{th} controller will not provide exactly the same prediction $\hat{\boldsymbol{x}}_j(k+l|k)$ for other subsystems as in (2.11a) to (2.11c), unless each subsystem solves the centralized problem, which is certainly not the case. In practice, $\hat{\boldsymbol{v}}_i(k+l|k)$ is an estimation of the predicted interaction from solving the centralized problem as in the RHS of (2.29).

2.5.2 Design of CDMPC

As stated in Mohseni, 2013, the key in designing a CDMPC method is to answer the following two questions: 1) how the coordinating terms $\{CoorTm\}_i$'s are defined and 2) how $\{CoorTm\}_i$'s are calculated. The first question will lead to the modification of the existing decentralized MPC network and the second to the development of the coordinator.

When the sum of the coordinating terms of the CDMPC network is put to zero, i.e., $\sum_{i=1}^{N}\{CoorTm\}_i=0$, the aggregated objective of the distributed control network would be exactly the same as the centralized control objective (2.14a). If it is the case, with the interaction variable $\hat{\boldsymbol{v}}_i(k+l|k)$ being the centralized predicted interaction as in the RHS of (2.29), the aggregation of the distributed control problems (2.28) from $i = 1$ to N becomes the centralized problem (2.14a) to (2.14b). Since in practice $\hat{\boldsymbol{v}}_i(k+l|k)$ is only an estimation of the interaction, $\{CoorTm\}_i$ should be able to compensate this estimation.

The task of the coordinator is to build a numerical path that can drive the sum of $\{CoorTm\}_i$ to zero through the iterative communication between the coordinator and the local controllers.

For the two types of CDMPC discussed in this thesis, the designs of the local controllers and the coordinator will be introduced in §3.1 and §4.1, respectively.

Chapter 3

Computational Properties of Prediction-driven CDMPC

In this chapter, convergence conditions, rate of convergence and the computational complexity are derived for unconstrained prediction-driven CDMPC. Before any derivation begins, a short introduction to prediction-driven CDMPC is presented. Note that for prediction-driven CDMPC, key information communicated between coordinator and local controllers is the predicted states and inputs, as well as the price vector.

3.1 Prediction-driven CDMPC

In Marcos, 2011, a prediction-driven CDMPC for unconstrained linear systems was proposed, where the $\{CoorTm\}_i$ in the general form (2.28a) is defined as $\boldsymbol{p}^{(s)^T}\boldsymbol{\Theta}_i \left[\begin{array}{c} X_i \\ \Delta U_i \end{array}\right]$. Following (Marcos, 2011), the local distributed MPC controller is formulated as follows:

$$\min_{X_i,\Delta U_i} \mathcal{J}_i = \frac{1}{2}\left((X_i-\boldsymbol{r}_i)^T\boldsymbol{Q}_i(X_i-\boldsymbol{r}_i)+\Delta U_i^T\boldsymbol{R}_i\Delta U_i\right)+\boldsymbol{p}^{(s)^T}\boldsymbol{\Theta}_i\left[\begin{array}{c} X_i \\ \Delta U_i \end{array}\right] \tag{3.1a}$$

subject to:

$$\boldsymbol{G}_{ii}\left[\begin{array}{c} X_i \\ \Delta U_i \end{array}\right]=\boldsymbol{g}_i-V_i, \tag{3.1b}$$

where $\boldsymbol{p}^{(s)}$, the estimation of centralized MPC problem's Lagrange multipliers, is a $H_pn\times 1$ vector and will be referred to as the 'price vector'. Its superscript '(s)' indicates that it is a coordinating variable calculated by the coordinator at the s^{th} communication cycle and communicated back to the local controller. The price vector $\boldsymbol{p}^{(s)}$ is the same for all subsystems; therefore, it does not have subscript i. $\boldsymbol{\Theta}_i$ is a $H_pn\times(H_pn_i+H_uq_i)$ matrix built in the following block-wise fashion:

$$\boldsymbol{\Theta}_i = \begin{bmatrix} \boldsymbol{G}_{1i} \\ \vdots \\ \boldsymbol{O} \\ \vdots \\ \boldsymbol{G}_{Ni} \end{bmatrix} \leftarrow \begin{array}{c} i^{th} \text{ block is a} \\ H_p n_i \times (H_p n_i + H_u q_i) \\ \text{zero matrix,} \end{array} \tag{3.2}$$

where $\boldsymbol{G}_{ji} = [\boldsymbol{G}_{A_{ji}}, \boldsymbol{G}_{B_{ji}}], j \neq i$, and $\boldsymbol{G}_{A_{ji}}$, $\boldsymbol{G}_{B_{ji}}$ were defined in equations (2.20) and (2.21), with reversed subscript 'i' and 'j'. Equation (3.1b) is the compact matrix form of constraints (2.28b) and (2.28c), where $\boldsymbol{G}_{ii}$ was defined in equations (2.27), and $\boldsymbol{g}_i$ in (2.23). V_i in equation (3.1b) is the estimated predicted interaction on the i^{th} subsystem. It is defined as $V_i = [\hat{\boldsymbol{v}}_i(k|k)^T, ..., \hat{\boldsymbol{v}}_i(k + H_p - 1|k)^T]^T$. In this CDMPC method, V_i is calculated by the local controller with the following equation:

$$V_i = \sum_{j \neq i} \boldsymbol{G}_{ij} \begin{bmatrix} X_j^{(s)} \\ \Delta U_j^{(s)} \end{bmatrix}, \tag{3.3}$$

where $\boldsymbol{G}_{ij} = [\boldsymbol{G}_{A_{ij}}, \boldsymbol{G}_{B_{ij}}]$ and $\boldsymbol{G}_{A_{ij}}$, $\boldsymbol{G}_{B_{ij}}$ are defined in equations (2.20) and (2.21). Note that $X_j^{(s)}$ and $\Delta U_j^{(s)}$ are both coordinating variables determined by coordinator. Equation (3.3) is the matrix form adapted from equation (2.29). It uses the coordinator-predicted states and inputs of other subsystems to estimate the RHS of (2.29).

After the i^{th} local MPC problem is solved and optimal value for $X^*_{i(s)}$, $\Delta U^*_{i(s)}$ are obtained, the local controller sends $\Delta U^*_{i(s)}$ to the coordinator. The '(s)' in the subscripts is used to denote that these optimal values are calculted in the s^{th} communication cycle by the local controllers. This notation should not be confused with that of coordinating variables, which put '(s)' in the superscript.

The coordinator increases the cycle counter $s \leftarrow s+1$, collects $\Delta U^*_{i(s-1)}, i = 1, ..., N$ and assigns

$$\Delta U_i^{(s)} = \Delta U^*_{i(s-1)}, \tag{3.4}$$

$$\Delta U^{(s)} = \begin{bmatrix} \Delta U_1^{(s)} \\ \vdots \\ \Delta U_N^{(s)} \end{bmatrix}. \tag{3.5}$$

Then, the states are predicted by the coordinator using the model of the entire plant model (2.14b):

$$\boldsymbol{G}_A X^{(s)} = \boldsymbol{g} - \boldsymbol{G}_B \Delta U^{(s)}, \tag{3.6}$$

where $\boldsymbol{G}_A$ and $\boldsymbol{G}_B$ have been defined in equation (2.18). After the predicted states are updated, the coordinator uses both $X^{(s)}$ and $U^{(s)}$ to obtain the updated price vector $\boldsymbol{p}^{(s)}$:

$$\boldsymbol{G}^T\boldsymbol{p}^{(s)} = -\begin{bmatrix} \boldsymbol{Q} & \\ & \boldsymbol{R} \end{bmatrix}\begin{bmatrix} X^{(s)} \\ \Delta U^{(s)} \end{bmatrix} + \begin{bmatrix} \boldsymbol{Q}^T\boldsymbol{r} \\ \boldsymbol{0}_{H_u q} \end{bmatrix}, \tag{3.7}$$

where $\boldsymbol{G} = [\boldsymbol{G}_A, \boldsymbol{G}_B]$.

It can be seen that (3.6) and (3.7) are derived from the first-order optimality conditions of problem (2.14):

$$\begin{cases} \begin{bmatrix} \boldsymbol{Q} & \\ & \boldsymbol{R} \end{bmatrix}\begin{bmatrix} X \\ \Delta U \end{bmatrix} - \begin{bmatrix} \boldsymbol{Q}^T\boldsymbol{r} \\ \boldsymbol{0}_{H_u q} \end{bmatrix} + \boldsymbol{G}^T\boldsymbol{\lambda} = \boldsymbol{0}_{H_p n + H_u q}, \\ \boldsymbol{G}\begin{bmatrix} X \\ \Delta U \end{bmatrix} = \boldsymbol{g}, \end{cases} \tag{3.8}$$

where $\boldsymbol{\lambda}$ are the Lagrange multipliers of this optimization problem; however, unlike in (3.8) where X, ΔU and λ are solved for simultaneously, only $X^{(s)}$ and $\boldsymbol{p}^{(s)}$ are determined from (3.6) and (3.7) and in the coordinator $\Delta U^{(s)}$, $X^{(s)}$ and $\boldsymbol{p}^{(s)}$ are obtained subsequently. Therefore, while (3.8) is a system with equal number of equations and variables, (3.6) and (3.7) is an overdetermined linear system, i.e., a system with more equations than variables. Specifically, since $X^{(s)}$ is calculated first by

$$X^{(s)} = \boldsymbol{G}_A^{-1}(\boldsymbol{g} - \boldsymbol{G}_B\Delta U^{(s)}), \tag{3.9}$$

where $\boldsymbol{G}_A$ is a full rank square matrix (for proof of $\boldsymbol{G}_A$ being full rank, please refer to Appendix A.1), only (3.7) is an overdetermined linear system. Therefore, $\boldsymbol{p}^{(s)}$ can only be obtained through approximation. The most common approximation technique is the least-squares approximation, which will give:

$$\boldsymbol{p}^{(s)} = (\boldsymbol{G}\boldsymbol{G}^T)^{-1}\boldsymbol{G}\left(\begin{bmatrix} \boldsymbol{Q}^T\boldsymbol{r} \\ \boldsymbol{0}_{H_u q} \end{bmatrix} - \begin{bmatrix} \boldsymbol{Q} & \\ & \boldsymbol{R} \end{bmatrix}\begin{bmatrix} X^{(s)} \\ \Delta U^{(s)} \end{bmatrix}\right), \tag{3.10}$$

where $\boldsymbol{G}\boldsymbol{G}^T$ is invertible since it has a full row rank. The detailed proof can be found in Appendix A.2. Equation (3.10) can be written in a more concise manner, by eliminating the zero matrices:

$$\begin{aligned} \boldsymbol{p}^{(s)} &= (\boldsymbol{G}\boldsymbol{G}^T)^{-1}(\boldsymbol{G}_A\boldsymbol{Q}^T\boldsymbol{r} - \boldsymbol{G}_A\boldsymbol{Q}X^{(s)} - \boldsymbol{G}_B\boldsymbol{R}\Delta U^{(s)}) \\ &= (\boldsymbol{G}\boldsymbol{G}^T)^{-1}(\boldsymbol{G}_A\boldsymbol{Q}\boldsymbol{r} - \boldsymbol{G}_A\boldsymbol{Q}X^{(s)} - \boldsymbol{G}_B\boldsymbol{R}\Delta U^{(s)}). \end{aligned} \tag{3.11}$$

Note that in the previous chapter, it was shown that $\boldsymbol{Q} = \boldsymbol{Q}^T$.

After the coordinator calculates $X^{(s)}$ and $\boldsymbol{p}^{(s)}$, it sends $X^{(s)}$, $\Delta U^{(s)}$ and $\boldsymbol{p}^{(s)}$ to the local controllers. The i^{th} local MPC controller then computes V_i according to (3.3) and solves optimization problem (3.1). It is assumed that the local controllers have all of the information needed to partition $X^{(s)}$ and $\Delta U^{(s)}$ (which is reasonable, since the i^{th} controller just needs to store $\boldsymbol{G}_{A_{ij}}$ and $\boldsymbol{G}_{B_{ij}}$ to calculate equation (3.3) and their dimensions can be easily obtained), so that the coordinator does not need to go through the partition and information selection for each specific subsystem as described in Marcos, 2011.

The above iteration process is terminated when the coordinator determines that $\|\Delta U^{(s)} - \Delta U^{(s-1)}\| < \epsilon$, where ϵ is a pre-defined accuracy threshold.

The general algorithm of this prediction-driven coordinated DMPC scheme comes is:

Algorithm 1 Implementation of Prediction-driven CDMPC

```
**Initialization**
Coordinator: Iteration counter $s \leftarrow 0$.
Coordinator: The coordinating variables $X^{(0)}$, $\Delta U^{(0)}$ and $\boldsymbol{p}^{(0)}$ are arbitrarily determined.
**repeat**
    Coordinator: $X^{(s)}$, $\Delta U^{(s)}$ and $\boldsymbol{p}^{(s)}$ are sent to to local controllers.
    Local Controllers: Local optimization problem (3.1) is solved.
    Local Controllers: The optimal solutions $\Delta U^*_{i(s)}$ are sent to coordinator.
    Coordinator: Iteration counter $s \leftarrow s + 1$.
    Coordinator: Calculate $\Delta U^{(s)}$, $X^{(s)}$ and $\boldsymbol{p}^{(s)}$ based on (3.4) to (3.5), (3.6) and (3.7), with respect.
**until** $\|\Delta U^{(s)} - \Delta U^{(s-1)}\| < \epsilon$
```

3.2 Convergence Analysis

Though the convergence accuracy of prediction-driven, coordinated MPC has already been studied and proved in Marcos, 2011, no work has been done regarding its rate of convergence, another important feature in convergence analysis. In this section we begin with writing the algorithm in the form of an iterative method, which will lead to an easy-to-compute convergence condition and eventually the algorithm's rate of convergence.

3.2.1 An Iterative Formulation

The prediction-driven CDMPC algorithm is iterative in nature. Specifically, the coordinating variable $\Delta U^{(s+1)}$ can be written as a function solely dependent on $\Delta U^{(s)}$:

$$\Delta U^{(s+1)} = \phi(\Delta U^{(s)}). \tag{3.12}$$

Since the research scope of this thesis is limited to unconstrained linear systems, an explicit expression of function ϕ can be found, which is also linear. The following sections give a detailed, step-by-step analysis to find out the expression of function ϕ.

Local MPC Controllers

We begin with analysis of the local MPC controllers. In the s^{th} communication cycle, $X^{(s)}$, $\Delta U^{(s)}$ and $\boldsymbol{p}^{(s)}$ are sent to local MPC controllers. Then, the coordinating term in the i^{th} local optimization problem is ${\boldsymbol{p}^{(s)}}^T \boldsymbol{\Theta}_i \begin{bmatrix} X_i \\ \Delta U_i \end{bmatrix}$ and V_i in the constraint is $\sum_{j \neq i} \boldsymbol{G}_{ij} \begin{bmatrix} X_j^{(s)} \\ \Delta U_j^{(s)} \end{bmatrix}$. The optimal solution of problem (3.1) can be obtained by writing down its optimality condition and solving the linear equations:

$$\begin{cases} \begin{bmatrix} \boldsymbol{Q}_i & \\ & \boldsymbol{R}_i \end{bmatrix} \begin{bmatrix} X_i \\ \Delta U_i \end{bmatrix} - \begin{bmatrix} \boldsymbol{Q}_i^T \boldsymbol{r} \\ \boldsymbol{0}_{H_u q_i} \end{bmatrix} + \boldsymbol{\Theta}_i^T \boldsymbol{p}^{(s)} + \boldsymbol{G}_{ii}^T \boldsymbol{\lambda}_i = \boldsymbol{0}_{H_p n_i + H_u q_i}, \\ \boldsymbol{G}_{ii} \begin{bmatrix} X_i \\ \Delta U_i \end{bmatrix} = \boldsymbol{g}_i - V_i. \end{cases} \tag{3.13}$$

Note that $\boldsymbol{Q}_i = \boldsymbol{Q}_i^T$. Using the partitions $\boldsymbol{G}_{ij} = [\boldsymbol{G}_{A_{ij}}, \boldsymbol{G}_{B_{ij}}]$, $\boldsymbol{\Theta}_i = [\boldsymbol{\Theta}_{A_i}, \boldsymbol{\Theta}_{B_i}]$ for $i, j = 1, ..., N$ and replacing V_i with the RHS of equation (3.3), equation (3.13) can be rewritten as:

$$\begin{cases} -\boldsymbol{Q}_i \boldsymbol{r}_i + \boldsymbol{Q}_i X_i + \boldsymbol{\Theta}_{A_i}^T \boldsymbol{p}^{(s)} + \boldsymbol{G}_{A_{ii}}^T \boldsymbol{\lambda}_i &= \boldsymbol{0}_{H_p n_i}, \\ \boldsymbol{R}_i \Delta U_i + \boldsymbol{\Theta}_{B_i}^T \boldsymbol{p}^{(s)} + \boldsymbol{G}_{B_{ii}}^T \boldsymbol{\lambda}_i &= \boldsymbol{0}_{H_u q_i}, \\ \boldsymbol{G}_{A_{ii}} X_i + \boldsymbol{G}_{B_{ii}} \Delta U_i + \sum_{j \neq i} (\boldsymbol{G}_{A_{ij}} X_j^{(s)} + \boldsymbol{G}_{B_{ij}} \Delta U_j^{(s)}) &= \boldsymbol{g}_i. \end{cases} \tag{3.14}$$

where $\boldsymbol{\lambda}_i$ are the Lagrange multipliers of problem (3.1a) to (3.1b), and $\boldsymbol{\Theta}_{A_i}$ and $\boldsymbol{\Theta}_{B_i}$ are defined as:

$$\boldsymbol{\Theta}_{A_i} = \begin{bmatrix} \boldsymbol{G}_{A_{1i}} \\ \vdots \\ \boldsymbol{O}_{H_p n_i \times H_p n_i} \\ \vdots \\ \boldsymbol{G}_{A_{Ni}} \end{bmatrix}, \boldsymbol{\Theta}_{B_i} = \begin{bmatrix} \boldsymbol{G}_{B_{1i}} \\ \vdots \\ \boldsymbol{O}_{H_p n_i \times H_u q_i} \\ \vdots \\ \boldsymbol{G}_{B_{Ni}} \end{bmatrix}. \tag{3.15}$$

Equation (3.14) can be aggregated from $i = 1$ to N and written in a plant-wide fashion, as:

$$\begin{cases} -\boldsymbol{Q}\boldsymbol{r} + \boldsymbol{Q} X_{MPC} + \bar{\boldsymbol{\Theta}}_A^T \boldsymbol{p}^{(s)} + \bar{\boldsymbol{G}}_A^T \boldsymbol{\lambda}_{MPC} &= \boldsymbol{0}_{H_p n}, \\ \boldsymbol{R} \Delta U_{MPC} + \bar{\boldsymbol{\Theta}}_B^T \boldsymbol{p}^{(s)} + \bar{\boldsymbol{G}}_B^T \boldsymbol{\lambda}_{MPC} &= \boldsymbol{0}_{H_u q}, \\ \bar{\boldsymbol{G}}_A X_{MPC} + \bar{\boldsymbol{G}}_B \Delta U_{MPC} + \bar{\boldsymbol{\Theta}}_A X^{(s)} + \bar{\boldsymbol{\Theta}}_B \Delta U^{(s)} &= \boldsymbol{g}, \end{cases} \tag{3.16}$$

where $X_{MPC} = [X_1^T, \cdots, X_N^T]^T$, $\Delta U_{MPC} = [\Delta U_1^T, \cdots, \Delta U_N^T]^T$ and $\boldsymbol{\lambda}_{MPC} = [\boldsymbol{\lambda}_1^T, \cdots, \boldsymbol{\lambda}_N^T]^T$. Their subscript '$MPC$' distinguishes the solutions from the centralized solutions of the first-order optimality condition (3.8) and indicates that they are the aggregation of local MPC solutions.

$\bar{\boldsymbol{G}}_A$, $\bar{\boldsymbol{G}}_B$, $\bar{\boldsymbol{\Theta}}_A$ and $\bar{\boldsymbol{\Theta}}_B$ are respectively defined as:

$$\bar{\boldsymbol{G}}_A = diag(\boldsymbol{G}_{A_{11}}, ..., \boldsymbol{G}_{A_{NN}}), \tag{3.17}$$

$$\bar{\boldsymbol{G}}_B = diag(\boldsymbol{G}_{B_{11}}, ..., \boldsymbol{G}_{B_{NN}}), \tag{3.18}$$

$$\bar{\boldsymbol{\Theta}}_A = \boldsymbol{G}_A - \bar{\boldsymbol{G}}_A, \tag{3.19}$$

$$\bar{\boldsymbol{\Theta}}_B = \boldsymbol{G}_B - \bar{\boldsymbol{G}}_B, \tag{3.20}$$

where $\boldsymbol{G}_{A_{ii}}$ and $\boldsymbol{G}_{B_{ii}}$ have been defined in equations (2.19) and (2.21), respectively.

Equation (3.16) has an unique solution as long as the matrix:

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{O} & \bar{\boldsymbol{G}}_A^T \\ \boldsymbol{O} & \boldsymbol{R} & \bar{\boldsymbol{G}}_B^T \\ \bar{\boldsymbol{G}}_A & \bar{\boldsymbol{G}}_B & \boldsymbol{O} \end{bmatrix} \tag{3.21}$$

is invertible. Given the way that $\boldsymbol{G}_{A_{ii}}$'s are built, it can be proven that $\boldsymbol{W}$ is invertible (please refer to Appendix A.3 for detailed proof). Therefore, we are able to solve for the unknown variables X_{MPC}, ΔU_{MPC} and $\boldsymbol{\lambda}_{MPC}$.

First, X_{MPC} and ΔU_{MPC} are written as linear functions of $\boldsymbol{\lambda}_{MPC}$, using the first two equations in (3.16):

$$X_{MPC} = -\boldsymbol{Q}^{-1}(\bar{\boldsymbol{\Theta}}_A^T \boldsymbol{p}^{(s)} + \bar{\boldsymbol{G}}_A^T \boldsymbol{\lambda}_{MPC}) + \boldsymbol{r}, \tag{3.22}$$

$$\Delta U_{MPC} = -\boldsymbol{R}^{-1}(\bar{\boldsymbol{\Theta}}_B^T \boldsymbol{p}^{(s)} + \bar{\boldsymbol{G}}_B^T \boldsymbol{\lambda}_{MPC}), \tag{3.23}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are both positive definite and thus invertible. By substituting equations (3.22), (3.23) into the third equation in (3.16), $\boldsymbol{\lambda}_{MPC}$ can be represented as a linear combination of the coordinating variables $X^{(s)}$, $\Delta U^{(s)}$ and $\boldsymbol{p}^{(s)}$:

$$\boldsymbol{\lambda}_{MPC} = \boldsymbol{\Psi}^{-1}(\bar{\boldsymbol{G}}_A \boldsymbol{r} - \boldsymbol{\Omega} \boldsymbol{p}^{(s)} + \bar{\boldsymbol{\Theta}}_A X^{(s)} + \bar{\boldsymbol{\Theta}}_B \Delta U^{(s)} - \boldsymbol{g}), \tag{3.24}$$

where:

$$\boldsymbol{\Psi} = \bar{\boldsymbol{G}}_A \boldsymbol{Q}^{-1} \bar{\boldsymbol{G}}_A^T + \bar{\boldsymbol{G}}_B \boldsymbol{R}^{-1} \bar{\boldsymbol{G}}_B^T, \tag{3.25}$$

$$\boldsymbol{\Omega} = \bar{\boldsymbol{G}}_A \boldsymbol{Q}^{-1} \bar{\boldsymbol{\Theta}}_A^T + \bar{\boldsymbol{G}}_B \boldsymbol{R}^{-1} \bar{\boldsymbol{\Theta}}_B^T. \tag{3.26}$$

The invertibility of matrix $\boldsymbol{\Psi}$ is equivalent to that of matrix $\boldsymbol{W}$, the detailed proof of which can also be found in Appendix A.3.

From equations (3.24) and (3.23), we have:

$$\Delta U_{MPC} = -\boldsymbol{R}^{-1} \bar{\boldsymbol{\Theta}}_B^T \boldsymbol{p}^{(s)} - \boldsymbol{R}^{-1} \bar{\boldsymbol{G}}_B^T \boldsymbol{\Psi}^{-1}(\bar{\boldsymbol{G}}_A \boldsymbol{r} - \boldsymbol{\Omega} \boldsymbol{p}^{(s)} + \bar{\boldsymbol{\Theta}}_A X^{(s)} + \bar{\boldsymbol{\Theta}}_B \Delta U^{(s)} - \boldsymbol{g}). \tag{3.27}$$

Note that the procedure for solving $\boldsymbol{\lambda}_{MPC}$, ΔU_{MPC} and X_{MPC} sequentially presented here is equivalent to solving them simultaneously using the inversion of $\boldsymbol{W}$:

$$\begin{bmatrix} X_{MPC} \\ \Delta U_{MPC} \\ \boldsymbol{\lambda}_{MPC} \end{bmatrix} = \boldsymbol{W}^{-1} \begin{bmatrix} \boldsymbol{Qr} - \bar{\boldsymbol{\Theta}}_A^T \boldsymbol{p}^{(s)} \\ -\bar{\boldsymbol{\Theta}}_B^T \boldsymbol{p}^{(s)} \\ \boldsymbol{g} - \bar{\boldsymbol{\Theta}}_A^T X^{(s)} - \bar{\boldsymbol{\Theta}}_B^T \Delta U^{(s)} \end{bmatrix}, \tag{3.28}$$

because the dimensions of X_{MPC}, ΔU_{MPC} and $\boldsymbol{\lambda}_{MPC}$ are the same as those of the first, second and third equation in (3.16), respectively. It is easier to compute and to understand each step when the unknown variables are solved for sequentially.

ΔU_{MPC} obtained from (3.27) will be sent to the coordinator, where it will be taken as $\Delta U^{(s+1)}$. Therefore, equation (3.27) can be rewritten with $\Delta U^{(s+1)}$ being a linear function of $\Delta U^{(s)}$, $X^{(s)}$ and $\boldsymbol{p}^{(s)}$:

$$\begin{aligned} \Delta U^{(s+1)} =& - \boldsymbol{R}^{-1} \bar{\boldsymbol{G}}_B^T \boldsymbol{\Psi}^{-1} \bar{\boldsymbol{\Theta}}_B \Delta U^{(s)} - \boldsymbol{R}^{-1} \bar{\boldsymbol{G}}_B^T \boldsymbol{\Psi}^{-1} \bar{\boldsymbol{\Theta}}_A X^{(s)} + \\ & \boldsymbol{R}^{-1} (\bar{\boldsymbol{G}}_B^T \boldsymbol{\Psi}^{-1} \boldsymbol{\Omega} - \bar{\boldsymbol{\Theta}}_B^T) \boldsymbol{p}^{(s)} + \boldsymbol{R}^{-1} \bar{\boldsymbol{G}}_B^T \boldsymbol{\Psi}^{-1} (\boldsymbol{g} - \bar{\boldsymbol{G}}_A \boldsymbol{r}). \end{aligned} \tag{3.29}$$

Equation (3.29) shows the structure of iteration, in which the predicted input in a new communication cycle ($\Delta U^{(s+1)}$) is determined from the variables in the current cycle ($\Delta U^{(s)}$, $X^{(s)}$ and $\boldsymbol{p}^{(s)}$). While only the first term contains $\Delta U^{(s)}$ among the RHS terms of equation (3.29), $X^{(s)}$ and $\boldsymbol{p}^{(s)}$ can be represented in terms of $\Delta U^{(s)}$ from the computation in the coordinator. This will lead to an iterative formulation with $\Delta U^{(s)}$ being the only variable, as will be seen in the following section.

Coordinator

In §3.1 it was discussed that in the coordinator, the plant-wide predicted states $X^{(s)}$ are calculated by equation (3.9) and the predicted price $\boldsymbol{p}^{(s)}$ is calculated by (3.11). Equation (3.9) shows that $X^{(s)}$ is a linear function of $\Delta U^{(s)}$. The price vector $\boldsymbol{p}^{(s)}$ can be written as a linear function of $\Delta U^{(s)}$ as well, if equation (3.9) is substituted into (3.11):

$$\boldsymbol{p}^{(s)} = (\boldsymbol{G}\boldsymbol{G}^T)^{-1} \boldsymbol{G}_A \boldsymbol{Q} (\boldsymbol{r} - \boldsymbol{G}_A^{-1} \boldsymbol{g}) + (\boldsymbol{G}\boldsymbol{G}^T)^{-1} (\boldsymbol{G}_A \boldsymbol{Q} \boldsymbol{G}_A^{-1} \boldsymbol{G}_B - \boldsymbol{G}_B \boldsymbol{R}) \Delta U^{(s)}. \tag{3.30}$$

By replacing $X^{(s)}$ and $\boldsymbol{p}^{(s)}$ in equation (3.29) with the RHS of equation (3.9) and equation (3.30), we get the iterative formulation of $\Delta U^{(s)}$, as previously mentioned:

$$\Delta U^{(s+1)} = \mathbf{c_1} + \mathbf{C_2} \Delta U^{(s)}, \tag{3.31}$$

where:

$$\begin{aligned} \mathbf{c_1} =& \boldsymbol{R}^{-1} \bar{\boldsymbol{G}}_B^T \boldsymbol{\Psi}^{-1} (\boldsymbol{g} - \bar{\boldsymbol{\Theta}}_A \boldsymbol{G}_A^{-1} \boldsymbol{g} - \bar{\boldsymbol{G}}_A \boldsymbol{r}) \\ & - \boldsymbol{R}^{-1} (\bar{\boldsymbol{G}}_B^T \boldsymbol{\Psi}^{-1} \boldsymbol{\Omega} - \bar{\boldsymbol{\Theta}}_B^T) (\boldsymbol{G}\boldsymbol{G}^T)^{-1} \boldsymbol{G}_A \boldsymbol{Q} (\boldsymbol{G}_A^{-1} \boldsymbol{g} - \boldsymbol{r}) \end{aligned} \tag{3.32}$$

and :

$$\begin{aligned}\mathbf{C_2} =& \boldsymbol{R}^{-1}\bar{\boldsymbol{G}}_B^T\boldsymbol{\Psi}^{-1}(\bar{\boldsymbol{\Theta}}_A\boldsymbol{G}_A^{-1}\boldsymbol{G}_B - \bar{\boldsymbol{\Theta}}_B) \\ &+ \boldsymbol{R}^{-1}(\bar{\boldsymbol{G}}_B^T\boldsymbol{\Psi}^{-1}\boldsymbol{\Omega} - \bar{\boldsymbol{\Theta}}_B^T)(\boldsymbol{G}\boldsymbol{G}^T)^{-1}(\boldsymbol{G}_A\boldsymbol{Q}\boldsymbol{G}_A^{-1}\boldsymbol{G}_B - \boldsymbol{G}_B\boldsymbol{R}).\end{aligned} \tag{3.33}$$

In equation (3.31), the coefficient vector $\mathbf{c_1}$ is only dependent on $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{Q}, \boldsymbol{R}, H_p, H_u$ and $\boldsymbol{g}$, while the coefficient matrix $\mathbf{C_2}$ only depends on $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{Q}, \boldsymbol{R}, H_p, H_u$, i.e.:

$$\mathbf{c_1} = \mathbf{c_1}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{Q}, \boldsymbol{R}, H_p, H_u, \boldsymbol{g}), \tag{3.34}$$

and:

$$\mathbf{C_2} = \mathbf{C_2}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{Q}, \boldsymbol{R}, H_p, H_u). \tag{3.35}$$

Among the parameters of $\mathbf{c_1}$ and $\mathbf{C_2}$, $\boldsymbol{A}$ and $\boldsymbol{B}$ represent system properties and remain unchanged for a linear time-invariant system. $\boldsymbol{Q}, \boldsymbol{R}, H_p$ and H_u come from the local MPC design, which are fixed for an existing decentralized network. $\boldsymbol{g}$ contains information for $\boldsymbol{x}_i(k)$ and $\boldsymbol{u}_i(k-1)$, $i = 1, ..., N$ and is constant in the k^{th} control interval. Therefore the linear coefficients $\mathbf{c_1}$ and $\mathbf{C_2}$ are constant during the k^{th} control interval, which would make (3.31) a linear iterative method. Moreover, the coefficient matrix $\mathbf{C_2}$ will not change as the time changes from k to $k+1$. The time invariance of $\mathbf{C_2}$ is a key feature in the convergence analysis of prediction-driven CDMPC.

3.2.2 Convergence Condition

It has been proved that when the prediction-driven CDMPC converges, i.e., $\Delta U^{(s)} \to \Delta U^{(\infty)}$ as $s \to \infty$, the limit $\Delta U^{(\infty)}$ equals the centralized solution ΔU^* (Marcos, 2011). Though in (Marcos, 2011) the convergence condition is given, it is presented as an abstract function in ΔU and therefore not easy to verify. In this section, a different, explicit convergence condition is provided as in **Theorem 3.2.1**. Prior to presenting this convergence condition, **Lemma 3.2.1** is introduced first as a requirement to prove it.

Lemma 3.2.1. (Quarteroni *et al.*, 2000) *Let $\boldsymbol{M} \in \mathbb{C}^{l\times l}$ be a complex valued matrix and $\rho(\boldsymbol{M})$ is its spectral radius; then*

$$\lim_{k\to\infty} \boldsymbol{M}^k = \boldsymbol{O} \text{ if and only if } \rho(\boldsymbol{M}) < 1.$$

Theorem 3.2.1. *In the prediction-driven CDMPC algorithm* **Algorithm 1**, *$\Delta U^{(s)}$ will converge to the centralized solution ΔU^* if and only if the spectral radius of $\mathbf{C_2}$ is smaller than 1 ($\mathbf{C_2}$ defined in (3.33)), i.e., $\rho(\mathbf{C_2}) < 1$,regardless of the selection of $\Delta U^{(0)}$.*

Proof From the result in (Marcos, 2011), the limit $\Delta U^{(\infty)} \triangleq \lim_{s\to\infty} \Delta U^{(s)}$ is equal to ΔU^*. So to prove **Theorem 3.2.1**, we need to prove that the limit $\Delta U^{(\infty)}$ exists if and only if $\rho(\mathbf{C_2}) < 1$.

From equation (3.31):

$$\Delta U^{(s+1)} - \Delta U^{(s)} = \mathbf{C_2}(\Delta U^{(s)} - \Delta U^{(s-1)}), s \geq 1, \tag{3.36}$$

which indicates that:

$$\Delta U^{(s+1)} - \Delta U^{(s)} = \mathbf{C_2}^s(\Delta U^{(1)} - \Delta U^{(0)}). \tag{3.37}$$

If $\rho(\mathbf{C_2}) < 1$, according to **Lemma 3.2.1**:

$$\lim_{s\to\infty} \mathbf{C_2}^s = \boldsymbol{O}. \tag{3.38}$$

Then for (3.37):

$$\lim_{s\to\infty} LHS = \lim_{s\to\infty} RHS = \boldsymbol{O}(\Delta U^{(s)} - \Delta U^{(s-1)}) = \mathbf{0}, \tag{3.39}$$

i.e.:

$$\lim_{s\to\infty} \Delta U^{(s+1)} = \lim_{s\to\infty} \Delta U^{(s)}, \tag{3.40}$$

which proves the existence of $\Delta U^{(\infty)}$, as the limit of $\Delta U^{(s)}$ as $s \to \infty$.

When the iterative method converges to the limit,

$$\Delta U^{(\infty)} = \mathbf{c_1} + \mathbf{C_2}\Delta U^{(\infty)}. \tag{3.41}$$

Subtract equation (3.41) for (3.31) and denote:

$$\boldsymbol{e}^{(s)} = \Delta U^{(s)} - \Delta U^{(\infty)},$$

then we have:

$$\boldsymbol{e}^{(s+1)} = \mathbf{C_2}\boldsymbol{e}^{(s)}, \tag{3.42}$$

which implies by recursion that:

$$\boldsymbol{e}^{(s)} = \mathbf{C_2}^s\boldsymbol{e}^{(0)}. \tag{3.43}$$

If $\rho(\mathbf{C_2}) \geq 1$, then there exists at least one eigenvalue with its module no less than one. Denote this eigenvalue as λ. Without loss of generality, suppose $\boldsymbol{e}^{(0)}$ is the eigenvector associated with λ, then we have $\mathbf{C_2}\boldsymbol{e}^{(0)} = \lambda\boldsymbol{e}^{(0)}$ and $\boldsymbol{e}^{(s)} = \lambda^s\boldsymbol{e}^{(0)}$. Since in general $\boldsymbol{e}^{(0)} \neq \mathbf{0}$, we have:

$$\lim_{s\to\infty} \boldsymbol{e}^{(s)} = \lim_{s\to\infty} \lambda^s\boldsymbol{e}^{(0)} = \begin{cases} \boldsymbol{e}^{(0)}, \text{if } \lambda = 1, \\ \infty, \text{if } \lambda > 1, \end{cases}$$

but by definition $\lim_{s\to\infty} \boldsymbol{e}^{(s)} = \Delta U^{(\infty)} - \Delta U^{(\infty)} = 0$, which is a contradiction. Therefore it is proved that the iterative algorithm will converge to a limit $\Delta U^{(\infty)}$ only if $\rho(\mathbf{C_2}) < 1$. $\square$

Since the matrix $\mathbf{C_2}$ is time invariant, the convergence can be checked off-line if the prediction-driven CDMPC algorithm is to be implemented on a given decentralized network.

3.2.3 Rate of Convergence

Convergence of the prediction-driven algorithm is a basic requirement of successful implementation; however, it may take a long time for the coordinating algorithm to converge, even though the condition in **Theorem 3.2.1** is satisfied. Thus, the rate of convergence of an iterative method, which describes the speed of the method converging to its limit, is essential as well. In numerical testing, we found those examples that are slow to converge always have large spectral radius for $\mathbf{C_2}$, while those are faster with smaller spectral radius. As would be expected, the rate of convergence is closely related to $\rho(\mathbf{C_2})$.

To quantify the relation between the convergence rate and spectral radius, the analyzing procedure in Quarteroni *et al.*, 2000 is followed. First we introduce **Lemma 3.2.2**:

Lemma 3.2.2. (Quarteroni *et al.*, 2000) *For any matrix norm $\|\cdot\|$ and square matrix $\boldsymbol{M}$, there is:*

$$\rho(\boldsymbol{M}) = \lim_{k\to\infty} \|\boldsymbol{M}^k\|^{1/k}.$$

Next the following definition is given:

Definition 3.2.1. *In a linear iterative formula $\boldsymbol{z}^{(k+1)} = \boldsymbol{H}\boldsymbol{z}^{(k)} + \boldsymbol{f}$, $R_k(\boldsymbol{H}) = -\frac{1}{k}\log\|\boldsymbol{H}^k\|$ is called its* ***average convergence rate*** *after* ***k*** *steps.*

The average convergence rate is usually expensive to compute, because $\boldsymbol{H}^k$ has to be calculated in advance and it varies with k; however, with **Lemma 3.2.2** it is easy to compute the limit of R_k, which is defined as the **asymptotic convergence rate**:

$$R(\boldsymbol{H}) = \lim_{k\to\infty} R_k(\boldsymbol{H}) = -\log\rho(\boldsymbol{H}). \tag{3.44}$$

$R(\boldsymbol{H})$ represents the average convergence rate as the number of iterations goes to infinity, and is more commonly used.

