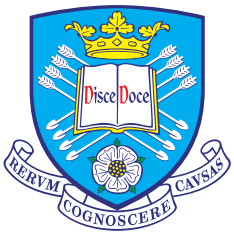

The University Of Sheffield.

Distributed Model-based Control for Gas Turbine Engines

University Technology Centre sponsored by Rolls-Royce

Author: Romain Guicherd

PhD Thesis

A dissertation submitted for the degree of
Doctor of Philosophy

Department of Automatic Control & Systems Engineering
University of Sheffield
United Kingdom

April, 2019

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this report are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

"Imagination is more important than knowledge"

Albert Einstein

Abstract

Controlling a gas turbine engine is a fascinating problem. As one of the most complex systems developed, it relies on thermodynamics, fluid mechanics, materials science as well as electrical, control and systems engineering. The evolution of gas turbine engines is marked with an increase in the number of actuators. Naturally, this increase in actuation capability has also been followed by the improvement of other technologies such as advanced high-temperature and lighter materials, improving the efficiency of the aero engines by extending their physical limits. An improvement in the way to control the engine has to be undertaken in order for these technological improvements to be fully harnessed. This starts with the selection of a novel control system architecture and is followed by the design of new control techniques.

Model-based control methods relying on distributed architectures have been studied in the past for their ability to handle constraints and to provide optimal control strategies. Applying them to gas turbine engines is interesting for three main reasons. First of all, distributed control architectures provide greater modularity during the design than centralized control architectures. Secondly, they can reduce the life cycle costs linked to both the fuel burnt and the maintenance by bringing optimal control decisions. Finally, distributing the control actions can increase flight safety through improved robustness as well as fault tolerance.

This thesis is concerned with the optimal selection of a distributed control system architecture that minimizes the number of subsystem to subsystem interactions. The control system architecture problem is formulated as a binary integer linear programming problem where cuts are added to remove the uncontrollable partitions obtained. Then a supervised-distributed control technique is presented whereby a supervisory agent optimizes the joint communication and system performance metrics periodically. This online optimal technique is cast as a semi-definite programming problem including a bilinear matrix equality and solved using an alternate convex search. Finally, an extension of this online optimal control technique is presented for non-linear systems modelled by linear parameter-varying models.

Keywords: distributed control, model-based control, optimal control, convex optimization, gas turbine engine.

Acknowledgements

First of all, I would like to give a special thanks to my supervisory team, especially Professor Visakan Kadirkamanathan for his valuable support and guidance throughout my research as well as for his general availability.

Nothing would have been possible without the help of Dr Paul Trodden, who was always there to answer my technical questions and to suggest new ideas and research directions. His support resulted in a large number of very helpful meetings and complex questions that has shaped my research.

Special thanks to Dr Andrew Mills for his help and more applied angle in adequation with industrial research. This allowed me to get an early perspective on the PhD as well as a broader view concerning the industrial aspects of gas turbine engine control.

I am also very grateful to the members of the Rolls-Royce University Technology Centre for the regular group meetings and the stimulating conversations. These always sparked curiosity with regards to new research topics, contributing to making my stay very enjoyable. Being in contact with Rolls-Royce was essential, primarily to bring an industrial depth to this research project and, in particular, to access research directions coming from real engineering problems.

Last but not least, I would like to thank my family for their constant support during my research as well as the precious motivation and encouragement they have always given me.

Contents

List of Figures

List of Tables

Glossary

Abbreviation	Description	Page
BILP	Binary Integer Linear Programming	56, 80, 87, 96
BMI	Bilinear Matrix Inequality	57, 107, 149, 171
CP	Cone Programming	54, 56
EEC	Electronic Engine Controller	6, 7
EPR	Engine Pressure Ratio	3
FADEC	Full Authority Digital Engine Controller	6, 7
FLOPS	Floating Point Operations Per Second	137, 138
GP	Geometric Programming	55
GTE	Gas Turbine Engine	1, 2, 4, 7, 8
ILP	Integer Linear Programming	56, 77
LMI	Linear Matrix Inequality	53, 54, 57, 114, 121, 125, 130–133, 135–138, 149, 151, 155–157, 162, 163, 165, 166
LP	Linear Programming	49–52, 56, 58, 59, 66, 96

Abbreviation	Description	Page
LPV	Linear Parameter-varying	xvii, 23, 24, 147–155, 157–162, 164–166, 168, 170, 171
LQR	Linear Quadratic Regulator	24, 26–29, 109, 116
LTI	Linear Time-invariant	16, 27, 117
MILP	Mixed Integer Linear Programming	56
MPC	Model Predictive Control	27–30, 34, 67
NP	Non-deterministic Polynomial	89, 91, 112, 197, 199
P	Polynomial Deterministic	197
PID	Proportional Derivative Integral	7, 8, 29
QCQP	Quadratic Constraint Quadratic Programming	52
QP	Quadratic Programming	28, 51, 52, 56
SDP	Semi-definite Programming	52, 53, 56, 57, 107, 117, 126, 137–140, 149, 153, 154, 158, 171
SOCP	Second-order Cone Programming	52, 56

Notation

Scalar Sets

$\mathbb{N}$	Set of natural numbers
$\mathbb{N}^*$	Set of natural numbers, zero excluded
$[\![a, b]\!]$	Set of integers $\{a, \dots, b\}$
$\mathbb{Z}$	Set of integers
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^*$	Set of real numbers, zero excluded
$\mathbb{R}_+$	Set of positive real numbers
$\mathbb{R}_+^*$	Set of strictly positive real numbers
$\mathbb{C}$	Set of complex numbers
$\mathbb{C}_+$	Set of complex numbers with positive real part
$\mathbb{C}_+^*$	Set of complex numbers with strictly positive real part
$\mathbb{R}^n$	Set of real vectors of dimension n
$\mathbb{R}^{n\times m}$	Set of real matrices of dimension $n \times m$
$\mathbb{S}^n$	Set of symmetric matrices of dimension n
$\mathbb{S}_+^n$	Set of symmetric positive semi-definite matrices of dimension n
$\mathbb{S}_{++}^n$	Set of symmetric positive definite matrices of dimension n

Generalized Inequalities

$x \geq y$	Element-wise inequality between x and y
$x > y$	Strict element-wise inequality between x and y
$X \succeq 0$	X is positive semi-definite: $X \in \mathbb{S}_+^n$
$X \succ 0$	X is positive definite: $X \in \mathbb{S}_{++}^n$
$X \succeq Y$	Matrix inequality between symmetric matrices: $(X - Y) \in \mathbb{S}_+^n$
$X \succ Y$	Strict matrix inequality between symmetric matrices: $(X - Y) \in \mathbb{S}_{++}^n$
$X \succeq_K Y$	Conic inequality between X and Y: $(X - Y) \in K$

$X \succ_K Y$	Strict conic inequality between X and Y: $(X - Y) \in \operatorname{int} K$

Norms

$\|z\|$	The magnitude of a complex number z
$\|x\|_0$	l_0-norm operator applied to the vector x
$\|x\|_1$	l_1-norm operator applied to the vector x
$\|x\|_2$	l_2-norm operator applied to the vector x
$\|x\|_Q$	Q-weighted l_2-norm operator applied to the vector x
$\|x\|_\infty$	l_∞-norm operator applied to the vector x
$\|X\|_2$	Spectral norm operator applied to the matrix X

Vectors & Matrices

e_i	i^{th} canonical unit vector of appropriate dimension
I_n	Identity matrix in $\mathbb{R}^{n \times n}$
$0_{n \times m}$	Zero matrix of dimension $n \times m$, used without subscripts
$A^\top$	Transpose of the matrix A
$\operatorname{rank}(A)$	Rank of the matrix A
$\operatorname{tr}(A)$	Trace of the matrix A
$\operatorname{vec}(A)$	Vectorization of the matrix A
$\lambda_{max}(A), \lambda_{min}(A)$	Maximum, minimum eigenvalue of the matrix A
$\sigma_{max}(A), \sigma_{min}(A)$	Maximum, minimum singular value of the matrix A
$\|A\|$	The magnitude operator is applied to each element of A
$A \circ B$	Hadamard product of two matrices A and B
$[A\|B]$	Horizontal concatenation of the matrices A and B

Set Operations

$S_1 \cup S_2$	Union of sets S_1 and S_2
$S_1 \cap S_2$	Intersection of sets S_1 and S_2
$S_1 \times S_2$	Cartesian product of sets S_1 and S_2
$S_1 \oplus S_2$	Minkowski sum of sets S_1 and S_2
$S_1 \ominus S_2$	Pontryagin difference of sets S_1 and S_2

Topology & Convex Analysis

$\operatorname{card} S$	Cardinality of a set S
$\operatorname{co} S$	Convex hull of a set S
$\operatorname{int} S$	Interior of a set S

Combinatorial Functions

$n!$	Factorial of a natural number n

$x^{(n)}$	Rising factorial
$(x)_n$	Falling factorial
$\binom{n}{k}$	Binomial coefficient n choose k
$\begin{bmatrix} n \\ k \end{bmatrix}$	Stirling number of the first kind
$\begin{Bmatrix} n \\ k \end{Bmatrix}$	Stirling number of the second kind
$\left\lfloor \begin{matrix} n \\ k \end{matrix} \right\rfloor$	Lah numbers
B_n	Bell number
δ_{ij}	Kronecker delta function

Functions & Derivatives

$\operatorname{dom} f$	Domain of a function f
$\operatorname{epi} f$	Epigraph of a function f
∇f	Derivative of a function f

"Dīvide et imperā"

Philip of Macedonia

Chapter 1

Introduction

During the past few years air travel has become more accessible, triggering an increase in the number of flights as well as in the number of passengers. For instance, during the year 2014 alone, 3.1 billion people flew worldwide representing a total of approximately 31 million flights. Every year these numbers increase, with this tendency, 6 billion passengers are expected to travel by 2030. Therefore, a corresponding increase in fuel consumption as well as in the emissions is expected. Nonetheless, it is interesting to note that the fuel consumption, per passenger per kilometre flown, has consistently been decreasing (Benito and Alonso, 2018). This trend is due to the use of new technologies in civil aviation, bringing a reduction in aircraft weight as well as a better engine fuel and propulsive efficiency for Gas Turbine Engine (GTE). The year 2014 marked an important milestone as the centenary of commercial flight operations (Airbus, 2014), thus a signifiant data history has been gathered, starting to reveal the trends in the aerospace sector. Also, even if air transportation is the main application for gas turbine engines it is far from being the only one. Indeed, marine power generation, train power systems and other means of transportation also take advantage of the high power density of gas turbine engines in order to produce energy, mainly under electrical form. Consequently, it is very likely that gas turbine engines will prosper in the future and keep following the same technological evolution as before. The following three paragraphs presents the three main features of gas turbine engines that are at the heart of this research project.

An aircraft gas turbine engine has to deliver thrust as well as power under various forms, including electrical, pneumatic and hydraulic. In order to achieve such a goal in an efficient manner an engine has to be equipped with multiple built-in subsystems (Richter, 2012). Firstly, a gas turbine engine is composed of different modules representing the thermodynamic processes acting on the fluid mass flow

later ejected at high velocity to generate thrust (Linke-Diesinger, 2008). Secondly, other accessory systems such as pumps and electrical generators, are connected to one of the shafts of the gas turbine engine in order to generate energy under electrical and hydraulic form. Finally, fluid subsystems are also added to ensure the safe handling of the engine and to allow for hot air extraction. The hot air is used for purposes such as de-icing, cabin climate control and pneumatic actuation. It is therefore appropriate possible to describe a GTE as a set of integrated subsystems that together form what is called a power plant. The subsystem organization of an aero power plant ranks gas turbine engines in the category of complex medium-scale systems. Subsequently, the design of gas turbine engine control laws has to take into account the interactions between all subsystems, and control the power plant as a whole. Thus, the first control design step is to select the control system architecture appropriately, so that it well describes the system dynamics, and in particular accounts for the subsystem to subsystem dynamical interactions. Currently engine control units are fully centralized and new control system architectures constitute an important research topic.

Gas turbine engines use ever more advanced technologies, and as a result, they become more efficient at extracting the chemical energy from the fuel. These new technologies include for example the development of lighter and high temperature materials, as well as new techniques like the geared fan design or the variable nozzle area. The direct consequence of this technological development is a decrease in fuel consumption but also of the emissions, such as the pollution and the noise as well as the maintenance burden. However, the drawback of using more advanced technologies is that more complexity is added to the engine design. Future generations of engine will also include more sensors and actuators, and thus it will allow the engine manufacturers to control more engine parameters (Jaw and Mattingly, 2009). This trend of increasing the number of engine control variables allows gas turbine engines to be run closer to their structural and thermal limits, which translates to having more degrees of freedom in the choice of control input variables, and therefore to perform closer to the optimal operating point. Subsequently, new control laws need to be developed to fully harness the increase in gas turbine engine controllability and to provide the best performance and reliability possible. Based on the control system architecture selected the control law design can be completed. In the case where the control system architecture is not fully centralized, the control law design will also include a communication scheme between the local controllers in charge of controlling subsets of the gas turbine.

Finally, gas turbine engines are based on thermodynamic processes involving an open Joules-Brayton cycle (Richter, 2012). This cycle relies on the fact that

chemical energy is released as heat in the combustion chamber, and then converted into mechanical work when driving the turbine modules and producing thrust. The efficiency of this thermodynamic cycle is increased by mechanically compressing the working fluid beforehand, within the compressor stages. The thermodynamic efficiency of a gas turbine engine is calculated as a function of the Engine Pressure Ratio (EPR), the ratio of the turbine discharge pressure and the compressor inlet pressure. The higher the EPR, the better the thermodynamic efficiency of the engine. Since gas turbine engines are subject to complex phenomena such as the behaviour of the dynamics in the compressor and turbine stages, they are highly non-linear systems. Therefore, modelling a gas turbine engine is usually done by linearizing the system dynamics at multiple operating conditions, which yields a set of linear models covering the operating envelope (Balas, 2002). Then the control laws are designed based on gain scheduling techniques, in order to adapt the controller to the changing system dynamics. Subsequently, designing new gas turbine control laws must be performed accounting for the non-linearity of the gas turbine engine dynamics as well as the newly design control system architecture.

This thesis is concerned with the three points presented previously. First of all, the selection of a control system architecture is considered. This task can be achieved through the partitioning of the gas turbine system model into overlapping or non-overlapping controllable subsystems. Each subsystem is then fitted with a local controller. Secondly, the control law design can be performed based on the obtained system partitioning. This design is performed optimally online by minimizing an objective function that represents the total energy of the system. Finally, the last aspect presented within this thesis tackles the control law design for a special type of non-linear systems, modelled by a linear parameter-varying model.

In order to understand the context of this research project, a literature review presenting the state of the art in control system is included after the introduction. This chapter is followed by a chapter introducing the mathematical background necessary to the understanding of the following chapters. Finally, a conclusion summarizes the main contributions of this thesis and proposes some future research directions. The remainder of this introduction includes a summary of the gas turbine engine history as well as of the engine control development phases. Therefore, it provides the necessary motivations for the different research problems raised during this project. This introduction concludes with a detailed presentation of the content of the following thesis chapters as well as how they are articulated together.

1.1 History of Gas Turbine Engines

The invention of the GTE was a huge leap forward and was a key contribution to the development of aviation, especially in the civil sector. This innovation resulted from the technological race between Sir Frank Whittle in Britain and Hans von Ohain in Germany, working independently on similar technologies (Whittle, 1945). The very first gas turbine engine powered flight took place in 1939 in Germany with the Heinkel He 178 aircraft. At the very early stages of the engine development, the goal was simply to control the thrust of the gas turbine using a single input, the fuel flow rate. The main initial issues were to maintain a controlled combustion as well as to prevent the combustion chamber from overheating. Very quickly engineers realized that they needed more control input variables in order to improve the engine handling capabilities as well as to operate them safely. For instance, protecting the gas turbines against surging and flaming out is achieved by enforcing limits to the control variables, such as a minimal fuel flow rate to avoid flaming out, and a maximal fuel flow rate to prevent the engine from over heating. A GTE is designed to operate within a control envelope that is defined by structural and thermal boundaries, where the engine is kept undamaged and remains under control. As the main source of propulsive power, gas turbine engines have to provide thrust safely over the entire aircraft flight envelope. Therefore, from the very start, the development of gas turbine engines has been linked to the advancement of control system technologies, and this trend still continues today.

In the past few years engineers introduced further control inputs, not only to protect the GTE, but also to increase its efficiency and to allow for greater operating ranges. Controlling an engine is crucial in providing safe operation, but it can also be used to run the engine with greater efficiency and thus to reduce the operating life cycle costs. More controllability allows the engine to be operated closer to its optimal performance. Hence, it goes with a reduction of fuel burnt, emissions and maintenance cost, and it triggers an increase in engine life.

Being able to reach higher thrusts with less fuel burnt makes it possible to carry more passengers for the same cost. The project *Clean Sky* is the major European project that aims at reducing emissions in civil aviation. *Clean Sky*'s projects are helping to dramatically slash the air industry's carbon dioxide, noise and nitrous oxide footprints by developing new engine architectures (such as the open rotor), improved wing aerodynamics, lighter composite structures, smarter aircraft trajectories, and more electric on-board energy. Therefore, an improvement of the engine control architecture as well as the control laws will help to support this goal.

1.2 History of Gas Turbine Engine Control

The development phases of the gas turbine engine controllers can be decomposed into four distinct periods. These periods are leading to the current integration and advanced control era that represents the current era of engine control (Jaw and Mattingly, 2009). Each of these periods are one or two decades long and are presented in this section.

1.2.1 Infancy Period

The initial engine control period saw the development of mechanical control devices. During that time all the control actions were performed by gear trains, cams and linkages. The design of the engine controller was based on a frequency domain analysis, relying on Bode plots in order to achieve the desired gain and phase margins. By the end of this period, the development of hydromechanical systems combined with the first tube based electronic controllers was undertaken. Nonetheless, electronic engine controllers offered poor performance during these early stages due to their inability to cope with harsh operating environments. Finally, most of the performance studies at this period was focused on the steady state analysis. This was due to the complexity to perform efficient and fast computations, and therefore only first order shaft dynamic modelling could be used.

1.2.2 Growth Period

During the growth period, the engine control techniques such as hydromechanical control were still used. However, this period was marked by the introduction of new control variables such as geometry control for the compressor, the intake and the nozzle. The design of control laws during this time relied on classical control techniques, using multiple control loops designed by considering that other variables remained around their steady state values. The control loops were closed one after the other using the successive loop-closure technique, accounting for dynamical interactions. Soon after, state space models started to be developed for optimal control purposes, but were not yet embedded in engine controllers. Finally, this period saw the use of Newton techniques as well as computer simulations to evaluate the transient engine performance.

1.2.3 Electronic Period

The hydromechanical units reached a limit to control the engines, since much larger and heavier units were needed to fulfil all the requirements. A more practical solu-

tion was to use an Electronic Engine Controller (EEC) to provide supervisory and trim control functions. During that time the electronic engine controllers were still coupled with a hydromechanical unit. This unit is more conservative but used only as a backup controller. Engineers quickly realized that a digital engine controller offered significantly more flexibility with regards to modifications and updates when compared to the previous hydromechanical units. Also, electronic engine controllers reduced the development cycle time and provided more functionality allowing, for example, to record engine data for health monitoring purposes. This period also brought more attention on multivariable control techniques as well as real time engine models.

