UNIVERSITY OF SOUTHAMPTON

Algorithms for analysis and synthesis of systems with synchronization errors

by

Marek Przedwojski

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

April 2012

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Marek Przedwojski

This thesis addresses currently open problems in the stability analysis and control of discrete linear systems with clock synchronisation errors. Such errors can lead to instability of an overall system even in the case when it is composed of linear sub-systems that are stable. Previous work in this general area has focused almost exclusively on stability analysis and this thesis therefore focuses on the synthesis problem of how to design control laws that ensure stability and performance of the overall system in the presence of clock synchronisation errors and, in particular, on the robustness problem. For many applications, intensive matrix computations are required and hence the time complexity of the algorithms used is critical. A major part of the new results in this thesis is the development of two new algorithms for undertaking the computations in the case where uncertainty is present and an investigation of their merits relative to linear matrix inequality and brute force alternatives. An identification method for detecting the presence of clock synchronisation errors from system data is also developed.

Contents

Acknowledgements **vii**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 General description . . . 7
1.3 Organization of the Thesis . . . 9

2 Models for synchronization errors in linear systems and open research questions **11**
2.1 Centralized control . . . 11
2.1.1 The common clock case . . . 11
2.1.2 The case of different clocks . . . 22
2.2 Decentralized control . . . 23
2.3 On the stability of systems with synchronization errors . . . 26
2.3.1 Preliminaries . . . 26
2.3.2 Asynchronous algorithms . . . 27
2.3.3 The control problem . . . 28

3 Stabilization using a polytopic uncertainty setting **33**
3.1 Introduction . . . 33
3.2 Stability analysis . . . 33
3.3 Estimation of the polytope uncertainty . . . 36
3.3.1 Preliminaries . . . 36
3.3.2 Convex hull algorithms . . . 38
3.3.3 Minimum Volume Enclosing Ellipsoids . . . 42
3.3.4 Choosing the vertices of the polytope . . . 59
3.4 Relaxed LMI conditions . . . 76
3.4.1 Stability . . . 77
3.4.2 Relaxed conditions with reduced conservativeness . . . 79
3.5 Disturbance attenuation control . . . 81
3.5.1 Introduction . . . 81
3.5.2 Controller design . . . 84
3.5.3 Numerical tests . . . 87
3.6 Conclusions . . . 88

4 Stabilization using a norm bounded uncertainty setting **89**
4.1 Introduction . . . 89
4.2 Norm bounded uncertainty analysis . . . 90

4.3 Estimation of the norm bounded uncertainty . . . 95
4.3.1 The measure of norm bounded uncertainty . . . 95
4.3.2 Outer and inner approximation of the uncertainty . . . 100
4.3.3 LMI based method . . . 102
4.3.4 The new method . . . 105
4.3.5 Numerical tests . . . 112
4.3.6 Conclusions . . . 114

5 Estimation of synchronization errors in the common clock case 115
5.1 Introduction . . . 115
5.2 Preliminaries . . . 115
5.3 Estimation of synchronization errors . . . 116
5.4 Identifiability of clock synchronization errors . . . 118
5.5 Example . . . 120
5.6 Conclusions . . . 121

6 Representative applications 123
6.1 Introduction . . . 123
6.2 Multi-agent systems . . . 123
6.3 Application to iterative learning control with synchronization errors . . . 127
6.4 Application to solving specific LMI control problems . . . 135
6.5 Conclusions . . . 140

7 Conclusions 141
7.1 Novel contributions . . . 141
7.2 Directions for future research . . . 143

A Iterative learning control for systems with uncertain dynamics 147

Bibliography 153

Acknowledgements

This thesis would not have been possible without the help, support and patience of my principal supervisor, Professor Eric Rogers. Also his sense of humor has been of great help in overcoming many of the difficulties.

I am truly grateful to my second supervisor, Dr Ivan Markovsky, for encouraging and personal guidance during the research. His support, valuable advice and inspiring discussion, both on practical and theoretical matters, have been invaluable to me.

I would like also to express my grateful thanks to Professor Krzysztof Gałkowski, who initiated the idea of this research project.

Lastly, I offer my regards and thanks to all of those who supported me in any respect during the completion of the project.

Chapter 1

Introduction

Advances in the underlying technologies, such as electronics and communications, have enabled the design and implementation of complex systems composed of many sub-systems and driven by a clock frequency. This is also one of the reasons why digital control plays such a prominent role in control systems theory, design and implementation and often the assumption made is that the states of the sub-systems change at the same time instances. However, in large scale systems and/or high speed circuitry this assumption may be violated and signal propagation delays arise. These, in turn, can cause sub-systems to change their states at different instances of time and hence the presence of clock synchronization errors that alter the overall system behavior such that the designed system is not actually implemented.

One more recent area where clock synchronization errors can arise is in the analysis of models of swarms, which may be biologically inspired. A group of autonomous systems can be modeled as interconnected sub-systems and if some or all of these operate at a different clock frequency another form of clock synchronization errors can arise. The implications of clock synchronization errors on systems is the subject area of this thesis and the remainder of this chapter gives a general level introduction and describes the layout of the following chapters.

1.1 Motivation

Digital systems are driven by a clock as illustrated in Figure 1.1. Subsystems change their state (switches) driven by the rising or falling edge of the clock signal. In the ideal situation switching of all the subsystems occurs simultaneously and instantaneously (synchronously). If, however, this assumption is violated then the clock signal may reach the subsystems at different time instances and thereby cause them not to change their states simultaneously.

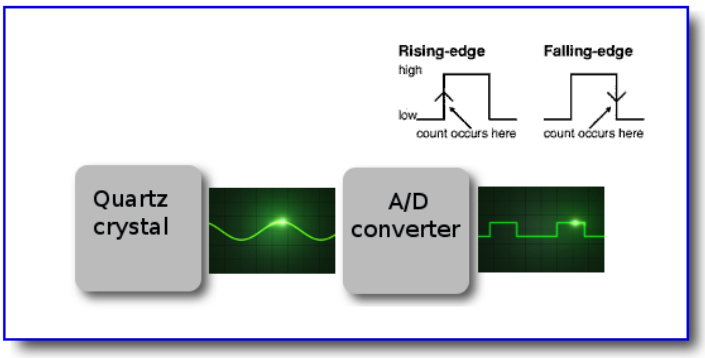

FIGURE 1.1: Typical system clock for digital systems.

One case where this problem can arise is high clock speed circuitry due to signal propagation delays and is known as the common clock case. A more general scenario involves different clocks with different rates feeding the interconnected subsystems that is known as the different clocks case. Asynchronous switching of subsystems has a very significant effect on the overall system response and can destabilize a stable system, see Kleptzyn et al. (1984). It is also important to distinguish these cases with instantaneous switching from *non-atomic* switching, where the switching of a subsystem occurs in non-negligible time. Stability in the first case was investigated by Lorand (2004). Next, some examples are given.

Discrete-time market models.
We consider the situation of two countries A and B which each have a stock exchange. Let x_A denote the price index in A and x_B that in B country. If there is no connection between these two countries the price indices obey

$$x_A(k+1) = x_A(k) + d_A(k), \quad x_B(k+1) = x_B(k) + d_B(k), \quad k = 0, 1, 2, \ldots,$$

where the time index k is in days and d_A, d_B are random walks, i.e. independent random variables $d_A(0), d_A(1), \ldots$ and $d_B(0), d_B(1), \ldots$, which represent stochastic fluctuations of the indices. In reality, the nature of the economies in each country influences that in the other. Consequently suppose that if the price index changes by 10% in country A then that in country B changes by 5%. Suppose also that if the price index changes by 10% in country B the corresponding change in country A is 1%. Then the following is a model for these cases

$$\begin{cases} x_A(k+1) = x_A(k) + 0.1 \cdot \frac{x_B(k) - x_B(k-1)}{x_B(k-1)} \cdot x_A(k) + d_A(k) = f_A(x_A, x_B) \\ x_B(k+1) = x_B(k) + 0.5 \cdot \frac{x_A(k) - x_A(k-1)}{x_A(k-1)} \cdot x_B(k) + d_B(k) = f_B(x_a, x_B) \end{cases},$$

which is valid if the countries update the prices *synchronously* (at the same point in time). Synchronous price updating is unlikely to arise in the real world due, amongst other causes, to global time shifts. In particular, suppose that the stock exchange in country A operates in daylight hours and that in country B in night hours. Suppose

also, for simplicity, that the prices are updated once a day and that odd k correspond to night hours in country B. Then the model becomes

$$\begin{bmatrix} x_A(k+1) \\ x_B(k+1) \end{bmatrix} = \begin{cases} \begin{bmatrix} x_A(k) \\ f_B(x_A(k), x_B(k)) \end{bmatrix}, & k \text{ is even} \\ \begin{bmatrix} f_A(x_A(k), x_B(k)) \\ x_B(k) \end{bmatrix}, & k \text{ is odd} \end{cases}.$$

Such a system is termed asynchronous.

Asynchronous algorithms.

Consider a system of linear equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{1.1}$$

for some matrix

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n1} & \dots & \dots & a_{nn} \end{bmatrix} \in \mathbb{R}^n$$

and vectors $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^n$. Now let

$$\boldsymbol{D} = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & a_{nn} \end{bmatrix}$$

and $\boldsymbol{E} = \boldsymbol{D} - \boldsymbol{A}$ and consider the following iterative algorithm assuming $\boldsymbol{D}$ invertible

$$\boldsymbol{x}(k+1) = \boldsymbol{D}^{-1}\boldsymbol{E}\boldsymbol{x}(k) + \boldsymbol{D}^{-1}\boldsymbol{b} = \boldsymbol{A}'\boldsymbol{x}(k) + \boldsymbol{b}'. \tag{1.2}$$

If the algorithm converges the limit is the fixed point $\boldsymbol{z} \in \mathbb{R}^n$

$$\boldsymbol{z} = \boldsymbol{D}^{-1}\boldsymbol{E}\boldsymbol{z} + \boldsymbol{D}^{-1}\boldsymbol{b},$$

which is also a solution of the system (1.1). The algorithm (1.2) is termed *synchronous* if the whole state vector $\boldsymbol{x}$ is created at the same time instance. However, in parallel computing this may not be true and the state vector entries may be updated at different moments of time leading to *asynchronous algorithms*.

Assume that n processors perform computations independently. The ith processor computes only part x_i of the state vector $\boldsymbol{x}$ which is held in a shared memory. The processors have access to the full state but may operate at different clock rates T_i, and let the time

to compute x_i be denoted by δt_i. This general situation is illustrated in Figure 1.2.

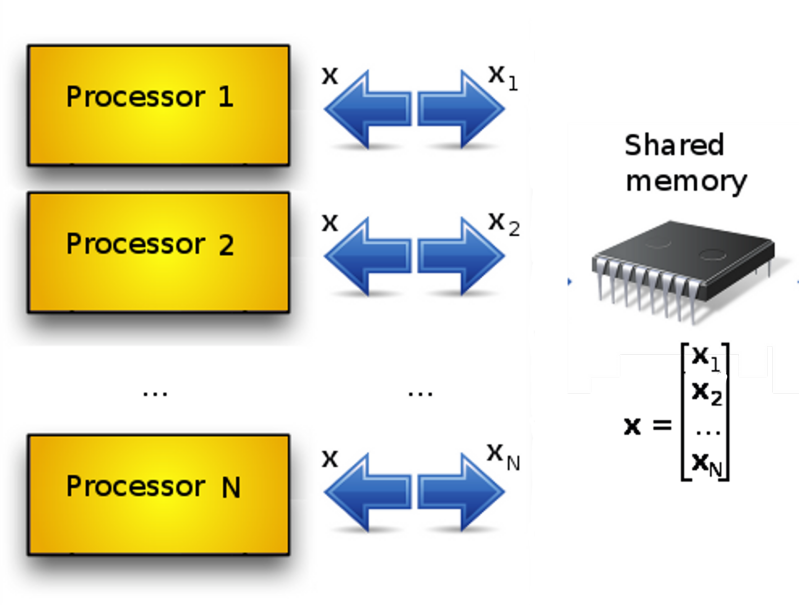

FIGURE 1.2: Asynchronous algorithms

Suppose that the time δt_i is negligible and consider again (1.2). Introduce the time index p that is incremented by one whenever an updating occurs. Then the evolution of $\boldsymbol{x}(k)$ may be described by the following model

$$x_i(p+1) = \begin{cases} x_i(p) & p \notin i(p) \\ \sum_{j=1}^{n} a'_{ij} x_j(p) + b'_j & p \in i(p) \end{cases},$$

where $i(p) \subseteq \{1, \ldots, n\}$ describes which indices update simultaneously during pth event and a'_{ij}, $b'j$ denote the entries in $\boldsymbol{A}'$ and $\boldsymbol{b}'$ respectively. However, the assumption that the calculation of the state vector entry takes no time is unrealistic. In general $\delta t_i(p) > 0$ if $i \in i(p)$. If we assume that the time index p is incremented by one whenever the calculation starts then the model should involve delays because the new value is not available to the other processors immediately and another calculation may start using a past value of the state vector. Similarly, if the time index p is incremented whenever the computation finishes and the new value is available to the other processors then in the hypothetical situation a calculation may start before another, already started, finishes. The situation is illustrated in Figure 1.3.

Regardless of the moments of time chosen as the incrementation of the time index p the general model in this case is

$$x_i(p+1) = \begin{cases} x_i(p) & p \notin i(p) \\ \sum_{j=1}^{n} a'_{ij} x_j(p - d(p)) + b'_j & p \in i(p) \end{cases},$$

for some bounded delays $0 \leq d(p) \leq d$.

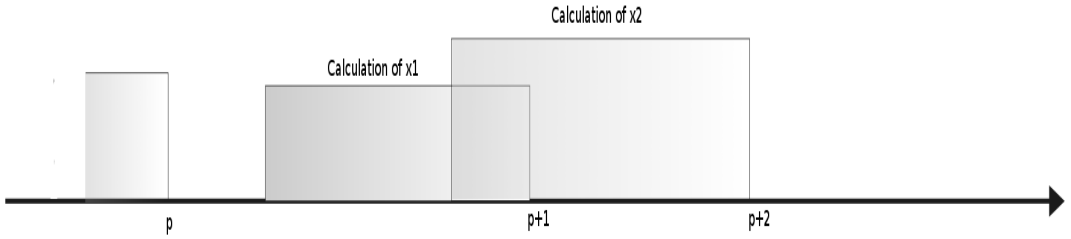

FIGURE 1.3: Switching of subsystems with non-negligible switching times. For the calculation of $x_2(p+2)$ the value of $\boldsymbol{x}(p)$ is used instead of $\boldsymbol{x}(p+1)$.

The stability of *asynchronous algorithms* have been extensively studied since the 1960s due to its importance in parallel computation. The first stability results were published by Chazan and Miranker (1969), see also the monographs by Bertsekas and Tsitsiklis (1989) and Asarin et al. (1992).

Asynchronous Control

Assume that a digital controller operates with clock period T_1 and the plant changes state (switches) with period T_2. Here we assume that the switching is instantaneous and, for simplicity, that $T_2 = 2/3 \cdot T_1$. We also assume that they are out of phase and hence they never update synchronously. The updating scheme is illustrated in Figure 1.4.

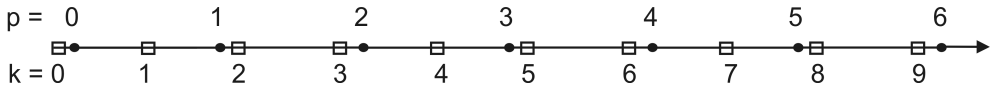

FIGURE 1.4: Plant and controller updating asynchronously. □ - plant updates, • - input updates.

Assume the nominal system model is

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

and consider the updating at time index k which is incremented by one each time a

switching of occurs. Hence

$$\begin{aligned}
\boldsymbol{x}(1) &= \boldsymbol{A}\boldsymbol{x}(0) + \boldsymbol{B}\boldsymbol{u}(0) \\
\boldsymbol{x}(2) &= \boldsymbol{A}\boldsymbol{x}(1) + \boldsymbol{B}\boldsymbol{u}(1) \\
\boldsymbol{x}(3) &= \boldsymbol{A}\boldsymbol{x}(2) + \boldsymbol{B}\boldsymbol{u}(2) = \boldsymbol{A}\boldsymbol{x}(2) + \boldsymbol{B}\boldsymbol{u}(1) \\
\boldsymbol{x}(4) &= \boldsymbol{A}\boldsymbol{x}(3) + \boldsymbol{B}\boldsymbol{u}(3) \\
\boldsymbol{x}(5) &= \boldsymbol{A}\boldsymbol{x}(4) + \boldsymbol{B}\boldsymbol{u}(4) \\
\boldsymbol{x}(6) &= \boldsymbol{A}\boldsymbol{x}(5) + \boldsymbol{B}\boldsymbol{u}(5) = \boldsymbol{A}\boldsymbol{x}(5) + \boldsymbol{B}\boldsymbol{u}(4) \\
\boldsymbol{x}(7) &= \boldsymbol{A}\boldsymbol{x}(6) + \boldsymbol{B}\boldsymbol{u}(6) \\
\boldsymbol{x}(8) &= \boldsymbol{A}\boldsymbol{x}(7) + \boldsymbol{B}\boldsymbol{u}(7) \\
\boldsymbol{x}(9) &= \boldsymbol{A}\boldsymbol{x}(8) + \boldsymbol{B}\boldsymbol{u}(8) = \boldsymbol{A}\boldsymbol{x}(8) + \boldsymbol{B}\boldsymbol{u}(7) \\
\ldots &= \ldots \quad .
\end{aligned}$$

Starting from $k = 2$ for every 3 time increments the input does not update. Introduce the time index p that is incremented whenever the input updates. From the controller point of view we have

$$\boldsymbol{x}(p=1) = \boldsymbol{x}(k=1), \quad \boldsymbol{x}(p=2) = \boldsymbol{x}(k=3), \quad \boldsymbol{x}(p=3) = \boldsymbol{x}(k=4), \quad \boldsymbol{x}(p=4) = \boldsymbol{x}(k=6)$$
$$\boldsymbol{x}(p=5) = \boldsymbol{x}(k=7), \quad \boldsymbol{x}(p=6) = \boldsymbol{x}(k=9), \ldots$$

and hence for time index p

$$\begin{aligned}
\boldsymbol{x}(1) &= \boldsymbol{A}\boldsymbol{x}(0) + \boldsymbol{B}\boldsymbol{u}(0) \\
\boldsymbol{x}(2) &= \boldsymbol{A}^2\boldsymbol{x}(1) + \boldsymbol{B}\boldsymbol{u}(1) \\
\boldsymbol{x}(3) &= \boldsymbol{A}\boldsymbol{x}(2) + \boldsymbol{B}\boldsymbol{u}(2) \\
\boldsymbol{x}(4) &= \boldsymbol{A}^2\boldsymbol{x}(3) + \boldsymbol{B}\boldsymbol{u}(3) \\
\boldsymbol{x}(5) &= \boldsymbol{A}\boldsymbol{x}(4) + \boldsymbol{B}\boldsymbol{u}(4) \\
\boldsymbol{x}(6) &= \boldsymbol{A}^2\boldsymbol{x}(5) + \boldsymbol{B}\boldsymbol{u}(5) \\
\boldsymbol{x}(7) &= \boldsymbol{A}\boldsymbol{x}(6) + \boldsymbol{B}\boldsymbol{u}(6) \\
\ldots &= \ldots \quad .
\end{aligned}$$

The controller has to stabilize a switched system and not the nominal plant. In the general case there will be a series of interconnected plants operating on different clock frequencies and a controller that is fed by its own clock. This situation is much more complicated and a general model is derived next.

1.2 General description

In the ideal situation the discrete-time system with no input satisfies

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k)) \quad ,$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector and $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear operator. The time index k represents discrete moments of time $\{T, 2T, \ldots, kT, \ldots\}$ in which the state of the system is changing. However, the assumption that the whole state vector is changing at one point of time may sometimes be too strong to describe the dynamics of the system. In the general setting we assume that the system consists of N subsystems. We assume also that the subsystems correspond to the parts of state vector

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix}$$

and that each subsystem $\boldsymbol{x}_i$ is driven by a clock with a period T_i, $i = 1, \ldots, N$. Now all the subsystems switch or update at discrete instances defined by the sequence $\{T_i, 2T_i, \ldots, kT_i, \ldots\}$ and these sequences may be different for different subsystems. By the updating (or switching) of a subsystem we mean that the corresponding part of the state vector is being recalculated to take new values. We also assume that switching occurs instantaneously, see Figure 1.5.

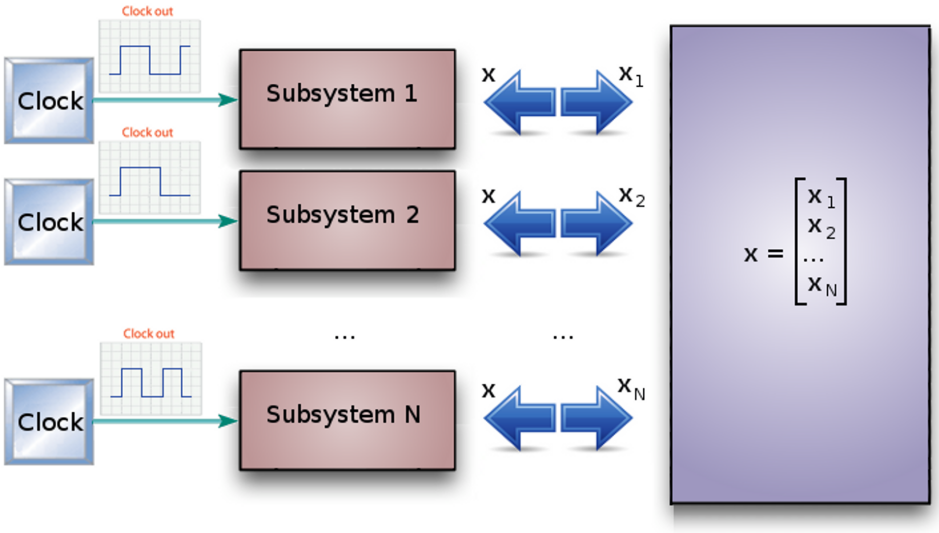

FIGURE 1.5: Each subsystem is triggered by its own clock. Driven by the edge of the clock signal a subsystem recalculates the corresponding part of the state vector. All subsystems update at a different point in time. The whole state vector is available to all of the subsystems.

Suppose, for simplicity, that $\boldsymbol{x} \in \mathbb{R}^2$ and there are two subsystems corresponding to x_1 and x_2 respectively. Assume also that the clock periods are equal $T_1 = T_2$ but are out of phase in the sense that x_1 always updates always before x_2. The updating will be defined by the sequence

$$\{T_1, T_2, 2T_1, 2T_2, , \dots\}$$

and we introduce the time index p whenever an updating occurs. It is convenient to introduce the following notation. For the operator $\boldsymbol{f} : \mathbb{R}^2 \mapsto \mathbb{R}^2$ we define $\boldsymbol{f}_{\{1\}}$ and $\boldsymbol{f}_{\{2\}}$ as

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \end{bmatrix}, \quad \boldsymbol{f}_{\{1\}}(\cdot) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ x_2 \end{bmatrix}, \quad \boldsymbol{f}_{\{2\}}(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ f_2(\boldsymbol{x}) \end{bmatrix}.$$

In general for $f : \mathbb{R}^n \to \mathbb{R}^n$

$$\boldsymbol{f}_{\{\dots,i,\dots,j,\dots\}}(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ \dots \\ f_1(\boldsymbol{x}) \\ \dots \\ f_j(\boldsymbol{x}) \\ \dots \\ x_n \end{bmatrix}, \quad \{\dots, i, \dots, j, \dots\} \subseteq \{1, \dots, n\}$$

and

- at time T_1

$$\boldsymbol{x}(1) = \begin{bmatrix} f_1(\boldsymbol{x}(0)) \\ x_2(0) \end{bmatrix} = \boldsymbol{f}_{\{1\}}(\boldsymbol{x}(0)).$$

- at time T_2

$$\boldsymbol{x}(2) = \begin{bmatrix} x_1(1) \\ f_2(\boldsymbol{x}(1)) \end{bmatrix} = \boldsymbol{f}_{\{2\}}(\boldsymbol{x}(1)) = (\boldsymbol{f}_{\{2\}} \circ \boldsymbol{f}_{\{1\}})(\boldsymbol{x}(0)).$$

- ...

- at time $(p+1)T_1$

$$\boldsymbol{x}(2p+1) = \begin{bmatrix} f_1(\boldsymbol{x}(2p)) \\ x_2(2p) \end{bmatrix} = (\boldsymbol{f}_{\{1\}} \circ \underbrace{\boldsymbol{f}_{\{2\}} \circ \boldsymbol{f}_{\{1\}} \circ \cdots \circ \boldsymbol{f}_{\{2\}} \circ \boldsymbol{f}_{\{1\}}}_{p \text{ times}})(\boldsymbol{x}(0)).$$

- ...

The system that results is completely different from its original synchronous counterpart where

$$\boldsymbol{x}(2p+1) = (\underbrace{\boldsymbol{f} \circ \cdots \circ \boldsymbol{f}}_{2p+1 \text{ times}})(\boldsymbol{x}(0)).$$

Moreover, the response may be drastically different and a stable system can become unstable under an asynchronous updating regime.
In general a *system with clock synchronization errors* is a discrete-time system with original dynamics

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k)),$$

which updates according to

$$\boldsymbol{x}(p+1) = \boldsymbol{f}_{i(p)}(\boldsymbol{x}(p)), \qquad i(p) \subseteq \{1, \ldots, n\}.$$

Assuming linear dynamics we obtain

$$\boldsymbol{x}(p+1) = \boldsymbol{A}_{i(p)}\boldsymbol{x}(p-1), \qquad i(p) \subseteq \{1, \ldots, n\},$$

for some matrix $\boldsymbol{A}$. For the case of $i(p) = \{\ldots, p, \ldots, q, \ldots\}$ the matrix $\boldsymbol{A}_{i(p)}$ is defined as

$$\boldsymbol{A}_{i(p)} = \begin{bmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{p1} & a_{p2} & \ldots & a_{p(n-1)} & a_{pn} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{q1} & a_{q2} & \ldots & a_{q(n-1)} & a_{qn} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{bmatrix},$$

where a_{ij} are the entries of $\boldsymbol{A}$.

1.3 Organization of the Thesis

The problem of interconnected systems with clock synchronization errors has attracted relatively little attention in the literature. The first work by Kleptzyn et al. (1984) introduced a model and showed that even small synchronization errors can affect overall system stability. In more recent work Lorand (2004) studied the stability properties of different types of synchronization errors and in Lorand and Bauer (2005) a model was proposed for a distributed system with different clock frequencies and its stability properties analyzed. A Toeplitz operator approach was used in Lorand and Bauer (2006b) to address the problem of different clock frequencies in networked systems. Also a method of identifying clock synchronization errors from the system output in the presence of a common clock was developed. The overall aim of this thesis is to develop algorithms for the analysis and control, or synthesis, of interconnected systems with clock synchronization errors.

In Chapter 2 a new model for studying clock synchronization errors in interconnected systems is developed that differs from that used previously in that the structure is not a priori assumed. This chapter then gives the results of a stability analysis of this new model and in Chapter 3 stabilization by state feedback is developed by formulating the general problem in terms of the computation of approximate polytopic uncertainty. Chapter 4 gives the results of using norm bounded uncertainty in this problem domain and Chapter 5 a method for estimating clock synchronization errors for systems with a common clock. Chapter 6 considers the application of the developed methods to three different areas and Chapter 7 gives the main conclusions of this thesis together with directions for future research.

Chapter 2

Models for synchronization errors in linear systems and open research questions

In this chapter models to represent the effects of synchronization errors in linear systems are developed for both the centralised and decentralised control cases. The differences with previous work are explained and the chapter concludes with the introduction of the open research questions addressed in the remainder of the thesis.

2.1 Centralized control

In this case we assume that a common input signal is fed to subsystems. The input signal updates according to its own clock T independently of the subsystems clocks. This is the most common possible implementation in which there is a single digital controller that controls the whole system, see Figure 2.1.

2.1.1 The common clock case

We consider discrete-time linear time-invariant systems that are decomposed into subsystems. Each subsystem changes its state (switches) at certain time instances defined by rising or falling edge of a clock signal. In the case of a common clock the clock signals have the same frequency but could be out of phase, mainly due to signal propagation delays (in distributed systems). A different case arises in multi-agent systems where each agent is driven by a clock of common frequency but with a different phase shift. Asynchronous switching leads to different system behaviour than in the synchronous case. The subsystems switch at different points of time but the number of switching

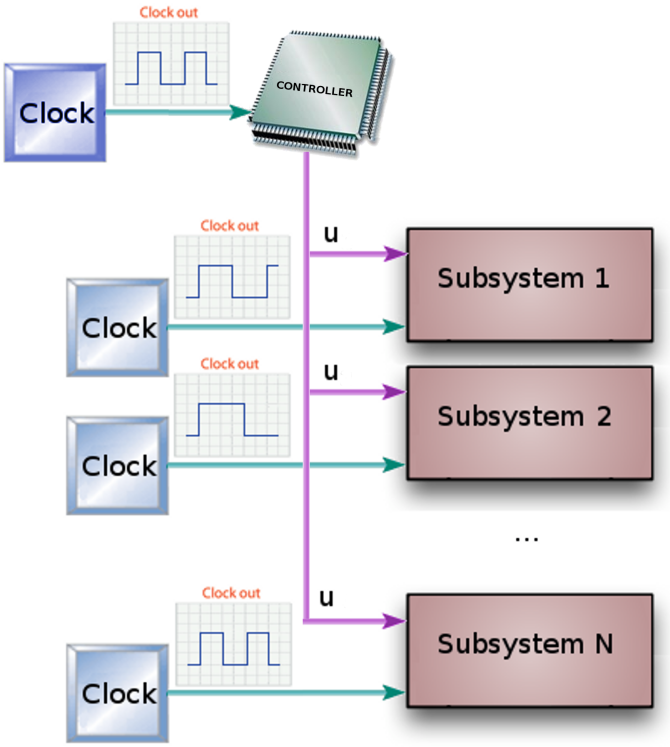

FIGURE 2.1: The common controller case.

events over a full clock period is constant. Figure 2.2 shows an example time-line for two systems: synchronous and asynchronous. Updating events are marked by symbols – triangles, squares and circles.

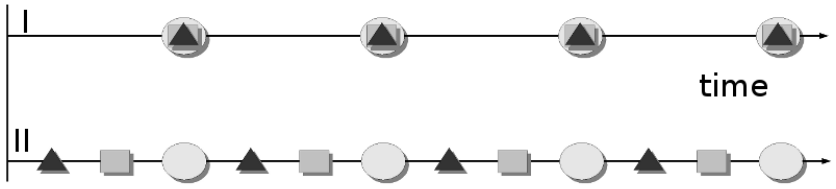

FIGURE 2.2: Switching event pattern for synchronous and asynchronous systems. Symbols: ▲–x_1 updates, □–x_2 updates, ◯–input updates. Systems: I– synchronous system, II–system with synchronization error.

The switching event pattern is periodic due to constant phase shifts (If a clock signal reaches a subsystem with small delay, this delay is constant). This simplifies the analysis and results for what is the simplest case among all types of synchronization errors. Next a model for this case is developed.

We consider discrete-time linear time-invariant systems defined by state space model

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{aligned}, \tag{2.1}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^n$, $\boldsymbol{u}(k) \in \mathbb{R}^m$, $\boldsymbol{y}(k) \in \mathbb{R}^p$ are the state vector, the input vector and the output vector at time index k respectively. The matrices $\boldsymbol{A} \in \mathbb{R}^{n\times n}, \boldsymbol{B} \in \mathbb{R}^{n\times m}, \boldsymbol{C} \in \mathbb{R}^{p\times n}, \boldsymbol{D} \in \mathbb{R}^{p\times m}$ have constant entries. We assume that the basis of the state representation is fixed and that every state vector entry x_i is fed by a clock with rate T_i. The input is driven by its own clock $\boldsymbol{T}$. The clock rates are equal but could be out of phase and the time index is incremented by one whenever the input updates. In the case when there are no synchronization errors the state variables update (i.e., new values are calculated) at the same time instances. In the presence of synchronization errors, there could be more than one event of updating over a full clock period. In this case, the state space model (2.1) is no longer valid.

To model this latter behaviour, assume first that there are d events of switching (updating) in one full clock period. The switching events are described by the sequence s of mutually disjoint subsets of indices

$$s = (i_1, i_2, \ldots, i_d), \quad i_j \subseteq \{1, \ldots, n\}, \quad j = 1, \ldots, d. \tag{2.2}$$

The subset i_k, $k = 1, \ldots, d$, contains the indices of the state variables that updated simultaneously during the k-th event. These subsets satisfy

$$p \neq r \;\Rightarrow\; i_p \cap i_r = \emptyset \quad \text{for} \quad p, r = 1, \ldots, d \tag{2.3}$$

and

$$\bigcup_{j=1}^{d} i_j = \{1, \ldots, n\}, \tag{2.4}$$

which means that all the state variables are updated over a full clock period. The set of all possible sequences that describe switching events will be denoted by $\mathcal{S}$.

Now assume that s defined in (2.2) describes the switching event pattern for the given discrete-time linear system. When the j-th event occurs the state vector updates according to

$$\boldsymbol{x}^j(k) = \boldsymbol{A}_{i_j}\boldsymbol{x}^{j-1}(k) + \boldsymbol{B}_{i_j}\boldsymbol{u}(k), \tag{2.5}$$

where $\boldsymbol{x}^0(k) = \boldsymbol{x}(k)$ and the model matrix $\boldsymbol{A}_{i_j} \in \mathbb{R}^{n\times n}$, e.g., given $i_j = \{\ldots, p, \ldots, q, \ldots\}$,

is

$$\boldsymbol{A}_{i_j} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{p(n-1)} & a_{pn} \\ \dots & \dots & \dots & \dots & \dots \\ a_{q1} & a_{q2} & \dots & a_{q(n-1)} & a_{qn} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix} \tag{2.6}$$

and

$$\boldsymbol{B}_{i_j} = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ b_{p1} & b_{p2} & \dots & b_{p(m-1)} & b_{pm} \\ \dots & \dots & \dots & \dots & \dots \\ b_{q1} & b_{q2} & \dots & b_{q(m-1)} & b_{qm} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}. \tag{2.7}$$

This representation means that during the jth event only those state vector entries that correspond to i_j are updated. We assume that $\boldsymbol{x}(k+1) = \boldsymbol{x}^d(k)$ and the full new state vector is created after all switching events. The resulting state space model is

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}_s\boldsymbol{x}(k) + \boldsymbol{B}_s\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{aligned}, \tag{2.8}$$

where the matrices $\boldsymbol{A}_s \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B}_s \in \mathbb{R}^{n\times m}$ are defined as follows

$$\boldsymbol{A}_s = \boldsymbol{A}_{i_d} \cdots \boldsymbol{A}_{i_1} \tag{2.9}$$

and

$$\boldsymbol{B}_s = \boldsymbol{B}_{i_d} + \boldsymbol{A}_{i_d}\boldsymbol{B}_{i_{d-1}} + \cdots + \boldsymbol{A}_{i_d} \cdots \boldsymbol{A}_{i_2}\boldsymbol{B}_{i_1}. \tag{2.10}$$

Example 2.1. *Consider a 2nd order system with zero input and state matrix*

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.$$

Assuming that that first entry is updated before the second, the sequence s that describes this event is

$$s = (\{1\}, \{2\}).$$

Event 1: $i_1 = \{1\}$*, and the state transition is*

$$\begin{bmatrix} x_1^1(k) \\ x_2^1(k) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} = \boldsymbol{A}_{\{1\}} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}.$$

Event 2: $i_2 = \{2\}$*, and the state transition is*

$$\begin{bmatrix} x_1^2(k) \\ x_2^2(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1^1(k) \\ x_2^1(k) \end{bmatrix} = \boldsymbol{A}_{\{2\}} \begin{bmatrix} x_1^1(k) \\ x_2^1(k) \end{bmatrix}.$$

The state vector after one clock period is

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} = \boldsymbol{A}_{\{2\}} \boldsymbol{A}_{\{1\}} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix},$$

for $k = 0, 1, \ldots$

The work by Lorand (2004) gives a similar model with the structure of the subsystems already imposed. In this model the state vector $\boldsymbol{x} = [\ \boldsymbol{x}_i\ ]_{i=1,\ldots,N}$ consists of vectors $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$, $\sum_{i=1}^{N} n_i = n$, that are relevant to the ith subsystem. The matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ (2.1) are partitioned as $\boldsymbol{A} = [\ \boldsymbol{A}_{ij}\ ]_{i,j=1,\ldots,N}$ with $\boldsymbol{A}_{ij} \in \mathbb{R}^{n_i \times n_j}$ and $\boldsymbol{B} = [\ \boldsymbol{B}_i\ ]_{i=1,\ldots,N}$ with $\boldsymbol{B}_i \in \mathbb{R}^{n_i \times m}$. In the model the sequence of subsets $s' = (i_1, \ldots, i_e)$ describes the order in which subsystems update between two consecutive input updating events. Each subset $i_j \subset \{1, \ldots, N\}$, $i = 1, \ldots, e$ determines which subsystems update simultaneously during the jth event. For given $i_j = \{\ldots, p, \ldots, q, \ldots\}$ the asynchronous equation (2.5) takes the same form as the new model in this section but the model matrices are

$$\boldsymbol{A}_{i_j} = \begin{bmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \boldsymbol{A}_{p1} & \boldsymbol{A}_{p2} & \ldots & \boldsymbol{A}_{p(n-1)} & \boldsymbol{A}_{pN} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \boldsymbol{A}_{q1} & \boldsymbol{A}_{q2} & \ldots & \boldsymbol{A}_{q(n-1)} & \boldsymbol{A}_{qN} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{B}_{i_j} = \begin{bmatrix} 0 \\ 0 \\ \ldots \\ \boldsymbol{B}_p \\ \ldots \\ \boldsymbol{B}_q \\ \ldots \\ 0 \\ 0 \end{bmatrix}. \tag{2.11}$$

Similar reasoning now yields the state model updating of the form(2.8) with $s = s'$ and

$$\boldsymbol{A}_{s'} = \boldsymbol{A}_{i_e} \cdots \boldsymbol{A}_{i_1},$$

$$\boldsymbol{B}_{s'} = \boldsymbol{B}_{i_e} + \boldsymbol{A}_{i_e}\boldsymbol{B}_{i_{e-1}} + \cdots + \boldsymbol{A}_{i_e} \cdots \boldsymbol{A}_{i_2}\boldsymbol{B}_{i_1}.$$

By appropriate selection of the set of allowable sequences $\mathcal{S}$ in the first case, i.e., forbidding some sequences, we can impose the subsystems structure that makes the two models equivalent.

The number of synchronization errors

As discussed previously, the number of sequences s for a given n-order system can be very large. To estimate this value for the common clock case, let $\mathcal{S}_n$ denote the set of allowable sequences for an n-order system. By the length of the sequence $s \in \mathcal{S}_n$ we mean the number of elements (sets) in this sequence. Let $\pi_k(n)$ denote the number of k-element sequences for an n-order system. Hence

$$\pi_1(n) = 1, \qquad n = 1, 2, \ldots,$$

since for an n-order system there is only one sequence s_1 of length 1

$$s_1 = (\{1, 2, \ldots, n\}).$$

Also for $k > 1$

$$\pi_k(n) = 0, \qquad \text{for } n < k$$

and the set S_n does not contain sequences of length greater than n. Also

$$\pi_i(n) + \pi_2(n) + \cdots + \pi_n(n) = \text{card}(\mathcal{S}_n) =: \pi(n),$$

where $\text{card}(\cdot)$ denotes the cardinal number.

Consider a sequence of length k for some n-order system

$$s = (i_1, i_2, \ldots, i_k)$$

and suppose that another state variable is added to the system. The sequence s will change in order to include the additional state, which may be placed in s either as an element of the subset i_j, $j = 1, 2, \ldots, k$, or before each i_j, or after the i_k. Thus every k-element sequence for an n-order system generates k sequences of length k and $k + 1$

sequences of length $k+1$ for $n+1$-order system. Consider the following graph

$$\begin{array}{ccccccc}
\pi_1(n) & \longrightarrow & \pi_1(n) & = & \pi_1(n+1) \\
 & \searrow & & & \\
 & & 2\pi_1(n) & & \\
\pi_2(n) & \longrightarrow & +2\pi_2(n) & = & \pi_2(n+1) \\
 & \searrow & & & \\
 & & 3\pi_2(n) & & \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
 & \searrow & \vdots & & \\
 & & k\pi_{k-1}(n) & & \\
\pi_k(n) & \longrightarrow & +k\pi_k(n) & = & \pi_k(n+1), \quad n \geq k \\
 & \searrow & & & \\
 & & (k+1)\pi_k(n) & & \\
\cdots & \cdots & \cdots & \cdots & \cdots
\end{array}$$

Hence the equations are

$$\begin{aligned}
\pi_1(n) &= 1 \qquad n = 1, 2, \ldots \\
\pi_k(n+1) &= \left\{ \begin{array}{ll} 0 & \text{for } n < k \\ k(\pi_k(n) + \pi_{k-1}(n)) & \text{for } n \geq k \end{array} \right\} \qquad \text{for } k \geq 2.
\end{aligned} \tag{2.12}$$

Equation (2.12) allows the construction of a simple algorithm evaluating the number of synchronization errors $\pi(n)$ for n-order system. The steps are as follows

- **Step 1:** Start with the vector

$$\boldsymbol{p}(1) = [\,\pi_1(1)\,] = [\,1\,].$$

- **Step 2: for each** $k = 2, 3, \ldots, n$
  At each iteration k construct the new vector

$$\boldsymbol{p}(k) = [\,\pi_1(k),\, \pi_2(k),\, \ldots,\, \pi_k(k)\,]^{\mathrm{T}},$$

  based on the knowledge of $\boldsymbol{p}(k-1)$ and (2.12).

- **Step 3:** The number of synchronization errors is given by

$$\pi(n) = \mathbf{1}^{\mathrm{T}}\boldsymbol{p}(n) = \pi_1(n) + \cdots + \pi_n(n).$$

The procedure is summarized in Algorithm 2.1.1.

Algorithm 2.1.1 Calculates the number of errors for given order of the system

function ERRORSNUMBER(n)
  for $k := 2, \ldots, n$ **do**
    $\pi_1(k) \leftarrow 1$
    for $l := 2, \ldots, k-1$ **do**
      $\pi_l(k) \leftarrow k(\pi_l(k-1) + \pi_{l-1}(k-1))$
    end for
    $\pi_k(k) \leftarrow k \cdot \pi_{k-1}(k-1)$
  end for
  $\boldsymbol{p}(n) \leftarrow [\,\pi_1(n), \pi_2(n), \ldots, \pi_n(n)\,]^{\mathrm{T}}$
  $\pi(n) \leftarrow \mathbf{1}^{\mathrm{T}}\boldsymbol{p}(n)$
  return $\pi(n)$
end function

As an example consider the case for $n = 6$, where

$$\begin{aligned}
&\boldsymbol{p}(1) = [\,1\,] && \pi(1) = \mathbf{1}^{\mathrm{T}}\boldsymbol{p}(1) = 1\\
&\boldsymbol{p}(2) = [\,1,\ 2\,]^{\mathrm{T}} && \pi(2) = 3\\
&\boldsymbol{p}(3) = [\,1,\ 6,\ 6\,]^{\mathrm{T}} && \pi(3) = 13\\
&\boldsymbol{p}(4) = [\,1,\ 14,\ 36,\ 24\,]^{\mathrm{T}} && \pi(4) = 75\\
&\boldsymbol{p}(5) = [\,1,\ 30,\ 150,\ 240,\ 120\,]^{\mathrm{T}} && \pi(5) = 541\\
&\boldsymbol{p}(6) = [\,1,\ 62,\ 540,\ 1560,\ 1800,\ 720\,]^{\mathrm{T}} && \pi(6) = 4683
\end{aligned}$$

For the estimation of $\pi(n)$, begin with $\pi_2(n)$ and the following equations

$$\pi_2(n+1) = 2(\pi_2(n) + \pi_1(n)) = 2\pi_2(n) + 2, \quad \pi_2(1) = 0, \quad n = 1, 2, \ldots,$$

or

$$\pi_2(n) - 2\pi_2(n-1) = 2, \qquad n = 2, 3, \ldots \quad .$$

These equations can be written in the following form by successively multiplying by the power of 2

$$\begin{aligned}
\pi_2(n) - 2\pi_2(n-1) &= 2\\
2\pi_2(n-1) - 2^2\pi_2(n-2) &= 2^2\\
2^2\pi_2(n-2) - 2^3\pi_2(n-3) &= 2^3\\
\ldots &= \ldots\\
2^{n-3}\pi_2(3) - 2^{n-2}\pi_2(2) &= 2^{n-2}\\
2^{n-2}\pi_2(2) - 2^{n-1}\pi_2(1) &= 2^{n-1}\\
\pi_2(1) &= 0.
\end{aligned}$$

Moreover

$$2 + 2^2 + 2^3 + \cdots + 2^{n-1} = 2\sum_{i=0}^{n-2} = 2\frac{2^{n-1} - 1}{2 - 1} = 2^n - 2$$

and by summation

$$\pi_2(n) = 2^n - 2.$$

In the case of $\pi_3(n)$

$$\begin{aligned}
\pi_3(n) - 3\pi_3(n-1) &= 3\pi_2(n-1) \\
3\pi_3(n-1) - 3^2\pi_3(n-2) &= 3^2\pi_2(n-2) \\
3^2\pi_3(n-2) - 3^3\pi_3(n-3) &= 3^3\pi_2(n-3) \\
\dots &= \dots \\
3^{n-4}\pi_3(4) - 3^{n-3}\pi_3(3) &= 3^{n-3}\pi_2(3) \\
3^{n-3}\pi_3(3) - 3^{n-2}\pi_3(2) &= 3^{n-2}\pi_2(2) \\
\pi_3(2) &= 0
\end{aligned}$$

and hence

$$\pi_3(n) = \sum_{i=1}^{n-2} 3^i \pi_2(n-i) = \sum_{i=1}^{n-2} 3^i (2^{n-i} - 2).$$

In general for $k \leq n$

$$\begin{aligned}
\pi_k(n) - k\pi_k(n-1) &= k\pi_{k-1}(n-1) \\
k\pi_k(n-1) - k^2\pi_k(n-2) &= k^2\pi_{k-1}(n-2) \\
3^2\pi_k(n-2) - 3^3\pi_k(n-3) &= k^3\pi_{k-1}(n-3) \\
\dots &= \dots \\
k^{n-k-1}\pi_k(k+1) - k^{n-k}\pi_k(k) &= k^{n-k}\pi_{k-1}(k) \\
k^{n-k}\pi_k(k) - k^{n-k+1}\pi_k(k-1) &= k^{n-k+1}\pi_{k-1}(k-1) \\
\pi_k(k-1) &= 0,
\end{aligned}$$

leads to

$$\pi_k(n) = \sum_{i=1}^{n-k+1} k^i \pi_{k-1}(n-i).$$

also

$$\pi_k(k) = k\pi_{k-1}(k-1)$$

and hence

$$\pi_n(n) = n(n-1)\cdots 2 \cdot 1 = n!.$$

Consequently a lower bound for the number of clock synchronization errors is

$$\pi(n) \geq n!.$$

This bound may be obtained in a different and simpler way. Recall that every sequence of length k for an nth order system generates k sequences of length k and $k+1$ sequences of length $k+1$. Starting with the sequence $(\{1\})$ for $n = 1$ we obtain for $n = 2$ one sequence $(\{1,2\})$ of length 1 and two sequences $(\{2\},\{1\})$, $(\{1\},\{2\})$ of length 2. Hence the system with 2 state variables has $1 + 2 = 3$ different sequences. Further adding of a new state variable gives that sequence $(\{1,2\})$ generates 1 sequence $(\{1,2,3\})$ of length 1 and 2 sequences $(\{3\},\{1,2\})$,$(\{1,2\},\{3\})$ of length 2. Similarly $(\{2\},\{1\})$ generates 2 sequences of $(\{2,3\},\{1\})$,$(\{2\},\{1,3\})$ of length 2 and 3

$(\{3\},\{2\},\{1\}),(\{2\},\{3\},\{1\}),(\{2\},\{1\},\{3\})$ of length 3. The sequence $(\{1\},\{2\})$ gives $(\{1,3\},\{2\}),(\{1\},\{2,3\})$ and $(\{3\},\{1\},\{2\}),(\{1\},\{3\},\{2\}),(\{1\},\{2\},\{3\})$.

This process may be represented as a binary tree; see Figure 2.3. The depth of the tree equals $n-1$. The value of $\pi(n)$ is calculated in the following way. For every leaf we calculate the product of the values of the nodes at the path from the beginning to that leaf. Summing up all the products for every leaf gives $\pi(n)$.

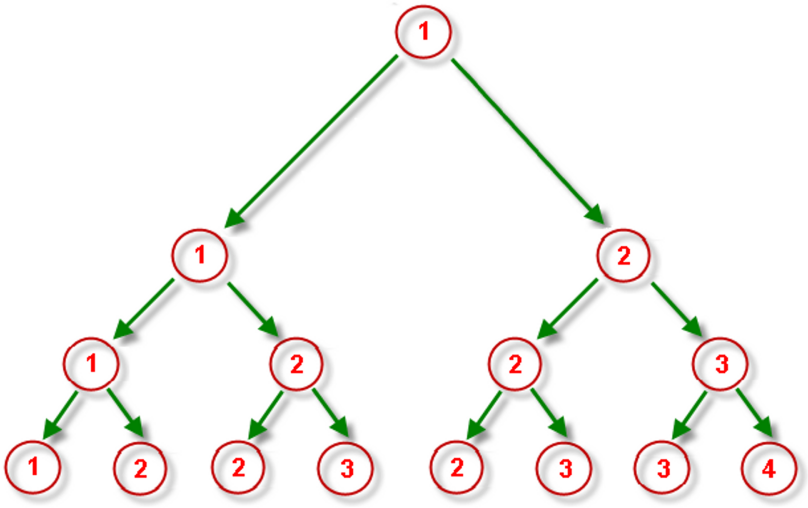

FIGURE 2.3: The tree expanded to the depth of three

As an example

$$\begin{aligned}\pi(1) &= 1\\ \pi(2) &= 1+2=3\\ \pi(3) &= 1+2+4+6=13\\ \pi(4) &= 1+2+4+6+8+12+18+24=75\end{aligned}$$

The lower bound is obtained by calculating the product only for the leaf with the highest value (i.e. $\pi(n) \geq n!$) The upper limit is obtained if we assume that the value at all nodes equals the depth plus one. In this case products equal $n!$ and we have 2^{n-1} leaves and hence

$$n! \leq \pi(n) \leq 2^{n-1}n!. \tag{2.13}$$

Fitting a curve to points $\pi(n)$, $n = 1, 2, \ldots, 20$ gives the more realistic approximation

$$\pi(n) \sim \lfloor(\sqrt{2})^n n!\rfloor, \tag{2.14}$$

where $\lfloor\cdot\rfloor$ denotes the floor function. Table 2.1.1 shows the estimation for $n = 1, \ldots, 20$.

n	$\pi(n)$	$(\sqrt{2})^n n!$	$2^{(n-1)}n!$
1	1	1	1
2	3	4	4
3	13	16	24
4	75	96	192
5	541	678	1920
6	4683	5760	23040
7	47293	57021	322560
8	545835	645120	5160960
9	7087261	8211037	92897280
10	102247563	116121600	1.8579e+09
11	1622632573	1806428157	4.0875e+10
12	2.809157e+10	3.065610e+10	9.8100e+11
13	5.268583e+11	5.636056e+11	2.5506e+13
14	1.064134e+13	1.115882e+13	7.1416e+14
15	2.302832e+14	2.367143e+14	2.1425e+16
16	5.315655e+15	5.356234e+15	6.8560e+17
17	1.303708e+17	1.287726e+17	2.3310e+19
18	3.385535e+18	3.278015e+18	8.3917e+20
19	9.280159e+19	8.808046e+19	3.1889e+22
20	2.677688e+21	2.491292e+21	1.2755e+24

TABLE 2.1: Comparison of the number of synchronization errors with the estimated values.

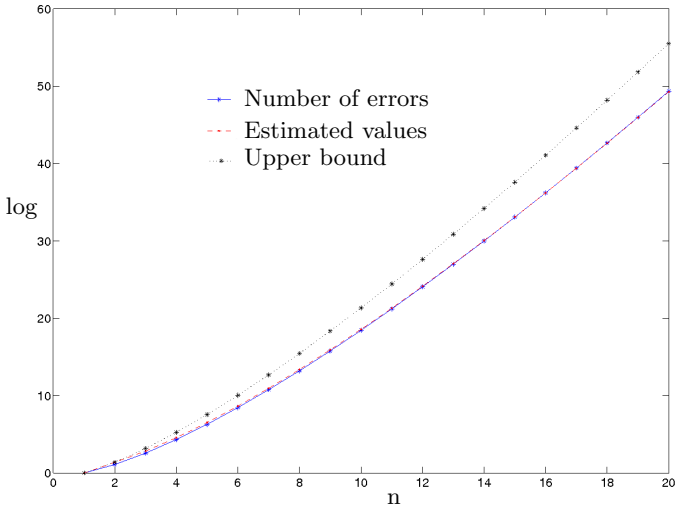

FIGURE 2.4: Comparison of the number of synchronization errors with the estimated values on a logarithmic scale.

The number of possible matrices representing synchronization errors grows very quickly with the order of the system.

2.1.2 The case of different clocks

We consider again a discrete-time linear system decomposed into subsystems. Each subsystem is fed by a clock but whose rates for the subsystems may be different. This case is more complicated than the previous one since between two consecutive input updating events there can be a variable number of switching events, see Figure 2.5 in the case of two subsystems x_1 and x_2 with clock rates T_1 and T_2, respectively, where the input clock rate $T \in (T_1, T_2)$.

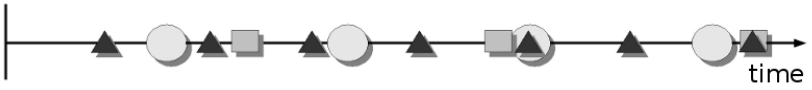

FIGURE 2.5: Switching event pattern for an example asynchronous system. Symbols: ▲–x_1 updates, □–x_2 updates, ◯–input updates.

Consider again the state space model (2.1). where every state vector entry x_i is fed by a clock with rate T_i, $i = 1, 2, ..., n$ where T is the clock rate of the input. The clock rates are not assumed to be equal and can be out of phase. We next derive the model that captures the effects of asynchronous switching.

The basic assumption is that the state variable x_i updates at time points determined by the corresponding clock period T_i. If a single state variable updates it means that its new value is calculated without affecting the other variables. If we have simultaneous updating of more than one variable, only the corresponding values are recalculated. In order to capture the effects of asynchronous switching we introduce a time index k that is incremented by one whenever the input updates and a sequence $s(k)$ of mutually disjoint subsets of indices. The sequence $s(k)$ describes the switching event pattern over a clock period $[kT, (k+1)T)$ and depends on the time index k. The basic assumption employed is that input is constant during that clock period. Let k be fixed with

$$s(k) = (i_1, \ldots, i_d), \quad i_j \subseteq \{1, \ldots, n\} \quad j = 1, \ldots, d \tag{2.15}$$

and hence d events of switching occur over full clock period.

The subsets in (2.15) contain indices of state vector entries that switch simultaneously during the corresponding event. When the jth event of switching occurs the state vector updates according to

$$\boldsymbol{x}^j(k) = \boldsymbol{A}_{i_j}\boldsymbol{x}^{j-1}(k) + \boldsymbol{B}_{i_j}\boldsymbol{u}(k), \tag{2.16}$$

where $\boldsymbol{x}^0(k) = \boldsymbol{x}(k)$ and the model matrix $\boldsymbol{A}_{i_j} \in \mathbb{R}^{n\times n}$ for e.g. given $i = \{\ldots, p, \ldots, q, \ldots\}$

is

$$\boldsymbol{A}_{i_j} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{p(n-1)} & a_{pn} \\ \dots & \dots & \dots & \dots & \dots \\ a_{q1} & a_{q2} & \dots & a_{q(n-1)} & a_{qn} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix} \tag{2.17}$$

and

$$\boldsymbol{B}_{i_j} = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ b_{p1} & b_{p2} & \dots & b_{p(m-1)} & b_{pm} \\ \dots & \dots & \dots & \dots & \dots \\ b_{q1} & b_{q2} & \dots & b_{q(m-1)} & b_{qm} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}. \tag{2.18}$$

This means that during jth event the state vector entries that correspond to i_j are recalculated. We assume that $\boldsymbol{x}(k+1) = \boldsymbol{x}^d(k)$. Back substitution gives the state space model

$$\begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}_{s(k)}\boldsymbol{x}(k) + \boldsymbol{B}_{s(k)}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) & = & \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{array}, \tag{2.19}$$

where for $s(k) = (i_1, \dots, i_d)$

$$\boldsymbol{A}_{s(k)} = \boldsymbol{A}_{i_d} \cdots \boldsymbol{A}_{i_1} \tag{2.20}$$

and

$$\boldsymbol{B}_{s(k)} = \boldsymbol{B}_{i_d} + \boldsymbol{A}_{i_d}\boldsymbol{B}_{i_{d-1}} + \cdots + \boldsymbol{A}_{i_d} \cdots \boldsymbol{A}_{i_2}\boldsymbol{B}_{i_1}. \tag{2.21}$$

This is a time-varying system because the sequence $s(k)$ may vary with time.