Since the prediction-driven CDMPC algorithm can be formulated as an iterative method by equation (3.31), its asymptotic convergence rate is:

$$R(\mathbf{C_2}) = -\log\rho(\mathbf{C_2}), \tag{3.45}$$

where $\mathbf{C_2}$ is define in §3.2.1 . As we have expected, with larger $\rho(\mathbf{C_2})$ (closer to 1), the value of $R(\mathbf{C_2})$ is smaller, indicating the algorithm converges slower.

Remark 3.2.1. *For a sequence of scalars $\{y^{(0)}, y^{(1)}, ...\}$, if the limit y exists, the rate of convergence can be defined with the ratio of two successive errors:*

$$R = \lim_{k\to\infty} \frac{|e^{(k+1)}|}{|e^{(k)}|}, \tag{3.46}$$

where $e^{(k)} = y^{(k)} - y$. For example, in the sequence $y^{(k+1)} = \frac{1}{2}y^{(k)} + 3$, the following relationship exists:

$$e^{(k+1)} = \frac{1}{2}e^{(k)}. \tag{3.47}$$

Therefore,

$$R = \lim_{k\to\infty} \frac{|e^{(k+1)}|}{|e^{(k)}|} = \frac{\frac{1}{2}e^{(k)}}{e^{(k)}} = \frac{1}{2}. \tag{3.48}$$

In a multi-dimensional case as in the sequence $\boldsymbol{z}^{(k+1)} = \boldsymbol{H}\boldsymbol{z}^{(k)} + \boldsymbol{f}$, the successive errors follow the relationship

$$\boldsymbol{e}^{(k+s)} = \boldsymbol{H}^s \boldsymbol{e}^{(k)}, \tag{3.49}$$

where $\boldsymbol{e}^{(k)} = \boldsymbol{z}^{(k)} - \boldsymbol{z}$ and $\boldsymbol{z}$ is the limit of $\boldsymbol{z}^{(k)}$. To measure the ratio of errors, the norm operation can be employed similarly to the absolute value operation. It can be seen that $\boldsymbol{H}^s$ is a factor that decides how much $\boldsymbol{e}^{(k)}$ reduces in s steps, so the average convergence rate may be defined with $\|\boldsymbol{H}^s\|^{\frac{1}{s}}$. Note that in a scalar sequence, if the rate of convergence is smaller (closer to 0), the sequence converges faster. The situation is different in iterative methods, where a larger rate means faster convergence. This is only a matter of convention, and can be changed into concordance by reversing the numerator and denominator in equation (3.46). Since this work is discussed within the context of iterative methods, the way of defining convergence rate in this context is used, which is presented in equation (3.44).

Remark 3.2.2. *When $k < \infty$, there is $\|A^k\|^{1/k} \geq \rho(A)$ for any matrix norm, so the asymptotic convergence rate in equation (3.44) will always overestimate the actual convergence rate in practice. This will also affect estimation of the number of communication cycles in the next section.*

3.3 Computational Complexity

Computational complexity of an algorithm is used to describe the computational resources needed to solve a problem using that particular algorithm. Generally, the resourse of interest is the computing time, so the complexity T can be measured by the execution time of an algorithm as a function of problem size n. The basic assumption of complexity theory suggests that it takes exactly the same time to perform an arithmetic operation on any computing platform, so measuring T can also be done by counting the number of required arithmetic operations. In practice, the number of operations is a more consistent measurement than the execution time, since real comupting platforms vary in speed. We denote $T = F(n)$ to be the function relating the number of arithmetic operations to the problem size.

In most cases, we are only interested in the asymptotic behavior of $F(n)$, i.e., the function behavior as problem size goes to infinity. Among different types of notation developed to quantify this behavior, the most frequently used one is the O-notation, which describes an algorithm's **asymptotic upper bound** (Cormen, 2009). It is a useful tool to measure an algorithm's efficiency, as well as its relative behavior with other algorithms. Therefore, the O-notation will be used as the complexity analysis tool in this section. For the mathematical background of O-notation, please refer to Appendix B. Typically, if the complexity of an algorithm can be bounded by a polynomial, i.e., it can be written as $O(n^c)$, $c \in \mathbb{R}^+$, then the algorithm is said to have a polynomial-time complexity. A polynomial-time algorithm is considered to be efficiently solvable (Goldreich, 2008), indicating that the increased computating time, as the problem size grows, can be handled by a computer. It is a desired characteristic for DMPC algorithms.

3.3.1 Theoretical Analysis

According to Cheng, 2007, two different parts contribute to the total complexity of a coordinated DMPC algorithm: 1) the complexity within each communication cycle, which is denoted as T_{NonCo}; and 2) the number of communication cycles (CCN) needed for convergence. Since a parallel computing environment is assumed (i.e., each local MPC is solved on its own computer), the complexity within a single communication cycle can be represented as the sum of coordinator complexity and the maximum local MPC complexity within one cycle. Therefore, the complexity T of a coordinated DMPC can be represented as:

$$\begin{aligned} T &= T_{NonCo} \times CCN \\ &= (T_{coor} + \max_i T_i) \times CCN, \end{aligned} \tag{3.50}$$

where: T_{coor} represents the coordinator complexity in one cycle; T_i the complexity of the i^{th} subsystem in one cycle; and CCN the number of communication cycles. T_{coor}, T_i and CCN will be discussed in the following sections.

It is assumed that throughout this section H_p and H_u are fixed finite integers and that $H_p \geq H_u$, $n_i \geq q_i$ and $n \geq q$.

Local MPCs: Solved Analytically

For a prediction-driven CDMPC, the i^{th} local MPC solves optimization problem (3.1) in each communication cycle. Since the problem is unconstrained, the solution can be obtained analytically through solving (3.14), which is equivalent to the following equation:

$$\begin{bmatrix} X_i \\ \Delta U_i \\ \boldsymbol{\lambda}_i \end{bmatrix} = \boldsymbol{W}_i^{-1}\boldsymbol{\beta}_i, \tag{3.51}$$

where

$$\boldsymbol{\beta}_i = \begin{bmatrix} \boldsymbol{Q}_i \boldsymbol{r}_i - \boldsymbol{\Theta}_{A_i}^T \boldsymbol{p}^{(s)} \\ -\boldsymbol{\Theta}_{B_i}^T \boldsymbol{p}^{(s)} \\ \boldsymbol{g}_i - \sum_{j \neq i} \left(\boldsymbol{G}_{A_{ij}} X_j^{(s)} + \boldsymbol{G}_{B_{ij}} \Delta U_j^{(s)} \right) \end{bmatrix}, \tag{3.52}$$

$$\boldsymbol{W}_i = \begin{bmatrix} \boldsymbol{Q}_i & \boldsymbol{O} & \boldsymbol{G}_{A_{ii}}^T \\ \boldsymbol{O} & \boldsymbol{R}_i & \boldsymbol{G}_{B_{ii}}^T \\ \boldsymbol{G}_{A_{ii}} & \boldsymbol{G}_{B_{ii}} & \boldsymbol{O} \end{bmatrix}. \tag{3.53}$$

The matrix $\boldsymbol{W}_i$ is time invariant, so its inverse can be calculated off-line and stored as a constant coefficient matrix. Therefore, only matrix multiplication and summation are needed to compute equation (3.51). It is assumed that each arithmetic operation with one matrix entry has complexity $O(1)$, then two $k \times l$ matrices adding together require $O(kl)$ complexity, and a $k \times l$ matrix multiplied by a $l \times m$ matrix has $O(klm)$ complexity. In each communication cycle, we need to update the coordinating variables in $\boldsymbol{\beta}_i$ first, then do the matrix multiplication $\boldsymbol{W}_i^{-1}\boldsymbol{\beta}_i$. Table 3.1 shows the detailed steps for analytically solving the i^{th} local MPC problem, the dimensions of parameters in the computation and the complexity of each step.

The four steps listed in Table 3.1 are performed sequentially, so the complexity of the i^{th} local MPC is calculated by adding the complexities for each of the four steps together, as stated in **Property B.0.3** in Appendix B. Since complexity analysis focuses on the case when problem size is very large, only the step with largest complexity contributes to the overall complexity as the problem size goes to infinity, and complexity of other steps are neglected(**Property B.0.4**). From Table 3.1, it can be seen that the complexity of step 3 is smaller than that of step 4, so it does not need to be taken into consideration when the

Table 3.1: Local MPC (Solved Analytically) Complexity in Prediction-driven CDMPC: Step by Step

Step Description	**Parameters and Dimensions**	**Step Complexity**
1. Calculate $\boldsymbol{\theta}_i = \begin{bmatrix} \boldsymbol{\Theta}_{A_i}^T \\ \boldsymbol{\Theta}_{B_i}^T \end{bmatrix} \boldsymbol{p}^{(s)}$	$\boldsymbol{\Theta}_{A_i} : H_p n \times H_p n_i$, $\boldsymbol{\Theta}_{B_i} : H_p n \times H_u q_i$, $\boldsymbol{p}^{(s)} : H_p n \times 1$	$O(H_p n(H_p n_i + H_u q_i))$
2. Calculate $V_i = \sum_{j \neq i} \left(\boldsymbol{G}_{A_{ij}} X_j^{(s)} + \boldsymbol{G}_{B_{ij}} \Delta U_j^{(s)} \right)$	$\boldsymbol{G}_{A_{ij}} : H_p n_i \times H_p n_j$, $\boldsymbol{G}_{B_{ij}} : H_p n_i \times H_u q_j$, $X_j^{(s)} : H_p n_j \times 1$, $\Delta U_j^{(s)} : H_u q_j \times 1$	$O\left((H_p n_i) \sum_{j \neq i} (H_p n_j + H_u q_j) \right)$
3. Calculate $\boldsymbol{\beta}_i = \begin{bmatrix} \boldsymbol{Q}_i \boldsymbol{r}_i \\ \boldsymbol{0}_{H_u q_i} \\ \boldsymbol{g}_i \end{bmatrix} - \begin{bmatrix} \boldsymbol{\theta}_i \\ V_i \end{bmatrix}$	$\boldsymbol{\theta}_i : (H_p n_i + H_u q_i) \times 1$, $V_i : H_p n_i \times 1$	$O(2H_p n_i + H_u q_i)$
4. Calculate $\begin{bmatrix} X_i \\ \Delta U_i \\ \boldsymbol{\lambda}_i \end{bmatrix} = \boldsymbol{W}_i^{-1} \boldsymbol{\beta}_i$	$\boldsymbol{W}_i : (2H_p n_i + H_u q_i)^2$, $\boldsymbol{\beta}_i : (2H_p n_i + H_u q_i) \times 1$	$O\left((2H_p n_i + H_u q_i)^2 \right)$

addition is performed. Therefore, T_i is:

$$\begin{aligned} T_i =& O \left((H_p n_i) \sum_{j \neq i} (H_p n_j + H_u q_j) + (H_p n)(H_p n_i + H_u q_i) + (2H_p n_i + H_u q_i)^2 \right) \\ =& O \left(H_p^2 (n_i n - n_i^2) + H_p H_u (n_i q - n_i q_i) + H_p^2 n_i n + H_p H_u q_i n + 4H_p^2 n_i^2 + \right. \\ & \left. 4H_p H_u n_i q_i + H_u^2 q_i^2 \right) \end{aligned} \tag{3.54}$$

Since it is assumed $q_i \leq n_i$, $q \leq n$ and $H_u \leq H_p$:

$$\begin{aligned} 0 \leq H_p^2 (n_i n - n_i^2) \leq & H_p^2 n_i n, \\ 0 \leq H_p H_u (n_i q - n_i q_i) \leq H_p H_u n_i q \leq H_p H_u n_i n \leq & H_p^2 n_i n, \\ 0 \leq H_p H_u q_i n \leq H_p H_u n_i n \leq & H_p^2 n_i n, \\ 0 \leq H_u^2 q_i^2 \leq H_p H_u n_i q_i \leq & H_p^2 n_i^2. \end{aligned}$$

According to **Property B.0.4**, the terms in the O-notation representing T_i can be reduced to:

$$T_i = O(H_p^2 n_i n + 4H_p^2 n_i^2),$$

but $H_p^2 n_i^2 \leq H_p^2 n_i n$, so:

$$T_i = O(H_p^2 n_i n + 4H_p^2 n_i n) = O(5H_p^2 n_i n).$$

The constant is omitted according to **Property B.0.3**, and the complexity of the analytically solved i^{th} local MPC is:

$$T_{i_{anl}} = O(n_i n). \tag{3.55}$$

Note that equation (3.51) has redundant computation for solving the i^{th} local MPC, since only ΔU_i is sent to coordinator for coordination. Also, the implementation of MPC control needs only the first move on the control horizon, for which knowing ΔU_i would be adequate. Indeed, ΔU_i can be solved by the following equation:

$$\begin{aligned}\Delta U_{i(s)}^{*} = &-\boldsymbol{R}_i^{-1}\boldsymbol{G}_{B_{ii}}^T\boldsymbol{\Psi}_i^{-1}\sum_{i\neq j}\left(\boldsymbol{G}_{B_{ij}}\Delta U_j^{(s)} + \boldsymbol{G}_{A_{ij}}X_j^{(s)}\right) + \\ &\boldsymbol{R}_i^{-1}(\boldsymbol{G}_{B_{ii}}^T\boldsymbol{\Psi}_i^{-1}\boldsymbol{\Omega}_i - \sum_{i\neq j}\boldsymbol{G}_{B_{ij}}^T)\boldsymbol{p}^{(s)} + \boldsymbol{R}_i^{-1}\boldsymbol{G}_{B_{ii}}^T\boldsymbol{\Psi}_i^{-1}(\boldsymbol{g}_i - \boldsymbol{G}_{A_{ii}}\boldsymbol{r}_i),\end{aligned} \tag{3.56}$$

where :

$$\boldsymbol{\Psi}_i = \boldsymbol{G}_{A_{ii}}\boldsymbol{Q}_i^{-1}\boldsymbol{G}_{A_{ii}}^T + \boldsymbol{G}_{B_{ii}}\boldsymbol{R}_i^{-1}\boldsymbol{G}_{B_{ii}}^T, \tag{3.57}$$

$$\boldsymbol{\Omega}_i = \boldsymbol{G}_{A_{ii}}\boldsymbol{Q}_i^{-1}\sum_{i\neq j}\boldsymbol{G}_{A_{ij}}^T + \boldsymbol{G}_{B_{ii}}\boldsymbol{R}_i^{-1}\sum_{i\neq j}\boldsymbol{G}_{B_{ij}}^T. \tag{3.58}$$

The dimensions of coefficient matrices in equation (3.56) are smaller than $\boldsymbol{W}_i$; however, the complexity analysis would be the same as what have been presented, and would yield the same result of $T_{i_{anl}} = O(n_i n)$.

Local MPC: Solved Numerically

In real plants, most MPC problems cannot be solved analytically due to the presence of constraints, and complexity analysis for these situations depends upon the particular algorithms chosen to solve the constrained optimization problem. To remove this additional consideration, we have limited our investigations to the unconstrained case and expect that our work will provide some insight into practice.

The i^{th} MPC problem (3.1) is essentially a QP. Among the alternative solution methods for QP, the interior point method (IPM) is often used based on its proven polynomial complexity and efficiency in application. In this work it is assumed that the local MPC is numerically solved using IPM. The results of IPM complexity analysis usually vary with specific assumptions and different algorithms, but the purpose of this discussion is to provide general insight into complexity rather than give highly accurate results. For this reason, we will refer to the results from the literature.

As a numerical method, IPM involves iteration and would finally reach an ε-optimal solution, where ε is a pre-defined accuracy threshold. Specifically, if an IPM is to solve the following QP problem:

$$\begin{aligned} &\min_{\boldsymbol{z}} \frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \boldsymbol{h}^T\boldsymbol{z} \\ &\text{subject to: } \boldsymbol{F}\boldsymbol{z} = \boldsymbol{f}, \\ &\qquad\qquad \boldsymbol{z} \geq \boldsymbol{0}, \end{aligned} \tag{3.59}$$

where: $\boldsymbol{F} \in \mathbb{R}^{l\times m}$ with a full row rank $l \leq m$; $\boldsymbol{H} \in \mathbb{R}^{m\times m}$ is a positive semidefinite matrix; and $\boldsymbol{z} \in \mathbb{R}^m$. The dual problem of (3.59), is:

$$\begin{aligned} &\max_{\boldsymbol{y},\boldsymbol{s}} -\frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \boldsymbol{f}^T\boldsymbol{y} \\ &\text{subject to: } \boldsymbol{F}^T\boldsymbol{y} + \boldsymbol{s} - \boldsymbol{H}\boldsymbol{z} = \boldsymbol{h}, \\ &\qquad\qquad \boldsymbol{s} \geq \boldsymbol{0}, \end{aligned} \tag{3.60}$$

where $\boldsymbol{s} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^l$. The first-order optimality conditions of the primal-dual problem is:

$$\begin{aligned} \boldsymbol{F}\boldsymbol{z} &= \boldsymbol{f}, \\ \boldsymbol{F}^T\boldsymbol{y} + \boldsymbol{s} - \boldsymbol{H}\boldsymbol{z} &= \boldsymbol{h}, \\ z_i s_i &= 0, i = 1, ..., m, \\ \boldsymbol{z}, \boldsymbol{s} &\geq \boldsymbol{0}. \end{aligned} \tag{3.61}$$

The interior-point method would perturb the third equation in (3.61), called the complimentary condition, with a parameter μ, and we would rewrite (3.61) as:

$$\begin{aligned} \boldsymbol{F}\boldsymbol{z} &= \boldsymbol{f}, \\ \boldsymbol{F}^T\boldsymbol{y} + \boldsymbol{s} - \boldsymbol{H}\boldsymbol{z} &= \boldsymbol{h}, \\ z_i s_i &= \mu, i = 1, ..., m, \\ \boldsymbol{z}, \boldsymbol{s} &\geq \boldsymbol{0}. \end{aligned} \tag{3.62}$$

Then an IPM iteratively drives μ in the linear system (3.62) to zero, as the number of iteration grows. In each iteration, Newton's method is used to solve for the Newton direction in $2m + l$ dimensions. The complexity of IPM is therefore composed of two parts: 1) the number of iterations required, and 2) the complexity of Newton's method within each iteration. It is stated in (Gondzio, 2012) that the best known IPM for linear and quadratic programming has a proven upper bound to find the ε-optimal solution in $O(\sqrt{m}\ln(1/\varepsilon))$ iterations. In our case, Newton's method is used for solving a linear system, which requires

$O((2m+l)^3)$ operation (Cheng, 2007; Nash and Sofer, 1996). With $l \leq m$, the complexity of Newton's method is $O(m^3)$. Therefore, the overall complexity for an IPM to solve the QP (3.59) is $O(m^{3.5}\ln(1/\varepsilon))$

The i^{th} local MPC problem is transformed into the general form (3.59), which would give $m = H_p n_i + H_u q_i$, $l = H_p n_i$. Thus, the process of actually solving the QP requires $O((H_p n_i + H_u q_i)^{3.5}\ln(1/\varepsilon))$ operations. With the assumption of $q_i \leq n_i$, the IPM procedure has a complexity of $O\left(n_i^{3.5}\ln(1/\varepsilon)\right)$.

Prior to actually solving the problem (3.1) using IPM, the updating steps 1 to 3 in Table 3.1 still need to be performed. The complexity of the three steps added together is:

$$\begin{aligned} T_{i_{up}} =& O\left((H_p n_i)\sum_{j\neq i}(H_p n_j + H_u q_j) + (H_p n)(H_p n_i + H_u q_i) + (2H_p n_i + H_u q_i)\right) \\ =& O\left(H_p^2(n_i n - n_i^2) + H_p H_u(n_i q - n_i q_i) + H_p^2 n_i n + H_p H_u q_i n + 2H_p n_i + H_u q_i\right) \\ =& O\left(\max\left(H_p^2(n_i n - n_i^2), H_p H_u(n_i q - n_i q_i), H_p^2 n_i n, H_p H_u q_i n, 2H_p n_i, H_u q_i\right)\right) \\ =& O(Hp^2 n_i n) \\ =& O(n_i n). \end{aligned} \tag{3.63}$$

Therefore, the complexity of numerically solving the i^{th} local MPC in total is:

$$\begin{aligned} T_{i_{num}} =& T_{i_{up}} + T_{i_{IPM}} \\ =& O\left(n_i n + n_i^{3.5}\ln(1/\varepsilon)\right). \end{aligned} \tag{3.64}$$

In (3.64), if $\frac{n}{n_i} < \infty$, $T_{i_{IPM}}$ will finally exceed $T_{i_{up}}$, i.e., $O(n_i^{3.5}\ln(1/\varepsilon)) > O(n_i n)$ when $n_i, n \to \infty$. This can be applied to any real plant, so we conclude that the complexity of the i^{th} local MPC, if it is solved numerically, is:

$$T_{i_{num}} = O\left(n_i^{3.5}\ln(1/\varepsilon)\right). \tag{3.65}$$

Coordinator

Equations (3.9) and (3.11) are the computations carried out in the coordinator. It can be seen that the coordinator's computation involves only matrix multiplication and matrix addition, whether the local MPCs use analytical or numerical methods. For ease of analysis, we rewrite these two equations into the form:

$$X^{(s)} = \boldsymbol{\gamma}_1 - \boldsymbol{\Gamma}_1 \Delta U^{(s)}, \tag{3.66}$$

$$\boldsymbol{p}^{(s)} = \boldsymbol{\gamma}_2 - \boldsymbol{\Gamma}_2 X^{(s)} - \boldsymbol{\Gamma}_3 \Delta U^{(s)}, \tag{3.67}$$

where

$$
\begin{aligned}
\boldsymbol{\gamma}_1 &= \boldsymbol{G}_A^{-1}\boldsymbol{g},\\
\boldsymbol{\gamma}_2 &= (\boldsymbol{G}\boldsymbol{G}^T)^{-1}\boldsymbol{G}_A\boldsymbol{Q}\boldsymbol{r},\\
\boldsymbol{\Gamma}_1 &= \boldsymbol{G}_A^{-1}\boldsymbol{G}_B,\\
\boldsymbol{\Gamma}_2 &= (\boldsymbol{G}\boldsymbol{G}^T)^{-1}\boldsymbol{G}_A\boldsymbol{Q},\\
\boldsymbol{\Gamma}_3 &= (\boldsymbol{G}\boldsymbol{G}^T)^{-1}\boldsymbol{G}_B\boldsymbol{R}.
\end{aligned}
$$

In the above equations, $\boldsymbol{\gamma}_1$, $\boldsymbol{\gamma}_2$, $\boldsymbol{\Gamma}_1$, $\boldsymbol{\Gamma}_2$ and $\boldsymbol{\Gamma}_3$ are all constant matrices (vectors), which could be computed and stored beforehand. As in the analysis of local MPCs, the complexity of each computation step and the dimensions of related parameters of equations (3.66) and (3.67) are listed in Table 3.2.

Table 3.2: Coordinator Complexity in Prediction-driven CDMPC: Step by Step

Step Description	**Parameters and Dimensions**	**Step Complexity**
1. Calculate $X^{(s)} = \boldsymbol{\gamma}_1 - \boldsymbol{\Gamma}_1\Delta U^{(s)}$	$\boldsymbol{\gamma}_1 : (H_pn)\times 1$, $\boldsymbol{\Gamma}_1 : H_pn \times H_uq$, $\Delta U^{(s)} : H_uq \times 1$	$O(H_pH_unq + H_pn)$
2. Calculate $\boldsymbol{p}^{(s)} = \boldsymbol{\gamma}_2 - \boldsymbol{\Gamma}_2 X^{(s)} - \boldsymbol{\Gamma}_3\Delta U^{(s)}$	$\boldsymbol{\gamma}_2 : H_pn\times 1$, $\boldsymbol{\Gamma}_2 : (H_pn)^2$, $X^{(s)} : H_pn \times 1$, $\boldsymbol{\Gamma}_3 : H_pn \times H_uq$, $\Delta U^{(s)} : H_uq \times 1$	$O\left((H_pn)^2 + H_pH_unq + H_pn\right)$

From Table 3.2, the complexity of coordinator can be derived as follows:

$$
\begin{aligned}
T_{coor} =& O(H_pH_unq + H_pn + H_pn^2 + H_pH_unq + H_pH_unq)\\
=& O\left(\max\left(H_pH_unq, H_pn, H_p^2n^2, H_pH_unq, H_pH_unq\right)\right)\\
=& O(H_p^2n^2)\\
=& O(n^2).
\end{aligned}
\tag{3.68}
$$

Communication Cycle Number

The number of required communication cycles, or CCN, of prediction-driven CDMPC algorithm is closely related to the algorithm's rate of convergence, which was discussed in §3.2.3. Although we are not able to obtain an upper bound on CCN, it is possible to give estimations of the number. When deriving an estimation of CCN, the condition $\rho(\mathbf{C_2}) < 1$ is always assumed.

From (3.43), there is:

$$\|\boldsymbol{e}^{(s)}\| = \|\mathbf{C_2}^s \boldsymbol{e}^{(0)}\| \leq \|\mathbf{C_2}^s\| \|\boldsymbol{e}^{(0)}\|.$$

That is:

$$\frac{\|\boldsymbol{e}^{(s)}\|}{\|\boldsymbol{e}^{(0)}\|} \leq \|\mathbf{C_2}^s\|. \tag{3.69}$$

If the algorithm is to reduce the initial error by a factor σ, i.e., it is required that:

$$\frac{\|\boldsymbol{e}^{(CCN)}\|}{\|\boldsymbol{e}^{(0)}\|} \leq \sigma, \tag{3.70}$$

the requirement (3.70) can be satisfied when:

$$\|\mathbf{C_2}^{CCN}\| \leq \sigma, \text{ or } CCN \geq -\log\sigma / R_{CCN}(\mathbf{C_2}), \tag{3.71}$$

where $R_{CCN}(\mathbf{C_2}) = -\frac{1}{CCN}\log\|\mathbf{C_2}^{CCN}\|$ is defined in **Definition 3.2.1**. Note that the inequality (3.71) is nonlinear in CCN and there is no simple way to extract CCN from $\|\mathbf{C_2}^{CCN}\|$, but if the asymptotic convergence rate is used to approximate the average rate, the estimated value of CCN can be obtained, as:

$$\widehat{CCN}_1 \approx -\log\sigma / R(\mathbf{C_2}) = \log\sigma / \log\rho(\mathbf{C_2}). \tag{3.72}$$

The estimation (3.72) may present difficulties in application, because it is hard to determine whether or not $\frac{\|\boldsymbol{e}^{(s)}\|}{\|\boldsymbol{e}^{(0)}\|} \leq \sigma$ is satisfied. If (3.70) is used as stopping criteria, $\Delta U^{(\infty)}$ is required first and numerically solving for $\Delta U^{(\infty)}$ is moot; however, estimation (3.72) can be linked to the stopping criterion $\|\Delta U^{(CCN+1)} - \Delta U^{(CCN)}\| < \epsilon$ discussed in §3.1.2. The equation (3.37) gives that:

$$\Delta U^{(s+1)} - \Delta U^{(s)} = \mathbf{C_2}^s(\Delta U^1 - \Delta U^0),$$

which indicates that for $\|\Delta U^{(s+1)} - \Delta U^{(s)}\|$ the relationship:

$$\frac{\|\Delta U^{(s+1)} - \Delta U^{(s)}\|}{\|\Delta U^{(1)} - \Delta U^{(0)}\|} \leq \|\mathbf{C_2}^s\|$$

also exists. Then there is:

$$\|\Delta U^{(s+1)} - \Delta U^{(s)}\| \leq \|\mathbf{C_2}^s\| \|\Delta U^{(1)} - \Delta U^{(0)}\|. \tag{3.73}$$

Therefore, $\|\mathbf{C_2}^s\| \leq \sigma$ would guarantee $\|\Delta U^{(s+1)} - \Delta U^{(s)}\| \leq \sigma\|\Delta U^{(1)} - \Delta U^{(0)}\|$. If we denote $\delta = \|\Delta U^{(1)} - \Delta U^{(0)}\|$ and $\sigma = \epsilon/\delta$, then the algorithm terminates once:

$$CCN \geq -(\log\epsilon/\delta)/R_{CCN}(\mathbf{C_2}). \tag{3.74}$$

Again we use $R(\mathbf{C_2})$ to approximate $R_{CCN}(\mathbf{C_2})$ and get:

$$\widehat{CCN}_2 \approx \frac{-(\log \epsilon - \log \delta)}{R(\mathbf{C_2})} = \frac{\log \epsilon - \log \delta}{\log \rho(\mathbf{C_2})}. \tag{3.75}$$

From here, (3.75) is the formula that will be used for estimating CCN.

As stated in **Remark 3.2.2**, $\|A^k\|^{1/k} \geq \rho(A)$ when $k < \infty$, which means $\log \sigma / R_s(\mathbf{C_2}) \geq \log \varepsilon / R(\mathbf{C_2})$. Therefore, the estimation in (3.72) and (3.75) are not upper bounds on communication cycle numbers. Figure 3.1 shows a comparison of estimated versus real CCN taken from a set of numerical experiments. It is clear that the estimated CCN can be smaller or larger than the real value in different cases. For this reason, CCN cannot be presented in O-notation.

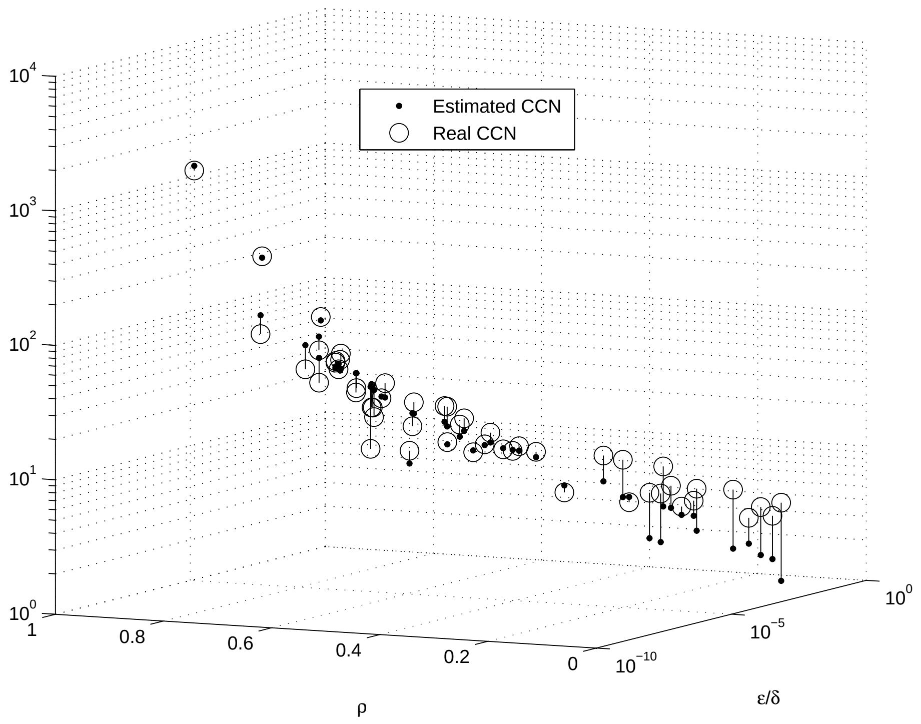


Figure 3.1: Estimated versus real communication cycle numbers. The result is obtained from 51 randomly generated system. x axis and y axis are $\sigma = \epsilon/\delta$ and $\rho(\mathbf{C_2})$, respectively.

What can also be seen from Figure 3.1 is that for the tested examples, none of the estimated communication cycle numbers deviate more than one order of magnitude from its real value. Therefore, the estimation given in (3.75) is sufficient to capture the main behavior in the communication cycles.

Algorithm Complexity and a Comparison with Centralized Complexity

The above analysis for T_{coor}, $\max_i T_i$ and CCN are summarized in Table 3.3. Since the

Table 3.3: Complexity of Prediction-driven CDMPC and Centralized MPC

Complexity Type	Solving Analytically	Solving Numerically
T_{coor}	$O(n^2)$	$O(n^2)$
$\max_i T_i$	$O(\max(n_i)n)$	$O(\max(n_i)^{3.5}\ln(1/\varepsilon))$
CCN	$\approx (\log\epsilon - \log\delta)/\log\rho(\mathbf{C_2})$	
T_{CEN}	$O(n^2)$	$O(n^{3.5}\ln(1/\varepsilon))$

upper bound of communication cycle numbers is unavailable, the conclusion of the overall complexity cannot be written in the O-notation and as a result, is an estimation as well:

$$\begin{aligned} T_{PRED} &= (T_{coor} + \max_i T_i) \times CCN \\ &= \left(O(n^2) + \max_i O(n_i n)\right) \times CCN \\ &\approx \begin{cases} \left(O(n^2 + \max(n_i)n)\right) \times \frac{\log\epsilon - \log\delta}{\log\rho(\mathbf{C_2})}, \text{MPC solved analytically,} \\ \left(O(n^2 + \max(n_i)^{3.5}\ln(1/\varepsilon))\right) \times \frac{\log\epsilon-\log\delta}{\log\rho(\mathbf{C_2})}, \text{MPC solved numerically.} \end{cases} \end{aligned} \tag{3.76}$$

Other than the complexity of the prediction-driven CDMPC algorithm, we are also interested in its comparison with the complexity of centralized MPC. Since a parallel computating environment is assumed, the DMPCs are actually distributing the computation load to more processing nodes, with the inter-unit communications as the trade-off. We would like to know whether or not this trade-off is acceptable. For this purpose, the complexity of centralized MPC, T_{CEN}, is also listed in Table 3.3. This complexity can be obtained by replacing $\max(n_i)$ with n in the maximum local MPC complexity $\max_i T_i$.

When the MPCs are solved analytically, T_{CEN} and T_{coor} each has a complexity of $O(n^2)$. This means that their complexity is on a same level. By counting the arithmetic operations in the coordinator and the centralized MPC respectively, we found that they have the ratio $r = \frac{T_{coor}}{T_{CEN}} = \frac{H_p^2 n^2 + 2H_pH_u nq}{(H_p^2+1)n^2 + (H_pH_u+1)nq}$, as $n, q \to \infty$. The value of the ratio depends on the actual values of H_p, H_u, n and q. $r < 1$ requires $q < \frac{n}{H_pH_u - 1}$, which normally means that $q \ll n$. Note that in this comparison, the maximum subsystem and communication cycles are not included, making it more difficult for the prediction-driven CDMPC algorithm to be more efficient that the centralized MPC.

In the case that the MPCs are solved numerically, T_{CEN} becomes $O(n^{3.5}\ln(1/\varepsilon))$ while T_{coor} remains to be $O(n^2)$. The prediction-driven CDMPC algorithm in a single communication cycle is more efficient than the centralized MPC as $n \to \infty$. To see this, assume that the maximum subsystem size $\max(n_i) = n - 1$, which is the largest subsystem size we can have. Despite this, T_{coor} is still smaller than the difference between T_{CEN} and $\max_i(T_i)$: the function $f(x) = n^x - (n-1)^x$ is monotonically increasing when $n > 1$ and $n^3 - (n-1)^3 = O(n^2)$, so $n^{3.5} - (n-1)^{3.5} = O(n^q)$, where $q > 2$; therefore, the difference between $T_{CEN} = O(n^{3.5}\ln(1/\varepsilon))$ and $\max_i(T_i) = O(\max(n_i)^{3.5}\ln(1/\varepsilon))$ is at least $O(n^q)$, which is larger than T_{coor} when $n \to \infty$. If the communication cycle number is small, the prediction-driven CDMPC algorithm will be more computationally favorable than the centralized MPC.

3.3.2 Empirical Analysis

Several numerical experiments were designed and conducted to study the properties of empirical complexity of prediction-driven CDMPC. The theoretical complexity results listed in Table 3.3 and how the parameters such as the plant state size n, subsystem state size n_i, etc. will affect the algorithm's complexity in practice, were investigated in this study. Each experiment was done with a set of Monte Carlo simulations, where a class of plants and MPC controllers were generated randomly. The details of how the system and MPC parameters are generated can be found in Appendix C.1.

As stated in §3.3.1, computational complexity of an algorithm can be measured by the execution time of the algorithm. Therefore, the prediction-driven CDMPC algorithm's execution time t_{PRED} is used to represent its empirical complexity. Specifically, we record the communication cycle number CCN, the accumulated execution time of coordinator $\sum t_{coor}$ and the accumulated execution time of the maximum-sized subsystem $\sum t_{i_{max}}$ for the complexity analysis' three component parts, as in Table 3.3. T_{coor}, $\max_i T_i$ and T_{PRED} can then be represented by t_{coor}, $t_{i_{max}}$ and t_{PRED}, which are calculated as follows:

$$t_{coor} = \sum t_{coor}/CCN \tag{3.77}$$

$$t_{i_{max}} = \sum t_{i_{max}}/CCN \tag{3.78}$$

$$t_{NonCo} = t_{coor} + t_{i_{max}} \tag{3.79}$$

$$t_{PRED} = \sum t_{coor} + \sum t_{i_{max}} \tag{3.80}$$

The experiments were all done using the MATLAB® 2011b platform, with an Intel® Core™i5-2500K machine with 3.3 GHz processor speed and 8 GB of RAM memory. The experiments inherited all the assumptions given in Chapter 1, as well as those made before the theoretical analysis in §3.3.1. It is further assumed in all the experiments that:

- $\boldsymbol{A}$ is stable;

- $\boldsymbol{B}_{ij}$ are zero matrices;
- $\boldsymbol{Q}$ and $\boldsymbol{R}$ are both diagonal matrices.

The major objective of the empirical study is to verify the theoretical complexity results listed in Table 3.3 through the numerical experiments. Note that T_{coor} is only determined by n, the number of states in the plant. $\max_i T_i$, if the local MPCs are solved analytically, is dependent on n and $\max(n_i)$, which is the largest state number among all subsystems. If local MPCs are solved numerically, $\max_i T_i$ is determined by $\max(n_i)$ and the pre-defined accuracy threshold ε. Experiment 1 and Experiment 2 are therefore designed to identify the relationships between plant/largest subsystem size and the empirical complexities. For both experiments, there will be one scenario that solves the MPCs analytically and another numerically.