1.2.4 Integration Period

During the past three decades the use of Full Authority Digital Engine Controllers (FADECs) has become a standard technique for engine controllers. This control unit is lighter and smaller than the previous EEC and includes a dual-channel architecture for redundancy as well as more built-in test functions. During this period, multivariable control techniques were implemented based on the previous studies conducted. An engine model is often embedded in the FADEC to improve the control performance and provide health monitoring capabilities. The integration of the flight control system with the propulsion control system is also achieved, improving the aircraft manoeuvrability and subsequently giving a tactical advantage in the military. This integration usually relies on vectorized thrust. In addition to this, model-based control techniques are implemented to benefit from the increase in control and output variables. In addition to this, the introduction of technologies such as smart wireless sensors allowed the sensing of remote parameters in the engine and to communicate them wirelessly on-board as well as to ground stations.

1.3 Control System Architecture Research Direction

The standard control system architecture implemented since the use of electronic engine controllers is the dual-channel centralized architecture. The engine controller is composed of a single unit where two identical channels are embedded. Each channel is a control computer fully capable of controlling the gas turbine engine. These controllers are named FADEC where full authority implies that the control is applied from engine start-up to engine shut-down and that they are in charge of all the flight phases without any manual override possible. In order to reach the safety level required, and to decrease the probability of failure per flight hours to meet the certification level. The dual-channel architecture is necessary, the engine

sensors and actuators are all linked directly to one or two of the FADEC channels, for redundancy purposes. The centralized architecture of the EEC comes with an engine health monitoring system. The aim is not only to monitor the health of the engine but also to gather data about all the engines of the fleet to perform statistical analyses. Digital engine controllers bring significant advantages, such as a reduction in specific fuel consumption, weight reduction, as well as a greater operation safety compared to former engine control techniques (Sobey and Suggs, 1963). Also, it reduces the workload for the flight crew, allowing the removal of the flight engineer from the flight deck.

The current research is focused on redesigning the engine control architecture to move towards a more distributed control system (Merrill et al., 2010). The purpose of a distributed control architecture is to take advantage of the great number of control parameters and subsystems to provide a more efficient and safer control architecture (Thompson et al., 1999). For instance a FADEC relying on a centralized architecture is not protected against a single point of failure, due to the lack of physical segregation. On the contrary, distributed architectures are more robust to the failure of one or multiple local controllers. Also, the centralized architecture suffer from a lack of modularity. Subsequently, with the increase in complexity of gas turbine engines, a lot of resources are required to design, test and certify every new centralized FADEC unit. Hence, distributing the control actions will allow to decrease the engine overall life cycle costs.

A modification of the engine control system starts with the partitioning of the system into joint, or disjoint, pairs of sets of state and input variables. The architectural problem is then reduced to the optimization of these subsets of system variables. Thus, computing the subsystems can be cast as an optimization problem such that the amount of interactions between all the subsystems is minimized while all the subsystems stay controllable. Such an optimization problem belongs to the class of combinatorial optimization and its formulation requires the use of integer variables.

1.4 Control Method Research Direction

So far only classical control methods have been applied for engine control, this includes control laws such as Proportional Derivative Integral (PID). They provide control commands to keep good safety margins, sometimes at the expense of the specific fuel consumption. These controllers are preceded by an operating scheduler that modifies the control gains as a function of the flight conditions. In order to provide protections for the GTE and to avoid exceeding the physical system bound-

aries, minima and maxima selectors are added to the controller structure (Austin Spang and Brown, 1999). The standard engine controllers can be divided in two parts. The first part consists of a steady state regulator that aims at maintaining the GTE at a desired set-point. Naturally, the second part is in charge of the transient phases. It controls the engine during accelerations and decelerations occurring after a pilot input is given (Thompson, 1992). Following this, the control is handed back to one of the steady state regulators. This standard controller ensures that the engine performs with an acceptable efficiency and that the engine parameters remain within the required boundaries. This research project aims at giving a framework for distributed model-based control applicable to complex medium-scale systems such as gas turbine engines. The main feature of this research project is to provide an algorithm to optimize the subsystem to subsystem communication jointly with the overall system performance as a function of the system state variables.

Gas turbine engines have evolved a lot since their creation, with new materials and technologies developed continuously. This evolution brings new control input variables that can be used to steer the GTE state variables more efficiently. However, including this capability will require the use of online optimization and will trigger an increase in the control design complexity. Controlling a system based on its dynamic model allows the controller to tailor the control actions to the plant behaviour as well as to predict its future state variables. Predictive controllers minimize an objective function, which is achieved by solving an optimization problem online over a prediction horizon. Therefore, the future state and input variable discrepancies, based on the system model, are minimized when the controller computes the next control input sequence. In contrast PID controllers use the error already perpetrated to control the system. Model-based control techniques provide a better way to handle the constraints compared to classical controllers. Distributing the control actions with the use of a model-based controller will allow the reduction of the total weight of the control system and increase its modularity (Eren et al., 2017).

These aims are achieved by formulating the distributed control problem as an online optimization that will periodically update the control laws of the subsystems as a function of their state variables. The different control modes used include the inter subsystem communication. The last point will be to evaluate the impact on system performance of different communication strategies. The distributed model-based control technique developed can be applied in other domains such as energy distribution networks or even in marine systems and automotive systems (Seok et al., 2017).

1.5 Thesis Structure

The thesis presents a literature review as well a mathematical background chapter in order to be able to facilitate the understanding of the following research outputs. After that, the main contributions of the research undertaken are articulated in three research chapters. These research chapters could be read as independent research contributions or linearly as a continuum. The remainder of this section presents a summary of what is included in each of the remaining chapters and a diagram highlighting the main thesis structure as well as the interconnections between chapters is provided.

Chapter 2: The second chapter presents a literature review in system and control theory. It introduces the notations used throughout the thesis along with the important properties developed in the system and control engineering field. Then, these properties are used to describe the concept of optimal control. First, the notion of infinite horizon linear quadratic regulator is introduced as a precursor of online optimal control. Following this, the idea behind model-based predictive control is presented along with its main features. This chapter concludes with the presentation of the different control system architectures and a summary of the notions presented, therefore motivating the research undertaken within the next chapters.

Chapter 3: In this chapter the mathematical background necessary for the understanding of the rest of the thesis is provided. It defines convex sets and convex functions along with their fundamental properties and illustrate them with fundamental examples. All the theoretical results developed are then combined in order to introduce the convex optimization framework. These critical tools of modern system and control theory are essential to the understanding of optimal control architecture selection and control method design. In addition, this chapter defines some common non-convex optimization problems along with the main techniques to solve them. Finally, the major methods and algorithms used to solve the optimization problems presented previously are discussed before a summary is provided.

Chapter 4: The control system architecture problem is tackled within this chapter. The partitioning of a system model will condition the structure of the controller as well as its design. In order to partition a system model, one has to know what states and inputs to group together to define subsystem models. The subsets of state and input variables describing the subsystems can be non-overlapping or overlapping when some state variables are shared between multiple subsystems. For a given partitioning, the total magnitude of the interactions between subsystem mod-

els is evaluated. Therefore, the partitioning problem seeking for weak interactions can be posed as a minimization problem. Initially, the problem is formulated as a non-linear integer minimization that is then relaxed into a linear integer programming problem. It is shown within this chapter that cuts can be applied to the initial search space in order to find the least interacting partitioning; only composed of controllable subsystems. The complexity of the partitioning algorithm is evaluated and a few examples are included in order to demonstrate the partitioning method.

Chapter 5: This chapter is concerned with the control of systems composed of multiple coupled subsystems. In such architectures, communication between different local controllers is desired in order to achieve a better overall control performance. Any resultant improvement in control performance needs, however, to be significant enough to warrant the additional design complexity and higher energy consumption and costs associated with introducing communication channels between local controllers. A practical distributed control design aims, therefore, to achieve an acceptable balance between minimizing the use of communication between controllers and maximizing the system-wide performance. In this chapter, a new approach to the problem of synthesizing stabilizing distributed control laws for discrete time linear systems that balances performance and communication is presented. The approach employs a supervisory agent that, periodically albeit not necessarily at every sampling instant, solves an optimization problem in order to synthesize a stabilizing state feedback control law for the system. The online optimization problem, which maximizes sparsity of the control law while minimizing an infinite-horizon performance cost, is formulated as a bilinear matrix inequality problem; subsequently, it is relaxed to a linear matrix inequality problem, and (i) convergence to a solution as well as (ii) that early termination guarantees a feasible (but suboptimal) control law are proved. Stability of the closed-loop system under what is a switched control law is guaranteed by the inclusion of dwell time constraint in the optimization problem. Finally, the efficacy of the approach is demonstrated through numerical simulation examples.

Chapter 6: The control design of non-linear systems modelled as discrete time linear parameter-varying systems is treated within this chapter. For such systems, the dynamical model of the system is varying within a compact set based on the value of an exogenous scheduling parameter, not controllable but measurable in real time. In the past, multiple techniques have been used to control this type of systems, for instance an optimal robust feedback control method has been studied based on a feedback gain robust to plant uncertainty. However, very few research approaches

have considered the synthesis of structured linear-parameter varying controller implemented in a distributed framework. This chapter considers the offline synthesis of a set of linear-parameter varying control laws, relying on different communication topologies and presenting different control performance. Then a supervisory unit selects online and periodically the best control mode candidate from the finite set of control laws computed offline. The controller selection is performed in order to maximize the system-wide performance while minimizing the subsystem to subsystem communication burden. The control law synthesis is formulated as a semi-definite programming problem with convex structural constraints, when the system is partitioned in non-overlapping subsystems and as a bilinear matrix inequality problem when some state variables are shared between different subsystems. Then online, the best control mode is selected based on the predicted control and communication costs amongst the stable switchable controller. A numerical example is provided to illustrate the effectiveness of the distributed control strategy.

Chapter 7: The main contributions presented within this thesis are summarized in this last chapter. The conclusion highlights the connections between the different techniques developed. Finally, some future work and research directions are proposed, based on the work presented within the previous chapters.

The reader will be able to find some secondary results in the appendices, they provide deeper explanations about some specific points used within the chapters of this thesis without altering the overall understanding. A bibliography at the end of the thesis contains all the references that were used in order to perform the research work, they are necessary to fully understand the thesis content as well as to get more perspective on it. This thesis is linked to published work and planned publications, the list of publications is as follows:

- Guicherd, R., Trodden, P. A., Mills, A. R., and Kadirkamanathan, V. (2017). Weak interactions based system partitioning using integer linear programming. *IFAC-PapersOnLine*, 50:3698–3704

- Guicherd, R., Trodden, P. A., Mills, A. R., and Kadirkamanathan, V. (2019). Supervised-distributed Control with Joint Performance and Communication Optimization. *International Journal of Control*. In Preparation

- Guicherd, R., Trodden, P. A., Mills, A. R., and Kadirkamanathan, V. (2020). Supervised-distributed Control with Joint Performance and Communication Optimization for LPV Systems. *Automatica*. In Preparation

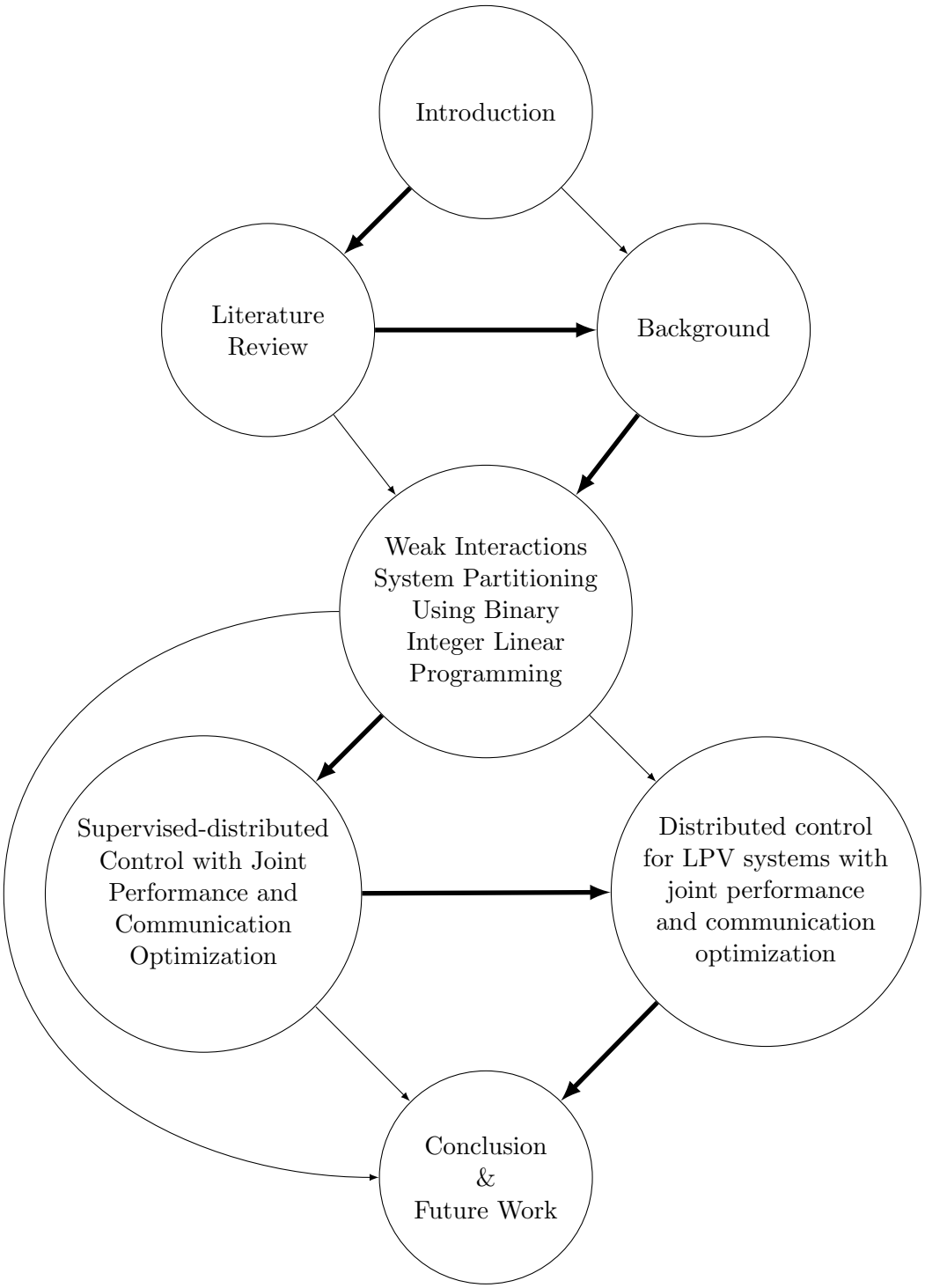

Figure 1.1: Thesis structure diagram presenting the main sequence of chapters and the interconnections between non-consecutive chapters respectively with the bold and thin arrows.

Chapter 2

Literature Review

2.1 Introduction

Control and system theory is concerned with the design of controllers, implemented in order to influence the behaviour of physical systems and therefore to achieve a given aim. More specifically control systems are used to regulate the controlled variables of a system to a fixed value or to provide tracking capabilities for these variables. Regulation is used to keep a variable at a given static set point, whereas tracking is used in order for a controlled variable to follow a specific time varying trajectory. Control system is far from being a new topic, indeed it goes back more than two millennia ago with one of the very first control system developed by Ktesibios that aimed at controlling the liquid flow within water clocks, subsequently improving their accuracy (Mayr, 1970). Most of the mechanical clocks designed after that were relying on a mechanical feedback control system (Lepschy et al., 1992). The Romans implemented mechanical control systems in order to regulate the water levels within the aqueducts throughout the use of a set of valves (Sontag, 1998). Also, the use of feedback control has been increasing more recently in history since the industrial revolution. At that time, mechanical control systems were the key to harness power from steam engines, they allowed rotational speed regulation. The control of systems is made possible through the knowledge of system engineering. System engineering deals with the design and the understanding of dynamical systems based on mathematical modelling. The term systems is used across a multitude of fields and it includes for example systems that are electrical, mechanical, chemical but also economical and biological. Consequently, control system engineering is a multidisciplinary topic, applicable in most of the other engineering fields and more (Aström and Kumar, 2014). Control theory is studied as a field of applied mathematics, hence providing theoretical tools for the design of control systems (Sontag,

1998). This chapter gives the necessary definitions about dynamical systems before introducing the different control techniques along with system stability. Finally, it introduces the different control system architectures before finishing by explaining the gaps in the literature and by presenting the three main contributions presented later within the chapter of this thesis.

2.2 Dynamical Systems

Control systems are applied to dynamical systems that are modeled usually from first principles. The mathematical models used to represent these physical systems can be linear or non-linear as well as continuous or discrete. These mathematical models developed are fundamental to the understanding of the system and subsequently to the design of the control systems (Bay, 1998). This section defines these different types of system models along with their characteristics.

2.2.1 Linear Systems

One of the simplest and most widely used system model is the linear system model. The mathematical models representing dynamical systems are expressed in the time domain using a differential or difference system of linear equations or in the frequency domain with a transfer function. When the dynamical system is represented in the time domain, the model can be formulated in continuous time or discrete time. First, the continuous time linear model is introduced before the discrete time one is presented.