2.2 Decentralized control

In this case we assume that each subsystem has its own input and that input is triggered by the sane clock as the subsystem. In other words updating the input for a subsystem and updating the subsystem occurs simultaneously. The updated input is used by the subsystem at the time of switching. Figure 2.6 gives a schematic of this case and each subsystem is assumed to be equipped with a digital controller driven by the same clock as the subsystem itself.

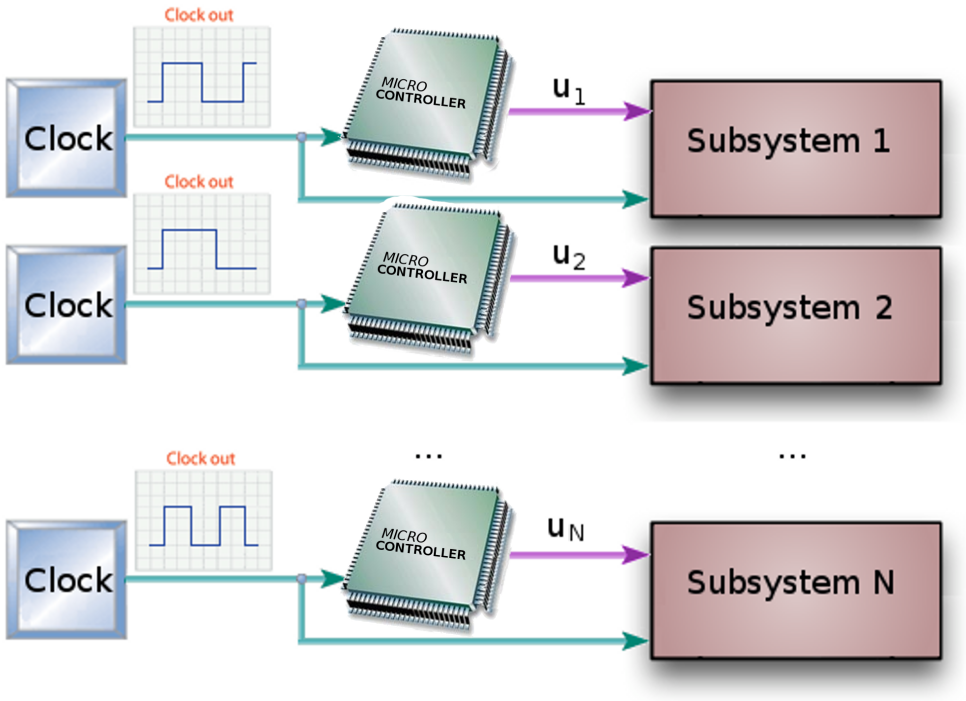

FIGURE 2.6: The case with decentralized control. Each subsystem has its own microcontroller that is triggered by the same clock.

Consider the nominal plant model

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

with state feedback law

$$\boldsymbol{u}(k) = \boldsymbol{K}\boldsymbol{x}(k),$$

resulting in the closed-loop system

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{K}\boldsymbol{x}(k).$$

Introduce the time index p which is incremented when a switching occurs and assume that every microcontroller produces the input

$$\boldsymbol{u}_1(p) = \cdots = \boldsymbol{u}_N(p) = \boldsymbol{K}\boldsymbol{x}(p), \qquad p = 0, 1, \ldots \quad . \tag{2.22}$$

Such an updated input is produced only when it is needed by the subsystem. The particular microcontroller uses the current state and provides the updated input only for the switching subsystem. The system behaves equivalently as the one satisfying the assumption in (2.22). The resulting model is

$$\boldsymbol{x}(p+1) = \boldsymbol{A}_{i(p)}(\boldsymbol{x}(p)) + \boldsymbol{B}_{i(p)}\boldsymbol{K}\boldsymbol{x}(p), \qquad i(p) \subseteq \{1, \ldots, n\},$$

with matrices $\boldsymbol{A}_{i(p)}$ and $\boldsymbol{B}_{i(p)}$ for $i(p) = \{\ldots, p, \ldots, q, \ldots\}$ given by

$$\boldsymbol{A}_{i(p)} = \begin{bmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{p1} & a_{p2} & \ldots & a_{p(n-1)} & a_{pn} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{q1} & a_{q2} & \ldots & a_{q(n-1)} & a_{qn} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{bmatrix}, \tag{2.23}$$

$$\boldsymbol{B}_{i(p)} = \begin{bmatrix} 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ b_{p1} & b_{p2} & \ldots & b_{p(m-1)} & b_{pm} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ b_{q1} & b_{q2} & \ldots & b_{q(m-1)} & b_{qm} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 0 \end{bmatrix}. \tag{2.24}$$

Note that $\boldsymbol{B}_{i(p)}\boldsymbol{K} = (\boldsymbol{B}\boldsymbol{K})_{i(p)}$, where index $i(p)$ is in the sense of the definition of $\boldsymbol{B}_{i(p)}$ (the rows other than those specified by $i(p)$ are zero). Finally we obtain the closed-loop system

$$\boldsymbol{x}(p+1) = (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})_{i(p)}\boldsymbol{x}(p), \qquad i(p) \subseteq \{1, \ldots, n\}.$$

Here the index $i(p)$ has the meaning as in the definition of $\boldsymbol{A}_{i(p)}$ (the rows other than those specified by $i(p)$ are from the identity matrix).

The model above differs from the one proposed in Lorand (2004) since the subsystems' structure is not assumed a priori. It is assumed that different state vector entries correspond to different subsystems and hence all possible synchronization errors can be considered. However, the subsystems' structure may be imposed by restricting the set of allowable sequences representing synchronization errors and making the model equivalent.

2.3 On the stability of systems with synchronization errors

2.3.1 Preliminaries

Consider a dynamical system

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), k), \qquad \boldsymbol{x} \in \mathbb{R}^n. \tag{2.25}$$

A point $x^\star \in \mathbb{R}^n$ is an *equilibrium point* of (2.25) from time k_0, if $f(\boldsymbol{x}^\star, k) = \boldsymbol{x}^\star$ for all $k \geq k_0$.

Definition 2.1. A system (2.25) is called *asymptotically stable* around its equilibrium $\boldsymbol{x}^\star$ if it satisfies the following two conditions

1. Given any $\epsilon > 0$, there exists $\delta_1 > 0$ such that if $\|\boldsymbol{x}(k_0) - \boldsymbol{x}^\star\| < \delta_1$, then $\|\boldsymbol{x}(k) - \boldsymbol{x}^\star\| < \epsilon$, $\forall k > k_0$.
2. $\exists \delta_2 > 0$ such that if $\|\boldsymbol{x}(k) - \boldsymbol{x}^\star\| < \delta_2$, then $\boldsymbol{x}(k) \to \boldsymbol{x}^\star$ as $k \to \infty$.

If the first condition of Definition 2.1 is satisfied then the equilibrium point is said to be *stable in the sense of Lyapunov*. Otherwise it is *unstable*. If the system is asymptotically stable and the second condition of Definition 2.1 holds for every $\delta_2 > 0$ then it is termed *globally asymptotically stable* otherwise it is *locally asymptotically stable*.
Consider a linear system

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k). \tag{2.26}$$

First observe that the zero vector $\mathbf{0} \in \mathbb{R}^n$ is an equilibrium point of (2.26) since

$$\mathbf{0} = \boldsymbol{A}\mathbf{0}.$$

Next assume that $\boldsymbol{x}^\star \neq \mathbf{0}$ is an equilibrium of (2.26). This implies that $\boldsymbol{x}^\star$ is an eigenvector of $\boldsymbol{A}$

$$\boldsymbol{x}^\star = \boldsymbol{A}\boldsymbol{x}^\star.$$

However, for all $\delta_2 > 0$ taking $\boldsymbol{x}(k_0) = (1 + \delta_2/2)\boldsymbol{x}^\star$ ($\|\boldsymbol{x}(k_0) - \boldsymbol{x}^\star\| < \delta_2$) gives

$$\boldsymbol{x}(k) = (1 + \frac{\delta_2}{2})\boldsymbol{x}^\star, \quad \text{for every } k \geq k_0.$$

Hence the equilibrium $\boldsymbol{x}^\star \neq \mathbf{0}$ cannot be asymptotically stable. The conclusion is that linear systems can be asymptotically stable only around the origin. Moreover, asymptotically stable linear systems are globally asymptotically stable. In the rest of the work only linear systems are considered and the term *stability* will refer to the *global asymptotic stability around the origin*.

2.3.2 Asynchronous algorithms

When considering stability of systems with clock synchronization errors the stability of asynchronous algorithms is very applicable where a starting point for the literature is Kozyakin (2003). Consider the iterative equation

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k), \tag{2.27}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state and the matrix $A = [\, a_{ij} \,]_{i,j=1,\dots,n}$ is known. Assume also that n processors calculates the n state vector entries separately, i.e. the ith processor calculates x_i independent of the remaining processors. Suppose also that calculations are performed at different instances of time. Introduce the time index p that is incremented by one whenever any calculation is performed and a sequence of subsets $i(p) \subseteq \{1, \dots, n\}$, $p = 0, 1, \dots$ describing which state vector entries are calculated simultaneously during the pth event. We say that the sequence of subsets $i(p), p = 0, 1, \dots$ is *admissible* if every $i \in \{1, \dots, n\}$ belongs to infinitely many subsets. This is equivalent to the condition that every state vector entry is updated infinitely many often. The state vector satisfies

$$\boldsymbol{x}(p+1) = \boldsymbol{A}_{i(p)}\boldsymbol{x}(p), \tag{2.28}$$

where $\boldsymbol{A}_{i(p)}$ is defined as in (2.17) or equivalently

$$x_j(p+1) = \begin{cases} x_j(p) & j \notin i(p) \\ \sum_{k=1}^{n} a_{jk} x_k(p) & j \in i(p) \end{cases}, \quad j = 1, \dots, n, \tag{2.29}$$

which models the evolution of the state in the case of asynchronous computations. Moreover, this equation can also model the autonomous system (2.27) for some clock synchronization error s. We assume in general that clock synchronization error may be a function of time, i.e. $s = s(k)$, $k = 0, 1, \dots$, but can be represented by some admissible sequence of subsets $i(p), p = 0, 1, \dots$. This happens, for example, in a system with different clock frequencies. Note that asymptotic stability of the asynchronous algorithm (2.29) for all admissible sequences $i(p)$ implies asymptotic stability of the system (2.27) in the case of all synchronization errors.

The first major results in the stability theory of asynchronous algorithms relevant to this work is Theorem 2.2. In this paper asynchronous algorithms of the form

$$x_j(p+1) = \begin{cases} x_j(p) & j \notin i(p) \\ \sum_{k=1}^{n} a_{jk} x_k(p - d(p)) & j \in i(p) \end{cases}, \quad j = 1, \dots, n \tag{2.30}$$

are considered where $0 < d(p) < d$ are the delays, i.e., it is assumed that when calculating the ith entry the ith processor has access only to past values of x represented by the delay $d(p)$. It is also assumed that the delays are bounded by some integer.

Theorem 2.2. *Chazan and Miranker (1969). Assume that* $\mathbf{A} = [\, a_{ij} \,]_{1 \le i,j \le n}$ *is a matrix*

with real entries. The asynchronous algorithm (2.30) is asymptotically stable in the class of all admissible sequences $i(p)$,$p = 0, 1, \ldots$, if and only if the spectral radius $\rho(|\mathbf{A}|) < 1$, where $|\mathbf{A}| = [\,|a_{ij}|\,]_{1 \le i,j \le n}$.

A second important theorem is from Asarin et al. (1992).

Theorem 2.3. *If the matrix $\mathbf{A}$ is symmetric, $\mathbf{A} = \mathbf{A}^T$, then the asynchronous algorithm (2.29) is asymptotically stable in the class of all admissible updating sequences $i(p), p = 0, 1, \ldots$ if and only if $\rho(\mathbf{A}) < 1$.*

Although both theorems provide sufficient and necessary conditions for stability only sufficient conditions may be used when considering stability of systems with clock synchronization errors. This is due to the fact that the set of allowable clock synchronization errors for a given system is usually a subset of the set of all admissible sequences of subsets (after converting synchronization errors to an infinite sequence of subsets). Hence a given system that is unstable for some admissible sequence may be stable in case of all synchronization errors.

2.3.3 The control problem

Consider state feedback control for synchronous system in the form

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k). \tag{2.31}$$

Theorems 2.2 and 2.3 may be used only in the case when the controller updates the output immediately after a switching occurs. This can be implemented as *decentralized control*. In the asynchronous algorithms setting the stabilization with a decentralized controller takes the form

$$x_j(p+1) = \begin{cases} x_j(p) & j \notin i(p) \\ \sum_{k=1}^{n} (a+bk)_{jk} x_k(p) & j \in i(p) \end{cases}, \quad j = 1, \ldots, n, \tag{2.32}$$

where $(a + bk)_{ij}, 1 \le i, j \le n$ are the entries of the matrix $\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}$ and $\boldsymbol{K}$ is the controller matrix. The corresponding full equation is

$$\boldsymbol{x}(p+1) = (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})_{i(p)} \boldsymbol{x}(k) \tag{2.33}$$

and both Theorems 2.2 and 2.3 can be used. Thus in order to ensure stability we require

$$\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K} = (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^{\mathrm{T}}, \qquad \rho(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}) < 1, \tag{2.34}$$

or

$$\rho(|\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}|) < 1. \tag{2.35}$$

Consider the closed loop system

$$\boldsymbol{x}(k+1) = (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})\boldsymbol{x}(k).$$

Consider the candidate Lyapunov function

$$V(k) = \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(k)$$

for a positive definite matrix $\boldsymbol{P} = \boldsymbol{P}^{\mathrm{T}} \succ \boldsymbol{0}$. Consider also the difference along the trajectory

$$\begin{aligned}\Delta V(k+1) = V(k+1) - V(k) &= \boldsymbol{x}(k+1)^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(k+1) - \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(k) \\ &= \boldsymbol{x}(k)^{\mathrm{T}}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^{\mathrm{T}}\boldsymbol{P}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})\boldsymbol{x}(k) - x(k)^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(k) \\ &= \boldsymbol{x}(k)^{\mathrm{T}}[(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^{\mathrm{T}}\boldsymbol{P}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}) - \boldsymbol{P}]\boldsymbol{x}(k).\end{aligned}$$

Then $\Delta V(k) < 0$ and hence stability is guaranteed if

$$(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^{\mathrm{T}}\boldsymbol{P}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}) - \boldsymbol{P} \prec \boldsymbol{0}.$$

Applying the Schur's complement formula now gives

$$\begin{bmatrix} -\boldsymbol{P}^{-1} & (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}) \\ (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^{\mathrm{T}} & -\boldsymbol{P} \end{bmatrix} \prec \boldsymbol{0}$$

and multiplying this last expression from the left and right by $\mathrm{diag}(\boldsymbol{I}, \boldsymbol{P}^{-1})$ now gives

$$\begin{aligned}&\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}^{-1} \end{bmatrix}\begin{bmatrix} -\boldsymbol{P}^{-1} & (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}) \\ (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^{\mathrm{T}} & -\boldsymbol{P} \end{bmatrix}\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}^{-1} \end{bmatrix} \\ &= \begin{bmatrix} -\boldsymbol{P}^{-1} & (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}) \\ \boldsymbol{P}^{-1}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^{\mathrm{T}} & -\boldsymbol{I} \end{bmatrix}\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}^{-1} \end{bmatrix} \\ &= \begin{bmatrix} -\boldsymbol{P}^{-1} & (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})\boldsymbol{P}^{-1} \\ \boldsymbol{P}^{-1}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^{\mathrm{T}} & -\boldsymbol{P}^{-1} \end{bmatrix} \prec \boldsymbol{0}\end{aligned}$$

and setting $\boldsymbol{W} = \boldsymbol{P}^{-1}$ and $\boldsymbol{K}\boldsymbol{W} = \boldsymbol{N}$ gives the stabilization condition

$$\begin{bmatrix} -\boldsymbol{W} & \boldsymbol{A}W + \boldsymbol{B}\boldsymbol{N} \\ \boldsymbol{W}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{N}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}} & -\boldsymbol{W} \end{bmatrix} \prec \boldsymbol{0}. \tag{2.36}$$

If (2.36) is satisfied for matrices $\boldsymbol{W} \succ \boldsymbol{0}$ and $\boldsymbol{N}$, a stabilizing $\boldsymbol{K}$ is given by

$$\boldsymbol{K} = \boldsymbol{N}\boldsymbol{W}^{-1}.$$

Example 2.2. *Consider the case when*

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

with

$$\boldsymbol{A} = \begin{bmatrix} 1.2 & 0.0 & -1.0 & -0.2 & -1.4 \\ 0.7 & 0.3 & -1.1 & 0.8 & 0.2 \\ 0.6 & 0.8 & -0.8 & -0.7 & 0.1 \\ 1.0 & -0.2 & -1.2 & -0.8 & 0.4 \\ 1.3 & 0.0 & -1.1 & 1.1 & -0.6 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} -0.6 & -0.4 & 0.1 & 0.0 & -1.0 \\ -0.2 & 0.1 & -0.1 & 0.5 & 0.5 \\ -0.5 & -0.8 & 0.9 & -0.7 & 0.8 \\ 0.3 & -0.7 & -0.6 & 0.8 & 0.6 \\ 0.8 & -0.5 & -0.1 & 0.8 & 0.8 \end{bmatrix}.$$

The system is unstable since $\rho(\boldsymbol{A}) = 1.7163$. *We will design a state feedback controller using the condition* (2.34). *Since the matrix* $\boldsymbol{W}$ *in the condition* (2.36) *is symmetric the equivalent condition is*

$$\boldsymbol{A}\boldsymbol{W} + \boldsymbol{B}\boldsymbol{N} = \boldsymbol{W}\boldsymbol{A}^T + \boldsymbol{N}^T\boldsymbol{B}^T,$$

The following Matlab code implements the design objective (2.34) *using the Yalmip parser*

```
n = size(A,1); l = size(B,2);
W = sdpvar(n,n,'symmetric');
N = sdpvar(l,n,'full');
F = [W > 0];
% Stability condition
F = [F, [-W, A*W+B*N; W*A'+N'*B', -W] < 0];
% Design objective
F = [F, A*W+B*N <= W*A' + N'*B' ];
F = [F, A*W+B*N >= W*A' + N'*B' ];
diagnostics = solvesdp(F);
display(yalmiperror(diagnostics.problem));
W = double(W); N = double(N); K=N*inv(W);
```

Solving the problem with the SeDuMi solver (Sturm (1999)) gives a stabilizing $\boldsymbol{K}$ *as*

$$\boldsymbol{K} = \begin{bmatrix} 0.4642 & 0.4193 & -1.0932 & -0.4846 & 0.4698 \\ 0.9160 & -0.0770 & -0.7493 & -1.3596 & -0.3958 \\ -1.5891 & -0.6732 & 1.1641 & -2.9090 & 2.2095 \\ -2.1115 & -0.2635 & 2.0734 & -2.4474 & 1.6116 \\ 0.3962 & -0.2881 & 0.0720 & 0.3437 & -1.3026 \end{bmatrix}.$$

This state feedback control law stabilizes the plant since $\rho(\boldsymbol{A}+\boldsymbol{BK}) = 6.9559 \times 10^{-6}$. *Also*

$$\boldsymbol{A}+\boldsymbol{BK} = (\boldsymbol{A}+\boldsymbol{BK})^T = 10^{-5}\times \begin{bmatrix} -0.0009 & -0.1995 & 0.1749 & 0.1072 & -0.0908 \\ -0.1995 & 0.1322 & 0.1540 & -0.1208 & 0.0446 \\ 0.1749 & 0.1540 & -0.1146 & -0.2330 & -0.1661 \\ 0.1072 & -0.1208 & -0.2330 & 0.3695 & -0.1291 \\ -0.0908 & 0.0446 & -0.1661 & -0.1291 & 0.5705 \end{bmatrix}$$

and (2.35) *is also satisfied since*

$$\rho(|\boldsymbol{A}+\boldsymbol{BK}|) = \rho\left(10^{-5} \times \begin{bmatrix} 0.0009 & 0.1995 & 0.1749 & 0.1072 & 0.0908 \\ 0.1995 & 0.1322 & 0.1540 & 0.1208 & 0.0446 \\ 0.1749 & 0.1540 & 0.1146 & 0.2330 & 0.1661 \\ 0.1072 & 0.1208 & 0.2330 & 0.3695 & 0.1291 \\ 0.0908 & 0.0446 & 0.1661 & 0.1291 & 0.5705 \end{bmatrix} \right) = 8.54\times 10^{-6}.$$

Hence this state feedback control law also stabilizes the plant in the case of all possible synchronization errors using the decentralized *control scheme.*

In the *centralized control* case the situation is quite different. The controller uses its own clock and applying the state feedback control law gives the controlled system model

$$x_j(p+1) = \begin{cases} x_j(p) & j \notin i(p) \\ \sum_{k=1}^n a_{jk}x_k(p) + \sum_{k=1}^n (bk)_{jk}x_k(p-d(p)) & j \in i(p) \end{cases}, \quad j = 1,\dots,n. \tag{2.37}$$

Unfortunately conditions similar to those in Theorems 2.2 and 2.3 cannot be formulated in the case of (2.37). The following example shows that these conditions do not guarantee stability.

Example 2.3. *Consider the system*

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

with

$$\boldsymbol{A} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and a clock synchronization error $s = (\{1\},\{2\})$. *Assume the controller gain matrix to be*

$$\boldsymbol{K}_1 = \begin{bmatrix} -4.7 & -5.6 \\ -6.8 & -7.9 \end{bmatrix}, \quad \boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}_1 = \begin{bmatrix} 0.3 & 0.4 \\ 0.2 & 0.1 \end{bmatrix}.$$

Hence $\rho(|\boldsymbol{A}+\boldsymbol{BK}|) = 0.5$. *The system dynamics in case of synchronization error* s *is*

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

where

$$\boldsymbol{A}_s = \boldsymbol{A}_{\{2\}}\boldsymbol{A}_{\{1\}} = \begin{bmatrix} 5 & 6 \\ 35 & 50 \end{bmatrix}, \quad \boldsymbol{B}_s = \boldsymbol{A}_{\{2\}}\boldsymbol{B}_{\{1\}} + \boldsymbol{B}_{\{2\}} = \begin{bmatrix} 1 & 0 \\ 7 & 1 \end{bmatrix},$$

but $\rho(\boldsymbol{A}_s + \boldsymbol{B}_s\boldsymbol{K}_1) = 1.66$ *and the closed loop system is unstable. Select the control law matrix* $\boldsymbol{K}_2$ *as*

$$\boldsymbol{K}_2 = \begin{bmatrix} -4.9 & -5.8 \\ -6.8 & -7.9 \end{bmatrix} \implies \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}_2 = \begin{bmatrix} 0.1 & 0.2 \\ 0.2 & 0.1 \end{bmatrix}. \tag{2.38}$$

The closed-loop state matrix in the synchronous case is symmetric and with spectral radius $\rho(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}_2) = 0.3$*. As in the previous case the closed loop system with synchronization error is unstable since* $\rho(\boldsymbol{A}_s + \boldsymbol{B}_s\boldsymbol{K}_2) = 1.17$*.*

Centralized control is the most natural way of implementing a controller. However, the theory does not support this case.

Centralized control implementations arise in many applications but, as yet, the case when synchronization errors arise is not resolved and this thesis produces substantial new results in this direction by addressing the following general questions

- Is it possible to stabilize the system against all synchronization errors using state feedback?
- Can the design be completed in a computationally efficient way?

In the case of centralized control, stabilization by state feedback against all synchronization errors means that all state matrices that arise have to be stabilized and as the number of synchronization errors that can arise grows the performance of, for example, Linear Matrix Inequality (LMI) solvers decays below any acceptable level. Chapters 3 and 4 of this thesis develop solutions to this problem by first formulating the problem in terms of the polytopic and norm bounded uncertainty descriptions from linear model based robust control theory. Chapter 5 then makes use of the behavioral approach to systems theory to answer the question: is it possible to identify a clock synchronization error given the system output ? Chapter 6 then considers some topical applications areas for the new results and finally Chapter 7 gives conclusions and areas for possible future research.

Chapter 3

Stabilization using a polytopic uncertainty setting

3.1 Introduction

Consider again the discrete linear system (2.1), which is asymptotically stable if, and only if, the spectral radius (modulus of the largest eigenvalue) of the state transition matrix $\boldsymbol{A}$ is less than one, or there exists a symmetric positive definite matrix $\boldsymbol{P}$, written $\boldsymbol{P} \succ \boldsymbol{0}$, such that

$$\boldsymbol{A}^T \boldsymbol{P} \boldsymbol{A} - \boldsymbol{P} \prec \boldsymbol{0}, \tag{3.1}$$

where "$\prec \boldsymbol{0}$" denotes negative definite.

From previous work Lorand and Bauer (2006a) and Kleptzyn et al. (1984), it is known that synchronization errors can effect the stability of the overall system, i.e. a system with no synchronization errors described by (2.1) can be stable but some of the systems (2.8) resulting from the presence of synchronization errors can be unstable. Also, the exact time sequence of arriving signals to subsequent sub-systems is not known, which makes stability analysis very difficult. We develop methods for this task by treating the complete set of possible systems as the effect of uncertainty on some nominal model. This releases Lyapunov type methods from robust control of linear time-invariant systems for use in this problem area where, in this chapter, a polytopic characterization is considered.

3.2 Stability analysis

Consider a system described by (2.1) in the presence of uncertainty in the model of the dynamics. Then one approach to robust control is to assume the system matrix $\boldsymbol{A}$

assumes values in a fixed polytope (see, for example, Boyd et al. (1994)):

$$\boldsymbol{A} \in \mathrm{Co}\{\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N\},$$

where matrices $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N$ are given vertices and

$$\mathrm{Co}\{\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N\} = \left\{ \sum_{i=1}^{N} \alpha_i \boldsymbol{A}_i : \ \alpha_i \geq 0, \ \sum_{i=1}^{N} \alpha_i = 1 \right\}$$

denotes the convex hull of $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_N$ (the polytope of matrices with given vertices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_N$). To investigate stability in the presence of such uncertainty it is only necessary to check if this property holds for the polytope vertices as this guarantees that every system matrix formed from a convex combination of them is also stable Boyd et al. (1994). Hence only the following set of Linear Matrix Inequalities (LMIs) needs to be satisfied for robust stability to hold

$$\boldsymbol{A}_i{}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{A}_i - \boldsymbol{P} \prec \boldsymbol{0} \tag{3.2}$$

for $i = 1, 2, \ldots, N$ where $\boldsymbol{P} \succ \boldsymbol{0}$.

For the system with no input and clock synchronization errors characterized by the d-element sequence of events $s = \{i_1, \ldots, i_d\}$ i.e.

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_s \boldsymbol{x}(k),$$

the system matrix $\boldsymbol{A}_s$ takes values in the polytope

$$\boldsymbol{A}_s \ \in \ \mathrm{Co}\{\boldsymbol{A}_i : i = 1, \ldots, N\}$$

Hence to check the stability for all possible synchronization errors it is sufficient to solve the LMIs (3.2) for all vertices.

Consider now a discrete linear time-invariant system with clock synchronization errors characterized by the d-element sequence of events $s = \{i_1, \ldots, i_d\}$

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

where system matrices $\boldsymbol{A}_s$ and $\boldsymbol{B}_s$ take values in the polytope

$$[\boldsymbol{A}_s \ \boldsymbol{B}_s] \ \in \ \mathrm{Co}\{[\boldsymbol{A}_i \ \boldsymbol{B}_i] : i = 1, \ldots, N\}$$

and apply the state feedback control law

$$\boldsymbol{u}(k) = \boldsymbol{K} \boldsymbol{x}(k).$$

Then

$$\boldsymbol{x}(k+1) = (\boldsymbol{A}_s + \boldsymbol{B}_s\boldsymbol{K})\boldsymbol{x}(k), \tag{3.3}$$

where

$$\boldsymbol{A}_s + \boldsymbol{B}_s\boldsymbol{K} \in \mathrm{Co}\{\boldsymbol{A}_i + \boldsymbol{B}_i\boldsymbol{K} : i = 1, \ldots, N\}.$$

Hence (3.3) is stable if there exists a $\boldsymbol{P} \succ \boldsymbol{0}$ such that the following set of inequalities is satisfied

$$(\boldsymbol{A}_i + \boldsymbol{B}_i\boldsymbol{K})^T\boldsymbol{P}(\boldsymbol{A}_i + \boldsymbol{B}_i\boldsymbol{K}) - \boldsymbol{P} \prec \boldsymbol{0} \qquad i = 1, \ldots, N. \tag{3.4}$$

The difficulty now is that this last system is not linear with respect to the matrix $\boldsymbol{K}$ and therefore cannot be easily solved numerically. However, using the Schur's complement formula and the approach in Crusius and Trofino (1999) we can replace (3.4) by the following system of LMIs

$$\begin{bmatrix} -\boldsymbol{Q} & \boldsymbol{A}_i\boldsymbol{Q} + \boldsymbol{B}_i\boldsymbol{R} \\ \boldsymbol{Q}\boldsymbol{A}_i{}^T + \boldsymbol{R}^T\boldsymbol{B}_i{}^T & -\boldsymbol{Q} \end{bmatrix} \prec \boldsymbol{0} \qquad i = 1, \ldots, N. \tag{3.5}$$

Also if this LMI system is feasible

$$\boldsymbol{K} = \boldsymbol{R}\boldsymbol{Q}^{-1}$$

is a stabilizing control law matrix.

The solution of (3.5) can be conservative since we solve the system of LMIs with common decision matrix $\boldsymbol{Q}$ (or Lyapunov function). To reduce this, it is possible to use, for example, variable Lyapunov functions Boyd et al. (1994). Also, an estimate of the number of sequences for a given n is, from (2.13),

$$n! \leq \mathrm{card}(\mathcal{S}) \leq 2^{n-1}n!. \tag{3.6}$$

The solution developed below consists of the following steps.

1. Calculate the vertices of a polytope that contains all product matrices representing system behavior in case of synchronization errors.

2. Find a stabilizing control law by solving the set of LMIs (3.5) for the vertices obtained in the previous step.

In order to efficiently compute the solution the number of the vertices of polytope computed in the first step here should be significantly smaller than the number of product matrices. Hence in order to manage the compromise between speed, accuracy and number of vertices a new algorithm is developed and compared in tests against direct computation.

3.3 Estimation of the polytope uncertainty

The basis of the algorithm given below is to treat all the product matrices as vectors and, by linear operations, to enclose them in a simple structure (a ball of unit radius) and hence lessen the computational load incurred in obtaining the convex hull containing them. Each step in this procedure is now detailed.

3.3.1 Preliminaries

Let $\mathbb{R}^{m\times n}$ be the space of the $m \times n$ matrices with real entries. Define the invertible map $\mathrm{vec}(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ as

$$\mathrm{vec}\left(\begin{bmatrix} m_{11} & \dots & m_{1n} \\ \vdots & \vdots & \vdots \\ m_{m1} & \dots & m_{mn} \end{bmatrix}\right) := \begin{bmatrix} m_{11} \\ \vdots \\ m_{m,1} \\ m_{12} \\ \vdots \\ m_{mn} \end{bmatrix} \tag{3.7}$$

and

$$\mathrm{vec}^{-1}\left(\begin{bmatrix} x_1 \\ \vdots \\ x_m \\ x_{m+1} \\ \vdots \\ x_{mn} \end{bmatrix}\right) = \begin{bmatrix} x_1 & x_{m+1} & \dots & x_{(n-1)\cdot m+1} \\ x_2 & x_{m+2} & \dots & x_{(n-1)\cdot m+2} \\ \dots & \dots & \dots & \dots \\ x_m & x_{2m} & \dots & x_{n\cdot m} \end{bmatrix}. \tag{3.8}$$

This map is linear, since for any $a, b \in \mathbb{R}$ and matrices $\boldsymbol{M}, \boldsymbol{N} \in \mathbb{R}^{m\times n}$,

$$\mathrm{vec}(a\boldsymbol{M} + b\boldsymbol{N}) = a \cdot \mathrm{vec}(\boldsymbol{M}) + b \cdot \mathrm{vec}(\boldsymbol{N}).$$

Let $\mathcal{M} \subset \mathbb{R}^{n\times(n+m)}$ denote the input set of compound product matrices, i.e.

$$\mathcal{M} = \{[\boldsymbol{A}_{s_i}, \boldsymbol{B}_{s_i}] : 1 \le i \le N\},$$

where $N = \mathrm{card}(\mathcal{S})$ and define the matrix of input points

$$\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N], \qquad \boldsymbol{x}_k = \mathrm{vec}([\boldsymbol{A}_{s_k},\ \boldsymbol{B}_{s_k}]), \quad k = 1, \dots, N. \tag{3.9}$$

The specific feature of the set $\boldsymbol{X}$ is that all the points lie on some hyperplane as illustrated in Figure 3.1. However, in order to proceed, we need the points to span the

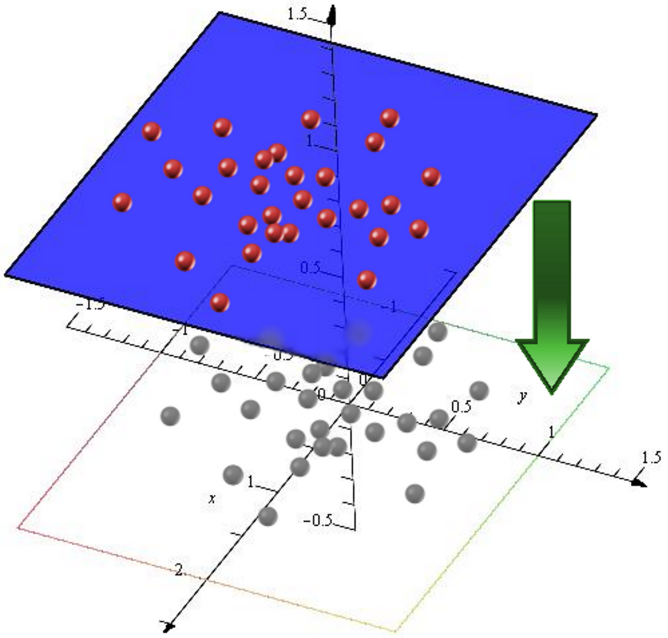

FIGURE 3.1: Input points on a hyperplane. Translation about the center makes it a subspace.

whole space. If we translate the hyperplane by a vector from that hyperplane we obtain a subspace.

Introduce the point representing the center

$$\boldsymbol{c}:=\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i \tag{3.10}$$

and the translated set

$$\boldsymbol{X}_c = [(\boldsymbol{x}_1 - \boldsymbol{c}), \ldots, (\boldsymbol{x}_N - \boldsymbol{c})]$$

Then the subspace spanned by $\boldsymbol{X}_c$ is denoted by $\mathbb{R}^d$, where d denotes the dimension of the hyperplane. Let

$$\boldsymbol{B} = [\,\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_d], \quad \boldsymbol{b}_k \in \mathbb{R}^{n(n+m)}, \quad k = 1, \ldots, d$$

be the orthonormal basis of the subspace $\mathbb{R}^d$ given as columns of the matrix $\boldsymbol{B}$. Then $\boldsymbol{B}^T \cdot \boldsymbol{B} = I$ and let $\boldsymbol{x'} \in \mathbb{R}^d$ denote the coordinates of $\boldsymbol{x} \in \mathbb{R}^{n(n+m)}$ in the basis $\boldsymbol{B}$. Assume also that this point lies on the hyperplane spanned by $\boldsymbol{X}_c$, i.e.,

$$\boldsymbol{x} = x'_1\boldsymbol{b}_1 + \cdots + x'_d\boldsymbol{b}_d = \boldsymbol{B}\boldsymbol{x'}.$$

Hence

$$\boldsymbol{x} = \boldsymbol{B}\boldsymbol{x'}, \qquad \boldsymbol{x'} = \boldsymbol{B}^{\mathrm{T}}\boldsymbol{x} \quad \text{or} \quad \boldsymbol{X}_c = \boldsymbol{B}\boldsymbol{X'}, \quad \boldsymbol{X'} = \boldsymbol{B}^{\mathrm{T}}\boldsymbol{X}_c. \tag{3.11}$$

3.3.2 Convex hull algorithms

The convexity of a given set is an important property and is heavily used in many optimization algorithms.

Definition 3.1. A set is convex if for every pair of points which are members of it a line joining them is also in the set.

The joining line of two points $\boldsymbol{x}_1$ and x_2 is defined as

$$t(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \boldsymbol{x}_1 = t\boldsymbol{x}_2 + (1-t)\boldsymbol{x}_1, \quad t \in [0,1].$$

The line may be written as the linear combination of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$

$$\alpha_1\boldsymbol{x}_1 + \alpha_2\boldsymbol{x}_2, \quad \alpha_1 \geq 0, \alpha_2 \geq 0, \quad \alpha_1 + \alpha_2 = 1,$$

termed a *convex combination*. The definition may be expanded to more points.

Definition 3.2. A linear combination of vectors $\sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i$ is a *convex combination* if the coefficients α_i, $i = 1, \ldots, d$ are nonnegative and $\sum_{i=1}^{n} \alpha_i = 1$.

The definition of a convex combination leads to that for a *convex polytope*.

Definition 3.3. A *convex hull* for a set of points

$$\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, x_N], \quad \boldsymbol{x}_i \in \mathbb{R}^d, \quad i = 1, \ldots, N,$$

denoted by $Co(\boldsymbol{X})$ is the minimal (in the sense of volume) convex set containing $\boldsymbol{X}$.

Definition 3.4. By a *convex polytope* described by vertices

$$\mathcal{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p\} \quad \boldsymbol{v}_i \in \mathbb{R}^d, \quad i = 1, \ldots, p,$$

we mean a convex hull of the vertices.

From this point onwards use of the word polytope means convex polytope. The polytope described by vertices $\mathcal{V}$ is defined as the set of all convex combination of the vertices

$$\left\{ \sum_i \alpha_i \boldsymbol{v}_i : \boldsymbol{v}_i \in \mathcal{V}, \alpha_i \geq 0, i = 1, \ldots, p, \quad \sum_i \alpha_i = 1 \right\}.$$

This representation is called a vertex representation. An alternative definition of a

polytope is the intersection of a set of half-spaces

$$\begin{aligned} a_{1,1}\boldsymbol{x}_1 + \cdots a_{d,1} &\leq h_1 \\ \cdots &\leq \cdots \\ a_{1,p}\boldsymbol{x}_1 + \cdots a_{d,p} &\leq h_p. \end{aligned}$$

The intersection of the supporting hyperplane and a polytope is a $d-1$ dimensional *facet* of the $(d-1)$-face. A $d-2$ dimensional *ridge* arises as the intersection of two facets. In general a facet and a ridge are the generalizations of the face and edge and forms a $(d-1)$-face and $(d-2)$-face in $\mathbb{R}^d$, respectively.

Algorithms.

Many algorithms exist for the case of $d = 2$ and $d = 3$ and the following have the widest use.

- **Brute Force.** For $d = 2$. This method considers each ordered pair of points (p, q) and then determines if all the remaining points lie within the half-plane lying to the right of the directed line $\overline{pq}$. The time complexity is $O(n^3)$.
- **Graham scan.** Graham (1972). First the points are sorted radially in $O(n \log n)$. The procedure starts with the left most point. This point is connected to all others. Then, according to the angles in polar coordinates, the points are connected in counterclockwise order. After obtaining a polygon it is converted to a convex hull by a simple algorithm termed *the three coin algorithm*. The total complexity is $O(n \log n)$. The algorithm is illustrated schematically in Figure 3.2.

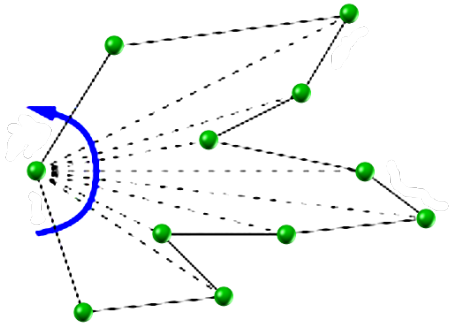

FIGURE 3.2: Graham scan algorithm

- **Gift wrapping.** The planar version of this algorithm is known as *Jarvis march* Jarvis (1973). The extension for $d = 3$ is due to Chan and Kapur (1970) and starts by computing the left most point. Then the algorithm performs a series of pivoting steps to find the next vertex. From the current position the next point chosen is the next vertex if it is the furthest point to the right when observing the

remaining points from the current position. The algorithm is illustrated in Figure 3.3. The time complexity is $O(nh)$, where h denotes the number of vertices of the hull. The algorithm may be generalized to higher dimensions.

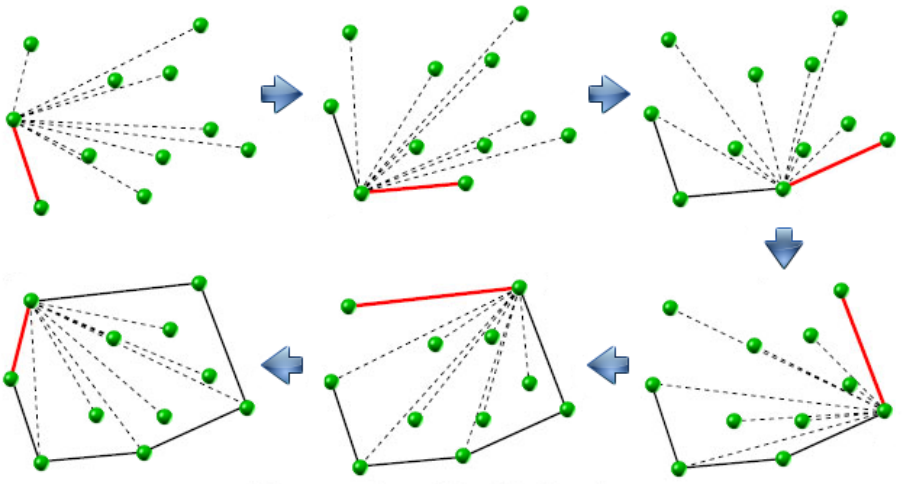

FIGURE 3.3: The Jarvis march.

- **Divide and Conquer.** For $d = 2$ or $d = 3$ this is the generalization of the *Merge-Sort* algorithm. The points are initially sorted by the x coordinate in $O(n \log n)$ time. The algorithm is as follows

  1. If the number of points equals 3 use the *brute force* method.
  2. Otherwise partition the set into two parts of equal number using the x coordinate. Compute the convex hulls of the parts recursively.
  3. Merge the two convex hulls .

  The total time complexity is $O(n \log n)$.

- **Quick hull.** This is a generalization of the *Quick Sort* algorithm. The basic idea is to discard points that are not on the hull as fast as possible. It begins by computing the points with the maximum and minimum x and y coordinates. By connecting these points a convex quadrilateral is obtained. All points inside the quadrilateral can be discarded. The idea is illustrated in Figure 3.4.

- **Incremental algorithm.** This algorithm operates by inserting one point at a time and incrementally updating the hull. If the point is outside the hull all the edges the point can see are deleted and the point is connected to its neighbours. After processing all of the points the desired convex hull is obtained. The time complexity is $O(n \log n)$. The *randomized* version chooses the points at random. This algorithm may be generalized to higher dimensions.

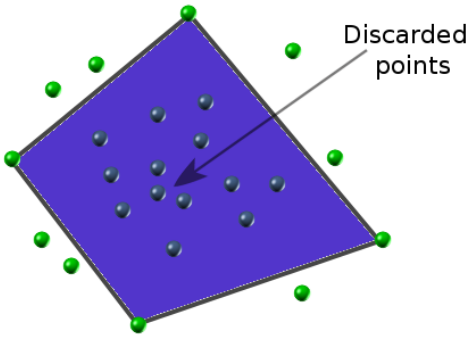

FIGURE 3.4: The Quick Hull algorithm.

- **Monotone chain.** The planar algorithm is due to Andrew (1979). The time complexity is $O(n \log n)$. Points are sorted lexicographically (first by x, then by y). It runs from right most point to the left most point in counterclockwise order constructing the upper (visible from the above) and the lower hull (the remaining part). This algorithm is illustrated schematically in Figure 3.5.

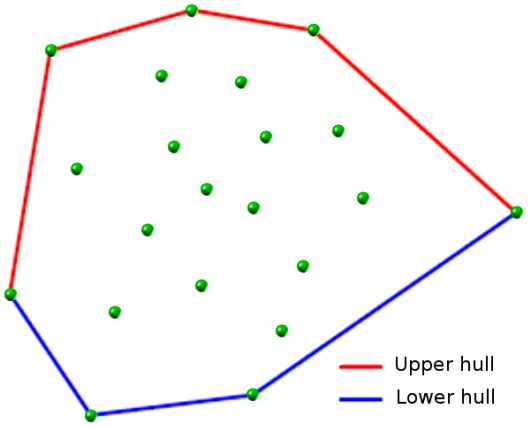

FIGURE 3.5: The Monotone chain algorithm.

- **Mariage-before-conquest (Kirkpatrick-Seidel algorithm).** The planar variant of *Divide and Conquer* algorithm is due to Kirkpatrick and Seidel (1986). The complexity is $O(n \log h)$.

- **Chan's algorithm.** For $d = 2$ or $d = 3$ this algorithm is due to Chan (1996). The time complexity is $O(nh)$. This algorithm combines two slower algorithms, *Graham's scan* and *Jarvis' match*, to form one that is faster than either of them.

A *Beneath-beyond* technique is a method based on a theorem in Grünbaum (1961) and simplified by Kallay (1981). This theorem allows the incremental processing of the points in order to compute a convex hull.

Theorem 3.5. ***(Simplified Beneath-Beyond)*** *Let $\mathcal{H}$ be a convex hull in $\mathbb{R}^d$, and let $\boldsymbol{p}$ be a point in $\mathbb{R}^d \setminus \mathcal{H}$. Then the facets F of $Co(\{\boldsymbol{p}\} \cup \mathcal{H})$ are such that:*

- *F is also a facet of $\mathcal{H}$ if and only if $\boldsymbol{p}$ is below F*
- *F is not a facet of $\mathcal{H}$ if and only if its apex is $\boldsymbol{p}$ and its base is a ridge of $\mathcal{H}$ with one incident facet below $\boldsymbol{p}$ and the other above $\boldsymbol{p}$.*

For any dimension the following algorithms are available

- **Gift wrapping.** A generalization of the *Gift wrapping* method Chand and Kapur (1970).
- **Divide and Conquer**. The generalization of the *Divide and Conquer* algorithm used in two or three dimensions combined with *Beneath-Beyond* Klimo (1988).
- **The quick-hull algorithm**. Barber et al. (1996). This algorithm combines the 2-dimensional *Quickhull* algorithm with the general-dimension *Beneath-Beyond* technique.

Performance

Different implementations including C-based were used for an exemplary and relatively small scale problem (8-dimensional with 10^4 points) in order to test the suitability of these algorithms for the current problem. The problem took hours to execute in every case and this questions their applicability in the current problem. Another issue corresponds to the resulting convex hull with number of vertices depending on the number of input points. In the ideal situation the number of vertices should depend only on the dimension of the problem and not on the size of the input set. This would make the approach to the current problem based on polytopic uncertainty description competitive with the brute force LMI solution. Hence alternatives must be developed as detailed next.

3.3.3 Minimum Volume Enclosing Ellipsoids

Computation of the enclosing polytope can now be performed in the subspace $\mathbb{R}^d$, where d has been defined in the previous section, i.e. a reduced dimension vector space. The starting point is to use any of the available methods to compute the minimal volume ellipsoids, i.e. the matrix E and the point $\boldsymbol{e}$, such that the set

$$\mathcal{E}^\star(\boldsymbol{E}, \boldsymbol{e}) = \{\boldsymbol{y} \in \mathbb{R}^d : (\boldsymbol{y} - \boldsymbol{e})^{\mathrm{T}} \boldsymbol{E} (\boldsymbol{y} - \boldsymbol{e}) \leq 1\} \tag{3.12}$$

is of minimal volume and contains $\boldsymbol{X}'$.

The idea behind the method is: First solve a minimization problem with large number of constraints (the uncertainty should contain all the given points) and then reduce the number of constraints by the computation of the Minimum Volume Enclosing Ellipsoid (MVEE) for the points. Once this stage is completed the constraints of the original problem can be replaced by a formulation in terms of the constructed ellipsoid or the ellipsoid can be used for significantly reducing the number of constraints.

MVEE are also known as Löwner-John ellipsoids John (1985) and can be computed using known algorithms such as those based on the ascent method first order algorithms Khachiyan (1996); Silverman and Titterington (1980). Second order methods that use variants of the Newton method are given in, for example, Nesterov and Nemirovski (1994) and Sun and Freund (2004).

The ellipsoid in general is defined as the image of a unit ball under an affine transformation, i.e. the set

$$\mathcal{E}(\boldsymbol{A}, \boldsymbol{c}) = \{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{c} : \|\boldsymbol{x}\|_2 \leq 1, \quad \boldsymbol{x} \in \mathbb{R}^d\}$$

represents the ellipsoid centered at $\boldsymbol{c} \in \mathbb{R}^d$. The ellipsoid may also be defined as the pre-image of a unit ball under an affine transformation

$$\mathcal{E}(\boldsymbol{Q}, \boldsymbol{c}) = \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{A}^{-1}(\boldsymbol{x} - \boldsymbol{c})\|_2 = \|\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{b}\| \leq 1\},$$

where $\boldsymbol{Q} = \boldsymbol{A}^{-1}$ and $\boldsymbol{b} = -\boldsymbol{A}^{-1}\boldsymbol{c}$. Expanding the norm as the dot product

$$\mathcal{E}(\boldsymbol{E}, \boldsymbol{c}) = \{\boldsymbol{x} \in \mathbb{R} : (\boldsymbol{x} - \boldsymbol{c})\boldsymbol{Q}^T\boldsymbol{Q}(\boldsymbol{x} - \boldsymbol{c}) = (\boldsymbol{x} - \boldsymbol{c})\boldsymbol{E}(\boldsymbol{x} - \boldsymbol{c}) \leq 1\}$$

gives

$$\mathcal{E}(\boldsymbol{A}, \boldsymbol{c}) = \left\{\boldsymbol{x} \in \mathbb{R}^d : (\boldsymbol{x} - \boldsymbol{c})\boldsymbol{A}^{-T}\boldsymbol{A}^{-1}(\boldsymbol{x} - \boldsymbol{c}) = (\boldsymbol{x} - \boldsymbol{c})(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}(\boldsymbol{x} - \boldsymbol{c}) \leq 1\right\}$$

and hence

$$\boldsymbol{E} = (\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}.$$

Consider the ellipsoid $\mathcal{E}(\boldsymbol{A}, \boldsymbol{0})$ centered at the origin as the image of the unit ball. The key point in understanding the connection between the ellipsoid and the properties of a matrix is the *Singular Value Decomposition* (SVD)

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}},$$

where the matrices

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1, \ldots, \boldsymbol{u}_d \end{bmatrix}, \qquad \boldsymbol{V}^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{v}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{v}_d^{\mathrm{T}} \end{bmatrix},$$

are orthogonal and

$$\boldsymbol{S} = \operatorname{diag}(\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_d) = \begin{bmatrix} \sigma_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_d \end{bmatrix}.$$

The left singular vectors are the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ and can also be obtained by eigendecomposition of the matrix $\boldsymbol{E}^{-1}$ since $\boldsymbol{E}^{-1} = \boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$.

If we consider the ball of unit radius, the first operation $\boldsymbol{V}^{\mathrm{T}}$ transforms the ball to itself since the orthogonal matrix $\boldsymbol{V}^{\mathrm{T}}$ represents rotation. The second operation $\boldsymbol{S}$ represents scaling. The ball is transformed into an ellipsis with axes lengths defined by the singular values $\sigma_1, \ldots, \sigma_d$. The final operation $\boldsymbol{U}$ rotates the ellipse.

Let

$$\boldsymbol{e}_i = [0, \ldots, 0, 1_{\text{ith entry}}, 0, \ldots, 0], \quad i = 1, \ldots, d$$

and for $i = 1, \ldots, d$

$$\boldsymbol{V}^{\mathrm{T}}\boldsymbol{v}_i = \begin{bmatrix} \boldsymbol{v}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{v}_d^{\mathrm{T}} \end{bmatrix} \boldsymbol{v}_i = \boldsymbol{e}_i.$$

Hence

$$\boldsymbol{A}\boldsymbol{v}_i = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{v}i = \boldsymbol{U}\boldsymbol{S}\boldsymbol{e}_i = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d]\, \sigma_i \boldsymbol{e}_i = \sigma_i \boldsymbol{u}_i$$

and each axis $\sigma_i \boldsymbol{e}_i$ is transformed into the axis $\sigma_i \boldsymbol{u}_i$, see Figure 3.6.

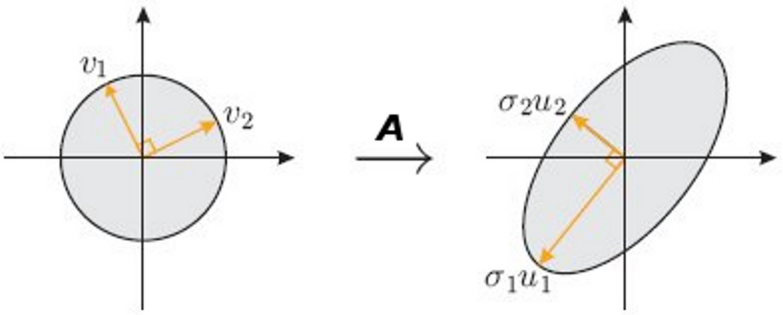

FIGURE 3.6: Visualization of the matrix operation in $\mathbb{R}^2$. The right singular vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are transformed into $\sigma_1 \boldsymbol{u}_1$ and $\sigma_2 \boldsymbol{u}_2$, respectively.

Consider the volume of the ellipsoid where $\boldsymbol{U}$ and $\boldsymbol{V}^{\mathrm{T}}$ are only rotations,

$$\text{vol } \mathcal{E}(\boldsymbol{A}, \boldsymbol{0}) = \text{vol } \mathcal{E}(\boldsymbol{S}, \boldsymbol{0}).$$

The calculation of the volume is by the integral

$$\text{vol } \mathcal{E}(\boldsymbol{S}, \boldsymbol{0}) = \int_{\mathcal{E}(\boldsymbol{S}, \boldsymbol{0})} \mathrm{d}x_1 \cdots \mathrm{d}x_d,$$

or, transforming the coordinates $x_i' = \boldsymbol{x}_i/\sigma_i$, $i = 1, \ldots, d$

$$\text{vol } \mathcal{E}(\boldsymbol{S}, \boldsymbol{0}) = \int_{\mathcal{B}(\boldsymbol{0},1)} \sigma_1 \cdots \sigma_d \mathrm{d}x_1' \cdots \mathrm{d}x_d' = \beta_d \cdot \det \boldsymbol{A},$$

where $\mathcal{B}(\boldsymbol{0}, 1)$ denotes the ball of unit radius centered at the origin and β_d its volume. In the case when the matrix is not square we may consider the volume as the product of nonzero singular values multiplied by β_d.

Consider the set

$$\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N].$$

Then the MVEE is a solution to the following optimization problem

$$\begin{aligned} \text{minimize}_{\text{over}\boldsymbol{Q},\boldsymbol{b}} \quad & \log\det \boldsymbol{Q}^{-1} = -\log\det \boldsymbol{Q} \\ \text{subject to} \quad & \|\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{b}\| \leq 1, \quad k = 1, \ldots, N \end{aligned}$$

The constraints may also be written as

$$(\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{b})^{\mathrm{T}}(\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{b}) - 1 \leq 0,$$

or, on applying the Schur's complement formula, the LMIs

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{b} \\ (\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{b}) & 1 \end{bmatrix} \succeq \boldsymbol{0}.$$

One solution method for this LMI is to use existing solvers such as SeDuMi Sturm (1999). However, more efficient algorithms exist. In particular, the following algorithm given in Khachiyan (1996), taken in turn from Kumar and Yildrim (2005), is considered.