The communication cycle number is a more complicated case. Although an estimation of CCN is already given, the variables in the estimation, initial error δ, and the spectral radius ρ, need to be calculated via complex matrix operations. Therefore, it is almost impossible to derive an analytical size-to-CCN mapping. It is expected that CCN will increase as the plant/subsystem size increases, because the difficulty of coordination in these cases will increase as well. Since the smallest stopping threshold ϵ for the prediction-driven CDMPC is 10^{-5}, while the tolerance for MATLAB® IPM solver is 10^{-8}, the optimal $\Delta U^*_{i(s)num}$ can be considered the same as $\Delta U^*_{i(s)anl}$. Therefore, whether or not MPCs are solved numerically will not change the required number of communication cycles, and the discussion of CCN in the two scenarios will be combined, for both Experiment 1 and Experiment 2.

Experiment 1: Empirical Complexity Varying With Number of Subsystems

In Experiment 1, all subsystems were fixed to have 2 states and 2 inputs, i.e., $\max(n_i) = 2$. The number of subsystems $N \in \mathbb{N}$, where

$$\begin{aligned}\mathbb{N} = \{&2, 4, 6, 8, 10, 15, 20, 25, 30, 35, 40, 50, 60, 70 \\ &80, 90, 100, 110, 120, 130, 140, 150, 160\}\,.\end{aligned} \tag{3.81}$$

The number of states in the plant is therefore $n = 2N$ and would vary accordingly to N. The Monte Carlo experiment is repeated 100 times for each element in set $\mathbb{N}$.

Scenario 1: In this scenario, the results are obtained when the MPC problems are solved analytically. As can be seen from Figure 3.2, t_{coor} and t_{CEN} have a similar growth rate, with t_{coor} always larger than t_{CEN}. This result shows that even in a single communication cycle, coordination is less efficient than the centralized MPC, if the MPC is solved analytically. Then, the prediction-driven CDMPC algorithm will take a longer total execution time than the centralized MPC. We also did least-squares fitting for the coordinator and the centralized MPC to determine whether they present second-order characteristics as predicted. If the curves are assumed to be second-order polynomials, the fitting results are:

$$\begin{aligned} t_{coor} =& (4.3407 \pm 0.81780) \times 10^{-9} n^2 + \\ & (4.8260 \pm 2.5379) \times 10^{-7} n + (5.0758 \pm 1.4234) \times 10^{-5}, \end{aligned} \tag{3.82}$$

$$\begin{aligned} t_{CEN} =& (3.5080 \pm 0.53005) \times 10^{-9} n^2 + \\ & (0.99160 \pm 1.6449) \times 10^{-7} n + (9.8237 \pm 9.2256) \times 10^{-6}, \end{aligned} \tag{3.83}$$

where the number after '$\pm$' shows the 95% confidence interval of the fitted parameter. If they are assumed to be third-order polynomials, the fitting results are:

$$\begin{aligned} t_{coor} =& (-1.6682 \pm 0.61989) \times 10^{-11} n^3 + (1.2158 \pm 0.29500) \times 10^{-9} n^2 + \\ & (-4.4644 \pm 3.8045) \times 10^{-7} n + (6.9631 \pm 1.1384) \times 10^{-5} \end{aligned} \tag{3.84}$$

$$\begin{aligned} t_{CEN} =& (-2.7902 \pm 6.4267) \times 10^{-12} n^3 + (4.8154 \pm 3.0584) \times 10^{-9} n^2 + \\ & (-0.56226 \pm 3.9443) \times 10^{-7} n + (1.2980 \pm 1.1803) \times 10^{-5}. \end{aligned} \tag{3.85}$$

For the centralized MPC execution time, the parameters of the third-order terms contains 0 in its 95% confidence interval, meaning that we cannot claim with 95% confidence that t_{CEN} is a third-order polynomial. For the coordinator execution time, though the fitted third-order coefficient does not contain 0 in the 95% confidence interval, it is negative. With this fitting, $t_{coor} \to -\infty$ as $n \to \infty$, which is clearly invalid. Therefore, both t_{coor} and t_{CEN} should be considered statistically second-order polynomials, which matches the hypothesis for them.

Figure 3.3 shows the longest execution time taken by the subsystems in one cycle. The hypothesis is that $T_{i_{anl}} = O(n_i n)$, so when $\max(n_i)$ is fixed, the execution time of subsystem is expected to grow with the plant size n, as can be seen in Figure 3.3. A least-squares fitting is also done for this data set, which, according to the hypothesis, should be linear in n. The result of the fitting, if assuming $t_{i_{max}}$ to be linear, quadratic and cubic function respectively, are presented as follows:

$$t_{i_{max}} = (6.1443 \pm 0.32662) \times 10^{-7} n + (3.9015 \pm 0.39015) \times 10^{-5}, \tag{3.86}$$

$$\begin{aligned} t_{i_{max}} =& (7.7160 \pm 1.5772) \times 10^{-10} n^2 + \\ & (3.6127 \pm 0.53162) \times 10^{-7} n + (5.2617 \pm 0.35303) \times 10^{-5}, \end{aligned} \tag{3.87}$$

$$\begin{aligned} t_{i_{max}} =& (-1.6181 \pm 1.8730) \times 10^{-12} n^3 + (1.5887 \pm 0.95735) \times 10^{-9} n^2 + \\ & (2.4848 \pm 1.3976) \times 10^{-7} n + (5.6206 \pm 0.53126) \times 10^{-5} \end{aligned} \tag{3.88}$$

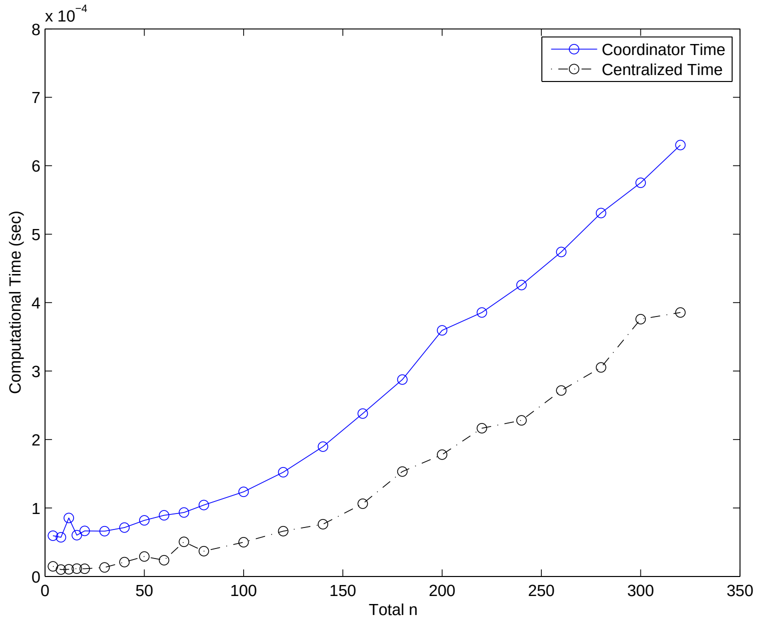


Figure 3.2: Prediction-driven CDMPC Experiment 1. Execution time of coordinator in one cycle versus execution time of centralized MPC, when MPCs are solved analytically. The solid line represents t_{coor} and the dash-dot line represents t_{CEN}. x axis is the plant state size and y axis is the execution time in seconds.

From equations (3.86) to (3.88), it is clear that $t_{i_{max}}$ is not statistically a third-order polynomial in n because the third-order coefficient contains 0 within its 95% confidence interval. Equation (3.87) indicates that empirically $t_{i_{max}}$ may be a second-order polynomial in n when MPCs are solved analytically.

Scenario 2: In this scenario, the MPC problems are solved numerically. Figure 3.4 shows the time taken by coordinator in one cycle and centralized MPC, respectively. It can be seen that, when the centralized MPC is solved by the interior-point method, $t_{coor} < t_{CEN}$. This is not surprising because our analysis shows that $T_{CEN_{num}} = O(n^{3.5} \ln(1/\varepsilon))$ and $T_{coor} = O(n^2)$.

The execution time of subsystems is plotted against the plant size in Figure 3.5. From equation (3.64) it can be seen that if n_i is fixed, $T_{i_{num}}$ will also have a complexity of $O(n_i n)$, indicating $t_{i_{max}}$ should be a linear function in n. As can be seen from the result, $t_{i_{max}}$ is increasing with n, although it appears to be sublinear. A possible reason for this mismatch with theory is due to the iteration feature of numerical methods. Each time when the optimization problem is solved, the computer needs to initialize a process such as read from memory and allocate new variables. The cost of initialization process is unknown, and is not included within the theoretical analysis. These are expected to influence the execution

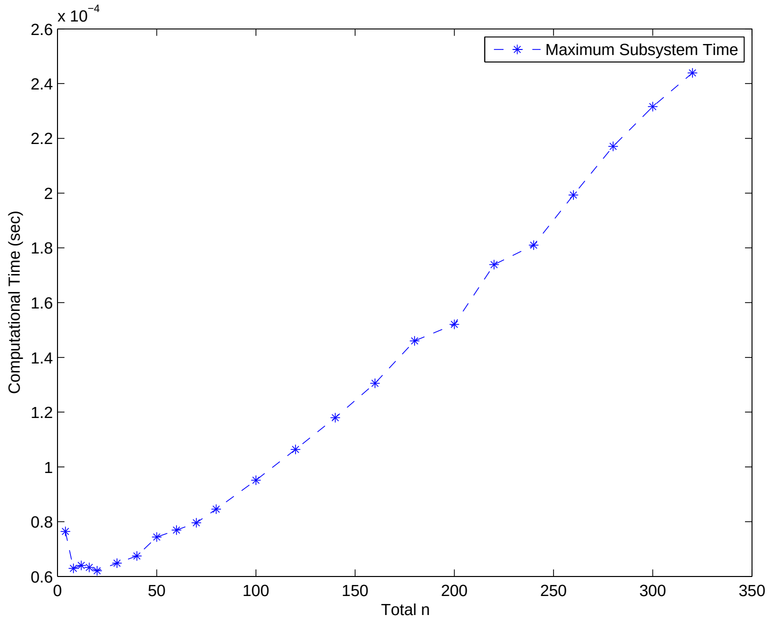


Figure 3.3: Prediction-driven CDMPC Experiment 1. The longest execution time of subsystems, when MPCs are solved analytically. x axis is the plant state size and y axis is the execution time in seconds.

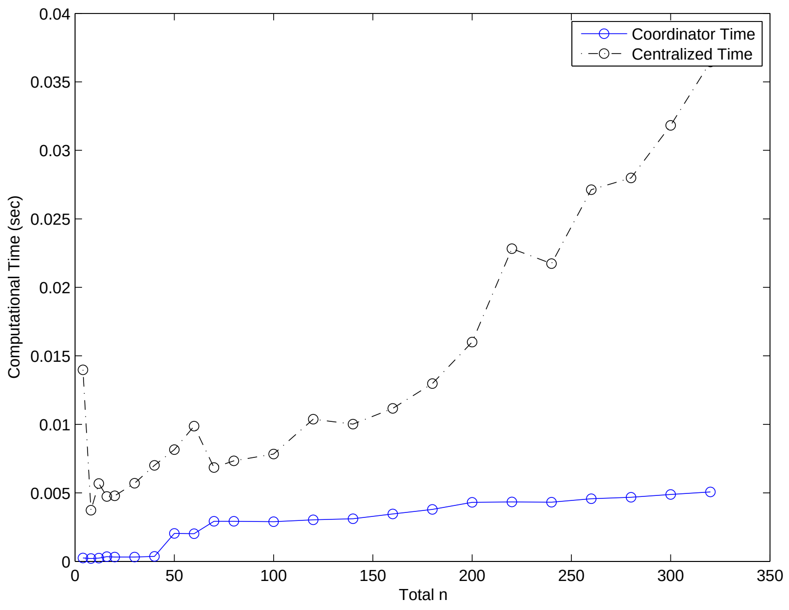


Figure 3.4: Prediction-driven CDMPC Experiment 1. Execution time of coordinator in one cycle versus execution time of centralized MPC, when MPC problems are solved numerically using the interior-point method. The solid line represents t_{coor} and the dash-dot line represents t_{CEN}. x axis is the plant state size and y axis is the execution time in seconds.

time more when the problem size is small.

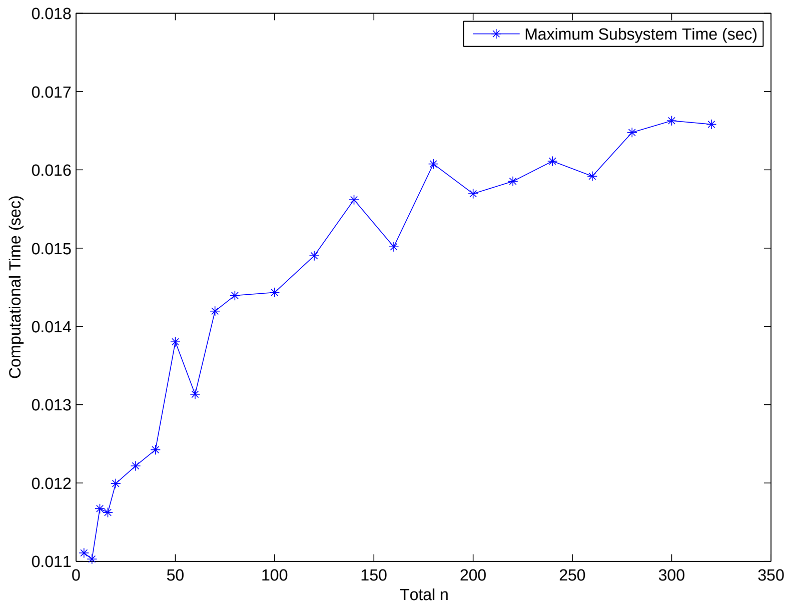


Figure 3.5: Prediction-driven CDMPC Experiment 1. Longest execution time of subsystems, when MPC problems are solved numerically using the interior-point method. x axis is the plant state size and y axis is the execution time in seconds.

In this scenario, t_{NonCo} is calculated according to equation (3.79) and compared with t_{CEN}, as is shown in Figure 3.6. When n is small, t_{NonCo} is larger than t_{CEN}, but it will cross t_{CEN} as the number of subsystem grows. This matches the analysis in §3.3.1 and indicates that if the number of communication cycles does not grow too quickly, the prediction-driven CDMPC is computationally more favorable than the centralized MPC when n is large.

Figure 3.7 shows the median levels of communication cycle numbers and Figure 3.8 shows CCN for all of the systems generated in this experiment. The subplots of Figure 3.8 zoom in on the y-axis successively to show how the distribution of CCN changes as n changes. It can be seen from Figure 3.7 and 3.8 that: (a) CCN tends to grow as the plant size increases; (b) the median level of CCN grows at a reasonable rate; and (c) there is higher probability of encountering a system with a very large CCN, when the plant size is large. As an additional remark to (c), there is still information outside the figure. As can be seen from Appendix C, in the system generation stage of these experiments, only systems with $\rho(\mathbf{C_2}) < 1$ will be considered as candidates for the DMPC algorithm. The difficulty in generating systems that meet this convergence condition increases as n increases.

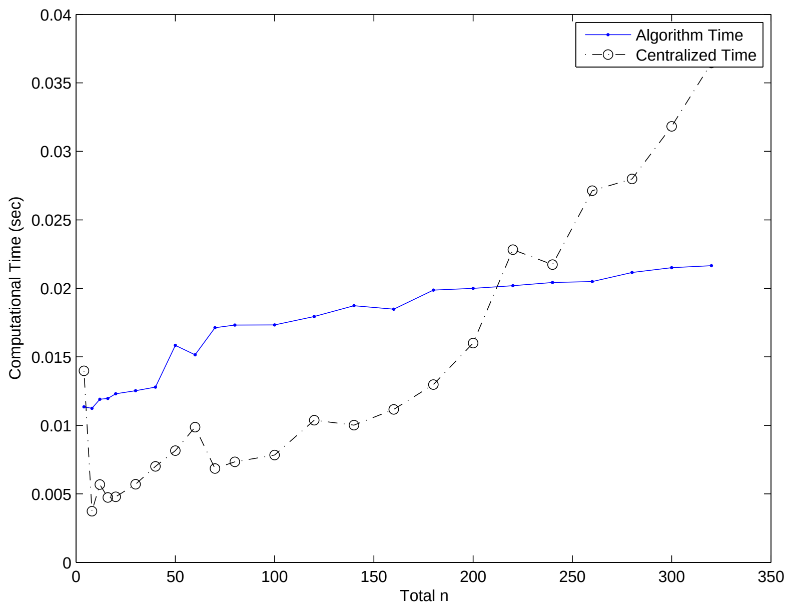


Figure 3.6: Prediction-driven CDMPC Experiment 1. Execution time of the prediction-driven CDMPC in a single cycle versus the execution time of centralized MPC, when MPC problems are solved numerically using the interior-point method. The solid line represents t_{NonCo} and the dash-dot line represents t_{CEN}. x axis is the plant state size and y axis is the execution time in seconds.

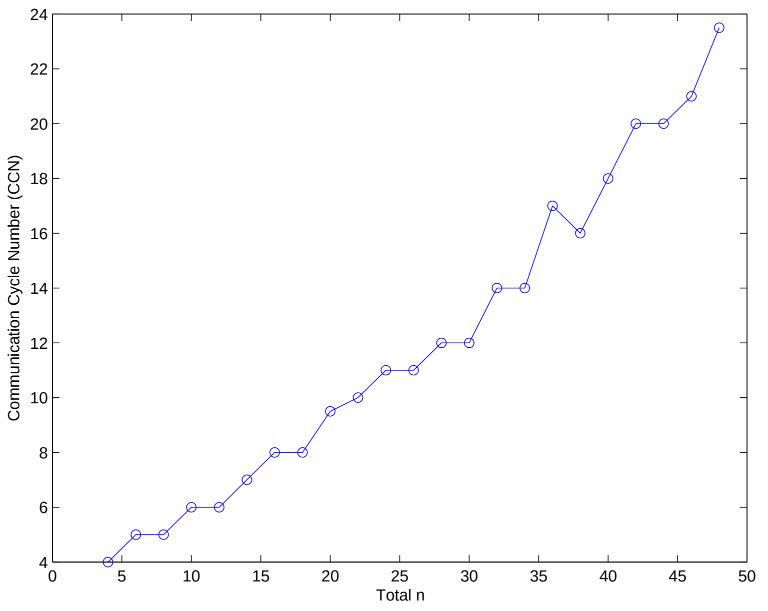


Figure 3.7: Prediction-driven CDMPC Experiment 1. The median of CCN at each experiment mode in Experiment 1, including all data points from Scenario 1 and Scenario 2, where the number of plant states $n \in \mathbb{N}$.

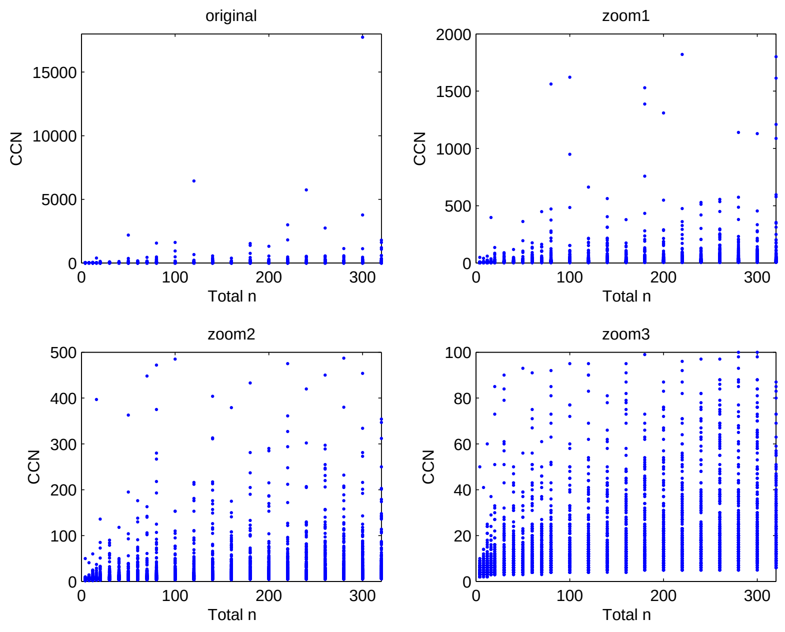


Figure 3.8: Prediction-driven CDMPC Experiment 1. All CCNs for every repetition in Experiment 1, including all data points from Scenario 1 and Scenario 2. The plots are zoomed successively. Top left: the original plot. Top right: the zoomed plot, where only CCN between 0 and 2000 are shown. Bottom left: the zoomed plot, where only CCN between 0 and 500 are shown. Bottom right: the zoomed plot, where only CCN between 0 and 100 are shown.

Experiment 2: Empirical Complexity Varying With the Maximum Subsystem Size

In Experiment 2, the plant sizes were fixed to be $n = q = 160$, i.e., there were a total of 160 states and 160 inputs in the plant. The number of subsystems, N, was also fixed to be 80. For each subsystem i, there is $n_i = q_i$, but the maximum n_i varies in different tests. Specifically, the subsystems' sizes were sequentially chosen from the following set of combinations:

$$\mathbb{NI} = \left\{ \begin{array}{l} \{\underbrace{2,2,2,...,2}_{80\ 2\text{'s}}\}, \{7, \underbrace{2,...,2}_{74\ 2\text{'s}} \underbrace{1,...,1}_{5\ 1\text{'s}}\}, \\ \{12, \underbrace{2,...,2}_{69\ 2\text{'s}} \underbrace{1,...,1}_{10\ 1\text{'s}}\}, \{17, \underbrace{2,...,2}_{64\ 2\text{'s}} \underbrace{1,...,1}_{15\ 1\text{'s}}\}, \\ ..., \{77, 2, 2, 2, 2 \underbrace{1,1,...,1,}_{75\ 1\text{'s}}\} \{81, \underbrace{1,1,...,1}_{79\ 1\text{'s}}\} \end{array} \right\}, \tag{3.89}$$

so the maximum subsystem size would vary from 2 to 81.

Scenario 1: The local MPCs and the centralized MPC were solved analytically in Scenario 1. Figure 3.9 presents the coordinator execution time in a single communication cycle, t_{coor} versus the centralized MPC execution time, t_{CEN}. It can be seen that both curves fluctuate with no obvious trends. The least-squares regression technique is used to fit linear relationships between t_{coor} and $\max(n_i)$, t_{CEN} and $\max(n_i)$, respectively. The resulting expressions are:

$$t_{coor} = (0.32067 \pm 8.0533) \times 10^{-7} \max(n_i) + (4.2955 \pm 0.39076) \times 10^{-4}, \quad (3.90)$$

$$t_{CEN} = (-0.18427 \pm 1.4072) \times 10^{-6} \max(n_i) + (1.9373 \pm 0.68282) \times 10^{-4}. \quad (3.91)$$

The two fitted equations (3.90) and (3.91) each has a very small estimated slope with the 95% confidence interval containing zero. Therefore, we cannot say with 95% percent confidence that the slopes of t_{coor} and t_{CEN} with respect to $\max(n_i)$ are not zero. Given the 'almost zero' estimates, it can be concluded that t_{coor} and t_{CEN} are statistically constant when $\max(n_i)$ varies but n is fixed. These results match the the theoretical complexity analysis for T_{coor} and $T_{CEN_{anl}}$, which states that their complexities are only related to the plant size n.

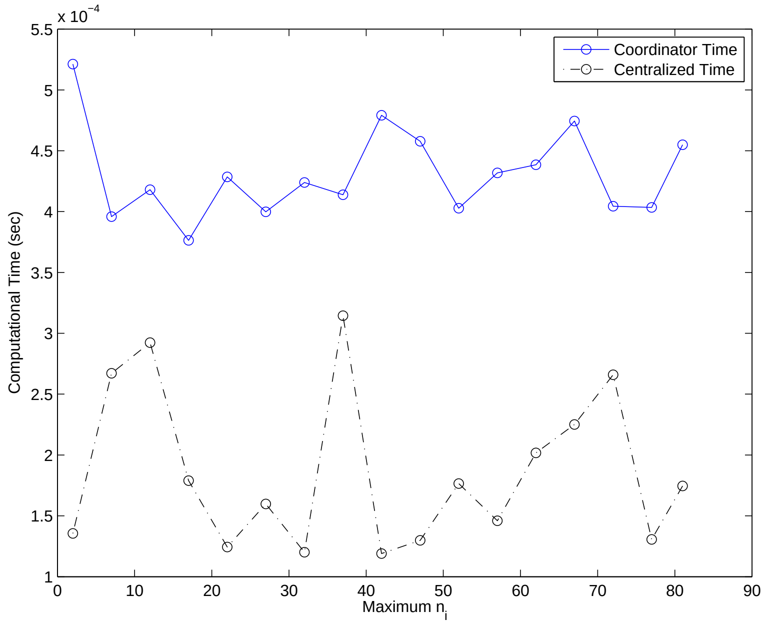


Figure 3.9: Prediction-driven CDMPC Experiment 2. Execution time of coordinator in one cycle versus execution time of centralized MPC, when MPCs are solved analytically. The solid line represents t_{coor} and the dash-dot line represents t_{CEN}. x axis is the maximum subsystem size and y axis is the execution time in seconds.

Figure 3.10 shows how $t_{i_{max}}$ varies when the maximum subsystem size increases. According to §3.3.1, $T_{i_{anl}} = O(n_i n)$, so the expected execution time of the maximum subsystem should be linear in $\max(n_i)$. To test this hypothesis, we assume $t_{i_{max}}$ to be first- and second-order polynomials with respect to $\max(n_i)$ and did the linear regression. The fitting results are:

$$\begin{aligned} t_{i_{max}} &= (2.8926 \pm 0.41814) \times 10^{-6} \max(n_i) + (2.0714 \pm 0.20289) \times 10^{-4}, \\ t_{i_{max}} &= (0.56671 \pm 1.9967) \times 10^{-8} \max(n_i)^2 + \\ &\quad (3.3664 \pm 1.7236) \times 10^{-6} \max(n_i) + (2.0061 \pm 0.31046) \times 10^{-4}, \end{aligned} \tag{3.92}$$

where the second-order coefficient of the quadratic contains 0 in its 95% confidence interval. Thus, $t_{i_{max}}$ is statistically not a quadratic function of $\max(n_i)$ but a linear one, which matches the hypothesis.

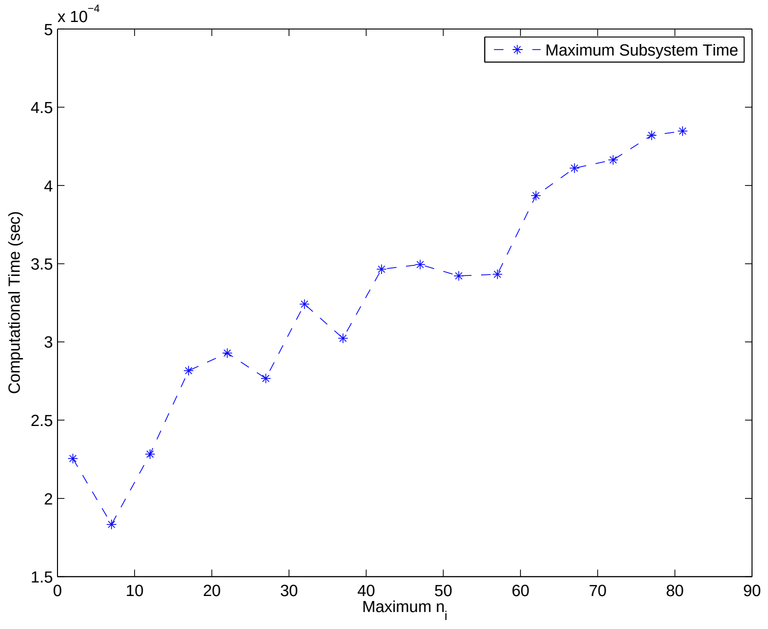


Figure 3.10: Prediction-driven CDMPC Experiment 2. The longest execution time of subsystems, when MPCs are solved analytically. x axis is the maximum subsystem state size and y axis is the execution time in seconds.

Scenario 2: The local and centralized MPCs are all solved numerically in Scenario 2. In Figure 3.11, the the coordinator execution time in one cycle t_{coor} and the centralized MPC excution time t_{CEN} are shown. The trend of t_{coor} is similar to that in Scenario 1, where the data points fluctuate around a certain value; however, this is not the case for t_{CEN}. From §3.3.1 we observed that the theoretical complexity of the centralized MPC is $T_{CEN_{num}} = O(n^3 \ln(1/\varepsilon))$ and expect $t_{CEN_{num}}$ to stay almost constant when n is fixed, but in the empirical test t_{CEN} presents polynomial characteristics and increases with the growth of the maximum subsystem size. This mismatch is possibly due to the 'presolve' step in the MATLAB® IPM solver, which aims to simplify the optimization's constraints and remove redundancies. If a linear equality constraint has only one variable and is feasible, it would be solved first and removed from the optimization problem in this step. When the subsystems are more balanced, i.e., the maximum subsystem size is smaller, it is easier to have such linear constraints and the IPM solver is actually solving a problem with size $\tilde{n} < n$.

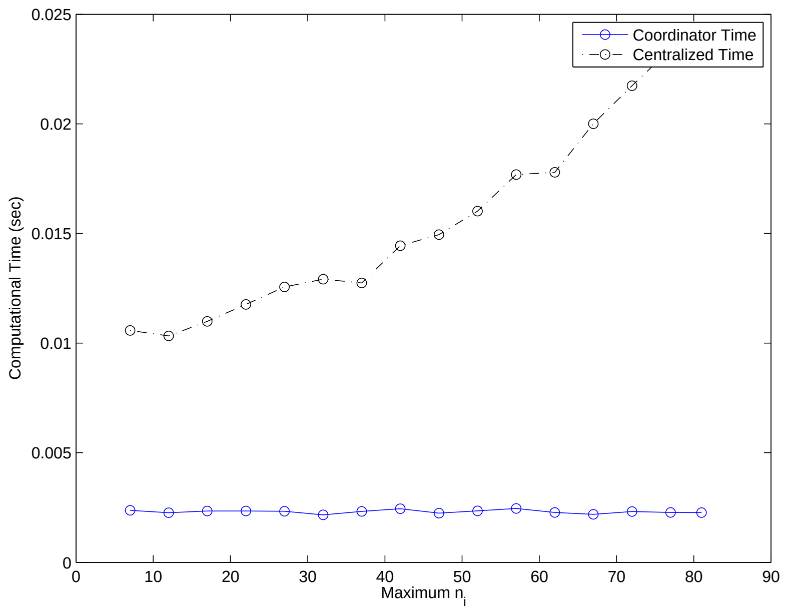


Figure 3.11: Prediction-driven CDMPC Experiment 2. Execution time of coordinator in one cycle versus execution time of centralized MPC, when MPC problems are solved numerically using the interior-point method. The solid line represents t_{coor} and the dash-dot line represents t_{CEN}. x axis is the maximum subsystem state size and y axis is the execution time in seconds.

Figure 3.12 shows how the maximum execution time of the subsystems increases as the size of the largest subsystem increases. A preliminary attempt to fit the curve suggests that it is likely a third-order or fourth-order polynomial, since the second-order fitting of the curve has large residuals and the fifth-order fitting has a negative highest order coefficient. The fitting results of third and fourth-order polynomials are:

$$\begin{aligned} t_{i_{max}} =&(5.6440 \pm 1.4142) \times 10^{-8} \max(n_i)^3 + (-4.5488 \pm 1.7910) \times 10^{-6} \max(n_i)^2 + \\ &(8.8064 \pm 6.3862) \times 10^{-7} \max(n_i) + (1.4120 \pm 0.060670) \times 10^{-2}, \end{aligned} \tag{3.93}$$

$$\begin{aligned} t_{i_{max}} =&(9.3357 \pm 4.2955) \times 10^{-10} \max(n_i)^4 + (-9.8622 \pm 7.1883) \times 10^{-8} \max(n_i)^3 + \\ &(3.7160 \pm 3.9615) \times 10^{-6} \max(n_i)^2 + (-6.3644 \pm 8.0244) \times 10^{-5} \max(n_i) + \\ &(1.4737 \pm 0.047094) \times 10^{-2}. \end{aligned} \tag{3.94}$$

Although (3.94) has dubious second- and first- order coefficients, its highest order shows that the curve is still possibly fourth-order. Since $T_{i_{maxnum}}$ was shown to be $O(\max(n_i)^{3.5}\ln(1/\varepsilon))$, the fitting results are in concordance the prediction.

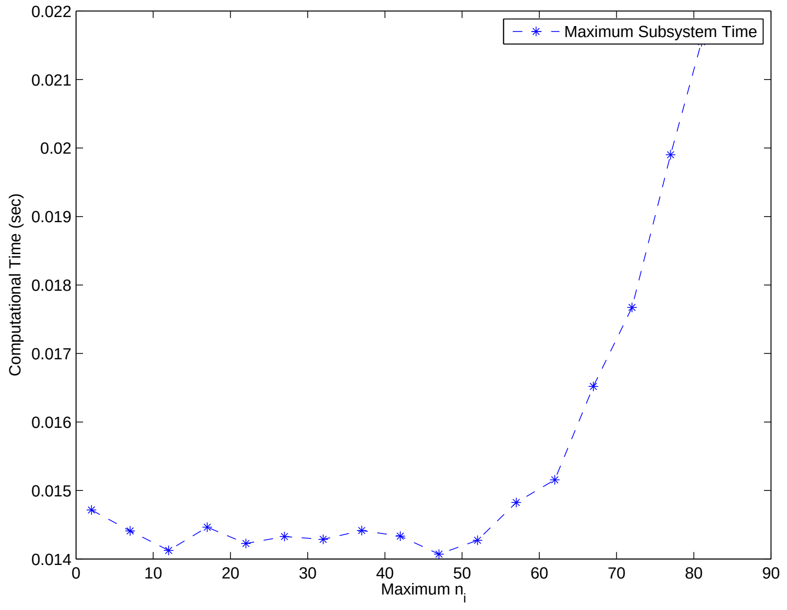


Figure 3.12: Prediction-driven CDMPC Experiment 2. Longest execution time of subsystems, when MPC problems are solved numerically using the interior-point method. x axis is the maximum subsystem size and y axis is the execution time in seconds.

As in the second scenario of Experiment 1, here t_{NonCo} is compared with t_{CEN}, and is presented in Figure 3.13. The curve of the prediction-driven CDMPC algorithm is at first above that of the centralized algorithm, but the two curves cross each other as the size of the maximum system grows.

The median levels of communication cycle numbers of all tests in Experiment 2 are presented in Figure 3.14, and all CCN datapoints in this experiment are plotted against the maximum subsystem size in Figure 3.15. From the two figures we observe that the number of communication cycles does increase as the maximum subsystem size grows, but this increase is slow. If they are compared with Figure 3.7 and Figure 3.8, it can be concluded that the maximum subsystem size is not a key factor that affects CCN.

3.4 Conclusions

In this chapter, the computational properties, namely the convergence condition, the rate of convergence and computational complexity of the unconstrained prediction-driven CDMPC

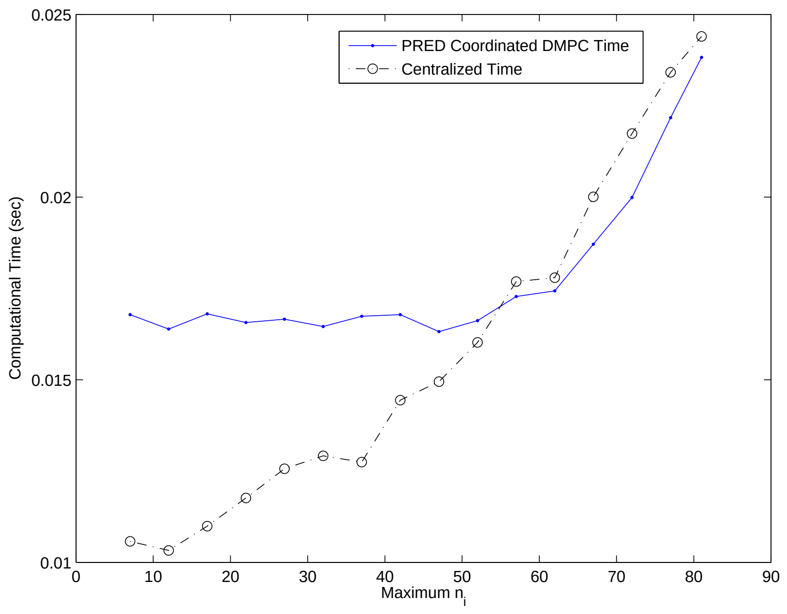


Figure 3.13: Prediction-driven CDMPC Experiment 2. Execution time of the prediction-driven CDMPC in a single cycle versus the execution time of centralized MPC, when MPC problems are solved numerically using the interior-point method. The solid line represents t_{NonCo} and the dash-dot line represents t_{CEN}. x axis is the plant state size and y axis is the execution time in seconds.

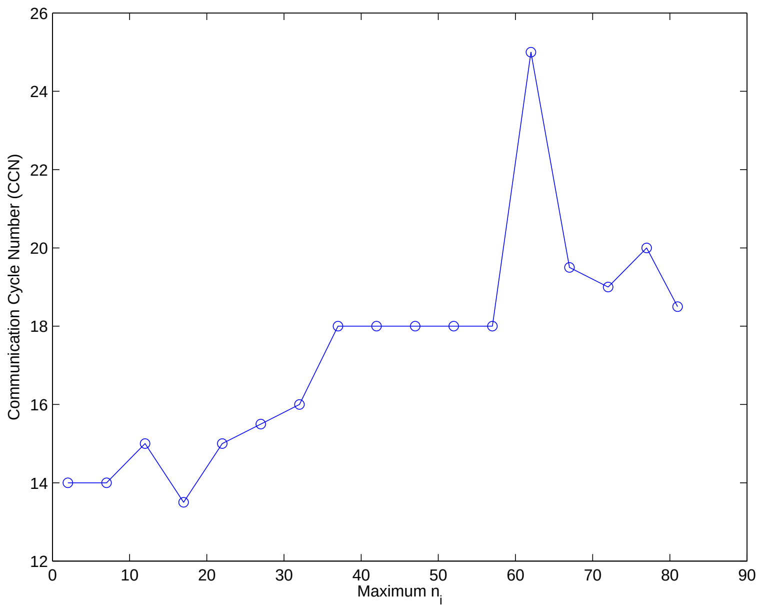


Figure 3.14: Prediction-driven CDMPC Experiment 2. The median of CCN at each experiment mode in Experiment 1, including all data points from Scenario 1 and Scenario 2, where the number of plant states $n \in \mathbb{N}$.