Continuous Dynamical Systems

In the continuous time case, dynamical systems can be represented by a continuous time differential system of linear equations as follows,

$$\dot{x}(t) = A(t)x + B(t)u(t) \tag{2.1a}$$

$$y(t) = C(t)x(t) + D(t)u(t), \tag{2.1b}$$

where, $t \in \mathbb{R}$ denotes the time, the vector $x \in \mathbb{R}^n$ represents the state variable, the vector $u \in \mathbb{R}^m$ is the input or control variable and finally $y \in \mathbb{R}^p$ represents the output variables, with n, m and p three strictly positive integers. The matrices $A(t)$, $B(t)$, $C(t)$ and $D(t)$ are of appropriate dimensions and are respectively the state, input, output and feedthrough matrices. This mathematical representation of a continuous linear dynamical system is called a state space representation, it

constitutes a general model for continuous time linear dynamical systems. The state, input and output variables are all vector functions of time. Based on the state space model as well as a fixed initial state $x(t_0)$, the trajectory of the state variables can be computed as follows,

$$\forall t \in [t_0, +\infty[,\ x(t) = \Phi(t, t_0)x(t_0) + \int_{t_0}^{t} \Phi(t, \tau)B(\tau)u(\tau)d\tau, \tag{2.2}$$

where, the matrix function $\Phi(t, \tau)$ is called the state transition matrix and complies with the following two properties,

$$\dot{\Phi}(t, \tau) = A(t)\Phi(t, \tau) \tag{2.3a}$$

$$\Phi(\tau, \tau) = I_n. \tag{2.3b}$$

The state transition matrix exists and is uniquely defined by the equation (2.3) when the state space matrices are smooth. It is possible to show that (2.2) is the solution to the system modeled by the state space (2.1) complying with the initial condition $x(t) = x(t_0)$ when $t = t_0$. This is proved by differentiating (2.2) with respect to t as follows,

$$\begin{aligned}
\dot{x}(t) &= \dot{\Phi}(t, t_0)x(t_0) + \int_{t_0}^{t} \dot{\Phi}(t, \tau)B(\tau)u(\tau)d\tau + \Phi(t, t)B(t)u(t) \\
&= A(t)\Phi(t, t_0)x(t_0) + \int_{t_0}^{t} A(t)\Phi(t, \tau)B(\tau)u(\tau)d\tau + B(t)u(t) \\
&= A(t)\left[\Phi(t, t_0)x(t_0) + \int_{t_0}^{t} \Phi(t, \tau)B(\tau)u(\tau)d\tau\right] + B(t)u(t) \\
&= A(t)x(t) + B(t)u(t).
\end{aligned} \tag{2.4}$$

Also, it is trivial to show that the equation (2.2) complies with the system initial condition, hence it is the solution to the dynamical system (2.1). In general, the computation of an analytical solution for the state transition matrix is difficult, it can be achieved based on an iterative method called the Peano-Baker integral series (Rugh, 1996). Even if getting an analytical solution can be a tedious task, it is possible to construct a numerical solution for the state transition matrix based on n linearly independent vectors. Then a general numerical solution is calculated by superposition of the n linearly independent solutions due to the linearity of the system model. By definition, the state transition matrix also complies with the property of composition. When the system is initialized from $x(t_0)$ and no input is

applied, the state variables at times t and t_1 can be obtained as follows,

$$\begin{aligned} x(t) &= \Phi(t, t_1)x(t_1) \\ &= \Phi(t, t_1)\Phi(t_1, t_0)x(t_0) \\ &= \Phi(t, t_0)x(t_0). \end{aligned} \tag{2.5}$$

However, in most cases linear continuous time dynamical systems are time invariant, therefore all the state space matrices are constant matrices and do not depend on the time variable. This type of models are called Linear Time-invariant (LTI) state space models and in this specific case the analytical computation of the state transition matrix is defined by the matrix exponential, such that

$$\Phi(t, t_0) = \mathrm{e}^{A(t-t_0)}\,. \tag{2.6}$$

Consequently, for a given fixed initial state $x(t_0)$ and an input function $u(t)$, an explicit formula exists in order to compute the state variable trajectory as follows,

$$\forall t \in [t_0, +\infty[,\ x(t) = \mathrm{e}^{A(t-t_0)}\,x(t_0) + \int_{t_0}^{t} \mathrm{e}^{A(t-\tau)}\,Bu(\tau)d\tau. \tag{2.7}$$

Continuous state space representations are useful to design analogue control systems, where all the system variables obtained are continuous functions of time. In addition to the control design task, state space models can be used to analyze the behaviour of a system such as the stability. In conclusion, the state space model of a dynamical system allows to predict the future values of the state and output variables, knowing the initial state as well as the input variable function.

Discrete Dynamical Systems

Some physical dynamical systems are behaving in a discrete fashion and therefore a continuous state space model cannot represent them appropriately. In addition, a continuous linear state space model can be discretized to obtain a discrete time linear state space model (Bay, 1998). Discrete state space models are similar to the continuous model representation obtained previously. The equation (2.8) shows a discrete time linear state space model.

$$x(k+1) = A_d(k)x(k) + B_d(k)u(k) \tag{2.8a}$$

$$y(k) = C_d(k)x(k) + D_d(k)u(k), \tag{2.8b}$$

where, $k \in \mathbb{Z}$ represents the time step index, and the other variables and matrices are the same as per the previous continuous state space model. When this model is obtained from a continuous state space model, it is computed by sampling the continuous model with a given sampling time $T_s \in \mathbb{R}_+^*$. Subsequently, the time step index k represents multiples of the sampling time T_s such that,

$$x\left((k+1)T_s\right) = A_d\left(kT_s\right)x\left(kT_s\right) + B_d\left(kT_s\right)u\left(kT_s\right) \tag{2.9a}$$

$$y\left(kT_s\right) = C_d\left(kT_s\right)x\left(kT_s\right) + D_d\left(kT_s\right)u\left(kT_s\right). \tag{2.9b}$$

In the discrete case, all the system variables are discrete sequences of values, constant on each interval $kT_s \leq t < (k+1)T_s$. In particular, the input variable $u(t)$ is constant and equal to $u(k)$ on the interval mentioned previously. Consequently, the input $u(k)$ can be considered as a constant in the integral (2.2) between two consecutive time steps, hence the matrices of the discrete time state space model can be computed using the solution of the continuous time model as follows,

$$x(k+1) = \Phi(k+1,k)x(k) + \int_k^{k+1} \Phi(k+1,\tau)B(\tau)d\tau u(k). \tag{2.10}$$

Therefore, the discrete time state space matrices are linked to the continuous state transition matrix as follows,

$$A_d(k) = \Phi(k+1,k) \tag{2.11a}$$

$$B_d(k) = \int_k^{k+1} \Phi(k+1,\tau)B(\tau)d\tau \tag{2.11b}$$

$$C_d(k) = C(kT_s) \tag{2.11c}$$

$$D_d(k) = D(kT_s) \tag{2.11d}$$

The solutions of the discrete state space equation (2.8) are obtained from a given initial condition $x(0)$ as well as a specific sequence of control inputs by applying recursively the dynamical relation (2.8). The discrete time model yields the following,

$$\begin{aligned} x(1) &= A_d(0)x(0) + B_d(0)u(0) \\ x(2) &= A_d(1)x(1) + B_d(1)u(1) \\ &= A_d(1)A_d(0)x(0) + A_d(1)B_d(0)u(0) + B_d(1)u(1). \end{aligned} \tag{2.12}$$

By induction the state variable $x(k)$ can be obtained based only on the value of

the initial state as well as the sequence of inputs from 0 to $k-1$ such that,

$$\forall k \in \mathbb{N}^*,\ x(k) = \left[\prod_{i=0}^{k-1} A_d(i)\right] x(0) + \sum_{i=1}^{k} \left[\prod_{j=i}^{k-1} A_d(j)\right] B_d(i-1)u(i-1), \tag{2.13}$$

where, the following product is defined by $\prod_{j=k}^{k-1} A_d(j) = I_n$. In a similar way as for the continuous time system, the state transition matrix can be defined based on (2.13) by the following relation,

$$\forall k \in \mathbb{N}^*,\ \Psi(k,l) = \prod_{i=l}^{k-1} A_d(i). \tag{2.14}$$

Finally, the solution of the discrete time state space model (2.8) is analogous to the one obtained in the continuous time case and can be computed from a specific initial condition as well as a sequence of input variables by the following,

$$x(k) = \Psi(k,0)x(0) + \sum_{i=1}^{k} \Psi(k,i)B(i-1)u(i-1). \tag{2.15}$$

If the system is time invariant the state space model simplifies,

$$A_d = \mathrm{e}^{ATs} \tag{2.16a}$$

$$B_d = \int_0^{T_s} \mathrm{e}^{A\tau}\, d\tau B \tag{2.16b}$$

$$C_d = C \tag{2.16c}$$

$$D_d = D. \tag{2.16d}$$

Therefore, the sequence of state variables becomes,

$$\forall k \in \mathbb{N}^*,\ x(k) = A_d^k x(0) + \sum_{i=1}^{k} A_d^{k-i} B_d u(i-1). \tag{2.17}$$

Since the continuous and discrete time state space models are used without confusion in the notations, the subscripts used for the discrete time model matrices are omitted in the remainder of this thesis. Discrete time state space models are used in digital control since they can account for the clock of a digital controller.

2.2.2 Controllability and Observability

Controllability and observability are two important properties of dynamical systems, the verification of these properties is done before being able to perform any design step (Zhou et al., 1996). The principal definitions are described in details within this section.

System Controllability

The definition of a controllable system is given as follows,

Definition 2.1 (Controllability). *The system model defined in equation* (2.1)*, more specifically the pair of state and input matrices* (A, B) *is said to be controllable if, for any initial state* $x(0)$*, time* $t_1 \in \mathbb{R}_+^*$ *and state* x_1*, there exist an input* $u(\cdot)$ *such that the solution of the equation* (2.2) *satisfies* $x(t_1) = x_1$*.*

The controllability property of a linear system model can be verified based on some algebraic criteria.

Theorem 2.1. *The following propositions are equivalent*

1. (A, B) *is controllable*
2. *The controllability Gramian* $W_c(t) = \int_0^t \mathrm{e}^{A\tau} BB^\top \mathrm{e}^{A^\top \tau} d\tau$*, is positive definite*
3. *The controllability matrix* $\mathcal{C} = \left[B \,\middle|\, AB \,\middle|\, A^2B \,\middle|\, \ldots \,\middle|\, A^{n-1}B\right]$ *is full row rank*
4. *The matrix* $[A - \lambda I_n \,|\, B]$ *has full row rank for all* $\lambda \in \mathbb{C}$
5. *The eigenvalues of* $A + BF$ *can be freely assigned (with complex conjugate pairs) by a suitable choice of* F

A dynamical system is said to be uncontrollable if it is not controllable. However, milder conditions exist in the case where the non-controllable modes of a system are stables. This condition is called stabilizability.

Definition 2.2 (Stabilizability). *The system model* (2.1)*, more precisely the pair of state and input matrices* (A, B) *is said to be stabilizable if there exist a linear state feedback* F *such that the system under control is stable i.e.* $A + BF$ *is stable.*

Theorem 2.2. *The following propositions are equivalent,*

1. (A, B) *is stabilizable*
2. *The matrix* $[A - \lambda I_n \,|\, B]$ *has full row rank for all* $\lambda \in \mathbb{C}_+$
3. *There exists a linear state feedback* F *such that* $A + BF$ *is Hurwitz*

The same results are also available for discrete time system defined as per equation (2.8).

System Observability

A system is said to be observable if it complies with the following definition,

Definition 2.3 (Observability). *The system model* (2.1)*, more specifically the pair of output and state matrices* (C, A) *is said to be observable if, for any* $t_1 \in \mathbb{R}_+^*$*, the initial state* $x(0)$ *can be determined from the input and output history of the system i.e.* $u(t)$ *and* $y(t)$ *with* $t \in [0, t_1]$*.*

Theorem 2.3. *The following propositions are equivalent,*

1. (C, A) *is observable*
2. *The observability Gramian* $W_o(t) = \int_0^t \mathrm{e}^{A^\top \tau} C^\top C \, \mathrm{e}^{A\tau} \, d\tau$*, is positive definite*
3. *The observability matrix* $\mathcal{C} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$ *has full column rank*
4. *The matrix* $\begin{bmatrix} A - \lambda I_n \\ C \end{bmatrix}$ *has full column rank for all* $\lambda \in \mathbb{C}$
5. *The eigenvalues of* $A+LC$ *can be freely assigned (with complex conjugate pairs) by a suitable choice of* L

A dynamical system is said to be unobservable if it is not observable. Nonetheless, there exists a milder concept regarding system observability called detectability and defined below.

Definition 2.4 (Detectability). *The system model* (2.1)*, or the pair* (C, A) *is detectable if there exist a matrix* L *such that* $A + LC$ *is stable.*

Theorem 2.4. *The following propositions are all equivalent,*

1. (C, A) *is detectable*
2. *The matrix* $\begin{bmatrix} A - \lambda I_n \\ C \end{bmatrix}$ *has full column rank for all* $\lambda \in \mathbb{C}_+$
3. *There exists a state observer* L *such that* $A + LC$ *is Hurwitz*

It is well know that the notions of controllability and observability for a system are dual concepts. For instance, if the pair $(A^\top, C^\top)$ is controllable, then the pair (C, A) is observable. In a similar way, the two properties of stabilizability and detectability are also dual.

2.2.3 System Stability

Stability of a system defines the behaviour of a dynamical system, most of the time the system state variable either converges towards an equilibrium point or diverges altogether. In some cases however, the trajectory does not converge nor diverge. System stability theory is concerned with all these different cases and its study started with the seminal work of A. M. Lyapunov and the development of the direct and indirect methods (Lyapunov, 1992).

Lyapunov Stability

Definition 2.5 (Lyapunov stability). *The equilibrium point x_e is stable in the sense of Lyapunov at time $t = t_0$ if there exists a $\delta(t_0) > 0$ such that,*

$$\|x(t_0) - x_e\| < \delta \Rightarrow \forall t \in [t_0, +\infty[, \ \|x(t) - x_e\| < \varepsilon. \tag{2.18}$$

The stability of a dynamical system in the sense of Lyapunov is a mild condition. It means that if a dynamical system starts close enough to an equilibrium point at t_0, it will remain relatively close to the equilibrium point. Lyapunov stability does not imply that the state variable of a system will converge to the equilibrium point.

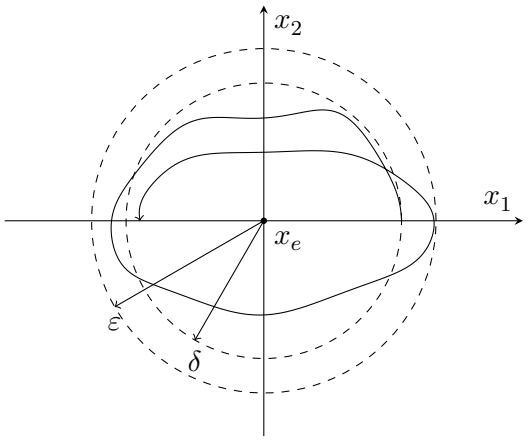

Figure 2.1: Example of Lyapunov stability.

The Figure 2.1 presents the possible phase portrait of a dynamical system stable in the sense of Lyapunov. As it is possible to see on the phase portrait, the constant δ can be smaller than the constant ε. Examples of dynamical systems stable in the sense of Lyapunov are the perfect harmonic oscillators. In the undisturbed case the state variables oscillate around an equilibrium point but never converge towards it. A system is said to be Lyapunov stable if there exist a Lyapunov function associated to it. This Lyapunov function represents the generalized energy of the system.

Therefore, studying the system stability can be done by studying its generalized energy rate of change, and making sure that there exist a non-increasing generalized energy function.

Asymptotic Stability

Definition 2.6 (Asymptotic stability). *The equilibrium point x_e is asymptotically stable if x_e is a stable equilibrium and there exists at time $t = t_0$ a $\delta(t_0)$ such that,*

$$\|x(t_0) - x_e\| < \delta \Rightarrow \lim_{t \to +\infty} \|x(t) - x_e\| = 0. \tag{2.19}$$

The Figure 2.2 presents the phase portrait of an asymptotically stable system. The system does not have to converge monotonically to the equilibrium point and therefore the trajectory of the system state can get closer or farther from the equilibrium point x_e, as long as it eventually converges towards it.

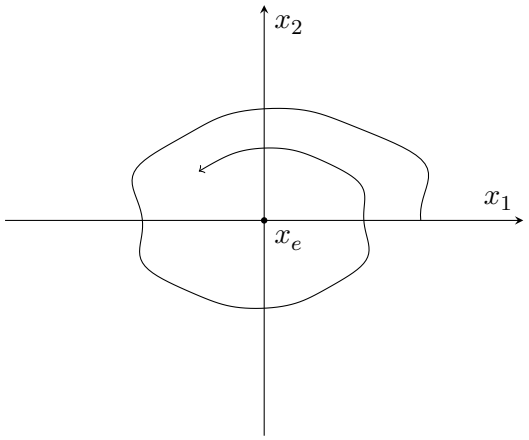

Figure 2.2: Example of asymptotic stability.

Exponential Stability

Definition 2.7 (Exponential stability). *The equilibrium point x_e is exponentially stable if there exists at time $t = t_0$ and a set of constants $(\lambda, a) \in \mathbb{R}_+^* \times \mathbb{R}_+^*$ such that,*

$$\|x(t_0) - x_e\| < \delta(t_0) \Rightarrow \|x(t) - x_e\| \leq \lambda \, \mathrm{e}^{-a(t-t_0)}, \tag{2.20}$$

where the constant a is called the rate of convergence of the system.

As it is presented in the phase portrait Figure 2.3, the state of the system approaches the equilibrium point with a given convergence rate. The exponential stability characteristic of a dynamical system encompasses the asymptotic stability

definition and also provides some information on the rate of convergence to the equilibrium point.

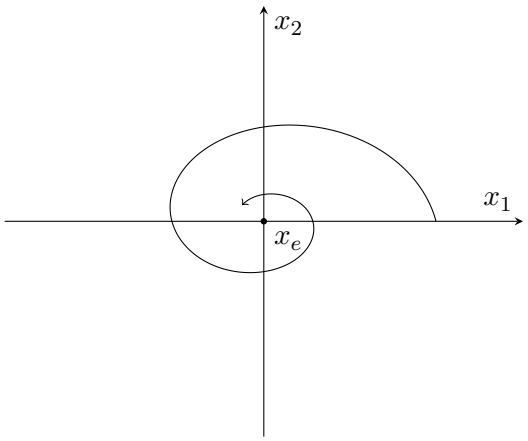

Figure 2.3: Example of exponential stability.

2.2.4 Non-linear Dynamical Systems

Non-linear systems are in general difficult to model, consequently, performing the control design can be a tedious task. The general form for a non-linear dynamical system models is expressed as follows,

$$\dot{x} = f(x, u, t) \tag{2.21a}$$

$$y = g(x, u, t), \tag{2.21b}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are respectively the state and input variables, t represents the time and the functions f and g are vector valued functions with the appropriate dimensions and domains. The difficulties with non-linear systems come from the fact that the superposition principle does not apply and that multiple equilibria may exist, also the system behaviour of a non-linear system is more complex in general. In practice, two main techniques are used in order to handle these difficulties. The first technique called Linear Parameter-varying (LPV) consist of modelling a non-linear system by a set of linear time invariant state space models varying with an exogenous parameter $\theta(t) \in \mathbb{R}^s$. This exogenous parameter can be measured in real time but is not known in advance, an LPV model is represented as per equation (2.22).

$$\dot{x}(t) = A(\theta(t))x(t) + B(\theta(t))u(t) \tag{2.22a}$$
$$y(t) = C(\theta(t))x(t) + D(\theta(t))u(t), \tag{2.22b}$$

In certain cases, it is possible to decompose a non-linear system model into multiple linear system models. This requires the system to be relatively smooth with regards to the time derivatives (Rugh and Shamma, 2000; Leith and Leithead, 2000). The linearization of the system model is performed as per equation (2.23), where the couple $(\tilde{x}_i, \tilde{u}_i)$ represents an equilibrium manifold composed of the points indexed by i. Since the Jacobian linearization of the system is computed along a manifold of equilibria the linearized models are time invariant.

$$\delta\dot{x} = \left.\frac{\partial f(x,u,t)}{\partial x}\right|_{(\tilde{x}_i,\tilde{u}_i)} \delta x + \left.\frac{\partial f(x,u,t)}{\partial u}\right|_{(\tilde{x}_i,\tilde{u}_i)} \delta u \tag{2.23a}$$
$$\delta y = \left.\frac{\partial g(x,u,t)}{\partial x}\right|_{(\tilde{x}_i,\tilde{u}_i)} \delta x + \left.\frac{\partial g(x,u,t)}{\partial u}\right|_{(\tilde{x}_i,\tilde{u}_i)} \delta u \tag{2.23b}$$

When the equilibrium points are parametrized by the exogenous variable $\theta(t)$, then the linearized model (2.23) yields the LPV model provided in equation (2.24) parametrized with the deviation system variables. If the parametrization is based on an endogenous system variable then the model is called quasi-LPV.

$$\left[\begin{array}{c|c} A(\theta(t)) & B(\theta(t)) \\ \hline C(\theta(t)) & D(\theta(t)) \end{array}\right] = \sum_{i=1}^{s} \theta_i(t) \left[\begin{array}{c|c} A_i & B_i \\ \hline C_i & D_i \end{array}\right] \tag{2.24}$$

Linearizing a non-linear system model is only valid at the vicinity of the equilibrium points chosen. Subsequently, the linear system models obtained are blended together based on the value of the scheduling parameter as per equation (2.24). Then, the control laws can be designed for each linear model and blended in the same way, or the control laws can be switched as a function of the scheduling parameter $\theta(t)$. Similarly, the non-linear system approximation (2.24) can be achieved in the discrete time case, using on the same techniques.