For $\epsilon > 0$ the ellipsoid $\mathcal{E}(\boldsymbol{E}, \boldsymbol{c})$ is a $(1+\epsilon)$-approximation to MVEE of $\boldsymbol{X}$ Khachiyan (1996); Kumar and Yildrim (2005) if

$$\boldsymbol{X} \subseteq \mathcal{E}(\boldsymbol{E}, \boldsymbol{c}) \quad \text{and} \quad \text{vol } \mathcal{E}(\boldsymbol{E}, \boldsymbol{c}) \leq (1+\epsilon)\text{vol MVEE}(\boldsymbol{X}). \tag{3.13}$$

Suppose also that the input points are mapped to $\mathbb{R}^{d+1}$ as

$$\boldsymbol{y}_k = \begin{bmatrix} \boldsymbol{x}_k \\ 1 \end{bmatrix}, \quad k = 1, \ldots, N$$

and introduce

$$\boldsymbol{Y} = [\pm\boldsymbol{y}_1, \ldots, \pm\boldsymbol{y}_N].$$

Note that the set $\boldsymbol{Y}$ is centrally symmetric and hence Khachiyan (1996); Nesterov and Nemirovski (1994)

$$\text{MVEE}(\boldsymbol{X}) = \text{MVEE}(\boldsymbol{Y}) \cap \mathcal{H},$$

where $\mathcal{H}$ is a hyperplane defined by

$$\mathcal{H} = \{\boldsymbol{y} \in \mathbb{R}^{d+1} : y_{d+1} = 1\}.$$

Since $\boldsymbol{Y}$ is symmetric, MVEE($\boldsymbol{Y}$) is centered at origin and hence the original problem is reduced to the following

$$\begin{array}{rl} \text{minimize}_{\text{over } \boldsymbol{M}} & -\log\det \boldsymbol{M} \\ \text{subject to} & \boldsymbol{y}_k^{\mathrm{T}} \boldsymbol{M} \boldsymbol{y}_k \leq 1, \quad k = 1, \ldots, N. \end{array}$$

The positive definite matrix $\boldsymbol{M}^\star$ together with Lagrange multipliers $\boldsymbol{\lambda}^\star \in \mathbb{R}^N$ are optimal if

$$\begin{array}{rcll} -(\boldsymbol{M}^\star)^{-1} + \Pi(\boldsymbol{\lambda}^\star) & = & 0 & \\ \lambda_i^\star(1 - \boldsymbol{y}_i^{\mathrm{T}} \boldsymbol{M}^\star \boldsymbol{y}_i) & = & 0, & i = 1, \ldots, N \\ \boldsymbol{y}_i^{\mathrm{T}} \boldsymbol{M}^\star \boldsymbol{y}_i & \leq & 1, & i = 1, \ldots, N, \end{array}$$

where

$$\Pi(\lambda^\star) = \sum_{i=1}^{N} \lambda_i \boldsymbol{y}_i \boldsymbol{y}_i^{\mathrm{T}}$$

and the dual optimization problem is

$$\begin{array}{rl} \text{maximize}_{\text{over } \boldsymbol{u}} & \log\det \Pi(\boldsymbol{u}) \\ \text{subject to} & \boldsymbol{y}_k^{\mathrm{T}} \boldsymbol{M} \boldsymbol{y}_k \leq 1, \quad k = 1, \ldots, N. \end{array} \tag{3.14}$$

The solution $\boldsymbol{u}^\star$ of this last problem is optimal if and only if the following conditions are satisfied

$$\begin{array}{rcll} \boldsymbol{y}_i^{\mathrm{T}} \Pi(\boldsymbol{u}^\star)^{-1} \boldsymbol{y}_i + s_i^\star & = & t^\star, & i = 1, \ldots, N \\ \mathbf{1}^{\mathrm{T}} \boldsymbol{u}^\star & = & 1 & \\ u_i^\star s_i^\star & = & 0, & i = 1, \ldots, N \\ \boldsymbol{u}^\star & \geq & 0 & \\ \boldsymbol{s}^\star & \geq & 0 & \end{array} \tag{3.15}$$

Taking the first condition and multiplying both sides by $u_i^\star$ and summing up for $i = 1, \ldots, N$ gives

$$\sum_{i=1}^{N} \boldsymbol{y}_i^{\mathrm{T}} \Pi(\boldsymbol{u}^\star)^{-1} \boldsymbol{y}_i = \operatorname{tr}\left(\Pi(\boldsymbol{u}^\star)^{-1} \left[\sum_{i=1}^{N} u_i^\star \boldsymbol{y}_i \boldsymbol{y}_i^{\mathrm{T}} \right] \right) = \operatorname{tr} \boldsymbol{I} = d + 1.$$

Hence $t^\star = d + 1$ and therefore

$$\boldsymbol{M}^\star = \frac{1}{d+1} \Pi(\boldsymbol{u}^\star)^{-1}, \quad \boldsymbol{\lambda}^\star = (d+1)\boldsymbol{u}^\star.$$

In order to compute the solution for the original problem consider the original ellipsoid

$$\text{MVEE}(\boldsymbol{X}) = \left\{ \boldsymbol{x} \in \mathbb{R}^d : \left(\frac{1}{d+1} \right) \left[\boldsymbol{x}^{\mathrm{T}},\ 1 \right] \Pi(\boldsymbol{u}^\star)^{-1} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} \right\} \tag{3.16}$$

and, by the definition of $\Pi(\boldsymbol{u}^\star)$,

$$\begin{aligned}\Pi(\boldsymbol{u}^\star) &= \begin{bmatrix} \boldsymbol{X}\boldsymbol{U}^\star\boldsymbol{X}^{\mathrm{T}} & \boldsymbol{X}\boldsymbol{u}^\star \\ (\boldsymbol{X}\boldsymbol{u}^\star)^{\mathrm{T}} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{I} & \boldsymbol{X}\boldsymbol{u}^\star \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X}\boldsymbol{U}^\star\boldsymbol{X}^{\mathrm{T}} - \boldsymbol{X}\boldsymbol{u}^\star(\boldsymbol{X}\boldsymbol{u}^\star)^{\mathrm{T}} & \boldsymbol{0} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & 0 \\ \boldsymbol{X}\boldsymbol{u}^\star & 1 \end{bmatrix},\end{aligned}$$

where $\boldsymbol{U}^\star = \operatorname{diag}(\boldsymbol{u}^\star)$. Hence

$$\Pi(\boldsymbol{u}^\star)^{-1} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -(\boldsymbol{X}\boldsymbol{u}^\star)^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} \left(\boldsymbol{X}\boldsymbol{U}^\star\boldsymbol{X}^{\mathrm{T}} - \boldsymbol{X}\boldsymbol{u}^\star(\boldsymbol{X}\boldsymbol{u}^\star)^{\mathrm{T}}\right)^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{X}\boldsymbol{u}^\star \\ \boldsymbol{0} & 1 \end{bmatrix}$$

and substitution in (3.16) yields

$$\mathrm{MVEE}(\boldsymbol{X}) = \{\boldsymbol{x} \in \mathbb{R}^d : (\boldsymbol{x} - \boldsymbol{c}^\star)^{\mathrm{T}}\boldsymbol{Q}^\star(\boldsymbol{x} - \boldsymbol{c}) \leq 1\},$$

where

$$\boldsymbol{Q}^\star = \frac{1}{d}\left(\boldsymbol{X}\boldsymbol{U}^\star\boldsymbol{X}^{\mathrm{T}} - \boldsymbol{X}\boldsymbol{u}^\star(\boldsymbol{X}\boldsymbol{u}^\star)^{\mathrm{T}}\right)^{-1}, \quad \boldsymbol{c}^\star = \boldsymbol{X}\boldsymbol{u}^\star.$$

The problem (3.14) is solved iteratively by considering the linearization

$$\begin{aligned} \text{maximize}_{\text{over } \boldsymbol{v}} \quad & \textstyle\sum_{i=1}^N v_i \boldsymbol{y}_i^{\mathrm{T}} \Pi(\boldsymbol{v}) \boldsymbol{y}_i \\ \text{subject to} \quad & \boldsymbol{1}^{\mathrm{T}}\boldsymbol{v} = 1, \quad \boldsymbol{v} \geq \boldsymbol{0} \end{aligned}$$

The feasible region is a unit simplex and hence the optimal solution is the unit vector $\boldsymbol{e}_j$ where

$$j = \arg\max_{i=1,\ldots,N} \boldsymbol{y}_i^{\mathrm{T}} \Pi(\boldsymbol{u}_k)^{-1} \boldsymbol{y}_i.$$

Let

$$\kappa_k = \max_{i=1,\ldots,N} \boldsymbol{y}_i^{\mathrm{T}} \Pi(\boldsymbol{u}_k)^{-1} \boldsymbol{y}_i$$

and the next iteration is given by

$$\boldsymbol{u}_{k+1} = (1 - \beta_k)\boldsymbol{u}_k + \beta_k \boldsymbol{e}_j,$$

where β_k is a solution to the following optimization problem

$$\text{maximize}_{\text{over}\beta} \quad \log\det \Pi((1 - \beta)\boldsymbol{u}_k + \beta \boldsymbol{e}_j)$$

and Khachiyan (1996)

$$\beta_k = \frac{\kappa_k - (d+1)}{(d+1)(\kappa_k - 1)},$$

with initial point

$$\boldsymbol{u}_0 = \frac{1}{N} \cdot \boldsymbol{1}.$$

The algorithm runs until, Kumar and Yildrim (2005),

$$\epsilon_k \leq (1+\epsilon)^{2/d+1} - 1,$$

where at each iteration $k - 0, 1, 2, \ldots$

$$\epsilon_k := \min\{v \geq 0 : \boldsymbol{y}_i^{\mathrm{T}} \left(\frac{1}{d+1}\right) \Pi(\boldsymbol{u}_k)^{-1} \boldsymbol{y}_i \leq 1 + v : \ i = 1, \ldots, N \ \}$$

Upon termination the algorithm returns a $(1+\epsilon)$-approximation to MVEE($\boldsymbol{X}$) (3.13).

The algorithm is summarized next.

Algorithm 3.3.1 Computing MVEE

function MVEEKHACHIYAN($\boldsymbol{X}$,ϵ)
  $i \leftarrow 0$, $\boldsymbol{u}_0 \leftarrow (1/N)\mathbf{1}$
  while not converged **do**
    $j \leftarrow \arg\max_{i=1,\ldots,N} \boldsymbol{y}_i^{\mathrm{T}} \Pi(\boldsymbol{u}_k)^{-1} \boldsymbol{y}_i$
    $\kappa \leftarrow \max_{i=1,\ldots,N} \boldsymbol{y}_i^{\mathrm{T}} \Pi(\boldsymbol{u}_k)^{-1} \boldsymbol{y}_i$
    $\beta \leftarrow \frac{\kappa-(d+1)}{(d+1)(\kappa-1)}$
    $\boldsymbol{u}_{k+1} \leftarrow (1-\beta)\boldsymbol{u}_k + \beta \boldsymbol{e}_j$
  end while
  $\boldsymbol{U} = \mathrm{diag}(\boldsymbol{u})$
  $\boldsymbol{Q} = \frac{1}{d}\left(\boldsymbol{X}\boldsymbol{U}\boldsymbol{X}^{\mathrm{T}} - \boldsymbol{X}\boldsymbol{u}(\boldsymbol{X}\boldsymbol{u})^{\mathrm{T}}\right)^{-1}$
  $\boldsymbol{c}^\star = \boldsymbol{X}\boldsymbol{u}$
  return $(\boldsymbol{Q}, \boldsymbol{c})$
end function

PCA based method

The second moments, termed moments of inertia, are used in computer graphics to determine the principal axes of a given set of points, Rocha et al. (2002); Prokop and Reeves (1992). For equally weighted points $[\, x_1, \ x_2 \,] \in \mathbb{R}^2$ the moments of order $(p+q)$ and a matrix of second moments are defined as

$$m_{pq} = \sum_{i=1}^{N} x_i^p y_i^q, \quad \boldsymbol{M}_2 = \begin{bmatrix} m_{20} & m_{11} \\ m_{11} & m_{02} \end{bmatrix}.$$

In three dimensions the inertia tensor is used to find a best-fit ellipsoid Karnesky et al. (2007). These particular examples are special cases of a more general method termed Principal Component Analysis (PCA) Jolliffe (2002).

Consider the centered data set

$$\boldsymbol{c} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_k, \quad \boldsymbol{X}' = [(\boldsymbol{x}_k - \boldsymbol{c}), \ldots, (\boldsymbol{x}_N - \boldsymbol{c})]\,.$$

In this method the principal axes are taken as the left singular vectors of $\boldsymbol{X}'$, which are

also the eigenvectors obtained by eigendecomposition of the matrix $\boldsymbol{\Sigma}$

$$\boldsymbol{\Sigma} = \frac{1}{N}\boldsymbol{X}'\boldsymbol{X}'^{\mathrm{T}} = \frac{1}{N}\sum_{k=1}^{N}(\boldsymbol{x}_k - \boldsymbol{c})(\boldsymbol{x}_k - \boldsymbol{c})^{\mathrm{T}}.$$

The matrix $N\boldsymbol{\Sigma}$ is also a matrix of second moments of $\boldsymbol{X}'$.

Consider the inertia tensor $\mathbb{I}$ in $\mathbb{R}^3$ and let $\boldsymbol{r}_k = (\boldsymbol{x}_k - \boldsymbol{c})$, then

$$\mathbb{I} = \sum_{k=1}^{N}\left(\|\boldsymbol{r}_k\|_2^2 \cdot \boldsymbol{I} - \boldsymbol{r}_k \cdot \boldsymbol{r}_k^{\mathrm{T}}\right) = (\operatorname{tr} N\boldsymbol{\Sigma}) \cdot \boldsymbol{I} - N\boldsymbol{\Sigma}$$

and the eigenvectors of $\mathbb{I}$ are those of $\boldsymbol{\Sigma}$. We assume initially zero length of axes and scale the ellipsoid in order to ensure it contains all the points. The scaling procedure is described next.

Scaling of the ellipsoid

Assume that $\boldsymbol{x} \in \mathbb{R}^d$ is a point outside the given ellipsoid $\mathcal{E}(\boldsymbol{P}, \boldsymbol{c})$. Let the length axes of the ellipsoid be given as $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_d]^{\mathrm{T}}$. Then we need to set the new length of the ellipsoid axes $\boldsymbol{\sigma'} = [\sigma'_1, \ldots, \sigma'_d]^{\mathrm{T}}$ such that this point lies on the surface and the new volume is minimal. Equivalently we need to solve the following problem.

$$\begin{aligned} &\text{minimize}_{\text{over } \boldsymbol{\sigma'}} \quad \det \boldsymbol{P} \\ &\text{subject to} \quad (\boldsymbol{x} - \boldsymbol{c})^{\mathrm{T}}\boldsymbol{P}^{-\mathrm{T}}\boldsymbol{P}^{-1}(\boldsymbol{x} - \boldsymbol{c}) = 1 \\ &\qquad\qquad \sigma'_1 \geq \sigma_1, \ldots, \sigma'_d \geq \sigma_d \end{aligned} \tag{3.17}$$

Consider the SVD

$$\boldsymbol{P} = \boldsymbol{U}\operatorname{diag}(\boldsymbol{\sigma})\boldsymbol{V}^{\mathrm{T}}.$$

Since the axes of the ellipsoid are defined by the matrix $\boldsymbol{U}$ we transform the coordinates by the inverse matrix in order to simplify the problem, i.e.,

$$\boldsymbol{x}' = \boldsymbol{U}^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{c}), \quad \boldsymbol{P}' = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{P} = \operatorname{diag}(\boldsymbol{\sigma})\boldsymbol{V}^{\mathrm{T}}.$$

The ellipsoid $\mathcal{E}(\boldsymbol{P}', \boldsymbol{0})$ is identical to $\mathcal{E}(\operatorname{diag}(\boldsymbol{\sigma}), \boldsymbol{0})$, which has the simple description $\sum_{i=1}^{d} x_i^2/\sigma_i^2 \leq 1$. The problem (3.17) now simplifies to the following, where squares are taken for convenience:

$$\begin{aligned} &\text{minimize}_{\text{over } \boldsymbol{\sigma'}} \quad \textstyle\prod_{i=1}^{d} \sigma'^2_i \\ &\text{subject to} \quad \textstyle\sum_{i=1}^{d} \frac{x'^2_i}{\sigma'^2_i} = 1 \\ &\qquad\qquad \sigma'_1 \geq \sigma_1, \ldots, \sigma'_d \geq \sigma_d \end{aligned} \tag{3.18}$$

See also Figure 3.7. The task is to find the lowest volume ellipsoid containing the new point at the boundary.

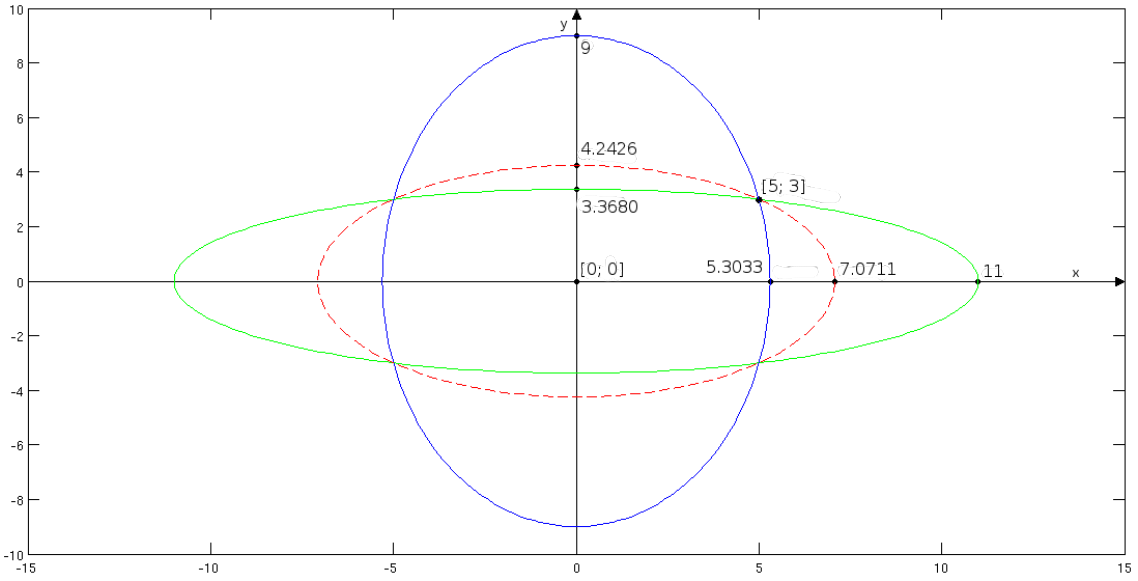

FIGURE 3.7: The point $[x, y]^{\mathrm{T}} \in \mathbb{R}^2$ lies on the boundary if $x^2/\sigma_x^2 + y/\sigma_y^2 = 1$. However, there are infinitely many ellipsoids satisfying this equation. The optimal one with the lowest volume, denoted by the dashed line, satisfies $x^2/\sigma_x^2 = y^2/\sigma_y^2 = 1/2$, i.e. $\sigma_x = \sqrt{2}|x|$ and $\sigma_y = \sqrt{2}|y|$.

Consider the auxiliary problem for $\boldsymbol{s} = [s_1, \ldots, s_e]^{\mathrm{T}}$

$$\begin{array}{ll} \text{minimize}_{\text{over } \boldsymbol{s}} & \prod_{i=1}^{e} s_i^2 \\ \text{subject to} & \sum_{i=1}^{e} \frac{y_i^2}{s_i^2} = a \end{array} \tag{3.19}$$

and define the Lagrangian

$$\mathcal{L}(\boldsymbol{s}, \lambda) = \prod_{i=1}^{e} s_i^2 + \lambda(\sum_{i=1}^{e} \frac{y_i^2}{s_i^2} - 1)$$

and partial derivatives

$$\frac{\partial \mathcal{L}}{\partial s_i} = 2s_i \prod_{j \neq i} s_j^2 - \lambda \frac{2y_i^2}{s_i^3} = 0, \qquad i = 1, \ldots, e.$$

Multiplying each equation by $s_i/2$ gives

$$\prod_{i=1}^{e} s_j^2 - \lambda \frac{y_i^2}{s_i^2} = 0, \qquad i = 1, \ldots, e.$$

Subtracting two equations with $i \neq j$ gives $(\lambda > 0)$

$$-\lambda \left(\frac{y_i^2}{s_i^2} - \frac{y_j^2}{s_j^2} \right) = 0 \iff \frac{y_i^2}{s_i^2} = \frac{y_j^2}{s_j^2}$$

and by the constraints $\sum_{i=1}^{e} \frac{y_i^2}{s_i^2} = a$, the optimal solution is

$$s_i^2 = e\frac{y_i^2}{a}, \quad \text{or} \quad s_i = \sqrt{e}\frac{|y_i|}{\sqrt{a}},$$

where $|\cdot|$ denotes the absolute value. If we omit the last constraint in (3.18) the optimal solution is given by setting $\boldsymbol{\sigma}' = \sqrt{d}|\boldsymbol{x}|$. However, some inequalities $\sigma'_i \geq \sigma_i$ may be violated and in this case we need to determine the set $\mathcal{V}$ of violated constraints

$$\sigma'_j \leq \sigma_j \implies j \in \mathcal{V}, \qquad j = 1, \ldots, d.$$

Then the auxiliary problem should be solved for the remaining variables and

$$a = 1 - \sum_{j \in \mathcal{V}} \frac{x'^2_j}{\sigma_j^2}.$$

This procedure is then repeated until none of the constraints is violated. In order to find the optimal solution of (3.17), the algorithms given below can be used. These algorithms also include the case when

$$\sigma_i = 0, \quad i \in \mathcal{I}_0 \subseteq \{1, \ldots, d\},$$

when the sum $\sum_{i=1}^{d} x_i^2/\sigma_i^2$ may not exist. However, the condition

$$\sum_{\substack{i=1 \\ \sigma_i = 0}}^{d} |x_i| > 0 \quad \textbf{or} \quad \sum_{\substack{i=1 \\ \sigma_i \neq 0}}^{d} \frac{x_i^2}{\sigma_i^2} > 1$$

is satisfied if and only if the point $\boldsymbol{x} \in \mathbb{R}^d$ is outside the ellipsoid $\mathcal{E}(\mathrm{diag}(\boldsymbol{\sigma}), \mathbf{0})$. The complete procedure for calculation of an approximation to MVEE is given in Algorithm 3.3.5 and is suitable for the case when $\boldsymbol{\sigma}$ contains zeros and also if the points do not span the whole space (points are lying on the hyperplane).

Algorithm 3.3.2 Fitting the axes

function FitAxes($\boldsymbol{x}'$,$\boldsymbol{\sigma}$)
  $a \leftarrow 1$, $\mathcal{I} \leftarrow \{1, \ldots, d\}$
  repeat
    $e \leftarrow \text{card}(\mathcal{I})$, $\mathcal{V} \leftarrow \varnothing$
    for $i \in \mathcal{I}$ **do**
      $\sigma'_i \leftarrow \sqrt{e}\frac{|x'_i|}{\sqrt{a}}$
      if $\sigma'_i \leq \sigma_i$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{i\}$, $\sigma'_i \leftarrow \sigma_i$
      end if
    end for
    $\mathcal{I} \leftarrow \mathcal{I} - \mathcal{V}$, $a \leftarrow a - \sum_{\substack{j \in \mathcal{V} \\ \sigma_j \neq 0}} \frac{x'^2_j}{\sigma^2_j}$
  until $\mathcal{V} = \varnothing$
  return $\boldsymbol{\sigma}$
end function

Algorithm 3.3.3 Scalling the Ellipsoid

function ScaleEllipsoid($\boldsymbol{U}$,$\boldsymbol{\sigma}$, $\boldsymbol{c}, \boldsymbol{X}$)
  for $k \leftarrow 1, \ldots, N$ **do**
    $\boldsymbol{x}' \leftarrow \boldsymbol{U}^{\mathrm{T}} \cdot (\boldsymbol{x}_k - \boldsymbol{c})$
    if $\sum_{\substack{i=1 \\ \sigma_i=0}}^{d} |x'_i| > 0$ **or** $\sum_{\substack{i=1 \\ \sigma_i \neq 0}}^{d} x'^2_i/\sigma^2_i > 1$ **then**
      $\boldsymbol{\sigma} \leftarrow$ FitAxes($\boldsymbol{x}'$,$\boldsymbol{\sigma}$)
    end if
  end for
  return $\boldsymbol{\sigma}$
end function

Algorithm 3.3.4 Scaling the Ellipsoid - Overloaded method

function ScaleEllipsoid($\boldsymbol{P}$,$\boldsymbol{c}, \boldsymbol{X}$)
  $(\boldsymbol{U}, \boldsymbol{\sigma}, \boldsymbol{V}) \leftarrow$ svd($\boldsymbol{P}$) ▷ Singular Value Decomposition $\boldsymbol{P} = \boldsymbol{U} \cdot \text{diag}(\boldsymbol{\sigma}) \cdot \boldsymbol{V}^{\mathrm{T}}$
  $\boldsymbol{\sigma}' \leftarrow$ ScaleEllipsoid($\boldsymbol{U}, \boldsymbol{\sigma}, \boldsymbol{c}, \boldsymbol{X}$)
  $\boldsymbol{P} \leftarrow \boldsymbol{U} \cdot \text{diag}(\boldsymbol{\sigma}') \cdot \boldsymbol{V}^{\mathrm{T}}$
  return $\boldsymbol{P}$
end function

Algorithm 3.3.5 Computing the approximation to MVEE

function MVEEPCA($\boldsymbol{X}$)
  $\boldsymbol{c} \leftarrow \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_k$ ▷ Center
  $\boldsymbol{\Sigma} \leftarrow \frac{1}{N}\sum_{k=1}^{N}(\boldsymbol{x}_k - \boldsymbol{c})(\boldsymbol{x}_k - \boldsymbol{c})^{\mathrm{T}}$
  $\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{-1} = \boldsymbol{\Sigma}$ ▷ Eigendecomposition
  $\boldsymbol{U} \leftarrow \boldsymbol{V}$
  $\boldsymbol{\sigma} \leftarrow$ ScaleEllipsoid($\boldsymbol{U}, \boldsymbol{0}, \boldsymbol{c}, \boldsymbol{X}$)
  $\boldsymbol{P} \leftarrow \boldsymbol{U} \cdot \text{diag}(\boldsymbol{\sigma})$
  return $(\boldsymbol{P}, \boldsymbol{c})$
end function

Remarks on complexity

The computation of the initial Enclosing Ellipsoid requires the computation of the SVD of the data matrix $\boldsymbol{X}' \in \mathbb{R}^{d\times N}$ with $N >> d$. The variant of the QR method Golub and Kahan (1964) requires $O(Nd^2)$ operations. The second method requires the calculation of the matrix $\boldsymbol{\Sigma}$, which is $O(Nd^2)$, and the eigendecomposition , which using QR method variant Golub and van Loan (1996) is $O(d^3)$. Summarizing ($N >> d$) the total complexity of the first step is (Nd^2). However, the second method is very sensitive to errors although it may be faster. **Lifted PCA method and relation to the Khachiyan algorithm**

A variant for the ellipsoid computation in higher dimensions is to map the inputs to $\mathbb{R}^{d+1}$

$$\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{x}_k \\ 1 \end{bmatrix}, \quad k = 1, \ldots, N.$$

Define the second set based on $\boldsymbol{Y}$ as

$$\boldsymbol{Z}_- = [-\boldsymbol{z}_1, \ldots, -\boldsymbol{z}_N], \quad \boldsymbol{Z}_+ = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N], \quad \boldsymbol{Z} = [\boldsymbol{Z}_-, \boldsymbol{Z}_+],$$

where $\boldsymbol{Z}$ is centrally symmetric. The key observation Khachiyan (1996); Nesterov and Nemirovski (1994) at this stage is that

$$\text{MVEE}(\boldsymbol{X}) = \text{MVEE}(\boldsymbol{Z}) \cap \mathcal{H}, \tag{3.20}$$

where $\mathcal{H}$ is the hyperplane

$$\mathcal{H} = \{\boldsymbol{z} \in \mathbb{R}^{d+1} : z_{d+1} = 1\}$$

Since $\boldsymbol{Z}$ is symmetric, MVEE($\boldsymbol{Z}$) is centered at origin. Thus the problem of finding MVEE($\boldsymbol{Y}$) is reduced to the following

$$\begin{aligned} \text{minimize}_{\text{over } \boldsymbol{Q}} \quad & -\log\det \boldsymbol{Q} \\ \text{subject to} \quad & \boldsymbol{z}_k^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{z}_k \leq 1, \quad k = 1, \ldots, N \end{aligned}$$

Recall that for $\epsilon > 0$ the ellipsoid $\mathcal{E}(\boldsymbol{E}, \boldsymbol{c})$ is a $(1+\epsilon)$ approximation to MVEE($\boldsymbol{X}$) if

$$\boldsymbol{X} \subseteq \mathcal{E}(\boldsymbol{E}, \boldsymbol{c}) \quad \text{and} \quad \text{vol } \mathcal{E}(\boldsymbol{E}, \boldsymbol{c}) \leq (1+\epsilon)\text{vol MVEE}(\boldsymbol{X}). \tag{3.21}$$

In the Khachiyan algorithm Khachiyan (1996) the $(1+\epsilon)$ approximation (3.21) is obtained by constructing a series of ellipsoids

$$\mathcal{E}(\boldsymbol{Q}_i, \boldsymbol{0}) = \left\{\boldsymbol{z} \in \mathbb{R}^{d+1} : \boldsymbol{z}^{\mathrm{T}} \boldsymbol{Q}_i \boldsymbol{z} \leq 1\right\}$$

with

$$\boldsymbol{Q}_i = \frac{1}{d+1}(\Pi(\boldsymbol{u}_i))^{-1},$$

where $\Pi : \mathbb{R}^N \mapsto \mathbb{R}^{(d+1)\times(d+1)}$ is an operator such that

$$\Pi(\boldsymbol{u}) = \sum_{k=1}^{N} u_k \boldsymbol{z}_k \boldsymbol{z}_k^{\mathrm{T}}.$$

The vector $\boldsymbol{u}_i \in \mathbb{R}^N$ is updated iteratively starting from

$$\boldsymbol{u}_0 = \frac{1}{N}\mathbf{1} = \left[\frac{1}{N}, \ldots, \frac{1}{N}\right]$$

until the desired convergence is achieved. Note that the matrix $\boldsymbol{\Sigma}$ is used to form the initial ellipsoid

$$\boldsymbol{\Sigma} = \frac{1}{N}\boldsymbol{Z}_+\boldsymbol{Z}_+^{\mathrm{T}} = \Pi(\boldsymbol{u}_0), \quad \mathcal{E}(\boldsymbol{Q}_0, \mathbf{0}) = \left\{\boldsymbol{z} \in \mathbb{R}^{d+1} : \boldsymbol{z}^{\mathrm{T}} \frac{1}{(d+1)} \boldsymbol{\Sigma}^{-1} \boldsymbol{z} \leq 1\right\},$$

whose axes are defined by the eigenvectors of the matrix $\boldsymbol{\Sigma}$. In this sense the method based on PCA, when 'lifted' to higher dimensions, forms the initial step of the Khachiyan algorithm. However, the method developed in this section is supported by a scaling procedure that determines the axes length and the resulting ellipsoid is different. Assume that after scaling the ellipsoid

$$\mathcal{E}(\boldsymbol{P}, \mathbf{0}) = \left\{\boldsymbol{P}\boldsymbol{z} : \|\boldsymbol{z}\|_2 \leq 1, \quad \boldsymbol{z} \in \mathbb{R}^{d+1}\right\}$$

is obtained. On substituting

$$\boldsymbol{E} = (\boldsymbol{P}\boldsymbol{P}^{\mathrm{T}})^{-1} = \begin{bmatrix} \boldsymbol{E}_d & \boldsymbol{e} \\ \boldsymbol{e}^{\mathrm{T}} & e \end{bmatrix},$$

the alternative representation

$$\mathcal{E}(\boldsymbol{E}, \mathbf{0}) = \left\{\boldsymbol{z} \in \mathbb{R}^{d+1} : \boldsymbol{z}^{\mathrm{T}}\boldsymbol{E}\boldsymbol{z} \leq 1\right\}$$

is obtained. Using (3.24), the ellipsoid in $\mathbb{R}^d$ is given as the set

$$\left\{\boldsymbol{x} \in \mathbb{R}^d : [\boldsymbol{x}^{\mathrm{T}} \;\; 1] \begin{bmatrix} \boldsymbol{E}_d & \boldsymbol{e} \\ \boldsymbol{e}^{T} & e \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} \leq 1\right\}$$

and the ellipsoid equation yields

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{E}_d\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{e} + \boldsymbol{e}^{\mathrm{T}}\boldsymbol{x} + e \leq 1.$$

Now let $\boldsymbol{c} = -\boldsymbol{E}_d^{-1}\boldsymbol{e}$ ($\boldsymbol{e} = -\boldsymbol{E}_d\boldsymbol{c}$) and adding and subtracting $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{E}_d\boldsymbol{c}$ gives

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{E}_d\boldsymbol{x} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{E}_d\boldsymbol{c} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{E}_d\boldsymbol{x} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{E}_d\boldsymbol{c} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{E}_d\boldsymbol{c} + e \leq 1.$$

Hence

$$(\boldsymbol{x}-\boldsymbol{c})^{\mathrm{T}}\boldsymbol{E}_d(\boldsymbol{x}-\boldsymbol{c}) \leq 1+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{E}_d\boldsymbol{c}-e,$$

resulting in the ellipsoid equation

$$(\boldsymbol{x}-\boldsymbol{c})^{\mathrm{T}}\frac{\boldsymbol{E}_d}{1+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{E}_d\boldsymbol{c}-e}(\boldsymbol{x}-\boldsymbol{c}) \leq 1.$$

In $\mathbb{R}^d$ the resulting ellipsoid has representation

$$\mathcal{E}(\boldsymbol{M},\boldsymbol{c}) = \left\{\boldsymbol{x}\in\mathbb{R}^d : (\boldsymbol{x}-\boldsymbol{c})^{\mathrm{T}}\boldsymbol{M}(\boldsymbol{x}-\boldsymbol{c}) \leq 1\right\},$$

where the matrix $\boldsymbol{M}$ and center $\boldsymbol{c}$, respectively, are

$$\boldsymbol{M} = \frac{\boldsymbol{E}_d}{1+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{E}_d\boldsymbol{c}-e}, \quad \boldsymbol{c} = -\boldsymbol{E}_d^{-1}\boldsymbol{e}.$$

The representation of the ellipsoid as the image of a ball yields

$$\mathcal{E}(\boldsymbol{P}_d,\boldsymbol{c}) = \{\boldsymbol{P}_d\boldsymbol{x}+\boldsymbol{c} : \boldsymbol{x}\in\mathbb{R}^d, \|\boldsymbol{x}\|_2 \leq 1\},$$

where $\boldsymbol{P}_d = \boldsymbol{M}^{-1/2}$.

Numerical tests

Considering simple planar problems, note that the *Lifted PCA* and *PCA* produce different ellipsoids and they should be treated as separate methods. Example results are given in Figure 3.8. The ellipsoid marked red produced by the *PCA* method and the ellipsoid marked black obtained by *Lifted PCA* method have similar volume but they are essentially different. The ellipsoid marked green is calculated by the *Khachiyan* algorithm with a tolerance of $\epsilon = 10^{-3}$.

In the performance tests the following methods were compared.

- LMI method (Interior Point method solver, SeDuMi Sturm (1999))
- The Khachiyan algorithm with tolerance levels $\epsilon = 10^{-1}, 10^{-2}$ and 10^{-3}, respectively
- PCA method
- Lifted PCA method

We compared the volume of the ellipsoids and the time of execution. Recall that for the ellipsoid defined as the image of a d-ball of unit radius

$$\mathcal{E}(\boldsymbol{A},\boldsymbol{c})\{\boldsymbol{A}\boldsymbol{x}+\boldsymbol{c} : \boldsymbol{x}\in\mathbb{R}^d, \quad \|\boldsymbol{x}\|_2 \leq 1\},$$

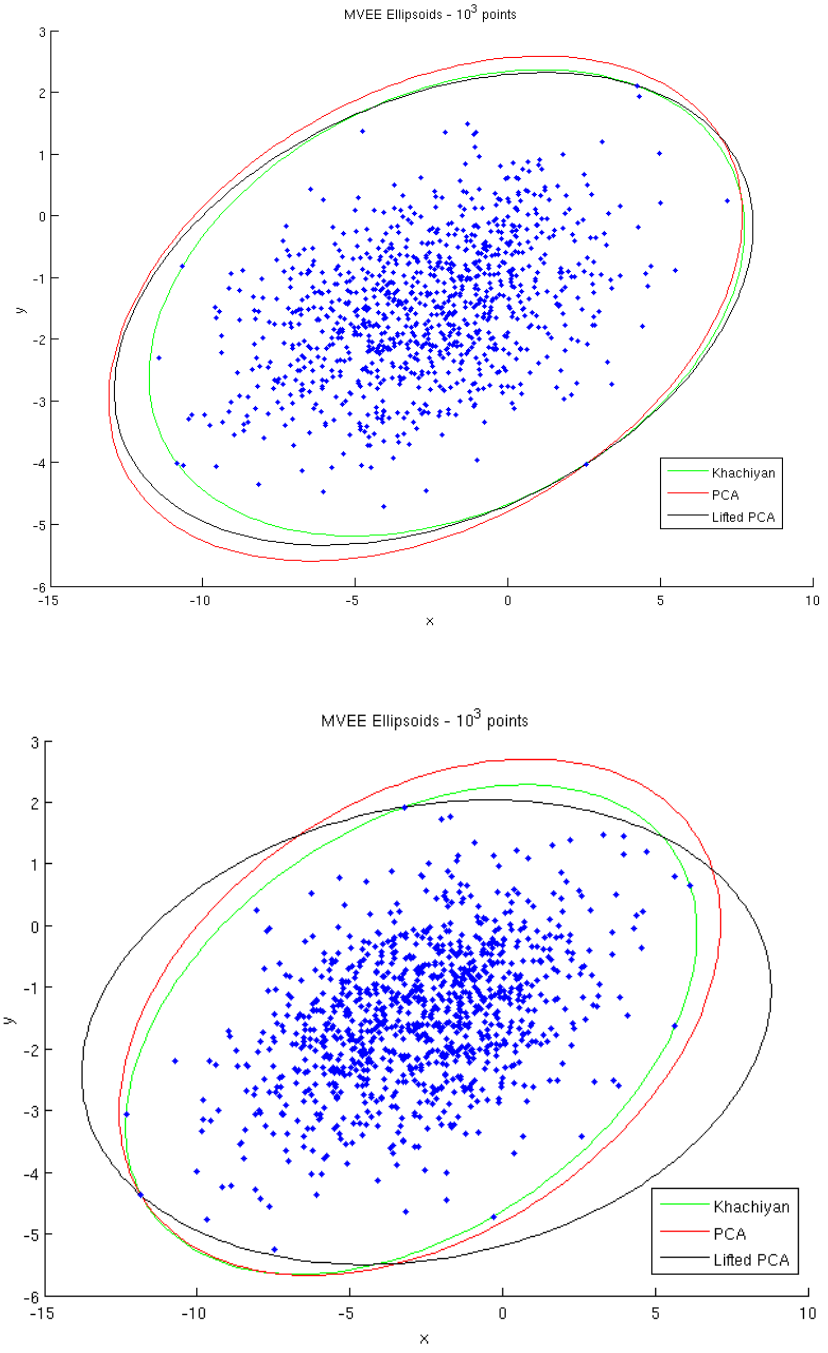

FIGURE 3.8: Approximations to MVEE in $\mathbb{R}^2$ calculated by different methods. The plots show that *Lifted PCA* does not necessary produce results close to the *Khachiyan* or *PCA* methods.

the volume of the full dimensional ellipsoid is given by

$$\mathrm{vol}(\mathcal{E}) = \beta_d \det \boldsymbol{A}, \qquad \beta_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)},$$

where β_d is the volume of the unit d-ball. For two ellipsoids defined by $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ we

introduce

$$\gamma_{\boldsymbol{A}_1/\boldsymbol{A}_2} = \sqrt[d]{\frac{\text{vol}(\mathcal{E}(\boldsymbol{A}_1))}{\text{vol}(\mathcal{E}(\boldsymbol{A}_2))}}.$$

The γ-ratio determines by which factor axes of one ellipsoid must be multiplied in order to have the same volume as the other, i.e.

$$\text{vol}(\mathcal{E}(\boldsymbol{A}_1)) = \text{vol}\mathcal{E}(\gamma \boldsymbol{A}_2)).$$

LOG-VOLUME (POINTS=10^3)

dim	PCA	Lifted PCA	LMI (SeDuMi)	K. $\epsilon = 10^{-1}$	K. $\epsilon = 10^{-2}$	K. $\epsilon = 10^{-3}$
2	5.76551	5.79919	5.70872	5.42415	5.66259	5.70375
3	5.53576	5.57044	5.42192	5.06427	5.37193	5.41605
4	9.0760	9.0669	8.8831	8.5487	8.8296	8.8768
5	10.7036	10.7794	20.0089	9.89259	10.1867	10.2355
10	23.9601	23.9586	38.9069	22.5006	22.791	22.8363
15	35.9185	35.8856	63.2548	33.3952	33.6709	33.7197
20	53.7483	53.9172	129.306	50.4073	50.6905	50.737
50	122.2612	122.5670	229.8759	114.8549	115.1406	115.1847

TIME (POINTS=10^3)

dim	PCA	Lifted PCA	LMI (SeDuMi)	K. $\epsilon = 10^{-1}$	K. $\epsilon = 10^{-2}$	K. $\epsilon = 10^{-3}$
2	0.019254	0.014615	2.12466	0.023189	0.11345	1.04374
3	0.01353	0.013749	3.56992	0.031978	0.199314	1.70919
4	0.0202	0.0171	4.2159	0.0581	0.3473	2.9575
5	0.013582	0.014659	11.7073	0.071477	0.460068	4.04431
10	0.015525	0.01576	6.99843	0.268748	1.92072	17.1915
15	0.038233	0.024704	37.1233	0.583153	3.88356	38.7803
20	0.018632	0.019201	72.3632	1.02944	7.71546	74.8426
50	0.0482	0.0411	705.2070	8.2035	72.9755	729.6316

TABLE 3.1: Logarithm of volume and time of execution in seconds vs dimension for 10^3 points. K-Khachiyan algorithm, ϵ denotes tolerance.

Tables 3.1 and 3.2 contain results of tests performed on an *Intel Core i3* under *Linux*. Note that the *LMI method* was outperformed in every test and should not be used in this application area. The *Lifted PCA* and *PCA* produce very similar results. Note also that for the *Khachiyan* method the volume differs very slightly for different tolerance levels but these greatly influence the time of execution. The *Khachiyan* method constructs a series of ellipsoids

$$\mathcal{E}_1 \subset \mathcal{E}_2 \subset \cdots \mathcal{E}_k \subset \cdot$$

which converges to the *MVEE*. Thus the volume grows with tolerance but very slightly in these tests. The important fact is that the *Khachiyan* algorithm produces a lower volume approximation for the MVEE and there is no guarantee that it contains all the points. Next a comparison is given between

LOG-VOLUME (DIM=10)

Points	PCA	Lifted PCA	LMI (SeDuMi)	K. $\epsilon = 10^{-1}$	K. $\epsilon = 10^{-2}$	K. $\epsilon = 10^{-3}$
100	23.4219	23.4848	31.1792	21.4609	21.754	21.799
200	22.011	22.082	25.0948	20.2269	20.506	20.5529
500	24.7301	24.7398	73.3696	23.1999	23.487	23.5365
1000	28.2505	28.2243	53.2986	26.5686	26.8545	26.9015
2000	27.1666	27.1669	91.5111	25.7928	26.1025	26.1471
5000	25.9439	26.0212	55.9731	24.5013	24.8326	24.881
10000	26.7204	26.7354	55.4568	25.5825	25.8717	25.9157

TIME (DIM=10)

Points	PCA	Lifted PCA	LMI (SeDuMi)	K. $\epsilon = 10^{-1}$	K. $\epsilon = 10^{-2}$	K. $\epsilon = 10^{-3}$
100	0.005087	0.005072	1.01254	0.023846	0.213075	2.01656
200	0.003929	0.00494	1.5844	0.053168	0.401307	3.80675
500	0.017511	0.008626	4.61823	0.12975	0.905349	8.79397
1000	0.016124	0.016771	10.033	0.262211	1.8601	17.3088
2000	0.027962	0.028722	17.1304	0.509298	3.9045	35.2845
5000	0.064767	0.065532	42.7013	1.22875	9.11955	86.2326
10000	0.137571	0.129724	131.618	2.83523	19.7803	170.491

TABLE 3.2: Time of execution in seconds vs points in 10-dimensional space . K-Khachiyan algorithm, ϵ denotes tolerance.

DIMENSION (POINTS = 10^4)

dim	PCA log-vol	K log-vol	$\gamma_{\text{PCA/K}}$	PCA time	K time	PCA/K time ratio
10	28.6967	27.5247	1.12435	0.157498	2.85559	18.131
20	55.5204	53.0304	1.13258	0.151705	10.4567	68.928
50	128.301	121.802	1.13879	0.227958	91.6697	402.134
100	254.958	240.525	1.15526	0.336734	630.138	1871.32
200	767.512	741.3994	1.13947	0.817819	6919.89	8461.4
500	2136.05	–	–	5.02039	–	–
1000	4576.5	–	–	35.6404	–	–

POINTS (DIM=50)

points	PCA log-vol	K log-vol	$\gamma_{\text{PCA/K}}$	PCA time	K time	PCA/K time ratio
10^2	118.799	109.601	1.20197	0.026503	0.22557	8.51111
10^3	132.215	122.26	1.22032	0.034145	7.9283	232.195
10^4	129.532	122.888	1.14211	0.202416	91.9413	454.219
10^5	126.229	120.251	1.127	1.97794	923.628	466.964
10^6	136.440	129.801	1.132	19.8724	9426.294	474.341

TABLE 3.3: Logarithm of volume and time of execution in seconds vs dimension and points. PCA - *Lifted PCA*, K - *Khachiyan* method with $\epsilon = 10^{-1}$.

- Lifted PCA
- Khachiyan with $\epsilon = 10^{-1}$

Table 3.3 gives the numerical results. These show that the time ratio depends on dimension of the space rather than the number of the points. For example in 200-dimensional

space and for 10^4 points the first method is over 8000 times faster than the second. The *Lifted PCA* method proved to be robust and fast even for high dimension and a very large number of points. Both methods have much in common as it was shown earlier. Sometimes it is reasonable to enrich the *Lifted PCA* with few iterations from the *Khachiyan* algorithm in order to improve the results, depending on the dimension of the space.

3.3.4 Choosing the vertices of the polytope

Assume the ellipsoid is defined by the positive definite matrix $\boldsymbol{E}$ and center $\boldsymbol{c}$

$$\mathcal{E}^\star(\boldsymbol{E}, \boldsymbol{c}) = \{\boldsymbol{x} \in \mathbb{R}^d : (\boldsymbol{x} - \boldsymbol{c})^{\mathrm{T}} \boldsymbol{E} (\boldsymbol{x} - \boldsymbol{c}) \leq 1\}.$$

Using the Cholesky factorization method construct the matrix $\boldsymbol{H}$ such that

$$\boldsymbol{E} = \boldsymbol{H}^{\mathrm{T}} \boldsymbol{H}$$

and let

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{y}, \quad \boldsymbol{f} = \boldsymbol{H}\boldsymbol{e}.$$

Then the image of the set $\mathcal{E}^\star(\boldsymbol{E}, \boldsymbol{c})$ can be written as

$$\boldsymbol{H}\left(\mathcal{E}^\star(\boldsymbol{E}, \boldsymbol{c})\right) = \{\boldsymbol{z} \in \mathbb{R}^d : (\boldsymbol{z} - \boldsymbol{f})^{\mathrm{T}} (\boldsymbol{z} - \boldsymbol{f}) \leq 1\} = \mathcal{B}(\boldsymbol{f}, 1) \tag{3.22}$$

and $\boldsymbol{H} : \mathbb{R}^d \to \mathbb{R}^d$ defined by

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{y} \qquad \boldsymbol{y} = \boldsymbol{H}^{-1}\boldsymbol{z},$$

maps the ellipsoid into the ball of unit radius centered at $\boldsymbol{f}$. Also

$$\boldsymbol{z}' = \boldsymbol{z} - \boldsymbol{f}, \quad \boldsymbol{z} = \boldsymbol{z}' + \boldsymbol{f},$$

transforms the ball centered at $\boldsymbol{f}$ into the ball centered at the origin.

Hypercube

The most simple way is to enclose the ball in a hypercube $\mathcal{C}$. Define the 2^d vertices as $[\pm 1, \ldots, \pm 1]^{\mathrm{T}}$. Then the construction is illustrated in Figure 3.9
The volume is given by

$$\mathrm{vol}\,(\mathcal{C}) = 2^d.$$

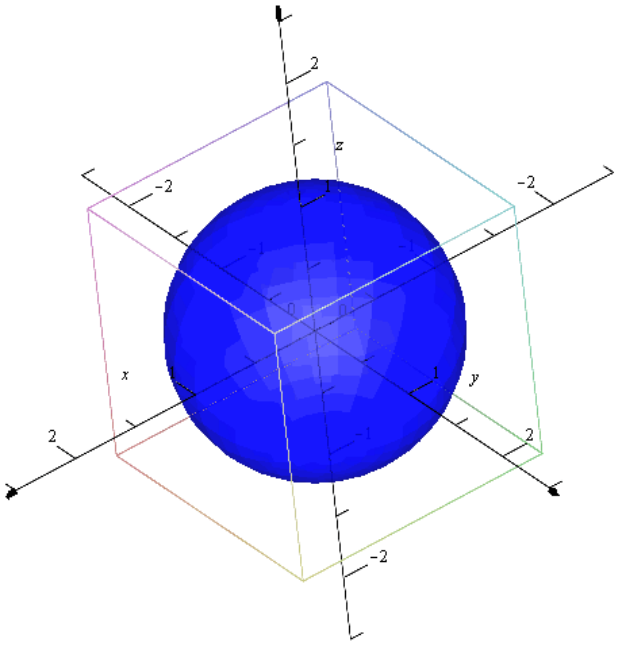

FIGURE 3.9: Chosen point in $\mathbb{R}^3$.The presented figure is a cube (hexahedron).

Hyperdipyramid

Let $k > 0$ be fixed and $\mathcal{D}(k)$ denotes the convex set defined by the vertices

$$\boldsymbol{d}_i(k) = k \cdot \boldsymbol{e}_i, \quad i = 1, \ldots, d,$$

where $\boldsymbol{e}_i = [0, \ldots, 0, 1_{\text{ith entry}}, 0, \ldots, 0]^{\mathrm{T}}$. Equivalently

$$\begin{aligned} \boldsymbol{d}_1(k) &= [\,k, 0, 0, \ldots, 0]^{\mathrm{T}} \\ \boldsymbol{d}_2(k) &= [\,0, k, 0, \ldots, 0]^{\mathrm{T}} \\ \cdots \quad &\cdots \quad \cdots \\ \boldsymbol{d}_d(k) &= [\,0, 0, 0, \ldots, k\,]^{\mathrm{T}}, \end{aligned}$$

i.e.,

$$\mathcal{D}(k) := \boldsymbol{Co}\{\pm\boldsymbol{d}_1(k), \ldots, \pm\boldsymbol{d}_d(k)\}. \tag{3.23}$$

Now consider the points $\boldsymbol{d}_i(k)$, $i = 1, \ldots, d$ with positive entries. These span the $(d-1)$-dimensional hyperplane in $\mathbb{R}^d$ and

$$\boldsymbol{p} = [\,p_1, p_2, \ldots, p_d\,]^{\mathrm{T}} \in \mathbb{R}^d$$

belongs to this plane if for some k

$$p_1 + p_2 + \cdots + p_d = k. \tag{3.24}$$

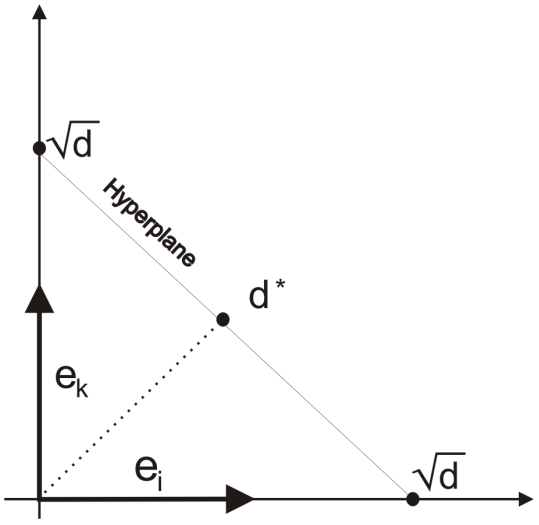

FIGURE 3.10: Vertices $\sqrt{d}\boldsymbol{e}_j, j = 1, \ldots, d$ span the hyperplane. By the symmetry, the closest hyperplane point to the origin $\boldsymbol{d}^\star$ has all coordinates equal.