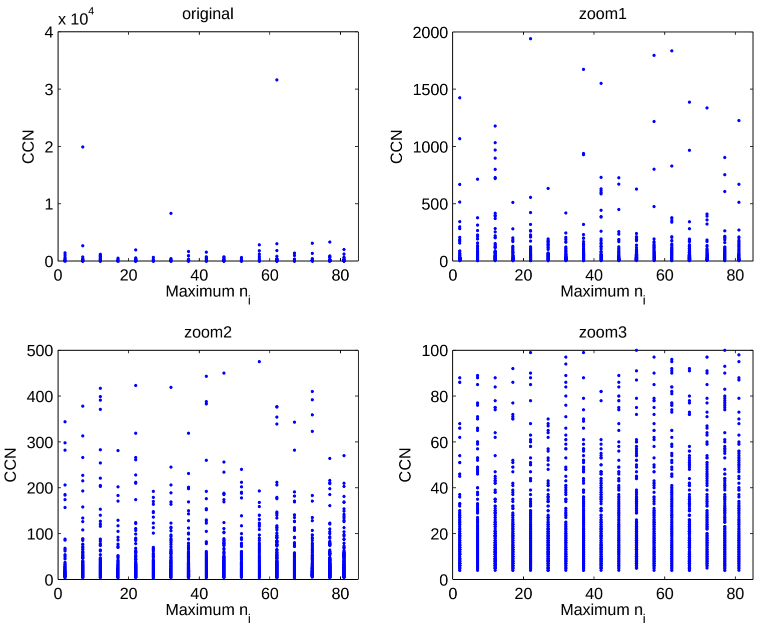


Figure 3.15: Prediction-driven CDMPC Experiment 2. All CCNs for every repetition in Experiment 2, including all data points from Scenario 1 and Scenario 2. The plots are zoomed successively. Top left: the original plot. Top right: the zoomed plot, where only CCN between 0 and 2000 are shown. Bottom left: the zoomed plot, where only CCN between 0 and 500 are shown. Bottom right: the zoomed plot, where only CCN between 0 and 100 are shown.

were investigated. For a given prediction-driven CDMPC algorithm, its convergence condition and rate of convergence are controlled by the spectral radius of a coefficient matrix defined in (3.33), which is determined only by system properties and local MPC designs. The spectral radius also affects the required communication cycle numbers of the CDMPC algorithm, which is a component part of the CDMPC algorithm's complexity. The upper bound of CCN is unknown, but a reasonable estimate is derived.

From both the theoretical and empirical analysis, the complexity of the prediction-driven CDMPC in a single communication cycle is polynomial in the problem size. In particular, when the MPC problems are solved numerically, the complexity of the single-cycled CDMPC has a lower growth rate with respect to the problem size than that of the centralized MPC, indicating that the CDMPC can get to the optimal solution more efficiently than the centralized MPC for sufficiently large-scale systems, if CCN does not grows too quickly. Empirical tests of a number of systems show that, although CCN in general has a reasonable growth rate with problem size, there is a higher probability of encountering a system with very large CCN when the problem size increases. Due to this, CCN can be considered to be the bottleneck in the computational complexity of prediction-driven CDMPC algorithm. The experiments also show that the total number of subsystems affects CCN more than the sizes of subsystems.

The results given in this chapter can be used to judge whether a specific prediction-driven CDMPC would converge or not. They may also be applied to estimate the computational costs of such a CDMPC method, and to determine when CDMPC may provide some computational advantage to centralized MPC.

Chapter 4

Computational Properties of Price-driven CDMPC

This chapter studies the computational properties of another type of coordinated DMPC, which is the price-driven CDMPC. With an analysis that parallels to Chapter 2, the unconstrained price-driven CDMPC is shown to have guaranteed convergence to the centralized MPC within two communication cycles. This fast convergence results in a low computational load, especially for a large-scale system.

4.1 Price-driven CDMPC

The price-driven Coordinated Distributed MPC (Marcos, 2011), or the Goal Coordinated Distributed MPC in (Mohseni, 2013) has the coordinating term $\{CoorTm\}_i$ in the general local MPC formulation (2.28a) to be $\boldsymbol{p}^{(s)T}\boldsymbol{\Phi}_i \begin{bmatrix} X_i \\ \Delta U_i \\ V_i \end{bmatrix}$. As in prediction-driven CDMPC, $\boldsymbol{p}^{(s)}$ is the so-called price vector and is fixed in the local MPC controller. The matrix $\boldsymbol{\Phi}_i$ is an $H_p n \times (2H_p n_i + H_u q_i)$ matrix formulated as follows:

$$\boldsymbol{\Phi}_i = \left[\begin{array}{cc:c} \boldsymbol{G}_{A_{1i}} & \boldsymbol{G}_{B_{1i}} & \boldsymbol{O} \\ & \vdots & \\ \boldsymbol{O} & \boldsymbol{O} & -\boldsymbol{I} \\ & \vdots & \\ \boldsymbol{G}_{A_{Ni}} & \boldsymbol{G}_{B_{Ni}} & \boldsymbol{O} \end{array}\right] = \left[\begin{array}{c:c} \boldsymbol{G}_{1i} & \boldsymbol{O} \\ \vdots & \vdots \\ \boldsymbol{O} & -\boldsymbol{I} \\ \vdots & \vdots \\ \boldsymbol{G}_{Ni} & \boldsymbol{O} \end{array}\right]. \tag{4.1}$$

The $\boldsymbol{G}_{A_{ji}}$ and $\boldsymbol{G}_{B_{ji}}$ matrices in (4.1) were defined in (2.20) and (2.21), respectively. The vectors X_i and ΔU_i were defined in (2.12) and (2.13), respectively. The vecotr V_i, as in the prediction-driven CDMPC, is the interaction vector and is defined as $V_i = [\hat{\boldsymbol{v}}_i(k|k), ..., \hat{\boldsymbol{v}}_i(k+H_p-1|k)]^T$ and now it is determined by the i^{th} local MPC controller as an optimization variable. The local MPC controller in the price-driven CDMPC method is then given as:

$$\min_{X_i,\Delta U_i,V_i} \mathcal{J}_i = \frac{1}{2}\big((X_i-\boldsymbol{r}_i)^T\boldsymbol{Q}_i(X_i-\boldsymbol{r}_i)+\Delta U_i^T\boldsymbol{R}_i\Delta U_i\big)+\boldsymbol{p}^{(s)^T}\boldsymbol{\Phi}_i\begin{bmatrix} X_i\\ \Delta U_i\\ V_i\end{bmatrix} \tag{4.2a}$$

subject to:

$$\big[\,\boldsymbol{G}_{ii}\,\vdots\,\boldsymbol{I}\,\big]\begin{bmatrix} X_i\\ \Delta U_i\\ V_i\end{bmatrix}=\boldsymbol{g}_i. \tag{4.2b}$$

The product $\boldsymbol{\Phi}_i\begin{bmatrix} X_i\\ \Delta U_i\\ V_i\end{bmatrix}$ is denoted as E_i, which is called the **local interaction error vector** (Marcos, 2011; Mohseni, 2013). If E_i is summed from $i=1$ to N, the summation is the **overall interaction error vector** E, i.e.,

$$E=\sum_{i=1}^{N}E_i=\sum_{i=1}^{N}\boldsymbol{\Phi}_i\begin{bmatrix} X_i\\ \Delta U_i\\ V_i\end{bmatrix}. \tag{4.3}$$

Note that if, in the equation (2.29), LHS is subtracted from the RHS and is written in matrix form, it would be:

$$\sum_{j=1,j\neq i}^{N}\boldsymbol{G}_{ij}\begin{bmatrix} X_j\\ \Delta U_j\end{bmatrix}-V_i, i=1,...,N.$$

If the above difference is aggregated from $i=1$ to N, the result is the overall interaction error vector E:

$$\begin{aligned}
&\begin{bmatrix} \sum_{j\neq 1} \boldsymbol{G}_{1j} \begin{bmatrix} X_j \\ \Delta U_j \end{bmatrix} - V_1 \\ \sum_{j\neq 2} \boldsymbol{G}_{2j} \begin{bmatrix} X_j \\ \Delta U_j \end{bmatrix} - V_2 \\ \vdots \\ \sum_{j\neq N} \boldsymbol{G}_{Nj} \begin{bmatrix} X_j \\ \Delta U_j \end{bmatrix} - V_N \end{bmatrix} \\
=&\begin{bmatrix} -V_1 \\ \boldsymbol{G}_{21} \begin{bmatrix} X_1 \\ \Delta U_1 \end{bmatrix} \\ \vdots \\ \boldsymbol{G}_{N1} \begin{bmatrix} X_1 \\ \Delta U_1 \end{bmatrix} \end{bmatrix} + \begin{bmatrix} \boldsymbol{G}_{12} \begin{bmatrix} X_2 \\ \Delta U_2 \end{bmatrix} \\ -V_2 \\ \vdots \\ \boldsymbol{G}_{N2} \begin{bmatrix} X_2 \\ \Delta U_2 \end{bmatrix} \end{bmatrix} + \cdots + \begin{bmatrix} \boldsymbol{G}_{1N} \begin{bmatrix} X_N \\ \Delta U_N \end{bmatrix} \\ \boldsymbol{G}_{2N} \begin{bmatrix} X_N \\ \Delta U_N \end{bmatrix} \\ \vdots \\ -V_N \end{bmatrix} \\
=&\boldsymbol{\Phi}_1 \begin{bmatrix} X_1 \\ \Delta U_1 \\ V_1 \end{bmatrix} + \boldsymbol{\Phi}_2 \begin{bmatrix} X_2 \\ \Delta U_2 \\ V_2 \end{bmatrix} + \cdots + \boldsymbol{\Phi}_N \begin{bmatrix} X_N \\ \Delta U_N \\ V_N \end{bmatrix} \\
=&\sum_{i=1}^{N} E_i \\
=&E.
\end{aligned} \tag{4.4}$$

Therefore, by penalizing the 'local part' of the overall interaction error vector E using a properly designed coordinating variable $\boldsymbol{p}^{(s)}$, the CDMPC algorithm is expected to iteratively drive E to $\mathbf{0}$ so that the aggregation of the N distributed MPC controllers would be equivalent to the centralized MPC problem (2.14).

Remark 4.1.1. *The formulation of local MPC controllers in the price-driven CDMPC method is similar to that in the prediction-driven method. Their main difference arises from whether or not V_i's are treated as the optimization variables in the local MPCs. Note that $\boldsymbol{\Phi}_i$ is closely related to the $\boldsymbol{\Theta}_i$ matrix, which is defined in §3.1.2. If $\tilde{\boldsymbol{I}}_i$ is used to denote the following matrix:*

$$\begin{bmatrix} \boldsymbol{O} \\ \vdots \\ \boldsymbol{I} \\ \vdots \\ \boldsymbol{O} \end{bmatrix} \quad \leftarrow \quad \begin{array}{l} i^{th} \textit{ block is a} \\ H_p n_i \times H_p n_i, \\ \textit{identity matrix,} \end{array}$$

then (3.57) can be rewritten as:

$$\boldsymbol{\Phi}_i = \left[\, \boldsymbol{\Theta}_i \,\vdots\, -\tilde{I}_i \,\right].$$

If in the local MPC problem (4.2), the vector V_i is excluded from the optimization variables, then the objective (4.2a) would be equivalent to the objective (3.1a), since the block $-\tilde{\boldsymbol{I}}_i$

can be eliminated from the matrix $\boldsymbol{\Phi}_i$. $\boldsymbol{\Phi}_i \begin{bmatrix} X_i \\ \Delta U_i \\ V_i \end{bmatrix}$ *is constructed using the information in equation (2.29), so if* V_i *is deleted from the objective function, another equation is needed to represent that part of information, which results in (3.3) in the prediction driven CDMPC method.*

After the optimal solutions $X^*_{i(s)}$, $\Delta U^*_{i(s)}$ and $V^*_i(s)$ of the local MPC problem (4.2) are determined, the local MPCs need to conduct additional computations for the coordinator to successfully update the price vector $\boldsymbol{p}^{(s)}$. The first matrix calculated is called the **sensitivity matrix**, which is defined as:

$$\nabla_{\boldsymbol{p}} Z_i = \frac{dZ_i}{d\boldsymbol{p}}, \tag{4.5}$$

where $Z_i \triangleq \begin{bmatrix} X_i \\ \Delta U_i \\ V_i \end{bmatrix}$. According to Cheng, 2007 and Marcos, 2011, the sensitivity matrix can be obtained from the following equation[1]:

$$\boldsymbol{\Lambda}_i \begin{bmatrix} \nabla_{\boldsymbol{p}} Z_i \\ \nabla_{\boldsymbol{p}} \boldsymbol{\lambda}_i \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\Phi}_i^T \\ \boldsymbol{O}_{H_p n_i \times H_p n} \end{bmatrix}, \tag{4.6}$$

where:

$$\boldsymbol{\Lambda}_i = \left[\begin{array}{ccc:c} \boldsymbol{Q}_i & & & \boldsymbol{G}_{A_{ii}}^T \\ & \boldsymbol{R}_i & & \boldsymbol{G}_{B_{ii}}^T \\ & & \boldsymbol{O}_{H_p n_i} & \boldsymbol{I}_{H_p n_i} \\ \hdashline \boldsymbol{G}_{A_{ii}} & \boldsymbol{G}_{B_{ii}} & \boldsymbol{I}_{H_p n_i} & \end{array}\right] \tag{4.7}$$

and $\boldsymbol{\lambda}_i$ are the Lagrange multipliers associated with the constraint (4.2b). Equation (4.6) is obtained by differentiating the first-order optimality condition of the i^{th} local MPC problem with respect to $\boldsymbol{p}^{(s)}$. It can be proved that for the unconstrained MPC problem (4.2), $\boldsymbol{\Lambda}_i$ is invertible (for a detailed proof, please refer to Appendix A.4). Therefore, $\nabla_{\boldsymbol{p}} Z_i$ is the first $2H_p n_i + H_u q_i$ rows of the matrix:

$$\boldsymbol{\Lambda}_i^{-1} \begin{bmatrix} -\boldsymbol{\Phi}_i^T \\ \boldsymbol{O}_{H_p n_i \times H_p n} \end{bmatrix}. \tag{4.8}$$

The local MPCs must also calculate the optimal local interaction error vector $E^*_{i(s)}$ and the optimal **local Hessian matrix** $\boldsymbol{H}^*_{i(s)}$, which are linear combinations of $Z^*_{i(s)}$ and $\nabla_{\boldsymbol{p}} Z^*_{i(s)}$, respectively:

$$E^*_{i(s)} = \boldsymbol{\Phi}_i Z^*_{i(s)}, \tag{4.9}$$

$$\boldsymbol{H}^*_{i(s)} = \boldsymbol{\Phi}_i \nabla_{\boldsymbol{p}} Z^*_{i(s)}. \tag{4.10}$$

[1]The sensitivity matrix presented here is for unconstrained MPC problems. In Cheng, 2007 and Marcos, 2011, the formulation of the sensitivity matrix for the more general, constrained price-driven CDMPC method is discussed.

Then, $E^*_{i(s)}$ and $\boldsymbol{H}^*_{i(s)}$ are sent to the coordinator.

In the coordinator, $E^{(s)} = \sum_{i=1}^{N} E^*_{i(s)}$ and the Hessian matrix:

$$\boldsymbol{H}^{(s)} = \sum_{i=1}^{N} \boldsymbol{H}^*_{i(s)} \tag{4.11}$$

are calculated. The price vector is updated by equation (4.12):

$$\boldsymbol{p}^{(s+1)} = \boldsymbol{p}^{(s)} - {\boldsymbol{H}^{(s)}}^{-1} E^{(s)}. \tag{4.12}$$

Then the iteration counter is increased by one: $s \leftarrow s + 1$, and the coordinator sends the updated $\boldsymbol{p}^{(s)}$ to the local MPC controllers. The communication between the local controllers and the coordinator is repeated iteratively, until the coordinator finds $\|E^{(s)}\|$ to be smaller than ϵ, which is a pre-defined accuracy level.

Remark 4.1.2. *The coordinator is designed from solving the* ***dual optimization problem*** *of the aggregated MPC problem:*

$$\min_{X,\Delta U,V} \mathcal{J} = \sum_{i=1}^{N} \mathcal{J}_i \tag{4.13a}$$

subject to:

$$\left[\, \boldsymbol{G}_{ii} \,\vdots\, \boldsymbol{I} \,\right] \begin{bmatrix} X_i \\ \Delta U_i \\ V_i \end{bmatrix} = \boldsymbol{g}_i, \tag{4.13b}$$

$$i = 1, ..., N,$$

where X, ΔU and V are the concatenations of X_i's, ΔU_i's and V_i's, respectively:

$$X = [X_1^T, X_2^T, ..., X_N^T]^T, \tag{4.14}$$

$$\Delta U = [\Delta U_1^T, \Delta U_2^T, ..., \Delta_N^T]^T, \tag{4.15}$$

$$V = [V_1^T, X_2^T, ..., X_N^T]^T. \tag{4.16}$$

The dual problem is presented as:

$$\begin{aligned} \max_{\boldsymbol{p}} \quad & \min_{X,\Delta U,V} \mathcal{J}(X, \Delta U, V, \boldsymbol{p}) \\ & \text{subject to:} \\ & (4.13\text{b}), i = 1, ..., N, \end{aligned} \tag{4.17}$$

which is an unconstrained convex maximization problem (Mohseni, 2013, Boyd and Vandenberghe, 2004). Therefore, its maximum can be found where the gradient of the optimization problem, which is:

$$E = \sum_{i=1}^{N} \boldsymbol{\Phi}_i \begin{bmatrix} X_i \\ \Delta U_i \\ V_i \end{bmatrix},$$

is zero (Mohseni, 2013). The Hessian of the dual problem can be obtained by:

$$\boldsymbol{H} = \frac{dE}{d\boldsymbol{p}} = \sum_{i=1}^{N} \boldsymbol{\Phi}_i \frac{d\begin{bmatrix} X_i \\ \Delta U_i \\ V_i \end{bmatrix}}{d\boldsymbol{p}} = \sum_{i=1}^{N} \boldsymbol{\Phi}_i \nabla_{\boldsymbol{p}} Z_i.$$

These lead to equations (4.10) and (4.11). If Newton's method is used to find the point where the gradient E is $\mathbf{0}$, it would result in equation (4.12). In the constrained price-driven CDMPC, Newton's method has to be modified by an optimal step θ:

$$\boldsymbol{p}^{(s+1)} = \boldsymbol{p}^{(s)} - \theta {\boldsymbol{H}^{(s)}}^{-1} E^{(s)}, \tag{4.18}$$

where $0 < \theta \leq 1$ is the maximum value that will not cause a change in the set of active constraints. In the context of unconstrained DMPC, however, the active set change is not a concern and a full Newton's step will always be taken.

An implementable algorithm of the unconstrained price-driven CDMPC is described as follows:

Algorithm 2 Implementation of Price-driven CDMPC

```
Initialization
Coordinator: Iteration counter $s \leftarrow 0$.
Coordinator: The coordinating variable $\boldsymbol{p}^{(0)}$ is arbitrarily determined.
repeat
    Coordinator: $\boldsymbol{p}^{(s)}$ is sent to to local controllers.
    Local Controllers: Local optimization problem (4.2) are solved.
    Local Controllers: Local controllers calculates $\nabla_{\boldsymbol{p}} Z_i$, $E_i^*(s)$ and $\boldsymbol{H}_i^*(s)$ according to equations (4.6), (4.9) and (4.10), respectively.
    Local Controllers: The vector $E_i^*(s)$ and the matrix $\boldsymbol{H}_i^*(s)$ are sent to coordinator.
    Coordinator: Calculate $\boldsymbol{p}^{(s)}$ based on (4.12).
    Coordinator: Iteration counter $s \leftarrow s + 1$.
until $\|E^{(s)} - E^{(s-1)}\| < \epsilon$
```

4.2 Convergence Analysis

The price-driven CDMPC method has been observed to converge quickly. Specifically, in Marcos, 2011 it is mentioned that when there is no change in active set, convergence occurs at the second communication cycle; however, there was no mathematical analysis to support this claim. In this section, we will follow a similar approach as in §3.2 and discover why the fast convergence of the unconstrained price-driven CDMPC occurs. Moreover, the rate of convergence of the method will also be given.

4.2.1 An Iterative Formulation

As its name indicates, in the price-driven DMPC method it is the price vector $\boldsymbol{p}^{(s)}$ that is transmitted between the coordinator and local MPCs and updated in each communication cycle. The algorithm can therefore be written as:

$$\boldsymbol{p}^{(s+1)} = \varphi(\boldsymbol{p}^{(s)}). \tag{4.19}$$

In this subsection, the explicit expression of φ will be derived. We will see that when the local MPCs are unconstrained, the function φ is linear and remains unchanged within the k^{th} control interval.

Local MPC Controllers

As in the analysis of §3.2.1, the first optimality condition of the i^{th} local MPC optimization problem in the s^{th} communication cycle can be written as:

$$\begin{cases} \begin{bmatrix} \boldsymbol{Q}_i & & \\ & \boldsymbol{R}_i & \\ & & \boldsymbol{O}_{H_pn_i} \end{bmatrix} \begin{bmatrix} X_i \\ \Delta U_i \\ V_i \end{bmatrix} - \begin{bmatrix} \boldsymbol{Q}_i^T \boldsymbol{r} \\ \boldsymbol{0}_{H_uq_i} \\ \boldsymbol{0}_{H_pn_i} \end{bmatrix} + \boldsymbol{\Phi}_i^T \boldsymbol{p}^{(s)} + \begin{bmatrix} \boldsymbol{G}_{ii} \,\vdots\, \boldsymbol{I} \end{bmatrix}^T \boldsymbol{\lambda}_i = \boldsymbol{0}_{2H_pn_i+H_uq_i}, \\ \begin{bmatrix} \boldsymbol{G}_{ii} \,\vdots\, \boldsymbol{I} \end{bmatrix} \begin{bmatrix} X_i \\ \Delta U_i \\ V_i \end{bmatrix} = \boldsymbol{g}_i, \end{cases} \tag{4.20}$$

where $\boldsymbol{p}^{(s)}$ is the fixed coordinating variable and $\boldsymbol{\lambda}_i$ are the Lagrange multipliers of the optimization problem (4.2). The $\boldsymbol{Q}_i^T = \boldsymbol{Q}_i$ relationship is still applicable in (4.20). Applying the partitions $\boldsymbol{\Phi}_i = \begin{bmatrix} \boldsymbol{\Theta}_i \,\vdots\, -\tilde{I}_i \end{bmatrix}$, $\boldsymbol{\Theta}_i = [\boldsymbol{\Theta}_{A_i}, \boldsymbol{\Theta}_{B_i}]$ and $\boldsymbol{G}_{ij} = [\boldsymbol{G}_{A_{ij}}, \boldsymbol{G}_{B_{ij}}]$, the zero matrices in (4.20) can be eliminated and the equation set can be rewritten as:

$$\begin{cases} -\boldsymbol{Q}_i\boldsymbol{r}_i + \boldsymbol{Q}_iX_i + \boldsymbol{\Theta}_{A_i}^T\boldsymbol{p}^{(s)} + \boldsymbol{G}_{A_{ii}}^T\boldsymbol{\lambda}_i &= \boldsymbol{0}_{H_pn_i}, \\ \boldsymbol{R}_i\Delta U_i + \boldsymbol{\Theta}_{B_i}^T\boldsymbol{p}^{(s)} + \boldsymbol{G}_{B_{ii}}^T\boldsymbol{\lambda}_i &= \boldsymbol{0}_{H_uq_i}, \\ -\tilde{\boldsymbol{I}}_i^T\boldsymbol{p}^{(s)} + \boldsymbol{\lambda}_i &= \boldsymbol{0}_{H_pn_i}, \\ \boldsymbol{G}_{A_{ii}}X_i + \boldsymbol{G}_{B_{ii}}\Delta U_i + V_i &= \boldsymbol{g}_i. \end{cases} \tag{4.21}$$

Equation (4.21) can be further written in compact matrix form as:

$$\begin{bmatrix} \boldsymbol{Q}_i & & & \boldsymbol{G}_{A_{ii}}^T \\ & \boldsymbol{R}_i & & \boldsymbol{G}_{B_{ii}}^T \\ & & \boldsymbol{O}_{H_pn_i} & \boldsymbol{I}_{H_pn_i} \\ \boldsymbol{G}_{A_{ii}} & \boldsymbol{G}_{B_{ii}} & \boldsymbol{I}_{H_pn_i} & \end{bmatrix} \begin{bmatrix} X_i \\ \Delta U_i \\ V_i \\ \boldsymbol{\lambda}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_i\boldsymbol{r}_i - \boldsymbol{\Theta}_{A_i}^T\boldsymbol{p}^{(s)} \\ -\boldsymbol{\Theta}_{B_i}^T\boldsymbol{p}^{(s)} \\ \tilde{\boldsymbol{I}}_i^T\boldsymbol{p}^{(s)} \\ \boldsymbol{g}_i \end{bmatrix}, \tag{4.22}$$

where the coefficient matrix on the LHS is $\boldsymbol{\Lambda}_i$, and as stated in §4.1, it is invertible. Therefore the optimal solution exists and is unique.

For convenience of analysis, the optimality condition (4.21) is aggregated from $i = 1$ to N:

$$\begin{cases} -\boldsymbol{Q}\boldsymbol{r} + \boldsymbol{Q}X_{MPC} + \bar{\boldsymbol{\Theta}}_A^T\boldsymbol{p}^{(s)} + \bar{\boldsymbol{G}}_A^T\boldsymbol{\lambda}_{MPC} &= \boldsymbol{0}_{H_pn}, \\ \boldsymbol{R}\Delta U_{MPC} + \bar{\boldsymbol{\Theta}}_B^T\boldsymbol{p}^{(s)} + \bar{\boldsymbol{G}}_B^T\boldsymbol{\lambda}_{MPC} &= \boldsymbol{0}_{H_uq}, \\ -\boldsymbol{p}^{(s)} + \boldsymbol{\lambda}_{MPC} &= \boldsymbol{0}_{H_pn}, \\ \bar{\boldsymbol{G}}_AX_{MPC} + \bar{\boldsymbol{G}}_B\Delta U_{MPC} + V_{MPC} &= \boldsymbol{g}, \end{cases} \tag{4.23}$$

where $\bar{\boldsymbol{G}}_A$, $\bar{\boldsymbol{G}}_B$, $\bar{\boldsymbol{\Theta}}_A$ and $\bar{\boldsymbol{\Theta}}_B$ are identically defined as in equations (3.17), (3.18), (3.19) and (3.20), respectively. The corresponding matrix formulation of (4.23) is:

$$\boldsymbol{\Lambda}\begin{bmatrix} X_{MPC} \\ \Delta U_{MPC} \\ V_{MPC} \\ \boldsymbol{\lambda}_{MPC} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}\boldsymbol{r} - \bar{\boldsymbol{\Theta}}_A^T\boldsymbol{p}^{(s)} \\ -\bar{\boldsymbol{\Theta}}_B^T\boldsymbol{p}^{(s)} \\ \boldsymbol{I}_{H_pn}^T\boldsymbol{p}^{(s)} \\ \boldsymbol{g} \end{bmatrix}, \tag{4.24}$$

where:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{Q} & & & \bar{\boldsymbol{G}}_A^T \\ & \boldsymbol{R} & & \bar{\boldsymbol{G}}_B^T \\ & & \boldsymbol{O}_{H_pn} & \boldsymbol{I}_{H_pn} \\ \bar{\boldsymbol{G}}_A & \bar{\boldsymbol{G}}_B & \boldsymbol{I}_{H_pn} & \end{bmatrix}, \tag{4.25}$$

and, as $\boldsymbol{\Lambda}_i$, $\boldsymbol{\Lambda}$ is also invertible. For the derivation to be understood more clearly, rather than using $\boldsymbol{\Lambda}^{-1}$ to solve for the unknown variables, we will follow the step-by-step fashion used in §3.2.1 to express X_{MPC}, ΔU_{MPC},V_{MPC} and $\boldsymbol{\lambda}_{MPC}$ in $\boldsymbol{p}^{(s)}$ explicitly.

From the third equation in (4.23) it can be directly obtained that:

$$\boldsymbol{\lambda}_{MPC} = \boldsymbol{p}^{(s)}, \tag{4.26}$$

which is substituted back to the first and second equations in (4.23). Because of the relationships $\boldsymbol{G}_A = \bar{\boldsymbol{G}}_A + \bar{\boldsymbol{\Theta}}_A$ and $\boldsymbol{G}_B = \bar{\boldsymbol{G}}_B + \bar{\boldsymbol{\Theta}}_B$, the two equations then become:

$$\boldsymbol{Q}X_{MPC} = -\boldsymbol{G}_A^T\boldsymbol{p}^{(s)} + \boldsymbol{Q}\boldsymbol{r}, \tag{4.27}$$

$$\boldsymbol{R}\Delta U_{MPC} = -\boldsymbol{G}_B^T\boldsymbol{p}^{(s)}. \tag{4.28}$$

Both (4.27) and (4.28) are systems with equal number of unknown variables and linearly independent equations, therefore, the solutions are:

$$X_{MPC} = -\boldsymbol{Q}^{-1}\boldsymbol{G}_A^T\boldsymbol{p}^{(s)} + \boldsymbol{r}, \tag{4.29}$$

$$\Delta U_{MPC} = -\boldsymbol{R}^{-1}\boldsymbol{G}_B^T\boldsymbol{p}^{(s)}, \tag{4.30}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are both positive definite matrices and thus invertible. (4.29) and (4.30) are substituted into the last equation of (4.23), and V_{MPC} is:

$$\begin{aligned} V_{MPC} =& -(\bar{\boldsymbol{G}}_AX_{MPC} + \bar{\boldsymbol{G}}_B\Delta U_{MPC} + \boldsymbol{g}) \\ =& -\bar{\boldsymbol{G}}_A\boldsymbol{r} + \bar{\boldsymbol{G}}_A\boldsymbol{Q}^{-1}\boldsymbol{G}_A^T\boldsymbol{p}^{(s)} + \bar{\boldsymbol{G}}_B\boldsymbol{R}^{-1}\boldsymbol{G}_B^T\boldsymbol{p}^{(s)} - \boldsymbol{g} \\ =&(\bar{\boldsymbol{G}}_A\boldsymbol{Q}^{-1}\boldsymbol{G}_A^T + \bar{\boldsymbol{G}}_B\boldsymbol{R}^{-1}\boldsymbol{G}_B^T)\boldsymbol{p}^{(s)} - \bar{\boldsymbol{G}}_A\boldsymbol{r} - \boldsymbol{g}. \end{aligned} \tag{4.31}$$

Up to this point, all the optimization variables have been represented as linear functions of $\boldsymbol{p}^{(s)}$. The focus will move to the coordinator to see how $\boldsymbol{p}^{(s+1)}$ is expressed in $\boldsymbol{p}^{(s)}$.

Coordinator

From equation (4.12) it can be seen that the two key matrices to calculate $\boldsymbol{p}^{(s+1)}$ are $E^{(s)}$ and $\boldsymbol{H}^{(s)}$. How are they presented in $\boldsymbol{p}^{(s)}$? The overall interaction error $E^{(s)}$ is the sum of $\boldsymbol{\Phi}_i Z^*_{i(s)}$, where $Z^*_{i(s)} = \begin{bmatrix} X^*_{i(s)} \\ \Delta U^*_{i(s)} \\ V^*_{i(s)} \end{bmatrix}$ is a linear function in $\boldsymbol{p}^{(s)}$ from the previous analysis. Therefore, $E^{(s)}$ is a linear function in $\boldsymbol{p}^{(s)}$ as well. Specifically:

$$\begin{aligned} E^{(s)} =& \sum_{i=1}^{N} \boldsymbol{\Phi}_i Z^*_{i(s)} = \sum_{i=1}^{N} \left[\boldsymbol{\Theta}_{A_i} \,\vdots\, \boldsymbol{\Theta}_{B_i} \,\vdots\, -\tilde{\boldsymbol{I}}_i \right] \begin{bmatrix} X^*_{i(s)} \\ \Delta U^*_{i(s)} \\ V^*_{i(s)} \end{bmatrix} \\ =& \sum_{i=1}^{N} (\boldsymbol{\Theta}_{A_i} X^*_{i(s)} + \boldsymbol{\Theta}_{B_i} \Delta U^*_{i(s)} - \tilde{\boldsymbol{I}}_i V^*_{i(s)}). \end{aligned} \tag{4.32}$$

The sum-of-product $\sum_{i=1}^{N} \boldsymbol{\Theta}_{A_i} X^*_{i(s)}$ can be re-written in the following matrix multiplication:

$$\sum_{i=1}^{N} \boldsymbol{\Theta}_{A_i} X^*_{i(s)} = \left[\boldsymbol{\Theta}_{A_1} \,\vdots\, \boldsymbol{\Theta}_{A_2} \,\vdots\, \cdots \,\vdots\, \boldsymbol{\Theta}_{A_N} \right] \begin{bmatrix} X^*_{1(s)} \\ X^*_{2(s)} \\ \vdots \\ X^*_{N(s)} \end{bmatrix} = \bar{\boldsymbol{\Theta}}_A X_{MPC}. \tag{4.33}$$

Similarly, $\sum_{i=1}^{N} \boldsymbol{\Theta}_{B_i} \Delta U^*_{i(s)}$ and $\sum_{i=1}^{N} \tilde{\boldsymbol{I}}_i V^*_{i(s)}$ can be represented as:

$$\sum_{i=1}^{N} \boldsymbol{\Theta}_{B_i} \Delta U^*_{i(s)} = \bar{\boldsymbol{\Theta}}_B \Delta U_{MPC}, \tag{4.34}$$

$$-\sum_{i=1}^{N} \tilde{\boldsymbol{I}}_i V^*_{i(s)} = -\boldsymbol{I}_{H_p n} V_{MPC} = -V_{MPC}. \tag{4.35}$$

Then, substituting equations (4.33) to (4.35) into (4.32) and using the $\boldsymbol{p}^{(s)}$-representation of X_{MPC}, ΔU_{MPC} and V_{MPC}, equation (4.32) can be transformed as:

$$\begin{aligned} E^{(s)} =& \bar{\boldsymbol{\Theta}}_A X_{MPC} + \bar{\boldsymbol{\Theta}}_B \Delta U_{MPC} - V_{MPC} \\ =& \bar{\boldsymbol{\Theta}}_A \boldsymbol{r} - \bar{\boldsymbol{\Theta}}_A \boldsymbol{Q}^{-1} \boldsymbol{G}_A^T \boldsymbol{p}^{(s)} - \bar{\boldsymbol{\Theta}}_B \boldsymbol{R}^{-1} \boldsymbol{G}_B^T \boldsymbol{p}^{(s)} - \\ & (\bar{\boldsymbol{G}}_A \boldsymbol{Q}^{-1} \boldsymbol{G}_A^T + \bar{\boldsymbol{G}}_B \boldsymbol{R}^{-1} \boldsymbol{G}_B^T) \boldsymbol{p}^{(s)} + \bar{\boldsymbol{G}}_A \boldsymbol{r} + \boldsymbol{g} \\ =& -(\bar{\boldsymbol{\Theta}}_A + \bar{\boldsymbol{G}}_A) \boldsymbol{Q}^{-1} \boldsymbol{G}_A^T \boldsymbol{p}^{(s)} - (\bar{\boldsymbol{\Theta}}_B + \bar{\boldsymbol{G}}_B) \boldsymbol{R}^{-1} \boldsymbol{G}_B^T \boldsymbol{p}^{(s)} + \\ & (\bar{\boldsymbol{\Theta}}_A + \bar{\boldsymbol{G}}_A) \boldsymbol{r} + \boldsymbol{g} \\ =& -(\boldsymbol{G}_A \boldsymbol{Q}^{-1} \boldsymbol{G}_A^T + \boldsymbol{G}_B \boldsymbol{R}^{-1} \boldsymbol{G}_B^T) \boldsymbol{p}^{(s)} + \boldsymbol{G}_A \boldsymbol{r} + \boldsymbol{g}. \end{aligned} \tag{4.36}$$

It is stated in §4.1 that the local Hessian matrices $\boldsymbol{H}_i$ can be obtained from equations (4.6) and (4.10); however, the calculation involves matrix inversion so it is not intuitive. Since the gradient vector $E^{(s)}$ is already explicitly linear in $\boldsymbol{p}^{(s)}$, the Hessian $\boldsymbol{H}^{(s)}$ follows directly by taking the derivative of $E^{(s)}$ with respect to $\boldsymbol{p}^{(s)}$. This will give us:

$$\boldsymbol{H}^{(s)} = \frac{dE^{(s)}}{d\boldsymbol{p}^{(s)}} = -(\boldsymbol{G}_A\boldsymbol{Q}^{-1}\boldsymbol{G}_A^T + \boldsymbol{G}_B\boldsymbol{R}^{-1}\boldsymbol{G}_B^T), \tag{4.37}$$

or we can write:

$$E^{(s)} = \boldsymbol{H}^{(s)}\boldsymbol{p}^{(s)} + \boldsymbol{G}_A\boldsymbol{r} + \boldsymbol{g}.$$

As $\boldsymbol{H}^{(s)}$ is invertible (proof provided in Appendix A.5), according to (4.12), there is:

$$\begin{aligned}\boldsymbol{p}^{(s+1)} =& \boldsymbol{p}^{(s)} - {\boldsymbol{H}^{(s)}}^{-1}E^{(s)} \\ =& \boldsymbol{p}^{(s)} - {\boldsymbol{H}^{(s)}}^{-1}\left[\boldsymbol{H}^{(s)}\boldsymbol{p}^{(s)} + \boldsymbol{G}_A\boldsymbol{r} + \boldsymbol{g}\right] \\ =& \boldsymbol{p}^{(s)} - \boldsymbol{p}^{(s)} - {\boldsymbol{H}^{(s)}}^{-1}(\boldsymbol{G}_A\boldsymbol{r} + \boldsymbol{g}) \\ =& (\boldsymbol{G}_A\boldsymbol{Q}^{-1}\boldsymbol{G}_A^T + \boldsymbol{G}_B\boldsymbol{R}^{-1}\boldsymbol{G}_B^T)^{-1}(\boldsymbol{G}_A\boldsymbol{r} + \boldsymbol{g}) \\ \triangleq& \boldsymbol{\chi},\end{aligned} \tag{4.38}$$

where: $\boldsymbol{G}_A$ and $\boldsymbol{G}_B$ are composed of system matrices; $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{r}$ are fixed once the MPC controllers are designed; and $\boldsymbol{g}$ remains constant within the k^{th} control interval. Therefore, vector $\boldsymbol{\chi}$ is constant within the k^{th} control interval. In equation (4.38), since $\boldsymbol{p}^{(s)}$ is added and subtracted, $\boldsymbol{p}^{(s+1)}$ will always be $\boldsymbol{\chi}$, $s = 0, 1, ...$, i.e., $\boldsymbol{p}^{(1)} = \boldsymbol{p}^{(2)} = ... = \boldsymbol{\chi}$. This indicates that no matter how $\boldsymbol{p}^{(0)}$ is initialized, after one communication cycle $\boldsymbol{p}$ would be exactly the solution. That proves the observation of the second-iteration convergence of the price-driven CDMPC method, when there is no active set change.