2.3 Optimal Control

The maturation of state space system modelling along with the notions of controllability and observability presented previously led to the development of the Linear Quadratic Regulator (LQR) theory (Kalman, 1960). This technique belongs to the

field of optimal control, more generally it consists of minimizing a cost function representing the performance of the control system, it relies on the system model as well as a cost function that is usually a linear combination quadratic penalties on the state and input variables.

2.3.1 Linear Quadratic Regulator

The linear quadratic regulator technique minimizes the discrepancies on the desired values of the state and input variables for the continuous time invariant linear state space model (2.1) by minimizing an infinite horizon cost function as follows,

$$J_\infty = \int_0^{+\infty} x(t)^\top Q x(t) + u(t)^\top R u(t) dt \tag{2.25}$$

where $(Q, R) \in \mathbb{S}_+^n \times \mathbb{S}_{++}^m$ represent weighting matrices to penalize the state and input variables respectively. A variant of the linear quadratic regulator is formulated with a finite horizon cost function. In the infinite horizon case, minimizing the system performance index yields a linear state feedback control law under the assumptions that the system is stabilizable and that the pair $(Q^{\frac{1}{2}}, A)$ is observable. The linear control law is noted as follows,

$$\forall t \in \mathbb{R}_+,\, u(t) = Fx(t). \tag{2.26}$$

The minimization of the control cost J_∞ leads to the matrix algebraic Riccati equation (2.27) which then yields the control law $F \in \mathbb{R}^{m \times n}$ formulated as per equation (2.28).

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0 \tag{2.27}$$

The algebraic Riccati equation is a matrix equation in the variable $P \in \mathbb{S}^n$, this equation can be solved using the minimum principle of Pontryagin or the second Lyapunov method. The algebraic Riccati equation can have multiple solutions, however only the unique stabilizing solution $P \in \mathbb{S}_{++}^n$ is of interest in order to compute a stabilizing state feedback controller.

$$\begin{gathered} \forall t \in \mathbb{R}_+,\, u(t) = -R^{-1}B^\top P x(t) = Fx(t) \\ F = -R^{-1}B^\top P \end{gathered} \tag{2.28}$$

It can be noticed that solving the Riccati equation is equivalent to solving the Lyapunov equation (2.29) with the system dynamics in closed loop $A_{cl} = A + BF$

and the weighting matrix $Q_{cl} = Q + F^\top RF$.

$$A_{cl}^\top P + PA_{cl} = -Q_{cl} \tag{2.29}$$

Consequently, the control cost J_∞ with the initial condition $x(t_0) = x_0$ is evaluated as follows,

$$J_\infty = \int_0^{+\infty} x(t)^\top Qx(t) + u(t)^\top Ru(t)dt = x_0^\top Px_0. \tag{2.30}$$

Similar results hold in the linear discrete time case, the integral operator used to expressed the infinite horizon control cost (2.30) is replaced by a discrete time summation such that,

$$J_\infty = \sum_{k=0}^{+\infty} x(k)^\top Qx(k) + u(k)^\top Ru(k). \tag{2.31}$$

In the same fashion, the Riccati and Lyapunov equations are replaced by their discrete time counterparts as per equations (2.33) and (2.34) respectively. In the discrete time case, the LQR state feedback control law is defined based on the Riccati solution P as follows,

$$F = -\left(B^\top PB + R\right)^{-1} B^\top PA^\top. \tag{2.32}$$

Where the discrete algebraic Riccati equation is defined such that,

$$A^\top PA - P - \left(A^\top PB\right)\left(R + B^\top PB\right)^{-1}\left(B^\top PA\right) + Q = 0. \tag{2.33}$$

In the same way, the unique stable solution of the discrete Riccati equation (2.33) is a solution of the discrete algebraic Lyapunov equation (2.34), with $A_{cl} = A + BF$ and $Q_{cl} = Q + F^\top RF$.

$$A_{cl}^\top PA_{cl} - P = -Q_{cl} \tag{2.34}$$

Linear quadratic control relies on the solution of the matrix algebraic Riccati equation in the continuous and discrete time cases. Optimal LQR control optimizes a performance index offline based on the linear system state space model. In both the continuous and discrete cases, the control law $F \in \mathbb{R}^{m\times n}$ computes the value of an input variable based on a linear combination of the state variables. Different variants of the LQR optimal control have been developed to synthesize linear control laws robust to disturbances , such as the $\mathcal{H}_\infty$ and $\mathcal{H}_2$ control techniques (Kwakernaak, 1993; Gahinet and Apkarian, 1994; Zhou et al., 1996) or to synthesize control gains

robust to model uncertainty (Kothare et al., 1996). However, enforcing physical system constraints on the state and input variables is not practical with offline optimal control strategies and this feature is more suitably achieved with online optimal control methods.

2.3.2 Model-based Predictive Control

Similarly to the LQR control technique, Model Predictive Control (MPC) relies on solving an optimization problem maximizing the system performance in order to compute the future control inputs. In the case of model predictive control, the optimization is performed online based on the system dynamical model as well as the value of the current state variable. The dynamical system model is used to forecast the system behaviour over a prediction horizon that recedes towards the future at each time steps. Consequently, it is easy to realize that the system model plays a central role in MPC. This control technique has started originally in process control (Richalet et al., 1978; Richalet, 1993a,b; Qin and Badgwell, 2003). Since then, the theoretical aspects of model predictive control concerning stability and feasibility have been well understood and are gathered within some seminal textbooks (Maciejowski, 2002; Rossiter, 2003; Camacho and Bordons, 2004; Rawlings and Mayne, 2009). Note that model predictive control does not refer to a specific control strategy, it gathers a wide range of control techniques relying on a system model to compute optimal control inputs. Implementing a standard version of model predictive control for the a discrete time LTI system (2.9) can be achieved by solving the online optimization problem (2.35).

$$
\begin{aligned}
\underset{u_k}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \left\{ \|x(k)\|_Q^2 + \|u(k)\|_R^2 \right\} + \|x(N)\|_P^2 \\
\text{subject to} \quad & \forall k \in [\![0, N-1]\!] \\
& x(k+1) = Ax(k) + Bu(k) \\
& x(0) = x_0 \\
& (x(k+1), u(k)) \in \mathbb{X} \times \mathbb{U} \\
& (x(k+1), u(k)) \in \mathbb{W} \\
& x(N) \in \mathbb{X}_f \subseteq \mathbb{X},
\end{aligned}
\tag{2.35}
$$

where $(Q, R) \in \mathbb{S}_+^n \times \mathbb{S}_{++}^m$ respectively represent the state and input weighting matrices, $P \in \mathbb{S}_+^n$, $\mathbb{X}$, $\mathbb{U}$, $\mathbb{W}$ and $\mathbb{X}_f$ are sets of linear constraints respectively on the state variables, the input variables, a mix of the state and input variables and the final state variable. The parameter $N \in \mathbb{N}^*$ is called the prediction horizon. The

optimization problem (2.22) consists in minimizing a quadratic objective function subject to a set of linear constraints, therefore, it is a Quadratic Programming (QP) problem. The cost function is expressed as the classical quadratic system performance metric in the same way as for the LQR control technique. The linear constraints represent the system dynamics as well as limits on the state and the input variables. The state and input limits could represent physical limits enforced by actuator saturation or desirable state limits to keep the system undamaged. The state variable at the end of the prediction is subject to a different weighting matrix in order to account for the remaining control cost and emulate the infinite horizon cost. Solving the optimization problem (2.35) yields a sequence of optimal control inputs as well as a predicted state trajectory as follows,

$$u^\star = [u(0), u(1), \ldots, u(N-1)] \tag{2.36a}$$

$$x^\star = [x(1), x(2), \ldots, x(N)]\,. \tag{2.36b}$$

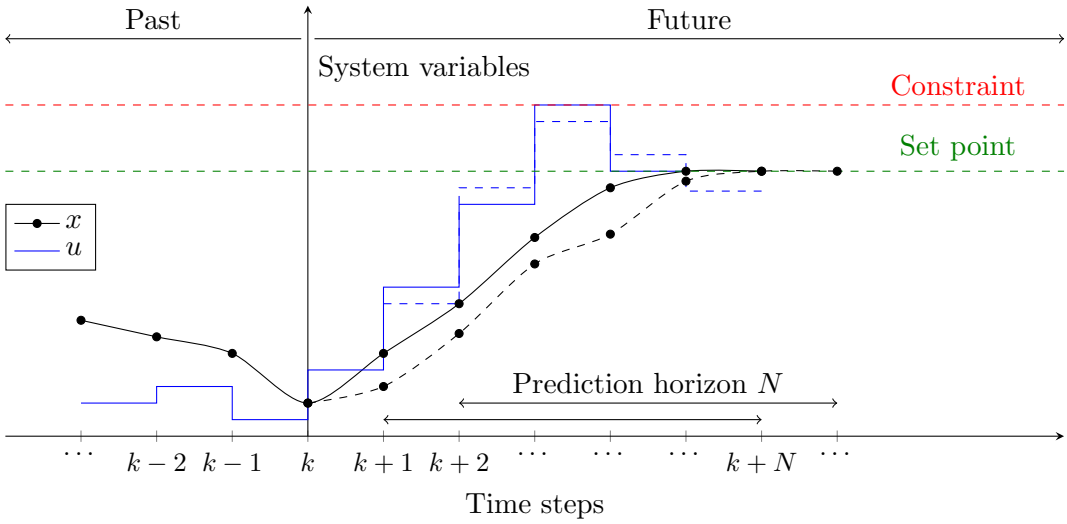

Figure 2.4: Model predictive control example.

The main idea behind MPC is illustrated in Figure 2.4, the black line represents the state variable that is steered to the green dashed line representing the set point. The blue piecewise constant plots represent optimal sequences of input variables. Finally, the red dashed line represents a physical system constraint on the input variable. At time step k the optimization problem (2.35) is solved based on the measurement of the current state variable $x(k)$ over the prediction horizon from $k+1$ to $k+N$. Then, the first input variable $u(k)$ from the optimal sequence of inputs (2.36) is applied to the system and the new state variable $x(k+1)$ is measured.

Therefore, it allows to run a new optimization over a new prediction horizon shifted towards the future, from $k+2$ to $k+1+N$. Then k is incremented and this process is repeated in a receding fashion.

Features of Model Predictive Control

Model predictive control has got many interesting features that made this control technique attractive in the past for process control and that still motivates its use today for more complex and faster control applications (Mayne, 2014).

Optimal Control MPC similarly to LQR emerges from the solution of an optimization problem, the main distinction however is that model predictive control solves the optimization problem online and recursively contrary to LQR where the solution is computed offline (Kalman, 1960). Unlike LQR the control inputs are sent in an open loop fashion, and feedback is used to account for unmeasured and unmodeled disturbances. Also, the minimized cost function can be tailored to the system by considering the use of different norms for the state, input or output variables discrepancies or a penalty on the input rate of change. Nonetheless, since MPC requires to solve an optimization problem online, enough time has to be allocated (Bartlett et al., 2002; Pannocchia et al., 2007). Therefore, implementing MPC for large system with fast dynamics is usually difficult.

Model-based Technique Different variants of model predictive control use different types of models but they all have in common the fact that the system model is very important. Indeed, an accurate dynamical model allow for accurate predictions and subsequently increases the performance of the control system. Model predictive control has been developed for linear and non-linear system as well as uncertain models (Morari and Lee, 1999). Different control system architectures can be taken into account by using distributed and decentralized system models. The use of large and complex models will slow the optimization because of the implementation of more decision variables and constraints.

Constraint Handling Constraints are inherent to physical systems and unlike linear quadratic regulator, model predictive control is well suited to system with constraints. Classical control techniques such as Proportional Derivative Integral (PID) controllers enforce the constraints after the control input is computed with saturations and anti-wind up techniques. In the case of MPC the system constraints are directly expressed within the optimization problem formulation (Mayne et al., 2000). In general, controlling a system optimally goes with operating close to the

system physical constraints (Maciejowski, 2002). Some constraints are hard and cannot be exceeded, some other constraints can be formulated as soft by adding a cost within the objective function when they are violated.

Multivariable Model Model predictive control techniques can be applied to multivariable system as there is no limitations on the size of the system model used within the online optimization problem. Using a multivariable model within the MPC framework is useful to take into account the different dynamical couplings within the state and input variables. Nonetheless, an increase in system dimensions will lead to a slower online optimization. Therefore, large-scale systems are usually controlled in a decentralized or distributed fashion in order to reduce the computational burden. Different cooperative and non-cooperative MPC schemes have been developed (Trodden and Richards, 2013; Negenborn and Maestre, 2014).

Robust control Controlling a system with MPC relies on a nominal system model that does not necessarily include a disturbance model. It can be noted that model predictive control has some inherent robustness to disturbances as well as model mismatch (Pannocchia et al., 2011). However, robust theoretical framework has been developed to tackle modeled disturbances. For instance, min-max MPC as well as tube MPC are two of the main techniques that take into account disturbances and provide some robustness (Scokaert and Mayne, 1998; Langson et al., 2004). Another aspect of MPC considered model uncertainty (Kothare et al., 1996).

2.4 Control System Architectures

The architecture of control systems has been studied for decades (Mesarović et al., 1970; Šiljak, 1991; Lunze, 1992), in the past the main motivation has been the automation of processes found in large scale industrial systems. Examples such as the steel industry, the chemical industry, power systems and traffic networks have benefited from the development of more decentralized control system architectures (Al-gherwi et al., 2010). These complex systems are most of the time composed of smaller interacting subsystems and therefore the design of a centralized controller can become tedious or even impossible due to the physical size of the system or because of its geographical spread (Scattolini, 2009). The main control architectures developed to answer these challenges includes fully decentralized structures where no information is shared between the local controllers, distributed control structures where only some information is shared between the local controllers. Finally, the hierarchical control structures are similar to the distributed control architectures in

many aspects. However, in the case of hierarchical control structures the controller is composed of different interconnected levels or layers often working at different time scales. This section presents the different types of control architectures along with their benefits and drawbacks. The next subsection present the centralized control system architecture.

2.4.1 Centralized Control Architecture

The first control system architecture presented is the fully centralized architecture, in this case all the information available from the system is provided to the controller. The controller then produces the control input used for the entire system. Therefore, this control structure requires a fully connected network between the system sensors and the centralized controller as well as between the centralized controller and the system actuators. The main benefit of centralized control system architectures is that the controller takes control decisions to improve the system wide performance and therefore it provides the best system performance. Also, a centralized controller can take into account the dynamical interactions between the subsystems when computing the control input.

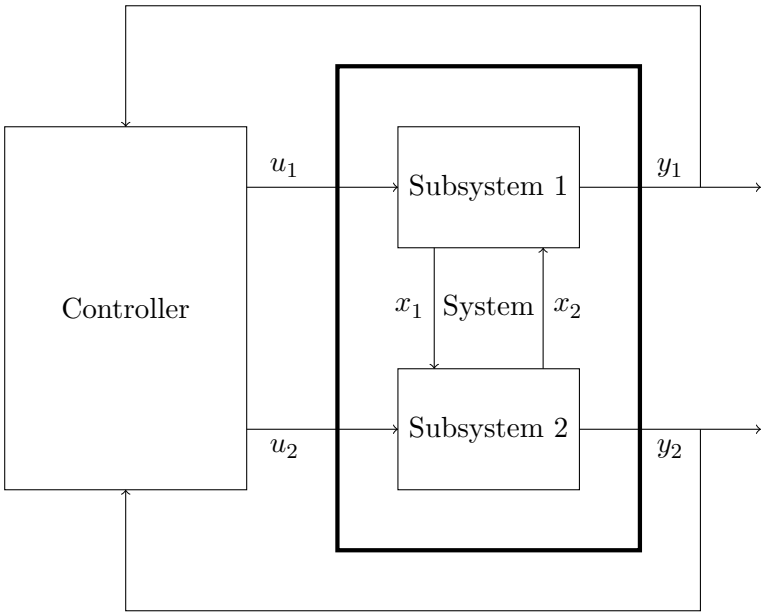

Figure 2.5: Centralized control system architecture.

The centralized control system structure is presented on Figure 2.5, in this example the system is composed of two subsystems. All the output information available from the system is provided to the system controller in order to compute the input variables of both subsystems simultaneously. Even if the centralized control

architecture provides the best system wide performance, it is not always possible to apply a centralized controller. The main barriers to the implementation of centralized control are the geographical separations between the subsystems as well as the communication links limited bandwidth, cost and reliability. Subsequently, for the systems having one or many of these previous characteristics the assumption of centralizing the system information does not hold. In order to answer the issues triggered by centralized structures other decentralized architectures have been developed. The next subsection presents the decentralized control system architecture.

2.4.2 Decentralized Control Architecture

Decentralized control architectures do not rely on the centralization of the subsystems information, in this case each subsystem is equipped with a local controller that receives only the information of a single subsystem in order to take a control decision for it. The local controllers are not exchanging any information. Consequently, the need for communication links between the system and the control layer is decreased, however, because each local controller is only in charge of a given subsystem without any knowledge of what the entire system is doing, the system wide performance can be poor (Schuler et al., 2014). More specifically, if the subsystems are strongly dynamically coupled, the design of a stabilizing decentralized controller can become a complex task and in some cases can even be infeasible.

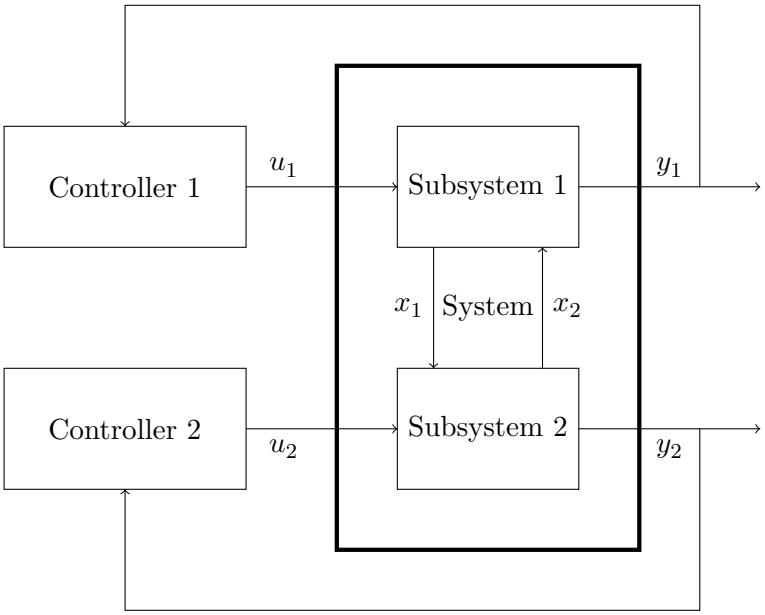

Figure 2.6: Decentralized control system architecture.