By symmetry, the point $\boldsymbol{d}^\star$ of the plane that is closest to the center of the ball is the point with equal coordinates, where these also have to satisfy (3.24). Hence

$$\boldsymbol{d}^\star = [\ \frac{k}{d}, \ldots, \frac{k}{d}\ ]^{\mathrm{T}}$$

and the distance to the origin is

$$\|\boldsymbol{d}^\star\|_2 = \sqrt{d \cdot \frac{k^2}{d^2}} = \sqrt{\frac{k^2}{d}} = \frac{k}{\sqrt{d}}$$

and since this point belongs to the surface of the ball

$$\frac{k}{\sqrt{d}} = 1 \quad \Rightarrow \quad k = \sqrt{d}.$$

By symmetry we also have that if we take the convex set $\mathcal{D} = \mathcal{D}(\sqrt{d})$, then by the next result the d-ball $\mathcal{B}(\mathbf{0}, 1)$ is enclosed by $\mathcal{D}$:

Theorem 3.6. *Let $\boldsymbol{x} \in \mathbb{R}^d$ be a point that belongs to the d-ball $\mathcal{B}_d(0,1)$. Then $\boldsymbol{x}$ belongs to $\mathcal{D}$.*

PROOF: Since $\boldsymbol{x} \in \mathbb{R}^d$ we have

$$\|\boldsymbol{x}\|_2^2 = \sum_i x_i^2 \leq 1$$

and it is required to prove that there exists a convex combination of the vertices of $\mathcal{D}$ that equals $\boldsymbol{x}$. Equivalently we need to prove the existence of $\alpha_{i,1} \geq 0, \alpha_{i,2} \geq 0$, $i = 1, \ldots, d$ such that

$$\boldsymbol{x} = \sum_i (\alpha_{i,1} - \alpha_{i,2})\sqrt{d}\boldsymbol{e}_i, \quad \sum_i \alpha_{i,1} + \alpha_{i,2} = 1.$$

Hence for $i = 1, \ldots, d$

$$(\alpha_{i,1} - \alpha_{i,2})\sqrt{d} = x_i \qquad \left(\text{or } (\alpha_{i,1} - \alpha_{i,2}) = \frac{x_i}{\sqrt{d}} \right) \tag{3.25}$$

and a nonnegative solution $\alpha_{i,1} \geq 0, \alpha_{i,2} \geq 0$ to the problem (3.25) exists provided

$$\alpha_{i,1} + \alpha_{i,2} \geq \frac{|x_i|}{\sqrt{d}}. \tag{3.26}$$

The solution for $i = 1, \ldots, d$ is of the form

$$\alpha_{i,1} = \frac{x_i}{\sqrt{d}} + \alpha_{i,2} \geq 0, \quad \alpha_{i,2} \geq \frac{1}{2}\left(\frac{|x_i|}{\sqrt{d}} - \frac{x_i}{\sqrt{d}} \right) \geq 0, \tag{3.27}$$

or, equivalently,

$$\alpha_{i,2} = \alpha_{i,1} - \frac{x_i}{\sqrt{d}} \geq 0, \quad \alpha_{i,1} \geq \frac{1}{2}\left(\frac{|x_i|}{\sqrt{d}} + \frac{x_i}{\sqrt{d}} \right) \geq 0$$

and the solution satisfies the convexity condition

$$\sum_i \alpha_{i,1} + \alpha_{i,2} = \sum_i \left(2\alpha_{i,1} - \frac{x_i}{\sqrt{d}} \right) = \sum_i \left(2\alpha_{i,2} + \frac{x_i}{\sqrt{d}} \right) = 1, \tag{3.28}$$

i.e.,

$$\sum_i \alpha_{i,1} = \frac{1}{2} + \frac{x_i}{2\sqrt{d}}, \qquad \left(\text{or } \sum_i \alpha_{i,2} = \frac{1}{2} - \frac{x_i}{2\sqrt{d}} \right). \tag{3.29}$$

Consider the following optimization problem

$$\begin{array}{ll} \text{maximize} & \|\boldsymbol{x}\|_1 \\ \text{subject to} & \|\boldsymbol{x}\|_2^2 = 1 \end{array}$$

where if $\boldsymbol{x}^\star$ is an optimal solution then the point with absolute value entries $\boldsymbol{x}^{\star\star} = |\boldsymbol{x}^\star|$

is also the optimal solution. Moreover, it is an optimal solution of the problem with differentiable objective function

$$\begin{array}{ll} \text{maximize} & \sum_i x_i \\ \text{subject to} & \|\boldsymbol{x}\|_2^2 = 1 \end{array}$$

Define the Lagrangian

$$\mathcal{L}(\boldsymbol{x}, \lambda) = \sum_i x_i + \lambda(\|\boldsymbol{x}\|_2^2 - 1)$$

and necessary conditions for optimality yield

$$\frac{\partial \mathcal{L}(\boldsymbol{x}, \lambda)}{\partial x_i} = 1 + 2\lambda x_i = 0, \qquad i = 1, \ldots, d.$$

Subtracting two equations $i \neq j$ gives for $\lambda > 0$

$$\lambda(x_i - x_j) = 0 \iff x_i = x_j \quad i \neq j$$

and since $\|\boldsymbol{x}\|_2^2 = 1$ we have

$$x_i^\star = \frac{1}{\sqrt{d}}, \quad \sum_i x_i^\star = \sqrt{d}.$$

Hence

$$\max_{\|\boldsymbol{x}\|_2=1} \sum_i \frac{|x_i|}{\sqrt{d}} = \max_{\|\boldsymbol{x}\|_2 \leq 1} \sum_i \frac{|x_i|}{\sqrt{d}} = \frac{\sqrt{d}}{\sqrt{d}} = 1, \tag{3.30}$$

which implies

$$\sum_i \frac{|x_i|}{\sqrt{d}} \leq 1. \tag{3.31}$$

and also

$$\sum_i \frac{x_i}{\sqrt{d}} \leq 1, \quad \text{or} \quad 1 - \sum_i \frac{x_i}{\sqrt{d}} \geq 0$$

By the convexity condition (3.28)

$$\sum_i \alpha_{i,2} = \frac{1}{2}\left(1 - \frac{|x_i|}{\sqrt{d}}\right) \geq 0 \tag{3.32}$$

There are infinitely many choices of $\alpha_{i,2} \geq 0$, $i = 1, \ldots, N$, such that (3.32) is satisfied. The coefficients $\alpha_{i,1}$, $i = 1, \ldots, N$ can be determined from (3.27). The convexity condi-

tion (3.28) is guaranteed by (3.32). The particular solution is for some k, $1 \leq k \leq d$

$$
\begin{aligned}
&\text{for } i = 1, \ldots, d, \quad i \neq k: \\
&\qquad \alpha'_{i,1} = \frac{x_i}{\sqrt{d}}, \quad \alpha'_{i,2} = 0, \quad \text{if } x_i \geq 0 \\
&\qquad \alpha'_{i,1} = 0, \quad \alpha'_{i,2} = \frac{|x_i|}{\sqrt{d}}, \quad \text{if } x_i < 0 \\
&\alpha'_{k,1} = \frac{x_k}{\sqrt{d}} + \frac{1}{2}\left(1 - \sum_i \frac{|x_i|}{\sqrt{d}}\right), \quad \alpha'_{k,2} = \frac{1}{2}\left(1 - \sum_i \frac{|x_i|}{\sqrt{d}}\right), \quad \text{if } x_k \geq 0 \\
&\alpha'_{k,1} = \frac{1}{2}\left(1 - \sum_i \frac{|x_i|}{\sqrt{d}}\right), \quad \alpha'_{k,2} = \frac{|x_k|}{\sqrt{d}} + \frac{1}{2}\left(1 - \sum_i \frac{|x_i|}{\sqrt{d}}\right), \quad \text{if } x_k < 0,
\end{aligned}
$$

also

$$(\alpha'_{i,1} - \alpha'_{i,1})\sqrt{d} = x_i, \quad i = 1, \ldots, d$$

and $\alpha'_{i,1} \geq 0, \alpha'_{i,2} \geq 0$ for $i = 1, \ldots, d$, $i \neq k$. By (3.31) we have that $\alpha'_{k,1} \geq 0$ and $\alpha'_{k,2} \geq 0$ and also

$$\sum_i \alpha_{i,1} + \alpha_{i,2} = \frac{|x_k|}{\sqrt{d}} + \sum_{i \neq k} \frac{|x_i|}{\sqrt{d}} + \left(1 - \sum_i \frac{|x_i|}{\sqrt{d}}\right) = 1.$$

Hence if $\boldsymbol{x} \in \mathcal{B}_d(0,1)$ then $\boldsymbol{x} \in \mathcal{D}$. This means the entire ball is enclosed by the convex hull of $\mathcal{D}$ and proof is complete. □

Figure 3.11 gives a graphical interpretation of the above result

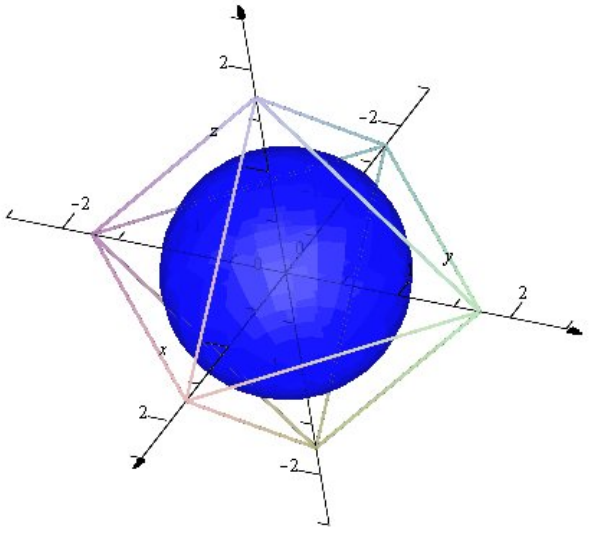

FIGURE 3.11: Chosen point in $\mathbb{R}^3$.The plot is an octahedron (square dipyramid).

Consider a d-dimensional simplex in $\mathbb{R}^d$ defined by the $d+1$ points $\mathbf{0}, \boldsymbol{d}_1(\sqrt{d}), \ldots, \boldsymbol{d}_d(\sqrt{d})$

$$\Delta_{\sqrt{d}} = \{[x_1, \ldots, x_d]^{\mathrm{T}} : \sum_{i=1}^{d} x_i \leq \sqrt{d} \text{ and } x_i \geq 0 \text{ for all } i\}.$$

Then $\mathrm{vol}(\mathcal{D}(\sqrt{d})) = 2^d \mathrm{vol}(\Delta^d)$ and the volume of the simplex is given by Sommerville (1958)

$$\mathrm{vol}(\Delta_{\sqrt{d}}) = \frac{1}{d!} \det \begin{bmatrix} \boldsymbol{d}_1(\sqrt{d}) & \ldots & \boldsymbol{d}_d(\sqrt{d}) & \mathbf{0} \\ 1 & \ldots & 1 & 1 \end{bmatrix}.$$

By the Laplace expansion of the determinant with respect to the last column

$$\mathrm{vol}(\Delta_{\sqrt{d}}) = \frac{1}{d!} \det(\sqrt{d} \boldsymbol{I}_{d \times d}) = \frac{\sqrt{d}^d}{d!} = \frac{d^{d/2}}{d!}$$

and hence for even d

$$\mathrm{vol}(\Delta_{\sqrt{d}}) = \frac{d}{d} \cdot \frac{d}{d-1} \cdots \frac{d}{d-d/2+1} \cdot \frac{1}{(d/2)!} < \left(\frac{d}{d-d/2+1} \right)^{d/2} \cdot \frac{1}{(d/2)!},$$

or

$$\mathrm{vol}(\Delta_{\sqrt{d}}) < \left(\frac{d}{d/2} \right)^{d/2} \cdot \frac{1}{(d/2)!} = \frac{2^{d/2}}{(d/2)!}.$$

For odd d

$$\mathrm{vol}(\Delta_{\sqrt{d}}) = \frac{d}{d} \cdot \frac{d}{d-1} \cdots \frac{\sqrt{d}}{d-\lceil d/2 \rceil + 1} \cdot \frac{1}{\lfloor d/2 \rfloor !} < \left(\frac{d}{\lceil d/2 \rceil} \right)^{\lceil d/2 \rceil} \cdot \frac{1}{\lfloor d/2 \rfloor !} < \frac{2^{\lceil d/2 \rceil}}{\lfloor d/2 \rfloor !}.$$

Combining both these results gives

$$\mathrm{vol}(\Delta_{\sqrt{d}}) < \frac{2^{\lfloor d/2 \rfloor + 1}}{\lfloor d/2 \rfloor !}$$

and hence $\Delta_{\sqrt{d}} \ll 1$ for large d

$$\mathrm{vol}(\mathcal{D}) \ll \mathrm{vol}(\mathcal{C}).$$

Moreover

$$0 \leq \lim_{d \to \infty} \mathrm{vol}(\Delta_{\sqrt{d}}) \leq \lim_{d \to \infty} \frac{2^{\lfloor d/2 \rfloor + 1}}{\lfloor d/2 \rfloor !} = 0$$

and

$$0 \leq \lim_{d \to \infty} \mathrm{vol}(\mathcal{D}) \leq \lim_{d \to \infty} 2^d \frac{2^{\lfloor d/2 \rfloor + 1}}{\lfloor d/2 \rfloor !} = 0$$

and

$$\lim_{d \to \infty} \mathrm{vol}(\mathcal{C}) = +\infty.$$

The number of vertices is $2d$ in comparison to 2^d in the case of the hypercube.

Improved hyperdipyramid

Further improvement may be obtained if redundant parts are eliminated. The d-dimensional unit hypercube contains the entire unit radius ball and the improvement is to include the common part of the polytope obtained earlier and the hypercube. Since both polytopes contain all points , so does the common part

$$\mathcal{D}_{\text{imp}} = \mathcal{D} \cap \mathcal{C}.$$

Consider the situation when the ball is centered at the origin and the vertices of a polytope $\mathcal{D}$ are defined as $\pm\sqrt{d} \cdot \boldsymbol{e}_i$, $i = 1, \ldots, d$ where $\boldsymbol{e}_i = [0, \ldots, 0, 1_{\text{ith entry}}, 0, \ldots, 0]^{\text{T}}$ are unit Cartesian vectors. The vertex $\sqrt{d} \cdot \boldsymbol{e}_k$. is now replaced by the set of vertices located on the hyperplane $\boldsymbol{x}_k = 1$ in the case of $\mathcal{D}_{\text{imp}}$, see Figure 3.12.

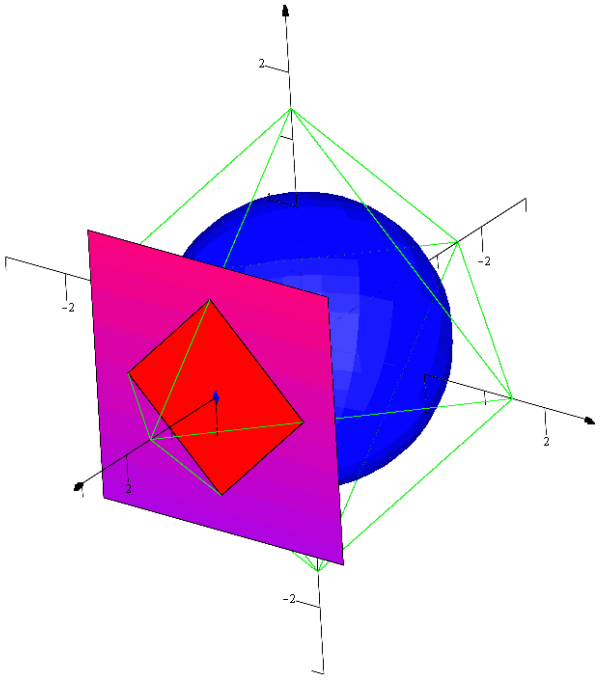

FIGURE 3.12: The cutting plane in $\mathbb{R}^3$ for x.

Now consider the line between $\sqrt{d} \cdot \boldsymbol{e}_k$ and the other vertex $\sqrt{d} \cdot \boldsymbol{e}_i$, $i \neq k$

$$t \cdot \sqrt{d}\,(\boldsymbol{e}_k - \boldsymbol{e}_i) + \sqrt{d}\boldsymbol{e}_i, \qquad t \in \mathbb{R}.$$

This line cuts the hyperplane $\boldsymbol{x}_k = 1$ at one point $(t = 1/\sqrt{d})$

$$\boldsymbol{p}_{k,i,1} = \boldsymbol{e}_k + (\sqrt{d} - 1)\boldsymbol{e}_i$$

and when considering the line from the vertex $-\sqrt{d} \cdot \boldsymbol{e}_i$, this point is

$$\boldsymbol{p}_{k,i,2} = \boldsymbol{e}_k - (\sqrt{d} - 1)\boldsymbol{e}_i.$$

Define the set

$$\mathcal{P}_k = \left\{ \boldsymbol{p}_{k,i,1}, \boldsymbol{p}_{k,i,2} : i = 1, \ldots, d, \quad i \neq k \right\}, \quad \text{card}(\mathcal{P}_k) = 2(d-1)$$

and the equivalent set for $-\sqrt{d}\boldsymbol{e}_k$ is

$$\mathcal{P}_{-k} = \left\{ \boldsymbol{p}_{-k,i,1}, \boldsymbol{p}_{-k,i,2} : i = 1, \ldots, d, \quad i \neq k \right\}, \quad \text{card}(\mathcal{P}_{-k}) = 2(d-1),$$

where

$$\boldsymbol{p}_{-k,i,1} = -\boldsymbol{e}_k + (\sqrt{d}-1)\boldsymbol{e}_i, \quad \boldsymbol{p}_{-k,i,2} = -\boldsymbol{e}_k - (\sqrt{d}-1)\boldsymbol{e}_i.$$

See Figure 3.13.

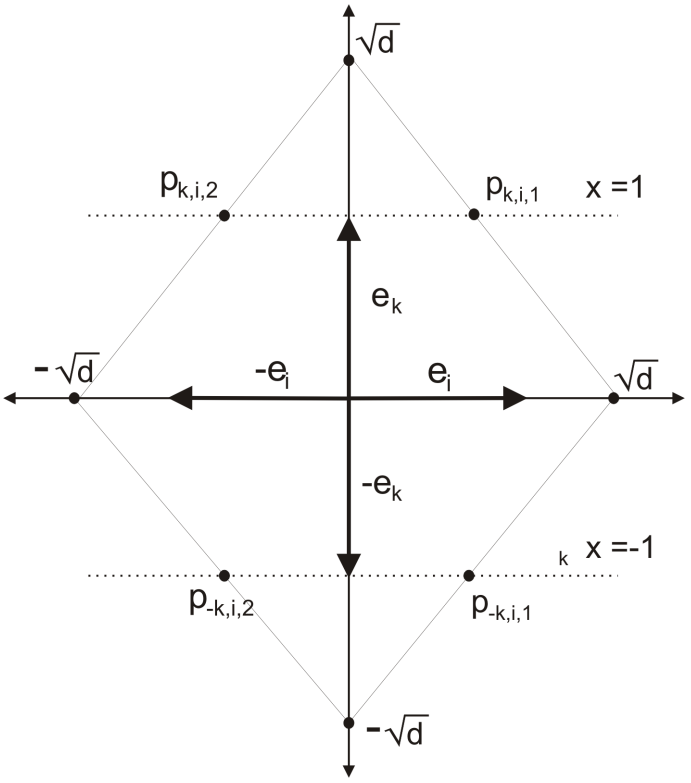

FIGURE 3.13: Consider the vertex $\sqrt{d}\boldsymbol{e}_k$. The straight lines coming from the vertices $\sqrt{d}\boldsymbol{e}_i$ and $-\sqrt{d}\boldsymbol{e}_i$ have two common intersection points with the hyperplane $x_k = 1$.

Lemma 3.7. *Assume that the point $\boldsymbol{x} \in \mathbb{R}^d$ belongs to $\mathcal{D}$. Then*

$$\sum_i |x_i| \leq \sqrt{d}.$$

PROOF: Suppose that there exists a convex combination $\alpha_{i,1} \geq 0$, $\alpha_{i,2} \geq 0$, $i = 1, \ldots, d$ such that

$$\boldsymbol{x} = \sum_i (\alpha_{i,1} - \alpha_{i,2})\sqrt{d}\boldsymbol{e}_i, \quad \sum_i \alpha_{i,1} + \alpha_{i,2} = 1$$

and note that

$$|\alpha_{i,1} - \alpha_{i,2}| = \frac{|x_i|}{\sqrt{d}}.$$

Summing over i gives

$$1 = \sum_i \alpha_{i,1} + \alpha_{i,2} = \sum_i |\alpha_{i,1}| + |\alpha_{i,2}| \geq \sum_i |\alpha_{i,1} - \alpha_{i,2}| = \sum_i \frac{|x_i|}{\sqrt{d}}$$

and hence

$$\sum_i |x_i| \leq \sqrt{d}.$$

□

Lemma 3.8. *Assume that the point $\boldsymbol{x} \in \mathbb{R}^d$ belongs to $\mathcal{D} \cap \mathcal{C}$. Then there exists a convex combination $\alpha_{i,1} \geq 0$, $\alpha_{i,2} \geq 0$, $i = 1, \ldots, d$ of the vertices of $\mathcal{D}$*

$$\boldsymbol{x} = \sum_i (\alpha_{i,1} - \alpha_{i,2})\sqrt{d}\boldsymbol{e}_i, \quad \sum_i \alpha_{i,1} + \alpha_{i,2} = 1, \tag{3.33}$$

such that

$$(\alpha_{i,1} + \alpha_{i,2})\sqrt{d} \leq 1, \qquad i = 1, \ldots, d.$$

Proof: Let

$$\beta = \left(1 - \sum_i \frac{|x_i|}{\sqrt{d}}\right), \quad 0 \leq \beta \leq 1,$$

where $\beta \geq 0$ is guaranteed by Lemma 3.7. Now let

$$\gamma_i = 1 - |x_i|, \qquad i = 1, \ldots, d$$

and since $\boldsymbol{x} \in \mathcal{C}$ we have $|x_i| \leq 1$, $i = 1, \ldots, d$ and $\gamma_i \geq 0$. Note also that

$$\gamma = \sum_i \gamma_i = d - \sum_i |x_i| \geq d - \sqrt{d}.$$

Using the ratio γ_i/γ introduce

$$\beta_i = \beta \frac{\gamma_i}{\gamma} \geq 0, \quad i = 1, \ldots, d$$

and define for $i = 1, \ldots, d$

$$\alpha'_{i,1} = \frac{x_i}{\sqrt{d}} + \frac{\beta_i}{2}, \quad \alpha'_{i,2} = \frac{\beta_i}{2}, \quad \text{if } x_i \geq 0$$
$$\alpha'_{i,1} = \frac{\beta_i}{2}, \quad \alpha'_{i,2} = \frac{|x_i|}{\sqrt{d}} + \frac{\beta_i}{2}, \quad \text{if } x_i < 0$$

Moreover

$$\alpha'_{i,1} - \alpha'_{i,2} = \frac{x_i}{\sqrt{d}}, \qquad \alpha'_{i,1} + \alpha'_{i,2} = \frac{|x_i|}{\sqrt{d}} + \beta_i$$

and summing over i gives

$$\sum_i \alpha'_{i,1} + \alpha'_{i,2} = \sum_i \frac{|x_i|}{\sqrt{d}} + \beta = \sum_i \frac{|x_i|}{\sqrt{d}} + \left(1 - \sum_i \frac{|x_i|}{\sqrt{d}}\right) = 1.$$

Also

$$\begin{aligned}
\alpha'_{i,1} + \alpha'_{i,2} &= \frac{|x_i|}{\sqrt{d}} + \beta_i \\
&= \frac{|x_i|}{\sqrt{d}} + \beta\frac{\gamma_i}{\gamma} \\
&= \frac{|x_i|}{\sqrt{d}} + \left(1 - \frac{\sum_i |x_i|}{\sqrt{d}}\right) \frac{1 - |x_i|}{d - \sum_i |x_i|} \\
&= \frac{|x_i|}{\sqrt{d}} + \frac{1 - |x_i|}{\sqrt{d}} \left(\frac{\sqrt{d} - \sum_i |x_i|}{d - \sum_i |x_i|}\right) \\
&\leq \frac{|x_i|}{\sqrt{d}} + \frac{1 - |x_i|}{\sqrt{d}} = \frac{1}{\sqrt{d}}
\end{aligned}$$

and it has been shown that for the point $\boldsymbol{x} \in \mathcal{D} \cap \mathcal{C}$ there exists a convex combination of vertices from $\mathcal{D}$ such that

$$(\alpha'_{i,1} + \alpha'_{i,2})\sqrt{d} \leq 1, \qquad i = 1, \ldots, d.$$

□

Theorem 3.9. *Let $\boldsymbol{x} \in \mathcal{R}^d$ be a point that belongs to $\mathcal{D} \cap \mathcal{C}$. Then $\boldsymbol{x}$ belongs to the polytope defined by the vertices*

$$\mathcal{P}_1 \cup \mathcal{P}_{-1} \cup \cdots \cup \mathcal{P}_d \cup \mathcal{P}_{-d}.$$

Proof: Consider the linear combination of the vertices from $\mathcal{P}_k$ and $\mathcal{P}_{-k}$ with nonnegative coefficients

$$\begin{aligned}
&\sum_{i \neq k} \left(\alpha_{k,i,1}\boldsymbol{p}_{k,i,1} + \alpha_{k,i,2}\boldsymbol{p}_{k,i,2} + \alpha_{-k,i,1}\boldsymbol{p}_{-k,i,1} + \alpha_{-k,i,2}\boldsymbol{p}_{-k,i,2}\right) \\
&\qquad = \sum_{i \neq k} (\alpha_{k,i,1} + \alpha_{k,i,2} - \alpha_{-k,i,1} - \alpha_{-k,i,2})\boldsymbol{e}_k \\
&\qquad\quad + \sum_{i \neq k} (\alpha_{k,i,1} + \alpha_{-k,i,1} - \alpha_{k,i,2} - \alpha_{-k,i,2})(\sqrt{d} - 1)\boldsymbol{e}_i. \quad (3.34)
\end{aligned}$$

Let $\boldsymbol{x}$ be a point from the interior of $\mathcal{D} \cap \mathcal{C}$ expressed as a convex combination of the

vertices of $\mathcal{D}$

$$\boldsymbol{x} = \sum_i \alpha_{i,1}\sqrt{d}\boldsymbol{e}_i - \alpha_{i,2}\sqrt{d}\boldsymbol{e}_i = \sum_i (\alpha_{i,1} - \alpha_{i,2})\sqrt{d}\boldsymbol{e}_i, \quad |x_i| < 1, i = 1, \ldots, d,$$

where by Lemma 3.8 it can be assumed that

$$(\alpha_{i,1} + \alpha_{i,2})\sqrt{d} \leq 1.$$

This fact will be used later and the combination is convex

$$\sum_i \alpha_{i,1} + \alpha_{i,2} = 1, \quad \alpha_{i,1} \geq 0, \quad \alpha_{i,2} \geq 0, \quad i = 1, \ldots, d \tag{3.35}$$

and it will now be shown that $\boldsymbol{x}$ can be expressed as a convex combination of the vertices from $\mathcal{D}$ and with the vertices $\sqrt{d}\boldsymbol{e}_k$ and $-\sqrt{d}\boldsymbol{e}_k$ replaced by vertices from $\mathcal{P}_k$ and $\mathcal{P}_{-k}$, respectively. For this the combination (3.34) must replace the $(\alpha_{k,1} - \alpha_{k,2})\sqrt{d}\boldsymbol{e}_k$ in the convex combination (3.35), i.e.

$$\sum_{i \neq k} \alpha_{k,i,1} + \alpha_{k,i,2} - \alpha_{-k,i,1} - \alpha_{-k,i,2} = (\alpha_{k,1} - \alpha_{k,2})\sqrt{d}. \tag{3.36}$$

Let

$$\alpha_{k,i,1} = C_1\alpha_{i,1}, \quad \alpha_{k,i,2} = C_1\alpha_{i,2}, \quad C_1 = \frac{\alpha_{k,1}\sqrt{d}}{1 - (\alpha_{k,1} + \alpha_{k,2})} \geq 0$$

$$\alpha_{-k,i,1} = C_2\alpha_{i,1}, \quad \alpha_{-k,i,2} = C_2\alpha_{i,2}, \quad C_2 = \frac{\alpha_{k,2}\sqrt{d}}{1 - (\alpha_{k,1} + \alpha_{k,2})} \geq 0$$

and (3.36) is satisfied since

$$\begin{aligned}
&\sum_{i \neq k} (\alpha_{k,i,1} + \alpha_{k,i,2} - \alpha_{-k,i,1} - \alpha_{-k,i,2}) \\
&\quad = \sum_{i \neq k} \left(\frac{\alpha_{k,1}\sqrt{d}}{1 - (\alpha_{k,1} + \alpha_{k,2})}(\alpha_{i,1} + \alpha_{i,2}) - \frac{\alpha_{k,2}\sqrt{d}}{1 - (\alpha_{k,1} + \alpha_{k,2})}(\alpha_{i,1} + \alpha_{i,2}) \right) \\
&\quad = \frac{(\alpha_{k,1} - \alpha_{k,2})\sqrt{d}}{1 - (\alpha_{k,1} + \alpha_{k,2})} \sum_{i \neq k} (\alpha_{i,1} + \alpha_{i,2}) \\
&\quad = \frac{(\alpha_{k,1} - \alpha_{k,2})\sqrt{d}}{1 - (\alpha_{k,1} + \alpha_{k,2})} (1 - (\alpha_{k,1} + \alpha_{k,2})) \\
&\quad = (\alpha_{k,1} - \alpha_{k,2})\sqrt{d}.
\end{aligned}$$

Note also that

$$\begin{aligned}
&\alpha_{k,i,1} - \alpha_{k,i,2} + \alpha_{-k,i,1} - \alpha_{-k,i,2} \\
&\quad = \frac{\alpha_{k,1}\sqrt{d}}{1-(\alpha_{k,1}+\alpha_{k,2})}(\alpha_{i,1}-\alpha_{i,2}) + \frac{\alpha_{k,2}\sqrt{d}}{1-(\alpha_{k,1}+\alpha_{k,2})}(\alpha_{i,1}-\alpha_{i,2}) \\
&\quad = \frac{(\alpha_{k,1}+\alpha_{k,2})\sqrt{d}}{1-(\alpha_{k,1}+\alpha_{k,2})}(\alpha_{i,1}-\alpha_{i,2}).
\end{aligned}$$

Hence the linear combination of the vertices from $\mathcal{P}_k$ and $\mathcal{P}_{-k}$ equals

$$\begin{aligned}
&\sum_{i\neq k}\left(\alpha_{k,i,1}\boldsymbol{p}_{k,i,1} + \alpha_{k,i,2}\boldsymbol{p}_{k,i,2} + \alpha_{-k,i,1}\boldsymbol{p}_{-k,i,1} + \alpha_{-k,i,2}\boldsymbol{p}_{-k,i,2}\right) \\
&\quad = \sum_{i\neq k}(\alpha_{k,i,1} + \alpha_{k,i,2} - \alpha_{-k,i,1} - \alpha_{-k,i,2})\boldsymbol{e}_k \\
&\qquad + \sum_{i\neq k}(\alpha_{k,i,1} - \alpha_{k,i,2} + \alpha_{-k,i,1} - \alpha_{-k,i,2})(\sqrt{d}-1)\boldsymbol{e}_i \\
&\quad = (\alpha_{k,1}-\alpha_{k,2})\sqrt{d}\boldsymbol{e}_k + \sum_{i\neq k}\left(\frac{(\alpha_{k,1}+\alpha_{k,2})\sqrt{d}}{1-(\alpha_{k,1}+\alpha_{k,2})}(\alpha_{i,1}-\alpha_{i,2})\right)(\sqrt{d}-1)\boldsymbol{e}_i \qquad (3.37)
\end{aligned}$$

and $(\alpha_{k,1}+\alpha_{k,2})\sqrt{d} \leq 1$. Now set

$$\alpha'_{i,1} = C_3\alpha_{i,1}, \quad \alpha'_{k,i,2} = C_3\alpha_{i,2}, \quad C_3 = \frac{1-(\alpha_{k,1}+\alpha_{k,2})\sqrt{d}}{1-(\alpha_{k,1}+\alpha_{k,2})} \geq 0.$$

Moreover

$$\begin{aligned}
&(\alpha'_{i,1}-\alpha'_{i,2})\sqrt{d} + (\alpha_{k,i,1} - \alpha_{k,i,2} + \alpha_{-k,i,1} - \alpha_{-k,i,2})(\sqrt{d}-1) \\
&\quad = \frac{1-(\alpha_{k,i,1}+\alpha_{k,i,2})\sqrt{d}}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}(\alpha_{i,1}-\alpha_{i,2})\sqrt{d} + \frac{(\alpha_{k,i,1}+\alpha_{k,i,2})\sqrt{d}}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}(\alpha_{i,1}-\alpha_{i,2})(\sqrt{d}-1) \\
&\quad = (\alpha_{i,1}-\alpha_{i,2})\sqrt{d}\left[\frac{1-(\alpha_{k,i,1}+\alpha_{k,i,2})\sqrt{d}}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})} + \frac{(\alpha_{k,i,1}+\alpha_{k,i,2})}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}(\sqrt{d}-1)\right] \\
&\quad = (\alpha_{i,1}-\alpha_{i,2})\sqrt{d}\left[\frac{1-(\alpha_{k,i,1}+\alpha_{k,i,2})\sqrt{d}}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})} + \frac{(\alpha_{k,i,1}+\alpha_{k,i,2})\sqrt{d}}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})} - \frac{(\alpha_{k,i,1}+\alpha_{k,i,2})}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}\right] \\
&\quad = (\alpha_{i,1}-\alpha_{i,2})\sqrt{d}\left[\frac{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}\right] \\
&\quad = (\alpha_{i,1}-\alpha_{i,2})\sqrt{d}. \qquad (3.38)
\end{aligned}$$

By (3.37) and (3.38) the combination of the vertices $\sqrt{d}\boldsymbol{e}_i$ and $-\sqrt{d}\boldsymbol{e}_i$ with coefficients

$\alpha'_{i,1}$ and $\alpha'_{i,2}$, respectively, and the vertices from $\mathcal{P}_k$ and $\mathcal{P}_{-k}$ equal $\boldsymbol{x}$, i.e.

$$\begin{aligned}
&\sum_{i\neq k}(\alpha'_{i,1}-\alpha'_{i,2})\sqrt{d}\boldsymbol{e}_i+\sum_{i\neq k}\left(\alpha_{k,i,1}\boldsymbol{p}_{k,i,1}+\alpha_{k,i,2}\boldsymbol{p}_{k,i,2}+\alpha_{-k,i,1}\boldsymbol{p}_{-k,i,1}+\alpha_{-k,i,2}\boldsymbol{p}_{-k,i,2}\right)\\
&\quad=\sum_{i\neq k}(\alpha'_{i,1}-\alpha'_{i,2})\sqrt{d}\boldsymbol{e}_i+(\alpha_{k,1}-\alpha_{k,2})\sqrt{d}\boldsymbol{e}_k\\
&\qquad+\sum_{i\neq k}\left(\alpha_{k,i,1}-\alpha_{k,i,2}+\alpha_{-k,i,1}-\alpha_{-k,i,2}\right)(\sqrt{d}-1)\boldsymbol{e}_i\\
&\quad=\sum_{i\neq k}\left((\alpha'_{i,1}-\alpha'_{i,2})\sqrt{d}+(\alpha_{k,i,1}-\alpha_{k,i,2}+\alpha_{-k,i,1}-\alpha_{-k,i,2})(\sqrt{d}-1)\right)\boldsymbol{e}_i\\
&\qquad+(\alpha_{k,1}-\alpha_{k,2})\sqrt{d}\boldsymbol{e}_k\\
&\quad=\sum_{i\neq k}(\alpha_{i,1}-\alpha_{i,2})\sqrt{d}\boldsymbol{e}_i+(\alpha_{k,1}-\alpha_{k,2})\sqrt{d}\boldsymbol{e}_k\\
&\quad=\sum_{i}(\alpha_{i,1}-\alpha_{i,2})\sqrt{d}\boldsymbol{e}_i\\
&\quad=\boldsymbol{x}
\end{aligned}$$

and the linear combination is convex

$$\begin{aligned}
&\sum_{i\neq k}\left(\alpha'_{i,1}+\alpha'_{i,2}\right)+\sum_{i\neq k}\left(\alpha_{k,i,1}+\alpha_{k,i,2}+\alpha_{-k,i,1}+\alpha_{-k,i,2}\right)\\
&\quad=\frac{1-(\alpha_{k,i,1}+\alpha_{k,i,2})\sqrt{d}}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}\sum_{i\neq k}(\alpha_{i,1}+\alpha_{i,2})+\frac{\alpha_{k,i,1}\sqrt{d}}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}\sum_{i\neq k}(\alpha_{i,1}+\alpha_{i,2})\\
&\qquad+\frac{\alpha_{k,i,2}\sqrt{d}}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}\sum_{i\neq k}(\alpha_{i,1}+\alpha_{i,2})\\
&\quad=\left(\frac{1-(\alpha_{k,i,1}+\alpha_{k,i,2})\sqrt{d}}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}+\frac{(\alpha_{k,i,1}+\alpha_{k,i,2})\sqrt{d}}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}\right)\sum_{i\neq k}(\alpha_{i,1}+\alpha_{i,2})\\
&\quad=\frac{1}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}\sum_{i\neq k}(\alpha_{i,1}+\alpha_{i,2})\\
&\quad=\frac{1}{1-(\alpha_{k,i,1}+\alpha_{k,i,2})}\left(1-(\alpha_{k,i,1}+\alpha_{k,i,2})\right)\\
&\quad=1.
\end{aligned}$$

Hence it has been shown that if the vertices $\sqrt{d}\boldsymbol{e}_k$ and $-\sqrt{d}\boldsymbol{e}_k$ are replaced by the vertices $\mathcal{P}_k$ and $\mathcal{P}_{-k}$ respectively then the point $\boldsymbol{x}$ can be expressed as the convex combination of the new vertices. Repeating the procedure for $k=1,\ldots,d$ enables $\boldsymbol{x}$ to be written as the convex combination of vertices from $\mathcal{P}_k$, $\mathcal{P}_{-k}$, $k=1,\ldots,d$. □

An example of the *improved hyperdypiramid* in $\mathbb{R}^3$ is illustrated in Figure 3.14.

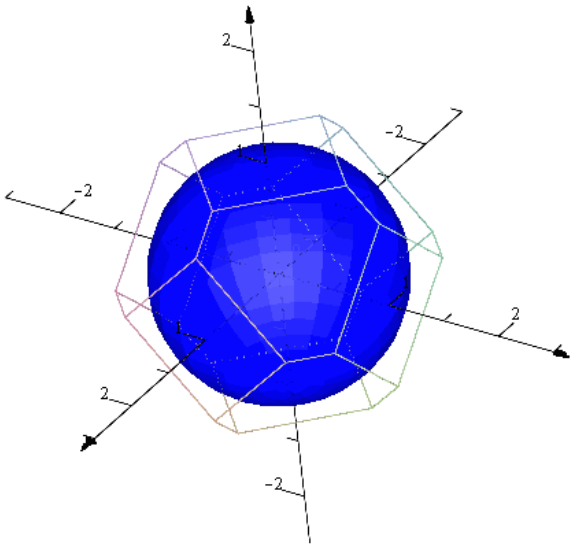

FIGURE 3.14: Chosen point in $\mathbb{R}^3$.The presented figure is a tetradecahedron (14-sided polyhedron).

Volume of $\mathcal{D}_{\text{imp}}$

The polytope $\mathcal{D}_{\text{imp}}$ is created from $\mathcal{D}$ by removing redundant $2d$ parts. Hence

$$\text{vol}(\mathcal{D}_{\text{imp}}) = \text{vol}(\mathcal{D}) - 2d \cdot \text{vol}(\mathcal{R}),$$

where $\text{vol}(\mathcal{R})$ denotes the volume of the redundant part. The redundant simplices are given by

$$\mathcal{R}_k = \text{Co}(\{\sqrt{d}\boldsymbol{e}_k\} \cup \mathcal{P}_k), \quad \mathcal{R}_{-k} = \text{Co}(\{-\sqrt{d}\boldsymbol{e}_k\} \cup \mathcal{P}_{-k}), \quad k = 1, \ldots, d$$

Consider $\mathcal{R}_k$ and translate the Cartesian basis by the vector $\boldsymbol{e}_k$. The new coordinates of the vertices in $\mathcal{P}_k$ are given as

$$\boldsymbol{p}'_{k,i,j} = (-1)^{j-1}(\sqrt{d}-1)\boldsymbol{e}_i \qquad j = 1, 2.$$

The redundant part forms a half of the hyperdypiramid in the new coordinates

$$\mathcal{R}_k = \text{Co}(\{(\sqrt{d}-1)\boldsymbol{e}_k\} \cup \{\pm(\sqrt{d}-1)\boldsymbol{e}_i : i = 1, \ldots, d, i \neq k\})$$

and since the hyperdypiramid is built of 2^d simplices

$$\text{vol}(\mathcal{R}_k) = \frac{1}{2}2^d \text{vol}(\Delta_{\sqrt{d}-1}).$$

Hence

$$\text{vol}(\mathcal{D}_{\text{imp}}) = 2^d \left(\text{vol}(\Delta_{\sqrt{d}}) - d \cdot \text{vol}(\Delta_{\sqrt{d}-1})\right)$$

and

$$0 \leq \lim_{d\to\infty} \text{vol}(\mathcal{D}_{imp}) \leq \lim_{d\to\infty} \text{vol}(\mathcal{D}) = 0.$$

All of the transformations used in the analysis here are linear and invertible. Hence it is routine to argue that the convex hull in $\mathbb{R}^d$ remains convex in $\mathbb{R}^{m\times n}$. The polytope obtained is used to produce the set of LMIs. If these are feasible then we accept them as a solution.

The final algorithm consists of the following steps:

1. Convert the matrices to vectors.
2. Translate the points about the center to obtain a subspace.
3. Form the orthonormal basis of the subspace spanned by the points and obtain the new coordinates.
4. Calculate the MVEE containing all the points in the subspace.
5. Transform the MVEE into a ball of unit radius.
6. Choose the vertices of a polytope.
7. By back transformation obtain the vertices in a matrix space.

Algorithm 3.3.6 summarizes this procedure.

Algorithm 3.3.6 Returns a polytope containing given set of matrices

function HYPERDIPYRAMID($[\boldsymbol{A}_s, \boldsymbol{B}_s], s \in \mathcal{S}$)
  for $k := 1, \ldots, N$ **do**
    $\boldsymbol{x}_k \leftarrow \text{vec}([\boldsymbol{A}_{s_k}, \boldsymbol{B}_{s_k}])$
  end for
  $\boldsymbol{c} \leftarrow \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i$
  $\boldsymbol{X}_c \leftarrow [(\boldsymbol{x}_1 - \boldsymbol{c}), \ldots, (\boldsymbol{x}_N - \boldsymbol{c})]$ ▷ Translation about the center of mass
  $\boldsymbol{B} \leftarrow [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ ▷ Calculate the orthonormal basis of span($\boldsymbol{X}_c$)
  $\boldsymbol{X}' \leftarrow \boldsymbol{B}^{\mathrm{T}}\boldsymbol{X}_c$
  $(\boldsymbol{P}, \boldsymbol{e}) \leftarrow$ ELLIPSOID($\boldsymbol{X}'$). $\boldsymbol{E} \leftarrow (\boldsymbol{P}\boldsymbol{P}^{\mathrm{T}})^{-1}$ ▷ Calculate approximation to MVEE($\boldsymbol{X}'$)
  $\boldsymbol{E} = \boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}$ ▷ Cholesky factorization
  $\boldsymbol{D} = [\sqrt{d}\boldsymbol{e}_1 + \boldsymbol{f}, -\sqrt{d}\boldsymbol{e}_1 + \boldsymbol{f}, \ldots, \sqrt{d}\boldsymbol{e}_d + \boldsymbol{f}, -\sqrt{d}\boldsymbol{e}_d + \boldsymbol{f}]$ ▷ Hyperdypiramid
  $\boldsymbol{D} \leftarrow \boldsymbol{H}^{-1}\boldsymbol{D} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{2d}]$ ▷ Back-transformation
  $\boldsymbol{D} \leftarrow [(\boldsymbol{v}_1 + c), \ldots, (\boldsymbol{v}_{2d} + c)] = [\boldsymbol{v}'_1, \ldots, \boldsymbol{v}'_{2d}]$
  $\boldsymbol{V}_i = \text{vec}^{-1}(\boldsymbol{v}'_i)$, $i = 1, \ldots, 2d$
  return $\{\boldsymbol{V}_1, \ldots, \boldsymbol{V}_{2d}\}$
end function

Numerical tests

A comparison of the volumes is given in the Table 3.4. The reference volume of the unit d-ball is given by

$$\text{vol}(\mathcal{B}_d(0,1)) = \frac{\pi^{d/2}}{\Gamma(d/2+1)},$$

where $\Gamma(\cdot)$ is the gamma function. A comparison of the number of vertices is given in the Table 3.5.

dim	vol($\mathcal{C}$)	vol($\mathcal{D}$)	vol($\mathcal{D}_{imp}$)	vol($\mathcal{B}_d(0,1)$)
2	4	4	3.314	3.142
3	8	6.928	5.359	4.189
4	16	10.67	8	4.935
5	32	14.91	11.06	5.264
6	64	19.2	14.25	5.168
7	128	23.05	17.23	4.725
8	256	26.01	19.66	4.059
9	512	27.77	21.27	3.299
10	1024	28.22	21.91	2.55
15	3.277e+04	16.58	13.76	0.3814
20	1.049e+06	4.413	3.854	0.02581
30	1.074e+09	0.05808	0.05397	2.192e-05
40	1.1e+12	0.0001482	0.0001421	3.605e-09
50	1.126e+15	1.103e-07	1.076e-07	1.73e-13

TABLE 3.4: Volume of the polytopes in comparison to that of the d-ball of unit radius.

dim	vert($\mathcal{C}$)	vert($\mathcal{D}$)	vert($\mathcal{D}_{imp}$)
2	4	4	8
3	8	6	24
4	16	8	48
5	32	10	80
6	64	12	120
7	128	14	168
8	256	16	224
9	512	18	288
10	1024	20	360
15	3.277e+04	30	840
20	1.049e+06	40	1520
30	1.074e+09	60	3480
40	1.1e+12	80	6240
50	1.126e+15	100	9800

TABLE 3.5: Number of vertices of the polytopes.

Even though the advantage of *Hyperdipyramid* over *Hyperdypiramid improved* is somewhat small this may still be crucial in some cases.

The main advantage of Algorithm 3.3.6 is that is fast and requires the computation of a much lower number of LMIs. If n is the state dimension of the system, the number

of vertices of the polytope is $2(n^2 - n)$. Consider a robust stabilization problem for an nth order system against all the synchronization errors in the common clock case. The controller should stabilize all the possible systems that arise due to the synchronization errors. One way to obtain the solution is to solve a set of LMIs for all the possible matrices. This method will be termed *direct computation*. The other way is to compute a bounding hyperdypiramid and to solve a set of LMIs for vertices of the polytope. Table 3.6 gives a comparison of the time needed to compute the solution by both methods. For the computation of MVEE the Khachiyan algorithm was used.

n	direct computation avg time (sec)	computation with new algorithm avg time(sec)
1	-	-
2	0.187	0.3
3	0.829	0.7
4	10.109	2.8
5	148.14	11.2
6	12000	101.8
7	—	6000

TABLE 3.6: Time of solving the robust control problem for given n-order system with synchronization errors.

Note that for $n = 6$ the method is over 100 times faster than direct computation and this advantage will increase for $n > 6$.

3.4 Relaxed LMI conditions

The common Lyapunov function approach to stability and stabilization is very conservative. We may relax the conditions by using a parameter dependent Lyapunov function approach de Oliveira et al. (1999). However, it is only possible in the *common clock* case when systems are time-invariant. For *different clock frequencies* the systems involved are time-varying and may be described as switching systems. It is known that switching between two stable systems may result in an unstable system. An example of this fact for continuous-time systems is given in Leith et al. (2003). Example 3.1 gives a discrete time counterpart.

Example 3.1. *Consider the autonomous switching system*

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_{\sigma(k)}\boldsymbol{x}(k), \quad \boldsymbol{A}_{\sigma(k)} \in \{\boldsymbol{A}_1, \boldsymbol{A}_2\}, \quad \boldsymbol{x}(k) \in \mathbb{R}^2,$$

with switching signal

$$\sigma(k) = \begin{cases} 1 & \text{if } k \text{ is even} \\ 2 & \text{otherwise} \end{cases}$$

and

$$\boldsymbol{A}_1 = \begin{bmatrix} -0.1 & 0.5 \\ -1.5 & -0.2 \end{bmatrix}, \quad \boldsymbol{A}_2 = \begin{bmatrix} -0.2 & -1.5 \\ 0.5 & -0.1 \end{bmatrix}.$$

The matrices are stable with spectral radius $\rho(\boldsymbol{A}_1) = \rho(\boldsymbol{A}_2) = 0.8775$. *However, the switching system is unstable since*

$$\rho(\boldsymbol{A}_1 \cdot \boldsymbol{A}_2) = \rho(\boldsymbol{A}_2 \cdot \boldsymbol{A}_1) = 2.28.$$

As Example 3.1 shows, the stability of each matrix does not guarantee the stability of the switched system. Parameter-dependent Lyapunov function conditions can only guarantee the existence of a Lyapunov function for each matrix separately. This is insufficient in the case of *different clock frequencies* and only the existence of a common Lyapunov function may provide stability. Hence the remainder of this section only considers the *common clock* case.

3.4.1 Stability

Consider discrete-time systems described by the state space model

$$\boldsymbol{x}(k+1) = \boldsymbol{A}(\alpha)\boldsymbol{x}(k), \tag{3.39}$$

where $\boldsymbol{A}(\alpha)$ is a member of a convex polytopic set

$$\mathcal{A} = \left\{ \boldsymbol{A}(\alpha) : \quad \boldsymbol{A}(\alpha) = \sum_{i=1}^{N} \alpha_i \boldsymbol{A}_i, \ \sum_{i=1}^{N} \alpha_i = 1, \ \alpha_i \leq 0 \right\}. \tag{3.40}$$

The following theorem de Oliveira et al. (1999) gives sufficient conditions for stability based on the parameter dependent Lyapunov function.

Theorem 3.10. *An uncertain system (3.39) is robustly stable in the uncertain domain (3.40) if there exist symmetric matrices* $\boldsymbol{P}_i$ *and a matrix* $\boldsymbol{G}$ *such that*

$$\begin{bmatrix} \boldsymbol{P}_i & \boldsymbol{A}_i^T \boldsymbol{G}^T \\ \boldsymbol{G}\boldsymbol{A}_i & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P}_i \end{bmatrix} \succ \boldsymbol{0} \tag{3.41}$$

for all $i = 1, \ldots, N$.

Consider the linear discrete-time system

$$\boldsymbol{x}(k+1) = \boldsymbol{A}(\alpha)\boldsymbol{x}(k) + \boldsymbol{B}(\beta)\boldsymbol{u}(k), \tag{3.42}$$

where $\boldsymbol{A}(\alpha) \in \mathcal{A}$ and $\boldsymbol{B}(\beta)$ is a member of the convex polytope defined by

$$\mathcal{B} = \left\{ \boldsymbol{B}(\beta) : \quad \boldsymbol{B}(\beta) = \sum_{i=1}^{M} \beta_i \boldsymbol{B}_i, \sum_{i=1}^{M} \beta_i = 1, \beta_i \geq 0 \right\}. \tag{3.43}$$

We search for state feedback matrix $\boldsymbol{K}$ such that

$$\boldsymbol{u}(k) = \boldsymbol{K}\boldsymbol{x}(k) \tag{3.44}$$

and $\boldsymbol{A}(\alpha) + \boldsymbol{B}(\beta)\boldsymbol{K}$ is asymptotically stable for all $\boldsymbol{A}(\alpha) \in \mathcal{A}$ and $\boldsymbol{B}(\beta) \in \mathcal{B}$.

Theorem 3.11. *de Oliveira et al. (1999). The uncertain system (3.42) is robustly stable in the uncertainty domains (3.40) and (3.43) if there exist symmetric matrices $\boldsymbol{P}_{ij}$ and a matrix G such that*

$$\begin{bmatrix} \boldsymbol{P}_{ij} & \boldsymbol{A}_i\boldsymbol{G} + \boldsymbol{B}_j\boldsymbol{L} \\ \boldsymbol{G}^T\boldsymbol{A}_i^T + \boldsymbol{L}^T\boldsymbol{B}_j^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P}_{ij} \end{bmatrix} \succ \boldsymbol{0} \tag{3.45}$$

for all $i = 1, \ldots, N$, $j = 1, \ldots, M$. If (3.45) is feasible then

$$\boldsymbol{K} = \boldsymbol{L}\boldsymbol{G}^{-1}. \tag{3.46}$$

Consider a system

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k). \tag{3.47}$$

Let $\mathcal{S}$ denote the set of all possible sequences describing the switching pattern for the system (3.47). First all matrices $\boldsymbol{A}_s, \boldsymbol{B}_s, s \in \mathcal{S}$ representing the system behaviour in case of synchronization errors are computed. Next the following tests are performed:

- **Test 1:** Compute a common polytope for the matrices $\boldsymbol{A}_s$ and $\boldsymbol{B}_s$ such that

$$\begin{bmatrix} \boldsymbol{A}_s & \boldsymbol{B}_s \end{bmatrix} \in \mathrm{Co}\left\{ \begin{bmatrix} \boldsymbol{A}_i & \boldsymbol{B}_i \end{bmatrix} : i = 1, \ldots, N \right\}, \tag{3.48}$$

  using developed Algorithm 3.3.6. A controller is evaluated using a common quadratic Lyapunov function approach and sufficient LMI conditions.

- **Test 2:** Common Lyapunov function conditions are used and a controller is evaluated using matrices $\boldsymbol{A}_s$ and $\boldsymbol{B}_s$ as vertices of the polytope.

- **Test 3:** Compute two different polytopes for the matrices $\boldsymbol{A}_s$ and $\boldsymbol{B}_s$ such that

$$\boldsymbol{A}_s \in \mathrm{Co}\left\{\boldsymbol{A}_i : i = 1, \ldots, N\right\},$$

$$\boldsymbol{B}_s \in \mathrm{Co}\left\{\boldsymbol{B}_j : j = 1, \ldots, M\right\}, \tag{3.49}$$

  using the developed Algorithm 3.3.6. Parameter-dependent Lyapunov function conditions are used in order to find a controller.

- **Test 4:** Parameter-dependent Lyapunov function conditions are used but with matrices $\boldsymbol{A}_s$ and $\boldsymbol{B}_s$ taken as vertices of the polytope.

All the methods were tested in Matlab with summary results in Table 3.7

n	Test 1	Test 2	Test 3	Test 4
2	0.07	0.02	0.72	0.45
3	0.18	0.24	89.54	80.13
4	0.58	1.70	$\star$	$\star$

TABLE 3.7: Average time of computation in seconds ($\star$-few hours).

Methods based on parameter-dependent Lyapunov function (tests 3 and 4) are significantly slower. The number of LMIs to be solved by common Lyapunov function approach methods (tests 1 and 2) equals the number of vertices of the polytope whereas the method based on parameter-dependent function needs to solve the number of LMIs which is a second power of the number of vertices. Moreover, methods based on a common function solve a set of LMIs for two variable matrices which makes the number of decision variables (the number of entries of matrices to find) constant. Conversely, in every LMI used by methods based on parameter-dependent function there is an extra variable matrix. In this case the number of decision variables is a linear function of the number of LMIs and is not constant as in the first case. All these factors influence the performance of algorithms.

The conditions based on the parameter-dependent function are also too conservative. If the controller matrix $\boldsymbol{K}$ exists then $\forall s_1, s_2 \in \mathcal{S}$ the matrix $\boldsymbol{A}_{s_1} + \boldsymbol{B}_{s_2}\boldsymbol{K}$ is stable. This is far too strong for what is required. Much weaker condition such that $\forall s \in \mathcal{S}$ $\boldsymbol{A}_s + \boldsymbol{B}_s\boldsymbol{K}$ is stable guarantees the robustness of a controller against all the synchronization errors. The conservativeness is reduced by the modification developed next.

3.4.2 Relaxed conditions with reduced conservativeness

Consider systems of the form

$$\boldsymbol{x}(k+1) = \boldsymbol{A}(\alpha)\boldsymbol{x}(k) + \boldsymbol{B}(\alpha)\boldsymbol{u}(k), \tag{3.50}$$

where the matrices $[\ \boldsymbol{A}(\alpha), \boldsymbol{B}(\alpha)\ ]$ belong to the convex polytope set

$$\mathcal{V} = \{\ [\ \boldsymbol{A}(\alpha),\ \boldsymbol{B}(\alpha)\ ]:$$
$$[\ \boldsymbol{A}(\alpha),\ \boldsymbol{B}(\alpha)\ ] = \sum_{i=1}^{N} \alpha_i [\ \boldsymbol{A}_i,\ \boldsymbol{B}_i\ ], \sum_{i=1}^{N} \alpha_i = 1, \alpha_i \geq 0 \Bigg\}. \tag{3.51}$$

Theorem 3.12. *The uncertain system (3.50) is robustly stable in the uncertainty domain (3.51) if there exist symmetric matrices* $\boldsymbol{P}_i$, $i = 1, \ldots, N$ *and a matrix* $\boldsymbol{G}$ *such that*

$$\begin{bmatrix} \boldsymbol{P}_i & \boldsymbol{A}_i\boldsymbol{G} + \boldsymbol{B}_i\boldsymbol{L} \\ \boldsymbol{G}^T\boldsymbol{A}_i^T + \boldsymbol{L}^T\boldsymbol{B}_i^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P}_i \end{bmatrix} \succ \boldsymbol{0} \tag{3.52}$$

for all $i = 1, \ldots, N$. *If (3.52) is feasible then*

$$\boldsymbol{K} = \boldsymbol{L}\boldsymbol{G}^{-1}. \tag{3.53}$$

Proof. *Substituting (3.53) into (3.52) gives*

$$\begin{bmatrix} \boldsymbol{P}_i & (\boldsymbol{A}_i + \boldsymbol{B}_i\boldsymbol{K})\boldsymbol{G} \\ \boldsymbol{G}^T(\boldsymbol{A}_i + \boldsymbol{B}_i\boldsymbol{K})^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P}_i \end{bmatrix} \succ \boldsymbol{0}, \tag{3.54}$$

which is a transposed version of the LMI of Theorem 3.10 and this proves the stability of $(\boldsymbol{A}_i + \boldsymbol{B}_i\boldsymbol{K})$ *(for full details see proof of Theorem 3 in de Oliveira et al. (1999)).*

If a control law (3.53) exists for a polytope bounding matrices $[\ \boldsymbol{A}_s, \boldsymbol{B}_s\ ]$ then it is guaranteed that $\forall s \in \mathcal{S}$ the matrix $\boldsymbol{A}_s + \boldsymbol{B}_s\boldsymbol{K}$ is stable.

The tests detailed above were also applied again with those under 3 and 4 modified to use these new conditions. Table 3.8 gives the results in summary form.

n	Test 1	Test 2	Test 3 modified	Test 4 modified
2	0.07	0.02	0.36	0.29
3	0.18	0.24	1.5	1.05
4	0.58	1.70	24.47	26.58
5	2.10	33.54	342.07	$\star$

Table 3.8: Average time of computation in seconds ($\star$ - few hours).

The method above is slower than those based on the common Lyapunov function because still the number of decision variables depends linearly on the number of LMIs. However, due to reduced conservativeness, this method can now give us a solution in more cases than the common function methods.