4.2.2 Convergence Condition and Rate of Convergence

After obtaining $\boldsymbol{p}^{(s)}$'s value for $s \geq 1$, we are in a position to present **Theorem 4.2.1**:

Theorem 4.2.1. *When the local MPCs are all unconstrained, if the solution of the centralized MPC exists, the price-driven CDMPC is guaranteed to converge to the centralized MPC, regardless of the selection of* $\boldsymbol{p}^{(0)}$.

Proof The optimal predicted states of centralized MPC are denoted to be X^*, the optimal predicted input changes to be ΔU^* and the associated Lagrange multipliers to be $\boldsymbol{\lambda}^*$. It was proved in Marcos, 2011 that when $\|E^{(s)}\| \to 0$, there are $X_{MPC} = X^*$, $\Delta U_{MPC} = \Delta U^*$ and $\boldsymbol{p}^{(s)} = \boldsymbol{\lambda}^*$. Therefore, the proof of **Theorem 4.2.1** only needs to show that the stopping criterion $\|E^{(s)}\| \to 0$ is guaranteed. To see this, we substitute (4.38) into (4.36) and get:

$$\begin{aligned} E^{(s)} =& -(\boldsymbol{G}_A\boldsymbol{Q}^{-1}\boldsymbol{G}_A^T + \boldsymbol{G}_B\boldsymbol{R}^{-1}\boldsymbol{G}_B^T)\boldsymbol{p}^{(s)} + \boldsymbol{G}_A\boldsymbol{r} + \boldsymbol{g} \\ =& \boldsymbol{H}^{(s)}\boldsymbol{p}^{(s)} + \boldsymbol{G}_A\boldsymbol{r} + \boldsymbol{g} \\ =& \boldsymbol{H}^{(s)}\left[-\boldsymbol{H}^{(s)^{-1}}(\boldsymbol{G}_A\boldsymbol{r} + \boldsymbol{g})\right] + \boldsymbol{G}_A\boldsymbol{r} + \boldsymbol{g} \\ =& \boldsymbol{0}, \forall s \geq 1. \end{aligned} \tag{4.39}$$

When $E^{(s)} = \boldsymbol{0}$, there is $\|E^{(s)}\| = 0$ and the convergence to the centralized MPC follows. $\square$

The proof also supports the observation that the unconstrained price-driven CDMPC method takes two communication cycles to converge. Particularly, if **Algorithm 2** is applied, the algorithm will not stop until $\boldsymbol{p}^{(2)}$ is calculated and the iteration counter will be 2 when the algorithm terminates. It should also be noted that the required communication cycle may vary by 1 depending on how a specific algorithm is designed. In Cheng, 2007 the algorithm requires three iteration to converge when there is no active set change.

Theorem 4.2.1 can also be interpreted in the context of iterative method. Though $\boldsymbol{p}^{(s)}$ is a constant vector after the initialization step, it can still be viewed as a special case of linear iterative method:

$$\boldsymbol{p}^{(s+1)} = \mathbf{d_1} + \mathbf{D_2}\boldsymbol{p}^{(s)}, \tag{4.40}$$

with $\mathbf{D_2} = \boldsymbol{O}$. Therefore, the convergence of the price-driven CDMPC method can also be analyzed using the techniques of iterative method analysis. In §3.2.2 it was proved that the prediction driven CDMPC converges to the centralized solution if and only if $\rho(\mathbf{C_2}) < 1$, where $\mathbf{C_2}$ is defined in (3.33). The analogy stands true for the price-driven CDMPC as well. A price-driven CDMPC can be formulated as (4.40) and will converge to the centralized solution if and only if $\rho(\mathbf{D_2}) < 1$. This statement can be proved using the same approach as in §3.2.2, so the proof is omitted here. The coefficient matrix $\mathbf{D_2}$ is $\boldsymbol{O}$, making $\rho(\mathbf{D_2}) = 0$. Consequently, the convergence condition $\rho(\mathbf{D_2}) < 1$ is always satisfied, which guarantees the convergence.

We are also curious about the rate of convergence of the price-driven CDMPC method. In this case, there would be some difficulties in the analysis when using the iterative method approach. According to **Definition 3.2.1**, the average convergence rate of (4.40) after s steps is

$$R_s(\mathbf{D_2}) = -\frac{1}{s}\log\|\mathbf{D_2}^s\|, \tag{4.41}$$

and the asymptotic convergence rate is

$$R(\mathbf{D_2}) = -\log\rho(\mathbf{D_2}), \tag{4.42}$$

which would both be ∞ because $\mathbf{D_2}$ is a zero matrix. A similar situation arises in the attempt to use the definition mentioned in **Remark 3.2.1**, where the rate of convergence is defined as the limit of two successive errors. If we define the difference between $\boldsymbol{p}^{(s)}$ and its limit $\boldsymbol{p}^{(\infty)}$ to be the error vector $\boldsymbol{e}^{(s)}$ and use matrix norm to measure the 'length' of error, there is

$$R = \lim_{s\to\infty} \frac{\|\boldsymbol{e}^{(s+1)}\|}{\|\boldsymbol{e}^{(s)}\|}. \tag{4.43}$$

As shown in (4.38), $\boldsymbol{p}^{(s)} = \boldsymbol{p}^{(\infty)}$, $\forall s \geq 1$. The error vector $\boldsymbol{e}^{(s)}$ therefore remains $\mathbf{0}$ after the first step, resulting in the '0 over 0' structure in equation (4.43). The reason for these difficulties is that the concept of convergence rate is developed for sequences that tend to have infinite number of members, or algorithms with infinite steps. For example, in the prediction-driven CDMPC, the rate of convergence gives us some insight into how far it is from the limit and how many iterations are needed to be accurate enough. If an algorithm always reaches the exact solution in a few steps, these will be certain information and will not be our concern.

If the two rate of convergence results are compared against each other, the one obtained in the iterative method framework is more preferable than the other. Though a specific rate does not exist, the result '∞' can be interpreted as 'converges very fast'.

4.3 Computational Complexity

The fast-convergence of the price-driven CDMPC gives us a taste of its efficiency for the unconstrained MPCs. In this section, we will discuss the complexity of the price-driven CDMPC to see how much computational effort it requires to reach the centralized solution and whether or not it is indeed a computational preferable method.

4.3.1 Theoretical Analysis

Since CDMPCs all have the same computation hierarchy, the complexity of price-driven CDMPC shares the same expression with that of prediction-driven CDMPC, which is

$$\begin{aligned} T &= T_{NonCo} \times CCN \\ &= (T_{coor} + \max_i T_i) \times CCN. \end{aligned}$$

As discussed in §4.2.1, the communication cycle number is a fixed number when local MPCs are all unconstrained, so CCN can be taken as a coefficient. According to the third rule of **Property B.0.3**, CCN can be omitted when calculating the complexity of the price-driven

CDMPC, which is thus presented as:

$$T_{PRI} = T_{coor} + \max_i T_i. \tag{4.44}$$

As in §3.3.1, the two component parts will be analyzed respectively, where the situations of MPCs solved analytically and numerically are discussed separately.

Local MPCs: Solved Analytically

From §4.1, the matrices sent from local MPC to the coordinator are the local interaction error vector $E_{i(s)}$ and the local Hessian matrix $\boldsymbol{H}_{i(s)}$, where $\boldsymbol{H}_{i(s)} = \boldsymbol{\Phi}_i \nabla_{\boldsymbol{p}} Z_i$ and $\nabla_{\boldsymbol{p}} Z_i$ is the first $2H_p n_i + H_u q_i$ rows of:

$$\boldsymbol{\Lambda}_i^{-1} \begin{bmatrix} -\boldsymbol{\Phi}_i^T \\ \boldsymbol{O}_{H_p n_i \times H_p n} \end{bmatrix}$$

Note that $\boldsymbol{\Lambda}_i^{-1}$ and $\boldsymbol{\Phi}_i$ are fixed matrices for an existing decentralized MPC network, so both $\nabla_{\boldsymbol{p}} Z_i$ and $\boldsymbol{H}_{i(s)}$ can be calculated off-line beforehand and '(s)' in the subscript of $\boldsymbol{H}_{i(s)}$ can be eliminated from here. Therefore, the computations that take place in the i^{th} local distributed MPCs include: 1) updating the coordinating term $\boldsymbol{p}^{(s)T} \boldsymbol{\Phi}_i$; 2) solving the MPC problem (4.2); 3) calculating the local interaction error vector $E_{i(s)}$.

The optimal $X_{i(s)}^*$, $\Delta U_{i(s)}^*$ and $V_i^*(s)$ are obtained from the problem's first-order optimality condition (4.20), which is equivalent to:

$$\begin{bmatrix} X_i \\ \Delta U_i \\ V_i \\ \boldsymbol{\lambda}_i \end{bmatrix} = \boldsymbol{\Lambda}_i^{-1} \begin{bmatrix} \boldsymbol{Q}_i \boldsymbol{r}_i - \boldsymbol{\Theta}_{A_i}^T \boldsymbol{p}^{(s)} \\ -\boldsymbol{\Theta}_{B_i}^T \boldsymbol{p}^{(s)} \\ \tilde{\boldsymbol{I}}_i^T \boldsymbol{p}^{(s)} \\ \boldsymbol{g}_i \end{bmatrix}, \tag{4.45}$$

as discussed in §4.2.1. The complexity of each step in the i^{th} local MPC and the dimensions of related parameters are listed in Table 4.1. Adding up the complexity of each step, there is the complexity of the analytically-solved local MPC:

$$\begin{aligned} T_{i_{anl}} =& O\left(2H_p n(2H_p n_i + H_u q_i) + (3H_p n_i + H_u q_i) + (3H_p n_i + H_u q_i)^2\right) \\ =& O\left(\max(4H_p^2 n n_i, 2H_p H_u n q_i, 9H_p^2 n_i^2, 6H_p H_u n_i q_i, H_u q_i^2)\right), \end{aligned} \tag{4.46}$$

where:

$$0 \le H_p H_u n q_i \le H_p^2 n n_i,$$

$$0 \le H_u q_i^2 \le H_p H_u n_i q_i \le H_p^2 n_i^2,$$

$$0 \le H_p^2 n_i^2 \le H_p^2 n n_i.$$

Table 4.1: Local MPC (Solved Analytically) Complexity in Price-driven CDMPC: Step by Step

Step Description	**Parameters and Dimensions**	**Step Complexity**
1. Calculate $\boldsymbol{\phi}_i = \begin{bmatrix} \boldsymbol{\Theta}_{A_i}^T \\ \boldsymbol{\Theta}_{B_i}^T \\ -\tilde{\boldsymbol{I}}_i^T \end{bmatrix} \boldsymbol{p}^{(s)}$	$\boldsymbol{\Theta}_{A_i} : H_p n \times H_p n_i$, $\boldsymbol{\Theta}_{B_i} : H_p n \times H_u q_i$, $\tilde{\boldsymbol{I}}_i : H_p n \times H_p n_i$ $\boldsymbol{p}^{(s)} : H_p n \times 1$	$O(H_p n(2H_p n_i + H_u q_i))$
2. Calculate $\boldsymbol{\eta}_i = \begin{bmatrix} \boldsymbol{Q}_i \boldsymbol{r}_i \\ \boldsymbol{0}_{H_p n_i + H_u q_i} \\ \boldsymbol{g}_i \end{bmatrix} - \begin{bmatrix} \boldsymbol{\phi}_i \\ \boldsymbol{0}_{H_p n_i} \end{bmatrix}$	$\boldsymbol{\phi}_i : (2H_p n_i + H_u q_i) \times 1$	$O(3H_p n_i + H_u q_i)$
3. Calculate $\begin{bmatrix} X_i \\ \Delta U_i \\ V_i \\ \boldsymbol{\lambda}_i \end{bmatrix} = \boldsymbol{\Lambda}_i^{-1} \boldsymbol{\eta}_i$	$\boldsymbol{\Lambda}_i : (3H_p n_i + H_u q_i)^2$, $\boldsymbol{\eta}_i : (3H_p n_i + H_u q_i) \times 1$	$O\left((3H_p n_i + H_u q_i)^2\right)$
4. Calculate $E_{i(s)} = \boldsymbol{\Phi}_i \begin{bmatrix} X_i \\ \Delta U_i \\ V_i \end{bmatrix}$	$\boldsymbol{\Phi}_i : H_p n \times$ $(2H_p n_i + H_u q_i)$, $X_i : H_p n_i \times 1$, $\Delta U_i : H_u q_i \times 1$ $V_i : H_p n_i \times 1$	$O\left(H_p n(2H_p n_i + H_u q_i)\right)$

Equation (4.46) is then simplified to:

$$\begin{aligned} T_{i_{anl}} =& O\left(2H_p^2 n n_i + 9H_p^2 n_i^2\right) \\ =& O(2H_p^2 n n_i + 9H_p^2 n n_i) \\ =& O(n_i n). \end{aligned} \tag{4.47}$$

The complexity result is the same as that of the analytically solved local MPC in the prediction-driven CDMPC. This is reasonable, because when the local MPCs are solved analytically, only matrix multiplications and summations are involved. Their complexities of the two matrix operations can be obtained by counting the matrix dimensions. In both cases, the matrices with the largest dimension are in their coordinating terms, which link the local MPC to the coordinator. The resulting complexities therefore have n_i coming from the subsystems and n coming from the coordinator.

Local MPCs: Solved Numerically

Complexity analysis of the local MPCs in the price-driven CDMPCs is also extended to a more practical scenario, where the unconstrained MPC problems are solved by IPM. The complexity of IPM for solving general QP problems was discussed in §3.3.1. Note that in the price-driven CDMPC, the local MPC problems are still QPs, so all the complexity analysis of IPM in §3.3.1 applies here. We know that solving an QP with the form of (3.59) requires $O(m^{3.5}\ln(1/\varepsilon))$ complexity, where m is the number of decisions variable and ε is the pre-defined accuracy threshold of the IPM algorithm. When the MPC problem (4.2) is transformed into the general QP problem (3.59), there is $m = 2H_pn_i + H_uq_i$, then it takes the i^{th} local MPC controller $O((2H_pn_i + H_uq_i)^{3.5}\ln(1/\varepsilon))$ operations to solve for $X^*_{i(s)}$, $\Delta U^*_{i(s)}$ and $V^*_{i(s)}$. As $q_i \leq n_i$, the complexity of solving the i^{th} local MPC problem with IPM is:

$$T_{i_{IPM}} = O(n_i^{3.5}\ln(1/\varepsilon)). \tag{4.48}$$

Before and after $X^*_{i(s)}$, $\Delta U^*_{i(s)}$ and $V^*_{i(s)}$ are solved, the local controllers need to update the coordinating term and calculate the local interaction error $E_{i(s)}$, respectively. That is to say, the numerically solved local MPCs must also complete Step 1 and Step 4 in Table 4.1. Then the complexity of the local MPC in total is:

$$\begin{aligned} T_{i_{num}} =& T_{i_{up}} + T_{i_{IPM}} + T_{i_E} \\ =& O\left(2H_pn(2H_pn_i + H_uq_i) + n_i^{3.5}\ln(1/\varepsilon)\right) \\ =& O\left(max(4H_p^2nn_i, n_i^{3.5}\ln(1/\varepsilon))\right), \end{aligned} \tag{4.49}$$

where $T_{i_{IPM}} > T_{i_{up}} = T_{i_E}$ when $n, n_i \to \infty$ and $\frac{n}{n_i} < \infty$. Therefore, the complexity of the i^{th} local MPC when it is solved by IPM is:

$$T_{i_{num}} = O\left(n_i^{3.5}\ln(1/\varepsilon)\right). \tag{4.50}$$

Coordinator

In each communication cycle, the coordinator updates the price vector according to equation (4.12), where $E^{(s)} = \sum_{i=1}^{N} E_{i(s)}$ and $\boldsymbol{H}^{(s)} = \sum_{i=1}^{N} \boldsymbol{H}_{i(s)}$. As discussed in the analysis of local MPCs, $\boldsymbol{H}_{i(s)}$ are fixed, so the Hessian $\boldsymbol{H}^{(s)}$ remains unchanged when the local MPCs are unconstrained. Consequently, the Hessian can be calculated off-line and its superscript '(s)' will be omitted. Table 4.2 lists the computation steps in coordinator. Note that these steps are the same in both situations when the local MPC problems are solved analytically and numerically. The coordinator complexity is the summation of complexities from Step

Table 4.2: Coordinator Complexity in Price-driven CDMPC: Step by Step

Step Description	**Parameters and Dimensions**	**Step Complexity**
1. Calculate $E^{(s)} = \sum_{i=1}^{N} E_{i(s)}$	$E_{i(s)} : H_p n \times 1$	$O(NH_p n)$
2. Calculate $S^{(s)} \triangleq -\boldsymbol{H}^{-1} E^{(s)}$	$E^{(s)} : H_p n \times 1$, $\boldsymbol{H} : H_p n \times H_p n$	$O(H_p^2 n^2)$
3. Calculate $\boldsymbol{p}^{(s+1)} = \boldsymbol{p}^{(s)} + S^{(s)}$	$S^{(s)} : H_p n \times 1$, $\boldsymbol{p}^{(s)} : H_p n \times 1$	$O(H_p n)$

1 to Step 3:

$$\begin{aligned} T_{coor} =& O(NH_p n + H_p^2 n^2 + H_p n) \\ =& O(\max(NH_p n, H_p^2 n^2, H_p n)). \end{aligned} \tag{4.51}$$

In (4.51), there is always $0 \le N \le n$, because each subsystem has at least one state. As a result, $0 \le H_p n \le NH_p n \le H_p^2 n^2$, so the complexity of the coordinator is:

$$T_{coor} = O(H_p^2 n^2) = O(n^2). \tag{4.52}$$

Algorithm Complexity and a Comparison with Centralized Complexity

Table 4.3 summarizes the above complexity analysis. The overall complexity of the price-

Table 4.3: Complexity of Price-driven CDMPC and Centralized MPC

Complexity Type	**Solving Analytically**	**Solving Numerically**
T_{coor}	$O(n^2)$	$O(n^2)$
$\max_i T_i$	$O(\max(n_i)n)$	$O(\max(n_i)^{3.5} \ln(1/\varepsilon))$
CCN	constant coefficient	
T_{CEN}	$O(n^2)$	$O(n^{3.5} \ln(1/\varepsilon))$

driven CDMPC method is then:

$$\begin{aligned} T_{PRI} =& T_{coor} + \max_i T_i \\ =& \begin{cases} O(n^2 + \max(n_i)n), \text{MPC solved analytically}, \\ O(n^2 + \max(n_i)^{3.5} \ln(1/\varepsilon)), \text{MPC solved numerically}, \end{cases} \end{aligned} \tag{4.53}$$

Table 4.3 also lists the complexity of centralized MPC as a comparison to the the price-driven CDMPC. The number of communication cycles for the price-driven CDMPC is fixed to be 2, making it more efficient than the prediction-driven DMPC. Within the parallel computing environment, it is expected that a properly designed price-driven CDMPC network should have computational advantage over the centralized MPC for large-scale systems as well.

When MPCs are solved analytically, the ratio of required operations between the coordinator in one cycle and the centralized MPC is $r = \frac{T_{coor}}{T_{CEN}} = \frac{H_p^2 n^2}{(H_p^2+1)n^2+(H_pH_u+1)nq}$ when $n, q \to \infty$. There is always $r < 1$ regardless of the value of H_p, H_q, n and q. The operations of the i^{th} subsystem in this case is $(4H_p^2+1)n_in + n_iq + 2H_pH_uq_in + 9H_p^2n_i^2 + 6H_pH_un_iq_i + H_u^2q_i^2$. Consequently, whether or not T_{PRI} is smaller than T_{CEN} depends on H_p, H_q, n, n_i, q and n_i. Taking a simplified scenario as example, we assume $n_i = q_i, \forall i$ and $H_p = H_u$. Under these assumptions, it requires:

$$2\left(H_p^2n^2 + (6H_p^2+2)\max(n_i)n + 16H_p^2\max(n_i)^2\right) < (2H_p^2+2)n^2 \tag{4.54}$$

if the price-driven CDMPC is to be faster than the centralized MPC. Solving (4.54) for $\max(n_i)$ would yield

$$0 < \max(n_i) < \frac{\sqrt{73H_p^2+6H_p+1} - (3H_p^2+1)}{16H_p^2}n,$$

where the upper bound is a monotonically increasing function with respect to H_p and has limits of $\lim_{H_p\to 1}\frac{\sqrt{73H_p^2+6H_p+1}-(3H_p^2+1)}{16H_p^2}n = 0.3090n$, $\lim_{H_p\to\infty}\frac{\sqrt{73H_p^2+6H_p+1}-(3H_p^2+1)}{16H_p^2}n = 0.3465n$. The result shows that if the number of subsystems is large enough and the local MPCs have balanced computational load, the price-driven CDMPC is computationally more favorable than the centralized MPC.

For the case of numerically solved MPCs, the comparison is almost the same as in the prediction-driven, coordinated DMPC because $T_{coor_{PRED}}$, $T_{coor_{PRI}}$ are both $O(n^2)$ and $T_{i_{PRED}}$, $T_{i_{PRI}}$ are both $O(n_i^{3.5}\ln(1/\varepsilon))$. The only difference in the comparison is that CCN in the price-driven CDMPC is a small constant, which can be removed from the analysis. Therefore we conclude that the price-driven CDMPC algorithm is a preferable choice for a large-scale system, when the MPCs are solved numerically.

4.3.2 Empirical Analysis

Two sets of experiments were conducted to study the empirical complexity of the price-driven CDMPC and to verify the theoretical analysis in §4.3.1. In these two experiments, the systems and parameters were generated using the same method as in the empirical complexity study of the prediction-driven CDMPC. The experiments also share the assumptions with the prediction-driven CDMPC experiments, as well as the computing environment and platform, which were all described in §3.3.2. The empirical complexities of the coordinator in one cycle, the maximum-sized subsystem and the price-driven CDMPC algorithm are respectively represented by t_{coor}, $t_{i_{max}}$ and t_{PRI}, where t_{coor} was defined in (3.77), $t_{i_{max}}$ in (3.78) and:

$$t_{PRI} = (t_{coor} + t_{i_{max}}) \times CCN = 2(t_{coor} + t_{i_{max}}). \tag{4.55}$$

The execution time t_{CEN} was also recorded to represent the empirical complexity of the centralized MPC.

The objective of the first experiment was to discover how t_{coor}, $t_{i_{max}}$, t_{PRI} and t_{CEN} scale as the plant state size n grows. To achieve this goal, we chose identically sized subsystems and gradually increased the number of the subsystems. The theoretical analysis predicts that t_{coor} and t_{CEN} should be quadratic in n while $t_{i_{max}}$ is a linear function of n, when the MPCs are solved analytically. If the MPCs are solved numerically, t_{coor} and $t_{i_{max}}$ have the same hypotheses as the other scenario; whereas t_{CEN} is expected to increase polynomially with an order higher than 3. Experiment 2 focused on the parameter $\max(n_i)$, so the plant state size n, plant input size q and the number of subsystems N are fixed. According to Table 4.3, t_{coor} and t_{CEN} should remain constant when n is fixed, no matter how the MPCs are solved. For the subsystems, $t_{i_{max_{anl}}}$ is expected to be linear in $\max(n_i)$ and $t_{i_{max_{num}}}$ be polynomial with an order higher than 3.

The theoretical result of CCN always being 2 can be tested by counting the communication cycle numbers in each repetition of the experiments. Since all the systems were generated from the Monte Carlo simulations, we did not design an individual experiment for CCN.

Experiment 1: Empirical Complexity Varying With Number of Subsystems

All subsystems are fixed to be 2×2 in Experiment 1. The number of subsystems N are sequentially chosen from the set $\mathbb{N}$, which was defined in 3.81. Details of the experiment can be found in Appendix C.

Scenario 1: The MPCs are all solved analytically in Scenario 1. The execution times of the coordinator in one communication cycle and the centralized MPC are compared in Figure 4.1. If we substitute $n = q$, $H_p = 2H_u$ into the coordinator-to-centralized ratio r, which was introduced in §4.3.1, there is $r = \frac{1}{2}$. As can be seen from the figure, the curve of t_{coor} is always below that of t_{CEN}, where the ratio r is empirically about 0.3. Both curves present polynomial characteristics. The least-squares fitting were performed to see if they are second-order polynomials as predicted. Since there is a visible outlier in the curve of t_{CEN}, it is removed first before the fitting. When both curves are assumed to be quadratic,

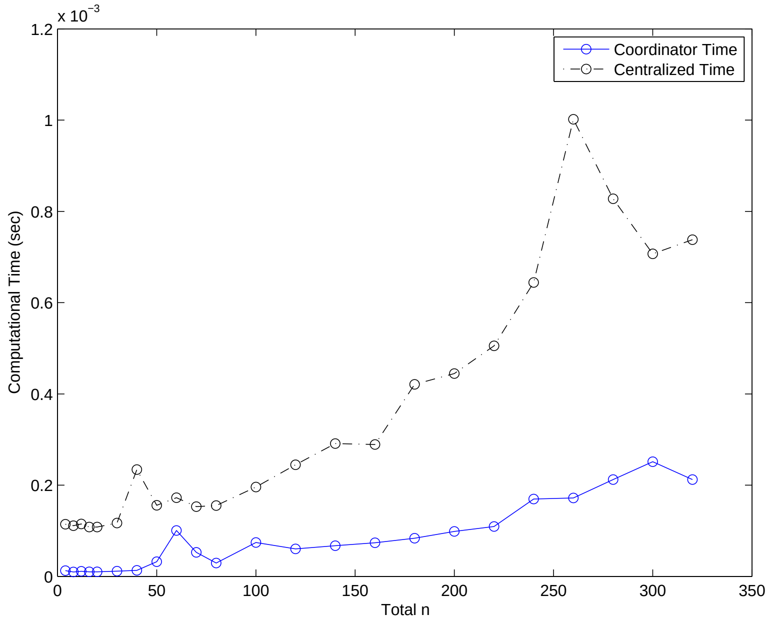


Figure 4.1: Price-driven CDMPC Experiment 1. Execution time of coordinator in one cycle versus execution time of centralized MPC, when MPCs are solved analytically. The solid line represents t_{coor} and the dash-dot line represents t_{CEN}. x axis is the plant state size and y axis is the execution time in seconds.

the fitting results are:

$$\begin{aligned} t_{coor} =&(1.5632 \pm 1.2054) \times 10^{-9} n^2 + \\ &(2.0253 \pm 3.7409) \times 10^{-7} n + (1.4788 \pm 2.0981) \times 10^{-5}, \end{aligned} \tag{4.56}$$

$$\begin{aligned} t_{CEN} =&(4.4648 \pm 2.8710) \times 10^{-9} n^2 + \\ &(8.5037 \pm 8.9136) \times 10^{-7} n + (9.9568 \pm 4.9992) \times 10^{-5}. \end{aligned} \tag{4.57}$$

If t_{coor} and t_{CEN} are assumed to be third-order polynomials, they are fitted as:

$$\begin{aligned} t_{coor} =&(0.76901 \pm 1.4466) \times 10^{-11} n^3 + (-2.0402 \pm 6.8842) \times 10^{-9} n^2 + \\ &(6.3080 \pm 8.8784) \times 10^{-7} n + (0.60882 \pm 2.6566) \times 10^{-5} \end{aligned} \tag{4.58}$$

$$\begin{aligned} t_{CEN} =&(-3.4597 \pm 3.2486) \times 10^{-11} n^3 + (2.0698 \pm 1.5467) \times 10^{-8} n^2 + \\ &(-1.0665 \pm 1.9752) \times 10^{-6} n + (1.3817 \pm 5.8252) \times 10^{-4}. \end{aligned} \tag{4.59}$$

Note that both of the third-order fittings do not make sense: for t_{coor}, the third-order coefficient contains 0 in its 95% interval; for t_{CEN}, the third-order coefficient has negative value. For these reasons, we say that t_{coor}, and t_{CEN} are both statistically quadratic in n, as predicted.

Figure 4.2 shows the maximum of the subsystem execution time . From Table 4.3, the complexity of the analytically solved i^{th} local MPC is $O(n_i n)$. The linear and quadratic fitting of $t_{i_{max}}$ are:

$$t_{i_{max}} = (7.7774 \pm 1.2741) \times 10^{-7} n + (1.0341 \pm 0.20701) \times 10^{-4}, \tag{4.60}$$

$$\begin{aligned} t_{i_{max}} = {} & (-1.0076 \pm 1.5626) \times 10^{-9} n^2 + \\ & (1.0798 \pm 0.48494) \times 10^{-7} n + (9.1795 \pm 2.7198) \times 10^{-5}, \end{aligned} \tag{4.61}$$

$$\tag{4.62}$$

and the second-order fitting is apparently invalid due to the negative quadratic coefficient. Therefore the empirical complexity $t_{i_{max}}$ can be concluded to be statistically linear in n.

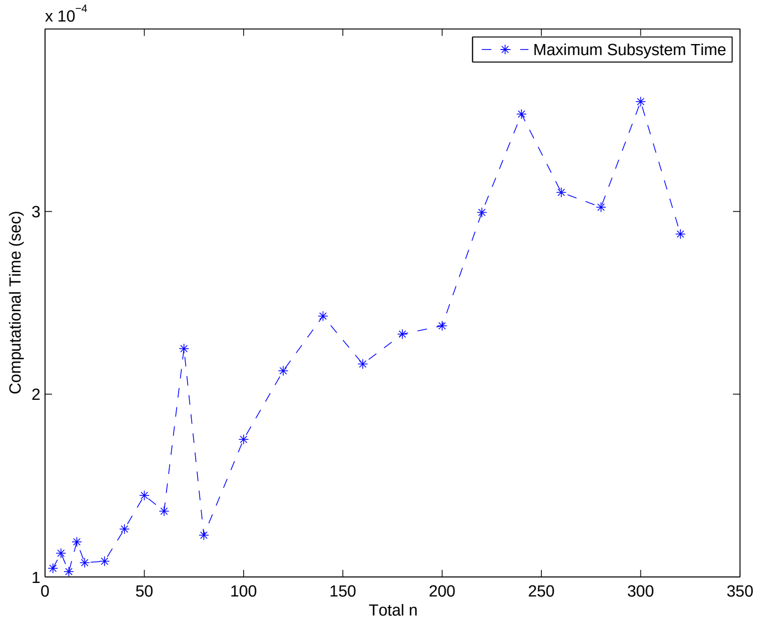


Figure 4.2: Price-driven CDMPC Experiment 1. The longest execution time of subsystems, when MPCs are solved analytically. x axis is the plant state size and y axis is the execution time in seconds.

In Figure 4.3, we present the execution time of the price-driven CDMPC in a single cycle and the centralized MPC. As can be seen, t_{NonCo} is above t_{CEN}, and then the two curves cross since t_{CEN} grows more rapidly. Note that the number of communication cycles is only two. When the number of subsystems grows larger, t_{PRI} would eventually be smaller than t_{CEN} as well.

Scenario 2: For the scenario where the MPCs are solved numerically, the comparison of t_{coor} and t_{CEN} are presented in Figure 4.4. When least-squares regression is applied to fit the curves, the first datapoint, which is a visible outlier, is removed from the set of

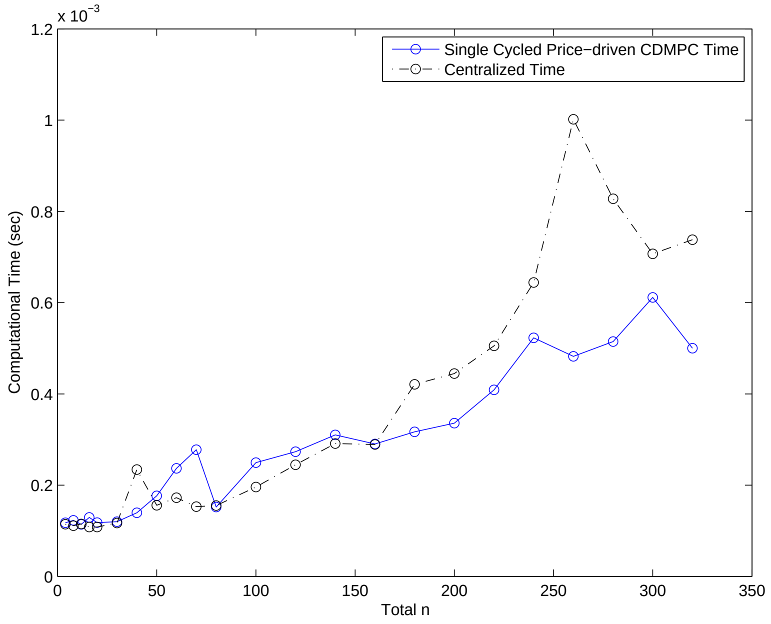


Figure 4.3: Price-driven CDMPC Experiment 1. Execution time of the price-driven CDMPC in a single cycle versus the execution time of centralized MPC, when MPC problems are solved analytically. The solid line represents t_{NonCo} and the dash-dot line represents t_{CEN}. x axis is the plant state size and y axis is the execution time in seconds.

datapoints. The second- and third-order fitting of t_{CEN} are:

$$\begin{aligned} t_{CEN} =&(3.7132 \pm 0.26204) \times 10^{-7} n^2 + \\ &(-2.1573 \pm 0.82791) \times 10^{-5} n + (3.9734 \pm 0.48237) \times 10^{-3}, \end{aligned} \tag{4.63}$$

$$\begin{aligned} t_{CEN} =&(1.5974 \pm 3.1867) \times 10^{-10} n^3 + (2.9527 \pm 1.5395) \times 10^{-7} n^2 + \\ &(-1.2229 \pm 2.0401) \times 10^{-5} n + (3.7597 \pm 0.64409) \times 10^{-3}, \end{aligned} \tag{4.64}$$

respectively. For t_{coor}, the second- and third-order fitting results are:

$$\begin{aligned} t_{coor} =&(1.0905 \pm 0.14653) \times 10^{-9} n^2 + \\ &(2.3599 \pm 0.45474) \times 10^{-7} n + (7.1609 \pm 2.5504) \times 10^{-6}, \end{aligned} \tag{4.65}$$

$$\begin{aligned} t_{coor} =&(-0.46838 \pm 1.8009) \times 10^{-12} n^3 + (1.3100 \pm 0.85702) \times 10^{-9} n^2 + \\ &(2.0990 \pm 1.1053) \times 10^{-7} n + (7.6908 \pm 3.3073) \times 10^{-6}, \end{aligned} \tag{4.66}$$

Note that the third-order coefficient of the fitted polynomials contain 0 in their 95% confidence intervals, so we conclude that the empirical complexities of the numerically solved centralized MPC and the coordinator are both second-order polynomials.

According to the theoretical analysis, the maximum time taken by the local MPCs should be linear to the plant size n, which is the same as in Scenario 1 when MPCs are solved analytically. Figure 4.5 shows this and it indeed has a similar pattern as Figure 4.2.

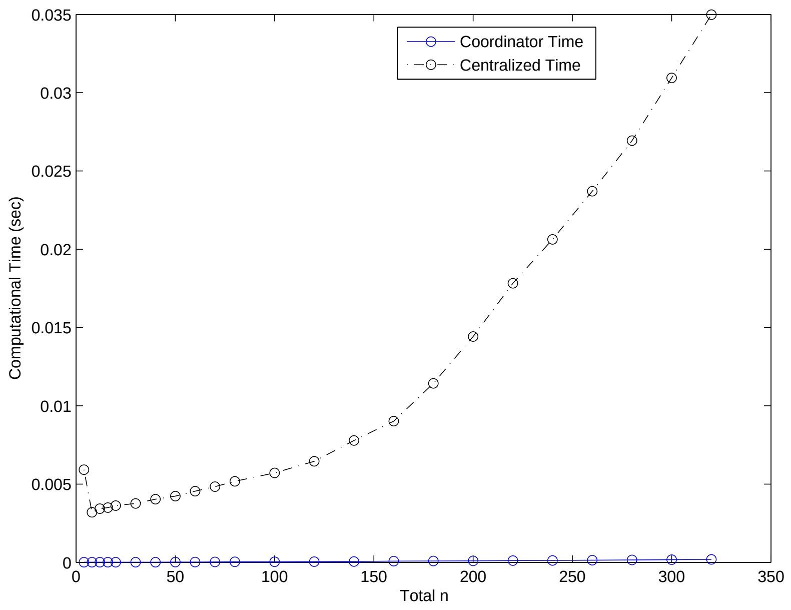


Figure 4.4: Price-driven CDMPC Experiment 1. Execution time of coordinator in one cycle versus execution time of centralized MPC, when MPCs are solved numerically using the interior-point method. The solid line represents t_{coor} and the dash-dot line represents t_{CEN}. x axis is the plant state size and y axis is the execution time in seconds.

Fitting $t_{i_{max}}$ with the assumptions that it is linear and quadratic functions in n, we have

$$t_{i_{max}} = (1.4159 \pm 0.28790) \times 10^{-6} n + (2.7644 \pm 0.046777) \times 10^{-3}, \tag{4.67}$$

$$\begin{aligned} t_{i_{max}} &= (-5.4213 \pm 2.6836) \times 10^{-9} n^2 + \\ &(3.0411 \pm 0.83283) \times 10^{-7} n + (2.7019 \pm 0.046709) \times 10^{-3}, \end{aligned} \tag{4.68}$$

$$\tag{4.69}$$

where the second-order fitting of $t_{i_{max}}$ would tend to $-\infty$ as $n \to \infty$ and is therefore invalid. Consequently, $t_{i_{max}}$ can be considered as linear in n when $\max(n_i)$ is fixed, whether the local MPCs are solved analytically or numerically, which matches the theoretical analysis.

The comparison of the single-cycle price-driven CDMPC and the centralized MPC in execution time is presented in Figure 4.6. In this scenario, the curve of t_{CEN} is always above that of t_{NonCo} and the difference between the two curves increases rapidly as n increases.

In Figure 4.7, the communication cycle numbers of every test in both Scenario 1 and 2 are shown. As predicted, all CCN are 2 in the unconstrained price-driven CDMPC, which proves the theoretical analysis of it. With such a small constant CCN, the price-driven CDMPC has advantage over the centralized MPC, especially when the MPCs are solved numerically by the interior-point method.