Figure 2.6 presents the decentralized control system architecture with two sub-

systems, in this architecture one local controller is in charge of one subsystem only. Therefore, the entire system has to be partitioned into disjoint sets containing some of the system input and output variables. The local controllers are then designed based on the dynamics of the subsystem they are connected to (Alessio et al., 2011). As it is presented Figure 2.6, subsystem to subsystem interactions can exist, they can be due to the internal states of the two subsystems or can be more direct and caused by the input variables. Subsequently, when the subsystem to subsystem interactions are weak the design of the decentralized architecture is straight forward. On the contrary, when the subsystems are strongly coupled it is well known that in some cases only poor performance can be achieved and that system stability can be an issue (Scattolini, 2009). Intuitively, the local controller within the decentralized architecture will only take into account the control of their own subsystem. Consequently, when the subsystem are strongly coupled there will be a fight for control that can be the root cause of instability. In this type of control system architecture, the interactions are treated as disturbances that have to be rejected. The next subsection presents the distributed architecture, this type of architecture is similar to the decentralized architecture, however some communication is allowed between the local subsystems in order to take into account for the couplings.

2.4.3 Distributed Control Architecture

In distributed control system architectures, a communication network is established between the local controllers so that each local controller has some information on the other local controllers and subsystems as shown Figure 2.7. The information shared mainly consists of input as well as output variables and can include the predicted state and input variables (Christofides et al., 2013). The distinctions between the different distributed architectures are with regards to the communication network topology, the information exchange rate and the type of control algorithm used by the local controllers.

Different communication network topologies can be implemented, for example Figure 2.7 presents a fully connected communication structure. However, if the information was only sent from the first controller to the second controller then the communication network topology would only be partially connected. A partially connected network topology can be adapted to the case of large-scale systems where the subsystem to subsystem interactions only comes from the direct neighbours. Regarding the information exchange rate different communication protocols can be implemented. The information exchange between the local controllers can be performed once every sampling time step or can occur multiple times per sampling time step, it corresponds respectively to non-iterative and iterative control tech-

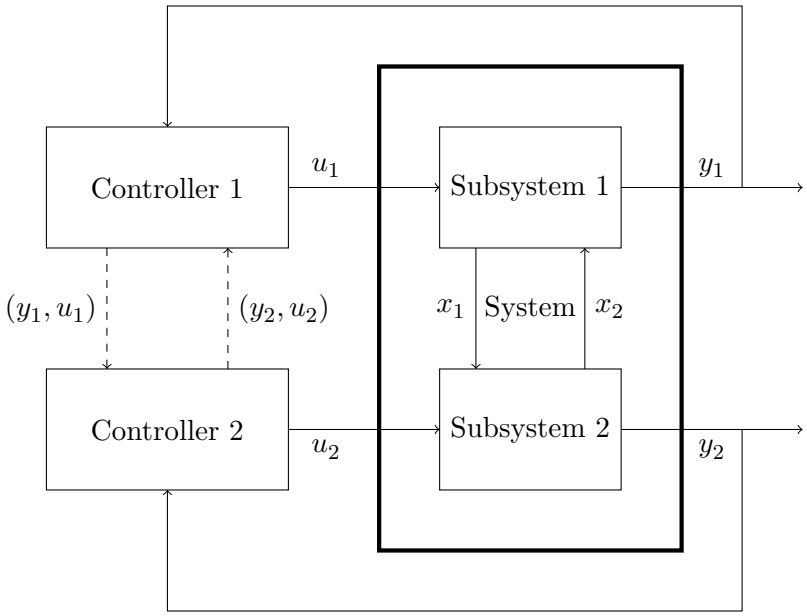

Figure 2.7: Distributed control system architecture.

niques (Maestre et al., 2011). Finally, the type of distributed control algorithms implemented can focus on a subsystem performance index or on a system wide performance metric. These two techniques are respectively called independent or non-cooperative and cooperative control algorithms. The local controllers can perform the optimization sequentially or in parallel (Richards and How, 2007; Trodden, 2014). Distributed MPC is still an active field of research (Maestre and Negenborn, 2013).

2.4.4 Hierarchical Control Architecture

The hierarchical control system architecture is a compromise between the centralized and distributed control structures (Mesarović et al., 1970; Scattolini, 2009). This type of control architecture includes multiple control layers usually working at different time scales, it also comprises local controllers in charge of sending the control input to their own local subsystems. Very often two layers of control are included, the first layer is the supervisory layer located on top, it is often used to coordinate the actions of the local controllers. For instance, it can be used to readjust optimization weights in an optimal control framework, or to modify the set points in a more classical framework. As it is presented Figure 2.8, a supervisory agent is connected to the local controllers and can modulate their behaviour, while the local controllers are in charge of feeding control inputs to the subsystem they are connected to. Different types of hierarchical control methods have been developed

in the past and are presented within this subsection.

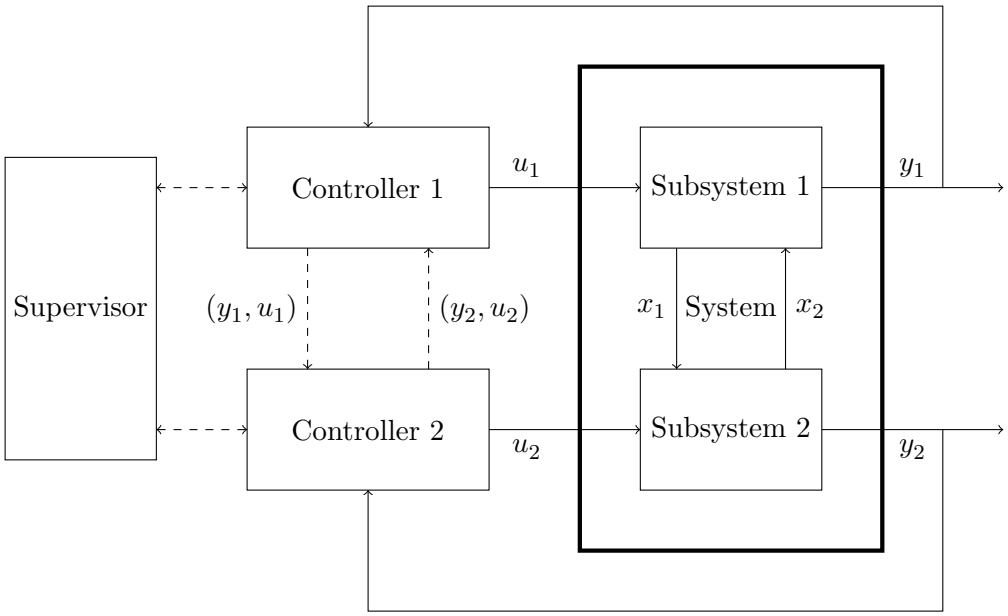

Figure 2.8: Hierarchical control system architecture.

Hierarchical Control for Coordination

This first kind of control architecture is used to coordinates the actions of local controllers. The local controllers can exchange the value of the state variables of the subsystem they are connected to as well as their predicted state trajectory. Then an iterative procedure can be implemented between the supervisor and the local controllers in order to reach a system consensus. Throughout this technique such a control architecture can take into account the dynamical couplings between the different subsystems. The supervisory agent and the local controllers can be working at different time scales, however, working at the same time scales allow the use of iterative procedures.

Hierarchical Control of Multilayer Systems

In this kind of multilayer systems, the local controllers are working at different time scales. Such an architecture is suitable in two main cases, first of all, it is adapted to systems having subsystems with different dynamic behaviours. Indeed a few medium and large scale systems can be partitioned into slow and fast dynamical subsystems. The second case is when the overall system optimization and the local controllers are working with different time scales. The top layer compute optimal values which are the set points used by the lower control layers, these values are sent through a

top-down communication links. Bottom-up communication is used in order for the local controllers to share possible disturbances and to provide the required feedback to the top layer.

2.5 Summary

Control system theory and systems engineering are interdisciplinary fields and therefore, they can be applied to a broad set of physical systems, including gas turbine systems. The design of a controller requires not only the selection of a control system architecture but also the design and the implementation of a control technique. This literature review chapter first presented the field of systems engineering that is key to the understanding of the behaviours of dynamical systems as well as the analysis of some fundamental properties, such as stability, controllability and observability. The main aspect developed within this chapter concerned the modelling of dynamical systems. Dynamical system models allow to predict the future state and output variables of a system and thus they provide a solid foundation to the development of optimal and robust control techniques. The analysis of physical systems based on dynamical models is fundamental to the understanding of the interactions between the subsystems composing a system and is subsequently central to the choice of control system architecture. A presentation of optimal control and model-based predictive control has been provided along with the different control system architectures developed in the past. Consequently, redesigning the gas turbine engine control systems will be achieved by selecting an architecture, optimally with regards to an objective function. Following this, a control method will be developed based on the architecture selection performed previously.

Chapter 3

Background

3.1 Introduction

The background developed in this chapter will then be used within the subsequent chapters of this thesis in order to solve system and control optimization problems. Convex analysis is one of the main branches of mathematics, it studies convex sets, convex functions and provides the theoretical foundation necessary in order to perform convex optimization. Convex optimization is a subset of the more general field studying all the different types of mathematical optimization problems. Convex analysis is at the crossroad of linear algebra as well as non-linear mathematical analysis. The mathematical concepts and objects described within this chapter all comply with some specific prerequisites that confer them remarkable properties. The properties of convex sets and convex functions are then used in order to ascertain some characteristics on convex optimization programming problems. One of the main and central idea of convex optimization relies on the fact that local information can be used to assert global characteristics on the optimization problem. Therefore, convex optimization problems can be solved numerically very efficiently and reliably even when no analytical solution exists. Interior-point methods have been developed in order to solve convex optimization problems in polynomial time (Nesterov and Nemirovskii, 1994). In the past few decades, convex optimization has proved to be a very powerful tool in multiple fields such as automatic control, systems engineering, information theory, signal processing and finance. For instance, linear programming as well as least-square optimization problems are special cases of convex optimization programming problems.

This chapter will present the background in convex optimization by introducing definitions, properties as well as examples on respectively convex sets, functions and optimization programming problems. Finally, some common non-convex problems

are discussed along with the standard algorithm and methods to solve them. Convex analysis has been covered widely in the literature (Rockafellar, 1970; Hiriart-Urruty and Lemaréchal, 2001; Boyd and Vandenberghe, 2010), the background presented here is not exhaustive and the reader is invited to refer to the textbooks cited within this thesis if more information is needed.

3.2 Convex Sets

3.2.1 Introduction

Set theory defines mathematical sets of objects. Amongst all the mathematical sets, the convex sets have specific properties useful to perform convex optimization. This section will give the definition of a convex set, present some examples of convex sets and introduce some of their basic properties.

3.2.2 Definition of a Convex Set

A convex set is defined as follows,

Definition 3.1 (Convex set). *A set $\mathcal{S}$ is convex if, for all elements x and y belonging to $\mathcal{S}$, the line segment joining x and y also lies in the set $\mathcal{S}$,*

$$\forall \theta \in [0,1],\ \forall (x,y) \in \mathcal{S}^2,\ \theta x + (1-\theta) y \in \mathcal{S}. \tag{3.1}$$

Convex sets are essential in order to define convex functions and therefore to define convex optimization programming problems. The Figure 3.1 presents different types of convex sets that can be represented in a two-dimensional plane.

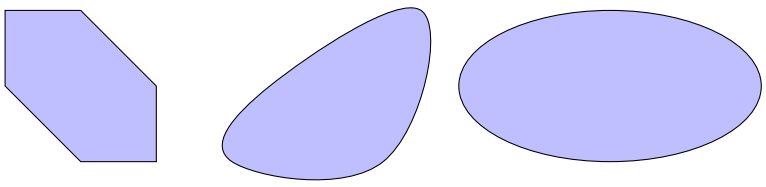

Figure 3.1: Example of two-dimensional convex sets.

Definition 3.2 (Convex combination). *For a finite number $n \in \mathbb{N}^*$ of points $\{x_1, \ldots, x_n\}$ belonging to a set $\mathcal{S}$, a convex combination is defined by the following*

relation,

$$\begin{aligned} \forall i \in [\![1, n]\!], \theta_i \geq 0, \sum_{i=1}^{n} \theta_i = 1, \\ x = \sum_{i=1}^{n} \theta_i x_i. \end{aligned} \tag{3.2}$$

The vector x belongs to the set $\mathcal{S}$ for any convex combinations of points $\{x_1, \ldots, x_n\}$ if and only if the set $\mathcal{S}$ is convex.

Based on the definition of a convex set and of a convex combination, it is possible to show via induction that a convex set contains all the possible convex combinations of its elements. The most natural convex sets are the polyhedra and the polytopes, which can be defined by a finite set of affine inequalities. A polytope is a defined as a bounded polyhedron.

Definition 3.3 (Affine set). *A set $\mathcal{S}$ is affine if, for all elements x and y belonging to $\mathcal{S}$, the line joining x and y also lies in the set $\mathcal{S}$,*

$$\forall \theta \in \mathbb{R}, \forall (x, y) \in \mathcal{S}^2, \theta x + (1 - \theta) y \in \mathcal{S}. \tag{3.3}$$

Convex sets subsume affine sets, indeed if a set is affine it will contain all the lines joining any couples of its points. Subsequently, it will also contain all the line segments joining any pairs of points, therefore it will also be convex. However, the converse is not true.

3.2.3 Examples of Convex Sets

This subsection presents different examples of convex sets and their definitions along with some proofs of convexity.

Definition 3.4 (Unit simplex). *The unit simplex set in dimension n is a special case of a polytope and is defined by,*

$$\theta = [\theta_1, \ldots, \theta_n]^\top \in \Lambda_n, \tag{3.4a}$$

$$\Lambda_n = \left\{ \theta \in \mathbb{R}^n \,\middle|\, \sum_{i=1}^{n} \theta_i = 1, \forall i \in [\![1, n]\!], \theta_i \geq 0 \right\}. \tag{3.4b}$$

The Figure 3.2 represents the unit simplex in dimension three, it is possible to see the simplex as a set of linear inequalities. Therefore, all simplexes are convex regardless of their dimension. The more general simplex set in dimension n can also be seen as the hyperplane defined by $\theta_1 + \ldots + \theta_n = 1$, restricted to the closed

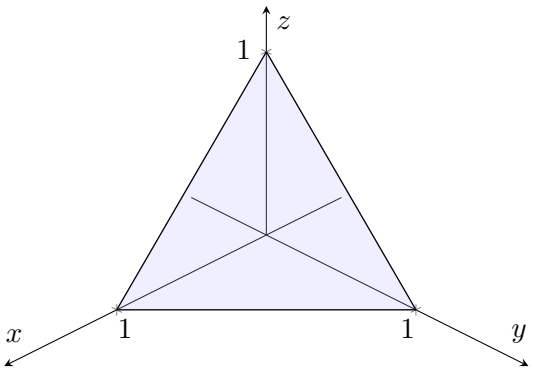

Figure 3.2: Representation of the simplex in dimension three.

positive orthant in $\mathbb{R}^n$. The general case of a set defined by linear inequalities and equalities is the polyhedron. The definition of a polyhedron is given as follows,

Definition 3.5 (Polyhedron). *A set $\mathcal{P}$ is a polyhedron when there exists a set of affine inequalities represented by,*

$$\mathcal{P} = \{x \in \mathbb{R}^n \,|\, Gx \leq h\}\,. \tag{3.5}$$

A polyhedron can be understood as the intersection of multiple half-spaces. Therefore, because each half-space is a convex set and because a polyhedron is a finite intersection of convex sets, a polyhedron is a convex set. Linear equalities define affine sets and are therefore a representation of convex sets.

Example 3.1 (Affine set). *The set defined by $\mathcal{S} = \{x \in \mathbb{R}^n \,|\, Ax = b\}$, where $A \in \mathbb{R}^{p\times n}$ and $b \in \mathbb{R}^p$, is such that,*

$$\forall (x, y) \in \mathcal{S}^2,\, Ax = b,\, Ay = b \tag{3.6a}$$
$$\forall (x, y) \in \mathcal{S}^2,\, \forall \theta \in \mathbb{R},\, \theta Ax = \theta b,\, (1-\theta)Ay = (1-\theta)b \tag{3.6b}$$
$$\forall (x, y) \in \mathcal{S}^2,\, \forall \theta \in \mathbb{R},\, \theta Ax + (1-\theta)Ay = \theta b + (1-\theta)b \tag{3.6c}$$
$$\forall (x, y) \in \mathcal{S}^2,\, \forall \theta \in \mathbb{R},\, A(\theta x + (1-\theta)y) = b. \tag{3.6d}$$

The set presented in the Example 3.1 does not include any restriction on the value of θ, therefore it is affine and is convex by definition. A more general definition of a polyhedron could include linear equalities. Although, linear equalities can always be reformulated as a couple of linear inequalities. Another particular example of a convex set is the ellipsoid, as defined in Definition 3.6.

Definition 3.6 (Ellispoid). *An ellipsoid $\mathcal{E}$ is defined as follows,*

$$\mathcal{E} = \left\{ x \in \mathbb{R}^n \,\middle|\, (x - x_c)^\top P^{-1}(x - x_c) \leq 1,\, P \in \mathbb{S}^n_{++} \right\}, \tag{3.7}$$

where the vector $x_c \in \mathbb{R}^n$ defines the center of the ellipsoid. The lengths of the semi-axes are given by the square roots of the eigenvalues of the matrix P.

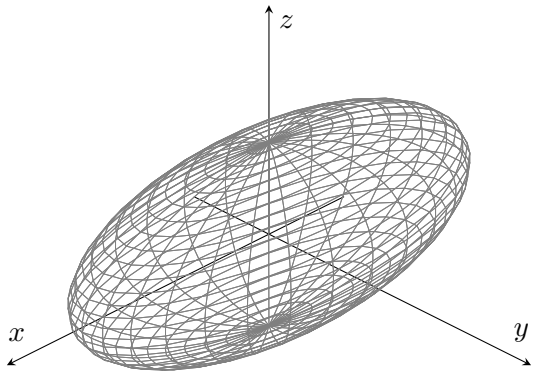

Figure 3.3: Representation of an ellipsoid in dimension three.

The Figure 3.3 represents an ellipsoid in three dimensions. Any section generated by the intersection of the ellipsoid with a plane is an ellipse. A convex set similar to the ellipsoid is the norm ball, which is defined as per Definition 3.7.

Definition 3.7 (Norm ball). *A norm ball is defined in $\mathbb{R}^n$ for a given norm such that,*

$$\mathcal{B} = \{x \in \mathbb{R}^n \,|\, \|x - x_c\| \leq r\}\,, \tag{3.8}$$

where $x_c \in \mathbb{R}^n$ is the center of the ball and $r \in \mathbb{R}_+$ is the radius of the ball. The open norm ball is defined by a strict inequality.