3.5 Disturbance attenuation control

3.5.1 Introduction

Consider the discrete-time system

$$\begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{w}(k) \\ \boldsymbol{y}(k) & = & \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{w}(k) \end{array}, \tag{3.55}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{w} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^p$. Let $\boldsymbol{T}(z)$ denote the transfer-function matrix of (3.55)

$$\boldsymbol{T}(z) = \boldsymbol{C}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D}. \tag{3.56}$$

Consider the continuous time signal $x(t)$ with Laplace transform $X(s)$

$$X(s) = \int_{0-}^{\infty} x(t)e^{-st}\mathrm{d}t,$$

where sampling the signal with period $T_s = 1$ gives

$$X(e^s) = \sum_{m=0}^{\infty} x(m)e^{-sm}.$$

Substituting $z = e^s$ we obtain one-sided (unilateral) $\mathcal{Z}$-transform of a discrete signal $x(m)$

$$X(z) = \sum_{m=0}^{\infty} x(m)z^{-m}.$$

Now consider for (3.55) the underlying continuous-time system with transfer-function matrix $\boldsymbol{T}(s)$ and frequency response matrix $\boldsymbol{T}(j\omega)$. Similar reasoning based on sampling with period $T_s = 1$ leads to the frequency response matrix $\boldsymbol{T}(e^{j\omega})$ with H_2 norm defined as

$$\|\boldsymbol{T}(z)\|_2 = \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{tr}\ \left[\boldsymbol{T}(e^{j\omega})^H \boldsymbol{T}(e^{j\omega})\right] \mathrm{d}\omega},$$

where $(\cdot)^H$ denotes the conjugate transpose. The H_2 norm measures the steady-state covariance (power) of the output response $\boldsymbol{y} = \boldsymbol{T}\boldsymbol{w}$ to unit white noise input $\boldsymbol{w}$

$$\|\boldsymbol{T}\|_2^2 = \lim_{t\to\infty} E[\boldsymbol{y}(t)^{\mathrm{T}}\boldsymbol{y}(t)], \quad E[\boldsymbol{w}(t)\boldsymbol{w}(\tau)^{\mathrm{T}}] = \delta(t-\tau)\boldsymbol{I}.$$

For a discrete system the H_∞ norm is given by

$$\|\boldsymbol{T}(z)\|_\infty = \max_{\omega\in[0,\pi]} \sigma_{\max}, \left(\boldsymbol{T}(e^{j\omega})\right)$$

where $\sigma_{\max}$ denotes the largest singular value. This norm measures the peak gain across all input/output channels.

The following results are proved in, for example, Oliveira et al. (2002).

Theorem 3.13. (H_2 norm): *Assuming* $\boldsymbol{D} = \boldsymbol{0}$, *the inequality* $\|\boldsymbol{T}(z)\|_2^2 < \gamma$ *holds if and only if there exists symmetric matrices* $\boldsymbol{P}$ *and* $\boldsymbol{W}$ *such that*

$$\text{trace } \boldsymbol{W} \leq \gamma, \qquad \begin{bmatrix} \boldsymbol{W} & \boldsymbol{CP} \\ \boldsymbol{PC}^T & \boldsymbol{P} \end{bmatrix} \succ 0, \tag{3.57}$$

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{AP} & \boldsymbol{B} \\ \boldsymbol{PA}^T & \boldsymbol{P} & \boldsymbol{0} \\ \boldsymbol{B}^T & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \succ 0. \tag{3.58}$$

Theorem 3.14. (H_∞ norm): *The inequality* $\|\boldsymbol{T}(z)\|_\infty^2 < \gamma$ *holds if, and only if, there exists a symmetric matrix* $\boldsymbol{P}$ *such that*

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{AP} & \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{PA}^T & \boldsymbol{P} & \boldsymbol{0} & \boldsymbol{PC}^T \\ \boldsymbol{B}^T & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{D}^T \\ \boldsymbol{0} & \boldsymbol{CP} & \boldsymbol{D} & \gamma \boldsymbol{I} \end{bmatrix} \succ 0. \tag{3.59}$$

Theorem 3.15. (Extended H_2 norm): *Assuming that* $\boldsymbol{D} = 0$, *the inequality* $\|\boldsymbol{T}(z)\|_2^2 < \gamma$ *holds if and only if there exists a matrix* $\boldsymbol{G}$ *and symmetric matrices* $\boldsymbol{P}$ *and* $\boldsymbol{W}$ *such that*

$$\text{trace } \boldsymbol{W} \leq \gamma, \qquad \begin{bmatrix} \boldsymbol{W} & \boldsymbol{CG} \\ \boldsymbol{GC}^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P} \end{bmatrix} \succ 0, \tag{3.60}$$

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{AG} & \boldsymbol{B} \\ \boldsymbol{GA}^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P} & \boldsymbol{0} \\ \boldsymbol{B}^T & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \succ 0. \tag{3.61}$$

Theorem 3.16. (Extended H_∞ norm): *The inequality* $\|\boldsymbol{T}(z)\|_\infty^2 < \gamma$ *holds if and only if there exist a matrix* $\boldsymbol{G}$ *and a symmetric matrix* $\boldsymbol{P}$ *such that*

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{AG} & \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{GA}^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P} & \boldsymbol{0} & \boldsymbol{G}^T\boldsymbol{C}^T \\ \boldsymbol{B}^T & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{D}^T \\ \boldsymbol{0} & \boldsymbol{CG} & \boldsymbol{D} & \gamma \boldsymbol{I} \end{bmatrix} \succ 0 \tag{3.62}$$

is feasible.

Consider a system described by (3.55) with uncertainty such that matrix

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \tag{3.63}$$

takes values in a polytope Φ defined as

$$\Phi = \{\boldsymbol{M}(\alpha) : \boldsymbol{M}(\alpha) = \sum_{i=1}^{N} \alpha_i \boldsymbol{M}_i, \alpha_i \geq 0, \sum_{i=1}^{N} \alpha_i = 1\}. \tag{3.64}$$

The following theorems are proved in, for example Oliveira et al. (2002) using the parameter-dependent Lyapunov function approach.

Theorem 3.17. (Extended guaranteed H_2 cost): *If there exist symmetric matrices $\boldsymbol{W}_i$, $\boldsymbol{P}_i$, $i = 1, \ldots, N$ and a matrix $\boldsymbol{G}$ such that*

$$\text{trace } \boldsymbol{W}_i \leq \gamma, \qquad \begin{bmatrix} \boldsymbol{W}_i & \boldsymbol{C}_i\boldsymbol{G} \\ \boldsymbol{G}\boldsymbol{C}_i^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P}_i \end{bmatrix} \succ 0 \tag{3.65}$$

$$\begin{bmatrix} \boldsymbol{P}_i & \boldsymbol{A}_i\boldsymbol{G} & \boldsymbol{B}_i \\ \boldsymbol{G}\boldsymbol{A}_i^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P}_i & \boldsymbol{0} \\ \boldsymbol{B}_i^T & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \succ 0 \tag{3.66}$$

holds for all $i = 1, \ldots, N$ where the matrices $\boldsymbol{A}_i, \boldsymbol{B}_i, \boldsymbol{C}_i$ and $\boldsymbol{D}_i$ define the vertices of the polytope $\boldsymbol{M}_i, i = 1, \ldots, N$ then the inequality $\|\boldsymbol{T}(z)\|_2^2 < \gamma$ holds for all matrices $\boldsymbol{M}$ in the domain Φ.

Theorem 3.18. (Extended guaranteed H_∞ cost): *If there exist symmetric matrices $\boldsymbol{P}_i$, $i = 1, \ldots, N$ and a matrix $\boldsymbol{G}$ such that*

$$\begin{bmatrix} \boldsymbol{P}_i & \boldsymbol{A}_i\boldsymbol{G} & \boldsymbol{B}_i & \boldsymbol{0} \\ \boldsymbol{G}\boldsymbol{A}_i^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P}_i & \boldsymbol{0} & \boldsymbol{G}^T\boldsymbol{C}_i^T \\ \boldsymbol{B}_i^T & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{D}_i^T \\ \boldsymbol{0} & \boldsymbol{C}_i\boldsymbol{G} & \boldsymbol{D}_i & \gamma\boldsymbol{I} \end{bmatrix} \succ 0 \tag{3.67}$$

hold for $i = 1, \ldots, N$ and where the matrices $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$ and $\boldsymbol{D}_i$ define the vertices of the polytope Φ, then the inequality $\|\boldsymbol{T}(z)\|_\infty^2 < \gamma$ hold for all matrices in the domain Φ.

For the common Lyapunov function approach we have the following theorems Oliveira et al. (2002)

Theorem 3.19. (Quadratic guaranteed H_2 cost): *If there exist symmetric matrices $\boldsymbol{P}$, $\boldsymbol{W}$ such that*

$$\text{trace } \boldsymbol{W} \leq \gamma, \qquad \begin{bmatrix} \boldsymbol{W} & \boldsymbol{C}_i\boldsymbol{P} \\ \boldsymbol{P}\boldsymbol{C}_i^T & \boldsymbol{P} \end{bmatrix} \succ 0 \tag{3.68}$$

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{A}_i\boldsymbol{P} & \boldsymbol{B}_i \\ \boldsymbol{P}\boldsymbol{A}_i^T & \boldsymbol{P} & \boldsymbol{0} \\ \boldsymbol{B}_i^T & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \succ 0, \qquad i = 1, \ldots, N, \tag{3.69}$$

where the matrices $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$ and $\boldsymbol{D}_i$ define the vertices of the polytope Φ, Theorem 3.17 also holds.

Theorem 3.20. (Quadratic guaranteed H_∞ cost): *If there exist a symmetric matrix* $\boldsymbol{P}$ *such that*

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{A}_i\boldsymbol{P} & \boldsymbol{B}_i & \boldsymbol{0} \\ \boldsymbol{P}\boldsymbol{A}_i^T & \boldsymbol{P} & \boldsymbol{0} & \boldsymbol{P}\boldsymbol{C}_i^T \\ \boldsymbol{B}_i^T & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{D}_i^T \\ \boldsymbol{0} & \boldsymbol{C}_i\boldsymbol{P} & \boldsymbol{D}_i & \gamma\boldsymbol{I} \end{bmatrix} \succ 0, \qquad i = 1, \ldots, N, \tag{3.70}$$

where the matrices $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$ *and* $\boldsymbol{D}_i$ *define the vertices of the polytope* Φ, *Theorem 3.18 also holds.*

Corollary 3.21. *From the Theorems 3.17 and 3.18 we may derive conditions for common Lyapunov function by assuming that* $\boldsymbol{P}_i = \boldsymbol{P}$, $i = 1, \ldots, N$.

3.5.2 Controller design

Consider the system (3.55) with a control input $\boldsymbol{u} \in \mathbb{R}^m$ and an exogenous input $\boldsymbol{w} \in \mathbb{R}^m$

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{B}\boldsymbol{w}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{aligned} \tag{3.71}$$

and assume that the matrices take values in a convex polyhedron Φ defined in (3.64), i.e.

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \in \Phi. \tag{3.72}$$

Using the state feedback law

$$\boldsymbol{u}(k) = \boldsymbol{K}\boldsymbol{x}(k), \tag{3.73}$$

(3.71) becomes

$$\begin{aligned} \boldsymbol{x}(k+1) &= (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{w}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{w}(k) \end{aligned}, \tag{3.74}$$

where $\boldsymbol{w} \in \mathbb{R}^m$ is the exogenous input.

Theorem 3.22. (Common Lyapunov function H_2 state feedback): *There exists a control law of the form (3.73) such that the inequality* $\|\boldsymbol{T}(z)\|_2^2 < \gamma$ *holds if and only if there exist symmetric matrices* $\boldsymbol{P}$ *and* $\boldsymbol{W}$ *and a matrix* $\boldsymbol{Q}$ *such that*

$$\text{trace } \boldsymbol{W} \leq \gamma, \qquad \begin{bmatrix} \boldsymbol{W} & \boldsymbol{C}_i\boldsymbol{P} \\ \boldsymbol{P}\boldsymbol{C}_i^T & \boldsymbol{P} \end{bmatrix} \succ 0, \tag{3.75}$$

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{A}_i\boldsymbol{P} + \boldsymbol{B}_i\boldsymbol{Q} & \boldsymbol{B}_i \\ \boldsymbol{P}\boldsymbol{A}_i^T + \boldsymbol{Q}^T\boldsymbol{B}_i^T & \boldsymbol{P} & \boldsymbol{0} \\ \boldsymbol{B}_i^T & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \succ 0 \tag{3.76}$$

for $i = 1, \ldots, N$. The control law matrix is given by

$$\boldsymbol{K} = \boldsymbol{Q}\boldsymbol{P}^{-1}. \tag{3.77}$$

PROOF: By substituting (3.77) into LMIs and using Theorem (3.19) we obtain the result for the system with closed-loop system matrix (3.74).

Theorem 3.23. (Common Lyapunov function H_∞ state feedback): *There exists a controller in the form (3.73) such that the inequality $\|\boldsymbol{T}(z)\|_\infty^2 < \gamma$ holds if and only if there exist symmetric matrix $\boldsymbol{P}$ and a matrix $\boldsymbol{Q}$ such that*

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{A}_i\boldsymbol{P} + \boldsymbol{B}_i\boldsymbol{Q} & \boldsymbol{B}_i & \boldsymbol{0} \\ \boldsymbol{P}\boldsymbol{A}_i^T + \boldsymbol{Q}^T\boldsymbol{B}_i^T & \boldsymbol{P} & \boldsymbol{0} & \boldsymbol{P}\boldsymbol{C}_i^T \\ \boldsymbol{B}_i^T & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{D}_i^T \\ \boldsymbol{0} & \boldsymbol{C}_i\boldsymbol{P} & \boldsymbol{D}_i & \gamma\boldsymbol{I} \end{bmatrix} \succ 0 \tag{3.78}$$

for $i = 1, \ldots, N$. The control law matrix is given by

$$\boldsymbol{K} = \boldsymbol{Q}\boldsymbol{P}^{-1}. \tag{3.79}$$

PROOF: By substituting (3.79) into LMIs and using Theorem (3.20) we obtain the result for the system with closed-loop system matrix (3.74).

Theorem 3.24. (Parameter-dependent Lyapunov function H_2 state feedback): *There exists a controller in the form (3.73) such that the inequality $\|\boldsymbol{T}(z)\|_2^2 < \gamma$ holds if and only if there exist symmetric matrices $\boldsymbol{P}_i$ and $\boldsymbol{W}_i$, $i = 1, \ldots, N$ and matrices $\boldsymbol{G}$ and $\boldsymbol{Q}$ such that*

$$\text{trace } \boldsymbol{W}_i \leq \gamma, \qquad \begin{bmatrix} \boldsymbol{W}_i & \boldsymbol{C}_i\boldsymbol{G} \\ \boldsymbol{G}\boldsymbol{C}_i^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P}_i \end{bmatrix} \succ 0 \tag{3.80}$$

$$\begin{bmatrix} \boldsymbol{P}_i & \boldsymbol{A}_i\boldsymbol{G} + \boldsymbol{B}_i\boldsymbol{Q} & \boldsymbol{B}_i \\ \boldsymbol{G}\boldsymbol{A}_i^T + \boldsymbol{Q}^T\boldsymbol{B}_i^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P}_i & \boldsymbol{0} \\ \boldsymbol{B}_i^T & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \succ 0 \tag{3.81}$$

for $i = 1, \ldots, N$. The control law matrix is given by

$$\boldsymbol{K} = \boldsymbol{Q}\boldsymbol{G}^{-1}. \tag{3.82}$$

PROOF: By substituting (3.82) into LMIs and using Theorem (3.17) we obtain the result for the system with closed loop system matrix (3.74).

Theorem 3.25. (Parameter-dependent Lyapunov function H_∞ state feedback): *There exists a controller of the form (3.73) such that the inequality $\|\boldsymbol{T}(z)\|_2^2 < \gamma$ holds if and*

only if there exist symmetric matrices $\boldsymbol{P}_i$, $i = 1, \ldots, N$ *and matrices* $\boldsymbol{G}$ *and* $\boldsymbol{Q}$ *such that*

$$\begin{bmatrix} \boldsymbol{P}_i & \boldsymbol{A}_i\boldsymbol{G} + \boldsymbol{B}_i\boldsymbol{Q} & \boldsymbol{B}_i & \boldsymbol{0} \\ \boldsymbol{G}\boldsymbol{A}_i^T + \boldsymbol{Q}^T\boldsymbol{B}_i^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P}_i & \boldsymbol{0} & \boldsymbol{G}^T\boldsymbol{C}_i^T \\ \boldsymbol{B}_i^T & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{D}_i^T \\ \boldsymbol{0} & \boldsymbol{C}_i\boldsymbol{G} & \boldsymbol{D}_i & \gamma\boldsymbol{I} \end{bmatrix} \succ 0 \tag{3.83}$$

for $i = 1, \ldots, N$. *The control law matrix is given by*

$$\boldsymbol{K} = \boldsymbol{Q}\boldsymbol{G}^{-1}. \tag{3.84}$$

PROOF: By substituting (3.84) into LMIs and using Theorem (3.18) we obtain the result for the system with closed loop system matrix (3.74).

The following results for the common Lyapunov function approach are obtained by setting $\boldsymbol{P}_i = \boldsymbol{P}$, $\boldsymbol{W}_i = \boldsymbol{W}$ $i = 1, \ldots, N$ in the last two theorems.

Theorem 3.26. (Common Lyapunov function Extended H_2 state feedback): *There exists a control law of the form (3.73) such that the inequality* $\|\boldsymbol{T}(z)\|_2^2 < \gamma$ *holds if and only if there exist symmetric matrices* $\boldsymbol{P}$ *and* $\boldsymbol{W}$, $i = 1, \ldots, N$ *and matrices* $\boldsymbol{G}$ *and* $\boldsymbol{Q}$ *such that*

$$\text{trace } \boldsymbol{W} \leq \gamma, \qquad \begin{bmatrix} \boldsymbol{W} & \boldsymbol{C}_i\boldsymbol{G} \\ \boldsymbol{G}\boldsymbol{C}_i^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P} \end{bmatrix} \succ 0 \tag{3.85}$$

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{A}_i\boldsymbol{G} + \boldsymbol{B}_i\boldsymbol{Q} & \boldsymbol{B}_i \\ \boldsymbol{G}\boldsymbol{A}_i^T + \boldsymbol{Q}^T\boldsymbol{B}_i^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P} & \boldsymbol{0} \\ \boldsymbol{B}_i^T & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \succ 0 \tag{3.86}$$

for $i = 1, \ldots, N$. *The control law matrix is given by*

$$\boldsymbol{K} = \boldsymbol{Q}\boldsymbol{G}^{-1}. \tag{3.87}$$

Theorem 3.27. (Common Lyapunov function Extended H_∞ state feedback): *There exists a controller of the form (3.73) such that the inequality* $\|\boldsymbol{T}(z)\|_2^2 < \gamma$ *holds if and only if there exist symmetric matrix* $\boldsymbol{P}$, $i = 1, \ldots, N$ *and matrices* $\boldsymbol{G}$ *and* $\boldsymbol{Q}$ *such that*

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{A}_i\boldsymbol{G} + \boldsymbol{B}_i\boldsymbol{Q} & \boldsymbol{B}_i & \boldsymbol{0} \\ \boldsymbol{G}\boldsymbol{A}_i^T + \boldsymbol{Q}^T\boldsymbol{B}_i^T & \boldsymbol{G} + \boldsymbol{G}^T - \boldsymbol{P} & \boldsymbol{0} & \boldsymbol{G}^T\boldsymbol{C}_i^T \\ \boldsymbol{B}_i^T & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{D}_i^T \\ \boldsymbol{0} & \boldsymbol{C}_i\boldsymbol{G} & \boldsymbol{D}_i & \gamma\boldsymbol{I} \end{bmatrix} \succ 0 \tag{3.88}$$

for $i = 1, \ldots, N$. *The control law matrix is given by*

$$\boldsymbol{K} = \boldsymbol{Q}\boldsymbol{G}^{-1}. \tag{3.89}$$

3.5.3 Numerical tests

For systems given by (3.55) we first compute the set Ω of all possible system matrices describing behaviour in the presence of synchronization errors

$$\Omega = \left\{ \begin{bmatrix} \boldsymbol{A}_s & \boldsymbol{B}_s \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} : s \in \mathcal{S} \right\}, \tag{3.90}$$

where $\mathcal{S}$ denotes the set of all possible sequences describing synchronization errors. We also compute a bounding polytope Φ containing the set Ω, i.e.

$$\Phi = \left\{ \sum_{i=1}^{N} \alpha_i \begin{bmatrix} \boldsymbol{A}_i & \boldsymbol{B}_i \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} : \sum_{i=1}^{N} \alpha_i = 1, \quad \alpha_i \geq 0, \quad i = 1, \ldots, N \right\}, \qquad \Omega \subseteq \Phi. \tag{3.91}$$

For timing tests we use following methods in order to find a stabilizing control law matrix

- **Direct -** Computations are performed using elements of Ω as vertices of a polytope.
- **Polytope -** Computations are performed using vertices of Φ.
- **CLF -** Extended common Lyapunov function conditions are used.
- **PDLF -** Parameter-dependent Lyapunov function conditions are used.

The tests were performed separately for H_2 and H_∞ objectives and Tables 3.9 and 3.10 give the results obtained in a summary form.

	CLF		PDLF	
n	Direct	Polytope	Direct	Polytope
2	0.17	0.19	0.29	0.25
3	0.99	1.16	6.8	5.7
4	12	4.0	$\star$	$\star$

TABLE 3.9: Timing results for H_2 objective in seconds. ($\star$ - over 1 hour).

	CLF		PDLF	
n	Direct	Polytope	Direct	Polytope
2	0.37	0.44	0.46	0.43
3	1.91	1.78	2.05	2.79
4	5.65	2.46	$\star$	$\star$

TABLE 3.10: Timing results for H_∞ objective in seconds ($\star$ - over 1 hour).

These tests lead to the conclusion that the parameter dependent Lyapunov function approach is suitable only for the common clock case and only for small dimensioned problems.

3.6 Conclusions

The stability and stabilization of linear systems with clock synchronization errors involves working with a very large number of matrices generated by the state-space models. This chapter has shown how this problem can be addressed by the development of results based on embedding the system model in the polytopic uncertainty structure used in robust control. In general, the resulting tools lead to computational speed at the cost of some volume redundancy. Supporting numerical tests establish that the overall method is robust and suitable for large scale problems.

Chapter 4

Stabilization using a norm bounded uncertainty setting

4.1 Introduction

This chapter considers the problem of finding the norm bounded uncertainty of the minimal volume enclosing a given set of matrices that is usually very large. The particular focus is on the time taken to do the computation but at the expense of some volume redundancy. As in the previous chapter, a new method based on computation of the MVEE is developed by treating the norm bounded uncertainty as an ellipsoid in a vector space. In comparison to existing methods Boyd et al. (1994) (p. 59) the polynomial time complexity is reduced by one order.

The new results developed in this chapter make extensive use of the Frobenius norm in the definition of the uncertainty. Uncertainty defined in this way as opposed to the induced Euclidean norm and the result is some volume redundancy that cannot be removed. In the robust control literature, there has been some work using the Frobenius norm. In Lee et al. (1996) quadratic stability conditions were derived for continuous time linear systems and the H_∞ control of discrete time linear systems was considered in Boukas and Shi (1998); You and Gao (2000), with Lo and Lin (2006) treating the same problem for the continuous time case. All of this previous work only used a simplified structure for the uncertainty and this limits its scope.

4.2 Norm bounded uncertainty analysis

The system with norm bounded uncertainty may be written in the most general form

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}_p\boldsymbol{p}(k) + \boldsymbol{B}_u\boldsymbol{u}(k) + \boldsymbol{B}_w\boldsymbol{w}(k), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{q}(k) &= \boldsymbol{C}_q\boldsymbol{x}(k) + \boldsymbol{D}_{qp}\boldsymbol{p}(k) + \boldsymbol{D}_{qu}\boldsymbol{u}(k) + \boldsymbol{D}_{qw}\boldsymbol{w}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}_y\boldsymbol{x}(k) + \boldsymbol{D}_{yp}\boldsymbol{p}(k) + \boldsymbol{D}_{yu}\boldsymbol{u}(k) + \boldsymbol{D}_{yw}\boldsymbol{w}(k) \\ \boldsymbol{p}(k) &= \boldsymbol{\Delta}(k)\boldsymbol{q}(k), \qquad \|\boldsymbol{\Delta}(k)\| \leq 1, \end{aligned} \tag{4.1}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u}(k) \in \mathbb{R}^m$ is the control input vector, $\boldsymbol{w}(k) \in \mathbb{R}^m$ is the exogenous input vector and $\boldsymbol{y}(k) \in \mathbb{R}^p$ is the output vector. The last condition in (4.1) is equivalent to

$$\boldsymbol{p}^{\mathrm{T}}(k)\boldsymbol{p}(k) \leq \boldsymbol{q}^{\mathrm{T}}(k)\boldsymbol{q}(k).$$

A block diagram representation of (4.1) is given in Figure 4.1.

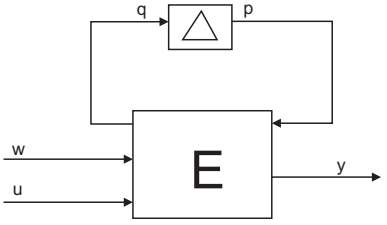

FIGURE 4.1: Block diagram representation of a system with norm bounded uncertainty.

For the purposes of the chapter, the following description suffices

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}_p\boldsymbol{p}(k) + \boldsymbol{B}_u\boldsymbol{u}(k), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{q}(k) &= \boldsymbol{C}_q\boldsymbol{x}(k) + \boldsymbol{D}_{qu}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \\ \boldsymbol{p}(k) &= \boldsymbol{\Delta}(k)\boldsymbol{q}(k), \qquad \|\boldsymbol{\Delta}(k)\| \leq 1, \end{aligned}$$

which may be rewritten as

$$\begin{aligned} \boldsymbol{x}(k+1) &= (\boldsymbol{A} + \boldsymbol{B}_p\boldsymbol{\Delta}(k)\boldsymbol{C}_q)\boldsymbol{x}(k) + (\boldsymbol{B}_u + \boldsymbol{B}_p\boldsymbol{\Delta}(k)\boldsymbol{D}_{qu})\boldsymbol{u}(k), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k), \qquad \|\boldsymbol{\Delta}(k)\| \leq 1. \end{aligned}$$

A more compact description is

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

where $\boldsymbol{x}(k) \in \mathbb{R}^n, \boldsymbol{u}(k) \in \mathbb{R}^m$ and $\begin{bmatrix}\bar{\boldsymbol{A}} & \bar{\boldsymbol{B}}\end{bmatrix} \in \Omega$. The set Ω is defined as

$$\Omega = \left\{[\boldsymbol{A}\ \boldsymbol{B}] + \boldsymbol{H}\boldsymbol{F}[\boldsymbol{E}1\ \boldsymbol{E}2] : \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} \preceq \boldsymbol{I}\right\},$$

or

$$[\bar{\boldsymbol{A}}\ \bar{\boldsymbol{B}}] = [\boldsymbol{A}\ \boldsymbol{B}] + \boldsymbol{H}\boldsymbol{F}[\boldsymbol{E}_1\ \boldsymbol{E}_2] = [\boldsymbol{A}\ \boldsymbol{B}] + \boldsymbol{H}\boldsymbol{F}\boldsymbol{E}$$

for some $\boldsymbol{F}$ such that $\boldsymbol{F}^T\boldsymbol{F} \preceq \boldsymbol{I}$.

Lemma 4.1. *Let $\boldsymbol{H}, \boldsymbol{E}$ be given real matrices of compatible dimension and suppose that $\boldsymbol{F}$ satisfies $\boldsymbol{F}^T\boldsymbol{F} \preceq \boldsymbol{I}$. Then for any $\epsilon > 0$*

$$\boldsymbol{H}\boldsymbol{F}\boldsymbol{E} + \boldsymbol{E}^T\boldsymbol{F}^T\boldsymbol{H}^T \preceq \epsilon\boldsymbol{H}\boldsymbol{H}^T + \epsilon^{-1}\boldsymbol{E}^T\boldsymbol{E} \tag{4.2}$$

Proof. Since

$$\left(\epsilon^{\frac{1}{2}}\boldsymbol{H}^{\mathrm{T}} - \epsilon^{-\frac{1}{2}}\boldsymbol{F}\boldsymbol{E}\right)^{\mathrm{T}}\left(\epsilon^{\frac{1}{2}}\boldsymbol{H}^{\mathrm{T}} - \epsilon^{-\frac{1}{2}}\boldsymbol{F}\boldsymbol{E}\right) \geq \boldsymbol{O}, \tag{4.3}$$

then by expansion

$$\epsilon^{-1}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{E} + \epsilon\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} \succeq \boldsymbol{H}\boldsymbol{F}\boldsymbol{E} + \boldsymbol{E}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}. \tag{4.4}$$

Also

$$\|\boldsymbol{F}\| \leq 1 \Leftrightarrow \lambda_{\max}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}) \leq 1 \Leftrightarrow \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} \leq I \tag{4.5}$$

and hence

$$\epsilon^{-1}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E} + \epsilon\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} \succeq \epsilon^{-1}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{E} + \epsilon\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} \succeq$$

$$\succeq \boldsymbol{H}\boldsymbol{F}\boldsymbol{E} + \boldsymbol{E}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}} \tag{4.6}$$

and proof is complete. □

Stability of a linear discrete-time system with norm bounded uncertainty

Consider the state space model

$$\boldsymbol{x}(k+1) = (\boldsymbol{A} + \Delta\boldsymbol{A})\boldsymbol{x}(k) + (\boldsymbol{B} + \Delta\boldsymbol{B})\boldsymbol{u}(k), \tag{4.7}$$

where

$$\begin{bmatrix} \Delta\boldsymbol{A} & \Delta\boldsymbol{B} \end{bmatrix} = \boldsymbol{H}\boldsymbol{F}\begin{bmatrix} \boldsymbol{E}_1 & \boldsymbol{E}_2 \end{bmatrix}, \qquad \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} \preceq \boldsymbol{I}. \tag{4.8}$$

Lemma 4.2. *The system (4.7) is stable if there exists a scalar $\epsilon > 0$ and a matrix $\boldsymbol{P} = \boldsymbol{P}^T \succ 0$ such that*

$$\begin{bmatrix} -\boldsymbol{P}^{-1} + \epsilon\boldsymbol{H}\boldsymbol{H}^T & \boldsymbol{A} \\ \boldsymbol{A}^T & \epsilon^{-1}\boldsymbol{E}^T\boldsymbol{E} - \boldsymbol{P} \end{bmatrix} \prec \boldsymbol{O}. \tag{4.9}$$

Proof. The system (4.7) is stable if there exists a symmetric positive definite matrix $\boldsymbol{P} = \boldsymbol{P}^{\mathrm{T}} \succ 0$ such that

$$(\boldsymbol{A} + \Delta\boldsymbol{A})^{\mathrm{T}}\boldsymbol{P}(\boldsymbol{A} + \Delta\boldsymbol{A}) - \boldsymbol{P} \prec \boldsymbol{O}, \tag{4.10}$$

or

$$(\boldsymbol{A}+\boldsymbol{H}\boldsymbol{F}\boldsymbol{E}_1)^{\mathrm{T}}\boldsymbol{P}(\boldsymbol{A}+\boldsymbol{H}\boldsymbol{F}\boldsymbol{E}_1)-\boldsymbol{P}\prec\boldsymbol{O}. \tag{4.11}$$

Applying the Schur's complement formula to this last expression gives

$$\begin{bmatrix} -\boldsymbol{P}^{-1} & \boldsymbol{A}+\boldsymbol{H}\boldsymbol{F}\boldsymbol{E}_1 \\ \boldsymbol{A}^{\mathrm{T}}+\boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}} & -\boldsymbol{P} \end{bmatrix}\prec 0 \tag{4.12}$$

and hence

$$\begin{bmatrix} -\boldsymbol{P}^{-1} & \boldsymbol{A}+\boldsymbol{H}\boldsymbol{F}\boldsymbol{E}_1 \\ \boldsymbol{A}^{\mathrm{T}}+\boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}} & -\boldsymbol{P} \end{bmatrix}=\begin{bmatrix} -\boldsymbol{P}^{-1} & \boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{T}} & -\boldsymbol{P} \end{bmatrix}$$

$$+\begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{O} \end{bmatrix}\boldsymbol{F}\begin{bmatrix} \boldsymbol{O} & \boldsymbol{E}_1 \end{bmatrix}+\begin{bmatrix} \boldsymbol{O} & \boldsymbol{E}_1 \end{bmatrix}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{O} \end{bmatrix}^{\mathrm{T}}\prec\boldsymbol{O}. \tag{4.13}$$

Applying Lemma 4.1 now gives

$$\begin{bmatrix} -\boldsymbol{P}^{-1} & \boldsymbol{A}+\boldsymbol{H}\boldsymbol{F}\boldsymbol{E}_1 \\ \boldsymbol{A}^{\mathrm{T}}+\boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}} & -\boldsymbol{P} \end{bmatrix}\prec\begin{bmatrix} -\boldsymbol{P}^{-1} & \boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{T}} & -\boldsymbol{P} \end{bmatrix}$$

$$+\epsilon\begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{O} \end{bmatrix}\begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{O} \end{bmatrix}^{\mathrm{T}}+\epsilon^{-1}\begin{bmatrix} \boldsymbol{O} & \boldsymbol{E}_1 \end{bmatrix}^{\mathrm{T}}\begin{bmatrix} \boldsymbol{O} & \boldsymbol{E}_1 \end{bmatrix}, \tag{4.14}$$

where $\epsilon>0$ is a scalar. Hence

$$\begin{bmatrix} -\boldsymbol{P}^{-1} & \boldsymbol{A}+\boldsymbol{H}\boldsymbol{F}\boldsymbol{E}_1 \\ \boldsymbol{A}^{\mathrm{T}}+\boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}} & -\boldsymbol{P} \end{bmatrix}$$

$$\prec\begin{bmatrix} -\boldsymbol{P}^{-1}+\epsilon\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} & \boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{T}} & \epsilon^{-1}\boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{E}_1-\boldsymbol{P} \end{bmatrix} \tag{4.15}$$

and if there exists a scalar $\epsilon>0$ and a matrix $P=P^T>0$ such that

$$\begin{bmatrix} -\boldsymbol{P}^{-1}+\epsilon\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} & \boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{T}} & \epsilon^{-1}\boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{E}_1-\boldsymbol{P} \end{bmatrix}\prec\boldsymbol{O}, \tag{4.16}$$

then (4.10) holds and the system (4.7) is stable. □

The following result gives sufficient conditions for stability of linear discrete-time systems described by (4.7) with norm bounded uncertainty in terms of an LMI.

Theorem 4.3. *The linear discrete-time system (4.7) with norm bounded uncertainty defined in (4.8) is stable if there exist matrix* $\bar{\boldsymbol{P}} = \bar{\boldsymbol{P}}^T \succ 0$ *such that*

$$\begin{bmatrix} -\bar{\boldsymbol{P}} & \bar{\boldsymbol{P}}\boldsymbol{A} & \bar{\boldsymbol{P}}\boldsymbol{H} & \boldsymbol{O} \\ \boldsymbol{A}^T\bar{\boldsymbol{P}} & -\bar{\boldsymbol{P}} & \boldsymbol{O} & \boldsymbol{E}_1^T \\ \boldsymbol{H}^T\bar{\boldsymbol{P}} & \boldsymbol{O} & -\boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{E}_1 & \boldsymbol{O} & -\boldsymbol{I} \end{bmatrix} \prec \boldsymbol{O}. \tag{4.17}$$

Proof. Consider the sufficient conditions given by (4.9). Pre and post-multiplying (4.9) by $\mathrm{diag}(\epsilon^{\frac{1}{2}}\boldsymbol{P}, \epsilon^{\frac{1}{2}}\boldsymbol{I})$ gives

$$\begin{bmatrix} \epsilon^{\frac{1}{2}}\boldsymbol{P} & \boldsymbol{O} \\ \boldsymbol{O} & \epsilon^{\frac{1}{2}}\boldsymbol{I} \end{bmatrix} \begin{bmatrix} -\boldsymbol{P}^{-1} + \epsilon\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} & \boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{T}} & \epsilon^{-1}\boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{E}_1 - \boldsymbol{P} \end{bmatrix}$$

$$\times \begin{bmatrix} \epsilon^{\frac{1}{2}}\boldsymbol{P} & \boldsymbol{O} \\ \boldsymbol{O} & \epsilon^{\frac{1}{2}}\boldsymbol{I} \end{bmatrix} = \begin{bmatrix} -\epsilon\boldsymbol{P} + \epsilon^2\boldsymbol{P}\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{P} & \epsilon\boldsymbol{P}\boldsymbol{A} \\ \epsilon\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P} & \boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{E}_1 - \epsilon\boldsymbol{P} \end{bmatrix}. \tag{4.18}$$

Now assume that $\bar{\boldsymbol{P}} = \epsilon\boldsymbol{P}$ and the sufficient condition of Lemma 4.2 is equivalent to

$$\begin{bmatrix} -\bar{\boldsymbol{P}} + \bar{\boldsymbol{P}}\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\bar{\boldsymbol{P}} & \bar{\boldsymbol{P}}\boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{T}}\bar{\boldsymbol{P}} & \boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{E}_1 - \bar{\boldsymbol{P}} \end{bmatrix} < 0, \tag{4.19}$$

also

$$\begin{bmatrix} -\bar{\boldsymbol{P}} + \bar{\boldsymbol{P}}\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\bar{\boldsymbol{P}} & \bar{\boldsymbol{P}}\boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{T}}\bar{\boldsymbol{P}} & \boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{E}_1 - \bar{\boldsymbol{P}} \end{bmatrix} = \begin{bmatrix} -\bar{\boldsymbol{P}} & \bar{\boldsymbol{P}}\boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{T}}\bar{\boldsymbol{P}} & -\bar{\boldsymbol{P}} \end{bmatrix}$$

$$+ \begin{bmatrix} \bar{\boldsymbol{P}}\boldsymbol{H} & 0 \\ 0 & \boldsymbol{E}_1^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{H}^{\mathrm{T}}\bar{\boldsymbol{P}} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{E}_1 \end{bmatrix} \tag{4.20}$$

and applying the Schur's complement formula gives

$$\begin{bmatrix} -\bar{\boldsymbol{P}} & \bar{\boldsymbol{P}}\boldsymbol{A} & \bar{\boldsymbol{P}}\boldsymbol{H} & \boldsymbol{O} \\ \boldsymbol{A}^{\mathrm{T}}\bar{\boldsymbol{P}} & -\bar{\boldsymbol{P}} & \boldsymbol{O} & \boldsymbol{E}_1^{\mathrm{T}} \\ \boldsymbol{H}^{\mathrm{T}}\bar{\boldsymbol{P}} & \boldsymbol{O} & -\boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{E}_1 & \boldsymbol{O} & -\boldsymbol{I} \end{bmatrix} \prec 0 \tag{4.21}$$

and the proof is complete. □

Stabilization of a system with norm bounded uncertainty

Consider again the system (4.7) with norm bounded uncertainty defined by (4.8) and apply the state feedback control law

$$\boldsymbol{u}(k) = \boldsymbol{K}\boldsymbol{x}(k) \tag{4.22}$$

to give

$$\boldsymbol{x}(k+1) = (\boldsymbol{A} + \Delta\boldsymbol{A} + (\boldsymbol{B} + \Delta\boldsymbol{B})\boldsymbol{K})\,\boldsymbol{x}(k), \tag{4.23}$$

where

$$[\Delta \boldsymbol{A} \quad \Delta \boldsymbol{B}] = \boldsymbol{H}\boldsymbol{F}[\boldsymbol{E}_1 \quad \boldsymbol{E}_2], \qquad \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} \preceq \boldsymbol{I}. \tag{4.24}$$

Theorem 4.4. *The system (4.23) with norm bounded uncertainty defined by (4.24) is stable under the state feedback control law (4.22) if there exist matrices* $\boldsymbol{Q} = \boldsymbol{Q}^T \succ \boldsymbol{O}$ *and* $\boldsymbol{R}$ *such that the following LMI is feasible*

$$\begin{bmatrix} -\boldsymbol{Q} & \boldsymbol{A}\boldsymbol{Q} + \boldsymbol{B}\boldsymbol{R} & \boldsymbol{H} & \boldsymbol{O} \\ \boldsymbol{Q}^T\boldsymbol{A}^T + \boldsymbol{R}^T\boldsymbol{B}^T & -\boldsymbol{Q} & \boldsymbol{O} & \boldsymbol{Q}^T\boldsymbol{E}_1^T + \boldsymbol{R}^T\boldsymbol{E}_2^T \\ \boldsymbol{H}^T & \boldsymbol{O} & -\boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{E}_1\boldsymbol{Q} + \boldsymbol{E}_2\boldsymbol{R} & \boldsymbol{O} & -\boldsymbol{I} \end{bmatrix} \prec 0. \tag{4.25}$$

The stabilizing state feedback control law matrix is given by $\boldsymbol{K} = \boldsymbol{R}\boldsymbol{Q}^{-1}$.

Proof. Using the definition of uncertainty the system can be written as

$$\boldsymbol{x}(k+1) = (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K} + \boldsymbol{H}\boldsymbol{F}[\ \boldsymbol{E}_1 + \boldsymbol{E}_2\boldsymbol{K}\ ])\,\boldsymbol{x}(k+1)), \tag{4.26}$$

or

$$\boldsymbol{x}(k+1) = (\boldsymbol{\Omega} + \boldsymbol{H}\boldsymbol{F}\boldsymbol{E})\boldsymbol{x}(k), \tag{4.27}$$

where $\boldsymbol{\Omega} = \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}$ and $\boldsymbol{E} = [\boldsymbol{E}_1 + \boldsymbol{E}_2\boldsymbol{K}]$. By Lemma 4.2 the system (4.27) is stable if there exist a scalar $\epsilon > 0$ and a matrix $\boldsymbol{P} = \boldsymbol{P}^{\mathrm{T}} \succ 0$ such that

$$\begin{bmatrix} -\boldsymbol{P}^{-1} + \epsilon\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} & \boldsymbol{\Omega} \\ \boldsymbol{\Omega}^{\mathrm{T}} & \epsilon^{-1}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E} - \boldsymbol{P} \end{bmatrix} \prec 0. \tag{4.28}$$

Pre and post-multiplying this last expression by diag($\epsilon^{-\frac{1}{2}}\boldsymbol{I}$,$\epsilon^{-\frac{1}{2}}\boldsymbol{P}^{-1}$) gives

$$\begin{bmatrix} \epsilon^{-\frac{1}{2}}\boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \epsilon^{-\frac{1}{2}}\boldsymbol{P}^{-1} \end{bmatrix} \begin{bmatrix} -\boldsymbol{P}^{-1} + \epsilon\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} & \boldsymbol{\Omega} \\ \boldsymbol{\Omega}^{\mathrm{T}} & \epsilon^{-1}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E} - \boldsymbol{P} \end{bmatrix}$$

$$\times \begin{bmatrix} \epsilon^{-\frac{1}{2}}\boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \epsilon^{-\frac{1}{2}}\boldsymbol{P}^{-1} \end{bmatrix} =$$

$$\begin{bmatrix} -\epsilon^{-1}\boldsymbol{P}^{-1} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} & \epsilon^{-1}\boldsymbol{\Omega}P^{-1} \\ \epsilon^{-1}\boldsymbol{P}^{-1}\boldsymbol{\Omega}^{\mathrm{T}} & \epsilon^{-2}\boldsymbol{P}^{-1}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}\boldsymbol{P}^{-1} - \epsilon^{-1}\boldsymbol{P}^{-1} \end{bmatrix} < 0 \tag{4.29}$$

and let $\boldsymbol{Q} = \boldsymbol{Q}^{\mathrm{T}} = \epsilon^{-1}\boldsymbol{P}^{-1}$ to obtain

$$\begin{bmatrix} -\boldsymbol{Q} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} & \Omega\boldsymbol{Q} \\ \boldsymbol{Q}^{\mathrm{T}}\Omega^{\mathrm{T}} & \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}\boldsymbol{Q} - \boldsymbol{Q} \end{bmatrix} < 0. \tag{4.30}$$

Also

$$\begin{bmatrix} -\boldsymbol{Q} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} & \boldsymbol{\Omega}\boldsymbol{Q} \\ \boldsymbol{Q}^{\mathrm{T}}\Omega^{\mathrm{T}} & \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}\boldsymbol{Q} - \boldsymbol{Q} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{Q} & \boldsymbol{\Omega}\boldsymbol{Q} \\ \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\Omega}^{\mathrm{T}} & -\boldsymbol{Q} \end{bmatrix}$$

$$+\begin{bmatrix} \boldsymbol{H} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{E}^{\mathrm{T}} \end{bmatrix}\begin{bmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix}\begin{bmatrix} \boldsymbol{H}^{T} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{EQ} \end{bmatrix} \prec \boldsymbol{O} \tag{4.31}$$

and applying the Schur's complement formula yields

$$\begin{bmatrix} -\boldsymbol{Q} & \boldsymbol{\Omega Q} & \boldsymbol{H} & \boldsymbol{O} \\ \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\Omega}^{\mathrm{T}} & -\boldsymbol{Q} & \boldsymbol{O} & \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{E}^{\mathrm{T}} \\ \boldsymbol{H}^{\mathrm{T}} & \boldsymbol{O} & -\boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{EQ} & \boldsymbol{O} & -\boldsymbol{I} \end{bmatrix} \prec \boldsymbol{O}. \tag{4.32}$$

Now let $\boldsymbol{KQ} = \boldsymbol{R}$ and expand terms containing $\boldsymbol{\Omega}$ and $\boldsymbol{E}$ to give

$$\begin{bmatrix} -\boldsymbol{Q} & \boldsymbol{AQ} + \boldsymbol{BR} & \boldsymbol{H} & \boldsymbol{O} \\ \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{R}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}} & -\boldsymbol{Q} & \boldsymbol{O} & \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{E}_1^{\mathrm{T}} + \boldsymbol{R}^{\mathrm{T}}\boldsymbol{E}_2^{T} \\ \boldsymbol{H}^{\mathrm{T}} & \boldsymbol{O} & -\boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{E}_1\boldsymbol{Q} + \boldsymbol{E}_2\boldsymbol{R} & \boldsymbol{O} & -\boldsymbol{I} \end{bmatrix} \prec 0. \tag{4.33}$$

If there exist matrices $\boldsymbol{Q} = \boldsymbol{Q}^{\mathrm{T}} \succ \boldsymbol{O}$ and $\boldsymbol{R}$ such that the above LMI is feasible then the system (4.23) is stabilizable by the state feedback control law with $\boldsymbol{K} = \boldsymbol{RQ}^{-1}$ and the proof is complete. □

4.3 Estimation of the norm bounded uncertainty

For systems with synchronization errors the problem is how to estimate the norm bounded uncertainty for a given set of matrices $\{\boldsymbol{A}_k, \boldsymbol{B}_k\}_{k=1}^{N}$. Moreover, the best case would be to compute the norm bounded uncertainty that is minimal in an appropriate sense. A first definition of the problem is

$$\begin{aligned} &\text{minimize}_{\text{over}\boldsymbol{A},\boldsymbol{B},\boldsymbol{H},\boldsymbol{E}} \qquad \mu(\{\boldsymbol{HFE} : \boldsymbol{F}^{T}\boldsymbol{F} \preceq \boldsymbol{I}\}) \\ &\text{subject to} \qquad [\boldsymbol{A}_k \ \boldsymbol{B}_k] = [\boldsymbol{A} \ \boldsymbol{B}] + \boldsymbol{HF}_k\boldsymbol{E}, \quad \boldsymbol{F}_k^{T}\boldsymbol{F}_k \preceq \boldsymbol{I}, \quad k = 1, \ldots, N. \end{aligned} \tag{4.34}$$

In the remainder of this chapter, a new method is developed and compared to the existing one. The conclusion being that this new method is more efficient in terms of time complexity. The problem considered in this chapter is very important since stability in the norm bounded uncertainty setting is only defined in terms of the matrices $\boldsymbol{A}$, $\boldsymbol{H}$ and $\boldsymbol{E}$ of the state space description (4.1). Hence only one LMI condition needs to be checked but to progress it is necessary to define in what sense is the uncertainty minimal.

4.3.1 The measure of norm bounded uncertainty

An obvious measure, denoted by μ in this thesis, is the volume of the set in the matrix space. Assume that the matrices involved lie on a hyperplane in the matrix space. Then

the volume is zero for every corresponding uncertainty and hence the problem is not well defined. To remove this problem, several measures exist and the most attractive of them is that in Boyd et al. (1994) (p. 59) where the diameter of the set Ω in the matrix space is defined as

$$d_\Omega = \max \{\|\boldsymbol{F} - \boldsymbol{G}\| : \boldsymbol{F}, \boldsymbol{G} \in \Omega\}.$$

In case of norm bounded uncertainty, the diameter may be expressed as

$$d_\Omega = \max \{\|\boldsymbol{H}\boldsymbol{F}_1\boldsymbol{E} - \boldsymbol{H}\boldsymbol{F}_2\boldsymbol{E}\| : \|\boldsymbol{F}_1\| \leq 1, \|\boldsymbol{F}_2\| \leq 1\}$$

and is equal to

$$2\sqrt{\lambda_{\max}(\boldsymbol{H}\boldsymbol{H}^T)\lambda_{\max}(\boldsymbol{E}^T\boldsymbol{E})}.$$

The following result gives an upper bound on the diameter.

Lemma 4.5. *The diameter of the norm bounded uncertainty d_Ω satisfies*

$$d_\Omega \leq 2\sqrt{\lambda_{max}(\boldsymbol{H}\boldsymbol{H}^T)\lambda_{max}(\boldsymbol{E}^T\boldsymbol{E})}.$$

Proof. By definition of the induced Euclidean norm and since $\lambda_{\max}(\boldsymbol{H}^T\boldsymbol{H}) = \lambda_{\max}(\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})$

$$\|\boldsymbol{H}\| \cdot \|\boldsymbol{E}\| = \sqrt{\lambda_{\max}(\boldsymbol{H}^T\boldsymbol{H})\lambda_{\max}(\boldsymbol{E}^T\boldsymbol{E})} = \sqrt{\lambda_{\max}(\boldsymbol{H}\boldsymbol{H}^T)\lambda_{\max}(\boldsymbol{E}^T\boldsymbol{E})}$$

and by the triangle inequality

$$\|\boldsymbol{H}\boldsymbol{F}_1\boldsymbol{E} - \boldsymbol{H}\boldsymbol{F}_2\boldsymbol{E}\| \leq \|\boldsymbol{H}\boldsymbol{F}_1\boldsymbol{E}\| + \|\boldsymbol{H}\boldsymbol{F}_1\boldsymbol{E}\|.$$

Hence exploiting submultiplicativity

$$\begin{aligned}\|\boldsymbol{H}\boldsymbol{F}_1\boldsymbol{E}\| &\leq \|\boldsymbol{H}\|\|\boldsymbol{F}_1\|\|\boldsymbol{E}\| = \|\boldsymbol{H}\|\|\boldsymbol{E}\| \\ \|\boldsymbol{H}\boldsymbol{F}_2\boldsymbol{E}\| &\leq \|\boldsymbol{H}\|\|\boldsymbol{F}_2\|\|\boldsymbol{E}\| = \|\boldsymbol{H}\|\|\boldsymbol{E}\|\end{aligned}$$

and then

$$\|\boldsymbol{H}\boldsymbol{F}_1\boldsymbol{E} - \boldsymbol{H}\boldsymbol{F}_2\boldsymbol{E}\| \leq 2\|\boldsymbol{H}\|\|\boldsymbol{E}\| = 2\sqrt{\lambda_{\max}(\boldsymbol{H}\boldsymbol{H}^T)\lambda_{\max}(\boldsymbol{E}^T\boldsymbol{E})}.$$

□

The following result establishes equality under some restrictions.