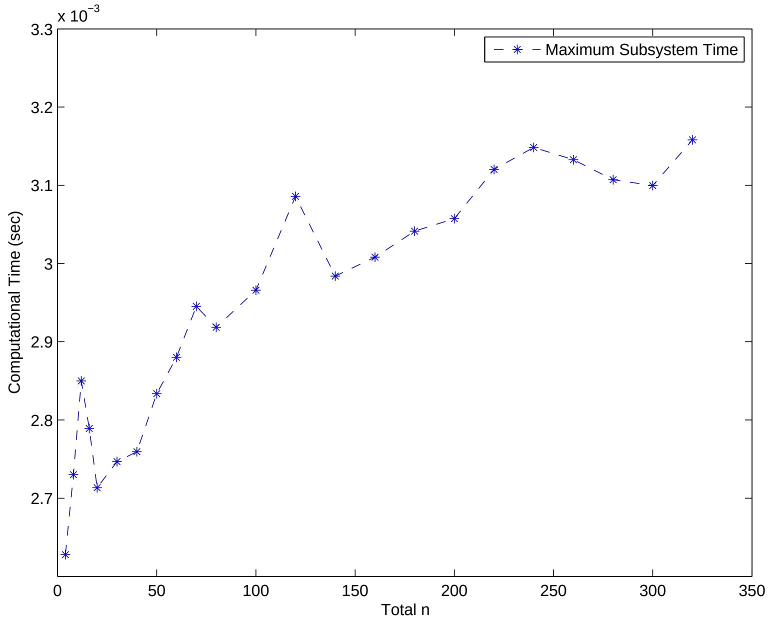


Figure 4.5: Price-driven CDMPC Experiment 1. The longest execution time of subsystems, when MPCs are solved numerically. x axis is the plant state size and y axis is the execution time in seconds.

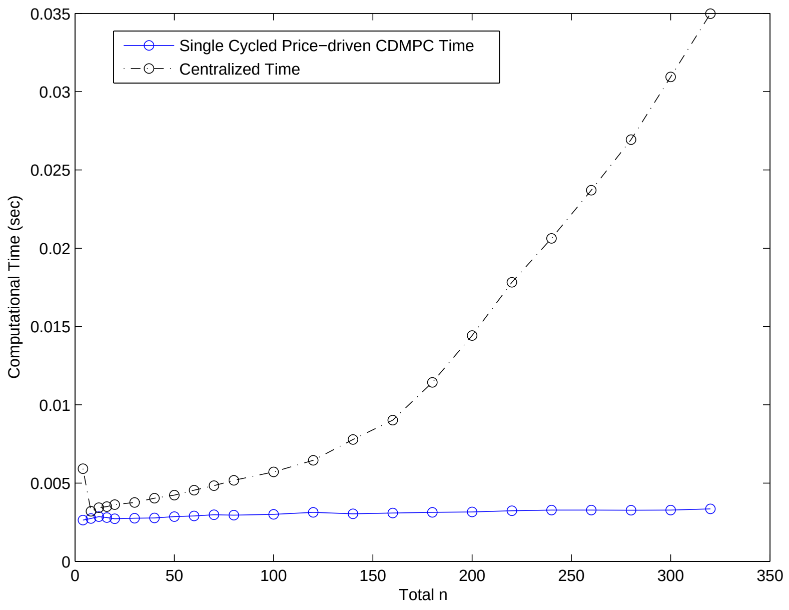


Figure 4.6: Price-driven CDMPC Experiment 1. Execution time of the prediction-driven CDMPC in a single cycle versus the execution time of centralized MPC, when MPC problems are solved numerically using the interior-point method. The solid line represents t_{NonCo} and the dash-dot line represents t_{CEN}. x axis is the plant state size and y axis is the execution time in seconds.

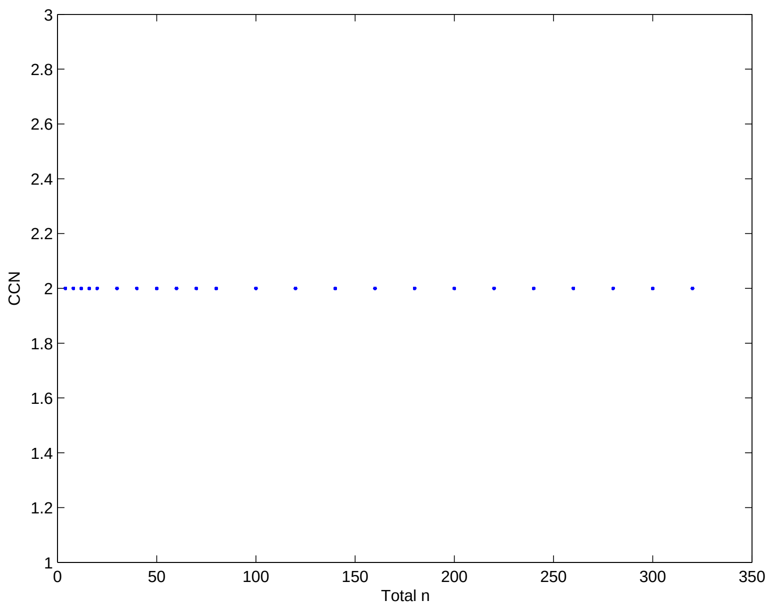


Figure 4.7: All CCNs for every repetition in Experiment 1, including all data points from Scenario 1 and Scenario 2.

Experiment 2: Empirical Complexity Varying With the Maximum Subsystem Size

This second experiment has the same settings as Experiment 2 in the empirical complexity study of the prediction-driven CDMPC, i.e., the plant sizes are fixed to be $n = q = 160$ while the maximum subsystem size varies. The sizes of the 80 subsystems are chosen sequentially from the set (3.89).

Scenario 1: The MPCs are solved analytically in Scenario 1. Figure 4.8 shows the execution time of both the single-cycled coordinator and the centralized MPC. The two curves both fluctuate without an obvious trend of increase or decrease. If they are assumed to be linear functions in $\max(n_i)$, the least-squares fitting results are:

$$t_{coor} = (0.39437 \pm 1.4755) \times 10^{-6} \max(n_i) + (4.2955 \pm 0.39076) \times 10^{-4}, \tag{4.70}$$

$$t_{CEN} = (-0.29937 \pm 2.6452) \times 10^{-6} \max(n_i) + (8.9907 \pm 7.1592) \times 10^{-5}. \tag{4.71}$$

The linear coefficient in equations (4.70) and (4.71) both contain 0 in their 95% confidence intervals. Therefore, we cannot say with 95% confidence that they are not zero. Additionally both of the fitted linear coefficients are very small so that they can almost be treated as 0. This agrees well with the theoretical analysis that changing $\max(n_i)$ will not affect the complexity of the coordinator and the centralized MPC.

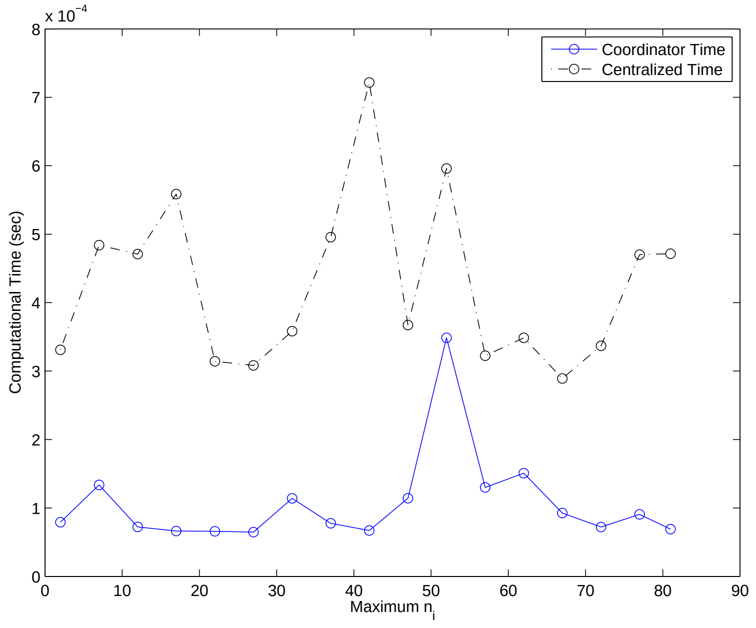


Figure 4.8: Price-driven CDMPC Experiment 2. Execution time of coordinator in one cycle versus execution time of centralized MPC, when MPCs are solved analytically. The solid line represents t_{coor} and the dash-dot line represents t_{CEN}. x axis is the maximum subsystem state size and y axis is the execution time in seconds.

The variation of $t_{i_{max}}$ as $\max(n_i)$ increases is plotted in Figure 4.9. As can be seen, $t_{i_{max}}$ shows an increasing trend as $\max(n_i)$ grows. According to the derivation presented in §4.3.1, the analytically solved local MPCs have complexity of $O(n_i n)$ and $t_{i_{max}}$ should be linear in $\max(n_i)$. We assume that $t_{i_{max}}$ is linear and quadratic functions of $\max(n_i)$, respectively, and have the least-squares fitting results:

$$\begin{aligned} t_{i_{max}} &= (2.5789 \pm 0.93023) \times 10^{-6} \max(n_i) + (2.9080 \pm 0.45136) \times 10^{-4}, \\ t_{i_{max}} &= (-0.57261 \pm 4.4884) \times 10^{-8} \max(n_i)^2 + \\ &\quad (3.0575 \pm 3.8746) \times 10^{-6} \max(n_i) + (2.8420 \pm 0.69791) \times 10^{-4}. \end{aligned} \tag{4.72}$$

Since the second-order fitting has a negative quadratic coefficient and the coefficient is very small comparing to the 95% confidence interval, $t_{i_{max}}$ should not be considered to be a quadratic function in $\max(n_i)$, statistically. Therefore, we can say that the maximum execution time of subsystems is statistically linear in $\max(n_i)$, which goes well with the hypothesis.

Scenario 2: When the MPCs are solved numerically using the interior-point method, the comparison of t_{coor} and t_{CEN} in execution time is shown in Figure 4.10. There are two different y axes in Figure 4.10, because the two curves differ by about two orders of magnitude. The curve of t_{coor} has a similar pattern as in Scenario 1, while that of t_{CEN}

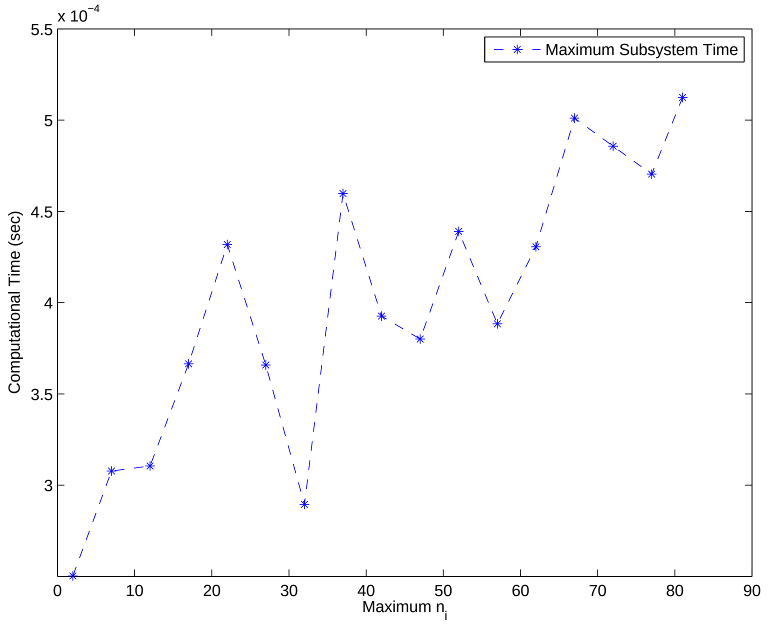


Figure 4.9: Price-driven CDMPC Experiment 2. The longest execution time of subsystems, when MPCs are solved analytically. x axis is the maximum subsystem state size and y axis is the execution time in seconds.

shows an increasing trend as $\max(n_i)$ gets bigger. The reason for this increase has been discussed in §3.3.2, which is possibly due to the 'presolve' step of the MATLAB® IPM solver.

The theoretical analysis in §4.3.1 predicted that the maximum execution time of subsystems should be a polynomial with order between 3 and 4 when local MPCs are solved numerically, using IPM. The polynomial characteristic can be seen in Figure 4.11. Attempts were made to fit $t_{i_{max}}$ as second- and third-order polynomials. The results are:

$$\begin{aligned} t_{i_{max}} =&(5.5767 \pm 1.0243) \times 10^{-7} \max(n_i)^2 + (-1.2426 \pm 0.88418) \times 10^{-5} \max(n_i) + \\ &(3.3387 \pm 0.15926) \times 10^{-3}. \end{aligned} \tag{4.73}$$

$$\begin{aligned} t_{i_{max}} =&(0.77805 \pm 5.1219) \times 10^{-9} \max(n_i)^3 + (4.6048 \pm 6.4865) \times 10^{-7} \max(n_i)^2 + \\ &(-0.92024 \pm 2.3129) \times 10^{-5} \max(n_i) + (3.3168 \pm 0.21973) \times 10^{-3}, \end{aligned} \tag{4.74}$$

The cubic fitting has 0 in the 95% confidence interval of the third-order coefficient, whose value is small compared to the size of the confidence interval. Therefore, the empirical complexity of $t_{i_{max}}$, when local MPCs are solved by IPM, is quadratic in $\max(n_i)$.

The execution time of the price-driven CDMPC is compared with that of the centralized MPC in Figure 4.12.

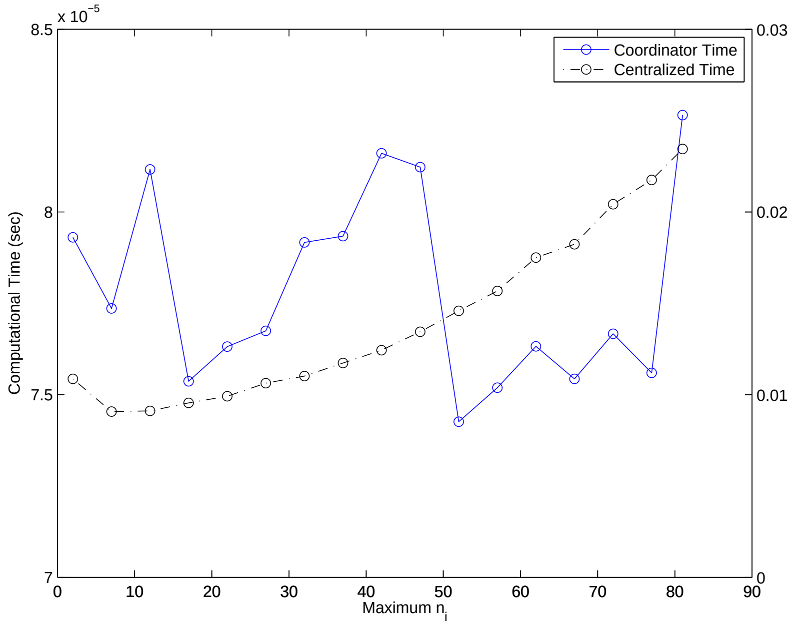


Figure 4.10: Price-driven CDMPC Experiment 2. Execution time of coordinator in one cycle versus execution time of centralized MPC, when MPCs are solved numerically using the interior-point method. t_{coor} is represented by the solid line and corresponds to y axis on the left. t_{CEN} is represented by the dash-dot line and corresponds to y axis on the right. x axis is the maximum subsystem state size.

In the end, the CCNs of each test are presented in Figure 4.13. All except 8 among 3400 tests have CCNs being 2. The 8 datapoints can be treated as outliers. Some of them are due to the floating point error of the computation machine, while some are due to the ill-conditioned subsystem matrices such as $\boldsymbol{H}_i$, $\boldsymbol{\Lambda}_i$, etc. generated in the Monte Carlo simulation. This can also explain why in (Cheng, 2007) some of the interior case CCNs have small deviation above 3, but never below 3.

4.4 Conclusions

This chapter studied the computational properties of the unconstrained price-driven CDMPC. It is an efficient method, with guaranteed convergence to the optimal solution within two communication cycles, as long as the centralized solution exists. The computational complexity of this CDMPC algorithm is polynomial in problem size. Moreover, it can reach the centralized solution faster than the centralized MPC, especially when the number of subsystems is large. The advantage is more obvious when the MPCs are solved numerically, which is the actual method used in practice. Compared with the prediction-driven CDMPC, price-driven CDMPC is a better choice for implementation, because of the guar-

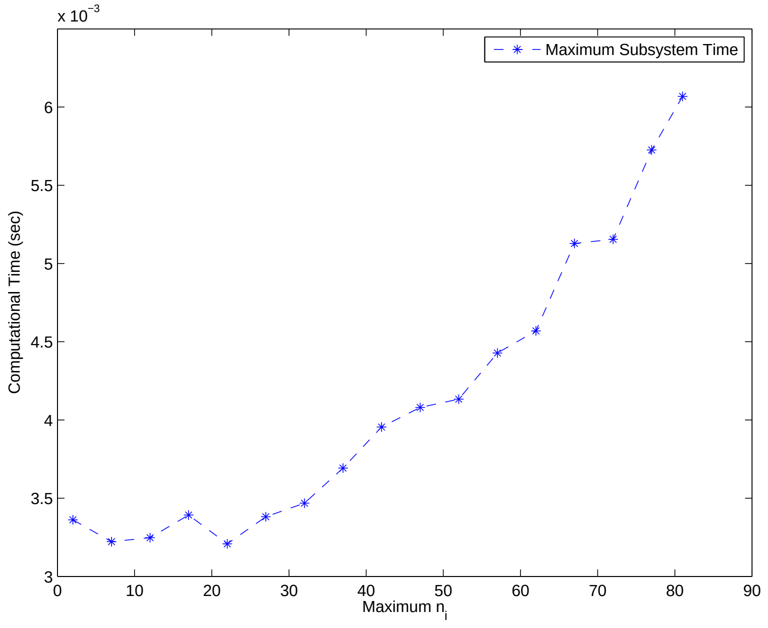


Figure 4.11: Price-driven CDMPC Experiment 2. The longest execution time of subsystems, when MPCs are solved analytically. x axis is the maximum subsystem state size and y axis is the execution time in seconds.

anteed convergence and small CCN.

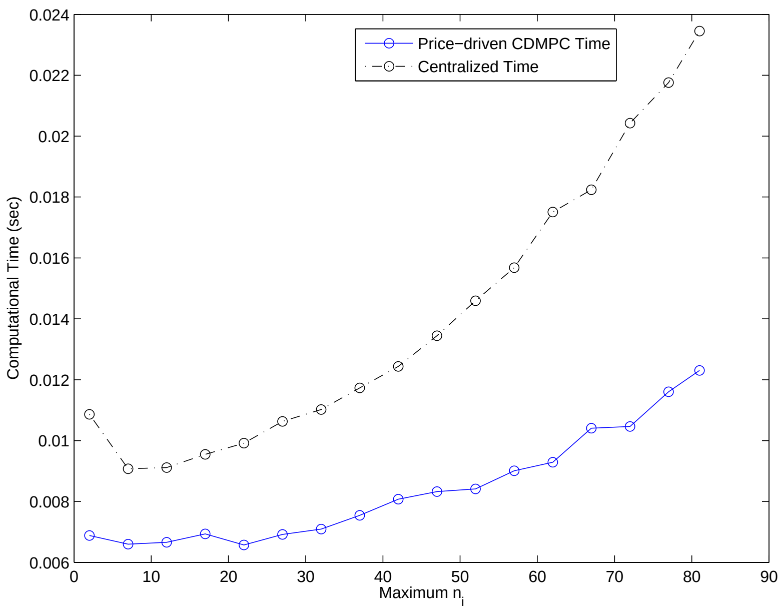


Figure 4.12: Price-driven CDMPC Experiment 2. Execution time of the prediction-driven CDMPC, including all communication cycles versus the execution time of centralized MPC, when MPC problems are solved numerically using the interior-point method. The solid line represents t_{NonCo} and the dash-dot line represents t_{CEN}. x axis is the plant state size and y axis is the execution time in seconds.

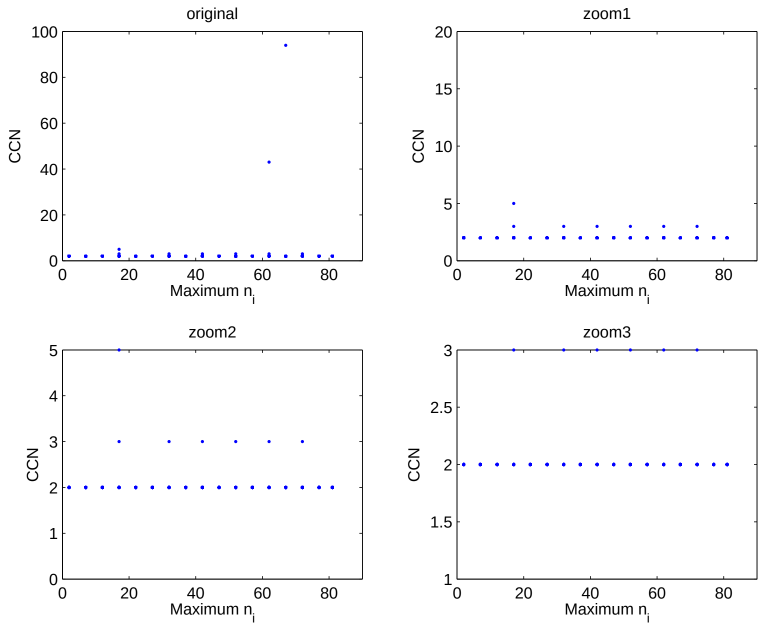


Figure 4.13: Price-driven CDMPC Experiments. All CCNs for every repetition in Experiment 2, including all data points from Scenario 1 and Scenario 2.

Chapter 5

Conclusions

5.1 Summary

The development of DMPCs is motivated by the increasing competition and the pursuit of profits in the process industry. The DMPC methods proposed in the literature seek to bring distributed control performance closer to the centralized control performance, while maintaining the flexibility of the decentralized control systems. Nonetheless, the improvement in performance often requires iterative communications between the nodes in the control network, which may result in high computation and communication costs, especially when the plant is a large-scale system with interconnected operation units. Therefore, this thesis has focused on DMPC features that have not received much prior attention.

In this work, the computational properties of two unconstrained linear coordinated, distributed MPCs, which are prediction-driven CDMPC and price-driven CDMPC, are studied. A thorough discussion including both theoretical analysis and empirical studies is provided. For both of the two CDMPC methods, the addressed properties are convergence conditions, rates of convergence and computational complexities, which are closely related to each other.

The unconstrained CDMPCs are first written as iterative functions, which are found to be linear in the coordinating variables. Whether or not a specific method would converge and its rate of convergence are controlled by the spectral radius ρ of the coordinating variable's coefficient matrix. The spectral radius of the coefficient matrix influences convergence as follows: the method will converge to the centralized MPC if and only if ρ is smaller than 1; a smaller ρ results in higher rate of convergence; and ρ affects the computational complexity of a CDMPC method, because its value would influence the number of communication cycles between the coordinator and local MPCs. If CCN is not taken into consideration, which is to say, in a single communication cycle, the two discussed CDMPCs both have polynomial complexity. When MPCs are numerically solved, they both present computational advantages over the numerically solved, centralized MPC. Despite this, the overall complexity of the two CDMPCs have significant difference due to CCN behavior. The price-driven CDMPC has a fixed CCN, so its complexity remains to be polynomial and it has a reasonable increase in the rate of computational load with the growth of problem size. On the contrary, the prediction-driven CDMPC has an indefinite CCN which can only be estimated and whose upper bound is unknown. Empirical study shows that when the problem size grows, there is a higher chance of encountering a system with very large CCN. In this case, the computational load of the CDMPC method is so heavy that the method may be impractical.

From the results obtained in this thesis, it is recommended that the price-driven method should be considered prior to the prediction-driven method, as the former one has a guaranteed covergence. If the prediction-driven method needs to be implemented, the convergence condition can be checked and the required CCN be estimated, which would provide a sense of how much computational load is needed before implementation.

5.2 Directions for Future Work

Although this thesis provides a thorough discussion on the convergence and complexity of two CDMPC methods, more challenges remain in the study of this topic.

From a theoretical perspecive, an immediate question arises for the prediction-driven CDMPC method: is it possible to reduce CCN so that overall computational load can be reduced as well? It is observed from Figure 3.1 that when $\rho(\mathbf{C_2})$ increases, CCN is very likely to increase. In the estimation of CCN, (3.75) indicates that a large $\rho(\mathbf{C_2})$ may result in a large CCN. Therefore, one possible way to achieve the goal of reducing computational load is to minimize $\rho(\mathbf{C_2})$, where $\mathbf{C_2}$, as stated in §3.2.1, is only determined by the properties of the plant and the design of local MPC controllers. The parameters of $\mathbf{C_2}$ are listed in (3.35). While $\boldsymbol{A}$ and $\boldsymbol{B}$ cannot be changed, the rest are MPC tuning parameters and can be adjusted to vary the value of $\rho(\mathbf{C_2})$. This implies that we can write an optimization problem of the following form:

$$\min_{\boldsymbol{Q},\boldsymbol{R},H_p,H_u} \rho(\mathbf{C_2}), \tag{5.1}$$

and get the smallest possible spectral radius of $\mathbf{C_2}$. Nevertheless, the objective function ρ is not continuous in H_p and H_u, and the mapping from the matrix variables $\boldsymbol{Q}$ and $\boldsymbol{R}$ to the scalar ρ is very complex. For these reasons, an extra in-depth study is needed to solve optizimization problem (5.1). The optimization would change the parameters in local MPC controllers, indicating that a trade-off between MPC performance and computational load may exist.

It is also of interest as to how the complexity varies with the 'strength' of interaction between subsystems, which is an immediate follow-up of the empirical computational study in the thesis. The biggest challenge in this study is how to measure the 'strength' of interaction, which is another topic that requires further research.

Besides the above two issues, the following extensions of the work are recommended as the possible direction of further research:

- As can be noticed, the research scope of this thesis is limited to unconstrained linear systems. This restriction leaves out the computational properties of the more general systems and makes them open problems. It is recommended that the scope of the research be expanded to the constrained linear systems first, and possibly further to the nonlinear systems.

- The scope of time in this thesis is limited within a control interval, which is assumed to be long enough for convergence. This is of course an idealized situation. If the result is to be put to application, the computational properties over a control time period should be studied as well, where only a finite number of communication cycles are allowed within each control interval. Analyzing approaches similar to the stability analyses in the literature may be applied. It is expected that the computational properties studied in this thesis will affect the properties over a control time period.

- Subsystems in this thesis are all synchronized with each other using a same sampling time. In a real plant, operation units may have very different rates so that a single sampling time is not suitable for every local process. For this reason, the computational properties for asynchronous subsystems is worth further investigation.

- The thesis only discusses the computational properties of two coordinated DMPC. As discussed in Chapter 1, a considerable amount of DMPC techniques have been developed in the past decade. The computational properties of other DMPC methods are also of interest. When a number of DMPC methods can be applied to a control network, knowing the computational loads of different DMPC methods can help us to choose from the candidates.

- It is assumed throughout the thesis that all state measurements are availale in the control network at every sampling time, which is often not true. In industrial practice, some process variables have to be measured off-line and thus, values cannot be obtained at each sampling time (a well-known example is the product concentration). Also, some process variables do not have measurements available. These are the situations that observers are required to estimate the states, which may also affect the DMPC algorithm's computational properties.

Appendix A

Proofs of Matrix Invertibility

This chapter provides proofs of invertibility of the following matrices: $\boldsymbol{G}_A$, which was defined in §2.4; $\boldsymbol{G}\boldsymbol{G}^T$, which was defined in §3.1; $\boldsymbol{W}$ and $\boldsymbol{\Psi}$, which were defined in §3.2; $\boldsymbol{W}_i$ and $\boldsymbol{\Psi}_i$, which were defined in §3.3; $\boldsymbol{\Lambda}_i$, which was defined in §4.1; and $\boldsymbol{\Lambda}$ and $\boldsymbol{H}$, which were defined in §4.2.

Prior to the proofs, the term **Schur complement** and some related lemmas are introduced, which will be applied in most of the invertibility proofs in this chapter. Assume that a square matrix $\boldsymbol{M} \in \mathbb{R}^{(l+m)\times(l+m)}$ is partitioned into the following four blocks:

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{S} & \boldsymbol{T} \\ \boldsymbol{U} & \boldsymbol{V} \end{bmatrix}, \tag{A.1}$$

where $\boldsymbol{S} \in \mathbb{R}^{l\times l}$, $\boldsymbol{T} \in \mathbb{R}^{l\times m}$, $\boldsymbol{U} \in \mathbb{R}^{m\times l}$ and $\boldsymbol{V} \in \mathbb{R}^{m\times m}$. If the square matrix $\boldsymbol{S}$ is nonsingular, then:

$$\boldsymbol{M}/\boldsymbol{S} \triangleq \boldsymbol{V} - \boldsymbol{U}\boldsymbol{S}^{-1}\boldsymbol{T} \tag{A.2}$$

is defined to be the **Schur complement of $\boldsymbol{M}$ relative in $\boldsymbol{S}$**. Similarly, if the square matrix $\boldsymbol{V}$ is nonsingular, the **Schur complement of $\boldsymbol{M}$ relative in $\boldsymbol{V}$** is defined as:

$$\boldsymbol{M}/\boldsymbol{V} \triangleq \boldsymbol{S} - \boldsymbol{T}\boldsymbol{V}^{-1}\boldsymbol{U}. \tag{A.3}$$

The Schur complement is a very useful tool in matrix analysis. A commonly used property of Schur complement is presented in **Lemma A.0.1**:

Lemma A.0.1. (Schur's Formula (Zhang, 2005)**)** *Let $\boldsymbol{M} \in \mathbb{R}^{(l+m)\times(l+m)}$ be a square matrix partitioned as in equation (A.1). If $\boldsymbol{S}$ is nonsingular, then:*

$$\det(\boldsymbol{M}) = \det(\boldsymbol{S})\det(\boldsymbol{M}/\boldsymbol{S}). \tag{A.4}$$

Similarly, if $\boldsymbol{V}$ is nonsingular, there is:

$$\det(\boldsymbol{M}) = \det(\boldsymbol{V})\det(\boldsymbol{M}/\boldsymbol{V}). \tag{A.5}$$

When $\boldsymbol{S}$ ($\boldsymbol{V}$) is nonsingular, there is $\det(\boldsymbol{S}) \neq 0$ ($\det(\boldsymbol{V}) \neq 0$). Then in equation (A.4) (equation (A.5)), $\det(\boldsymbol{M}) \neq 0$ is equivalent to $\det(\boldsymbol{M}/\boldsymbol{S}) \neq 0$ ($\det(\boldsymbol{M}/\boldsymbol{V}) \neq 0$), which leads to **Corollary A.0.1**:

Corollary A.0.1. *For the partitioned matrix $\boldsymbol{M}$ as in (A.1), if the block $\boldsymbol{S}$ ($\boldsymbol{V}$) is nonsingular, then $\boldsymbol{M}$ is invertible if and only if the Schur complement $\boldsymbol{M}/\boldsymbol{S}$ ($\boldsymbol{M}/\boldsymbol{V}$) is invertible.*

A.1 The Invertibility of $\boldsymbol{G}_A$

The matrix $\boldsymbol{G}_A$ is a block-wise matrix defined as:

$$
\begin{aligned}
\boldsymbol{G}_A &= \begin{bmatrix} \boldsymbol{G}_{A_{11}} & \boldsymbol{G}_{A_{12}} & \cdots & \boldsymbol{G}_{A_{1N}} \\ \boldsymbol{G}_{A_{21}} & \boldsymbol{G}_{A_{22}} & \cdots & \boldsymbol{G}_{A_{2N}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{G}_{A_{1N}} & \boldsymbol{G}_{A_{2N}} & \cdots & \boldsymbol{G}_{A_{NN}} \end{bmatrix} \\
&= \left[\begin{array}{cccc|cccc|c|cccc}
\boldsymbol{I}_{n_1} & & & & & & & & & & & & \\
-\boldsymbol{A}_{11} & \boldsymbol{I}_{n_1} & & & -\boldsymbol{A}_{12} & & & & & -\boldsymbol{A}_{1N} & & & \\
& \ddots & \ddots & & & \ddots & & & \cdots & & \ddots & & \\
& & -\boldsymbol{A}_{11} & \boldsymbol{I}_{n_1} & & & -\boldsymbol{A}_{12} & & & & & -\boldsymbol{A}_{1N} & \\
\hline
& & & & \boldsymbol{I}_{n_2} & & & & & & & & \\
-\boldsymbol{A}_{21} & & & & -\boldsymbol{A}_{22} & \boldsymbol{I}_{n_2} & & & & -\boldsymbol{A}_{2N} & & & \\
& \ddots & & & & \ddots & \ddots & & \cdots & & \ddots & & \\
& & -\boldsymbol{A}_{21} & & & & -\boldsymbol{A}_{22} & \boldsymbol{I}_{n_2} & & & & -\boldsymbol{A}_{2N} & \\
\hline
& \vdots & & & & \vdots & & & \ddots & & \vdots & & \\
\hline
& & & & & & & & & \boldsymbol{I}_{n_N} & & & \\
-\boldsymbol{A}_{N1} & & & & -\boldsymbol{A}_{N2} & & & & & -\boldsymbol{A}_{NN} & \boldsymbol{I}_{n_N} & & \\
& \ddots & & & & \ddots & & & & & \ddots & \ddots & \\
& & -\boldsymbol{A}_{N1} & & & & -\boldsymbol{A}_{N2} & & & & & -\boldsymbol{A}_{NN} & \boldsymbol{I}_{n_N}
\end{array}\right].
\end{aligned}
\tag{A.6}
$$

Note that the blocks of $\boldsymbol{G}_A$ ($\boldsymbol{G}_{A_{ii}}$'s and $\boldsymbol{G}_{A_{ij}}$'s, where $i, j = 1, ..., N$ and $i \neq j$) are themselves block matrices, with all of the non-zero elements on the lower half. The diagonal blocks of $\boldsymbol{G}_A$, i.e., $\boldsymbol{G}_{A_{ii}}$'s, are **unit lower triangular matrices**[1]. Although $\boldsymbol{G}_{A_{ij}}$'s are not square matrices, but the $H_p \times H_p$ blocks of them have a clear lower triangular structure

[1]A lower triangular matrix is a square matrix in which all elements above the diagonal are zero. For example, let $\boldsymbol{P} \in \mathbb{R}^{l \times l}$ be a lower triangular matrix and $p_{\iota\kappa}$, $\iota, \kappa = 1, ..., l$ be its entries, then $p_{\iota\kappa} = 0, \forall \iota < \kappa$. A unit lower triangular matrix is a lower triangular matrix with all diagonal elements being 1.(Horn and Johnson, 2012)

where all the diagonal blocks are zero matrices. We define such matrices to be **strictly block lower triangular matrices**:

Definition A.1.1. *Assume a $l \times m$ block matrix $\boldsymbol{L}$ has $K \times K$ blocks and has the following structure:*

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{J}_{11} & \boldsymbol{J}_{12} & \cdots & \boldsymbol{J}_{1K} \\ \boldsymbol{J}_{21} & \boldsymbol{J}_{22} & \cdots & \boldsymbol{J}_{21} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{J}_{K1} & \boldsymbol{J}_{K2} & \cdots & \boldsymbol{J}_{KK} \end{bmatrix}, \tag{A.7}$$

where $\boldsymbol{J}_{ij} \in \mathbb{R}^{l_i \times m_j}, i, j = 1, ...K$ and $\boldsymbol{J}_{ij} = \boldsymbol{O}$ for all $i \geq j$. Such a matrix is defined as a strictly block lower triangular matrix.

To prove that $\boldsymbol{G}_A$ is invertible, we first prove the invertibility of the matrix defined in equation (A.8), which has the same structure as $\boldsymbol{G}_A$. The proof is built on properties of unit lower triangular matrices and of strict lower block triangular matrices, **Corollary A.0.1** and mathematical induction.

Assume that a block matrix $\boldsymbol{BM} \in \mathbb{R}^{m \times m}$ has the following structure:

$$\boldsymbol{BM} = \begin{bmatrix} \boldsymbol{P}_{11} & \boldsymbol{L}_{12} & \cdots & \boldsymbol{L}_{1N} \\ \boldsymbol{L}_{21} & \boldsymbol{P}_{22} & \cdots & \boldsymbol{L}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{L}_{N1} & \boldsymbol{L}_{N2} & \cdots & \boldsymbol{P}_{NN} \end{bmatrix}, \tag{A.8}$$

where $\boldsymbol{P}_{ii}, i = 1, ..., N$ are unit lower triangular matrices, and $\boldsymbol{L}_{ij}, i, j = 1, ...N$, $i \neq j$ are strictly block lower triangular matrices with $S \times S$ blocks. It is further assumed that: a) $\boldsymbol{L}_{ij}$ (or $\boldsymbol{P}_{ii}$ if $j = i$) and $\boldsymbol{L}_{ik}$ (or $\boldsymbol{P}_{ii}$ if $k = i$) have the same partition in the row, and $\boldsymbol{L}_{ij}$ (or $\boldsymbol{P}_{jj}$ if $i = j$) and $\boldsymbol{L}_{kj}$ (or $\boldsymbol{P}_{jj}$ if $k = j$) have the same partition in the column, $\forall i, j, k = 1, ..., N$; and b) the α, β^{th} block in $\boldsymbol{L}_{ij}$, denoted as $\boldsymbol{J}_{ij_{\alpha\beta}}$ has the same number of columns as the number of rows in $\boldsymbol{J}_{ji_{\beta\gamma}}$, which is the β, α^{th} block in $\boldsymbol{L}_{ji}$. For example, if the dimension of $\boldsymbol{J}_{ij_{\alpha\beta}}$ is $l_{ij_{(\alpha)}} \times m_{ij_{(\beta)}}$ and the dimension of $\boldsymbol{J}_{ji_{\beta\gamma}}$ is $l_{ji_{(\beta)}} \times m_{ji_{(\gamma)}}$, then there is $m_{ij_{(\beta)}} = l_{ji_{(\beta)}}$. If assumptions b) is satisfied, we say that $\boldsymbol{L}_{ij}$ and $\boldsymbol{L}_{jk}$ are **compatible** with each other, $\forall i, j, k = 1, ..., N$, $i \neq j, j \neq k$. Note that assumption a) and b) also indicate that $\boldsymbol{P}_{ii}$ is compatible with $\boldsymbol{L}_{ij}$ and $\boldsymbol{L}_{ki}$, $\forall i, j, k = 1, ..., N$, $i \neq j, i \neq k$. The product $\boldsymbol{L}_{ij}\boldsymbol{L}_{jk}$ has the same structure as $\boldsymbol{L}_{ik}$ if $i \neq k$. If $i = k$, the product $\boldsymbol{L}_{ij}\boldsymbol{L}_{ji}$ will be a square matrix and is strictly lower triangular[2].