The last topological sets of importance are the conic sets, they are defined as follows,

Definition 3.8 (Cone). *A set K is said to be a cone if, for all elements x belonging to K, the following relation holds,*

$$\forall(\theta, x) \in \mathbb{R}_+ \times K,\, \theta x \in K. \tag{3.9}$$

The Figure 3.4 represents the full cone defined by the Euclidean norm in three dimensions. The intersection of this cone with a plane orthogonal to the z-axis gives circles. Restricting this cone to the half-plane z positive yields the second-order cone. A cone is not convex in general, however a cone is called a convex cone when it complies with the definition of a cone and of a convex set. The two previous definitions can be combined into the following definition,

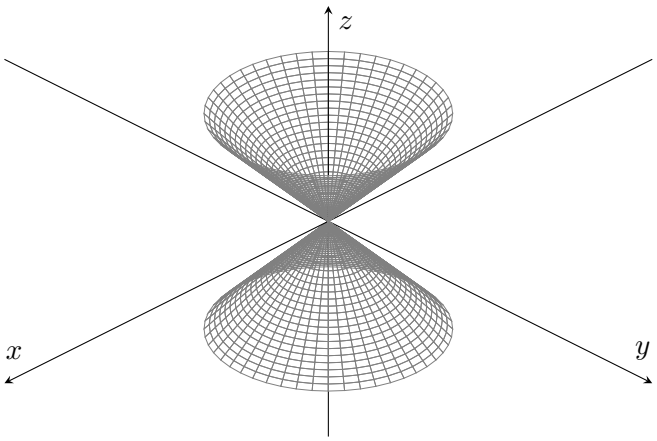

Figure 3.4: Representation of a cone in dimension three.

Definition 3.9 (Convex cone). *A set K is said to be a convex cone if, for all elements x and y belonging to K, the following relation holds,*

$$\forall(\theta_1, \theta_2) \in (\mathbb{R}_+)^2, \forall(x, y) \in K^2, \theta_1 x + \theta_2 y \in K. \tag{3.10}$$

A conic set is called a proper cone if it is a closed, convex, pointed and solid cone. A cone is pointed if it does not contain a full line and solid if its interior is different from the empty set. The proper cone used for most applications in control theory is the cone of positive semi-definite matrices.

Example 3.2 (Positive semi-definite cone). *The set of symmetric positive semi-definite matrices of dimension $n \in \mathbb{N}^*$ is a proper cone, also called the semi-definite cone, and it is denoted as follows,*

$$\mathbb{S}_+^n = \left\{ X \in \mathbb{R}^{n \times n} \,\middle|\, X \succeq 0 \right\}. \tag{3.11}$$

The set of symmetric positive definite matrices is the convex cone that is the interior of the positive semi-definite cone.

3.2.4 Convex Hulls

When a set S is not convex, it is possible to compute a convex set that contains S, it is called the convex hull of S. The convex hull of a set S, denoted $\operatorname{co} S$, is the set containing all convex combinations of the elements in S. Consequently, it is the smallest convex set containing S. In the case where S is already a convex set, the convex hull of S and S are the same set.

Definition 3.10 (Convex hull). *For a finite set of elements, $S = \{x_1, \ldots, x_n\}$,*

the convex hull of S is defined by,

$$\operatorname{co} S = \left\{ \sum_{i=1}^{n} \theta_i x_i \;\middle|\; [\theta_1, \ldots, \theta_n] \in \Lambda_n \right\}. \tag{3.12}$$

The convex hull of S can also be defined as the intersection of all convex sets containing S, and therefore it is the minimal convex set containing S.

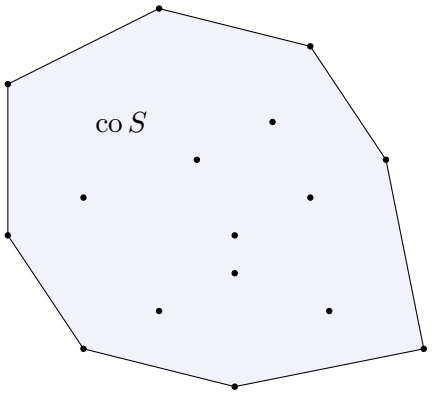

Figure 3.5: Convex hull for a discrete set of elements.

In Figure 3.5 the convex hull of a finite set of elements in the plane is represented by the shaded area.

3.2.5 Operations on Convex Sets

This subsection presents different operations on convex sets that preserve convexity. These different operations are also useful in order to prove or disprove that a given set is convex, and finally it could additionally be used to build a convex set.

Proposition 3.1 (Set operations). *A few basic set operations maintain convexity,*

1. *A finite or infinite intersection of convex sets is convex:*
   if S_α is convex for all $\alpha \in A$, then so is $\cap_{\alpha \in A} S_\alpha$

2. *Convexity is preserved through affine mapping:*
   if f is an affine function and S is a convex set, then $f(S) = \{f(x) \,|\, x \in S\}$ is convex

3. *The inverse image of a convex set by an affine function is also convex:*
   if f is an affine function and S is a convex set, then $f^{-1}(S) = \{x \,|\, f(x) \in S\}$ is convex

4. *The sum of two convex sets is convex:*
   if S_1 and S_2 are convex sets, then $S_1 \oplus S_2 = \{x + y \,|\, x \in S_1,\, y \in S_2\}$ is convex

5. *The projection of a convex set onto some of its coordinates is convex:*
   if S is a convex set, then $p(S) = \{x_1 \,|\, (x_1, x_2) \in S\}$ is convex

6. *The Cartesian product of convex sets is convex:*
   if S_1 and S_2 are convex sets, then $S_1 \times S_2 = \{(x, y) \,|\, x \in S_1, y \in S_2\}$ is also convex

3.2.6 Generalized Inequalities

Generalized inequalities are defined for a proper cone K as a partial ordering, they can be strict or non-strict. These inequalities represent the usual ordering used on $\mathbb{R}$ when $K = \mathbb{R}_+$, also element-wise inequalities on $\mathbb{R}^n$ are defined by generalized inequalities on the proper cone $K = \mathbb{R}^n_+$, the positive orthant in $\mathbb{R}^n$. Subsequently, generalized inequalities subsume the standard ordering on $\mathbb{R}$ as well as the element-wise ordering as special cases. Generalized inequalities on a proper cone K are denoted as follows,

$$x \preceq_K y \Leftrightarrow y - x \in K, \tag{3.13a}$$

$$x \prec_K y \Leftrightarrow y - x \in \operatorname{int} K. \tag{3.13b}$$

Example 3.3 (Positive semi-definite cone). *The generalized inequalities (strict and non-strict) on the positive semi-definite cone $K = \mathbb{S}^n_+$ are defined such that $X \preceq_K Y$ is equivalent to $Y - X$ is positive semi-definite. Since the interior of the proper cone $K = \mathbb{S}^n_+$ is the positive definite cone, $X \prec_K Y$ means $Y - X$ is positive definite. The partial ordering on the proper cone $K = \mathbb{S}^n_+$ is commonly used without any subscript in the literature. The same applies to this thesis, therefore, the previous generalized inequalities become $X \prec Y$ and $X \preceq Y$ respectively for the strict and non-strict inequalities on the positive semi-definite cone. A representation of a portion of the positive semi-definite cone $K = \mathbb{S}^2_+$ in dimension three is provided Figure 3.6.*

Proposition 3.2. *Generalized inequalities have similar properties to the standard ordering on $\mathbb{R}$, for a given proper cone K,*

1. *Generalized inequalities are preserved under addition:*
   $x \preceq_K y$ and $u \preceq_K v$ implies $x + u \preceq_K y + v$

2. *Generalized inequalities are transitive:*
   if $x \preceq_K y$ and $y \preceq_K z$ then $x \preceq_K z$

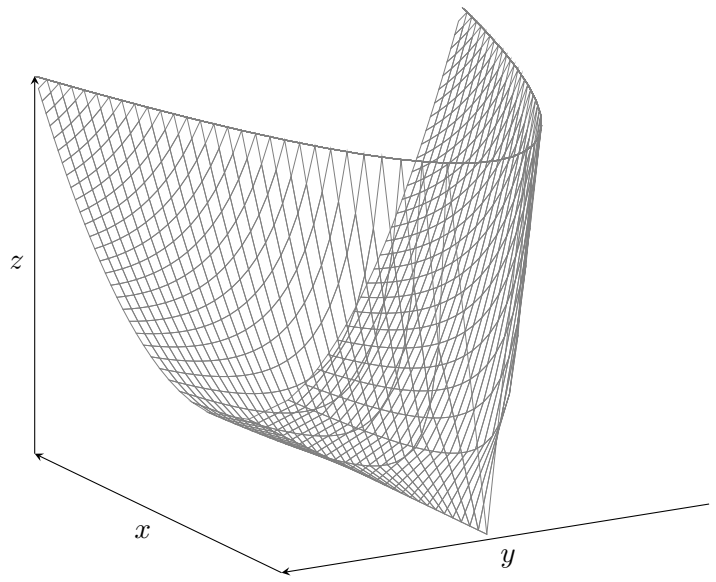

Figure 3.6: Representation of the semi-definite positive cone $\mathbb{S}_+^2$ in dimension three.

3. *Generalized inequalities are maintained under non-negative scaling:*
   if $\alpha \in \mathbb{R}_+$ *and* $x \preceq_K y$ *then* $\alpha x \preceq_K \alpha y$

4. *Generalized inequalities are reflexive:*
   therefore, $x \preceq_K x$

5. *Generalized inequalities are antisymmetric:*
   if $x \preceq_K y$ *and* $x \succeq_K y$ *then* $x = y$

6. *Generalized inequalities are preserved under limits:*
   if for all $i \in \mathbb{N}$, $x_i \preceq_k y_i$ *then* $x \preceq_K y$ *with* $\lim_{i\to\infty} x_i = x$ *and* $\lim_{i\to\infty} y_i = y$

All these properties are inherited from the definition of the proper cone K as well as the definition of the generalized inequalities.

3.3 Convex Functions

3.3.1 Introduction

The study of mathematical functions belongs to a branch of mathematics called analysis. Amongst the different mathematical functions, the convex functions present interesting properties and therefore play a very important role in mathematical optimization. This section define convexity for a function and introduces some of their properties along with the main operations that preserve convexity.

3.3.2 Definition of a Convex Function

Definition 3.11 (Convex function). *A function is said to be convex if its domain of definition* $\operatorname{dom} f$ *is a convex set and if for all elements* x *and* y *belonging to* $\operatorname{dom} f$ *the following inequality holds,*

$$\forall \theta \in [0,1],\ \forall (x,y) \in (\operatorname{dom} f)^2,\ f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y). \tag{3.14}$$

The function f *is said to be strictly convex if the inequality* (3.14) *is a strict inequality on the open segment whenever* x *and* y *are distinct.*

The Figure 3.7 presents a convex function along with one of the chord. It is possible to see that this function complies with the definition of convex functions since the chord is always above the function itself. Also, for linear and affine functions the equation (3.14) is always an equality.

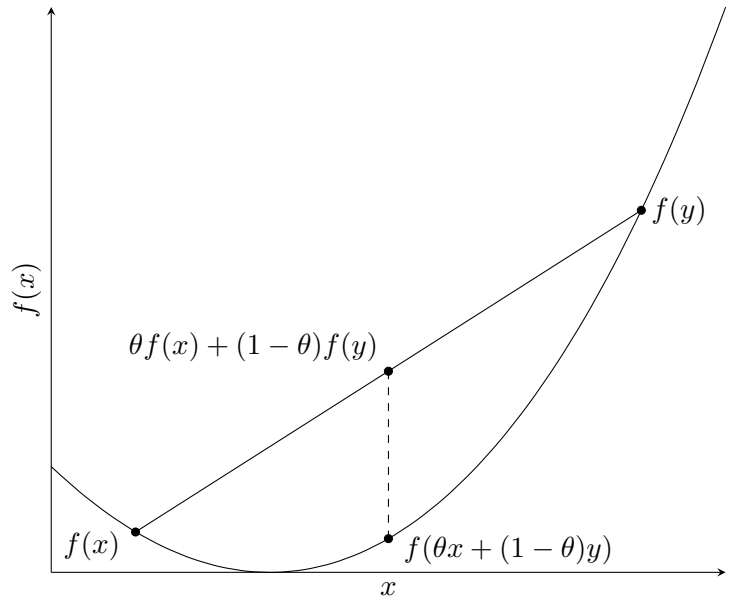

Figure 3.7: Representation of a convex function in two dimensions with a single global minimum.

The function presented in Figure 3.7 is actually a strictly convex function. A given function f is said to be concave if $-f$ is a convex function. Similarly, f is said to be strictly concave is $-f$ is strictly convex. Consequently, linear and affine functions are convex as well as concave, and they are the only functions having this property. In the case where f is differentiable, it is possible to define the affine function g as follows,

$$g(y) = f(x) + \nabla f(x)^{\top}(y - x), \tag{3.15}$$

where $(x, y) \in (\operatorname{int} \operatorname{dom} f)^2$. The function g is the first-order Taylor approximation of f and in the case where f is convex then g is a global under-estimator of the function f. Therefore, it is possible to deduce global properties on f based on local information from the first-order Taylor approximation. This important feature is what confers remarkable properties to convex functions and consequently to convex optimization problems. Very simply, if the derivative of f is equal to zero in x this directly implies that for all $y \in \operatorname{dom} f$, $f(y) \geq g(x)$. The Figure 3.8 represents the epigraph of a convex function and is defined as follows,

Definition 3.12 (Epigraph). *The epigraph of a function f with $\operatorname{dom} f \subseteq \mathbb{R}^n$ is defined by the following set included in $\mathbb{R}^{n+1}$,*

$$\operatorname{epi} f = \{(x, t) \,|\, x \in \operatorname{dom} f,\, t \geq f(x)\}\,. \tag{3.16}$$

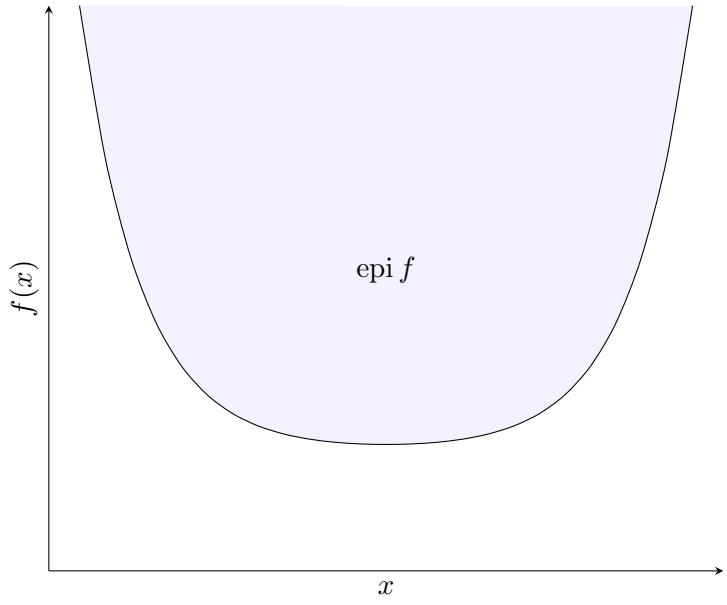

Figure 3.8: Representation of the epigraph for a convex function in dimension two.

The link between convex functions and convex sets is given by the epigraph of a function, a function f is convex if and only if $\operatorname{epi} f$ is a convex set. Figure 3.8 represents the convex epigraph of a convex function in two dimensions.

3.3.3 Examples of Convex Functions

This subsection presents a few examples of convex functions used later on within this thesis. The previous subsection already explained that linear and affine functions are convex functions. Other functions comply with the definition given previously, for example, by definition any norm operator is convex as explained in Example 3.4.

Example 3.4 (Norm). *A norm $\|\cdot\|$ defined on $\mathbb{R}^n$ complies with the triangle inequality and is absolutely homogeneous, therefore, the following relation holds,*

$$\forall \theta \in [0,1], \forall (x,y) \in \mathbb{R}^n, \|\theta x + (1-\theta)y\| \leq \|\theta x\| + \|(1-\theta)y\| \tag{3.17a}$$

$$\Leftrightarrow \|\theta x + (1-\theta)y\| \leq \theta\|x\| + (1-\theta)\|y\|. \tag{3.17b}$$

The relation (3.17a) *is the triangle inequality, the relation* (3.17b) *is linked to the absolutely homogeneous property as well as the fact that θ and $(1-\theta)$ are positive scalars.*

Functions such as power functions with a power greater than one, exponential functions, geometric mean as well as minus log-determinant are all convex functions.

3.3.4 Operations on Convex Functions

This subsection presents some operations that preserve the convexity of convex functions. Therefore, allowing not only to build new convex functions from existing ones but also to check for function convexity.

Proposition 3.3. *Given a set of convex functions $\{f_i \,|\, i \in [\![1,m]\!]\}$ as well as a convex function f the following properties hold,*

1. *A non-negative weighted sum of convex functions defines another convex function, therefore, $g(x) = \sum_{i=1}^{m} w_i f_i(x)$ is convex if for all $i \in [\![1,m]\!]$, w_i is positive*

2. *The composition of a convex function with an affine mapping is convex, i.e. $g(x) = f(Ax+b)$ is a convex function*

3. *The point-wise maximum of a finite set of convex functions defines another convex function $g(x) = \max_{i\in[\![1,m]\!]}\{f_i(x)\}$*

4. *The point-wise supremum of an infinite set of convex functions defines another convex function $g(x) = \sup_{y\in\mathcal{I}} f(x,y)$*

The point-wise maximum and supremum of a set of functions correspond to the intersection of the epigraphs of all the functions belonging to the set. Therefore, it is the intersection of a finite or infinite number of convex sets and it yields a convex set according to the properties of convex sets.

3.4 Convex Optimization Problems

3.4.1 Introduction

This section presents the necessary background knowledge regarding the mathematical optimization used later on within this thesis. Different types of optimization problems are introduced along with their particular notations. Convex optimization is one of the main fields of mathematical optimization. In the following section the main convex optimization programming problems will be presented. An optimization problem consists of finding the minimum of a cost function over a given feasible set defined by constraints. More generally, an optimization problem can be represented as follows,

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & \forall i \in [\![1, m]\!] \\ & f_i(x) \leq b_i, \end{aligned} \tag{3.18}$$

where, the vector $x \in \mathbb{R}^n$ is called the decision variable, $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the cost function or objective function and for all $i \in [\![1, m]\!]$, the functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are the constraint functions respectively associated to the bounds b_i. The set of feasible solutions is defined by the set of decision variables satisfying the constraints. Very often, the first step to solving an optimization problem is to check for feasibility. An optimization problem is said to be feasible if the feasible set defined by the constraint functions is not empty. A solution $x^\star$ is said to be optimal if it has the smallest cost function value amongst all the feasible solutions. If an optimization problem is not feasible, then no optimal solution can be found. Even though feasibility is a necessary condition, it is not a sufficient condition in order to guarantee the existence of an optimal solution. A great variety of optimization problems can be cast in the form of (3.18), the main distinction between these optimization problems come from the properties of the cost and constraint functions. An optimization problem is said to be convex if both the cost function and the feasibility set are convex. The underlying idea is that convex optimization problems are tractable (Rockafellar, 1993). The distinctions between the different types of convex optimization programming problems resides in the type of convex constraint set as well as the type of convex cost functions.

3.4.2 Linear Programming

The simplest convex optimization programming problem is the Linear Programming (LP), the cost function to be minimized is an affine function of the decision variable

and the constraints are all affine inequalities as well as equalities. Linear programming problems trace back to the inequality elimination principle (Fourier, 1827). The standard form of a linear program is given as follows,

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^\top x + d \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b, \end{aligned} \tag{3.19}$$

where $x \in \mathbb{R}^n$ is the decision variable, $G \in \mathbb{R}^{m \times n}$, $A \in \mathbb{R}^{p \times n}$, and h and b are vectors of appropriate dimensions, respectively representing affine component-wise inequality and equality. Any LP can efficiently be solved using standard algorithms such as the simplex method or an interior-point algorithm (Dantzig et al., 1955; Karmarkar, 1984; Nesterov and Nemirovskii, 1994). Linear programs have a simple geometric representation in low dimensions (Boyd and Vandenberghe, 2010).