Theorem 4.6. *The diameter d_Ω of the norm bounded uncertainty is equal to*

$$2\sqrt{\lambda_{max}(\boldsymbol{H}\boldsymbol{H}^T)\lambda_{max}(\boldsymbol{E}^T\boldsymbol{E})}.$$

PROOF. It is required to prove that

$$d_\Omega = \max\{\|\boldsymbol{H}\boldsymbol{F}_1\boldsymbol{E} - \boldsymbol{H}\boldsymbol{F}_2\boldsymbol{E}\| : \|\boldsymbol{F}_1\| \leq 1, \|\boldsymbol{F}_2\| \leq 1\} \geq 2\|\boldsymbol{H}\|\|\boldsymbol{E}\|. \tag{4.35}$$

Assume that $\boldsymbol{F}_2 = -\boldsymbol{F}_1 = \bar{\boldsymbol{F}}$ and $\bar{\boldsymbol{F}}$ is orthogonal then

$$\|\boldsymbol{H}\boldsymbol{F}_1\boldsymbol{E} - \boldsymbol{H}\boldsymbol{F}_2\boldsymbol{E}\| = 2\|\boldsymbol{H}\bar{\boldsymbol{F}}\boldsymbol{E}\|.$$

Obviously

$$d_\Omega = \max\{\|\boldsymbol{H}\boldsymbol{F}_1\boldsymbol{E} - \boldsymbol{H}\boldsymbol{F}_2\boldsymbol{E}\| : \|\boldsymbol{F}_1\| \leq 1, \|\boldsymbol{F}_2\| \leq 1\} \geq 2\|\boldsymbol{H}\bar{\boldsymbol{F}}\boldsymbol{E}\| \tag{4.36}$$

Next it is shown that there exist an orthogonal matrix $\boldsymbol{F}^\star$ such that

$$\|\boldsymbol{H}\boldsymbol{F}^\star\boldsymbol{E}\| \geq \|\boldsymbol{H}\|\|\boldsymbol{E}\|.$$

First note that for matrices $\boldsymbol{A}, \boldsymbol{B}$ and $\ker \boldsymbol{B} = \{\emptyset\}$

$$\|\boldsymbol{A}\boldsymbol{B}\| = \max_{\|\boldsymbol{x}\|\neq 0} \frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \max_{\|\boldsymbol{x}\|=1} \frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}'\|}{\|\boldsymbol{x}'\|}$$

for some $\boldsymbol{x}'$. Hence if $\boldsymbol{x}'' = \boldsymbol{x}'/\|\boldsymbol{B}\boldsymbol{x}'\|$ then

$$\frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}''\|}{\|\boldsymbol{x}''\|} = \frac{\|\boldsymbol{A}\boldsymbol{B}\frac{\boldsymbol{x}'}{\|\boldsymbol{B}\boldsymbol{x}'\|}\|}{\|\frac{\boldsymbol{x}'}{\|\boldsymbol{B}\boldsymbol{x}'\|}\|} = \frac{\frac{1}{\|\boldsymbol{B}\boldsymbol{x}'\|}\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}'\|}{\frac{1}{\|\boldsymbol{B}\boldsymbol{x}'\|}\|\boldsymbol{x}'\|} = \frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}'\|}{\|\boldsymbol{x}'\|}.$$

In fact, it is possible to multiply $\boldsymbol{x}'$ by every full rank matrix and since

$$\|\boldsymbol{B}\boldsymbol{x}''\| = \left\|\frac{\boldsymbol{B}\boldsymbol{x}'}{\|\boldsymbol{B}\boldsymbol{x}'\|}\right\| = 1,$$

therefore

$$\frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}''\|}{\|\boldsymbol{x}''\|} = \max_{\|\boldsymbol{x}\|\neq 0} \frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \max_{\|\boldsymbol{B}\boldsymbol{x}\|=1} \frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}.$$

Consequently

$$\|\boldsymbol{A}\boldsymbol{B}\| = \max_{\|\boldsymbol{x}\|=1} \frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}'\|}{\|\boldsymbol{x}'\|} = \frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}''\|}{\|\boldsymbol{x}''\|} = \max_{\|\boldsymbol{B}\boldsymbol{x}\|=1} \frac{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}. \tag{4.37}$$

Next assuming that $\boldsymbol{E}$ is a full rank matrix

$$\|\boldsymbol{H}\| = \max_{\|\boldsymbol{x}\|=1} \frac{\|\boldsymbol{H}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \max_{\|\boldsymbol{E}\boldsymbol{x}\|=1} \frac{\|\boldsymbol{H}\boldsymbol{E}\boldsymbol{x}\|}{\|\boldsymbol{E}\boldsymbol{x}\|}. \tag{4.38}$$

Combining (4.37) with the fact that for any orthogonal matrix $\boldsymbol{F}$, $\|\boldsymbol{E}\| = \|\boldsymbol{F}\boldsymbol{E}\|$, gives

$$\|\boldsymbol{E}\| = \|\boldsymbol{F}\boldsymbol{E}\| = \max_{\|\boldsymbol{x}\|=1} \frac{\|\boldsymbol{F}\boldsymbol{E}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \max_{\|\boldsymbol{E}\boldsymbol{x}\|=1} \frac{\|\boldsymbol{F}\boldsymbol{E}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}. \tag{4.39}$$

Assume the maximums (4.38) and (4.39) hold for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively, i.e. for some $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and any orthogonal matrix $\boldsymbol{F}$

$$\|\boldsymbol{H}\| = \frac{\|\boldsymbol{H}\boldsymbol{E}\boldsymbol{x}_1\|}{\|\boldsymbol{E}\boldsymbol{x}_1\|}, \|\boldsymbol{E}\boldsymbol{x}_1\| = 1, \qquad \|\boldsymbol{E}\| = \frac{\|\boldsymbol{F}\boldsymbol{E}\boldsymbol{x}_2\|}{\|\boldsymbol{x}_2\|}, \|\boldsymbol{E}\boldsymbol{x}_2\| = 1. \tag{4.40}$$

Now consider the norm

$$\begin{aligned} \|\boldsymbol{H}\bar{\boldsymbol{F}}\boldsymbol{E}\| &= \max_{\|x\|=1} \frac{\|\boldsymbol{H}\bar{\boldsymbol{F}}\boldsymbol{E}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \max_{\|\boldsymbol{x}\|=1} \frac{\|\boldsymbol{H}\bar{\boldsymbol{F}}\boldsymbol{E}\boldsymbol{x}\|}{\|\bar{\boldsymbol{F}}\boldsymbol{E}\boldsymbol{x}\|} \frac{\|\bar{\boldsymbol{F}}\boldsymbol{E}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \\ &= \max_{\|\boldsymbol{E}\boldsymbol{x}\|=1} \frac{\|\boldsymbol{H}\bar{\boldsymbol{F}}\boldsymbol{E}\boldsymbol{x}\|}{\|\bar{\boldsymbol{F}}\boldsymbol{E}\boldsymbol{x}\|} \frac{\|\bar{\boldsymbol{F}}\boldsymbol{E}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}. \end{aligned} \tag{4.41}$$

Since $\|\boldsymbol{E}\boldsymbol{x}_1\| = \|\boldsymbol{E}\boldsymbol{x}_2\| = 1$ there exists an orthogonal matrix $\boldsymbol{F}^\star$ such that

$$\boldsymbol{F}^\star\boldsymbol{E}\boldsymbol{x}_2 = \boldsymbol{E}\boldsymbol{x}_1 \tag{4.42}$$

and hence

$$\frac{\|\boldsymbol{H}\boldsymbol{F}^\star\boldsymbol{E}\boldsymbol{x}_2\|}{\|\boldsymbol{F}^\star\boldsymbol{E}\boldsymbol{x}_2\|} \cdot \frac{\|\boldsymbol{F}^\star\boldsymbol{E}\boldsymbol{x}_2\|}{\|\boldsymbol{x}_2\|} = \frac{\|\boldsymbol{H}\boldsymbol{E}\boldsymbol{x}_1\|}{\|\boldsymbol{E}\boldsymbol{x}_1\|} \cdot \frac{\|\boldsymbol{F}^\star\boldsymbol{E}\boldsymbol{x}_2\|}{\|\boldsymbol{x}_2\|} = \|\boldsymbol{H}\|\|\boldsymbol{E}\|. \tag{4.43}$$

Taking initially $\bar{\boldsymbol{F}} = \boldsymbol{F}^\star$ and combining with (4.41) gives

$$\|\boldsymbol{H}\boldsymbol{F}^\star\boldsymbol{E}\| = \max_{\|\boldsymbol{E}\boldsymbol{x}\|=1} \frac{\|\boldsymbol{H}\boldsymbol{F}^\star\boldsymbol{E}\boldsymbol{x}\|}{\|\boldsymbol{F}^\star\boldsymbol{E}\boldsymbol{x}\|} \frac{\|\boldsymbol{F}^\star\boldsymbol{E}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \geq \frac{\|\boldsymbol{H}\boldsymbol{F}^\star\boldsymbol{E}\boldsymbol{x}_2\|}{\|\boldsymbol{F}^\star\boldsymbol{E}\boldsymbol{x}_2\|} \cdot \frac{\|\boldsymbol{F}^\star\boldsymbol{E}\boldsymbol{x}_2\|}{\|\boldsymbol{x}_2\|} = \|\boldsymbol{H}\|\|\boldsymbol{E}\|. \tag{4.44}$$

Combining (4.36) and (4.44) gives

$$d_\Omega \geq 2\|\boldsymbol{H}\boldsymbol{F}^\star\boldsymbol{E}\| \geq 2\|\boldsymbol{H}\|\|\boldsymbol{E}\|$$

and using Lemma 4.5, the diameter satisfies

$$d_\Omega = 2\|\boldsymbol{H}\|\|\boldsymbol{E}\| = 2\sqrt{\lambda_{\max}(\boldsymbol{H}\boldsymbol{H}^T)\lambda_{\max}(\boldsymbol{E}^T\boldsymbol{E})}.$$

□

Although the diameter is a valid measure, some issue may arise. In particular there may exist two uncertainties with the same measure where one is a subset of the other. This

also makes the minimum non-unique. Next, a more natural measure of the uncertainty is proposed starting from the fact that for every matrix $\boldsymbol{F}$

$$\boldsymbol{F}^T\boldsymbol{F} \preceq \boldsymbol{I} \quad \Longleftrightarrow \quad \|\boldsymbol{F}\|_2 \leq 1. \tag{4.45}$$

Define the unit uncertainty in a vector space as

$$\Gamma_d = \mathrm{vec}(\{\boldsymbol{F} \in \mathbb{R}^{n\times(n+m)} : \|\boldsymbol{F}\|_2 \leq 1\}) = \{\boldsymbol{f} \in \mathbb{R}^d : \|\mathrm{vec}^{-1}(\boldsymbol{f})\|_2 \leq 1\}, \quad d = n(n+m),$$

where the operation $\mathrm{vec}(\cdot)$ is defined in (3.7) and (3.8). The volume is given by

$$\mathrm{vol}(\{\boldsymbol{F} \in \mathbb{R}^{n\times(n+m)} : \|\boldsymbol{F}\|_2 \leq 1\}) = \mathrm{vol}(\Gamma_d) = \int_{\Gamma_d} 1 \cdot \mathrm{d}f_1 \cdots \mathrm{d}f_d = \gamma_d.$$

Consider the uncertainty $\Omega = \{\boldsymbol{H}\boldsymbol{F}\boldsymbol{E} : \|\boldsymbol{F}\|_2 \leq 1\}$, where the image of Ω under the $\mathrm{vec}(\cdot)$ operation is given by

$$\mathrm{vec}(\Omega) = \{\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H} \cdot \boldsymbol{f} : \|\mathrm{vec}^{-1}(\boldsymbol{f})\|_2 \leq 1\}$$

and $\otimes$ denotes the Kronecker product of the matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{p\times q}$, i.e.

$$\boldsymbol{A} \otimes \boldsymbol{B} := \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{bmatrix} \in \mathbb{R}^{mp\times nq}.$$

Also the volume is given by

$$\mathrm{vol}(\Omega) = \int_{\mathrm{vec}(\Omega)} 1 \cdot \mathrm{d}f_1 \cdots \mathrm{d}f_d. \tag{4.46}$$

The linear operation $\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}$ maps the unit uncertainty Γ_d into $\mathrm{vec}(\Omega)$ and the variables in the integral (4.46) can be changed using

$$\begin{bmatrix} f_1' \\ \vdots \\ f_d' \end{bmatrix} = (\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H})^{-1} \cdot \begin{bmatrix} f_1 \\ \vdots \\ f_d \end{bmatrix}, \qquad \left|\frac{\partial(f_1,\dots,f_d)}{\partial(f_1',\dots,f_d')}\right| = |\det \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}|.$$

The set $\mathrm{vec}(\Omega)$ is simply Γ_d in the coordinates $f_1', \dots, f_d'$ and hence the volume of Ω is given by

$$\mathrm{vol}(\Omega) = \int_{\mathrm{vec}(\Omega)} 1 \cdot \mathrm{d}f_1 \cdots \mathrm{d}f_d = \int_{\Gamma_d} |\det \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}| \cdot \mathrm{d}f_1' \cdots \mathrm{d}f_d' = |\det \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}| \cdot \gamma_d. \tag{4.47}$$

Since

$$\{\boldsymbol{H}\boldsymbol{F}\boldsymbol{E} : \|\boldsymbol{F}\|_2 \leq 1\} = \{-\boldsymbol{H}\boldsymbol{F}\boldsymbol{E} : \|\boldsymbol{F}\|_2 \leq 1\},$$

$\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}$ can be taken as positive definite (hence $\det \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H} > 0$ in a non-degenerated case) and the original problem (4.34) is equivalent to

$$\begin{array}{ll} \text{minimize}_{\text{over}\boldsymbol{A},\boldsymbol{B},\boldsymbol{H},\boldsymbol{E}} & \det \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H} \\ \text{subject to} & [\boldsymbol{A}_k \ \boldsymbol{B}_k] = [\boldsymbol{A} \ \boldsymbol{B}] + \boldsymbol{H}\boldsymbol{F}_k\boldsymbol{E}, \\ & \boldsymbol{F}_k^T\boldsymbol{F}_k \preceq \boldsymbol{I}, \quad k = 1, \ldots, N \\ & \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H} \succ \boldsymbol{0}, \end{array} \tag{4.48}$$

with the measure $\mu(\{\boldsymbol{H}\boldsymbol{F}\boldsymbol{E} : \boldsymbol{F}^T\boldsymbol{F} \preceq \boldsymbol{I}\}) = \det \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}$.

Let $r = \operatorname{rank}(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H})$, where in the degenerate case $r < d$ the objective function (and a measure that is taken) equals the product of the nonzero singular values of $\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}$. Moreover, the set Ω is convex.

Consider a unit uncertainty and a matrix $\boldsymbol{F}(\alpha)$ that belongs to a line segment $\alpha\boldsymbol{F}_1 + (1-\alpha)\boldsymbol{F}_2$, $\alpha \in [0,1]$ $\|\boldsymbol{F}_1\|_2 \leq 1$, $\|\boldsymbol{F}_2\|_2 \leq 1$. Then

$$\|\boldsymbol{F}(\alpha)\|_2 = \|\alpha\boldsymbol{F}_1 + (1-\alpha)\boldsymbol{F}_2\|_2 \leq \alpha\|\boldsymbol{F}_1\|_2 + (1-\alpha)\|\boldsymbol{F}_2\|_2 \leq \alpha + 1 - \alpha = 1.$$

Hence the matrix and the line segment belongs to the uncertainty and therefore the unit uncertainty is convex and by this fact the uncertainty Ω is convex since operations $\boldsymbol{H}$ and $\boldsymbol{E}$ are linear. Consequently if the uncertainty contains the given set of matrices then it also contains their convex hull.

$$\{[\,\boldsymbol{A}_i,\, \boldsymbol{B}_i\,]\}_{i=1}^N \subset \Omega \implies$$
$$Co(\{[\,\boldsymbol{A}_i,\, \boldsymbol{B}_i\,]\}_{i=1}^N) = \left\{ \sum_{i=1}^N \alpha_i [\,\boldsymbol{A}_i,\, \boldsymbol{B}_i\,] : \sum_{i=1}^N \alpha_i = 1, \alpha_i \geq 0, i = 1, \ldots, N \right\} \subset \Omega$$

The problem (4.34) also describes a conversion between polytopic and the norm bounded uncertainty.

4.3.2 Outer and inner approximation of the uncertainty

For all matrices $\boldsymbol{F}$

$$\|\boldsymbol{F}\|_\infty \leq \|\boldsymbol{F}\|_2 \leq \|\boldsymbol{F}\|_F,$$

where $\|\cdot\|_\infty$ and $\|\cdot\|_F$ denote the infinity and Frobenius norms, respectively. For the sets

$$\Omega_\infty = \{[\boldsymbol{A} \ \boldsymbol{B}] + \boldsymbol{H}\boldsymbol{F}\boldsymbol{E} : \|\boldsymbol{F}\|_\infty \leq 1\}, \quad \Omega_F = \{[\boldsymbol{A} \ \boldsymbol{B}] + \boldsymbol{H}\boldsymbol{F}\boldsymbol{E} : \|\boldsymbol{F}\|_F \leq 1\}, \tag{4.49}$$

$\Omega_F \subseteq \Omega \subseteq \Omega_\infty$. Also the image of the sets of (4.49) under the vec operation are

$$\begin{aligned}\text{vec}(\Omega_\infty) &= \{\text{vec}([\boldsymbol{A}\ \boldsymbol{B}]) + \boldsymbol{E}^T \otimes \boldsymbol{H} \cdot \text{vec}(\boldsymbol{F}) : \|\text{vec}(\boldsymbol{F})\|_\infty \leq 1\} \\ \text{vec}(\Omega_F) &= \{\text{vec}([\boldsymbol{A}\ \boldsymbol{B}]) + \boldsymbol{E}^T \otimes \boldsymbol{H} \cdot \text{vec}(\boldsymbol{F}) : \|\text{vec}(\boldsymbol{F})\|_2 \leq 1\}\end{aligned}$$

and since $\|\boldsymbol{F}\|_F = \|\text{vec}(\boldsymbol{F})\|_2$, the Euclidean norm in the vector space equals the Frobenius norm in the matrix space.

Noting that $\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}$ is restricted to be positive definite, let $\boldsymbol{c} = \text{vec}([\boldsymbol{A}\ \boldsymbol{B}])$, $\boldsymbol{f} = \text{vec}(\boldsymbol{F})$ and the set

$$\mathcal{P}(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}, \boldsymbol{c}) = \{\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H} \cdot \boldsymbol{f} + \boldsymbol{c} : \|\boldsymbol{f}\|_\infty \leq 1\} \tag{4.50}$$

represents a hyper-parallelepiped as the image of a hypercube under a linear transformation. The volume is given by

$$\text{vol}(\mathcal{P}) = 2^d \cdot \det \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}$$

Similarly, the set

$$\mathcal{E}(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}, \boldsymbol{c}) = \{\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H} \cdot \boldsymbol{f} + \boldsymbol{c} : \|\boldsymbol{f}\|_2 \leq 1\} \tag{4.51}$$

represents an $r = \text{rank}(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H})$ dimensional ellipsoid in d dimensional space. Thus the norm bounded uncertainty with the Frobenius norm may be represented by a special class of ellipsoids in the vector space defined by the Kronecker product of two matrices. The d dimensional ellipsoid may also be represented as

$$\mathcal{E}(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}, \boldsymbol{c}) = \left\{\boldsymbol{x} \in \mathbb{R}^d : (\boldsymbol{x} - \boldsymbol{c})^{\mathrm{T}}[(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H})(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H})^{\mathrm{T}}]^{-1}(\boldsymbol{x} - \boldsymbol{c}) \leq 1\right\}. \tag{4.52}$$

The volume of the ellipsoid in the case when $r = d$ is given by

$$\text{vol}\ \mathcal{E}(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}) = \beta_d \cdot \det \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}, \quad \beta_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)},$$

where β_d denotes the volume of the d-dimensional ball of unit radius and $\Gamma(\cdot)$ is the Gamma function. Moreover, the outer and inner estimation of Ω is given by

$$\mathcal{E}(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}, \boldsymbol{c}) \subseteq \text{vec}(\Omega) \subseteq \mathcal{P}(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}, \boldsymbol{c}).$$

Also

$$\beta_d \cdot \det \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H} \leq \text{vol}(\Omega) \leq 2^d \det \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}$$

and the transformation considered is illustrated in Figure 4.2. The inner approximation (ellipsoid) is used by the method developed in this thesis to compute the uncertainty.

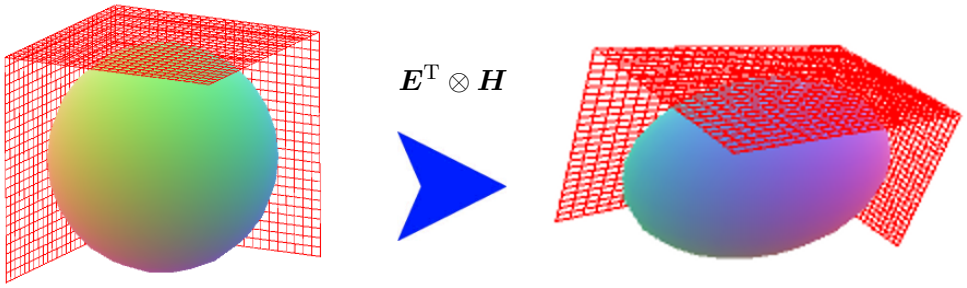

FIGURE 4.2: The uncertainty Ω in the vector space. The hypercube and unit radius ball is transformed by the $\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}$ operation into a hyper-parallelepiped and an ellipsoid, respectively. The uncertainty in the vector space vec(Ω) contains the ellipsoid and is enclosed by the hyper-parallelepiped.

4.3.3 LMI based method

This method was given in Boyd et al. (1994) (see p. 58) and solves the original problem (4.34) in the matrix space. As the measure of the norm bounded uncertainty, either the diameter of the set represented by

$$2\sqrt{\lambda_{\max}(\boldsymbol{H}\boldsymbol{H}^T)\lambda_{\max}(\boldsymbol{E}^T\boldsymbol{E})},$$

or

$$\text{tr } \boldsymbol{H}\boldsymbol{H}^T + \text{tr } \boldsymbol{E}^T\boldsymbol{E}$$

is used. Hence the method solves the following problem (with the second measure).

$$\begin{array}{ll} \text{minimize}_{\text{over}\boldsymbol{X},\boldsymbol{H},\boldsymbol{E}} & \text{tr } \boldsymbol{H}\boldsymbol{H}^T + \text{tr } \boldsymbol{E}^T\boldsymbol{E} \\ \text{subject to} & \boldsymbol{X}_k = \boldsymbol{X} + \boldsymbol{H}\boldsymbol{F}_k\boldsymbol{E}, \quad \boldsymbol{F}_k^T\boldsymbol{F}_k \preceq \boldsymbol{I}, \quad k = 1, \ldots, N \end{array}$$

The derivation is as follows.

Consider the polytope in terms of its vertices and a set Ω

$$\mathcal{P} = Co\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}, \quad \Omega = \{\boldsymbol{X} + \boldsymbol{H}\boldsymbol{F}\boldsymbol{E} : \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} \preceq \boldsymbol{I}\},$$

where $\boldsymbol{H}, \boldsymbol{E}$ are assumed to be square. For simplicity it is also assumed that $\boldsymbol{H}$ is invertible. The goal is to find the matrices $\boldsymbol{H}$ and $\boldsymbol{E}$ such that $\mathcal{P} \subseteq \Omega$. Note that $\boldsymbol{H}$ and $\boldsymbol{E}$ can be replaced by $\boldsymbol{H}\boldsymbol{U}$ and $\boldsymbol{V}\boldsymbol{E}$ where $\boldsymbol{U}, \boldsymbol{V}$ are any orthogonal matrices, without affecting the set Ω.
The vertex $\boldsymbol{X}_k$ belongs to the norm bounded uncertainty if there exists a matrix $\boldsymbol{F}_k^T\boldsymbol{F}_k \preceq$

$\boldsymbol{I}$ such that

$$\boldsymbol{X} + \boldsymbol{H}\boldsymbol{F}_k\boldsymbol{E} = \boldsymbol{X}_k,$$

or

$$\boldsymbol{F}_k\boldsymbol{E} = \boldsymbol{H}^{-1}(\boldsymbol{X}_k - \boldsymbol{X}).$$

Consider a vector $\boldsymbol{\xi}$ and hence

$$\boldsymbol{\pi} = \boldsymbol{F}_k\boldsymbol{E}\boldsymbol{\xi} = \boldsymbol{H}^{-1}(\boldsymbol{X}_k - \boldsymbol{X})\boldsymbol{\xi}.$$

Therefore $\mathcal{P} \subseteq \Omega$ if for every $\boldsymbol{\xi}$ and $i = 1, \ldots, N$, there exists $\boldsymbol{\pi}$ such that

$$\boldsymbol{H}\pi = (\boldsymbol{X}_k - \boldsymbol{X})\boldsymbol{\xi}, \qquad \boldsymbol{\pi}^{\mathrm{T}}\boldsymbol{\pi} \leq \boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}\boldsymbol{\xi}.$$

This gives the equivalent condition

$$(\boldsymbol{X}_k - \boldsymbol{X})^{\mathrm{T}}\boldsymbol{H}^{-\mathrm{T}}\boldsymbol{H}^{-1}(\boldsymbol{X}_k - \boldsymbol{X}) \preceq \boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}, \qquad k = 1, \ldots, N$$

and applying the Schur's complement formula yields

$$\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} \succ 0, \quad \begin{bmatrix} \boldsymbol{E}^{\mathrm{T}}\boldsymbol{E} & (\boldsymbol{X}_k - \boldsymbol{X})^{\mathrm{T}} \\ \boldsymbol{X}_k - \boldsymbol{X} & \boldsymbol{H}\boldsymbol{H}^{T} \end{bmatrix} \succeq 0 \qquad k = 1, \ldots, N.$$

However, this is not a LMI in $\boldsymbol{H}$ and $\boldsymbol{E}$, but $\boldsymbol{V} = \boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$ and $\boldsymbol{W} = \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}$ are symmetric matrices and this gives an equivalent LMI in terms of $\boldsymbol{X}, \boldsymbol{V}, \boldsymbol{W}$

$$\boldsymbol{W} \succ 0, \quad \begin{bmatrix} \boldsymbol{V} & (\boldsymbol{X}_k - \boldsymbol{X})^{\mathrm{T}} \\ \boldsymbol{X}_k - \boldsymbol{X} & \boldsymbol{W} \end{bmatrix} \succeq 0 \qquad k = 1, \ldots, N.$$

Hence the problem considered is equivalent to

$$\begin{array}{ll} \text{minimize}_{\text{over}\boldsymbol{X},\boldsymbol{V},\boldsymbol{W}} & \text{tr } \boldsymbol{V} + \text{tr } \boldsymbol{W} \\ \text{subject to} & (4.3), \end{array}$$

which may be solved by existing SDP software. After obtaining a solution it necessary to factorize $\boldsymbol{V}$ and $\boldsymbol{W}$. Since the matrices are symmetric either Cholesky factorization or the square root of a matrix can be used. Both factorization results represent the same norm bounded uncertainty. Another implicit issue is that $\boldsymbol{H}$ and $\boldsymbol{E}$ must be square. The matrix space method may be used to solve the original problem by taking $\boldsymbol{X}_k = [\, \boldsymbol{A}_k \;\; \boldsymbol{B}_k \,]$, $k = 1, \ldots, N$ and, after obtaining a solution, partitioning $\boldsymbol{X} = [\, \boldsymbol{A} \;\; \boldsymbol{B} \,]$.

First observe that the set

$$\Omega = \{\boldsymbol{A} + \boldsymbol{H}\boldsymbol{F}\boldsymbol{E} : \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} \preceq \boldsymbol{I}\}$$

is invariant if $\boldsymbol{H}$ and $\boldsymbol{E}$ are replaced by $\boldsymbol{H}\boldsymbol{U}$ and $\boldsymbol{V}\boldsymbol{E}$, respectively, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are

any orthogonal matrices and

$$\Omega' = \{\boldsymbol{A} + \boldsymbol{HUFVE} : \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} \preceq I\} =$$

$$= \{\boldsymbol{A} + \boldsymbol{HF'E} : \boldsymbol{F'} = \boldsymbol{UFV}, \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} \preceq \boldsymbol{I}\}.$$

Observe that

$$\boldsymbol{F'}^{\mathrm{T}}\boldsymbol{F'} = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{UFV} = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{FV} \preceq \boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} = \boldsymbol{I}.$$

and hence

$$\Omega' = \{\boldsymbol{A} + \boldsymbol{HF'E} : \boldsymbol{F'}^{\mathrm{T}}\boldsymbol{F'} \preceq \boldsymbol{I}\} = \Omega.$$

The LMI constraints for $\boldsymbol{W} = \boldsymbol{HH}^{\mathrm{T}}, \boldsymbol{V} = \boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$ are $\boldsymbol{W} \succ \boldsymbol{0}$ and $\boldsymbol{V} \succ \boldsymbol{0}$. Let $\boldsymbol{G}$,$\boldsymbol{D}$ be the Cholesky factorization of $\boldsymbol{W}$ and $\boldsymbol{V}$ respectively, such that

$$\boldsymbol{GG}^{\mathrm{T}} = \boldsymbol{W} = \boldsymbol{HH}^{\mathrm{T}}, \qquad \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D} = \boldsymbol{V} = \boldsymbol{E}^{\mathrm{T}}\boldsymbol{E},$$

which exist since the matrices $\boldsymbol{W}$ and $\boldsymbol{V}$ are positive definite. Assume that $\boldsymbol{H}, \boldsymbol{G}$ and $\boldsymbol{E}, \boldsymbol{D}$ are invertible and define

$$\boldsymbol{U}_H{:=}\boldsymbol{H}^{-1}\boldsymbol{G}(\boldsymbol{G} = \boldsymbol{HU}_H), \qquad \boldsymbol{U}_E{:=}\boldsymbol{DE}^{-1}(\boldsymbol{D} = \boldsymbol{U}_E\boldsymbol{E}).$$

Then, as shown below the matrices $\boldsymbol{U}_H$ and $\boldsymbol{U}_E$ are orthogonal hence the following sets are equal

$$\Omega = \{\boldsymbol{A} + \boldsymbol{HFE} : \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} \preceq \boldsymbol{I}\} = \{\boldsymbol{A} + \boldsymbol{GFD} : \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} \preceq \boldsymbol{I}\} = \Omega'.$$

Observe that

$$\boldsymbol{W} = \boldsymbol{HH}^{\mathrm{T}} \Rightarrow \boldsymbol{H}^{-1}\boldsymbol{W} = \boldsymbol{H}^{\mathrm{T}} \Rightarrow \boldsymbol{H}^{-1}\boldsymbol{WH}^{-\mathrm{T}} = \boldsymbol{I} \Rightarrow$$

$$\Rightarrow \boldsymbol{H}^{-1}\boldsymbol{GG}^{\mathrm{T}}\boldsymbol{H}^{-\mathrm{T}} = \boldsymbol{I} \Rightarrow \boldsymbol{U}_H\boldsymbol{U}_H^{\mathrm{T}} = \boldsymbol{I}$$

and also

$$\boldsymbol{W}^{-1} = (\boldsymbol{GG}^{\mathrm{T}})^{-1} \Rightarrow \boldsymbol{W}^{-1} = \boldsymbol{G}^{-\mathrm{T}}\boldsymbol{G}^{-1} \Rightarrow \boldsymbol{G}^{\mathrm{T}}\boldsymbol{W}^{-1}\boldsymbol{G} = \boldsymbol{I} \Rightarrow$$

$$\Rightarrow \boldsymbol{G}^{\mathrm{T}}\boldsymbol{H}^{-\mathrm{T}}\boldsymbol{H}^{-1}\boldsymbol{G} = \boldsymbol{I} \Rightarrow \boldsymbol{U}_H^{\mathrm{T}}\boldsymbol{U}_H = \boldsymbol{I}.$$

Hence $\boldsymbol{U}_H$ is an orthogonal matrix. A similar proof holds for orthogonality of $\boldsymbol{U}_E$. Consequently the matrices obtained by Cholesky factorization can be used in order to describe the uncertainty.

4.3.4 The new method

The new method is developed in the vector space setting where the input set of matrices is mapped to a set of vectors

$$\begin{aligned}&\mathbb{R}^{n(n+m)} \ni \boldsymbol{x}_k = \text{vec}([\,\boldsymbol{A}_k\ \boldsymbol{B}_k\,]), \quad k = 1, \ldots, N\\ &\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]\,, \quad \text{rank}(\boldsymbol{X}) = d,\end{aligned}$$

which are assumed to span the whole space ($d = n(n+m)$). In our approach the fact that Ω_F is an ellipsoid in the vector space is used. We solve initially the problem of finding a minimum volume ellipsoid representing Ω_F and containing all the points

$$\begin{aligned}&\text{minimize}_{\text{over}\boldsymbol{A},\boldsymbol{B},\boldsymbol{H},\boldsymbol{E}} \qquad \log\det\,(\boldsymbol{E}^T \otimes \boldsymbol{H})\\ &\text{subject to} \qquad (\boldsymbol{x}_k - \boldsymbol{c})^T[(\boldsymbol{E}^T \otimes \boldsymbol{H})(\boldsymbol{E}^T \otimes \boldsymbol{H})^{\mathrm{T}}]^{-1}(\boldsymbol{x}_k - \boldsymbol{c}) \le 1,\\ &\qquad\qquad k = 1, \ldots, N, \quad \boldsymbol{c} = \text{vec}([\,\boldsymbol{A}\ \boldsymbol{B}\,]).\end{aligned}$$

The solution is finally obtained using the fact that $\Omega_F \subseteq \Omega$, resulting in matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{H}, \boldsymbol{E}$ that represent the solution to the original problem (4.34).

The foundation of the method is the initial approximation of the MVEE, that is, the suboptimal solution of the problem

$$\begin{aligned}&\text{minimize}_{\text{over }\boldsymbol{P},\boldsymbol{c}} \qquad \log\det\,\boldsymbol{P}\\ &\text{subject to} \qquad (\boldsymbol{x}_k - \boldsymbol{c})^T(\boldsymbol{P}\boldsymbol{P}^{\mathrm{T}})^{-1}(\boldsymbol{x}_k - \boldsymbol{c}) \le 1, \quad k = 1, \ldots, N.\end{aligned} \tag{4.53}$$

One method for solving the MVEE is given in Khachiyan (1996), which computes a $(1+\varepsilon)$ approximation, that is, finds an ellipsoid $\mathcal{E}(\boldsymbol{M}, \boldsymbol{c})$ such that

$$\text{vol}\ \mathcal{E}(\boldsymbol{M}, \boldsymbol{c}) \le (1+\varepsilon)\text{vol MVEE}(\boldsymbol{X}).$$

The computational time complexity of this method is given by Kumar and Yildrim (2005) as

$$O(Nd^2([(1+\varepsilon)^{(2/d+1)} - 1]^{-1} + \log d + \log\log N)).$$

However, since the uncertainty ellipsoid Ω_F may not represent accurately the MVEE of Khachiyan (1996), the computational time may be wasted. Better results can be obtained using the developed Algorithm 3.3.5 based on PCA that produces a good approximation to MVEE and with lower complexity, which is illustrated in Figure 4.3.

In the step 1 of this method the initial axes of the enclosing ellipsoid are computed and in step 2 the axes length. In step 3, the ellipsoid representing uncertainty is fitted to the one obtained in step 2 and in step 4 the axes length of the new ellipsoid are computed in order to ensure it contains all the points. Step 5 then implements the contracting operation shrinking using a simple procedure. At the end of these 5 steps matrices representing the norm bounded uncertainty are obtained.

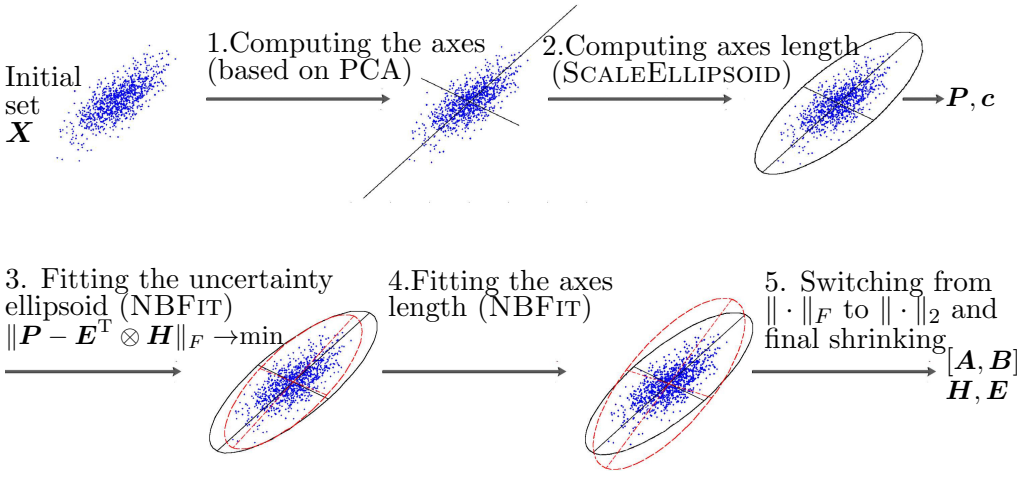

Figure 4.3: Illustration of the new method.

Fitting the axes of the uncertainty ellipsoid

Assume that the Enclosing Ellipsoid $\mathcal{E}(\boldsymbol{P}, \boldsymbol{c})$ contains all the points. Consider the SVD

$$\boldsymbol{P} = \boldsymbol{U}\mathrm{diag}(\boldsymbol{\sigma})\boldsymbol{V}^{\mathrm{T}}, \quad \boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n(n+m)}],$$

where the left singular vectors $\boldsymbol{U}$ together with the singular values $\boldsymbol{\sigma}$ define the axes of the ellipsoid. In order to fit the uncertainty to the ellipsoid we approximate in the Frobenius norm the ellipsoid matrix by the Kronecker product of two matrices of compatible dimensions. In particular, the problem solved is

$$\text{minimize}_{\text{over}\boldsymbol{E},\boldsymbol{H}} \qquad \|\boldsymbol{P} - \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}\|_F. \tag{4.54}$$

Approximation with the Kronecker product

The solution is given in van Loan and Pitsianis (1993). After rearranging the matrix $\boldsymbol{P}$ the initial point is obtained as a rank-1 approximation. As an example, consider the matrices

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}.$$

Hence, by rearranging the matrix $\boldsymbol{A}$,

$$\|\boldsymbol{A} - \boldsymbol{B} \otimes \boldsymbol{C}\|_F = \left\| \begin{bmatrix} a_{11} & a_{21} & a_{12} & a_{22} \\ a_{31} & a_{41} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{14} & a_{24} \\ a_{33} & a_{43} & a_{34} & a_{44} \end{bmatrix} - \begin{bmatrix} b_{11} \\ b_{21} \\ b_{12} \\ b_{22} \end{bmatrix} \begin{bmatrix} c_{11} & c_{21} & c_{12} & c_{22} \end{bmatrix} \right\|_F = \|\widetilde{\boldsymbol{A}} - \boldsymbol{b}\boldsymbol{c}^{\mathrm{T}}\|_F.$$

In general, consider the matrices

$$\boldsymbol{A} \in \mathbb{R}^{m\times n}, \quad \boldsymbol{B} \in \mathbb{R}^{m_1\times n_1}, \quad \boldsymbol{C} \in \mathbb{R}^{m_2\times n_2}, \quad m = m_1 \cdot m_2, \quad n = n_1 \cdot n_2$$

and consider the uniform block partitioning of the matrix $\boldsymbol{A}$ as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \dots & \boldsymbol{A}_{1,n_1} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \dots & \boldsymbol{A}_{2,n_1} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{m_1,1} & \boldsymbol{A}_{m_1,2} & \dots & \boldsymbol{A}_{m_1,n_1} \end{bmatrix}, \quad \boldsymbol{A}_{ij} \in \mathbb{R}^{m_2\times n_2}.$$

The rearrangement of the matrix $\boldsymbol{A}$ is defined as

$$R(\boldsymbol{A}) := \begin{bmatrix} \boldsymbol{A}_1 \\ \vdots \\ \boldsymbol{A}_{n_1} \end{bmatrix}, \quad \boldsymbol{A}_j = \begin{bmatrix} \mathrm{vec}(\boldsymbol{A}_{1,j})^{\mathrm{T}} \\ \vdots \\ \mathrm{vec}(\boldsymbol{A}_{m_1,j})^{\mathrm{T}} \end{bmatrix}, \quad j = 1, \dots, n_1$$

and hence

$$\|\boldsymbol{A} - \boldsymbol{B} \otimes \boldsymbol{C}\|_F = \|R(\boldsymbol{A}) - \boldsymbol{b} \otimes \boldsymbol{c}^{\mathrm{T}}\|_F, \quad \boldsymbol{b} = \mathrm{vec}(\boldsymbol{B}), \quad \boldsymbol{c} = \mathrm{vec}(\boldsymbol{C}).$$

Consider the SVD

$$\widetilde{\boldsymbol{A}} = R(\boldsymbol{A}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_m] \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{v}_n^{\mathrm{T}} \end{bmatrix}.$$

As the solution that minimizes $\|\boldsymbol{A} - \boldsymbol{B} \otimes \boldsymbol{C}\|_F$ take

$$\boldsymbol{b} = \sigma_1 \boldsymbol{u}_1, \quad \boldsymbol{c} = \boldsymbol{v}_1, \quad \boldsymbol{B} = \mathrm{vec}^{-1}(\boldsymbol{b}), \quad \boldsymbol{C} = \mathrm{vec}^{-1}(\boldsymbol{c}).$$

Next the solution is updated by solving a series of least squares problem (5 iterations in the test implementation). It is proved in van Loan and Pitsianis (1993) that if $\boldsymbol{C}$ is fixed then the matrix $\boldsymbol{B}$ with entries defined as follows

$$b_{ij} = \frac{\mathrm{tr}(\boldsymbol{A}_{ij}^{\mathrm{T}}\boldsymbol{C})}{\mathrm{tr}(\boldsymbol{C}^{\mathrm{T}}\boldsymbol{C})}, \quad 1 \le i \le m_1, 1 \le j \le n_1,$$

minimizes $\|\boldsymbol{A} - \boldsymbol{B} \otimes \boldsymbol{C}\|_F$. Similarly if $\boldsymbol{B}$ is fixed then the matrix $\boldsymbol{C}$ with entries

$$c_{ij} = \frac{\mathrm{tr}(\widehat{\boldsymbol{A}}_{ij}^{\mathrm{T}}\boldsymbol{B})}{\mathrm{tr}(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B})}, \quad 1 \le i \le m_2, 1 \le j \le n_2,$$

where (using Matlab notation)

$$\widehat{\boldsymbol{A}}_{ij} = \boldsymbol{A}(i : m_2 : m, j : n_2 : n)$$

minimizes $\|\boldsymbol{A} - \boldsymbol{B} \otimes \boldsymbol{C}\|_F$.

Fitting the length of uncertainty ellipsoid axes

Consider the SVD

$$\boldsymbol{E}^{\mathrm{T}} = \boldsymbol{U}_E \boldsymbol{S}_E \boldsymbol{V}_E^{\mathrm{T}}, \quad \boldsymbol{H} = \boldsymbol{U}_H \boldsymbol{S}_H \boldsymbol{V}_H^{\mathrm{T}},$$

where

$$\boldsymbol{S}_E = \mathrm{diag}(e_1, \ldots, e_{n+m}), \quad \boldsymbol{S}_H = \mathrm{diag}(h_1, \ldots, h_n).$$

By properties of the Kronecker product

$$\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H} = (\boldsymbol{U}_E \otimes \boldsymbol{U}_H) \cdot (\boldsymbol{S}_E \otimes \boldsymbol{S}_H) \cdot (\boldsymbol{V}_E^{\mathrm{T}} \otimes \boldsymbol{V}_H^{\mathrm{T}})$$

and the lengths of axes of the ellipsoid are given as

$$\boldsymbol{S}_E \otimes \boldsymbol{S}_H = \mathrm{diag}(e_1 h_1, \ldots, e_1 h_n, e_2 h_1, \ldots, e_{n+m} h_n).$$

However, after approximation by (4.54) there is no guarantee that the resulting ellipsoid contains all the points. New axes lengths need to be computed by the scaling procedure

$$(\sigma'_1, \ldots, \sigma'_{n(n+m)}) = \mathrm{ScaleEllipsoid}(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}, \boldsymbol{c})$$

and it is required to minimize the volume of the ellipsoid such that the new axes are not less than those obtained in order to guarantee that it contains all the points. The problem to be solved is

$$\begin{array}{l} \text{minimize}_{\text{over } \boldsymbol{e},\boldsymbol{h}} \quad \phi_{\text{obj}}(\boldsymbol{e}, \boldsymbol{h}) = \prod_{i=1}^{n+m} e_i^n \cdot \prod_{i=1}^{n} h_i^{n+m} \\ \text{subject to} \\ \qquad \begin{array}{rcl} e_1 h_1 & \geq & \sigma'_1 \\ e_1 h_2 & \geq & \sigma'_2 \\ \ldots & \ldots & \ldots \\ e_1 h_n & \geq & \sigma'_n \\ e_2 h_1 & \geq & \sigma'_{n+1} \\ \ldots & \ldots & \ldots \\ e_{n+m} h_n & \geq & \sigma'_{n(n+m)} \end{array} \end{array} \tag{4.55}$$

which, in general, is non-convex with bi-linear constraints. Considering groups of constraints with the same variables the functions $\boldsymbol{\epsilon} : \mathbb{R}^n \mapsto \mathbb{R}^{n+m}$ can be defined

$$\epsilon_i(\boldsymbol{h}) = \max \left\{ \frac{\sigma'_{(i-1)n+1}}{h_1}, \frac{\sigma'_{(i-1)n+2}}{h_2} \ldots, \frac{\sigma'_{i \cdot n}}{h_n} \right\}, \quad i = 1, \ldots, n+m$$

and $\boldsymbol{\eta} : \mathbb{R}^{n+m} \mapsto \mathbb{R}^n$

$$\eta_i(\boldsymbol{e}) = \max \left\{ \frac{\sigma'_i}{e_1}, \frac{\sigma'_{n+i}}{e_2} \ldots, \frac{\sigma'_{n(n+m-1)+i}}{e_{n+m}} \right\}, \quad i = 1, \ldots, n.$$

Assume that

$$[\boldsymbol{e}^{\star\mathrm{T}}, \boldsymbol{h}^{\star\mathrm{T}}]^{\mathrm{T}} = [e_1^\star, \ldots, e_{n+m}^\star, h_1^\star, \ldots, h_n^\star]^{\mathrm{T}}$$

is the minimizer of (4.55). The necessary conditions for the optimum are

$$\boldsymbol{e}^\star = \boldsymbol{\epsilon}(\boldsymbol{h}^\star), \quad \boldsymbol{h}^\star = \boldsymbol{\eta}(\boldsymbol{e}^\star), \tag{4.56}$$

since otherwise one of the points $[\boldsymbol{\epsilon}(\boldsymbol{h}^\star)^{\mathrm{T}}, \boldsymbol{h}^{\star\mathrm{T}}]^{\mathrm{T}}$ and $[\boldsymbol{e}^{\star\mathrm{T}}, \boldsymbol{\eta}(\boldsymbol{e}^\star)^{\mathrm{T}}]^{\mathrm{T}}$ would give a lower value of the objective function whilst still satisfying constraints.

Theorem 4.7. *Consider any point* $[\boldsymbol{e}^T, \boldsymbol{h}^T]^T$. *Then*

$$\begin{bmatrix} \tilde{\boldsymbol{e}} \\ \tilde{\boldsymbol{h}} \end{bmatrix} = \begin{bmatrix} (\boldsymbol{\epsilon} \circ \boldsymbol{\eta})(\boldsymbol{e}) \\ \boldsymbol{\eta}(\boldsymbol{e}) \end{bmatrix}, \quad \begin{bmatrix} \bar{\boldsymbol{e}} \\ \bar{\boldsymbol{h}} \end{bmatrix} = \begin{bmatrix} \epsilon(\boldsymbol{h}) \\ (\boldsymbol{\eta} \circ \boldsymbol{\epsilon})(\boldsymbol{h}) \end{bmatrix} \tag{4.57}$$

are stationary points satisfying (4.56).

Proof: Consider $\eta_1(\tilde{\boldsymbol{e}})$

$$\eta_1(\tilde{\boldsymbol{e}}) = \max\left\{ \frac{\sigma'_1}{\tilde{e}_1}, \frac{\sigma'_{n+1}}{\tilde{e}_2}, \ldots, \frac{\sigma'_{n(n+m-1)+1}}{\tilde{e}_{n+m}} \right\} \tag{4.58}$$

and since $\tilde{\boldsymbol{e}} = (\boldsymbol{\epsilon} \circ \boldsymbol{\eta})(\boldsymbol{e}) = \boldsymbol{\epsilon}(\tilde{\boldsymbol{h}})$ we have for $i = 1, \ldots, n+m$

$$\tilde{e}_i \geq \frac{\sigma'_{(i-1)n+1}}{\tilde{h}_1} \iff \tilde{h}_1 \geq \frac{\sigma'_{(i-1)n+1}}{\tilde{e}_i}. \tag{4.59}$$

Hence $\eta_1(\tilde{\boldsymbol{e}}) \leq \tilde{h}_1$ and it is now shown that $\eta_1(\tilde{\boldsymbol{e}}) = \tilde{h}_1$.

Recall that $\tilde{h}_1 = \eta_1(\boldsymbol{e})$

$$\tilde{h}_1 = \max\left\{ \frac{\sigma'_{(i-1)n+1)}}{e_i} : i = 1, \ldots, n+m \right\}$$

and assume that the maximum holds for $i = i_m$. Then

$$e_{i_m} = \frac{\sigma'_{(i_m-1)n+1)}}{\tilde{h}_1} \tag{4.60}$$

and also

$$\tilde{e}_{i_m} = \max\left\{ \frac{\sigma'_{(i_m-1)n+j}}{\tilde{h}_j} : j = 1, \ldots, n \right\}. \tag{4.61}$$

Consider $j = 1, \ldots, n, j \neq i_m$, where since $\tilde{h}_j = \eta_j(\boldsymbol{e})$

$$\tilde{h}_j \geq \frac{\sigma'_{(i_m-1)n+j}}{e_{i_m}} \iff \frac{\sigma'_{(i_m-1)n+j}}{\tilde{h}_j} \leq e_{i_m}$$

That implies, together with (4.60) and (4.61) that $\tilde{e}_{i_m} = e_{i_m}$ and also

$$\frac{\sigma'_{(i_m-1)n+1}}{\tilde{e}_{i_m}} = \tilde{h}_1.$$

Combining this last condition with (4.58) and (4.59) gives $\eta_1(\tilde{\boldsymbol{e}}) = \tilde{h}_1$. Repeating this procedure for remaining $\eta_i(\tilde{\boldsymbol{e}})$, $i = 2, \ldots, n$ establishes that $\boldsymbol{\eta}(\tilde{\boldsymbol{e}}) = \tilde{\boldsymbol{h}}$. Showing that $\boldsymbol{\epsilon}(\tilde{\boldsymbol{h}}) = \tilde{\boldsymbol{e}}$ is straightforward since $\tilde{\boldsymbol{e}} = (\boldsymbol{\epsilon} \circ \boldsymbol{\eta})(\boldsymbol{e}) = \boldsymbol{\epsilon}(\tilde{\boldsymbol{h}})$. □

For approximation purposes the method starts from the initial point

$$\left[\boldsymbol{e}^{\mathrm{T}},\ \boldsymbol{h}^{\mathrm{T}}\right]^{\mathrm{T}} = \left[\mathbf{1}^{\mathrm{T}},\ \mathbf{1}^{\mathrm{T}}\right]^{\mathrm{T}}, \tag{4.62}$$

computes (4.57) and selects the one $[\hat{\boldsymbol{e}}^{\mathrm{T}}, \hat{\boldsymbol{h}}^{\mathrm{T}}]^{\mathrm{T}}$ that gives the lowest value of the objective function.

Final contraction of the uncertainty

After fitting the length of the axes, the uncertainty is constructed using

$$\begin{aligned} \boldsymbol{E} &= \boldsymbol{V}_E \cdot \mathrm{diag}(\hat{e}_1, \ldots, \hat{e}_{n+m}) \cdot \boldsymbol{U}_E^{\mathrm{T}} \\ \boldsymbol{H} &= \boldsymbol{U}_H \cdot \mathrm{diag}(\hat{h}_1, \ldots, \hat{h}_n) \cdot \boldsymbol{V}_H^{\mathrm{T}} \\ [\ \boldsymbol{A},\ \boldsymbol{B}\ ] &= \mathrm{vec}^{-1}(\boldsymbol{c}). \end{aligned}$$

The use of the induced Euclidean matrix norm in the definition of the norm bounded uncertainty in comparison to the Frobenius norm for the ellipsoid results in some volume redundancy. To reduce this redundancy, the following simple linear procedure is used.

Let $f_k = \|\boldsymbol{H}^{-1}\big([\boldsymbol{A}_k,\ \boldsymbol{B}_k] - [\boldsymbol{A},\ \boldsymbol{B}]\big)\boldsymbol{E}^{-1}\|_2$ for $k = 1, \ldots, N$ and define

$$f_{\max} = \max\{f_k : k = 1, \ldots, N\} \leq 1.$$

Setting $\boldsymbol{H}' = f_{\max} \cdot \boldsymbol{H}$, or $\boldsymbol{E}' = f_{\max} \cdot \boldsymbol{E}$, and evaluating for $k = 1, \ldots, N$

$$f'_k = \|\boldsymbol{H}'^{-1}\big([\boldsymbol{A}_k,\ \boldsymbol{B}_k] - [\boldsymbol{A},\ \boldsymbol{B}]\big)\boldsymbol{E}^{-1}\|_2$$

gives

$$\max\left\{f'_k : k = 1, \ldots, N\right\} = 1.$$

The approximated uncertainty is given by $[\boldsymbol{A},\ \boldsymbol{B}]$ and $\boldsymbol{H}', \boldsymbol{E}$ and the method is summarized in Algorithm 4.3.1.

Input points on a hyperplane

Assume that the input considered lies on an r-dimensional hyperplane, i.e.,

$$\mathrm{rank}(\boldsymbol{X}') = r < n(n+m).$$

Algorithm 4.3.1 Fitting uncertainty to the ellipsoid

function FITNB($\boldsymbol{P}$,$\boldsymbol{c}$)
  MINIMIZE$_{\text{over}\boldsymbol{E},\boldsymbol{H}}$ $\|\boldsymbol{P} - \boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}\|_F$
  $(\boldsymbol{U}_E, \boldsymbol{S}_E, \boldsymbol{V}_E)$=SVD($\boldsymbol{E}$) ▷ SVD Decomposition $\boldsymbol{E} = \boldsymbol{U}_E \boldsymbol{S}_E \boldsymbol{V}_E^{\mathrm{T}}$
  $(\boldsymbol{U}_H, \boldsymbol{S}_H, \boldsymbol{V}_H)$=SVD($\boldsymbol{H}$) ▷ SVD Decomposition $\boldsymbol{H} = \boldsymbol{U}_H \boldsymbol{S}_H \boldsymbol{V}_H^{\mathrm{T}}$
  $\boldsymbol{\sigma}' \leftarrow$ SCALEELLIPSOID($\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}$, $\boldsymbol{c}$)
  $\begin{bmatrix} \tilde{\boldsymbol{e}} \\ \tilde{\boldsymbol{h}} \end{bmatrix} \leftarrow \begin{bmatrix} (\boldsymbol{\epsilon} \circ \boldsymbol{\eta})(\mathbf{1}) \\ \boldsymbol{\eta}(\mathbf{1}) \end{bmatrix}, \begin{bmatrix} \bar{\boldsymbol{e}} \\ \bar{\boldsymbol{h}} \end{bmatrix} \leftarrow \begin{bmatrix} \epsilon(\mathbf{1}) \\ (\boldsymbol{\eta} \circ \boldsymbol{\epsilon})(\mathbf{1}) \end{bmatrix}$
  if $\phi_{\text{obj}}(\tilde{\boldsymbol{e}}, \tilde{\boldsymbol{h}}) < \phi_{\text{obj}}(\bar{\boldsymbol{e}}, \bar{\boldsymbol{h}})$ **then**
    $\hat{\boldsymbol{e}} = \tilde{\boldsymbol{e}}$, $\hat{\boldsymbol{h}} = \tilde{\boldsymbol{h}}$
  else
    $\hat{\boldsymbol{e}} = \bar{\boldsymbol{e}}$, $\hat{\boldsymbol{h}} = \bar{\boldsymbol{h}}$
  end if
  $\boldsymbol{E} \leftarrow \boldsymbol{V}_E \cdot \mathrm{diag}(\hat{\boldsymbol{e}}) \cdot \boldsymbol{U}_E^{\mathrm{T}}$, $\boldsymbol{H} \leftarrow \boldsymbol{U}_H \cdot \mathrm{diag}(\hat{\boldsymbol{h}}) \cdot \boldsymbol{V}_H^{\mathrm{T}}$
  $[\boldsymbol{A},\ \boldsymbol{B}] \leftarrow \mathrm{vec}^{-1}(\boldsymbol{c})$
  for $k := 1, \ldots, N$ **do**
    $f_k = \|\boldsymbol{H}^{-1}\big([\boldsymbol{A}_k,\ \boldsymbol{B}_k] - [\boldsymbol{A},\ \boldsymbol{B}]\big)\boldsymbol{E}^{-1}\|_2$
  end for
  $f_{\max} \leftarrow \max\{f_k : k = 1, \ldots, N\}$, $\boldsymbol{H}' \leftarrow f_{\max} \cdot \boldsymbol{H}$
return $[\boldsymbol{A},\ \boldsymbol{B}], \boldsymbol{H}', \boldsymbol{E}$
end function

Then in this case the resulting enclosing ellipsoid is of rank r and the matrix $\boldsymbol{P}$ has r nonzero singular values. However, the Frobenius norm approximation (4.54) usually results in a full rank matrix. The singular values play the role of the weights for the singular vectors and better results can be obtained in this case if the remaining $n(n+m) - r$ singular values are set to some value and allowing them to be approximated. The proposed heuristic value is

$$\sigma_i = 0.15 \cdot \min\{\sigma_1, \ldots, \sigma_r\}, \quad i = r+1, \ldots, n(n+m) \tag{4.63}$$

and hence the ellipsoid dimension is increased from r to $n(n+m)$ by setting the length of remaining axes to a nonzero value.

Computational complexity

The computation of the initial enclosing ellipsoid requires the computation of the SVD of the data matrix $\boldsymbol{X}' \in \mathbb{R}^{d \times N}$ with $N \gg d$. A variant of the QR method Golub and Kahan (1964) implemented in LAPACK and used by Matlab requires $O(Nd^2)$ operations. The alternative method of computing ellipsoid axes requires the calculation of the matrix $\boldsymbol{\Sigma}$, which is $O(Nd^2)$, and its eigenvalue-eigenvector decomposition which , by using a variant of the QR method Golub and van Loan (1996), has a complexity $O(d^3)$.

Summarizing for $N \gg d$ the total complexity of the first step in the new method is (Nd^2). The second method is very sensitive to errors although it may be faster than the first and the approximate nature of the method makes it reasonable to use the second

method in some cases. The complexity of the Fitting Axes algorithm in the worst-case scenario is $O(d^2)$ and the scaling procedure is bounded by $O(Nd^2)$. The FitNB requires $O(Nd^2)$ operations, including SVD, and hence the overall complexity is $O(Nd^2)$.

4.3.5 Numerical tests

The following tests compare the developed heuristic method with the existing method based on LMI approach. Initially the problem is set using the Yalmip interface Lofberg (2004) and then exported to SeDuMi. However, in the tests this time is ignored and only that to solve the problem is considered. Also the comparisons are between the measure of the uncertainty, which is taken as

$$\mu(\{[\,\boldsymbol{A},\,\boldsymbol{B}\,]+\boldsymbol{HFE}:\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}\preceq\boldsymbol{I}\})=\det\boldsymbol{E}^{\mathrm{T}}\otimes\boldsymbol{H}$$

and the computation times, respectively.
Define

$$\gamma:=\sqrt[n(n+m)]{\frac{\det\boldsymbol{E}^{\mathrm{T}}\otimes\boldsymbol{H}}{\det\boldsymbol{E}_m^{\mathrm{T}}\otimes\boldsymbol{H}_m}},$$

where $\boldsymbol{H}$ and $\boldsymbol{E}$ describe the norm bounded uncertainty obtained by the heuristic method and $\boldsymbol{H}_m$ and $\boldsymbol{E}_m$ that obtained by the LMI based method, respectively. The coefficient γ determines the factor by which all axes of the ellipsoid $\boldsymbol{E}_m^{\mathrm{T}}\otimes\boldsymbol{H}_m$ should be multiplied in order to have the same volume as the ellipsoid $\boldsymbol{E}^{\mathrm{T}}\otimes\boldsymbol{H}$. In terms of uncertainty

$$\mu(\{\,\boldsymbol{HFE}:\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}\preceq\boldsymbol{I}\,\})=\mu(\{\,\gamma\boldsymbol{H}_m\boldsymbol{F}\boldsymbol{E}_m:\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}\preceq\boldsymbol{I}\,\}).$$

The tests were performed on a computer with Intel Core i3 2.2Ghz and 3GB RAM under Linux and MATLAB 2008a with SeDuMi 1.2. Representative results are given in Figures 4.4 and 4.5 and Table 4.1.