To illustrate the above matrix structure more clearly, let $\boldsymbol{M} \in \mathbb{R}^{m \times m}$ be a matrix following the structure in (A.8), with 3×3 blocks and each block is itself a 2×2 block

[2]A **strictly lower triangular matrix** is a lower triangular matrix with all diagonal elements being 0.(Horn and Johnson, 2012)

matrix:

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{P}_{11} & \boldsymbol{L}_{12} & \boldsymbol{L}_{13} \\ \boldsymbol{L}_{21} & \boldsymbol{P}_{22} & \boldsymbol{L}_{23} \\ \boldsymbol{L}_{31} & \boldsymbol{L}_{32} & \boldsymbol{P}_{33} \end{bmatrix}$$

$$= \begin{array}{c} \begin{array}{cccccc} m_{11_{(1)}} & m_{11_{(2)}} & m_{12_{(1)}} & m_{12_{(2)}} & m_{13_{(1)}} & m_{13_{(2)}} \end{array} \\ \left[\begin{array}{cc:cc:cc} \boldsymbol{K}_{11_{11}} & \boldsymbol{K}_{11_{12}} & \boldsymbol{J}_{12_{11}} & \boldsymbol{J}_{12_{12}} & \boldsymbol{J}_{13_{11}} & \boldsymbol{J}_{13_{12}} \\ \boldsymbol{K}_{11_{21}} & \boldsymbol{K}_{11_{22}} & \boldsymbol{J}_{12_{21}} & \boldsymbol{J}_{12_{22}} & \boldsymbol{J}_{13_{21}} & \boldsymbol{J}_{13_{22}} \\ \hdashline \boldsymbol{J}_{21_{11}} & \boldsymbol{J}_{21_{12}} & \boldsymbol{K}_{22_{11}} & \boldsymbol{K}_{22_{12}} & \boldsymbol{J}_{23_{11}} & \boldsymbol{J}_{23_{12}} \\ \boldsymbol{J}_{21_{21}} & \boldsymbol{J}_{21_{22}} & \boldsymbol{K}_{22_{21}} & \boldsymbol{K}_{22_{22}} & \boldsymbol{J}_{23_{21}} & \boldsymbol{J}_{23_{22}} \\ \hdashline \boldsymbol{J}_{31_{11}} & \boldsymbol{J}_{31_{12}} & \boldsymbol{J}_{32_{11}} & \boldsymbol{J}_{32_{12}} & \boldsymbol{K}_{33_{11}} & \boldsymbol{K}_{33_{12}} \\ \boldsymbol{J}_{31_{21}} & \boldsymbol{J}_{31_{22}} & \boldsymbol{J}_{32_{21}} & \boldsymbol{J}_{32_{22}} & \boldsymbol{K}_{33_{21}} & \boldsymbol{K}_{33_{22}} \end{array}\right] \end{array} \begin{array}{c} \\ l_{11_{(1)}} \\ l_{11_{(2)}} \\ l_{21_{(1)}} \\ l_{21_{(2)}} \\ l_{31_{(1)}} \\ l_{31_{(2)}} \end{array} . \tag{A.9}$$

The dimensions of the 36 small blocks $\boldsymbol{J}_{ij_{\alpha\beta}}$ and $\boldsymbol{K}_{ii_{\alpha\beta}}$, $i \neq j, i, j = 1, 2, 3, \alpha, \beta = 1, 2$, have the following relationship and are re-labeled as:

$$p_1 = l_{11_{(1)}} = l_{12_{(1)}} = l_{13_{(1)}} = m_{11_{(1)}} = m_{21_{(1)}} = m_{31_{(1)}},$$

$$p_2 = l_{11_{(2)}} = l_{12_{(2)}} = l_{13_{(2)}} = m_{11_{(2)}} = m_{21_{(2)}} = m_{31_{(2)}},$$

$$p_3 = l_{21_{(1)}} = l_{22_{(1)}} = l_{23_{(1)}} = m_{12_{(1)}} = m_{22_{(1)}} = m_{32_{(1)}},$$

$$p_4 = l_{21_{(2)}} = l_{22_{(2)}} = l_{23_{(2)}} = m_{12_{(2)}} = m_{22_{(2)}} = m_{32_{(2)}},$$

$$p_5 = l_{31_{(1)}} = l_{32_{(1)}} = l_{33_{(1)}} = m_{13_{(1)}} = m_{23_{(1)}} = m_{33_{(1)}},$$

$$p_6 = l_{31_{(2)}} = l_{32_{(2)}} = l_{33_{(2)}} = m_{13_{(2)}} = m_{23_{(2)}} = m_{33_{(2)}}.$$

If $\boldsymbol{L}_{31}$ is multiplied by $\boldsymbol{L}_{12}$, according to block matrix multiplication rule (Anton, 2006), there is:

$$\begin{aligned} \boldsymbol{L}_{31}\boldsymbol{L}_{12} &= \begin{bmatrix} \sum_{i=1}^{2} \boldsymbol{L}_{31_{1i}}\boldsymbol{L}_{12_{i1}} & \sum_{i=1}^{2} \boldsymbol{L}_{31_{1i}}\boldsymbol{L}_{12_{i2}} \\ \sum_{i=1}^{2} \boldsymbol{L}_{31_{2i}}\boldsymbol{L}_{12_{i1}} & \sum_{i=1}^{2} \boldsymbol{L}_{31_{2i}}\boldsymbol{L}_{12_{i2}} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{O}_{p_5\times p_3} & \boldsymbol{O}_{p_5\times p_4} \\ \boldsymbol{J}_{p_6\times p_3} & \boldsymbol{O}_{p_6\times p_4} \end{bmatrix}, \end{aligned}$$

whose structure is exactly the same as $\boldsymbol{L}_{32}$. If $\boldsymbol{L}_{21}$ is multiplied by $\boldsymbol{L}_{12}$, following the same derivation steps, there would be

$$\boldsymbol{L}_{21}\boldsymbol{L}_{12} = \begin{bmatrix} \boldsymbol{O}_{p_3\times p_3} & \boldsymbol{O}_{p_3\times p_4} \\ \boldsymbol{J}_{p_4\times p_3} & \boldsymbol{O}_{p_4\times p_4}, \end{bmatrix}$$

which is a strictly lower triangular matrix having the same dimension as $\boldsymbol{P}_{22}$.

Lemma A.1.1 and **Lemma A.1.2** are required to prove the invertibility of the block matrix $\boldsymbol{BM}$ defined in equation (A.8).

Lemma A.1.1. *The product of a unit lower triangular matrix multiplied by a strictly block lower triangular matrix is a strictly block lower triangular matrix, which has the same block-structure as the original strictly block lower triangular matrix.*

Proof Assume that a strictly block lower triangular matrix $\boldsymbol{L} \in \mathbb{R}^{m \times l}$ has $K \times K$ blocks as follows:

$$\begin{aligned} \boldsymbol{L} &= \left[\begin{array}{c:c:c:c} \boldsymbol{O}_{\boldsymbol{L}_{11}} & & & \\ \boldsymbol{J}_{21} & \boldsymbol{O}_{\boldsymbol{L}_{22}} & & \\ \vdots & \vdots & \ddots & \\ \boldsymbol{J}_{K1} & \boldsymbol{J}_{K2} & \cdots & \boldsymbol{O}_{\boldsymbol{L}_{KK}} \end{array}\right] \\ &\triangleq \left[\begin{array}{c:c:c:c} \boldsymbol{\Upsilon}_1 & \boldsymbol{\Upsilon}_2 & \cdots & \boldsymbol{\Upsilon}_K \end{array}\right], \end{aligned} \tag{A.10}$$

where $\boldsymbol{J}_{ij}, i, j = 1, ..., K$ and $i > j$ are $m_i \times l_j$ matrices. $\boldsymbol{O}_{\boldsymbol{L}_{ij}}$ denotes that the i, j^{th} block is a zero matrix, of which the dimension is $m_i \times l_j, i \leq j$. Let $\boldsymbol{P}$ be a $m \times m$ unit lower triangular matrix and partition it into:

$$\begin{aligned} \boldsymbol{P} &= \left[\begin{array}{cccc} \boldsymbol{P}_{11} & & & \\ \hdashline \boldsymbol{K}_{21} & \boldsymbol{P}_{22} & & \\ \hdashline \vdots & \vdots & \ddots & \\ \hdashline \boldsymbol{K}_{K1} & \boldsymbol{K}_{K2} & & \boldsymbol{P}_{KK} \end{array}\right] \\ &\triangleq \left[\begin{array}{c} \boldsymbol{\Gamma}_1^T \\ \hdashline \boldsymbol{\Gamma}_2^T \\ \hdashline \vdots \\ \hdashline \boldsymbol{\Gamma}_K^T \end{array}\right], \end{aligned} \tag{A.11}$$

where $\boldsymbol{P}_{ii} \in \mathbb{R}^{m_i \times m_i}$ are also unit lower triangular matrices and $\boldsymbol{K}_{ij} \in \mathbb{R}^{m_i \times l_j}, i, j = 1, ..., K, i > j$. The i, j^{th} block of $\boldsymbol{P}$, if it is a zero matrix, is denoted as $\boldsymbol{O}_{\boldsymbol{P}_{ij}}$. The dimension of $\boldsymbol{O}_{\boldsymbol{P}_{ij}}$ is $m_i \times l_j, i < j$.

The product of $\boldsymbol{P}$ multiplied by $\boldsymbol{L}$ can be represented as

$$\boldsymbol{T} = \boldsymbol{P}\boldsymbol{L} = \begin{bmatrix} \boldsymbol{T}_{11} & \boldsymbol{T}_{12} & \cdots & \boldsymbol{T}_{1K} \\ \boldsymbol{T}_{21} & \boldsymbol{T}_{22} & \cdots & \boldsymbol{T}_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{T}_{K1} & \boldsymbol{T}_{K2} & \cdots & \boldsymbol{T}_{KK} \end{bmatrix}, \tag{A.12}$$

where

$$\boldsymbol{T}_{\alpha\beta} = \boldsymbol{\Gamma}_\alpha^T \boldsymbol{\Upsilon}_\beta. \tag{A.13}$$

When $\beta \geq \alpha$, there is

$$\begin{aligned}
\boldsymbol{T}_{\alpha\beta} &= \left[\begin{array}{ccccccc} \boldsymbol{K}_{\alpha 1} & \cdots & \boldsymbol{P}_{\alpha\alpha} & \boldsymbol{O}_{\boldsymbol{P}_{\alpha,\alpha+1}} & \cdots & \boldsymbol{O}_{\boldsymbol{P}_{\alpha K}} \end{array} \right] \left[\begin{array}{c} \boldsymbol{O}_{\boldsymbol{L}_{1\beta}} \\ \vdots \\ \boldsymbol{O}_{\boldsymbol{L}_{\beta\beta}} \\ \boldsymbol{J}_{\beta+1,\beta} \\ \vdots \\ \boldsymbol{J}_{K\beta} \end{array} \right] \\
&= \sum_{\gamma=1}^{\alpha-1} \boldsymbol{K}_{\alpha\gamma} \boldsymbol{O}_{\boldsymbol{L}_{\gamma\beta}} + \boldsymbol{P}_{\alpha\alpha} \boldsymbol{O}_{\boldsymbol{L}_{\alpha\beta}} + \sum_{\gamma=\beta}^{K} \boldsymbol{O}_{\boldsymbol{P}_{\alpha\gamma}} \boldsymbol{J}_{\gamma\beta} \\
&= \boldsymbol{O}_{m_\alpha \times l_\beta},
\end{aligned} \tag{A.14}$$

i.e., the matrix $\boldsymbol{T}$ has a following strictly block lower triangular structure:

$$\begin{aligned}
\boldsymbol{T} &= \left[\begin{array}{cccc} \boldsymbol{O}_{m_1 \times l_1} & & & \\ \boldsymbol{T}_{21} & \boldsymbol{O}_{m_2 \times l_2} & & \\ \vdots & \vdots & \ddots & \\ \boldsymbol{T}_{K1} & \boldsymbol{T}_{K2} & \cdots & \boldsymbol{O}_{m_K \times l_K} \end{array} \right] \\
&= \left[\begin{array}{cccc} \boldsymbol{O}_{\boldsymbol{L}_{11}} & & & \\ \boldsymbol{T}_{21} & \boldsymbol{O}_{\boldsymbol{L}_{22}} & & \\ \vdots & \vdots & \ddots & \\ \boldsymbol{T}_{K1} & \boldsymbol{T}_{K2} & \cdots & \boldsymbol{O}_{\boldsymbol{L}_{KK}} \end{array} \right]
\end{aligned} \tag{A.15}$$

which proves the claim in **Lemma A.1.1**. □

Lemma A.1.2. *The inverse of a unit lower triangular matrix exists and is also a unit lower triangular matrix.*

Proposition A.1.1 and **Proposition A.1.2** are introduced to help us prove **Lemma A.1.2**.

Proposition A.1.1. *The determinant of a triangular matrix (lower, upper or diagonal) is the product of its diagonal elements.*

For example, if $\boldsymbol{U}$ is a $p \times p$ upper triangular matrix defined as follows:

$$\boldsymbol{U} = \left[\begin{array}{cccc} u_{11} & u_{12} & \cdots & u_{1p} \\ & u_{22} & \cdots & u_{2p} \\ & & \ddots & \vdots \\ & & & u_{pp} \end{array} \right], \tag{A.16}$$

its determinant would be $\det(\boldsymbol{U}) = \prod_{i=1}^{p} u_{ii}$.

Proposition A.1.2. *The inverse of an invertible lower (upper) triangular matrix is also a lower (upper) triangular matrix*

For example, for the $\boldsymbol{U}$ matrix defined in A.16, if it is invertible, $\boldsymbol{U}^{-1}$ would be

$$\boldsymbol{U}^{-1} = \begin{bmatrix} \tilde{u}_{11} & \tilde{u}_{12} & \cdots & \tilde{u}_{1p} \\ & \tilde{u}_{22} & \cdots & \tilde{u}_{2p} \\ & & \ddots & \vdots \\ & & & \tilde{u}_{pp} \end{bmatrix}. \tag{A.17}$$

The statements in **Proposition A.1.1** and **Proposition A.1.2** are so well-known that they can be found in many linear algebra textbooks such as Larson and Falvo, 2012 and Anton, 2006. Therefore we will use them directly.

The readers also need to know **Cramer's rule** and the definition of cofactor matrices to understand the proof of **Lemma A.1.2**.

Definition A.1.2. *The* ***cofactor matrix*** *of an $l \times l$ matrix*

$$\boldsymbol{T} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1l} \\ t_{21} & t_{22} & \cdots & t_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ t_{l1} & t_{l2} & \cdots & t_{ll} \end{bmatrix}, \tag{A.18}$$

is defined as

$$\operatorname{cof}(\boldsymbol{T}) = \begin{bmatrix} \operatorname{cof}(\boldsymbol{T})_{11} & \operatorname{cof}(\boldsymbol{T})_{12} & \cdots & \operatorname{cof}(\boldsymbol{T})_{1l} \\ \operatorname{cof}(\boldsymbol{T})_{21} & \operatorname{cof}(\boldsymbol{T})_{22} & \cdots & \operatorname{cof}(\boldsymbol{T})_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ \operatorname{cof}(\boldsymbol{T})_{l1} & \operatorname{cof}(\boldsymbol{T})_{l2} & \cdots & \operatorname{cof}(\boldsymbol{T})_{ll} \end{bmatrix}, \tag{A.19}$$

where, $\operatorname{cof}(\boldsymbol{T})_{ij}$ is called the ***$\boldsymbol{i,j}^{th}$ cofactor*** *of $\boldsymbol{T}$. $\operatorname{cof}(\boldsymbol{T})_{ij}$ is $(-1)^{i+j}$ multiplied by the determinant of $\boldsymbol{T}$'s i,j^{th} minor, which is a submatrix of $\boldsymbol{T}$ formed by deleting $\boldsymbol{T}$'s i^{th} row and j^{th} column. The mathematical expression of the i,j^{th} cofactor of $\boldsymbol{T}$ is:*

$$\operatorname{cof}(\boldsymbol{T})_{ij} = \det\left(\begin{bmatrix} t_{11} & \cdots & t_{1,j-1} & t_{1,j+1} & \cdots & t_{1l} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ t_{i-1,1} & \cdots & t_{i-1,j-1} & t_{i-1,j+1} & \cdots & t_{i-1,l} \\ t_{i+1,1} & \cdots & t_{i+1,j-1} & t_{i+1,j+1} & \cdots & t_{i+1,l} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ t_{l1} & \cdots & t_{l,j-1} & t_{l,j+1} & \cdots & t_{ll} \end{bmatrix}\right) \tag{A.20}$$

Proposition A.1.3. (Cramer's rule). *The inverse of a matrix $\boldsymbol{M}$ can be calculated by*

$$\boldsymbol{M}^{-1} = \frac{1}{\det(\boldsymbol{M})} \times \operatorname{cof}(\boldsymbol{M})^T. \tag{A.21}$$

With all the required propositions prepared above, we are ready for the proof of **Lemma A.1.2**.

Proof Assume $\boldsymbol{P}$ is a $m \times m$ unit lower matrix defined as follows:

$$\boldsymbol{P} = \begin{bmatrix} 1 & & & \\ p_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ p_{m1} & p_{m2} & \cdots & 1 \end{bmatrix}. \tag{A.22}$$

Since $\boldsymbol{P}$ is a triangular matrix, its determinant can be calculated by:

$$\det(\boldsymbol{P}) = \prod_{i=1}^{N} p_{ii} = \prod_{i=1}^{N} 1 = 1 \neq 0, \tag{A.23}$$

indicating that $\boldsymbol{P}$ is invertible. According to **Proposition A.1.2**, $\boldsymbol{P}^{-1}$ is a lower triangular matrix. If all of its diagonal elements are 1, $\boldsymbol{P}^{-1}$ is a unit lower triangular matrix .

According to Cramer's rule, the inverse of matrix $\boldsymbol{P}$ is:

$$\boldsymbol{P}^{-1} = \frac{1}{\det(\boldsymbol{P})} \times \operatorname{cof}(\boldsymbol{P})^T = \operatorname{cof}(\boldsymbol{P})^T. \tag{A.24}$$

The diagonal elements of $\operatorname{cof}(\boldsymbol{P})$, $\operatorname{cof}(\boldsymbol{P})_{ii}, i = 1, \ldots m$ are:

$$\begin{aligned} \operatorname{cof}(\boldsymbol{P})_{ii} &= \det\left(\begin{bmatrix} 1 & & & & & \\ \vdots & \ddots & & & & \\ p_{i-1,1} & \vdots & 1 & & & \\ p_{i+1,1} & \cdots & p_{i+1,i-1} & 1 & & \\ \vdots & \ddots & \vdots & \vdots & \ddots & \\ p_{l1} & \cdots & p_{l,i-1} & p_{l,i+1} & \cdots & 1 \end{bmatrix}\right) \times (-1)^{i+i} \\ &= \prod_{i=1}^{m-1} 1 \times (-1)^{2i} \\ &= 1. \end{aligned} \tag{A.25}$$

Therefore, all the diagonal elements of the lower triangular $\boldsymbol{P}^{-1}$ is 1, and **Lemma A.1.2** is proved. □

At the end of this section, we prove the invertibility $\boldsymbol{G}_A$ by proving the following statement:

Theorem A.1.1. *The block matrix constructed in (A.8) is invertible.*

Proof Mathematical induction:

i **Base step:** $N = 2$.

When $N = 2$, i.e., there are 2×2 blocks, the block matrix $\boldsymbol{BM}$ is written as

$$\boldsymbol{BM}^{(2)} = \begin{bmatrix} \boldsymbol{P}_{11} & \boldsymbol{L}_{12} \\ \boldsymbol{L}_{21} & \boldsymbol{P}_{22} \end{bmatrix}. \tag{A.26}$$

From **Lemma A.1.2**, $\boldsymbol{P}_{22}$ is invertible and $\boldsymbol{P}_{22}^{-1}$ is a unit lower triangular. The invertibility of $\boldsymbol{BM}^{(2)}$, based on **Lemma A.0.1**, is equivalent to its Schur complement relative in $\boldsymbol{P}_{22}$:

$$\boldsymbol{BM}^{(2)}/\boldsymbol{P}_{22} = \boldsymbol{P}_{11} - \boldsymbol{L}_{12}\boldsymbol{P}_{22}^{-1}\boldsymbol{L}_{21}. \tag{A.27}$$

According to **Lemma A.1.1**, $\boldsymbol{L}_{12}\boldsymbol{P}_{22}^{-1} \triangleq \tilde{\boldsymbol{L}}_{12}$ is a strictly block lower triangular matrix, and it is compatible with $\boldsymbol{L}_{21}$. Therefore, the diagonal blocks of $\boldsymbol{L}_{12}\boldsymbol{P}_{22}^{-1}\boldsymbol{L}_{21}$ would be

$$\tilde{\boldsymbol{L}}_{12_{\alpha\alpha}}\boldsymbol{L}_{21_{\alpha\alpha}} = \boldsymbol{O}_{n_\alpha \times m_\alpha}\boldsymbol{O}_{m_\alpha \times n_\alpha} = \boldsymbol{O}_{n_\alpha \times n_\alpha}, \alpha = 1, ..., S \tag{A.28}$$

which are square zero matrices. Therefore, the diagonal elements in $\boldsymbol{L}_{12}\boldsymbol{P}_{22}^{-1}\boldsymbol{L}_{21}$ are all 0, making it a strictly lower triangular matrix. Then the Schur complement becomes

$$\begin{aligned}\boldsymbol{BM}^{(2)}/\boldsymbol{P}_{22} =& \boldsymbol{P}_{11} - \boldsymbol{L}_{12}\boldsymbol{P}_{22}^{-1}\boldsymbol{L}_{21} \\ =& \begin{bmatrix} 1 & & & \\ * & 1 & & \\ \vdots & \vdots & \ddots & \\ * & * & \cdots & 1 \end{bmatrix} - \begin{bmatrix} 0 & & & \\ * & 0 & & \\ \vdots & \vdots & \ddots & \\ * & * & \cdots & 0 \end{bmatrix}, \\ =& \begin{bmatrix} 1 & & & \\ * & 1 & & \\ \vdots & \vdots & \ddots & \\ * & * & \cdots & 1 \end{bmatrix}\end{aligned} \tag{A.29}$$

which is a unit lower triangular matrix. The asterisks $*$ mean that the value of these element will not change the fact that $\boldsymbol{BM}^{(2)}/\boldsymbol{P}_{22}$ is unit lower triangular and is invertible. The invertibility of $\boldsymbol{BM}^{(2)}$ is hence proved.

ii **Induction steps:** $N = k > 2$. Assume that a block matrix $\boldsymbol{BM}^{(k)}$ with $k \times k$ blocks is constructed after (A.8) and is invertible. We are to prove that the block matrix $\boldsymbol{BM}^{(k+1)}$, which also follows the structure of (A.8) and is defined as:

$$\begin{aligned}\boldsymbol{BM}^{(k+1)} =& \left[\begin{array}{cccc:c} \boldsymbol{P}_{11} & \boldsymbol{L}_{12} & \cdots & \boldsymbol{L}_{1k} & \boldsymbol{L}_{1,k+1} \\ \boldsymbol{L}_{21} & \boldsymbol{P}_{22} & \cdots & \boldsymbol{L}_{2k} & \boldsymbol{L}_{2,k+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{L}_{k1} & \boldsymbol{L}_{k2} & \cdots & \boldsymbol{P}_{kk} & \boldsymbol{L}_{k,k+1} \\ \hdashline \boldsymbol{L}_{k+1,1} & \boldsymbol{L}_{k+1,2} & \cdots & \boldsymbol{L}_{k+1,k} & \boldsymbol{P}_{k+1,k+1} \end{array}\right] \\ =& \begin{bmatrix} \boldsymbol{BM}^{(k)} & \mathfrak{L}_{1,k+1} \\ \mathfrak{L}_{k+1,1} & \boldsymbol{P}_{k+1,k+1} \end{bmatrix},\end{aligned} \tag{A.30}$$

is invertible.

The Schur complement of $\boldsymbol{BM}^{(k+1)}$ relative in $\boldsymbol{P}_{k+1,k+1}$ is

$$\begin{aligned}\boldsymbol{BM}^{(k+1)}/\boldsymbol{P}_{k,k+1} =&\boldsymbol{BM}^{(k)} - \mathfrak{L}_{1,k+1}\boldsymbol{P}_{k+1,k+1}^{-1}\mathfrak{L}_{k+1,1} \\ =&\boldsymbol{BM}^{(k)} - \begin{bmatrix} \boldsymbol{L}_{1,k+1} \\ \vdots \\ \boldsymbol{L}_{k,k+1} \end{bmatrix} \boldsymbol{P}_{k+1,k+1}^{-1} \left[\boldsymbol{L}_{k+1,1} \cdots \boldsymbol{L}_{k+1,k}\right] \\ =&\boldsymbol{BM}^{(k)} - \begin{bmatrix} \boldsymbol{L}_{1,k+1}\boldsymbol{P}_{k+1,k+1}^{-1}\boldsymbol{L}_{k+1,1} & \cdots & \boldsymbol{L}_{1,k+1}\boldsymbol{P}_{k+1,k+1}^{-1}\boldsymbol{L}_{k+1,k} \\ \vdots & \ddots & \vdots \\ \boldsymbol{L}_{k,k+1}\boldsymbol{P}_{k+1,k+1}^{-1}\boldsymbol{L}_{k+1,1} & \cdots & \boldsymbol{L}_{k,k+1}\boldsymbol{P}_{k+1,k+1}^{-1}\boldsymbol{L}_{k+1,k} \end{bmatrix},\end{aligned} \tag{A.31}$$

where $\boldsymbol{L}_{i,k+1}\boldsymbol{P}_{k+1,k+1}^{-1}$ is a strictly lower block triangular matrix having the same structure as $\boldsymbol{L}_{i,k+1}$, $i = 1, ..., k$. Since $\boldsymbol{L}_{i,k+1}$ and $\boldsymbol{L}_{k+1,j}$, $i, j = 1, ...k$ are compatible with each other, the product $\boldsymbol{L}_{i,k+1}\boldsymbol{P}_{k+1,k+1}^{-1}\boldsymbol{L}_{k+1,j}$ has the same structure as $\boldsymbol{L}_{ij}$, $i \neq j, i, j = 1, ..., k$, and the product $\boldsymbol{L}_{i,k+1}\boldsymbol{P}_{k+1,k+1}^{-1}\boldsymbol{L}_{k+1,i}$ is a strict lower triangular matrix with the same dimension as $\boldsymbol{P}_{ii}$, $i = 1, ..., k$. The product $\boldsymbol{L}_{i,k+1}\boldsymbol{P}_{k+1,k+1}^{-1}\boldsymbol{L}_{k+1,j}$ is denoted to be $\boldsymbol{L}'_{ij}$, $\forall i, j = 1, ..., k$, and then there is

$$\begin{aligned}\boldsymbol{BM}^{(k+1)}/\boldsymbol{P}_{k,k+1} =&\boldsymbol{BM}^{(k)} - \begin{bmatrix} \boldsymbol{L}'_{11} & \cdots & \boldsymbol{L}'_{1k} \\ \vdots & \ddots & \vdots \\ \boldsymbol{L}'_{k1} & \cdots & \boldsymbol{L}'_{kk} \end{bmatrix} \\ =& \begin{bmatrix} \boldsymbol{P}_{11} - \boldsymbol{L}'_{11} & \cdots & \boldsymbol{L}_{1k} - \boldsymbol{L}'_{1k} \\ \vdots & \ddots & \vdots \\ \boldsymbol{L}_{k1} - \boldsymbol{L}'_{k1} & \cdots & \boldsymbol{P}_{kk} - \boldsymbol{L}'_{kk} \end{bmatrix},\end{aligned} \tag{A.32}$$

where $\tilde{\boldsymbol{L}}_{ij} \triangleq \boldsymbol{L}_{ij} - \boldsymbol{L}'_{ij}$ maintains the structure of $\boldsymbol{L}_{ij}$, $\forall i \neq j, i, j = 1, ..., k$, because $\boldsymbol{L}'_{ij}$ and $\boldsymbol{L}_{ij}$ are strictly block lower triangular matrices sharing a same structure. The matrices $\tilde{\boldsymbol{P}}_{ii} \triangleq \boldsymbol{P}_{ii} - \boldsymbol{L}'_{ii}$ are still unit lower triangular matrices, $\forall i = 1, ..., k$. Then $\boldsymbol{BM}^{(k+1)}/\boldsymbol{P}_{k,k+1}$ can be further represented as:

$$\boldsymbol{BM}^{(k+1)}/\boldsymbol{P}_{k,k+1} = \begin{bmatrix} \tilde{\boldsymbol{P}}_{11} & \cdots & \tilde{\boldsymbol{L}}_{1k} \\ \vdots & \ddots & \vdots \\ \tilde{\boldsymbol{L}}_{k1} & \cdots & \tilde{\boldsymbol{P}}_{kk} \end{bmatrix} \triangleq \widetilde{\boldsymbol{BM}}^{(k)}. \tag{A.33}$$

In equation (A.33), $\boldsymbol{B}\tilde{\boldsymbol{M}}^{(k)}$ is a block matrix with $k \times k$ blocks following the structure of (A.8), which is invertible according to the assumption. Therefore, $\boldsymbol{BM}^{(k+1)}$ is invertible if $\boldsymbol{BM}^{(k)}$ is invertible.

The invertibility of a matrix $\boldsymbol{BM}$ as constructed in (A.8) is hence proved. □

Since $\boldsymbol{G}_A$ is a matrix as structed in (A.8), it is invertible.

A.2 The Invertibility of $\boldsymbol{G}\boldsymbol{G}^T$

The concepts of **positive semidefinte** and **positive definite** matrices and **Lemma A.2.1** are introduced first prior to the proof. A symmetric matrix $\boldsymbol{S} \in \mathbb{R}^{m\times m}$ is said to be positive semidefinite if $\boldsymbol{z}^T\boldsymbol{S}\boldsymbol{z} \geq 0, \forall \boldsymbol{z} \in \mathbb{R}^m$. Moreover, if $\boldsymbol{z}^T\boldsymbol{S}\boldsymbol{z} > 0, \forall \boldsymbol{z} \neq \boldsymbol{0}$, $\boldsymbol{S}$ is said to be positive definite. Positive semidefinite and positive definite matrices have many important properties, some of which will be used in the invertibility proofs and are presented in **Property A.2.1** and **Property A.2.2**.

Property A.2.1. (Horn and Johnson, 2012) *The eigenvalues of a positive semidefinite matrix are all nonnegative. The eigenvalues of a positive definite matrix are all positive.*

Property A.2.2. (Horn and Johnson, 2012) *A positive semidefinite matrix $\boldsymbol{S}$ can always be written as the product of some matrix multiplying its transpose, i.e., $\exists \boldsymbol{Y}$ such that $\boldsymbol{S} = \boldsymbol{Y}^T\boldsymbol{Y}$. Conversely, the product $\boldsymbol{X}^T\boldsymbol{X}, \forall$real matrix $\boldsymbol{X}$, is always positive semidefinite.*

Lemma A.2.1. *A positive semidefinite matrix $\boldsymbol{S} \in \mathbb{R}^{m\times m}$ is positive definite if and only if it is invertible.*

Proof Denote the m eigenvalues of $\boldsymbol{S}$ to be $\lambda_i, i = 1, ..., m$, from **Property A.2.1** there is $\lambda_i \geq 0, \forall i = 1, ..., m$.

If $\boldsymbol{S}$ is positive definite, there is $\lambda_i > 0, \forall i = 1, ..., m$. Since $\det(\boldsymbol{S}) = \prod_{i=1}^m \lambda_i$ (Horn and Johnson, 2012), it can be easily seen that $\det(\boldsymbol{S}) > 0 \neq 0$. Therefore $\boldsymbol{S}$ is invertible.

Assume that $\boldsymbol{S}$ is invertible but not positive definite. Then $\boldsymbol{S}$ has at least one eigenvalue that is 0, which implies that $\det(\boldsymbol{S}) = \prod_{i=1}^m \lambda_i = 0$. $\boldsymbol{S}$ is therefore not invertible. This contradicts the assumption that $\boldsymbol{S}$ is invertible, which leads to the conclusion that a positive semidefinite matrix is invertible only if it is positive definite. □

Theorem A.2.1. *The matrix $\boldsymbol{G}\boldsymbol{G}^T$ is invertible, where $\boldsymbol{G} = [\boldsymbol{G}_A, \boldsymbol{G}_B]$, $\boldsymbol{G}_A$ and $\boldsymbol{G}_B$ are defined in (2.18).*

Proof Since $\boldsymbol{G} = [\boldsymbol{G}_A, \boldsymbol{G}_B]$, there is

$$\boldsymbol{G}\boldsymbol{G}^T = \boldsymbol{G}_A\boldsymbol{G}_A^T + \boldsymbol{G}_B\boldsymbol{G}_B^T. \tag{A.34}$$

From **Property A.2.2**, it can be seen that $\boldsymbol{G}_A\boldsymbol{G}_A^T$ and $\boldsymbol{G}_B\boldsymbol{G}_B^T$ are both positive semidefinite. Moreover, since $\boldsymbol{G}_A$ is invertible as proved in §A.1, $\det(\boldsymbol{G}_A) \neq 0$. Then:

$$\det(\boldsymbol{G}_A\boldsymbol{G}_A^T) = \det(\boldsymbol{G}_A)\det(\boldsymbol{G}_A^T) = \det(\boldsymbol{G}_A)^2 > 0,$$

implying that $\boldsymbol{G}_A\boldsymbol{G}_A^T$ is positive definite. Therefore, $\boldsymbol{z}^T\boldsymbol{G}_A\boldsymbol{G}_A^T\boldsymbol{z} > 0, \forall \boldsymbol{z} \neq 0$, and:

$$\begin{aligned} \boldsymbol{z}^T\boldsymbol{G}\boldsymbol{G}^T\boldsymbol{z} &= \boldsymbol{z}^T(\boldsymbol{G}_A\boldsymbol{G}_A^T + \boldsymbol{G}_B\boldsymbol{G}_B^T)\boldsymbol{z} \\ &= \boldsymbol{z}^T\boldsymbol{G}_A\boldsymbol{G}_A^T\boldsymbol{z} + \boldsymbol{z}^T\boldsymbol{G}_B\boldsymbol{G}_B^T\boldsymbol{z} > 0. \end{aligned} \tag{A.35}$$

From the definition of positive definite matrix, the conclusion immediately follows that $\boldsymbol{G}\boldsymbol{G}^T$ is positive definite, which is invertible according to **Lemma A.2.1**. □

A.3 The Invertibility of $\boldsymbol{W}$, $\boldsymbol{W}_i$, $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}_i$

The matrix $\boldsymbol{W}$ was defined as:

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{O} & \bar{\boldsymbol{G}}_A^T \\ \boldsymbol{O} & \boldsymbol{R} & \bar{\boldsymbol{G}}_B^T \\ \bar{\boldsymbol{G}}_A & \bar{\boldsymbol{G}}_B & \boldsymbol{O} \end{bmatrix}$$

and $\boldsymbol{\Psi}$ as:

$$\boldsymbol{\Psi} = \bar{\boldsymbol{G}}_A\boldsymbol{Q}^{-1}\bar{\boldsymbol{G}}_A^T + \bar{\boldsymbol{G}}_B\boldsymbol{R}^{-1}\bar{\boldsymbol{G}}_B^T$$

in §2.2.1. Note that $-\boldsymbol{\Psi}$ is the Schur complement of $\boldsymbol{W}$ relative in $diag(\boldsymbol{Q}, \boldsymbol{R})$. According to **Corollary A.0.1**, the invertibility of $\boldsymbol{W}$ is equivalent to that of $-\boldsymbol{\Psi}$, and hence equivalent to that of $\boldsymbol{\Psi}$.

Similarly, $\boldsymbol{W}_i$ and $\boldsymbol{\Psi}_i$ are defined as:

$$\boldsymbol{W}_i = \begin{bmatrix} \boldsymbol{Q}_i & \boldsymbol{O} & \boldsymbol{G}_{A_{ii}}^T \\ \boldsymbol{O} & \boldsymbol{R}_i & \boldsymbol{G}_{B_{ii}}^T \\ \boldsymbol{G}_{A_{ii}} & \boldsymbol{G}_{B_{ii}} & \boldsymbol{O} \end{bmatrix} \tag{A.36}$$

and:

$$\boldsymbol{\Psi}_i = \boldsymbol{G}_{A_{ii}}\boldsymbol{Q}_i^{-1}\boldsymbol{G}_{A_{ii}}^T + \boldsymbol{G}_{B_{ii}}\boldsymbol{R}_i^{-1}\boldsymbol{G}_{B_{ii}}^T, \tag{A.37}$$

respectively. $-\boldsymbol{\Psi}_i$ is the Schur complement of $\boldsymbol{W}_i$ relative in $diag(\boldsymbol{Q}_i, \boldsymbol{R}_i)$, so the invertibility of $\boldsymbol{W}_i$ and $\boldsymbol{\Psi}_i$ is equivalent as well.

Because of these equivalencies, only the invertibility proofs of $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}_i$ will be adequate. Another property of positive definite matrices are required to build the proofs:

Property A.3.1. (Horn and Johnson, 2012) *The inverse of a positive definite matrix is also positive definite.*

Theorem A.3.1. *The matrices $\boldsymbol{W}$, $\boldsymbol{W}_i$, $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}_i$ as defined respectively in (3.21), (3.53), (3.25) and (3.57) are invertible.*

Proof Matrices $\boldsymbol{Q}_i$ and $\boldsymbol{R}_i$ are positive definite, therefore, their inverses are also positive definite. According to **Property A.2.2**, there exist some matrices $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ such that $\boldsymbol{Q}_i^{-1} = \boldsymbol{X}_i^T \boldsymbol{X}_i$ and $\boldsymbol{R}_i^{-1} = \boldsymbol{Y}_i^T \boldsymbol{Y}_i$. Then it can be seen that $\boldsymbol{G}_{A_{ii}} \boldsymbol{Q}_i^{-1} \boldsymbol{G}_{A_{ii}}^T = (\boldsymbol{X}_i \boldsymbol{G}_{A_{ii}}^T)^T (\boldsymbol{X}_i \boldsymbol{G}_{A_{ii}}^T)$ and $\boldsymbol{G}_{B_{ii}} \boldsymbol{R}_i^{-1} \boldsymbol{G}_{B_{ii}}^T = (\boldsymbol{Y}_i \boldsymbol{G}_{B_{ii}}^T)^T (\boldsymbol{Y}_i \boldsymbol{G}_{B_{ii}}^T)$. Consequently, both of them are positive semidefinite.