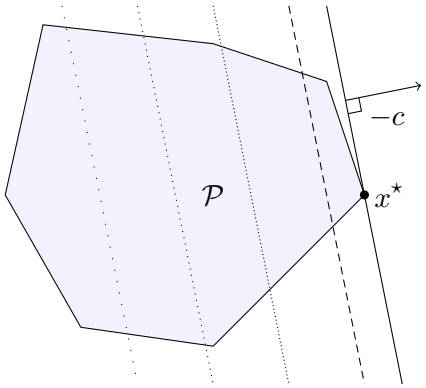

Figure 3.9: Geometric representation of a two-dimensional linear programming problem.

In the Figure 3.9, the feasible set is represented by the closed convex polyhedron $\mathcal{P}$. Since the objective function is affine, the level curves are the hyperplanes orthogonal to the vector c and represented by parallel lines. The optimal solution $x^\star$ is reached by moving inside the polyhedron $\mathcal{P}$ as far as possible in the direction $-c$. The intersection of the solid level line with the polyhedron shows the position of the optimum value. A classical example of LP problem is the diet problem, which consists of finding the cheapest diet satisfying given nutritional requirements. Another typical example is finding the center of a polyhedron (Chebyshev center), by trying to fit the largest ball inside a polyhedron (Boyd and Vandenberghe, 2010).

3.4.3 Quadratic Programming

Another type of convex optimization problem is the Quadratic Programming (QP) optimization, it can be formulated as per the equation (3.20). In this case the cost function is a quadratic function of the decision variable and the constraints are all affine.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^\top Px + q^\top x + r \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b, \end{aligned} \tag{3.20}$$

where $P \in \mathbb{S}^n_+$, $G \in \mathbb{R}^{m\times n}$ and $A \in \mathbb{R}^{p\times n}$. The vectors q, h and b are of appropriate dimensions and r is a scalar. Quadratic programming subsumes linear programming, indeed setting the matrix P to the zero matrix allow to get the LP optimization problem (3.19).

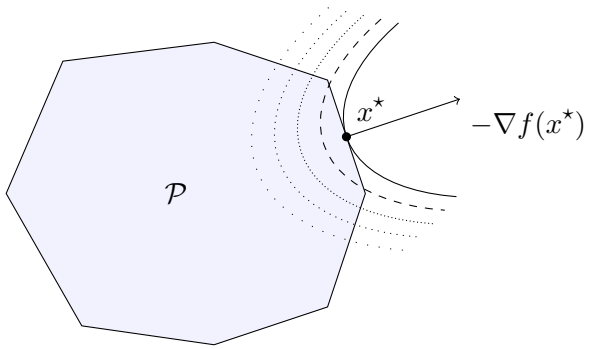

Figure 3.10: Geometric representation of a two-dimensional quadratic programming problem.

The level curves for a quadratic programming problem can be represented as per Figure 3.10, in the two dimensional case they are ellipses. Two types of quadratic programming exists based on the constraint set.

Quadratic Cost with Linear Constraints

The first type of quadratic programming is simply called quadratic programming, it consists of minimizing a quadratic convex cost function over a linear set of constraints. These programming problems are solved very efficiently and differ only from linear programming problems by their objective function. They play a very important role in optimal control when applied to linear discrete time systems.

Quadratic Cost with Quadratic Constraints

The second type of quadratic programming allows not only a quadratic cost function but also some convex quadratic constraints, this type of problem is called Quadratic Constraint Quadratic Programming (QCQP). It subsumes the standard quadratic programming problems with linear constraints and it is a special case of the second-order cone programming case presented within the next subsection.

3.4.4 Second-order Cone Programming

Another type of convex programming problem similar to quadratic programming is called the Second-order Cone Programming (SOCP). It relies on the minimization of a linear cost function over a convex set composed of the intersection of a set of affine constraints with second-order cones as follows,

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f^\top x \\ \text{subject to} \quad & \forall i \in [\![1, m]\!] \\ & \|A_i x + b_i\|_2 \leq c_i^\top x + d_i \\ & Fx = g, \end{aligned} \tag{3.21}$$

where f and c_i are vectors of $\mathbb{R}^n$, A_i and F are matrices of appropriate dimensions, b_i and g are vectors of appropriate dimensions and d_i is a scalar. This optimization formulation framework gathers a wider range of convex optimization programming problems. For instance, it is possible to see that any linear programming problem is subsumed by the second-order cone programming formulation. Indeed, if $A_i = 0$ for all $i \in [\![1, m]\!]$, then the optimization reduces to a LP. Also, if for all $i \in [\![1, m]\!]$, $c_i = 0$ the problem (3.21) is equivalent to a QCQP. Consequently, second-order cone programming subsumes LP and QP. For instance a robust LP can be expressed as a SOCP. Therefore, second-order cone programming is more general than the optimization problems presented previously.

3.4.5 Semi-definite Programming

Different types of proper cones can be used in order to formulate convex optimization problems, as it was the case with second-order cone programming, Semi-definite Programming (SDP) uses the positive semi-definite cone. A semi-definite programming

problem is formulated as follows,

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & F_0 + x_1 F_1 + \ldots + x_n F_n \preceq 0 \\ & Ax = b, \end{aligned} \tag{3.22}$$

where $x \in \mathbb{R}^n$ is the decision variable and for all $i \in [\![0, n]\!]$, F_i is a symmetric matrix of appropriate dimensions, $A \in \mathbb{R}^{p \times n}$ and the vectors b and c are of appropriate dimensions. Semi-definite programming is of particular interest for the development of this thesis and it is used for the online design of optimal control laws. Convex optimization and in particular SDP have been developed and used a lot in control theory for optimal and robust control as well as in signal processing (Boyd et al., 1994; Vandenberghe and Boyd, 1996; El Ghaoui and Niculescu, 2000). The constraint defined by the weighted sum of symmetric matrices is called a Linear Matrix Inequality (LMI) constraint, and defines a convex set. Subsequently, the feasible set of the optimization problem (3.22) is convex as being the intersection of the set $\{x \in \mathbb{R}^n \,|\, F(x) \preceq 0\}$ with the affine set defined such that $\{x \in \mathbb{R}^n \,|\, Ax = b\}$.

Linear Matrix Inequalities

Linear matrix inequalities are expressed under the following form,

$$F(x) = F_0 + \sum_{i=1}^{n} x_i F_i \preceq 0, \tag{3.23}$$

where the matrices F_i are all symmetric of dimension m. The set defined such that $\mathcal{S} = \{x \in \mathbb{R}^n \,|\, F(x) \preceq 0\}$ can be seen as the inverse image of the positive semi-definite cone $\mathbb{S}^m_{++}$ by the affine transformation $-F(x)$, hence it defines a convex set. Note that the set $\mathcal{S}$ can be formulated using a strict or non-strict generalized inequality which respectively defines an open or a closed convex set. From the definition given in equation (3.23), it seems that LMI constraints have a very specific and limited form. However, it is known that linear matrix inequalities can express a large set of problems from control theory such as problems including convex quadratic matrix inequalities, matrix norm inequalities. For instance, the problem of computing a solution to the discrete time Lyapunov equation, proving the stability of an autonomous system can be formulated as follows,

$$P \succ 0, \tag{3.24a}$$

$$A^\top P A - P \prec 0, \tag{3.24b}$$

where the matrix $A \in \mathbb{R}^{n \times n}$ defines the autonomous discrete time system, and the decision variable $P \in \mathbb{S}^n_{++}$ is a candidate Lyapunov solution. The matrix inequalities presented in (3.24) differ from the standard formulation given in equation (3.23), nonetheless it is possible to link the original definition with (3.24) by considering a decomposition of the variable P over the basis of symmetric matrices $\mathbb{S}^n$. Finally, note that multiple LMIs can be concatenated into a single LMI constraint by redefining the matrices F_i, which combines them on the diagonal of a single matrix.

Schur Complement

The Schur complement technique proves the existence of equivalences between LMIs and non-linear matrix inequalities and therefore brings a lot more depth to the use of semi-definite programming. This mathematical tool allows to convert a set of standard LMIs into a set of non-linear but convex inequalities. The different Schur complement equivalences are detailed in Lemma 3.1.

Lemma 3.1. *Consider the following symmetric matrix with affine dependence in x such that*

$$X = \begin{bmatrix} Q(x) & S(x) \\ S(x)^\top & R(x) \end{bmatrix}, \tag{3.25}$$

then the following relations hold,

1. $X \succ 0$ *if and only if* $R(x) \succ 0$, $Q(x) - S(x)R(x)^{-1}S(x)^\top \succ 0$
2. $X \succ 0$ *if and only if* $Q(x) \succ 0$, $R(x) - S(x)^\top Q(x)^{-1}S(x) \succ 0$
3. *If* $Q(x) \succ 0$, *then* $X \succeq 0$ *if and only if* $R(x) - S(x)^\top Q(x)^{-1}S(x) \succeq 0$
4. *If* $R(x) \succ 0$, *then* $X \succeq 0$ *if and only if* $Q(x) - S(x)R(x)^{-1}S(x)^\top \succeq 0$

3.4.6 Cone Programming

A more general modelling framework relying on generalized inequalities and gathering all the optimization problems defined previously exists. This framework is called Cone Programming (CP) and is based on a linear cost function minimized over an affine equality constraint as well as a generalized inequality constraint, it is defined as follows,

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & Fx + g \preceq_K 0 \\ & Ax = b \end{aligned} \tag{3.26}$$

where, K is a proper cone, F and g represent an affine mapping in K, the matrix A belongs to $\mathbb{R}^{p\times n}$ and the vectors b and c are of appropriate dimensions. For example, if the proper cone $K = \mathbb{R}^n_+$ is the non-negative orthant in $\mathbb{R}^n$ then the generalized inequality reduces to the element-wise inequality and therefore the optimization problem (3.26) reduces to a standard linear programming problem. In a similar fashion, if K is defined as the Lorentz cone or as the positive semi-definite cone, then the problem (3.26) is respectively equivalent to a second-order cone programming or a semi-definite programming. More generally, by defining the proper cone K by a Cartesian product as follows $K = K_1 \times \ldots \times K_r$, then the conic inequality of the optimization problem (3.26) can be used to represent any generalized conic inequality.

3.4.7 Summary

This section introduced some of the most common convex optimization programming problems, which all have in common the fact that they rely on disciplined problem modelling. A very well structured optimization model is key in order for a given optimization problem to be cast according to one of the standard forms presented previously. As it was explained before, Figure 3.11 illustrates how the different convex optimization programming problems are nested. From the inside out, linear programming can be seen as a special case of quadratic programming and quadratic programming problems are subsumed by second-order cone programming. Finally, second-order cone programming is subsumed by semi-definite programming that is only a subset of the larger set of cone programming. Cone programming includes different kind of convex cones such as the power and exponential cones not mentioned in detail here, as well as other kind of convex programming such as Geometric Programming (GP). All these optimization problems are similar in the way that they all belong to the category of convex optimization problems, therefore they are tractable and efficiently solvable in polynomial time. These features make them very interesting, especially for online optimization. To conclude, cone programming is a very powerful optimization framework that unifies most of the convex programming problems into disciplined programming.

3.5 Non-convex Optimization Problems

Not all optimization problems are convex and can be efficiently solved or are even tractable. The non-convexity can come from the objective function or can be linked to the non-convexity of the constraints and thus of the feasibility set. Most of the time, these problems are very difficult to solve to global optimality and simply

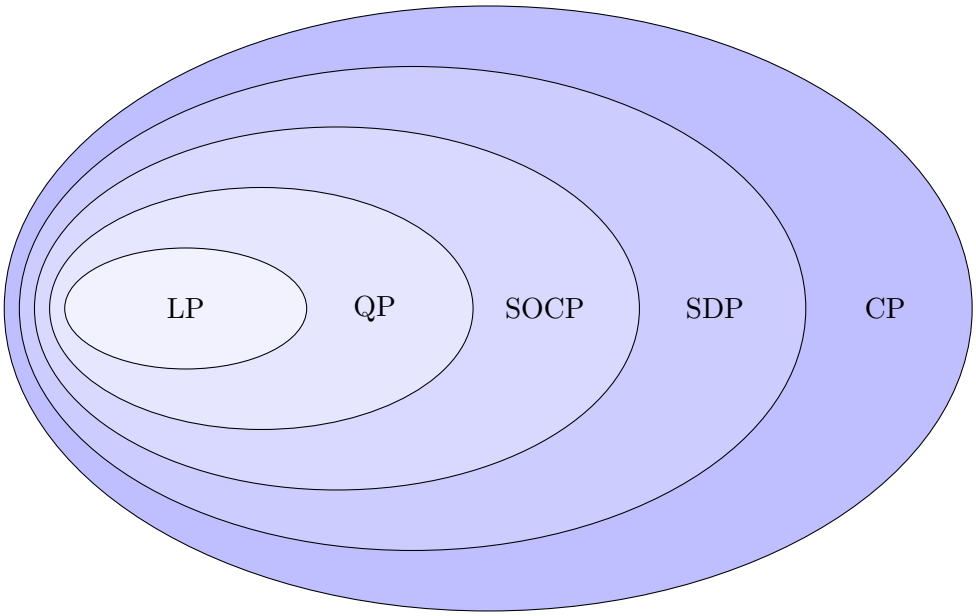

Figure 3.11: Set inclusions between different types of convex optimization problems.

verifying that a solution is optimal can be tedious. This section presents some of the non-convex optimization problems that will be encountered later on within this thesis.

3.5.1 Mixed-integer Linear Programming

The first type of non-convex optimization problem encountered is the Mixed Integer Linear Programming (MILP). This problem is similar to a standard LP problem, with the main distinction that some decision variables are restricted to be integers. The objective function and the constraints are affine functions of the decision variables. The standard for of a MILP is described in equation (3.27).

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^\top x + d \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \\ & x \in \mathbb{R}^n \times \mathbb{Z}^m, \end{aligned} \tag{3.27}$$

where x is the decision variable composed of n real and m integer entries, $G \in \mathbb{R}^{p\times(n+m)}$, $A \in \mathbb{R}^{q\times(n+m)}$ and h and b are vectors of appropriate dimensions. This type of problem comes in different variants, for example if all the decision variables have to take integer values, or if the decision variables have to be only binaries then the problem is called Integer Linear Programming (ILP) and Binary Integer

Linear Programming (BILP) respectively. By definition, the integer constraints are responsible for the non-convexity of the problem. The theory linked to integer linear programming problems is well developed and solving techniques have been created to avoid an exhaustive search of the feasible set (Schrijver, 1998).

3.5.2 Bilinear Matrix Inequalities

Biaffine matrix inequalities, also called Bilinear Matrix Inequality (BMI), are a generalization of LMI constraints. However, except in special cases bilinear matrix inequalities represent non-convex constraints and therefore, optimization problems involving BMIs are computationally difficult to solve to global optimality (Goh et al., 1995). Very often only one of the multiple local optima can be achieved easily. A standard form of the BMI optimization problem is given in equation (3.28).

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c_x^\top x + c_y^\top y \\ \text{subject to} \quad & F_0 + \sum_{i=1}^{n} x_i F_i + \sum_{j=1}^{m} y_j G_j + \sum_{i=1}^{n}\sum_{j=1}^{m} x_i y_j H_{ij} \preceq 0 \\ & A_x x = b_x \\ & A_y y = b_y, \end{aligned} \tag{3.28}$$

where $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ are the decision variables, for all $(i,j) \in [\![1,n]\!] \times [\![1,m]\!]$, the matrices F_0, F_i, G_i ad H_{ij} are symmetric with appropriate dimensions. The matrices A_x and A_y as well as the vectors c_x, b_x, c_y and b_y are of appropriate dimension. From the BMI problem formulation (3.28) it can be seen that fixing the variable x or y yields a semi-definite programming problem respectively in y and in x (VanAntwerp and Braatz, 2000). Consequently, one way to handle bilinear matrix inequalities is to solve them as alternate SDP problems (Goh et al., 1995). Bilinear matrix inequality problems are equivalent to LMI problems with rank constraints, and it is well known that rank constrained optimization problems are complex to solve (Recht et al., 2007).

3.6 Algorithms

This section presents the main algorithms and methods used to solve convex as well as non-convex programming problems. The very first algorithms developed to tackle linear programming problems called the Simplex method, the ellipsoid algorithm and the interior-point methods are discussed here.

3.6.1 Simplex Algorithm

The simplex algorithm has been developed to solve linear programming problems, it relies on the fact that if an optimal solution exists, it can be found at one of the vertex of the polytopic feasible set (Dantzig et al., 1955). In addition to this, it has been shown that if the objective function is not optimal when evaluated on a vertex, then an edge containing the vertex and moving away from it while improving on the objective function exists. The Simplex algorithm will use this principle to visit a sequence of feasible vertices such that the objective function always improves. The algorithm is terminated either when there is no improvement possible or if an improving edge is not bounded. In order to achieve this, all the constraints are transformed into equality constraints with the introduction of slack variables, also called basic variables. The standard LP defined before is recast in a new standard form presented in (3.29) used for the pivoting iterations of the simplex algorithm.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{3.29}$$

The linear programming problem is formulated using a tableau highlighting the improving directions while all the variables comply with the constraints of the initial optimization problem. Initializing the simplex method consist of computing an initial basic feasible solution used as a starting point for the following pivoting iterations. In the case where no basic feasible solutions can be computed, the linear program is infeasible.

Algorithm 3.1: Simplex algorithm.

Inputs **:** Linear program tableau
Initialization: Compute basic feasible solution
while *There exists an improvement direction* **do**
  Select improving pivot variable x_i
  Find best pivot entry a_{ij}
  Complete pivot operation around element a_{ij}
end
return
$x^\star$
$p^\star = c^\top x^\star$

The simplex algorithm is used a lot in practice to solve linear programming problems, even though it has been shown that the worst case complexity of this method is exponential (Klee and Minty, 1972). Consequently, other techniques have

been developed in order to improve on the worst case exponential complexity of the simplex. These new methods have been first used to solve LP, then, later on they have been generalized to all convex optimization problems.

3.6.2 Ellipsoid Algorithm

The ellipsoid algorithm relies on a sequence of ellipsoids decreasing in volume and always containing the optimal value of the optimization problem (Shor, 1977). It is the first method that has been shown to have a polynomial complexity when used to solve linear programs (Khachiyan, 1980). However, it can be applied to a wide variety of convex optimization problems. The main idea is that given a subgradient of the function to minimize evaluated at the center of the ellipsoid, it is possible to find a half-space that does not contain the optimal point. The half-space is defined by a hyperplane splitting the ellipsoid in two halves, one half contains the optimal value and the other one is pruned at the next iteration. The next iteration starts with the computation of a new ellipsoid of minimal volume that contains the intersection of the previous ellipsoid with the half-space known to include the optimal value. This process is repeated until the volume of the ellipsoid reaches a critical value. It has been proven that with this technique the volumes of the ellipsoids are decreasing geometrically. A special case of this technique in dimension one is the bisection method, in this case the sequence of ellipsoids are decreasing intervals on $\mathbb{R}$. The initialization of the algorithm is done with an initial ellipsoid $\mathcal{E}_0$, defined by a shape P_0 and a center x_0 and known to contain the optimal value of the objective function f as follows,

$$\mathcal{E}_0 = \left\{ x \in \mathbb{R}^n \,\middle|\, (x - x_0)^\top P_0^{-1} (x - x_0) \leq 1 \right\}. \tag{3.30}$$

Updating the ellipsoid at a given step k is performed by computing the ellipsoid of minimum volume such that it contains the intersection of the previous ellipsoid $\mathcal{E}_k$ with the half-space defined by the subgradient g_k.

$$\mathcal{E}_{k+1} \supseteq \mathcal{E}_k \cap \left\{ x \in \mathbb{R}^n \,\middle|\, g_k^\top (x - x_x) \leq 0 \right\} \tag{3.31}$$

Then these steps are repeated until the volume of the ellipsoid reaches a critical value, ensuring a certain bound on the optimal value.