Time & Measure vs points (dim = 32)

Number of points	SeDuMi time (sec.)	SeDumi meas.	Heuristic method time (sec.)	Heur. method meas.	Sedumi time ratio	SeDuMi γ factor
1000	21.305	91.128	0.122	101.176	173.541	1.368
2000	49.592	92.083	0.187	99.923	265.142	1.277
5000	141.492	90.437	0.410	99.936	344.950	1.345
10000	332.365	92.401	0.776	99.318	427.785	1.241
20000	832.383	95.198	1.544	106.236	532.953	1.411
50000	2816.886	98.576	3.775	108.662	746.116	1.371
100000	12180.152	93.043	7.418	100.111	1641.977	1.247

The numerical simulation of the method shows that it outperformed SeDuMi in speed, but at the cost of volume redundancy. For $N = 10^5$ points and $d = 32$ the new method

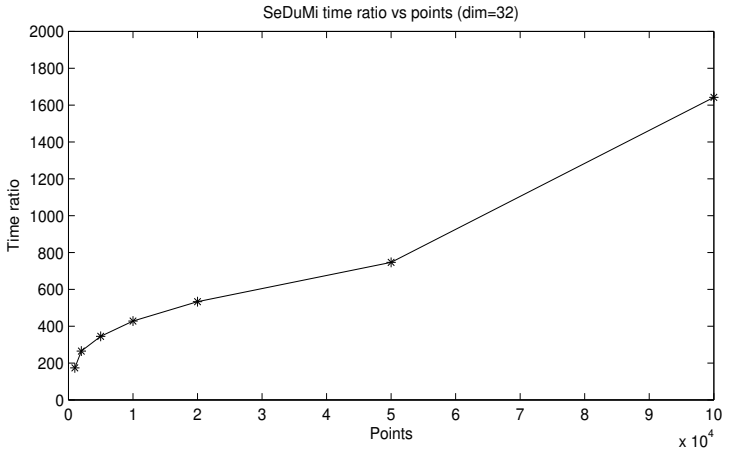

FIGURE 4.4: The ratio of the mean time of SeDuMi to the mean time of the developed Heuristic method vs points.

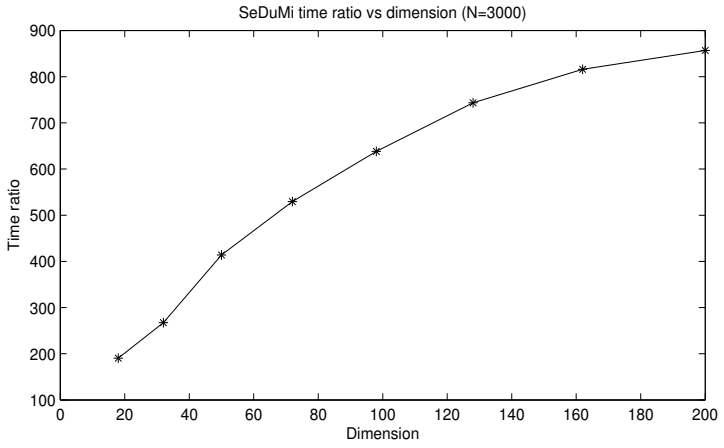

FIGURE 4.5: The ratio of the mean time of SeDuMi to the mean time of the developed Heuristic method vs dimension.

TIME & MEASURE VS DIMENSION FOR $N = 3000$ POINTS

n	Dim	SeDuMi time (sec.)	SeDuMi meas.	Heuristic method time(sec.)	Heur. method meas.	Sedumi time ratio	SeDuMi γ factor
3	18	41.781	49.365	0.219	54.627	190.372	1.339
4	32	75.629	95.284	0.282	103.279	267.381	1.283
5	50	158.393	150.621	0.382	162.507	413.888	1.268
6	72	254.429	221.111	0.480	248.506	529.537	1.463
7	98	412.109	311.797	0.645	341.200	638.212	1.349
8	128	621.550	417.427	0.835	462.891	743.548	1.426
9	162	946.934	528.711	1.160	569.284	815.922	1.284
10	200	1278.089	666.448	1.491	715.966	856.866	1.281

TABLE 4.1: Numerical data from the comparative tests.

was approximately 1600 times faster reducing the time of execution from over 3 hours to few seconds. For $d = 200$ and $N = 300$ the method was almost 900 times faster. The average uncertainty produced by the method was approximately 1.3 times greater that

the one given by SeDuMi.

For a system with clock synchronization errors the number of possible systems with such errors is given by (2.14) as $\sqrt{2}^n n!$ where n is the order of the model. For $n = 10$ and a system with an equal number of states and inputs, the problem of of estimating the uncertainty is defined by $N = 8.21 \times 10^7$ points and with dimension $d = 200$. Roughly taking the execution time of SeDuMi is 300 hours (100 times longer than the case when $N = 10^5$ and $d = 32$) and the execution time for the new method is 740 sec (100 times longer than the case when $N = 10^5$ and $d = 32$). Overall, the estimate reduces from 12.5 days to a few minutes.

4.3.6 Conclusions

The method developed for the construction of norm bounded uncertainty is fast and robust and allows for the solution of large scale problems. In the case of systems with clock synchronization errors this new method has no alternative and in other applications it can be used as a method for fast approximation of the uncertainty as opposed to the exact calculation, which is feasible only at the expense of an extremely long computation time and is not compatible with robust controller design, where iterations will be needed to arrive at an acceptable design for a given problem. Further research should concentrate on developing stability conditions for the uncertainty defined by the Frobenius norm.

Chapter 5

Estimation of synchronization errors in the common clock case

5.1 Introduction

The analysis in the previous two chapters assumed that the system model and synchronization errors were known. In many cases, however, the system may initially be running as implemented and then an error arises. This, in turn, poses the question of whether or not it is possible to detect errors from knowledge of the input and output, which is the subject of this chapter. The results given make use of the behavioral approach to linear systems theory and assume that the data is exact, i.e., noise free.

5.2 Preliminaries

Consider again the discrete-time linear system described by (2.1) with observability matrix

$$\boldsymbol{O}_k(\boldsymbol{A},\boldsymbol{C}) := \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \vdots \\ \boldsymbol{CA}^{k-1} \end{bmatrix} \tag{5.1}$$

and also the Toeplitz matrix

$$\boldsymbol{T}_k(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D}) := \begin{bmatrix} \boldsymbol{D} & 0 & \dots & 0 \\ \boldsymbol{CB} & \boldsymbol{D} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \boldsymbol{CA}^{k-2}\boldsymbol{B} & \dots & \boldsymbol{CB} & \boldsymbol{D} \end{bmatrix}. \tag{5.2}$$

For simplicity of notation we also introduce augmented output and input vectors

$$\boldsymbol{Y}_k := \begin{bmatrix} \boldsymbol{y}(1) \\ \boldsymbol{y}(2) \\ \vdots \\ \boldsymbol{y}(k) \end{bmatrix}, \qquad \boldsymbol{U}_k := \begin{bmatrix} \boldsymbol{u}(1) \\ \boldsymbol{u}(2) \\ \vdots \\ \boldsymbol{u}(k) \end{bmatrix}. \tag{5.3}$$

Let $\mathbb{T}$ be the time axis and $\mathbb{W}$ the signal space. In the behavioural approach Polderman and Willems (1998) a dynamical system is defined as a triple

$$\Sigma := (\mathbb{T}, \mathbb{W}, \mathcal{B}),$$

with the behaviour $\mathcal{B}$ which is a subset of $\mathbb{W}^{\mathbb{T}}$, where $\mathbb{W}^{\mathbb{T}}$ denotes the set of all maps from $\mathbb{T}$ to $\mathbb{W}$. Here $\mathbb{T} = \mathbb{N}$ and we partition $\mathbb{W}$ as $\mathbb{U} \times \mathbb{Y}$, where for $\boldsymbol{w} \in \mathbb{W}$ with corresponding partition $\boldsymbol{w} = (\boldsymbol{u}, \boldsymbol{y})$, $\boldsymbol{u}$ is an input and $\boldsymbol{y}$ is an output. The behaviour of the system (2.1) can be defined as

$$\mathcal{B}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}) := \left\{ (\boldsymbol{u}, \boldsymbol{y}) : \begin{array}{c} \text{there exists } \boldsymbol{x} \\ \text{such that (2.1) holds} \end{array} \right\}. \tag{5.4}$$

Note that the representation (2.1) is non-unique due to similarity transformations, i.e.,

$$\mathcal{B}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}) = \mathcal{B}(\boldsymbol{A}', \boldsymbol{B}', \boldsymbol{C}', \boldsymbol{D}'),$$

where

$$\boldsymbol{A}' = \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}, \quad \boldsymbol{B}' = \boldsymbol{T}^{-1}\boldsymbol{B}, \quad \boldsymbol{C}' = \boldsymbol{C}\boldsymbol{T}, \quad \boldsymbol{D}' = \boldsymbol{D} \tag{5.5}$$

for any invertible matrix $\boldsymbol{T} \in \mathbb{R}^{n \times n}$.

5.3 Estimation of synchronization errors

Assume that the system defined by (2.1) has clock synchronization errors. Then the system dynamics are of the form (2.8) where the matrices involved have unknown entries for some $s^\star \in \mathcal{S}$. Our purpose is to identify $s^\star$ from a given finite trajectory

$$(\boldsymbol{u}^\star, \boldsymbol{y}^\star)|_T := \big((\boldsymbol{u}^\star(1), \boldsymbol{y}^\star(1), \ldots, (\boldsymbol{u}^\star(T), \boldsymbol{y}^\star(T)\big).$$

The method of estimation is based on the fact that a solution $\boldsymbol{x}(1)$ of the system

$$\boldsymbol{Y}_T^\star = \boldsymbol{O}_T(\boldsymbol{A}_s, \boldsymbol{C})\boldsymbol{x}(1) + \boldsymbol{T}_T(\boldsymbol{A}_s, \boldsymbol{B}_s, \boldsymbol{C}, \boldsymbol{D})\boldsymbol{U}_T^\star, \tag{5.6}$$

where $\boldsymbol{Y}_T^\star$ and $\boldsymbol{U}_T^\star$ are built from the trajectory $(\boldsymbol{u}^\star, \boldsymbol{y}^\star)|_T$, exists for $T > n$ if and only if $(\boldsymbol{u}^\star, \boldsymbol{y}^\star)|_T$ is a trajectory of the system $\mathcal{B}(\boldsymbol{A}_s, \boldsymbol{B}_s, \boldsymbol{C}, \boldsymbol{D})$. Hence we must search over all

$s \in \mathcal{S}$ for a compatible system (5.6). which, in turn, gives us a concrete computational algorithm. The method is successful, however, if there is only one $\hat{s}$ such that a solution exists, in which case we may conclude that $s^\star = \hat{s}$. For such cases the synchronization error $s^\star$ is termed identifiable from the trajectory $(\boldsymbol{u}^\star, \boldsymbol{y}^\star)$.

Similar cases

Consider (2.8) and a sequence $s_1 = (i_1, i_2, \ldots, i_{d-1}, i_d)$ describing the switchings and the corresponding state matrix

$$\boldsymbol{A}_{s_1} = \boldsymbol{A}_{i_d}\boldsymbol{A}_{i_{d-1}} \cdots \boldsymbol{A}_{i_2}\boldsymbol{A}_{i_1}.$$

Assuming that all matrices $\boldsymbol{A}_{i_j},\ j = 1, \ldots, d$ are invertible, we have

$$\begin{aligned}\boldsymbol{A}_{i_d}^{-1}\boldsymbol{A}_{s_1}\boldsymbol{A}_{i_d} &= \boldsymbol{A}_{i_d}^{-1}\boldsymbol{A}_{i_d}\boldsymbol{A}_{i_{d-1}} \cdots \boldsymbol{A}_{i_2}\boldsymbol{A}_{i_1}\boldsymbol{A}_{i_d} \\ &= \boldsymbol{A}_{i_{d-1}}\boldsymbol{A}_{i_{d-2}} \cdots \boldsymbol{A}_{i_1}\boldsymbol{A}_{i_d} \\ &= \boldsymbol{A}_{s_2}.\end{aligned}$$

Hence the matrix $\boldsymbol{A}_{s_1}$ is similar to the matrix $\boldsymbol{A}_{s_2}$ corresponding to the sequence $s_2 = (i_d, i_1, \ldots, i_{d-2}, i_{d-1})$, i.e., s_1 shifted to the right by 1. For $\boldsymbol{A}_{s_2}$

$$\begin{aligned}\boldsymbol{A}_{i_{d-1}}^{-1}\boldsymbol{A}_{s_2}\boldsymbol{A}_{i_{d-1}} &= \boldsymbol{A}_{i_{d-1}}^{-1}\boldsymbol{A}_{i_{d-1}}\boldsymbol{A}_{i_{d-2}} \cdots \boldsymbol{A}_{i_1}\boldsymbol{A}_{i_d}\boldsymbol{A}_{i_{d-1}} \\ &= \boldsymbol{A}_{i_{d-2}}\boldsymbol{A}_{i_{d-3}} \cdots \boldsymbol{A}_{i_d}\boldsymbol{A}_{i_{d-1}} \\ &= \boldsymbol{A}_{s_3}\end{aligned}$$

and hence this matrix is similar to $\boldsymbol{A}_{s_3}$ corresponding to the sequence $s_3 = (i_{d-1}, i_d, \ldots, i_1, \ldots, i_{d-3}, i_{d-2})$, i.e., s_2 shifted right by one index. By repeating the procedure until the original sequence is obtained back we see that all the matrices corresponding to the shifted sequences are similar.

These similar cases are of special concern because the systems concerned may not be distinguishable from each other based on I/O data as the following example illustrates

Example 5.1. *Consider the system (2.1) with $n = m = 3$ and state space matrices*

$$\boldsymbol{A} = \begin{bmatrix} 0.1 & 0.2 & 0.4 \\ 0.2 & 0.1 & 0.3 \\ 0.3 & 0.1 & 0.2 \end{bmatrix}, \qquad \boldsymbol{B} = I_{3\times 3}$$

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{D} = \boldsymbol{0}_{1\times 3}$$

Consider three asynchronous cases described by the sequences 1) $s_1 = (\{1\}, \{2\}, \{3\})$, 2) $s_2 = (\{3\}, \{1\}, \{2\})$ and 3) $s_3 = (\{2\}, \{3\}, \{1\})$, respectively. The corresponding state

space model matrices are

$$1)\ \boldsymbol{A}_{s1} = \begin{bmatrix} 0.1 & 0.2 & 0.4 \\ 0.02 & 0.14 & 0.38 \\ 0.032 & 0.074 & 0.358 \end{bmatrix},\ \boldsymbol{B}_{s1} = \begin{bmatrix} 1 & 0 & 0 \\ 0.2 & 1 & 0 \\ 0.32 & 0.1 & 1 \end{bmatrix}$$

$$2)\ \boldsymbol{A}_{s2} = \begin{bmatrix} 0.22 & 0.24 & 0.08 \\ 0.134 & 0.178 & 0.076 \\ 0.3 & 0.1 & 0.2 \end{bmatrix},\ \boldsymbol{B}_{s2} = \begin{bmatrix} 1 & 0 & 0.4 \\ 0.2 & 1 & 0.38 \\ 0 & 0 & 1 \end{bmatrix}$$

$$3)\ \boldsymbol{A}_{s3} = \begin{bmatrix} 0.268 & 0.024 & 0.152 \\ 0.2 & 0.1 & 0.3 \\ 0.32 & 0.01 & 0.23 \end{bmatrix},\ \boldsymbol{B}_{s3} = \begin{bmatrix} 1 & 0.24 & 0.4 \\ 0 & 1 & 0 \\ 0 & 0.1 & 1 \end{bmatrix}$$

In this case, the initial conditions

$$1)\ \boldsymbol{x}_{0,1} = \begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix},\ 2)\ \boldsymbol{x}_{0,2} = \begin{bmatrix} 10 \\ 10 \\ 0 \end{bmatrix},\ 3)\ \boldsymbol{x}_{0,3} = \begin{bmatrix} 10 \\ 23.333 \\ 0 \end{bmatrix}$$

and the constant input $\boldsymbol{u}(k) = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$, $k = 1, 2, \ldots$ *correspond to the same output in all three cases. That means that the equation (5.6) holds for all three cases. Therefore these systems are not distinguishable from the trajectory* (u, y).

5.4 Identifiability of clock synchronization errors

Let $\mathcal{B}_s$ denote the behaviour generated by a system with clock synchronization s, i.e.

$$\mathcal{B}_s = \mathcal{B}(\boldsymbol{A}_s, \boldsymbol{B}_s, \boldsymbol{C}, \boldsymbol{D}).$$

A solution to (5.6) exists if and only if $(\boldsymbol{u}^\star, \boldsymbol{y}^\star)|_T$ belongs to $\mathcal{B}_s$, i.e.

$$\exists (\boldsymbol{u}, \boldsymbol{y}) \in \mathcal{B}_s, \quad (\boldsymbol{u}, \boldsymbol{y})|_T = (\boldsymbol{u}^\star, \boldsymbol{y}^\star)|_T, \tag{5.7}$$

where $w|_T$ denotes the restriction of $w \in \mathbb{W}^{\mathbb{N}}$ to the interval $[1, \ldots, T]$. Hence, there is only one $\hat{s} \in \mathcal{S}$ such that (5.6) has a solution in the following case:

1. There is only one system $\mathcal{B}$ with m inputs and order of n containing the trajectory $(\boldsymbol{u}^\star, \boldsymbol{y}^\star)$.
2. The map $s \mapsto \mathcal{B}_s$ is injective, i.e., $\forall s_1, s_2 \in \mathcal{S}$ $s_1 \neq s_2 \implies \mathcal{B}_{s_1} \neq \mathcal{B}_{s_2}$.

Conditions on the trajectory

As illustrated by Example 5.1, the trajectory $(\boldsymbol{u}^\star, \boldsymbol{y}^\star)|_T$ can belong to two different behaviours with m inputs and n states. In order to guarantee that there is only one such behaviour, conditions on the trajectory must be imposed. Following Willems et al. (2005) assume that

- the unknown data generating system is controllable,
- the trajectory $(\boldsymbol{u}^\star, \boldsymbol{y}^\star)|_T$ is exact (noise free),
- u is persistently exciting of order $k = n + l + 1$, where l is the order of the lag Willems (1991), i.e., the matrix

$$\mathcal{U} = \begin{bmatrix} \boldsymbol{u}^\star(1) & \boldsymbol{u}^\star(2) & \dots & \boldsymbol{u}^\star(T-k+1) \\ \boldsymbol{u}^\star(2) & \boldsymbol{u}^\star(3) & \dots & \boldsymbol{u}^\star(T-k+2) \\ \dots & \dots & \dots & \dots \\ \boldsymbol{u}^\star(k) & \boldsymbol{u}^\star(k+1) & \dots & \boldsymbol{u}^\star(T) \end{bmatrix} \tag{5.8}$$

  is full row rank.

Then the unknown system can be recovered from the trajectory $(\boldsymbol{u}^\star, \boldsymbol{y}^\star)$. In other words this condition is sufficient to guarantee that the obtained trajectory belongs to only one system with m inputs and n states. This eliminates the problem with similar cases.

Conditions for injectivity of the map $s \mapsto \mathcal{B}_s$

In order to determine the unknown clock synchronization error $s^\star$ we need the following condition: For all $s_1, s_2 \in \mathcal{S}$

$$s_1 \neq s_2 \quad \Longrightarrow \quad \mathcal{B}_{s_1} \neq \mathcal{B}_{s_2}. \tag{5.9}$$

Equivalently for all $s_1, s_2 \in \mathcal{S}$ such that $s_1 \neq s_2$ there is no invertible matrix $\boldsymbol{T} \in \mathbb{R}^{n\times n}$ such that

$$\boldsymbol{T}^{-1}\boldsymbol{A}_{s_1}\boldsymbol{T} = \boldsymbol{A}_{s_2}, \quad \boldsymbol{T}^{-1}\boldsymbol{B}_{s_1} = \boldsymbol{B}_{s_2}, \quad \boldsymbol{C}\boldsymbol{T} = \boldsymbol{C},$$

i.e.,

$$\boldsymbol{A}_{s_1}\boldsymbol{T} = \boldsymbol{T}\boldsymbol{A}_{s_2}, \quad \boldsymbol{B}_{s_1} = \boldsymbol{T}\boldsymbol{B}_{s_2}, \quad \boldsymbol{C}\boldsymbol{T} = \boldsymbol{C}. \tag{5.10}$$

The condition (5.10) is a linear system of equations in the unknown transformation matrix $\boldsymbol{T}$. In order to check injectivity of the map $s \mapsto \mathcal{B}_S$, however, $O(\mathrm{card}(S)^2)$ systems of this form have to be solved. This is impractical for realistic size problems, because $\mathrm{card}(S)$ for n order system can be approximated by $\sqrt{2}^n n!$. Therefore, the method involves searching over all $s \in \mathcal{S}$ for a solution of (5.6). However, we have a simple, but only necessary, condition for injectivity.

Assume that there is a zero coefficient in the state transition matrix of the system defined

by (2.1), i.e., $a_{pq} = 0$, for some $1 \leq p, q \leq n$. Let the sequence $s_1 = (i_1, \ldots, i_j, i_{j+1} \ldots, i_d)$, with $i_j = \{q\}$ and $i_{j+1} = \{p\}$, describe the switching event pattern. This yields the state space equations

$$\begin{aligned} \boldsymbol{x}^j(k) &= \boldsymbol{A}_{\{q\}}\boldsymbol{x}^{j-1}(k) + \boldsymbol{B}_{\{q\}}\boldsymbol{u}(k) \\ \boldsymbol{x}^{j+1}(k) &= \boldsymbol{A}_{\{p\}}\boldsymbol{x}^{j}(k) + \boldsymbol{B}_{\{p\}}\boldsymbol{u}(k) \end{aligned}$$

By substitution

$$\boldsymbol{x}^{j+1}(k) = \boldsymbol{A}_{\{p\}}\boldsymbol{A}_{\{q\}}\boldsymbol{x}^{j-1}(k) + (\boldsymbol{B}_{\{p\}} + \boldsymbol{A}_{\{p\}}\boldsymbol{B}_{\{q\}})\boldsymbol{u}(k) \tag{5.11}$$

and therefore, the new value of the pth entry is calculated after the calculation of the qth entry. But in the calculation of the pth entry the value of the qth entry is not used ($a_{pq} = 0$). Nothing changes if these variables update simultaneously in one event. This is a consequence of the facts that

$$\boldsymbol{A}_{\{p\}}\boldsymbol{A}_{\{q\}} = \boldsymbol{A}_{\{p,q\}}$$

and

$$\boldsymbol{A}_{\{p\}}\boldsymbol{B}_{\{q\}} = \boldsymbol{B}_{\{q\}} \Rightarrow (\boldsymbol{B}_{\{p\}} + \boldsymbol{A}_{\{p\}}\boldsymbol{B}_{\{q\}}) = \boldsymbol{B}_{\{p,q\}},$$

(which may be proved based on definitions (2.17) and (2.18) and the assumption $a_{pq} = 0$). The state transition (5.11) can be written as

$$\boldsymbol{x}^{j+1}(k) = \boldsymbol{A}_{\{p,q\}}\boldsymbol{x}^{j-1}(k) + \boldsymbol{B}_{\{p,q\}}\boldsymbol{u}(k).$$

If we consider the sequence of length $d - 1$

$$s_2 = (i_1, \ldots, i_{j-1}, i_j \cup i_{j+1}, i_{j+2}, \ldots, i_d),$$

then both s_1 and s_2 yield the same model. Therefore, if there is a zero entry in the system matrix, then the map $s \mapsto \mathcal{B}_s$ is not injective. By contraposition we obtain that nonzero entries in the system matrix is a necessary condition for the injectivity of the map $s \mapsto \mathcal{B}_s$.

5.5 Example

Consider the single-input single-output system

$$\begin{aligned} \boldsymbol{x}(k+1) &= \begin{bmatrix} 0.1 & 0.2 & 0.4 \\ 0.2 & 0.1 & 0.2 \\ 0.3 & 0.6 & 0.6 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 0.2 \\ 0.4 \\ 0.3 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \boldsymbol{x}(k) \end{aligned}$$

The state transition matrix has nonzero entries so the necessary condition is satisfied. Suppose also that the system has a synchronization error described by the sequence $s^\star = (\{1\}, \{2\}, \{3\})$, giving the state space equations

$$\begin{aligned} \boldsymbol{x}(k+1) &= \begin{bmatrix} 0.1 & 0.2 & 0.4 \\ 0.02 & 0.14 & 0.28 \\ 0.042 & 0.144 & 0.888 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 0.2 \\ 0.44 \\ 0.624 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \boldsymbol{x}(k) \end{aligned} \tag{5.12}$$

We aim to estimate the unknown $s^\star$ by observing a trajectory of (5.12). For the original system there are 13 different sequences so there are 13 candidate systems, all of which are observable and controllable. By the identifiability condition of Willems (2007), the number of observations should be $T \geq 4n+1 = 13$. Hence we apply to the system (5.12) the input

$$\boldsymbol{U}_5 = [\ 1,\ \ 2,\ \ 3,\ \ 4,\ \ 5,\ \ 6,\ \ 7,\ \ 8,\ \ 7,\ \ 8,\ \ 7,\ \ 6,\ \ 5\ ]^{\mathrm{T}},$$

in order to obtain 13 observations of the output assuming zero initial conditions. Then by solving (5.6), we have that a solution exists only for the system represented by the sequence

$$s_3 = (\{1\}, \{2\}, \{3\}).$$

Hence the synchronization error are correctly estimated.

5.6 Conclusions

This chapter has developed a method to identify synchronization error from input and output of an asynchronously operating system that arises from propagation delays in the operation of systems designed to operate under synchronization. The new algorithm can be easily implemented using existing software packages. Future research should aim to remove the assumption (1) and give conditions on the inputs that guarantee identifiability of the unknown system. Also, the more realistic case when the data are noisy should be treated.

Chapter 6

Representative applications

6.1 Introduction

As discussed earlier in this thesis clock synchronization errors can arise in large scale systems, high speed circuitry, economical markets, biological models and many other areas. This thesis has developed computationally efficient approaches to stability and control of such systems drawing on methods from robust control theory and related areas. This chapter considers the first application of these results to three problem areas.

6.2 Multi-agent systems

Consider a system consisting of multiple agents acting as a swarm. In particular, suppose that there are M agents with linear dynamics where the dynamics of agent i is governed by the state space model

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_i (n+1) = \begin{bmatrix} \delta_i & -\omega_i \\ \omega_i & \delta_i \end{bmatrix} \cdot \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_i (n) - \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_i (n) \right),$$

where $i = 1, 2, \ldots M$. The process input $[\, u_1 \quad u_2]_i^T$ is defined as

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} (n) = \frac{1}{M} \left(\sum_{i=1}^{M} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_i (n) \right) + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}_i (n),$$

where $[\, e_1,\, e_2\,]_i^T$, $i = 1, \ldots, M$ denotes an independent input to each agent. It is also assumed that

$$\delta_i^2 + \omega_i^2 = 1 - \varepsilon,$$

where $\varepsilon > 0$ models the contractive behaviour of the agent's movement. Working together the agents are required to meet at a specified point by executing rotation around

and contraction towards the center. Note that if $\varepsilon = 0$ then no contraction takes place. The structure here is illustrated in Figure 6.1

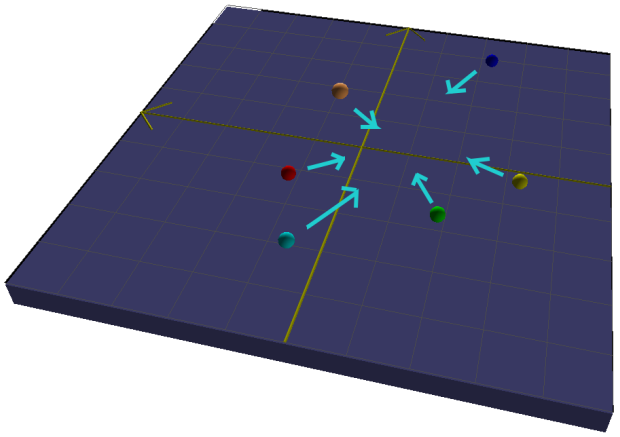

FIGURE 6.1: The dynamics of each agent consist of rotation and contraction. Agents aim to meet at one point which they calculate separately.

Introducing the vectors $\boldsymbol{x'} \in \mathcal{R}^{2M}$ and $\boldsymbol{u'} \in \mathcal{R}^{2M}$ as

$$x'_j = \begin{cases} (x_1)_{\frac{j+1}{2}} & \text{if j is odd} \\ (x_2)_{\frac{j}{2}} & \text{if j is even} \end{cases} \qquad j = 1, 2, \ldots 2M$$

$$u'_j = \begin{cases} (e_1)_{\frac{j+1}{2}} & \text{if j is odd} \\ (e_2)_{\frac{j}{2}} & \text{if j is even} \end{cases} \qquad j = 1, 2, \ldots 2M$$

enables the system model to be written as

$$\boldsymbol{x'}(n) = \boldsymbol{A}_1 \boldsymbol{x'}(n) - \boldsymbol{A}_1 \cdot \left(\boldsymbol{A}_2 \boldsymbol{x'}(n) + \boldsymbol{u} \right),$$

or

$$\boldsymbol{x'}(n) = \boldsymbol{A}_1 \left(\boldsymbol{I} - \boldsymbol{A}_2 \right) \boldsymbol{x'}(n) - \boldsymbol{A}_1 \boldsymbol{u'},$$

where

$$\boldsymbol{A}_1 = \begin{bmatrix} \delta_1 & -\omega_1 & 0 & \ldots & \ldots & 0 \\ \omega_1 & \delta_1 & 0 & \ldots & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \delta_i & -\omega_i & \ldots & 0 \\ 0 & \ldots & \omega_i & \delta_i & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & 0 & \delta_M & -\omega_M \\ 0 & \ldots & \ldots & 0 & \omega_M & \delta_M \end{bmatrix} \tag{6.1}$$

and

$$\boldsymbol{A}_2 = \begin{bmatrix} \frac{1}{M} & 0 & \dots & \dots & \frac{1}{M} & 0 \\ 0 & \frac{1}{M} & \dots & \dots & 0 & \frac{1}{M} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{1}{M} & 0 & \dots & \dots & \frac{1}{M} & 0 \\ 0 & \frac{1}{M} & \dots & \dots & 0 & \frac{1}{M} \end{bmatrix}. \tag{6.2}$$

Introducing

$$\boldsymbol{A}' = \boldsymbol{A}_1 \cdot (\boldsymbol{I} - \boldsymbol{A}_2) \quad \boldsymbol{B}' = -\boldsymbol{A}_1, \tag{6.3}$$

yields the state-space model

$$\boldsymbol{x'}(n+1) = \boldsymbol{A'x'}(n) + \boldsymbol{B'u'}. \tag{6.4}$$

In order to deal with synchronization errors we assume that agents' clocks are out of phase but they have the same period T. Then we can use the model of synchronization errors with only minor modifications. We consider only those product matrices that are relevant to the agent's work , i.e. we assume that pairs x_i, x_{i+1}, $i = 1, 3, 5, \dots, 2M-1$. work synchronously. Then we apply the algorithm to find an admissible controller.

Suppose that the parameters in the model here vary as follows

$$\epsilon \in [0, 1] \qquad \omega_i \in [0, \sqrt{1-\epsilon}] \qquad \delta_i^2 + \omega_i^2 = 1 - \epsilon.$$

Then in Figure 6.2 synchronization errors occur in the dark grey region, i.e. some product matrices are unstable. The stable region is marked in light grey. Figure 6.3 illustrates a case of agents movement in the presence of a synchronization error and with no control applied. Figure 6.4 illustrates the movement in the presence of a synchronization error and with control applied.

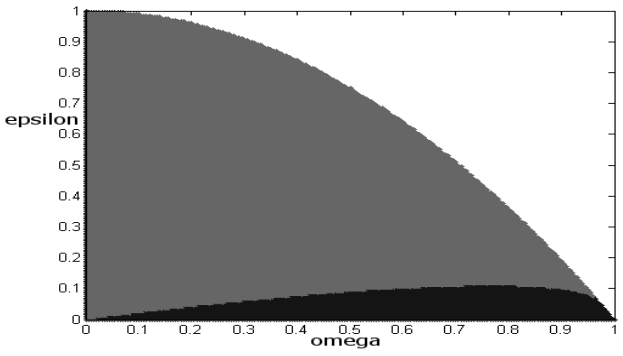

FIGURE 6.2: Unstable region; dark grey, stable region: light grey.

Consider the above system for $M = 6$ with

$$\delta_i = \delta_j \qquad \omega_i = \omega_j \qquad i, j = 1, 2, \ldots, 6,$$

where $\varepsilon = 0.1819$, $\omega_1 = 0.9$, $\delta_1 = 0.09$ and the system is controllable. If we assume that each agent works synchronously and that synchronization errors only arise when agents are performing common tasks, we have 4683 product matrices and some of them are unstable.

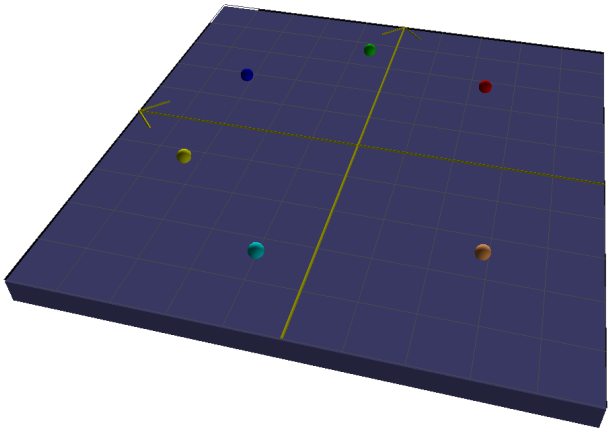

FIGURE 6.3: Synchronization error with no control. The agents calculate the center at different time instances and move towards different targets. In an extreme case they can move away the center.

Using Algorithm 3.3.6 we can find a control law to guarantee that the closed-loop system is stable independent of the synchronization errors. The computation time is 500 sec as compared to 640 sec for direct computation and this advantage will increase with a growing number of agents. The 500 sec computation arose from use of Khachiyan's algorithm for the computation of MVEE. This computation time is substantially reduced if we use the new Algorithm 3.3.5, see Table 3.1.

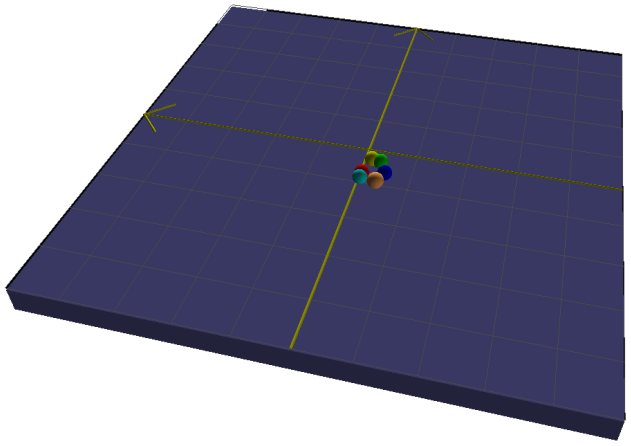

FIGURE 6.4: Synchronization error with control applied. This control forces the agents to meet at one point independent of synchronization error.

6.3 Application to iterative learning control with synchronization errors

Iterative learning control (ILC) is especially suited to controlling systems operating in a repetitive, or trial-to-trial, mode with the requirement that a reference trajectory $y_{ref}(p)$ defined over a finite interval $0 \leq p \leq \alpha$, where α denotes the trial duration, is followed to a high precision. Examples of such systems include robotic manipulators that are required to repeat a given task to high precision, chemical batch processes or, more generally, the class of tracking systems. Since the original work, the general area of ILC has been the subject of intense research effort. Initial sources for the literature here are the survey papers Bristow et al. (2006) , Ahn et al. (2007). These papers demonstrate that substantial progress has been made with many designs having seen at least experimental benchmarking and there has also been extension to a novel health-care application supported by clinical trial results Freeman et al. (2009); A.-M.Hughes et al. (2009).

Given that ILC has been a successfully implemented learning strategy for one dynamic system, leads to the question: If there are multiple similar agents, such as a fleet of robots, is it possible to benefit from exchanging information during the learning process? This question has recently received attention; see, for example, Schoellig et al. (2010)

The remainder of this chapter gives the first results on ILC design in the presence of synchronization errors. ILC updates the input in the iterative manner operating on some underlying system. In particular, it is assumed that the system can have a synchronization error. The important question arises about robustness of ILC control strategy against synchronization errors and whether the developed method can be applied in the case when it is not.

The analogy between Repetitive Processes and ILC

In ILC, a major objective is to achieve convergence of the trial-to-trial error. It is, however, possible that enforcing fast convergence could lead to unsatisfactory performance along the trial, and here this problem is addressed by first showing that ILC schemes can be designed for a class of discrete linear systems by extending techniques developed for linear repetitive processes Rogers et al. (2007). This allows us to use the strong concept of stability along the pass (or trial) for these processes, in an ILC setting, as a possible means of dealing with poor/unacceptable transients in the dynamics produced along the trials.

The system to be controlled is initially assumed to be adequately modeled by linear time-invariant dynamics that, after sampling, can be represented by a controllable and observable discrete linear state-space model defined by the triple $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}$. In an ILC setting this is written as

$$\begin{aligned} \boldsymbol{x}_k(p+1) &= \boldsymbol{A}\boldsymbol{x}_k(p) + \boldsymbol{B}\boldsymbol{u}_k(p), \;\; 0 \leq p \leq \alpha - 1 \\ \boldsymbol{y}_k(p) &= \boldsymbol{C}\boldsymbol{x}_k(p), \end{aligned} \tag{6.5}$$

where on trial k, $\boldsymbol{x}_k(p) \in \mathbb{R}^n$ is the state vector, $\boldsymbol{y}_k(p) \in \mathbb{R}^m$ is the output vector, $\boldsymbol{u}_k(p) \in \mathbb{R}^r$ is the vector of control inputs, and the trial duration $\alpha < \infty$. If the signal to be tracked is denoted by $\boldsymbol{y}_{ref}(p)$ then $\boldsymbol{e}_k(p) = \boldsymbol{y}_{ref}(p) - \boldsymbol{y}_k(p)$ is the error on trial k, and the most basic requirement is to force the error to converge in k. In particular, the objective of constructing a sequence of input functions such that the performance is gradually improving with each successive trial can be refined to a convergence condition on the input and error

$$\lim_{k\to\infty} ||\boldsymbol{e}_k|| = 0, \;\; \lim_{k\to\infty} ||\boldsymbol{u}_k - \boldsymbol{u}_\infty|| = 0,$$

where $||\cdot||$ is a signal norm in a suitably chosen function space with a norm-based topology.

Trial-to-trial error convergence does not require that (6.5) is stable since, for example, it is easily shown that an update law of the form $\boldsymbol{u}_{k+1}(p) = \boldsymbol{u}_k(p) + \boldsymbol{L}\boldsymbol{e}_k(p+1)$, where $\boldsymbol{L}$ is an $r \times m$ matrix to be designed, gives this property provided $\rho(\boldsymbol{I} - \boldsymbol{C}\boldsymbol{B}\boldsymbol{L}) < 1$. The reason for this is the finite trial duration over which even an unstable linear system can only produce a bounded output. For cases where $\rho(\boldsymbol{A}) \geq 1$, this allows for the production of large errors for small values of k and/or large values of α. Even if $\rho(\boldsymbol{A}) < 1$ there could be unacceptably large oscillations in the dynamics produced along the early trials for many practical applications, such as a gantry robot whose task is to collect an object from a location, place it on a moving conveyor, and then return for the next one and so on. If, for example, the object has an open top and is filled with liquid, and/or is fragile in nature, then unwanted vibrations during the transfer time could have very detrimental

effects. In such cases there is also a need to control the along-the-trial dynamics.

If the plant is unstable, that is, $\rho(\boldsymbol{A}) \geq 1$, or the along-the-trial dynamics are insufficiently damped, one option is to first design a stabilizing feedback control scheme for the plant and then apply ILC to the resulting controlled dynamics. This chapter follows an alternative route by using a 2D system setting for analysis and control law design. Moreover, the repetitive process setting guarantees monotonic trial-to-trial error convergence. The two directions of information propagation in the 2D system setting are from trial-to-trial, indexed by k, and along-the-trial, indexed by p. There has already been work in this setting using the well known Roesser Roesser (1975) and Fornasini-Marchesini Fornasini and Marchesini (1978) state space models. For example, in Kurek and Zaremba (1993) it was shown how trial-to-trial error convergence of linear ILC schemes in the discrete domain could be examined as a stability problem in terms of a Roesser state space model interpretation of the dynamics. In recent work Hladowski et al. (2008, 2012) a repetitive process setting has been used to produce control laws for trial-to-trial error convergence and control of the along-the-trial dynamics and the resulting designs have been experimentally verified. The results in this section extend this approach to the case of synchronization errors.

The unique characteristic of a repetitive process Rogers et al. (2007) is a series of sweeps, termed passes, through a set of dynamics defined over a fixed finite duration known as the pass length. On each pass an output, termed the pass profile, is produced which acts as a forcing function on, and hence contributes to, the dynamics of the next pass profile, or trial output in ILC. This, in turn, leads to the unique control problem where the output sequence of pass profiles generated can contain oscillations that increase in amplitude in the pass-to-pass direction.

Attempts to control repetitive processes using standard systems theory and algorithms fail precisely because such an approach ignores their inherent 2D systems structure, that is, information propagation occurs from pass-to-pass (k) and along a given pass (p), and also the initial conditions are reset before the start of each new pass. To remove these deficiencies, a rigorous stability theory has been developed Rogers et al. (2007) based on an abstract model of the dynamics in a Banach space setting that includes a very large class of processes with linear dynamics and a constant pass length as special cases, including those described by (6.7) below.

In terms of their dynamics, it is the pass-to-pass coupling, noting again their unique feature, which is critical. The process dynamics in this setting are described by

$$\boldsymbol{y}_{k+1} = \boldsymbol{L}_\alpha \boldsymbol{y}_k + \boldsymbol{b}_{k+1},\ k \geq 0, \tag{6.6}$$

where $\boldsymbol{y}_k \in E_\alpha$, E_α is a Banach space with norm $||\cdot||$, $\boldsymbol{L}_\alpha$ is a bounded linear operator mapping E_α into itself, $\boldsymbol{b}_{k+1} \in W_\alpha$, W_α is a linear subspace of E_α, and the norm of E_α is denoted by $||\cdot||$. The term $\boldsymbol{L}_\alpha \boldsymbol{y}_k$ represents the contribution from pass k to pass

$k+1$ and $\boldsymbol{b}_{k+1}$ represents terms that enter on pass $k+1$, that is, state initial conditions, control inputs and disturbances.

Consider discrete linear repetitive processes described by the following state space model over $0 \leq p \leq \alpha - 1, k \geq 1$,

$$\begin{aligned} \boldsymbol{x}_k(p+1) &= \boldsymbol{A}_d\boldsymbol{x}_k(p) + \boldsymbol{B}_d\boldsymbol{u}_k(p) + \boldsymbol{B}_{d0}\boldsymbol{y}_{k-1}(p) \\ \boldsymbol{y}_k(p) &= \boldsymbol{C}_d\boldsymbol{x}_k(p) + \boldsymbol{D}_d\boldsymbol{u}_k(p) + \boldsymbol{D}_{d0}\boldsymbol{y}_{k-1}(p), \end{aligned} \tag{6.7}$$

where on pass k, $\boldsymbol{x}_k(p) \in \mathbb{R}^n$ is the state vector, $\boldsymbol{y}_k(p) \in \mathbb{R}^m$ is the pass profile vector and $\boldsymbol{u}_k(p) \in \mathbb{R}^r$ is the control input vector. To complete the process description, it is necessary to specify the initial, or boundary, conditions, that is, the state initial vector on each pass and the initial pass profile. Here these are taken to be of the form $\boldsymbol{x}_{k+1} = \boldsymbol{d}_{k+1}$, $k \geq 0$, and $\boldsymbol{y}_0(p) = \boldsymbol{f}(p)$, respectively, where the entries in the $n \times 1$ vector $\boldsymbol{d}_{k+1}$ are known constants and those in the $m \times 1$ vector $\boldsymbol{f}(p)$ are known functions of p over the pass duration.

The basic premise in ILC is to improve performance by directly adjusting the input used on each new trial, and often this is expressed in the form

$$\boldsymbol{u}_{k+1}(p) = \boldsymbol{u}_k(p) + \Delta\boldsymbol{u}_{k+1}(p),\ k \geq 0. \tag{6.8}$$

Hence the problem is to develop an algorithm to select the adjustment $\Delta\boldsymbol{u}_{k+1}(p)$ to be added to the input $\boldsymbol{u}_k(p)$ used on the previous trial and thereby construct the current trial input. In this paper, the approach used for the forms of $\Delta\boldsymbol{u}_{k+1}(p)$ considered is to first show that the resulting controlled dynamics can be described by a discrete linear repetitive process state space model of the form (6.7) and then apply the stability theory to derive the corresponding control law design algorithm.

Introduce, for analysis purposes only the following vector for (6.5)

$$\boldsymbol{\eta}_{k+1}(p+1) = \boldsymbol{x}_{k+1}(p) - \boldsymbol{x}_k(p) \tag{6.9}$$

and select $\Delta\boldsymbol{u}_{k+1}(p)$ as

$$\Delta\boldsymbol{u}_{k+1}(p) = \boldsymbol{K}_1\boldsymbol{\eta}_{k+1}(p+1) + \boldsymbol{K}_2\boldsymbol{e}_k(p+1) \tag{6.10}$$

and hence

$$\boldsymbol{\eta}_{k+1}(p+1) = (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}_1)\boldsymbol{\eta}_{k+1}(p) + \boldsymbol{B}\boldsymbol{K}_2\boldsymbol{e}_k(p). \tag{6.11}$$

Also

$$\begin{aligned} \boldsymbol{e}_{k+1}(p) - \boldsymbol{e}_k(p) &= \boldsymbol{C}\boldsymbol{A}\left(\boldsymbol{x}_k(p-1) - \boldsymbol{x}_{k+1}(p-1)\right) + \boldsymbol{C}\boldsymbol{B}(\boldsymbol{u}_k(p-1) - \boldsymbol{u}_{k+1}(p-1)) \\ &= -\boldsymbol{C}\boldsymbol{A}\boldsymbol{\eta}_{k+1}(p) - \boldsymbol{C}\boldsymbol{B}\Delta\boldsymbol{u}_{k+1}(p-1). \end{aligned}$$

Using (6.10) gives

$$\boldsymbol{e}_{k+1}(p) = -\boldsymbol{C}(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}_1)\boldsymbol{\eta}_{k+1}(p) + (\boldsymbol{I}-\boldsymbol{C}\boldsymbol{B}\boldsymbol{K}_2)\boldsymbol{e}_k(p). \tag{6.12}$$

Introduce

$$\begin{aligned}\hat{\boldsymbol{A}} &= \boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}_1\\ \hat{\boldsymbol{B}}_0 &= \boldsymbol{B}\boldsymbol{K}_2\\ \hat{\boldsymbol{C}} &= -\boldsymbol{C}(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}_1)\\ \hat{\boldsymbol{D}}_0 &= \boldsymbol{I}-\boldsymbol{C}\boldsymbol{B}\boldsymbol{K}_2\end{aligned}$$

Then (6.11) and (6.12) can be written as

$$\begin{aligned}\boldsymbol{\eta}_{k+1}(p+1) &= \hat{\boldsymbol{A}}\boldsymbol{\eta}_{k+1}(p) + \hat{\boldsymbol{B}}_0\boldsymbol{e}_k(p)\\ \boldsymbol{e}_{k+1}(p) &= \hat{\boldsymbol{C}}\boldsymbol{\eta}_{k+1}(p) + \hat{\boldsymbol{D}}_0\boldsymbol{e}_k(p),\end{aligned} \tag{6.13}$$

which is of the form (6.7) and hence the repetitive process stability theory can be applied in the ILC case.

For the discrete linear repetitive processes considered, there are a wide range of stability along the trial tests, such as the following in Rogers et al. (2007)

Theorem 6.1. *A discrete linear repetitive process described by* (6.7) *is stable along the trial if and only if*

- $\rho(\boldsymbol{D}_{d0}) < 1$
- $\rho(\boldsymbol{A}_d) < 1$
- *all eigenvalues of* $\boldsymbol{G}(z) = \boldsymbol{C}_d(z\boldsymbol{I}-\boldsymbol{A}_d)^{-1}\boldsymbol{B}_{d0}+\boldsymbol{D}_{d0}$, $\forall|z|=1$ *have modulus strictly less than unity*

One approach to control law design for systems described by (6.7) is to use a Lyapunov function approach which is detailed next.

Consider a Lyapunov function of the form

$$V(k,p) = V_1(k,p) + V_2(k,p),$$

with

$$\begin{aligned}V_1(k,p) &= \boldsymbol{x}_{k+1}\boldsymbol{P}\boldsymbol{x}_{k+1}(p)\\ V_2(k,p) &= \boldsymbol{y}_k\boldsymbol{P}\boldsymbol{y}_k(p),\end{aligned}$$

where $\boldsymbol{P}_1 \succ \boldsymbol{0}$, $i = 1, 2$ with associated increment

$$\Delta V(k,p) = V_1(k,p+1) - V_1(k,p) + V_2(k+1,p) - V_2(k,p).$$

Then stability along the trial holds if $\Delta V(k,p) < 0$ for all k and p which is equivalent to the requirement that

$$\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{\Phi} - \boldsymbol{P} \prec \mathbf{0}, \qquad \boldsymbol{P} = \mathrm{diag}(\boldsymbol{P}_1, \boldsymbol{P}_2), \tag{6.14}$$

where

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{A}_d & \boldsymbol{B}_{d0} \\ \boldsymbol{C}_d & \boldsymbol{D}_{d0} \end{bmatrix}. \tag{6.15}$$

This last condition is the $2D$ Lyapunov equation characterization of stability and even though it has and identical structure to that for standard linear systems it is sufficient only. If, however, the example considered is single input single output this equation is necessary and sufficient for stability.

Theorem 6.2. *Hladowski et al. (2008) The ILC scheme (6.13) is stable along the trial if there exist matrices $\boldsymbol{X}_1 \succ \mathbf{0}$, $\boldsymbol{X}_2 \succ \mathbf{0}$, $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ such that the following LMI is feasible*

$$\boldsymbol{\Psi} = \begin{bmatrix} -\boldsymbol{X}_1 & \mathbf{0} & \boldsymbol{X}_1\boldsymbol{A}^T + \boldsymbol{R}_1^T\boldsymbol{B}^T & -\boldsymbol{X}_1\boldsymbol{A}^T\boldsymbol{C}^T - \boldsymbol{R}_1^T\boldsymbol{B}^T\boldsymbol{C}^T \\ \mathbf{0} & -\boldsymbol{X}_2 & \boldsymbol{R}_2^T\boldsymbol{B}^T & \boldsymbol{X}_2 - \boldsymbol{R}_2^T\boldsymbol{B}^T\boldsymbol{C}^T \\ \boldsymbol{A}\boldsymbol{X}_1 + \boldsymbol{B}\boldsymbol{R}_1 & \boldsymbol{R}_2 & -\boldsymbol{X}_1 & \mathbf{0} \\ -\boldsymbol{C}\boldsymbol{A}\boldsymbol{X}_1 - \boldsymbol{C}\boldsymbol{B}\boldsymbol{R}_1 & \boldsymbol{X}_2 - \boldsymbol{C}\boldsymbol{B}\boldsymbol{R}_2 & \mathbf{0} & -\boldsymbol{X}_2 \end{bmatrix} \prec \mathbf{0}. \tag{6.16}$$

If (6.16) holds, stabilizing control law matrices are given by

$$\boldsymbol{K}_1 = \boldsymbol{R}_1\boldsymbol{X}_1^{-1}, \quad \boldsymbol{K}_2 = \boldsymbol{R}_2\boldsymbol{X}_2^{-1}.$$

Iterative learning control in the presence of uncertainty in the plant dynamics proceeds by assuming that the uncertainty is described by a particular structure. Theorem A.1 gives LMI based conditions for control law design in the case when the plant considered has uncertainty of a polytopic type.

Example 6.1. *Consider the case of (6.5) when*

$$\boldsymbol{A} = \begin{bmatrix} 0.9672 & -0.8006 & 0.7381 & -0.2942 & 1.0797 \\ 0.2276 & 0.7791 & 0.2604 & -0.7895 & -0.8344 \\ -0.4045 & 1.0786 & 0.4765 & 0.7878 & 0.8748 \\ -0.0516 & 0.2259 & -0.9643 & 0.8008 & 0.6371 \\ 0.2298 & -1.1144 & -0.1776 & -0.1144 & -0.1320 \end{bmatrix}, \quad \boldsymbol{B} = \boldsymbol{I}_{5\times 5},$$

where $\rho(\boldsymbol{A}) = 1.71$ *and* $\boldsymbol{I}$ *denotes the identity matrix. In this case (6.14) is satisfied for the matrix* $\boldsymbol{P} = diag(\boldsymbol{P}_1, \boldsymbol{P}_2)$ *where*

$$\boldsymbol{P}_1 = \begin{bmatrix} 0.2027 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.1870 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.2713 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.1928 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2350 \end{bmatrix}$$

$$\boldsymbol{P}_2 = \begin{bmatrix} 0.2655 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.2575 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.2726 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.2665 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2638 \end{bmatrix} \tag{6.17}$$

and control law matrices

$$\boldsymbol{K}_1 = \begin{bmatrix} -1.1672 & 0.8006 & -0.7381 & 0.2942 & -1.0797 \\ -0.2276 & -0.7791 & 0.2604 & 0.7895 & 0.8344 \\ 0.4045 & -1.0786 & -0.4765 & -0.7878 & -0.8748 \\ 0.0516 & -0.2259 & 0.9643 & -0.8008 & -0.6371 \\ -0.2298 & 1.1144 & 0.1776 & 0.1144 & 0.5320 \end{bmatrix}$$

$$\boldsymbol{K}_2 = \begin{bmatrix} 0.5231 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.5231 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.5231 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.5231 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.5231 \end{bmatrix}. \tag{6.18}$$

Consider now the case with a common clock when there are 541 *possible synchronization errors (calculated by Algorithm 2.1.1). Computing the spectral radii of the matrices in this case shows that all systems generated by the clock synchronization errors are not stable. For* 225 *systems with errors and control law matrices (6.18) the spectral radius of the matrix* Φ *(6.15) exceeds one* ($\rho(\Phi) > 1$) *which prevents the existence of a matrix* $\boldsymbol{P}$ *satisfying (6.14). and hence there is no guarantee that the ILC scheme* (6.13) *with synchronization error is stable.*

For 7 *systems with synchronization errors the second condition of Theorem 6.1 is not satisfied and hence the ILC scheme is unstable. Using Algorithm 3.3.6 and solving the set of LMIs (A.1) for the vertices of the polytope gives the following stabilizing control*

law matrices

$$\boldsymbol{K}_1 = \begin{bmatrix} -0.0259 & 0.7949 & -0.7385 & 0.2899 & -1.0795 \\ -0.2274 & 0.2130 & -0.2614 & 0.7893 & 0.8365 \\ 0.4007 & -1.0783 & 0.5150 & -0.7874 & -0.8727 \\ 0.0473 & -0.2254 & 0.9600 & 0.1851 & -0.6340 \\ -0.2296 & 1.1158 & 0.1796 & 0.1139 & 1.1231 \end{bmatrix}$$

$$\boldsymbol{K}_2 = 10^{-3} \cdot \begin{bmatrix} 0.1390 & 0.0130 & -0.0005 & 0.0105 & -0.0001 \\ -0.0001 & 0.0167 & 0.0027 & 0.0005 & -0.0048 \\ 0.0091 & -0.0001 & 0.0191 & -0.0008 & -0.0046 \\ 0.0105 & -0.0011 & 0.0100 & 0.0304 & -0.0065 \\ -0.0009 & -0.0031 & -0.0039 & 0.0009 & 0.0189 \end{bmatrix}. \tag{6.19}$$

These control law matrices stabilize the system (6.5) for all synchronization errors, but further numerical investigation establishes that the trial-to-trial error convergence is very slow.