Furthermore, the matrix $\boldsymbol{G}_{A_{ii}}$, which is defined as

$$\boldsymbol{G}_{A_{ii}} = \underbrace{\begin{bmatrix} \boldsymbol{I}_{n_i} & & & & \\ -\boldsymbol{A}_{ii} & \boldsymbol{I}_{n_i} & & & \\ & -\boldsymbol{A}_{ii} & \boldsymbol{I}_{n_i} & & \\ & & \ddots & \ddots & \\ & & & -\boldsymbol{A}_{ii} & \boldsymbol{I}_{n_i} \end{bmatrix}}_{H_p \times H_p \text{ blocks}},$$

is a unit lower triangular matrix. This indicates that $\boldsymbol{G}_{A_{ii}}$ is invertible and has $\det(\boldsymbol{G}_{A_{ii}}) \neq 0$, which results in:

$$\det(\boldsymbol{G}_{A_{ii}} \boldsymbol{Q}_i^{-1} \boldsymbol{G}_{A_{ii}}^T) = \det(\boldsymbol{G}_{A_{ii}})^2 \det(\boldsymbol{Q}_i^{-1}) > 0. \tag{A.38}$$

Therefore, $\boldsymbol{G}_{A_{ii}} \boldsymbol{Q}_i^{-1} \boldsymbol{G}_{A_{ii}}^T$ is positive definite. Then:

$$\boldsymbol{z}_i^T \boldsymbol{\Psi}_i \boldsymbol{z}_i = \boldsymbol{z}_i^T \boldsymbol{G}_{A_{ii}} \boldsymbol{Q}_i^{-1} \boldsymbol{G}_{A_{ii}}^T \boldsymbol{z}_i + \boldsymbol{z}_i^T \boldsymbol{G}_{B_{ii}} \boldsymbol{R}_i^{-1} \boldsymbol{G}_{B_{ii}}^T \boldsymbol{z}_i > 0 \tag{A.39}$$

$\forall \boldsymbol{z}_i \in \mathbb{R}^{H_p n_i} \neq \boldsymbol{0}$. This proves that $\boldsymbol{\Psi}_i$ is positive definite, and its invertibility follows immediately.

The invertibility proof of $\boldsymbol{\Psi}$ can be done in exactly the same way as $\boldsymbol{\Psi}_i$, but note that:

$$\begin{aligned} \boldsymbol{Q} &= diag(\boldsymbol{Q}_1, \boldsymbol{Q}_2, ..., \boldsymbol{Q}_N), \\ \boldsymbol{R} &= diag(\boldsymbol{R}_1, \boldsymbol{R}_2, ..., \boldsymbol{R}_N), \\ \bar{\boldsymbol{G}}_A &= diag(\boldsymbol{G}_{A_{11}}, ..., \boldsymbol{G}_{A_{NN}}), \\ \bar{\boldsymbol{G}}_B &= diag(\boldsymbol{G}_{B_{11}}, ..., \boldsymbol{G}_{B_{NN}}). \end{aligned}$$

The inverse of the block-diagonal matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, are:

$$\begin{aligned} \boldsymbol{Q}^{-1} &= diag(\boldsymbol{Q}_1^{-1}, \boldsymbol{Q}_2^{-1}, ..., \boldsymbol{Q}_N^{-1}), \\ \boldsymbol{R}^{-1} &= diag(\boldsymbol{R}_1^{-1}, \boldsymbol{R}_2^{-1}, ..., \boldsymbol{R}_N^{-1}). \end{aligned}$$

From the block-matrix multiplication rule, there is:

$$\begin{aligned} \boldsymbol{z}^T \boldsymbol{\Psi} \boldsymbol{z} &= \boldsymbol{z}^T (\bar{\boldsymbol{G}}_A \boldsymbol{Q}^{-1} \bar{\boldsymbol{G}}_A^T + \bar{\boldsymbol{G}}_B \boldsymbol{R}^{-1} \bar{\boldsymbol{G}}_B^T) \boldsymbol{z} \\ &= \sum_{i=1}^{N} \boldsymbol{z}_i^T \boldsymbol{G}_{A_{ii}} \boldsymbol{Q}_i^{-1} \boldsymbol{G}_{A_{ii}}^T \boldsymbol{z}_i + \boldsymbol{z}_i^T \boldsymbol{G}_{B_{ii}} \boldsymbol{R}_i^{-1} \boldsymbol{G}_{B_{ii}}^T \boldsymbol{z}_i > 0, \end{aligned} \tag{A.40}$$

$\forall \boldsymbol{z} \in \mathbb{R}^{H_p n} \neq \boldsymbol{0}$, where $\boldsymbol{z} = [\boldsymbol{z}_1^T, \boldsymbol{z}_2^T, ..., \boldsymbol{z}_N^T]^T$ and $\boldsymbol{z}_i \in \mathbb{R}^{H_p n_i}$. This shows that $\boldsymbol{\Psi}$ is also a positive definite matrix and thus invertible.

Since $\boldsymbol{\Psi}_i$ being invertible ensures the existence of $\boldsymbol{W}_i{}^{-1}$, and $\boldsymbol{\Psi}$ ensures $\boldsymbol{W}^{-1}$, we conclude that all of the four matrices are invertible. □

A.4 The Invertibility of $\boldsymbol{\Lambda}_i$ and $\boldsymbol{\Lambda}$

The matrices $\boldsymbol{\Lambda}_i$ and $\boldsymbol{\Lambda}$ were defined as:

$$\boldsymbol{\Lambda}_i = \left[\begin{array}{ccc:c} \boldsymbol{Q}_i & & & \boldsymbol{G}_{A_{ii}}^T \\ & \boldsymbol{R}_i & & \boldsymbol{G}_{B_{ii}}^T \\ & & \boldsymbol{O}_{H_p n_i} & \boldsymbol{I}_{H_p n_i} \\ \hdashline \boldsymbol{G}_{A_{ii}} & \boldsymbol{G}_{B_{ii}} & \boldsymbol{I}_{H_p n_i} & \end{array}\right],$$

and:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{Q} & & & \bar{\boldsymbol{G}}_A^T \\ & \boldsymbol{R} & & \bar{\boldsymbol{G}}_B^T \\ & & \boldsymbol{O}_{H_p n} & \boldsymbol{I}_{H_p n} \\ \bar{\boldsymbol{G}}_A & \bar{\boldsymbol{G}}_B & \boldsymbol{I}_{H_p n} & \end{bmatrix},$$

in §4.1 and §4.2, respectively.

Theorem A.4.1. *The matrices $\boldsymbol{\Lambda}_i$ and $\boldsymbol{\Lambda}$ as defined in (4.7) and (4.25) are invertible.*

Proof Since $\boldsymbol{Q}_i$ and $\boldsymbol{R}_i$ are positive definite, the matrix $\boldsymbol{\Upsilon}_i \triangleq diag(\boldsymbol{Q}_i, \boldsymbol{R}_i)$ is nonsingular. According to **Corollary A.0.1**, the invertibility of $\boldsymbol{\Lambda}_i$ is equivalent to that of its Schur complement relative in $\boldsymbol{\Upsilon}_i$:

$$\begin{aligned} \boldsymbol{\Lambda}_i/\boldsymbol{\Upsilon}_i &= \begin{bmatrix} & \boldsymbol{I}_{H_p n_i} \\ \boldsymbol{I}_{H_p n_i} & \end{bmatrix} - \begin{bmatrix} \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{G}_{A_{ii}} & \boldsymbol{G}_{B_{ii}} \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_i^{-1} & \\ & \boldsymbol{R}_i^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{O} & \boldsymbol{G}_{A_{ii}}^T \\ \boldsymbol{O} & \boldsymbol{G}_{B_{ii}}^T \end{bmatrix} \\ &= \begin{bmatrix} & \boldsymbol{I}_{H_p n_i} \\ \boldsymbol{I}_{H_p n_i} & \boldsymbol{G}_{A_{ii}} \boldsymbol{Q}_i^{-1} \boldsymbol{G}_{A_{ii}}^T + \boldsymbol{G}_{B_{ii}} \boldsymbol{R}_i^{-1} \boldsymbol{G}_{B_{ii}}^T \end{bmatrix} \\ &= \begin{bmatrix} & \boldsymbol{I}_{H_p n_i} \\ \boldsymbol{I}_{H_p n_i} & \boldsymbol{\Psi}_i \end{bmatrix}, \end{aligned}$$

where $\boldsymbol{\Psi}_i$ was proved to be invertible in §A.3. We apply **Corollary A.0.1** again and found that the Schur complement $\boldsymbol{\Lambda}_i/\boldsymbol{\Upsilon}_i$ is invertible if and only if $-(\boldsymbol{\Psi}_i)^{-1}$ is invertible, which is readily given. Therefore, $\boldsymbol{\Lambda}_i$ is invertible.

The Schur complement of $\boldsymbol{\Lambda}$ relative in$\boldsymbol{\Upsilon} \triangleq diag(\boldsymbol{Q}, \boldsymbol{R})$ is:

$$\boldsymbol{\Lambda}/\boldsymbol{\Upsilon} = \begin{bmatrix} & \boldsymbol{I}_{H_p n} \\ \boldsymbol{I}_{H_p n} & \boldsymbol{\Psi} \end{bmatrix},$$

where the invertibility of $\boldsymbol{\Psi}$ was also given in §A.3. Similar to $\boldsymbol{\Lambda}_i$, $\boldsymbol{\Lambda}$ can be proved to be invertible applying **Corollary A.0.1** twice. □

A.5 The Invertibility of $\boldsymbol{H}$

The matrix $\boldsymbol{H}$ in the unconstrained context was defined as:

$$\boldsymbol{H} = -(\boldsymbol{G}_A\boldsymbol{Q}^{-1}\boldsymbol{G}_A^T + \boldsymbol{G}_B\boldsymbol{R}^{-1}\boldsymbol{G}_B^T).$$

in equation (4.37).

Theorem A.5.1. *The matrix $\boldsymbol{H}$ as defined in (4.37) is invertible.*

Proof Since $\boldsymbol{G}_A$ was proved to be nonsingular in §A.1, $\det(\boldsymbol{G}_A) \neq 0$. Then there is $\det(\boldsymbol{G}_A\boldsymbol{Q}^{-1}\boldsymbol{G}_A^T) = \det(\boldsymbol{G}_A)^2 \det(\boldsymbol{Q}^{-1}) > 0$. For $\forall \boldsymbol{z} \neq \boldsymbol{0}$, the following inequality holds:

$$\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} = -\boldsymbol{z}^T\boldsymbol{G}_A\boldsymbol{Q}^{-1}\boldsymbol{G}_A^T\boldsymbol{z} - \boldsymbol{z}^T\boldsymbol{G}_B\boldsymbol{R}^{-1}\boldsymbol{G}_B^T\boldsymbol{z} < 0, \tag{A.41}$$

indicating that $\boldsymbol{H}$ is negative definite. Therefore, $\boldsymbol{H}$ is invertible. □

Appendix B

Mathematical Background of Computational Complexity

The definitions and properties of O-notation in complexity analysis can be found in a number of algorithm textbooks such as Wilf, 2002, Brassard and Bratley, 1996 and Cormen, 2009, etc.. We refer to Brassard and Bratley, 1996 to introduce the concept of O-notation.

In this chapter, $\mathbb{N}$ denotes the set of natural numbers, i.e., $\{1, 2, 3, ...\}$, and $\mathbb{R}^+$ denotes the set of positive real numbers.

Definition B.0.1. *Assume that $t \in \mathbb{N}$, and there are two functions $f, g : \mathbb{N} \to \mathbb{R}^+$. We write:*

$$f(t) = O(g(t)), t \to \infty, \tag{B.1}$$

if $\exists C > 0$ and $M \in \mathbb{N}$, such that $f(t) < Cg(t)$, $\forall n > M$.

'$t \to \infty$' is omitted since we are interested in a large t in our discussion. Note that there is some abuse of notation in the way that we are writing $f(t) = O(g(t))$. From its definition, $O(g(t))$ is a set and what '$=$' really means is '$\in$'; however, it is the traditional way to write '$f(t) = O(g(t))$' and the convention will be followed.

From **Definition B.0.1**, **Property B.0.1** and **Property B.0.2** of O-notation can be derived:

Property B.0.1. (Basic arithmetic)

(1) **Sum** *If $f_1(t) = O(g_1(t))$, $f_2(t) = O(g_2(t))$, then $f_1(t) + f_2(t) = O(g_1(t) + g_2(t))$;*

(2) **Product** *If $f_1(t) = O(g_1(t))$, $f_2(t) = O(g_2(t))$, then $f_1(t)f_2(t) = O(f_1(t)f_2(t))$;*

(3) **Mutiplication with a Constant** *If $f(t) = O(g(t))$, then $Kf(t) = O(g(t))$ for $\forall K \neq 0$.*

Property B.0.2. (Maximum rule)

$$O(f_1(t) + f_2(t)) = O(max(f_1(t), f_2(t))). \tag{B.2}$$

The sum-rule of O-notation is applied when the complexity of sequentially executed codes is calculated, while the product-rule is required for iterative codes. **Property B.0.2** indicates that the complexity of the sequentially executed codes is contributed by its most time-consuming part, since the complexity of other parts are neglectable when $t \to \infty$.

If there are more than one variable to describe the problem size, O-notation is defined as follows:

Definition B.0.2. *Assume that $\boldsymbol{t}$ is a $k \times 1$ vector with all $t_i \in \mathbb{N}$, $i = 1, \cdots, k$ and there are two functions $f, g : \mathbb{N}^k \to \mathbb{R}^+$. We write:*

$$f(\boldsymbol{t}) = O(g(\boldsymbol{t})), t_1, \cdots, t_k \to \infty, \tag{B.3}$$

if $\exists C > 0$ and $M \in \mathbb{N}$ such that $f(\boldsymbol{t}) < Cg(\boldsymbol{t})$ for $\forall \boldsymbol{t}$ with $\forall t_i > M$.

Consequently, the basic arithmetic and maximum rule for multivariable O-notation are adapted as:

Property B.0.3. (Basic arithmetic for multiple variables)
(1) **Sum** *If $f_1(\boldsymbol{t}) = O(g_1(\boldsymbol{t}))$, $f_2(\boldsymbol{t}) = O(g_2(\boldsymbol{t}))$, then $f_1(\boldsymbol{t}) + f_2(\boldsymbol{t}) = O(g_1(\boldsymbol{t}) + g_2(\boldsymbol{t}))$;*
(2) **Product** *If $f_1(\boldsymbol{t}) = O(g_1(\boldsymbol{t}))$, $f_2(\boldsymbol{t}) = O(g_2(\boldsymbol{t}))$, then $f_1(\boldsymbol{t})f_2(\boldsymbol{t}) = O(f_1(\boldsymbol{t})f_2(\boldsymbol{t}))$;*
(3) **Mutiplication with a Constant** *If $f(t) = O(g(t))$, then $Kf(t) = O(g(t))$ for $\forall K \neq 0$.*

Property B.0.4. (Maximum rule for multiple variables)

$$O(g_1(\boldsymbol{t}) + g_2(\boldsymbol{t})) = O(max(g_1(\boldsymbol{t}), g_2(\boldsymbol{t}))). \tag{B.4}$$

Note that although **Property B.0.3** and **Property B.0.4** are presented for functions with same sets of variables, they can also be applied to functions with different sets of variables. To see this, assume that there are two functions $f_1(\boldsymbol{t}) = O(g_1(\boldsymbol{t}))$ and $f_2(\boldsymbol{s}) = O(g_2(\boldsymbol{s}))$, where $\boldsymbol{t} = [t_1, \cdots, t_m, l_1, \cdots, l_n]^T$ and $\boldsymbol{s} = [s_1, \cdots, s_o, l_1, \cdots, l_n]^T$, with $t_i, l_j, s_k \in \mathbb{N}(i = 1, ..., m; j = 1, ..., n; k = 1, ..., o)$. Define $\boldsymbol{p}$ to be $\boldsymbol{p} = [t_1, \cdots, t_m, s_1, \cdots, s_o, l_1, \cdots, l_n]^T$. Then $f_1(\boldsymbol{t})$, $f_2(\boldsymbol{s})$, $g_1(\boldsymbol{t})$ and $g_2(\boldsymbol{s})$ can be transformed to:

$$f_1(\boldsymbol{t}) = \tilde{f_1}(\boldsymbol{p}) = f_1(\boldsymbol{t}) + \mathbf{0}_{o\times 1}^T \boldsymbol{s};$$
$$f_2(\boldsymbol{s}) = \tilde{f_2}(\boldsymbol{p}) = f_2(\boldsymbol{s}) + \mathbf{0}_{m\times 1}^T \boldsymbol{t};$$
$$g_1(\boldsymbol{t}) = \tilde{g_1}(\boldsymbol{p}) = g_1(\boldsymbol{t}) + \mathbf{0}_{o\times 1}^T \boldsymbol{s};$$
$$g_2(\boldsymbol{s}) = \tilde{g_2}(\boldsymbol{p}) = g_2(\boldsymbol{s}) + \mathbf{0}_{m\times 1}^T \boldsymbol{t}.$$

It can be proved that $f_1(\boldsymbol{t}) = O(g_1(\boldsymbol{t}))$ is equivalent to $\tilde{f}_1(\boldsymbol{p}) = O(\tilde{g_1}(\boldsymbol{p}))$, and $f_2(\boldsymbol{s}) = O(g_2(\boldsymbol{s}))$ to $\tilde{f}_2(\boldsymbol{p}) = O(\tilde{g_2}(\boldsymbol{p}))$. **Property B.0.3** and **Property B.0.4** can then be applied to the functions of $\boldsymbol{p}$. For example, applying the summation rule in **Property B.0.3**, we can conclude that $\tilde{f}_1(\boldsymbol{p}) + \tilde{f}_2(\boldsymbol{p}) = O(\tilde{g_1}(\boldsymbol{p}) + \tilde{g_2}(\boldsymbol{p}))$, which is the same as $f_1(\boldsymbol{t}) + f_2(\boldsymbol{s}) = O(g_1(\boldsymbol{t}) + g_2(\boldsymbol{s}))$. The rest properties can be derived in the same way.

Appendix C

Numerical Experiments

We designed two numerical experiments to learn the empirical complexity of the two CDMPC methods, where the relationship between the sizes of systems and the complexity is the focus of our study. Experiment 1 fixes the sizes of subsystems and varies the number of subsystems, so that the overall plant size will change. Experiment 2 fixes the overall plant size and the number of subsystems and varies the sizes of subsystems, so that the computational load on each subsystem will change. Any selection from the various related situations in an experiment, is called a 'mode' in the experiment. For example, in Experiment 1, $N = 2$ is a mode. As stated in §3.3, Experiment 1 has 23 modes and Experiment 2 has 17 modes. Both experiments were conducted on both of the two addressed CDMPC methods. Within each CDMPC method, two scenarios were considered (MPCs solved analytically and numerically). For every scenario in each CDMPC method, each mode was repeated 100 times.

The procedure of the numerical experiments is described as follows:

- **Step 1.** For a certain scenario in a certain experiment, the r^{th} repitition of a specific mode is started. Based on the mode selection, an MPC network is generated. In the experiments of prediction-driven CDMPC, the generated systems must have $\rho(\mathbf{C_2}) < 1$, where $\mathbf{C_2}$ was defined in (3.33). Therefore, the generating programming will continue to produce new MPC networks until it finds out that the requirement of $\rho(\mathbf{C_2})$ is satisfied. System states and inputs are arbitrarily assigned to be their values at the k^{th} sampling time. Parameters that are required to execute the CDMPC algorithms are also generated.

- **Step 2.** The prediction-driven CDMPC (using **Algorithm 1**) or the price-driven CDMPC (using **Algorithm 2**) coordinates the generated control network until the termination criterion is met. When this happens, it is considered that the centralized solution is reached. A centralized controller also solves the same optimization problem. In Scenario 1, the MPC solutions are calculated based on analytical formulations. In Scenario 2, the MPC solutions are obtained by the MATLAB® IPM solver.

- **Step 3.** For the CDMPC algorithms, the number of communication cycles $CCN_{(r)}$, the accumulated execution time of the coordinator $\sum t_{coor(r)}$ and the longest accumulated execution time of subsystems $\sum t_{i_{max}(r)}$ are recorded. The execution time of the centralized MPC $t_{CEN(r)}$ is also recorded. At this point, the r^{th} repitition is finished. The execution time of the coordinator is $t_{coor(r)} = \sum t_{coor(r)}/CCN_{(r)}$ and the longest execution time of subsystems is $t_{i_{max}(r)} = \sum t_{i_{max}(r)}/CCN_{(r)}$.

- **Step 4.** After a mode is repeated for 100 times, the averages of $t_{coor(r)}$ and $t_{i_{max}(r)}$, $r = 1, ..., 100$ are taken as t_{coor} and $t_{i_{max}}$ of this mode, respectively. Similarly, t_{CEN} of this mode is the average of $t_{CEN(r)}$, $r = 1, ..., 100$.

Note that the CDMPC algorithms were introduced in Chapter 3 and Chapter 4. All of the other computations are arithmetic calculations. The only part that has not been discussed is how the CDMPC network is generated, which will be presented in the following section.

C.1 Parameter Generation

Generating a control network is essentially generating a list of parameters. According to their properties, the parameters can generally be classified into three categories: a) system parameters; b) local MPC design parameters; and c) CDMPC design parameters. System parameters determine the properties of a system. Examples are $\boldsymbol{A}$ and $\boldsymbol{B}$ in equation (2.3), the number of subsystems N, etc.. Local MPC design parameters determine the feature of the MPC scheme. For example, the weighting matrices $\boldsymbol{Q}_i$ and $\boldsymbol{R}_i$ in optimization problem (2.28a) to (2.28c), etc.. CDMPC design parameters, such as the accuracy threshold ϵ, ensure the CDMPC algorithm to run properly. A full list for the above three types of parameters could be found in Table C.1, Table C.2 and Table C.3. Note that for the system parameters, although knowing all of the subsystem information would be adequate to know the plant model, we still list both the subsystem parameters and plant parameters. Most parameters have their sets of values provided in the 'Description' column; however, the generation of system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ requires more explanation, which will be discussed in §C.1.1

C.1.1 System Parameters: $\boldsymbol{A}$ and $\boldsymbol{B}$ Matrices

The generation of system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are done in two steps. First, the off-diagonal matrices in $\boldsymbol{A}$, i.e. $\boldsymbol{A}_{ij}, i, j = 1, ..., N, i \neq j$ are generated. Second, all of the local state and input matrices $\boldsymbol{A}_{ii}$ and $\boldsymbol{B}_{ii}$, $i = 1, ..., N$, are generated. Note that $\boldsymbol{B}_{ij}$'s are zero matrices, so $\boldsymbol{B}$ is obtained after $\boldsymbol{B}_{ii}, i = 1, ..., N$ are generated. If we denote the aggregation of the diagonal matrices in $\boldsymbol{A}$ to be:

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_{11} & & & \\ & \boldsymbol{A}_{22} & & \\ & & \ddots & \\ & & & \boldsymbol{A}_{NN} \end{bmatrix} \tag{C.1}$$

and the aggregation of the off-diagonal matrices to be:

$$\underline{\boldsymbol{A}} = \boldsymbol{A} - \bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{A}_{12} & \cdots & \boldsymbol{A}_{1N} \\ \boldsymbol{A}_{21} & \boldsymbol{O} & \cdots & \boldsymbol{A}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{N1} & \boldsymbol{A}_{N2} & \cdots & \boldsymbol{O} \end{bmatrix}, \tag{C.2}$$

then the first step can be described as to generate $\underline{\boldsymbol{A}}$, and the second step to generate $\bar{\boldsymbol{A}}$ and $\boldsymbol{B}$. The spectral radius of the resulting $\boldsymbol{A}$ matrix, $\boldsymbol{A} = \bar{\boldsymbol{A}} + \underline{\boldsymbol{A}}$, needs to be checked to ensure that the system is stable.

Step 1, $\underline{\boldsymbol{A}}$: It is taken as a-priori knowledge in the numerical experiments that the off-diagonoal matrix $\underline{\boldsymbol{A}}$ is often sparse. The reason is that in a large-scale chemical plant, the operation units are often linked together by sequential flows and a few recycle flows. This results in a sparse feature of the plant model. The function to generate a sparse matrix is described in **Algorithm 3**. In **Algorithm 3**, the sparse density is defined as $d = \frac{k}{nm}$ for an $n \times m$ matrix, where k is the number of nonzero elements in the matrix.

Algorithm 3 *createSparse*

```
function CREATESPARSE($n$, $m$, $d$)
    generate a random $n \times m$ matrix $\boldsymbol{M}$, with sparse density $d$ and whose nonzero elements distributed uniformly on $(-1, 1)$;
    return $\boldsymbol{M}$
end function
```

MATLAB® has a function *sprand*, where the non-zero elements are uniformly distributed on the interval $(0, 1)$. In our experiments, the non-zero elements generated by *sprand* is rescaled to be uniformly distributed on $(-1, 1)$, and this modified function is served as *createSparse*. The algorithm used to generate $\underline{\boldsymbol{A}}$ is presented in **Algorithm 4**, where $d_{\underline{A}}$ is assigned to be $1/n$.[1]

Algorithm 4 Algorithm to generate $\underline{\boldsymbol{A}}$

```
$\underline{\boldsymbol{A}} = createSparse(n, n, d_{\underline{A}})$,

set the elements corresponding to the spots of $\bar{\boldsymbol{A}}$ to be 0,
return $\underline{\boldsymbol{A}}$.
```

Step 2, $\bar{\boldsymbol{A}}$ and $\boldsymbol{B}$: Since there is no special request for matrix $\boldsymbol{B}$, the local input matrix $\boldsymbol{B}_{ii}$ are generated using **Algorithm 3**, where the sparse density $d_{B_{ii}}$ is a random variable uniformley distributed on the interval (0.5, 0.75). Nevertheless, for $\boldsymbol{A}$ the challenge is that it must be stable, and $\bar{\boldsymbol{A}}$ would affect $\rho(\boldsymbol{A})$ more than the sparse off-diagonal matrix $\underline{\boldsymbol{A}}$. If simple functions like **Algorithm 3** are used to generate $\boldsymbol{A}_{ii}$'s, it would be very hard to produce a stable $\boldsymbol{A}$, according to some pre-tests. A stable $\boldsymbol{A}$ would be easier to generate if $\boldsymbol{A}_{ii}, i = 1, ..., N$ are all stable. It is also easier if $\boldsymbol{A}_{ii}$'s have fewer non-zero entries, i.e., to be more sparse, especially when the size of $\boldsymbol{A}_{ii}$ grows bigger. Based on these observations, we developed the algorithm to generate $\boldsymbol{A}_{ii}$ as is described in **Algorithm 5**. In **Algorithm 5**, $\boldsymbol{U}_i$ is the change of basis matrix for $diag(\boldsymbol{v}_{A_{ii}})$. $\boldsymbol{U}_i$ is generated as follows: $diag(\boldsymbol{v}_{U_i})$ is multiplied by a randomly generated permutation matrix, and then the product is added by a perturbation matrix $\boldsymbol{E}_i$.

Algorithm 5 *createAii*

```
function CREATEAII($n_i$)
    initialize $\boldsymbol{U}_i$ to be a $n_i \times n_i$ zero matrix,
    randomly generate two $n_i \times 1$ vectors $\boldsymbol{v}_{A_{ii}}$, $\boldsymbol{v}_{U_i}$, with their elements on the open interval
(-1, 1),
    randomly shuffle vector $[1, ..., n_i]^T$ to obtain an order vector $\boldsymbol{o} \in \mathbb{R}^{n_i}$,
    for $\alpha = 1 \to n_i$ do
        $\boldsymbol{U}_{i_{\boldsymbol{o}_\alpha, \alpha}} \leftarrow \boldsymbol{v}_{U_{i\boldsymbol{o}_\alpha}}$,
    end for
    $\boldsymbol{E}_i \leftarrow createSparse(n_i, n_i, d_{E_i})$
    $\boldsymbol{U}_i \leftarrow \boldsymbol{U}_i + \boldsymbol{E}_i$
    diagonalize $\boldsymbol{v}_{A_{ii}}$ to be $\boldsymbol{D}_{A_{ii}}$,
    $\boldsymbol{A}_{ii} = \boldsymbol{U}_i \boldsymbol{D}_{A_{ii}} \boldsymbol{U}_i^{-1}$, return $\boldsymbol{A}_{ii}$
end function
```

After $\boldsymbol{A}_{ii}$'s are obtained, $\boldsymbol{A}$ is calculated by:

$$\boldsymbol{A} = \bar{\boldsymbol{A}} + \underline{\boldsymbol{A}} = diag(A_{11}, ..., A_{NN}) + \underline{\boldsymbol{A}}.$$

[1]In our experiments, since the sizes were the varied parameter, we tried to keep the interaction strength in every repitition to be the same; however, the measurement of interaction strength is itself a difficult problem that does not have a commonly accepted result. So in practice we uses the sparse density to represent the strength of interaction. This sparse density presented here is obtained by trials and errors.

If this $\boldsymbol{A}$ is found to be stable, this $\boldsymbol{A}$ and $\boldsymbol{B}$ will be used to perform the CDMPC algorithm; if not, another set of $\boldsymbol{A}$ and $\boldsymbol{B}$ will be generated until a stable $\boldsymbol{A}$ is found.

Table C.1: System Parameters

Name	**Dimension**	**Description**
N	scalar	The number of subsystems, either determined by the mode in Experiment 1, or fixed to be 80 in Experiment 2
n	scalar	The number of states for the entire plant, in the experiments the relationship $n = 2N$ always exists
q	scalar	The number of inputs for the entire plant, in the experiments the relationship $q = n$ always exists
$n_i, i = 1, ..., N$	scalar	The number of states for the i^{th} subsystem, $\sum_i^N n_i = n$; either fixed to be 2 in Experiment 1, or determined by the mode in Experiment 2
$q_i, i = 1, ..., N$	scalar	The number of inputs for the i^{th} subsystem,$\sum_i^N q_i = q$; in the experiments the relationship $q_i = n_i$ always exists
$\boldsymbol{A}$	$n \times n$ matrix	The state matrix for the entire plant, defined in equation (2.3)
$\boldsymbol{B}$	$n \times q$ matrix	The input matrix for the entire plant, defined in equation (2.3)
$\boldsymbol{A}_{ii}, i = 1, ..., N$	$n_i \times n_i$ matrix	The state matrix for the i^{th} subsystem, defined in equation (2.11b)
$\boldsymbol{B}_{ii}, i = 1, ..., N$	$n_i \times q_i$ matrix	The input matrix for the i^{th} subsystem, defined in equation (2.11b)
$\boldsymbol{A}_{ij}, i \neq j$	$n_i \times n_j$ matrix	The state interaction matrix from the j^{th} to the i^{th} subsystem, defined in equation (2.11b)
$\boldsymbol{B}_{ij}, i \neq j$	$n_i \times q_j$ matrix	The input interaction matrix from the j^{th} to the i^{th} subsystem, defined in equation (2.11b)

Table C.2: Local MPC Design Parameters

Name	**Dimension**	**Description**
H_p	scalar	Prediction horizon, fixed to be 2 in the experiments
H_u	scalar	Control horizon, fixed to be 1 in the experiments
$X_{i_{sp}}, i = 1, ..., N$	$(H_p n_i) \times 1$ vector	The state set-point trajectory for the i^{th} subsystem throughout prediction horizon, fixed to be $\mathbf{0}$ in the experiments
$\boldsymbol{Q}_i, i = 1, ..., N$	$(H_p n_i) \times (H_p n_i)$ matrix	The weighting matrix of X_i, always diagonal in the experiments where each of the diagonal element is randomly generated and uniformly distributed on the interval $(0, 1000)$
$\boldsymbol{R}_i, i = 1, ..., N$	$(H_u q_i) \times (H_u q_i)$ matrix	The weighting matrix of ΔU_i, always diagonal in the experiments where each of the diagonal element is randomly generated and uniformly distributed on the interval $(0, 1000)$
$\boldsymbol{x}_i(k), i = 1, ..., N$	$n_i \times 1$ vector	The initial value of the states in the i^{th} subsystem at time k, arbitrarily assigned to be $\mathbf{1}$ in the experiments
$\boldsymbol{u}_i(0), i = 1, ..., N$	$q_i \times 1$ vector	The initial value of the inputs in the i^{th} subsystem at time k, arbitrarily assigned to be $\mathbf{0}$ in the experiments

Table C.3: CDMPC Design Parameters

Name	**Dimension**	**Description**
ϵ	scalar	The CDMPC methods' accuracy threshold, $\epsilon = 10^{\nu}$, where ν is randomly generated and uniformly distributed on the set $\{-1, -2, -3, -4, -5\}$
$X^{(0)}$	$(H_p n) \times 1$ vector	The initialization of the coordinator predicted states, arbtrarily assigned to be $\mathbf{0}$ in the experiments
$\Delta U^{(0)}$	$(H_u q) \times 1$ vector	The initialization of the coordinator predicted input changes, arbtrarily assigned to be $\mathbf{0}$ in the experiments
$\boldsymbol{p}^{(0)}$	$(H_p n) \times 1$ vector	The initialization of the price vector, arbtrarily assigned to be $\mathbf{0}$ in the experiments

Bibliography

Al-Gherwi, W., H. Budman and A. Elkamel (2011). A robust distributed model predictive control algorithm. *Journal of Process Control* **21**, 1127–1137.

Anton, H. (2006). *Elementary Linear Algebra*. Wiley.

Aske, E. M. B, S. Strand and S. Skogestad (2008). Coordinator mpc for maximizing plant throughput. *Computers and Chemical Engineering* **32**, 195–204.

Boyd, S.P. and L. Vandenberghe (2004). *Convex Optimization*. Cambridge.

Brassard, G. and P. Bratley (1996). *Fundamentals of Algorithmics*. Prentice Hall.

Camponogara, E., D. Jia, B. H. Krogh and S. Talukdar (2002). Distributed model predictive control. *IEEE Control Systems Magazine* **22**, 44–52.

Cheng, R. (2007). Decomposition and Coordination of Large-scale Operations Optimization. PhD thesis. University of Alberta.

Cheng, R., J. F. Forbes and W. S. Yip (2007). Price-driven coordination method for solving plant-wide mpc problems. *Journal of Process Control* **17**, 429–438.

Christofides, P. D., J. Liu and D. Muñoz de la Peña (2011). *Networked and Distributed Predictive Control: Methods and Nonlinear Process Network Applications*. Advances in Industrial Control Series. Springer-Verlag, London, England. (230 pages).

Christofides, P. D., R. Scattolini, D. Muñoz de la Peña and J. Liu (2013). Distributed model predictive control: A tutorial review and future research directions. *Computers & Chemical Engineering* **51**, 21–41.

Cohen, G. (1977). On an algorithm of decentralized optimal control. *Journal of Mathematical Analysis and Applications* **59**(2), 242–259.

Cormen, T. H. (2009). *Introduction to Algorithms*. MIT Press.

Goldreich, Oded (2008). *Computational Complexity: A Conceptual Perspective*. Cambridge University Press.

Gondzio, J. (2012). Interior point methods 25 years later. *European Journal of Operational Research* **218**, 587–601.

Horn, R. A. and C. R. Johnson (2012). *Matrix Analysis*. Cambridge University Press.

Larson, R. and D. C. Falvo (2012). *Elementary Linear Algebra*. Brooks Cole.

Liu, J., D. Muñoz de la Peña and P. D. Christofides (2009). Distributed model predictive control of nonlinear process systems. *AIChE Journal* **55**, 1171–1184.

Liu, J., D. Muñoz de la Peña and P. D. Christofides (2010*a*). Distributed model predictive control of nonlinear systems subject to asynchronous and delayed measurements. *Automatica* **46**, 52–61.

Liu, J., X. Chen, D. Muñoz de la Peña and P. D. Christofides (2010*b*). Sequential and iterative architectures for distributed model predictive control of nonlinear process systems. *AIChE Journal* **56**, 2137–2149.

Liu, J., X. Chen, D. Muñoz de la Peña and P. D. Christofides (2012). Iterative distributed model predictive control of nonlinear systems: Handling asynchronous, delayed measurements. *IEEE Transactions on Automatic Control* **57**, 528–534.

Lu, J.Z. (2003). Challenging control problems and emerging technologies in enterprise optimization. *Control Engineering Practice* **11**, 847–858.

Maciejowski, J. M. (2002). *Predictive Control with Constraints*. Prentice Hall.

Maestre, J. M., D. Muñoz de la Peña and E. F. Camacho (2011). Distributed model predictive control based on a cooperative game. *Optimal Control Applications and Methods* **32**, 153–176.

Mahmoud, M. S. (1977). Multilevel systems control and applications: A survey. *Systems, Man and Cybernetics, IEEE Transactions on* **7**(3), 125–143.

Mahmoud, M. S., W. G. Vogt and M. H. Mickle (1977). Multilevel control and optimization using generalized gradients technique. *International Journal of Control* **25**(4), 525–543.

Marcos, N. (2011). Coordinated-Distributed Optimal Control of Large-Scale Linear Dynamic Systems. PhD thesis. University of Alberta.

Mesarovic, M. D., D. Macko and Y. Takahara (1970). *Theory of Hierarchical, Multilevel Systems*. Academic Press.

Mohseni, P. G. (2013). Coordination Techniques for Distributed Model Predictive Control. PhD thesis. University of Alberta.

Nash, S. G. and A. Sofer (1996). *Linear And Nonlinear Programming*. McGraw-Hill.

Quarteroni, A., R. Sacco and F. Saleri (2000). *Numerical Mathematics*. Springer.

Rawlings, J. B. and B. T. Stewart (2008*a*). Coordinating multiple optimization-based controllers: New opportunities and challenges. *Journal of Process Control* **18**, 839–845.

Rawlings, J.B. and B.T. Stewart (2008*b*). Coordinating multiple optimization-based controllers: New opportunities and challenges. *Journal of Process Control* **18**, 839–845.

Scattolini, R. (2009). Architectures for distributed and hierarchical model predictive control - A review. *Journal of Process Control* **19**, 723–731.

Scheu, H. and W. Marquardt (2011). Sensitivity-based coordination in distributed model predictive control. *Journal of Process Control* **21**, 715–728.

Stewart, B. T., S. J. Wright and J. B. Rawlings (2011). Cooperative distributed model predictive control for nonlinear systems. *Journal of Process Control* **21**, 698–704.

Stewart, B.T., A. N. Venkat, J. B. Rawlings, S. J. Wright and G. Pannocchia (2010). Cooperative distributed model predictive control. *Systems and Control Letters* **59**, 460–469.

Sun, Y. and N. H. El-Farra (2008). Quasi-decentralized model-based networked control of process systems. *Computers and Chemical Engineering* **32**, 2016–2029.

Tippett, M. J. and J. Bao (2013). Distributed model predictive control based on dissipativity. *AIChE Journal* **59**, 787–804.

Venkat, A. N., J. B. Rawlings and S. J. Wright (2005). Stability and optimality of distributed model predictive control. In: *Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference ECC 2005*. Seville, Spain. pp. 6680–6685.

Wang, L. (2009). *Model Predictive Control System Design and Implementation Using MATLAB®*. Springer.

Wilf, Herbert S. (2002). *Algorithms and Complexity*. second edtion ed.. A.K. Peters.

Zhang, F. (2005). *The Schur Complement and Its Applications*. Springer.