Solving the set of LMIs (A.1) for all matrices representing synchronization errors gives the control law matrices

$$\boldsymbol{K}_1 = \begin{bmatrix} -0.9672 & 0.8006 & -0.7381 & 0.2942 & -1.0797 \\ -0.2276 & -0.7791 & -0.2604 & 0.7895 & 0.8344 \\ 0.4045 & -1.0786 & -0.4765 & -0.7878 & -0.8748 \\ 0.0516 & -0.2259 & 0.9643 & -0.8008 & -0.6371 \\ -0.2298 & 1.1144 & 0.1776 & 0.1144 & 0.1320 \end{bmatrix}$$

$$\boldsymbol{K}_2 = \begin{bmatrix} 0.5231 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.5231 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.5231 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.5231 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.5231 \end{bmatrix}, \tag{6.20}$$

which results in much better performance. The volume redundancy arising in the polytope results in control law matrices (6.19) with worse performance and prevents a solver from finding a better solution for systems with synchronization errors, where (6.20) demonstrates that such a solution exists for this plant.

If the norm bounded uncertainty description is used Theorem A.3 is the required background result.

Example 6.2. *Consider again the plant of Example 6.1 and use the norm bounded uncertainty to construct the set containing all matrices representing systems with clock synchronization errors by employing Algorithms 3.3.5 and 4.3.1. In this case attempting to find control law matrices using the LMIs of Theorem A.3 failed, which implies that the*

volume redundancy in this case prevents the computation of a solution for all systems with errors, where again (6.20) demonstrates that such a solution exists.

6.4 Application to solving specific LMI control problems

LMIs are now a standard computational tool in many areas of control, signal processing and other areas. Their importance in control theory has been greatly influenced by the work of Yakubovich (1962, 1964, 1967). With the development of the efficient interior point methods Nesterov and Nemirovski (1988, 1994) LMIs begun to be widely used in control systems design and implementation, see for example Boyd et al. (1994). Many currently LMI solvers implement interior point methods such as *SeDuMi* Sturm (1999) which uses *the primal-dual interior point method* Sturm (1997) algorithm or *MATLAB LMILab* that uses *the projective method* Nemirovski and Gahinet (1994), a variant of interior point method. This section gives some initial result on the application of the new algorithms developed in Chapters 3 and 4 to specific LMIs that arise in control theory.

The results of Chapter 4 of the estimation of the norm bounded uncertainty provides a suboptimal solution to the problem

$$\begin{array}{ll} \text{minimize}_{\text{over}\boldsymbol{A},\boldsymbol{B},\boldsymbol{H},\boldsymbol{E}} & \mu(\{\boldsymbol{H}\boldsymbol{F}\boldsymbol{E} : \boldsymbol{F}^T\boldsymbol{F} \preceq \boldsymbol{I}\}) \\ \text{subject to} & \boldsymbol{X}_k = \boldsymbol{X} + \boldsymbol{H}\boldsymbol{F}_k\boldsymbol{E}, \quad \boldsymbol{F}_k^T\boldsymbol{F}_k \preceq \boldsymbol{I}, \quad k = 1, \ldots, N, \end{array}$$

where $\mu(\cdot)$ is the measure of the uncertainty under the vec$(\cdot)$ operation. The suboptimal solution is obtained by finding the MVEE of specific structure, i.e., by solving the problem

$$\begin{array}{ll} \text{minimize}_{\text{over}\boldsymbol{A},\boldsymbol{B},\boldsymbol{H},\boldsymbol{E}} & \text{vol}(\mathcal{E}(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}, \text{vec}([\boldsymbol{A},\ \boldsymbol{B}])) \\ \text{subject to} & \text{vec}(\boldsymbol{X}_k) \in \mathcal{E}(\boldsymbol{E}^{\mathrm{T}} \otimes \boldsymbol{H}, \text{vec}([\boldsymbol{A},\ \boldsymbol{B}]), \quad k = 1, \ldots, N. \end{array}$$

This solution is also a suboptimal and a very close approximation to the solution of the LMI problem

$$\begin{array}{lll} \text{minimize}_{\text{over}\boldsymbol{X},\boldsymbol{V},\boldsymbol{W}} & \text{tr}\ \boldsymbol{V} + \text{tr}\ \boldsymbol{W} & \\ \text{subject to} \quad \boldsymbol{W} \succ 0, & \begin{bmatrix} \boldsymbol{V} & (\boldsymbol{X}_k - \boldsymbol{X})^{\mathrm{T}} \\ \boldsymbol{X}_k - \boldsymbol{X} & \boldsymbol{W} \end{bmatrix} \succeq 0 & k = 1, \ldots, N, \end{array}$$

where $\boldsymbol{V} = \boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$, $\boldsymbol{W} = \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}$.

Moreover, the tests results of Chapter 4 demonstrated that the new method developed in this thesis can be up to 10^3 faster than the SeDuMi solver. Next it is shown how this feature can be exploited in control applications.

Consider discrete linear time-invariant systems with no inputs and state dynamics de-

scribed by

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k).$$

Consider also the candidate Lyapunov function for $\boldsymbol{P} = \boldsymbol{P}^{\mathrm{T}} \succ \boldsymbol{0}$

$$V(k) = \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(k) > 0,$$

with associated increment

$$\begin{aligned} V(k+1) - V(k) &= \boldsymbol{x}(k+1)^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(k+1) - \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(k) \\ &= \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{A}\boldsymbol{x}(k) - \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(k) \\ &= \boldsymbol{x}(k)^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{A} - \boldsymbol{P})\boldsymbol{x}(k) \end{aligned}$$

The condition $\Delta V < 0$ is satisfied provided

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{A} - \boldsymbol{P} \prec \boldsymbol{0},$$

or equivalently

$$\boldsymbol{P} - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{A} \succ \boldsymbol{0}.$$

Applying the Schur's complement formula gives

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{A}^{\mathrm{T}} \\ \boldsymbol{A} & \boldsymbol{P}^{-1} \end{bmatrix} \succ \boldsymbol{0}. \tag{6.21}$$

Moreover, if $\boldsymbol{P} = \boldsymbol{P}^{\mathrm{T}} \succ \boldsymbol{0}$ solves (6.21) then the system is stable.

Consider the class of switching systems described by

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_{\sigma(k)}\boldsymbol{x}(k)$$

where

$$\boldsymbol{A}_{\sigma(k)} \in \{\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N\}$$

If a matrix $\boldsymbol{P} = \boldsymbol{P}^{\mathrm{T}} \succ \boldsymbol{0}$ exists such that

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{A}_i^{\mathrm{T}} \\ \boldsymbol{A}_i & \boldsymbol{P}^{-1} \end{bmatrix} \succ \boldsymbol{0}, \quad i = 1, \ldots, N, \tag{6.22}$$

then the switched system is stable. However, the condition (6.22) is not an LMI in $\boldsymbol{P}$, but multiplying both sides by $\mathrm{diag}(\boldsymbol{I}, \boldsymbol{P})$ gives the following set of LMIs

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P} \end{bmatrix} \begin{bmatrix} \boldsymbol{P} & \boldsymbol{A}_i^{\mathrm{T}} \\ \boldsymbol{A}_i & \boldsymbol{P}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P} \end{bmatrix} = \begin{bmatrix} \boldsymbol{P} & \boldsymbol{A}_i^{\mathrm{T}}\boldsymbol{P} \\ \boldsymbol{P}\boldsymbol{A}_i & \boldsymbol{P} \end{bmatrix} \succ \boldsymbol{0}, \qquad i = 1, \ldots, N. \tag{6.23}$$

Consider also the Cholesky factorization $\boldsymbol{P} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}$ and suppose that the following

problem has a solution

$$\begin{aligned} \text{there exists} \quad & \text{Ellipsoid } \mathcal{E}(\boldsymbol{Q}^{\mathrm{T}} \otimes \boldsymbol{Q}^{-1}, \boldsymbol{0}) \\ \text{subject to} \quad & \operatorname{vec}(\boldsymbol{A}_i) \in \operatorname{int}\left(\mathcal{E}(\boldsymbol{Q}^{\mathrm{T}} \otimes \boldsymbol{Q}^{-1}, \boldsymbol{0})\right), \quad i = 1, \ldots, N \end{aligned}$$

where $\operatorname{int}(\cdot)$ denotes the interior. Then there exist norm bounded uncertainty of the form

$$\boldsymbol{A}_i = \boldsymbol{Q}^{-1}\boldsymbol{F}_i\boldsymbol{Q}, \quad \boldsymbol{F}_i^{\mathrm{T}}\boldsymbol{F}_i < \boldsymbol{I}, \quad i = 1, \ldots, N$$

and hence (6.22) is satisfied with $\boldsymbol{P} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}$ and $\boldsymbol{P}^{-1} = \boldsymbol{Q}^{-1}\boldsymbol{Q}^{-\mathrm{T}} = (\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q})^{-1}$. However, the method developed in Chapter 4 is not suited to finding such ellipsoids. First it is required to find the MVEE $\mathcal{E}(\boldsymbol{A}, \boldsymbol{0})$ containing all the points. In the approximation step in solving the problem

$$\text{minimize } \|\boldsymbol{A} - \boldsymbol{Q}^{\mathrm{T}} \otimes \boldsymbol{Q}^{-1}\|_F$$

we use the least squares framework to sequentially update the matrices involved whilst still maintaining feasibility

$$b_{ij} = \frac{\operatorname{tr}(\boldsymbol{A}_{ij}^{\mathrm{T}}\boldsymbol{Q}^{-1})}{\operatorname{tr}((\boldsymbol{Q}^{-1})^{\mathrm{T}}\boldsymbol{Q}^{-1})}, \qquad \boldsymbol{Q} \leftarrow \boldsymbol{B}^{\mathrm{T}}, \quad \boldsymbol{Q}^{-1} \leftarrow (\boldsymbol{B}^{\mathrm{T}})^{-1}$$

$$c_{ij} = \frac{\operatorname{tr}(\widehat{\boldsymbol{A}}_{ij}^{\mathrm{T}}\boldsymbol{Q}^{\mathrm{T}})}{\operatorname{tr}(\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})}, \qquad \boldsymbol{Q} \leftarrow \boldsymbol{C}^{-1}, \quad \boldsymbol{Q}^{-1} \leftarrow \boldsymbol{C}.$$

The next step is to scale the ellipsoid to obtain the axes' length that guarantee the ellipsoid contains all the points

$$(\sigma_1', \ldots, \sigma_{n^2}') = \text{ScaleEllipsoid}(\boldsymbol{Q}^{\mathrm{T}} \otimes \boldsymbol{Q}^{-1}, \boldsymbol{0})$$

and by SVD decomposition

$$\boldsymbol{Q}^{\mathrm{T}} = \boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^{\mathrm{T}}, \ \boldsymbol{Q}^{-1} = \boldsymbol{V}\boldsymbol{S}^{-1}\boldsymbol{U}^{T}, \ \boldsymbol{S} = \begin{bmatrix} s_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & s_n \end{bmatrix}, \ \boldsymbol{S}^{-1} = \begin{bmatrix} 1/s_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 1/s_n \end{bmatrix},$$

the fitting step is different from that in Chapter 4. In particular, the volume of the resulting ellipsoid is not an issue and hence the problem of fitting is reduced to finding

a solution of the following system

$$\begin{aligned}
&\frac{s_1}{s_1} > \sigma'_1\\
&\frac{s_1}{s_2} > \sigma'_2\\
&\ldots > \ldots\\
&\frac{s_1}{s_n} > \sigma'_n\\
&\frac{s_2}{s_1} > \sigma'_{n+1}\\
&\ldots > \ldots\\
&\frac{s_n}{s_n} > \sigma'_{n^2}
\end{aligned} \tag{6.24}$$

or the system of linear inequalities

$$\begin{aligned}
1 &> \sigma'_1\\
s_1 &> \sigma'_2 s_2\\
\ldots &> \ldots\\
s_1 &> \sigma'_n s_n\\
s_2 &> \sigma'_{n+1} s_1\\
\ldots &> \ldots\\
s_n &> \sigma'_{n^2} s_n
\end{aligned}$$

The necessary condition for the solution to exist is

$$\sigma'_1 < 1, \quad \sigma'_{n+2} < 1, \quad \sigma'_{2n+3} < 1, \quad \ldots \quad \sigma'_{n^2} < 1. \tag{6.25}$$

Considering any two equations for $i \neq j$ gives

$$s_i > \sigma_{(i-1)n+j} s_j, \quad s_j > \sigma_{(j-1)n+i} s_i, \quad i, j = 1, \ldots, n, \quad i \neq j \tag{6.26}$$

and hence $s_i > \sigma_{(i-1)n+j}\sigma_{(j-1)n+i} s_i$. Consequently the necessary conditions also include

$$1 > \sigma_{(i-1)n+j}\sigma_{(j-1)n+i}, \quad i, j = 1, \ldots, n, \quad i \neq j. \tag{6.27}$$

Rewriting (6.26) in the following form

$$\frac{s_j}{\sigma_{(j-1)n+i}} > s_i > \sigma_{(i-1)n+j} s_j, \quad i, j = 1, \ldots, n, \quad i \neq j$$

and for each i dividing the resulting equation by s_j gives

$$\frac{1}{\sigma_{(j-1)n+i}} > s_i > \sigma_{(i-1)n+j}, \quad i = 1, \ldots, n. \tag{6.28}$$

The solution of (6.28), and hence the solution of the complete system (6.24) exists if (6.27) is satisfied. Therefore conditions (6.25) and (6.27) are necessary and sufficient for the solution of (6.24) to exist and they can be combined into the equivalent condition

$$1 > \sigma_{(i-1)n+j}\sigma_{(j-1)n+i}, \quad i,j = 1,\dots,n. \tag{6.29}$$

The method does not necessarily need to evaluate the solution.In particular, it is sufficient to determine if fitting is possible or not by checking (6.29). The complete procedure is summarized as Algorithm 6.4.1.

Algorithm 6.4.1 Check stability of the switched system

function CHECKSTABILITY($\boldsymbol{A}_k, \quad k = 1,\dots,N$)
  $\boldsymbol{x}_k = \mathrm{vec}(\boldsymbol{A}_k)$, $k = 1,\dots,N$
  $\boldsymbol{\Sigma} \leftarrow \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{x}_k\boldsymbol{x}_k^{\mathrm{T}}$
  $\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{-1} = \boldsymbol{\Sigma}$ ▷ Eigendecomposition
  $\boldsymbol{U} \leftarrow \boldsymbol{V}$, $\boldsymbol{\sigma} = \varepsilon \cdot \boldsymbol{1}$ ▷ ε-lowest possible float
  $\boldsymbol{P} \leftarrow \boldsymbol{U} \cdot \mathrm{diag}(\boldsymbol{\sigma})$
  $\boldsymbol{A} \leftarrow$ SCALEELLIPSOID($\boldsymbol{A}, \boldsymbol{0}, \boldsymbol{X}$)
  $\boldsymbol{Q} \leftarrow \boldsymbol{I}$ ▷ Initial ellipsoid can be determined also by Rank-1 approximation
  $\min_F = \|\boldsymbol{A} - \boldsymbol{Q}^{\mathrm{T}} \otimes \boldsymbol{Q}^{-1}\|_F$
  $\boldsymbol{Q}_s \leftarrow \boldsymbol{Q}$
  for $it = 1,\dots,MaxIt$ **do** ▷ Max iterations
    for $i,j = 1,\dots,N$ **do**

$$b_{ij} \leftarrow \frac{\mathrm{tr}(\boldsymbol{P}_{ij}^{\mathrm{T}}\boldsymbol{Q}^{-1})}{\mathrm{tr}((\boldsymbol{Q}^{-1})^{\mathrm{T}}\boldsymbol{Q}^{-1})}$$

    end for
    $\boldsymbol{Q} \leftarrow \boldsymbol{B}^{\mathrm{T}}, \quad \boldsymbol{Q}^{-1} \leftarrow (\boldsymbol{B}^{\mathrm{T}})^{-1}$
    if $\min_F > \|\boldsymbol{A} - \boldsymbol{Q}^{\mathrm{T}} \otimes \boldsymbol{Q}^{-1}\|_F$ **then**
      $\boldsymbol{Q}_s \leftarrow \boldsymbol{Q}$, $\min_F \leftarrow \|\boldsymbol{A} - \boldsymbol{Q}_s^{\mathrm{T}} \otimes \boldsymbol{Q}_s^{-1}\|_F$ ▷ Store the best result
    end if
    for $i,j = 1,\dots,N$ **do**

$$c_{ij} \leftarrow \frac{\mathrm{tr}(\widehat{\boldsymbol{P}}_{ij}^{\mathrm{T}}\boldsymbol{Q}^{\mathrm{T}})}{\mathrm{tr}(\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})}$$

    end for
    $\boldsymbol{Q} \leftarrow \boldsymbol{C}^{-1}, \quad \boldsymbol{Q}^{-1} \leftarrow \boldsymbol{C}$
    if $\min_F > \|\boldsymbol{A} - \boldsymbol{Q}^{\mathrm{T}} \otimes \boldsymbol{Q}^{-1}\|_F$ **then**
      $\boldsymbol{Q}_s \leftarrow \boldsymbol{Q}$, $\min_F \leftarrow \|\boldsymbol{A} - \boldsymbol{Q}_s^{\mathrm{T}} \otimes \boldsymbol{Q}_s^{-1}\|_F$ ▷ Store the best result
    end if
  end for
  $\boldsymbol{\sigma} \leftarrow$ SCALEELLIPSOID($\boldsymbol{Q}_s^{\mathrm{T}} \otimes \boldsymbol{Q}_s^{-1}, \boldsymbol{0}, \boldsymbol{X}$)
  if Condition (6.29) is satisfied **then**
    return STABLE ▷ Stable
  else
    return UNDETERMINED ▷ No conclusion possible
  end if
end function

Example 6.3. *Consider the case of* $N = 20,000$ 5×5 *matrices with randomly generated entries where*

$$\max\{\rho(\boldsymbol{A}_i) : i = 1, \ldots, N\} = 0.64123$$

Algorithm 6.4.1 produced the result that this system is stable in 0.611 *seconds. The same result was obtained using SeDuMi and LMI conditions (6.23) in* 89.107 *seconds. Hence the new method is faster.*

6.5 Conclusions

This chapter has considered the application of the new algorithms developed in Chapters 3 and 4 to three representative problems. The results given established basic feasibility but much further work is required in order to determine their true potential. This general area will be discussed in the next chapter.

Chapter 7

Conclusions

7.1 Novel contributions

In Chapter 2 models to represent the effects of synchronization errors in linear systems have been developed for both the centralised and decentralised control cases. Next the stability of asynchronous algorithms has been addressed resulting in theorems that can be applied to the second of these cases. These results do not apply to the centralized control case and only a brute force method. In particular the only existing method was a formulation of the stabilization problem in terms of LMIs for all state matrices that arise. However, as the number of synchronization errors that can arise grows very quickly with the order of the system, the performance of LMI solvers decays below any acceptable level. This motivated research to answer the following open research problems

- Is it possible to stabilize the system against all synchronization errors using state feedback ?
- Can the design be completed in a computationally efficient way ?

In Chapter 3 the solution of these problems in the presence of all clock synchronization errors has been developed by exploiting the polytopic uncertainty description from robust control theory. In particular, the complete set of possible systems is first written in this setting of uncertainty on some nominal model, which in turn releases convex hull algorithms for use in this area. Numerical tests then revealed that the efficiency of these algorithms is not acceptable for even small scale problems. This then motivated the development of the new algorithm for the computation of a polytope containing given set of matrices, that balances the trade-off between the volume and the number of vertices in the convex hull.

In this new algorithm the number of vertices depends on the dimension of the space but not on the number of input matrices and this makes it suitable for large scale problems.

Its computation is undertaken using the Minimum Volume Enclosing Ellipsoid that can be performed using, for example, the method of Khachiyan from the literature. An alternative method based on principal component analysis is another major contribution of this thesis. To provide a comparison of relative computational efficiency extensive numerical tests were first undertaken between the principal component analysis based method and the Khachiyan algorithm. The test data given in Tables 3.1 and 3.2 confirm that a speed up of the order 2 is possible. This factor also increases with increasing tolerance level in the Khachiyan method.

A further comparison was undertaken between the brute force LMI method and the polytopic description method with the Khachiyan algorithm. In this case the latter again outperformed the former. Also if the ellipsoid construction algorithm is undertaken using a method based on principal component analysis then even greater efficiency can be achieved.

Chapter 4 treats the same problem as Chapter 3 but using norm bounded uncertainty description from robust control. The method developed is fast and robust and allows the solution of large scale problems. Numerical tests, see Figures 4.4 and 4.5, show that this method markedly outperform the existing LMI formulation and solvers.

The analysis of Chapters 2–4 require the availability of a state space model of the plant dynamics and knowledge of the clock synchronization errors that arise. In at least some applications the clock synchronization error sequences may not be known. Hence if synchronization error occur in operation a valid question to ask is can this be detected from the knowledge of the plant input and output over a finite time duration. Chapter 5 has shown that he answer to this question is positive in the common clock case. In other cases this question is still unanswered, and is discussed again in the next section.

This thesis has focused on answering basic research questions but it is also essential to establish if they could be of use in applications. To this end, Chapter 6 gives basic results in three areas. The first of these is one formulation of the general rendezvous problem applicable to multi-agent systems. In this application a group of autonomous vehicles aim to meet at one point and it is assumed that they operate asynchronously, which is modelled by systems with clock synchronization errors. The new methods in this thesis are then applied to compute a stabilizing state feedback control law. The second application considers the problem of iterative learning control for systems with synchronization errors which can also have a multi-agent aspect. Finally, some preliminary results have been developed on using these methods to speeding up the solution of of an LMI that is generic to a number of control theory problems.

In summary the novel contributions in this thesis are as follows

- An algorithm for estimating the number of clock synchronization errors in the common clock case together with upper and lower bounds and an approximation of this number in terms of the order of the system (Chapter 2).
- An algorithm for the computation of the approximation to the minimum volume enclosing ellipsoid based on principal component analysis with improved computationally efficiency over alternatives (Chapter 3).
- An efficient algorithm for the computation of a polytope containing a given set of matrices. The number of vertices of the resulting polytope depends on the dimension of the space but not on the number of input matrices which makes it suitable for large scale problems (Chapter 3).
- An algorithm for the computation of the norm bounded uncertainty for a given set of matrices. This algorithm is also suitable for large scale problems (Chapter 4).
- A method of estimation of a clock synchronization errors from noise free output trajectory together with some preliminary results on identifiability conditions (Chapter 5).
- Basic feasibility results on the application of these new algorithms to three possible application areas, including the computation of stabilizing control laws (Chapter 6).

7.2 Directions for future research

The results in this thesis will be enhanced and extended by further research. In particular, the following major areas of work should be addressed.

- **Fitting an uncertainty ellipsoid to an ellipsoid of lower dimension**

  In the method of Chapter 4 the problem of fitting an uncertainty ellipsoid defined by (4.51) to the ellipsoid enclosing all the input points is critical. This was done by approximation of the ellipsoid matrix with the Kronecker product of two others and by fitting the length of uncertainty ellipsoid axes. In application to systems with synchronization errors, the enclosing ellipsoid is not of full dimension, which arises from the fact that input points (3.9) are located on some hyperplane, see Figure 3.1. However, the approximation step most often results in a full rank matrix and better results could be obtained in this case if the dimensions of the approximated ellipsoid is increased by setting the length of remaining axes to a

nonzero value. The proposed heuristic value is given by (4.63). Further research is required to determine whether or not this value is the best possible. If, however, such a theoretical justification cannot be established then an attempt should be made to obtain at least an algorithm that has increased efficiency for this essential task. This is of particular relevance for applications such as systems with clock synchronization errors.

- **Fitting the length of the uncertainty ellipsoid axes**

  This task, which is part of the method developed in Chapter 4, requires the solution of the optimization problem (4.55), which, in general, is non-convex with bi-linear constraints. Necessary conditions for the optimum are given by (4.56) and Theorem 4.7 shows that for any given starting point from the domain of solution the stationary points (4.57), which are obtained by manipulating of coordinates of the starting point, satisfy (4.56). The current method uses the particular starting point (4.62) in order to find a suboptimal solution to (4.55). Further research is needed on the problem of choosing the initial point resulting in a theoretical justification and/or an efficient algorithm that would give significantly better results than the current method.

- **State space model of subsystems with non-negligible switching times**

  In the model developed in Chapter 2, on which all results in this thesis are based, assumes that the switching of a subsystem is instantaneous, or the switching time is negligible. This allows formulation of the state space model in the case of centralized control. If in an application the switching times are not negligible this will introduce delays into the dynamics and it is then not possible to couple the subsystem models. An open research problem is the development of a model for centralized control in such cases.

- **New approach to solving LMIs that arise from control theory problem**

  Chapter 4 has developed a method of estimating the norm bounded uncertainty of a given example that is equivalent to solving a particular optimization problem subject to LMI constraints of specific structure. This method gives a suboptimal solution of this latter problem in a much faster way than existing LMI solver. Some initial progress on exploiting this fact for LMI problems in control theory, outside those considered in this thesis, has been given in Chapter 6. Further research is required to determine the full potential of this initial results.

- **Estimation of a clock synchronization error from input-output data.**

  The estimation method developed in Chapter 5 assumed that the plant conidered is noise-free and this assumption is unrealistic in some cases. Also, to guarantee that the error is identifiable from the output two assumptions have been made, but only a sufficient condition for the first and a necessary condition for the second

are currently available. Hence there are many open research questions. These include theoretical conditions for identifiability and efficient and well-conditioned algorithms in the case where the noise cannot be neglected.

- **Further application based studies.**

  Chapter 6 has given initial results on the application of the results in this thesis to two application areas that are of considerable interest both in terms of control theory and real world problems. There is a clear need to build on these initial results to address other aspects crucial to eventual use, such as in-depth simulation studies followed, where justified, by benchmark experimentation.

Appendix A

Iterative learning control for systems with uncertain dynamics

The following results for iterative learning control in the presence of plant uncertainty are used in Section 6.3.

Theorem A.1. *Consider the system (6.13) and assume the matrices take values in a polytope*

$$[\boldsymbol{A},\ \boldsymbol{B}] \in Co([\boldsymbol{A}_i,\ \boldsymbol{B}_i] : i = 1, \ldots, N).$$

Then this system is stable along the trial if there exist matrices $\boldsymbol{X}_1 \succ \boldsymbol{0}$, $\boldsymbol{X}_2 \succ \boldsymbol{0}$, $\boldsymbol{R}_1$ *and* $\boldsymbol{R}_2$ *such that the following LMIs are feasible for* $i = 1, \ldots, N$

$$\boldsymbol{\Psi}_i = \begin{bmatrix} -\boldsymbol{X}_1 & \boldsymbol{0} & \boldsymbol{X}_1\boldsymbol{A}_i^T + \boldsymbol{R}_1^T\boldsymbol{B}_i^T & -\boldsymbol{X}_1\boldsymbol{A}_i^T\boldsymbol{C}^T - \boldsymbol{R}_1^T\boldsymbol{B}_i^T\boldsymbol{C}^T \\ \boldsymbol{0} & -\boldsymbol{X}_2 & \boldsymbol{R}_2^T\boldsymbol{B}_i^T & \boldsymbol{X}_2 - \boldsymbol{R}_2^T\boldsymbol{B}_i^T\boldsymbol{C}^T \\ \boldsymbol{A}_i\boldsymbol{X}_1 + \boldsymbol{B}_i\boldsymbol{R}_1 & \boldsymbol{R}_2 & -\boldsymbol{X}_1 & \boldsymbol{0} \\ -\boldsymbol{C}\boldsymbol{A}_i\boldsymbol{X}_1 - \boldsymbol{C}\boldsymbol{B}_i\boldsymbol{R}_1 & \boldsymbol{X}_2 - \boldsymbol{C}\boldsymbol{B}_i\boldsymbol{R}_2 & \boldsymbol{0} & -\boldsymbol{X}_2 \end{bmatrix} \prec \boldsymbol{0}. \quad \text{(A.1)}$$

If (A.1) holds, stabilizing control law matrices are given by

$$\boldsymbol{K}_1 = \boldsymbol{R}_1\boldsymbol{X}_1^{-1}, \quad \boldsymbol{K}_2 = \boldsymbol{R}_2\boldsymbol{X}_2^{-1}. \quad \text{(A.2)}$$

Proof: Consider $[\boldsymbol{A},\ \boldsymbol{B}] \in \text{Co}([\boldsymbol{A}_i,\ \boldsymbol{B}_i] : i = 1, \ldots, N)$ expressed as a convex combi-

nation of the vertices

$$[\boldsymbol{A},\ \boldsymbol{B}] = \sum_{i=1}^{N} \alpha_i [\boldsymbol{A}_i,\ \boldsymbol{B}_i]. \tag{A.3}$$

Since

$$\boldsymbol{\Psi}_i \prec \boldsymbol{0}, \qquad i = 1, \ldots, N,$$

taking the convex combination of (A.3) gives

$$\sum_{i=1}^{N} \alpha_i \boldsymbol{\Psi}_i = \boldsymbol{\Psi} \prec \boldsymbol{0},$$

which is (6.16) and guarantees stability along the trial of the ILC for (A.3) with control law matrices given by (A.2). □

Consider the discrete linear repetitive process described by the state space model

$$\begin{aligned} \boldsymbol{x}_{k+1}(p+1) &= (\boldsymbol{A} + \Delta\boldsymbol{A})\boldsymbol{x}_{k+1}(p) + (\boldsymbol{B}_0 + \Delta\boldsymbol{B}_0)\boldsymbol{y}_k(p) + (\boldsymbol{B} + \Delta\boldsymbol{B})\boldsymbol{u}_{k+1}(p) \\ \boldsymbol{y}_{k+1}(p) &= (\boldsymbol{C} + \Delta\boldsymbol{C})\boldsymbol{x}_{k+1}(p) + (\boldsymbol{D}_0 + \Delta\boldsymbol{D}_0)\boldsymbol{y}_k(p) + (\boldsymbol{D} + \Delta\boldsymbol{D})\boldsymbol{u}_{k+1}(p) \end{aligned} \tag{A.4}$$

where matrices $\Delta\boldsymbol{A}, \Delta\boldsymbol{B}, \Delta\boldsymbol{B}_0, \Delta\boldsymbol{C}, \Delta\boldsymbol{D}, \Delta D_0$ represents the uncertainty defined as

$$\begin{bmatrix} \Delta\boldsymbol{A} & \Delta\boldsymbol{B}_0 & \Delta\boldsymbol{B} \\ \Delta\boldsymbol{C} & \Delta\boldsymbol{D}_0 & \Delta\boldsymbol{D} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_1 \\ \boldsymbol{H}_2 \end{bmatrix} \boldsymbol{F} \begin{bmatrix} \boldsymbol{E}_1 & \boldsymbol{E}_2 & \boldsymbol{E}_3 \end{bmatrix}, \quad \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} \preceq \boldsymbol{I}.$$

The following theorem (Paszke, 2002, Theorem 4.6) solves the stabilization problem for such repetitive processes.

Theorem A.2. *Suppose that a discrete linear repetitive process of the form (A.4) is controlled by a law of the form*

$$\boldsymbol{u}_{k+1}(p) = \begin{bmatrix} \boldsymbol{K}_1 & \boldsymbol{K}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{k+1}(p) \\ \boldsymbol{y}_k(p) \end{bmatrix}.$$

Then the resulting controlled process is stable along the trial for all admissible uncertainties if there exist matrices $\boldsymbol{W}_1 \succ \boldsymbol{0}$, $\boldsymbol{W}_2 \succ \boldsymbol{0}$, $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$ of compatible dimensions

and a scalar $\epsilon > 0$ *such that the following LMI holds*

$$\begin{bmatrix} -\boldsymbol{W}_1 + 2\epsilon \boldsymbol{H}_1 \boldsymbol{H}_1^T & 2\epsilon \boldsymbol{H}_2 \boldsymbol{H}_1^T & \boldsymbol{A}\boldsymbol{W}_1 + \boldsymbol{B}\boldsymbol{N}_1 \\ \mathbf{2}\epsilon \boldsymbol{H}_1 \boldsymbol{H}_2^T & -\boldsymbol{W}_2 + 2\epsilon \boldsymbol{H}_2 \boldsymbol{H}_2^T & \boldsymbol{C}\boldsymbol{W}_1 + \boldsymbol{D}\boldsymbol{N}_1 \\ \boldsymbol{W}_1 \boldsymbol{A}^T + \boldsymbol{N}_1^T \boldsymbol{B}^T & \boldsymbol{W}_1 \boldsymbol{C}^T + \boldsymbol{N}_1^T \boldsymbol{D}^T & -\boldsymbol{W}_1 \\ \boldsymbol{W}_2 \boldsymbol{B}_0^T & \boldsymbol{W}_2 \boldsymbol{D}_0^T + \boldsymbol{N}_2 \boldsymbol{D}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{E}_1 \boldsymbol{W}_1 + \boldsymbol{E}_3 \boldsymbol{N}_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$
$$\begin{bmatrix} \boldsymbol{B}_0 \boldsymbol{W}_2 + \boldsymbol{B}\boldsymbol{N}_2 & \mathbf{0} & \mathbf{0} \\ \boldsymbol{D}_0 \boldsymbol{W}_2 + \boldsymbol{D}\boldsymbol{N}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{W}_1 \boldsymbol{E}_1^T + \boldsymbol{N}_1^T \boldsymbol{E}_3^T & \mathbf{0} \\ -\boldsymbol{w}_2 & \mathbf{0} & \boldsymbol{W}_2 \boldsymbol{E}_2^T + \boldsymbol{N}_2^T \boldsymbol{E}_3^T \\ \mathbf{0} & -\epsilon \boldsymbol{I} & \mathbf{0} \\ \boldsymbol{E}_2 \boldsymbol{W}_2 + \boldsymbol{E}_3 \boldsymbol{N}_2 & \mathbf{0} & -\epsilon \boldsymbol{I} \end{bmatrix} \prec \mathbf{0}.$$

If this LMI condition is satisfied, stabilizing control law matrices matrices are given by

$$\boldsymbol{K}_1 = \boldsymbol{N}_1 \boldsymbol{W}_1^{-1}, \quad \boldsymbol{K}_2 = \boldsymbol{N}_2 \boldsymbol{W}_2^{-1}. \tag{A.5}$$

Theorem A.3. *Consider the system (6.13) and assume the matrices take values in the norm bounded uncertainty set*

$$[\ \boldsymbol{A},\ \boldsymbol{B}\ ] \in \left\{ [\ \boldsymbol{A}_0,\ \boldsymbol{B}_0\ ] + \boldsymbol{H}\boldsymbol{F}\,[\ \boldsymbol{E}_1,\ \boldsymbol{E}_2\ ] : \quad \boldsymbol{F}^T \boldsymbol{F} \preceq \boldsymbol{I} \right\}, \tag{A.6}$$

for some matrices $\boldsymbol{H}$, $\boldsymbol{E}_1$ *and* $\boldsymbol{E}_2$. *Then this system is stable along the trial if there exist matrices* $\boldsymbol{W}_1 \succ \mathbf{0}$, $\boldsymbol{W}_2 \succ \mathbf{0}$, $\boldsymbol{N}_1$ *and* $\boldsymbol{N}_2$ *and a scalar* $\epsilon > 0$ *such that the following LMI is feasible*

$$\begin{bmatrix} \tilde{\boldsymbol{M}} & \tilde{\boldsymbol{E}}^T \\ \tilde{\boldsymbol{E}} & -\epsilon \boldsymbol{I} \end{bmatrix} \prec 0, \tag{A.7}$$

where

$$\tilde{\boldsymbol{E}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \boldsymbol{E}_1 \boldsymbol{W}_1 + \boldsymbol{E}_2 \boldsymbol{N}_1 & \boldsymbol{E}_2 \boldsymbol{N}_2 \\ \mathbf{0} & \mathbf{0} & -\boldsymbol{E}_1 \boldsymbol{W}_1 - \boldsymbol{E}_2 \boldsymbol{N}_1 & -\boldsymbol{E}_2 \boldsymbol{N}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

and

$$\tilde{\boldsymbol{M}} = \begin{bmatrix} -\boldsymbol{W}_1 + \epsilon \boldsymbol{H}\boldsymbol{H}^T & \mathbf{0} & \boldsymbol{A}\boldsymbol{W}_1 + \boldsymbol{B}\boldsymbol{N}_1 & \boldsymbol{B}\boldsymbol{N}_2 \\ \mathbf{0} & -\boldsymbol{W}_2 + \epsilon \boldsymbol{C}\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{C} & -\boldsymbol{C}\boldsymbol{A}\boldsymbol{W}_1 - \boldsymbol{C}\boldsymbol{B}\boldsymbol{N}_1 & \boldsymbol{W}_2 - \boldsymbol{C}\boldsymbol{B}\boldsymbol{N}_2 \\ \boldsymbol{W}_1 \boldsymbol{A}^T + \boldsymbol{N}_1^T \boldsymbol{B}^T & -\boldsymbol{W}_1 \boldsymbol{A}^T \boldsymbol{C}^T - \boldsymbol{N}_1^T \boldsymbol{B}^T \boldsymbol{C}^T & -\boldsymbol{W}_1 & \mathbf{0} \\ \boldsymbol{N}_2^T \boldsymbol{B}^T & \boldsymbol{W}_2 - \boldsymbol{N}_2^T \boldsymbol{B}^T \boldsymbol{C}^T & \mathbf{0} & -\boldsymbol{W}_2 \end{bmatrix}.$$

If (A.7) holds, stabilizing control law matrices are given by

$$\boldsymbol{K}_1 = \boldsymbol{N}_1 \boldsymbol{W}_1^{-1}, \quad \boldsymbol{K}_2 = \boldsymbol{N}_2 \boldsymbol{W}_2^{-1}. \tag{A.8}$$

PROOF: Omitting uncertainty in Theorem A.2 gives the stability condition as

$$\begin{bmatrix} -\boldsymbol{W}_1 & \boldsymbol{0} & \hat{\boldsymbol{A}}\boldsymbol{W}_1 & \hat{\boldsymbol{B}}_0\boldsymbol{W}_2 \\ \boldsymbol{0} & -\boldsymbol{W}_2 & \hat{\boldsymbol{C}}\boldsymbol{W}_1 & \hat{\boldsymbol{D}}_0\boldsymbol{W}_2 \\ \boldsymbol{W}_1\hat{\boldsymbol{A}}^{\mathrm{T}} & \boldsymbol{W}_1\hat{\boldsymbol{C}}^{\mathrm{T}} & -\boldsymbol{W}_1 & \boldsymbol{0} \\ \boldsymbol{W}_2\hat{\boldsymbol{B}}_0{}^{\mathrm{T}} & \boldsymbol{W}_2\hat{\boldsymbol{D}}_0^{\mathrm{T}} & \boldsymbol{0} & -\boldsymbol{W}_2 \end{bmatrix} \prec \boldsymbol{0}.$$

Substituting for the matrices $\hat{\boldsymbol{A}}$, $\hat{\boldsymbol{B}}_0$, $\hat{\boldsymbol{C}}$ and $\hat{\boldsymbol{D}}_0$ yields

$$\begin{bmatrix} -\boldsymbol{W}_1 & \boldsymbol{0} & & \\ \boldsymbol{0} & -\boldsymbol{W}_2 & & \\ \boldsymbol{W}_1\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{N}_1^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}} & -\boldsymbol{W}_1\boldsymbol{A}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} - \boldsymbol{N}_1^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} & & \\ \boldsymbol{N}_2^{\mathrm{T}}\boldsymbol{B} & \boldsymbol{W}_2 - \boldsymbol{N}_2^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} & & \\ & & \boldsymbol{A}\boldsymbol{W}_1 + \boldsymbol{B}\boldsymbol{N}_1 & \boldsymbol{B}\boldsymbol{N}_2 \\ & & -\boldsymbol{C}\boldsymbol{A}\boldsymbol{W}_1 - \boldsymbol{C}\boldsymbol{B}\boldsymbol{N}_1 & \boldsymbol{W}_2 - \boldsymbol{C}\boldsymbol{B}\boldsymbol{N}_2 \\ & & -\boldsymbol{W}_1 & \boldsymbol{0} \\ & & \boldsymbol{0} & -\boldsymbol{W}_2 \end{bmatrix} \prec \boldsymbol{0}. \quad \text{(A.9)}$$

The control law matrices are given by (A.5).

Consider now the case when the system (6.5) is uncertain and the uncertainty is modelled by (A.6). Then ILC system is stable along the trial if

$$\begin{bmatrix} -\boldsymbol{W}_1 & \boldsymbol{0} & \boldsymbol{A}\boldsymbol{W}_1 + \boldsymbol{B}\boldsymbol{N}_1 & \boldsymbol{B}\boldsymbol{N}_2 \\ \boldsymbol{0} & -\boldsymbol{W}_2 & -\boldsymbol{C}\boldsymbol{A}\boldsymbol{W}_1 - \boldsymbol{C}\boldsymbol{B}\boldsymbol{N}_1 & \boldsymbol{W}_2 - \boldsymbol{C}\boldsymbol{B}\boldsymbol{N}_2 \\ \boldsymbol{W}_1\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{N}_1^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}} & -\boldsymbol{W}_1\boldsymbol{A}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} - \boldsymbol{N}_1^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} & -\boldsymbol{W}_1 & \boldsymbol{0} \\ \boldsymbol{N}_2^{\mathrm{T}}\boldsymbol{B} & \boldsymbol{W}_2 - \boldsymbol{N}_2^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} & \boldsymbol{0} & -\boldsymbol{W}_2 \end{bmatrix} +$$
$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \Delta\boldsymbol{A}\boldsymbol{W}_1 + \Delta\boldsymbol{B}\boldsymbol{N}_1 & \Delta\boldsymbol{B}\boldsymbol{N}_2 \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{C}\Delta\boldsymbol{A}\boldsymbol{W}_1 - \boldsymbol{C}\Delta\boldsymbol{B}\boldsymbol{N}_1 & -\boldsymbol{C}\Delta\boldsymbol{B}\boldsymbol{N}_2 \\ \boldsymbol{W}_1\Delta\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{N}_1^{\mathrm{T}}\Delta\boldsymbol{B}^{\mathrm{T}} & -\boldsymbol{W}_1\Delta\boldsymbol{A}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} - \boldsymbol{N}_1^{\mathrm{T}}\Delta\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{N}_2^{\mathrm{T}}\Delta\boldsymbol{B}^{\mathrm{T}} & -\boldsymbol{N}_2^{\mathrm{T}}\Delta\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \prec \boldsymbol{0}.$$

The second term on the left hand side in the above expression can be written as

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0}H & \boldsymbol{C}\boldsymbol{H} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{F} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{F} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{F} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{F} \end{bmatrix} \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{E}_1\boldsymbol{W}_1 + \boldsymbol{E}_2\boldsymbol{N}_1 & \boldsymbol{E}_2\boldsymbol{N}_2 \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{E}_1\boldsymbol{W}_1 - \boldsymbol{E}_2\boldsymbol{N}_1 & -\boldsymbol{E}_2\boldsymbol{N}_2 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}$$
$$+ \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{W}_1\boldsymbol{E}_1^{\mathrm{T}} + \boldsymbol{N}_1^{\mathrm{T}}\boldsymbol{E}_2^{\mathrm{T}} & -\boldsymbol{W}_1\boldsymbol{E}_1^{\mathrm{T}} - \boldsymbol{N}_1^{\mathrm{T}}\boldsymbol{E}_2^{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{N}_2^{\mathrm{T}}\boldsymbol{E}_2^{\mathrm{T}} & -\boldsymbol{N}_2^{\mathrm{T}}\boldsymbol{E}_2^{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{F}^{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{F}^{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{F}^{\mathrm{T}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{F}^{\mathrm{T}} \end{bmatrix}$$

$$\times \begin{bmatrix} \boldsymbol{H}^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{H}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Applying Lemma 4.1 now gives

$$\epsilon^{-1} \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{W}_1\boldsymbol{E}_1^{\mathrm{T}} + \boldsymbol{N}_1^{\mathrm{T}}\boldsymbol{E}_2^{\mathrm{T}} & -\boldsymbol{W}_1\boldsymbol{E}_1^{\mathrm{T}} - \boldsymbol{N}_1^{\mathrm{T}}\boldsymbol{E}_2^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{N}_2^{\mathrm{T}}\boldsymbol{E}_2^{\mathrm{T}} & -\boldsymbol{N}_2^{\mathrm{T}}\boldsymbol{E}_2^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} & \boldsymbol{E}_1\boldsymbol{W}_1 + \boldsymbol{E}_2\boldsymbol{N}_1 & \boldsymbol{E}_2\boldsymbol{N}_2 \\ \mathbf{0} & \mathbf{0} & -\boldsymbol{E}_1\boldsymbol{W}_1 - \boldsymbol{E}_2\boldsymbol{N}_1 & -\boldsymbol{E}_2\boldsymbol{N}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$+ \begin{bmatrix} -\boldsymbol{W}_1 + \epsilon\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} & \mathbf{0} & \boldsymbol{A}\boldsymbol{W}_1 + \boldsymbol{B}\boldsymbol{N}_1 & \boldsymbol{B}\boldsymbol{N}_2 \\ \mathbf{0} & -\boldsymbol{W}_2 + \epsilon\boldsymbol{C}\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{C} & -\boldsymbol{C}\boldsymbol{A}\boldsymbol{W}_1 - \boldsymbol{C}\boldsymbol{B}\boldsymbol{N}_1 & \boldsymbol{W}_2 - \boldsymbol{C}\boldsymbol{B}\boldsymbol{N}_2 \\ \boldsymbol{W}_1\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{N}_1^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}} & -\boldsymbol{W}_1\boldsymbol{A}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} - \boldsymbol{N}_1^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} & -\boldsymbol{W}_1 & \mathbf{0} \\ \boldsymbol{N}_2^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}} & \boldsymbol{W}_2 - \boldsymbol{N}_2^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}} & \mathbf{0} & -\boldsymbol{W}_2 \end{bmatrix}$$

$$= \epsilon^{-1}\tilde{\boldsymbol{E}}^{\mathrm{T}}\tilde{\boldsymbol{E}} + \tilde{\boldsymbol{M}} \prec \mathbf{0}.$$

By the Schur's complement formula

$$\begin{bmatrix} \tilde{\boldsymbol{M}} & \tilde{\boldsymbol{E}}^{\mathrm{T}} \\ \tilde{\boldsymbol{E}} & -\epsilon\boldsymbol{I} \end{bmatrix} \prec 0, \tag{A.10}$$

which is (A.7) for this case, hence the ILC scheme (6.13) is stable along the trial with control law matrices given by

$$\boldsymbol{K}_1 = \boldsymbol{N}_1\boldsymbol{W}_1^{-1}, \quad \boldsymbol{K}_2 = \boldsymbol{N}_2\boldsymbol{W}_2^{-1}.$$

□

Bibliography

A.-M.Hughes, C. T. Freeman, J. H. Burridge, P. H. Chappell, P. L. Lewin, and E. Rogers. Feasibility of iterative learning control mediated by functional electrical stimulation for reaching after stroke. *Journal of Neurorehabilitation and Neural Repair*, 23:559–568, 2009.

H. S. Ahn, Y. Q. Chen, and K. L. Moore. Iterative learning control: Brief survey and categorization. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 37: 1109–1121, 2007.

A. M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9:216–219, 1979.

E. Asarin, V. Kozyakin, M. Krasnoselskii, and N. Kuznetsov. *Stability Analysis of Desynchronized Discrete-Event Systems*. Nauka, Moscow, 1992.

C. Barber, D. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. On Math. Software*, 22(4):469–483, 1996.

D. Bertsekas and J. Tsitsiklis. *Parallel and distributed computation - Numerical methods*. Prentice Halls, Englewood Cliffs, New Jersey, 1989.

El-Kebir Boukas and Peng Shi. H_∞ control for discrete-time linear systems with frobenius norm-bounded uncertainties. *American Control Conference*, 1:557–561, 1998.

S. Boyd, L. E. Ghaoui, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. SIAM studies in Applied Mathematics, Philadelphia, 1994.

D. A. Bristow, M. Tharayil, and A. G. Alleyne. A survey of iterative learning control. *IEEE Control Systems Magazine*, 26:96–114, 2006.

D. R. Chan and S. S. Kapur. An algorithm for convex polytopes. *J. Assoc. Comput. Mach.*, 17:78–86, 1970.

T. Chan. Optimal output-sensitive convex hull algorithms in two or three dimensions. *Discrete and Computational Geometry*, 16:361–368, 1996.

D. Chand and S. Kapur. An algorithm for convex polytopes. *Journal of the ACM*, 7: 78–86, 1970.

D. Chazan and W. L. Miranker. Chaotic relaxation. *Linear Algebra and Appl.*, 43(2): 213–231, 1969.

C. A. R. Crusius and A. Trofino. Sufficient lmi conditions for output feedback control problems. *IEEE Trans. Automatic Control*, 44:1053–1057, 1999.

M. C. de Oliveira, J. Bernoussou, and J. C. Geromel. A new discrete-time robust stability condition. *System & Control Letters*, 4, 1999.

E. Fornasini and G. Marchesini. Doubly-indexed dynamical systems:state space models and structural properties. *Mathematical Systems Theory*, 12:59–72, 1978.

C. T. Freeman, A.-M. Hughes, J. H. Burridge, P. H. Chappell, P. L. Lewin, and E. Rogers. Iterative learning control of fes applied to the upper extremity for rehabilitation. *Control Engineering Practice*, 17:368–381, 2009.

G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *J. Soc. Indus. Appl. Math. Ser. B*, 2:205–224, 1964.

G. Golub and C. van Loan. *Matrix computations 3rd ed.* John Hopkins University Press, Baltimore, USA, 1996.

R. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

B. Grünbaum. Measures of the symmetry for convex sets. *Proc. of the Seventh Symposium in Pure Mathematics of the American Mathematical Society, Symposium on Convexity*, pages 233–270, 1961.

L. Hladowski, K. Galkowski, Z. Cai, E. Rogers, C. T. Freeman, and P. L. Lewin. A 2d systems approach to iterative learning control with experimental validation. *IFAC, Korea*, 2008.

L. Hladowski, K. Galkowski, Z. Cai, E. Rogers, C. T. Freeman, and P. L. Lewin. Output information based iterative learning control law design with experimental verification. *Journal of Dynamic Systems, Measurement and Control*, 134, 2012.

R. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2:18–21, 1973.

F. John. Extremum problems with inequalities as subsidiary conditions. In *Fritz John, Collected Papers*, pages 543–560. Birkhauser, Boston, Massachussetts, 1985.

I. Jolliffe. *Principal Component Analysis, 2nd edition.* Springer, New York, USA, 2002.

M. Kallay. Convex hull algorithms in higher dimensions. *Unpublished manuscript, Dept. Mathematics, Univ of Oklahoma,USA*, 1981.

R. Karnesky, C. Sudbrack, and D. Seidman. Best-fit ellipsoids of atom-probe tomographic data to study coalescence of $\gamma'(L1_2)$ precipitates in Ni-Al-Cr. *Scripta Materialia*, 57(4):353–356, 2007.

L. G. Khachiyan. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, 21:307–320, 1996.

D. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm. *SIAM Journal on Computing*, 15:287–299, 1986.

A. F. Kleptzyn, V. S. Kozyakin, M. Krasnoselskii, and N. A. Kuznetsov. Effects of small synchronization errors on complex systems. *Avtomatika i Telemekhanika. Translated in: Aut. and Remote control*, pages 1014–1018, 1984.

H. Klimo. Computing convex hulls in higher dimensions,. Master's thesis, Simon Fraser University, Canada, 1988.

V. S. Kozyakin. Asynchronous systems: a short survey and problems,. *Boole Centre for Research in Informatics, University College Cork - National University of Ireland*, Preprint No. 13/2003, 2003.

P. Kumar and E. Yildrim. Minimum-volume enclosing elliposids and core sets. *Journal Of Opt. Theory And App.*, 126:1–21, 2005.

J. E. Kurek and M. B. Zaremba. Iterative learning control synthesis based on 2-d system theory. *IEEE Transactions on Automatic Control*, 38:121–125, 1993.

J. H. Lee, W. H. Kwon, and J.-W. Lee. Quadratic stability and stabilization of linear systems with frobenius norm-bounded uncertainties. *IEEE Trans. Aut. Control*, 41 (3):453–456, 1996.

D.J. Leith, R.N. Shorten, W.E. Leithead, O. Manson, and P. Curran. Issues in the design of switched linear control systems: A benchmark study. *Int. Journal of Adaptive Control and Signal Processing*, pages 103–118, 2003.

Ji-Chang Lo and Ming-Long Lin. Robust h_∞ control for fuzzy systems with frobenius norm bounded uncertainties. *IEEE Trans. on Fuzzy Systems*, 14(1):1–15, 2006.

J. Lofberg. Yalmip: a toolbox for modeling and optimization in matlab. *IEEE International Symposium on Computer Aided Control System Design*, 2004.

C. Lorand. *A Theory of Synchronization Errors.* PhD thesis, University of Notre Dame, USA, 2004.

C. Lorand and P. Bauer. Distributed discrete time systems with synchronization errors: Models and stability. *IEEE Trans on Circ & Syst., CAS-II Express Briefs*, 52(4): 208–213, 2005.

C. Lorand and P. Bauer. Clock synchronization errors in circuits: Models, stability and fault detection. *IEEE Transaction on Circ. and Syst., CAS-I Regular papers*, 53(10), 2006a.

C. Lorand and P. Bauer. On synchronization errors in networked feedback systems. *IEEE Trans. on Circuits and Systems - CAS-I Regular Papers*, 53(10), 2006b.

A. Nemirovski and P. Gahinet. The projective method for solving linear matrix inequalities. *American Control Conference*, pages 840–844, 1994.

Yu. E. Nesterov and A. S. Nemirovski. A general approach to polynomial-time algorithms design for convex programming. *Technical report, Centr. Econ. & Math. Inst., USSR Acad. Sci.,Moscow*, 1988.

Yu. E. Nesterov and A. S. Nemirovski. Interior-point polynomial methods in convex programming. *SIAM Publications, Philadelphia*, 1994.

M. C. Oliveira, J. C. Geromel, and J. Bernussou. Extended H_2 and H_∞ norm characterizations and controller parametrizations for discrete-time systems. *Int. J. of Control*, 75(9), 2002.

W. Paszke. *Analysis and synthesis of multidimensional system classes using linear matrix inequalities.* PhD thesis, Zielona Gora, Poland, 2002.

J. W. Polderman and J. C. Willems. *Introduction to mathematical systems theory.* Springer, New York, 1998.

R. Prokop and A. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *Graphical Models and Image Processing*, 54(5), 1992.

L. Rocha, L. Velho, P. Cezar, and P. Carvalho. Image moments-based structuring and tracking of objects. *XV Brazilian Symposium on Computer Graphics and Image Processing*, 2002.

R. P. Roesser. A discrete state space model for linear image processing. *IEEE Trans. Automatic Control*, 20:1–10, 1975.

E. Rogers, K. Galkowski, and D. H. Owens. Control systems theory and applications for linear repetitive processes. *In Lecture Notes in control and information sciences*, 349, 2007.

A.P. Schoellig, J. Alonso-Mora, and R. D'Andrea. Independent vs. joint estimation in multi-agent iterative learning control. *49th IEEE Conference on Decision and Control*, pages 6949–6954, 2010.

B. W. Silverman and D. M. Titterington. Minimum covering ellipses. *SIAM Journal on Scientific and Statistical Computing*, 1:401–409, 1980.

D. Sommerville. *An Introduction to the Geometry of N Dimensions*. Dover Press, 1958.

J. Sturm. *Primal-Dual Interior Point Approach to Semidefinite Programming*. PhD thesis, Tilburg University, 1997.

J. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999.

P. Sun and R. M. Freund. Computation of minimum volume covering ellipsoids. *Operations Research*, 52:690–706, 2004.

C. van Loan and N. Pitsianis. Approximation with the kronecker products. *Linear Algebra for Large Scale and Real Time Applications, Kluwer Publications*, pages 293–314, 1993.

J. C. Willems. Paradigms and puzzles in the theory of dynamical systems. *IEEE Trans. Automat. Control*, 36(3):259–294, 1991.

J. C. Willems. The behavioral approach to open and interconnected systems. *Control Systems Magazine*, 27:46–99, 2007.

J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. M. de Moor. A note on persistency of excitation. *Systems and Control Letters*, 54:325–329, 2005.

V. A. Yakubovich. The solution of certain matrix inequalities in automatic control theory. *Soviet. Math. Dokl.*, 3:620–623, 1962.

V. A. Yakubovich. Solution of certain matrix inequalities encountered in nonlinear control theory. *Soviet. Math. Dokl.*, 5:652–656, 1964.

V. A. Yakubovich. The method of matrix inequalities in the stability theory of nonlinear control systems, I,II,III. *Automation and Remote Control*, 25–26:905–917,577–592,753–763, 1967.

L. You and F. Gao. Robust H_∞ control of discrete-time linear systems with delayed state and frobenius norm-bounded uncertainties. *Proc. 39th IEEE Conf. on Decision and Control*, 3:2754–2755, 2000.