The Algorithm 3.2 presented the unconstrained ellipsoid algorithm, nonetheless, it is possible to include convex constraints by following the gradient of a violated constraint, alternating between unconstrained ellipsoid iterations when no constraints are violated and feasible iterations when a constraint is violated. The ellipsoid method provides a solid theoretical background with polynomial complexity stand-

Algorithm 3.2: Ellipsoid algorithm.

Inputs : Initial ellipsoid $\mathcal{E}_0$ with center x_0
Initialization: Set $k = 0$
while $\sqrt{g_k^\top P_k g_k} \geq \varepsilon$ **do**
  Compute subgradient: g_k at x_k
  Normalize subgradient: $\tilde{g} = \frac{g_k}{\sqrt{g_k^\top P_k g_k}}$
  Update ellipsoid center: $x_{k+1} = x_k - \frac{1}{n+1} P_k \tilde{g}$
  Update ellipsoid shape: $P_{k+1} = \frac{n^2}{n^2-1}\left(P_k - \frac{2}{n+1} P_k \tilde{g}\tilde{g}^\top P_k\right)$
  $k = k + 1$
end
return
$x^\star = \text{argmin}_{i \in [\![1,k]\!]}\left(f(x_i)\right)$
$p^\star = \min_{i \in [\![1,k]\!]} f(x_i)$

ing as a generalization of the bisection method in higher dimensions. Nonetheless, it is not used a lot in practice due to the fact that better methods, such as the interior-point method have been developed more recently. Compared with the simplex method, the ellispoid algorithm does not move along the vertices on the outer part of the feasible set but converges from one starting point towards the optimum. A similar technique relying on an initial feasible point located within the constraint set has been developed to solve convex programs. This technique presented within the next subsection includes multiple variants called interior-point methods.

3.6.3 Interior-point Methods

Interior-point methods offer powerful polynomial time and practically applicable algorithms. This family of techniques has been introduced initially to tackle linear programming problems (Karmarkar, 1984), they have been later on generalized to all convex programming problems (Nesterov and Nemirovskii, 1994). This algorithm relies on a barrier function that is smooth and convex in the interior of the feasibility set and tends to infinity on the boundary.

Lagrange Duality

A convex optimization problem called the primal is linked to its dual problem by the Lagrange dual function. The basic idea behind Lagrange duality is to add the

constraints to the objective function to create an augmented objective function.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & \forall (i,j) \in [\![1,m]\!] \times [\![1,p]\!] \\ & f_i(x) \leq 0 \\ & h_j(x) = 0 \end{aligned} \tag{3.32}$$

The Lagrangian associated with the optimization problem (3.32) with decision variable $x \in \mathbb{R}^n$ is therefore expressed as follows,

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \nu_j h_j(x), \tag{3.33}$$

where λ and ν are called the Lagrange multiplier vectors, and $\text{dom}\, L = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$, with $\text{dom}\, f_0 = \mathcal{D}$. The Lagrange dual is the function calculated such that,

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu). \tag{3.34}$$

Consequently, as the infimum of a family of affine functions the Lagrange dual is concave. An important property of the Lagrange dual function is that any feasible solution x for the optimization problem (3.32), and any pair of Lagrange multipliers $(\lambda, \nu) \in \mathbb{R}_+^m \times \mathbb{R}^p$ provide a lower bound on the optimal value of the optimization problem noted $p^\star$.

$$g(\lambda, \nu) \leq p^\star \tag{3.35}$$

Subsequently, an important property of the Lagrange dual function is that it provides a non-trivial lower bound on the optimum value of the optimization problem. This has been achieved by formulating the hard constraints of the optimization problem (3.32) with soft constraints in an augmented cost function. Naturally, one will want to find the best lower bound on the optimal value $p^\star$ using the Lagrange dual function using a new optimizatin problem, formulated as per (3.36).

$$\begin{aligned} \underset{\lambda, \nu}{\text{maximize}} \quad & g(\lambda, \nu) \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned} \tag{3.36}$$

This new optimization problem is called the dual problem, while the previous problem presented in equation (3.32) is called the primal problem. The Lagrange dual consists of maximizing a concave function and therefore is a convex optimization problem. This property holds regardless of the convexity properties on the primal

problem.

Weak duality The optimal value of the Lagrange dual obtained as the optimal value of (3.36) is denoted by $d^\star$. It has been already established that the following inequality holds,

$$d^\star \leq p^\star. \tag{3.37}$$

The difference between $p^\star$ and $d^\star$ is called the optimal duality gap, this gap is always positive according to (3.35). This property is called weak duality and always holds even if one of the bounds is not finite.

Strong duality The property of strong duality holds when the optimal duality gap is zero and it naturally follows that,

$$d^\star = p^\star. \tag{3.38}$$

Strong duality holds when the primal is convex and under constraint qualification conditions. For example, if the primal is strictly feasible, in other word if there exists an element in the relative interior of the feasible set, then strong duality holds. The property of strong duality can be used as a certificate proving that optimality has been reached, and subsequently it provides a very powerful stopping criteria for the interior-point algorithm.

Optimization algorithm

The duality theory is used in order to solve convex optimization problems as well as to certify that the optimal solution has been obtained, therefore providing a termination criteria for the primal dual interior-point algorithm. Some conditions used as the generalization of the optimality conditions for unconstrained convex differentiable functions have been developed in the case of constrained convex optimization problems, they are called the KKT conditions (Brezhneva et al., 2009). Simply using the gradient of the objective function to compute the optimal solution of a constrained convex optimization problem is in general difficult. Subsequently, an algorithm computing a sequence of points converging to the optimal solution, called minimizing sequence, is implemented in practice. Different versions of the interior-point algorithm exist, the main variants are the barrier method and the primal dual interior-point algorithm. The barrier method relies on the definition of a self-concordant barrier function given in equation (3.39) for the optimization

problem (3.32).

$$\phi(x) = -\sum_{i=1}^{m} \log(-f_i(x)) \tag{3.39}$$

The barrier function $\phi(x)$ can be understood as a force field that keeps the objective function away from the inequality constraints. Changing the balance of this force field is achieved by weighing the sum composed of the objective function and the barrier function. The idea is to minimize the equality constrained problem and to increase the weight linked to the objective function until a stopping criteria is reached. This barrier function can be used for generalized inequalities. However, more efficient than the barrier method is the primal dual interior-point algorithm, an alternative that is presented in Algorithm 3.3. This technique is similar to the barrier method, however the update is performed on both the primal and dual variables by applying Newton's method directly to the KKT equations.

Algorithm 3.3: Primal dual interior-point algorithm.

Inputs : Strictly feasible primal and dual solution (x_0, λ_0, ν_0)
Initialization: Set $k = 0$
while $\|r_{pri}\| \geq \varepsilon_{feas}$ *or* $\|r_{dual}\| \geq \varepsilon_{feas}$ *or* $\hat{\eta} \geq \varepsilon$ **do**
  Set $t = \mu \frac{m}{\hat{\eta}}$
  Compute primal dual search $\Delta_{pd} = \left(\Delta x_{pd}, \Delta\lambda_{pd}, \Delta\nu_{pd}\right)$
  Line search with step length s
  Update primal and dual variables
  $(x_{k+1}, \lambda_{k+1}, \nu_{k+1}) = (x_k, \lambda_k, \nu_k) + s\left(\Delta x_{pd}, \Delta\lambda_{pd}, \Delta\nu_{pd}\right)$
  $k = k + 1$
end
return
$x^\star = x_k$
$p^\star = f_0(x^\star)$
$d^\star = g(\lambda_k, \nu_k)$

The variables r_{pri} and r_{dual} correspond respectively to the primal and dual residuals. The scalar μ denotes the increasing factor of the variable t, $\hat{\eta}$ represents the duality gap when the solution obtained is primal and dual feasible, and the constants ε and ε_{feas} denote the problem tolerance. The primal dual interior-point algorithm terminates at the iteration k, when x_k is primal feasible, (λ_k, ν_k) are dual feasible and the duality gap is smaller than the required ε precision. Note that the Lagrangian can be defined for generalized inequalities by replacing the Lagrange multiplier constraints with dual positive generalized inequalities. In addition to this, the KKT conditions have also been generalized to handle optimization problems including generalized inequalities. Therefore, the same primal dual interior-point method

can be extended to the more general case of conic programs using generalized inequalities. This is the generalization of interior-point methods that confers a lot of practicality and importance to this technique (Potra and Wright, 2000).

3.6.4 Branch and Bound Algorithm

The branch and bound technique has been proposed originally to tackle linear integer programming problems (Land and Doig, 1960). The main idea behind this optimization technique is that a mixed integer linear program is relaxed into a standard linear programming problem in order to be solved, this initial problem is called the root node. The next step is to branch the search space into multiple mixed integer linear programs that are solved and compared to an upper and a lower bounds on the solution, the algorithm stores the best achievable incumbents during the search. The initial problem is divided into a set of subproblems organized in the shape of a rooted tree graph, branching is done when an integer variable does not have an integer value in the solution for the relaxed problem. The branches of the tree correspond to the original problem solved over a smaller feasible set. If it is found that a branch cannot provide a better solution, it is pruned from the search tree. This process is performed within the search tree in a top down fashion until the best solution is found.

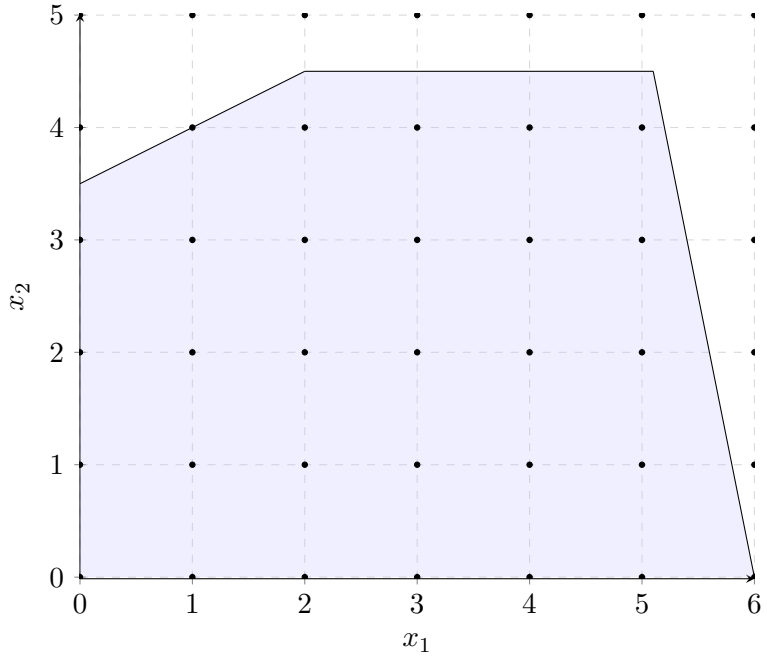

Figure 3.12: Representation of the feasible set for an integer linear program.

The branch and bound algorithm uses the linear relaxation of the integer problem

Algorithm 3.4: Branch and bound algorithm.

Inputs : Problem instance
Initialization: $L = \{P_0\}$, $J^\star_{feas} = +\infty$
while $L \neq \emptyset$ **do**
  Select node P in active set L
  Remove node P from active set L
  Branch problem P into problems $\{P_1, \ldots, P_k\}$
  for $i = 1$ **to** k **do**
    Compute lower bound J_i and solution x_i by solving the relaxed problem P_i
    if $J_i < J^\star_{feas}$ **then**
      if x_i *solution of initial problem* **then**
        Set $J^\star_{feas} = J_i$
        else
          Add problem P_i to active set L
        end
      end
      else
        Discard node P_i
      end
    end
  end
end
return
$x^\star = \text{argmin}\left(J^\star_{feas}\right)$
$p^\star = J^\star_{feas}$

in order to obtain an optimal mixed integer solution. In the worst case scenario the algorithm will evaluate all the possible nodes and perform an exhaustive search over the integer decision variables. Subsequently, the worst case computational complexity is exponential. Multiple variants have originated from the branch and bound technique, such as the branch and cut method. This last technique uses cuts in order to tighten the bounds obtained during the linear relaxation of the integer decision variables in the branches. The distinctions between these algorithms is found in the branching and cutting strategies. The integer values located within the shaded are presented in Figure 3.12 represent the feasibility set of a integer linear program. The maximization of x_1 and x_2 would lead to the top right vertex in the relaxed LP, the branch and bound would therefore branch the search space in two subspaces where the initial problem would be relaxed and solved again. The branch and cut would branch and then apply some integer cuts to the search space and terminates with the optimal solution.

3.7 Summary

This section has presented the principal properties and definitions of convex sets and convex functions which led to the formulation of convex optimization problems. In addition, some background has been provided regarding the formulation of standard non-convex optimization problems that are frequently encountered in control and systems theory. It is interesting to notice that even the non-convex problems rely on convex optimization techniques to obtain global or local solutions. Finally, the main algorithms and methods used to solve these optimization problems have been introduced. All these techniques will be used within the next chapters in order to solve the control architecture problems as well as the optimal control problems.

Chapter 4

Weak Interactions System Partitioning Using Binary Integer Linear Programming

4.1 Introduction

System models are widely used in control design especially with the development of techniques such as Model Predictive Control (MPC) (Richalet et al., 1978; Maciejowski, 2002; Rawlings and Mayne, 2009). Systems are growing in size and complexity and they are in most cases composed of interacting subsystems (Scattolini, 2009). For these large scale systems, the design of a centralized controller can be prohibitive due to the heavy computational resources required (Mayne, 2014; Adelipour et al., 2017). Also, if the system is geographically spread out, communication delays between the centralized controller and the actuators and sensors arise. An appropriate decomposition of the main system into subsystems could improve the system performance, ease the implementation of distributed control as well as bring a reduction in communication requirements. One way to solve this problem is to see the system as a concatenation of subsystems and to design local controllers for each subsystem (Xie et al., 2016). In a top-down approach the full model of the multivariable system is partitioned into subsystem models so that the decentralized controller can be designed. Decentralized control has been studied for decades and design procedures have been established (Šiljak, 1991; Lunze, 1992; Bakule, 2008). However, the system model partitioning problem has been overlooked, often because the system is already composed of physical subsystems. Every subsystem model is defined by a set of states and inputs. The weak interaction partitioning problem consist of defining these sets in order to minimize the coupling between subsystem

models. In other words, the problem of partitioning a dynamical system into subsystems can be seen as a packing problem where the aim is to pack the set of sensors and actuators in subsystems. In addition to this, a cost function representing the total level of interaction is minimized and constraints are employed such that each subsystem remains controllable. For instance, strongly coupled subsystem models can emerge from the main system model, particularly within chemical plants (Stewart et al., 2010) or heating systems (Moroşan et al., 2010). The ideal partitioning of a system model would yield completely decoupled subsystem models.

Defining the subsystems of a plant has been done in different ways in the past. One of the first methods employed to couple inputs and outputs was the relative gain array (Bristol, 1966). This method is used to find the best pairing at steady state between inputs and outputs and hence to choose the most relevant input to control a given output in a multi-input multi-output system. It can be seen as a response to the industrial need to control a multivariable process as a combination of single variable processes. The relative gain array has been extended to the block relative gain, allowing for suitable pairing for block decentralized control (Manousiouthakis et al., 1986; Kariwala et al., 2003). The extension of the relative gain array allows the design of multivariable controllers in a decentralized way. However it only links inputs and outputs together and does not provide a partitioning of the plant model. A technique similar to the relative gain array, is the Nyquist array method, allowing the design of single-input single-output controllers after rendering the model diagonally dominant (Leininger, 1979; Chen and Seborg, 2003). System partitioning can be performed by seeking the least interacting groups. Another technique used for system decomposition and integration is the design structure matrix also known as the dependency structure matrix or interaction matrix (Browning, 2001). This technique indicates the link between the elements it represents, moreover the links are directed. Elements along a row indicate that a contribution is provided to other elements whereas elements along a column indicates a dependency from other parts of the system. The attribution of weights within the interaction matrix is used in order to perform clustering and achieve system decomposition. Another similar technique employed clustering along with a genetic algorithm optimal search in order to (Xie et al., 2016). A graph partitioning algorithm has also been used in order to decompose a system into subsystems, it relies on a translation of the system model into a graph (Ocampo-Martinez et al., 2011). Then the partitioning is performed by seeking highly connected sub-graphs, also reducing the number of interconnections between them. Other works on decentralized control combined the controller design along with the controller topology, these two aspects are combined in an optimization function yielding a trade-off between the need for feedback

links and the loss of performance compared to the centralized controller architecture (Schuler et al., 2014). Finally, other works have studied the actuator partitioning problem (Jamoom et al., 1998; Motee and Sayyar-Rodsari, 2003). To the best of the author knowledge the problem addressing state space model partitioning has not been studied. Therefore this partitioning approach is a standalone work, making any real comparison difficult.

In this chapter, a binary integer linear programming based approached is proposed to the problem of partitioning a system model into a set of non-overlapping but coupled subsystem models, or overlapping subsystem models. The objective is to reduce the magnitude of the interactions between the subsystem models. Finally, cuts are added to rule out non-controllable partitionings in order for the algorithm to yield only controllable subsystem models.

Most of the content presented within this chapter has been already published (Guicherd et al., 2017). The chapter is organized as follows, section 4.2 states the problem and section 4.3 introduces the required notations leading to the optimization problem formulation. Section 4.4 demonstrates how the problem can be relaxed into a integer linear programming problem. In section 4.5 the partitioning cut principle is presented allowing to obtain only controllable subsystems. Section 4.6 explains the linear partitioning algorithm along with one of the auxiliary algorithm used to extract the subsystem models. Section 4.7 proposes a discussion dealing with the linear partitioning problem size as well as the partitioning algorithm complexity. In section 4.8, the system and subsystem graph representations are introduced. In order to illustrate the efficacy of the partitioning algorithm section 4.9 includes some numerical examples, finally section 4.10 concludes the chapter.

4.2 Problem Statement

Given a continuous linear time invariant controllable state space model defined by

$$\dot{x} = Ax + Bu, \tag{4.1}$$

where, the matrix A is the state matrix and the matrix B is the input matrix respectively with the appropriate sizes for n states and m inputs, therefore, $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$. Partitioning the system model (4.1) consists of decomposing the inputs as well as the states into groups representing subsystems. For a given number of partitions $N \in [\![2, \min(n, m)]\!]$ and for any subsystem $p \in [\![1, N]\!]$ the subsystem model

indexed by p can be expressed as follows,

$$\dot{x}_p = A_{pp}x_p + B_{pp}u_p + \sum_{\substack{j=1 \\ j\neq p}}^{N} \left\{ A_{pj}x_j + B_{pj}u_j \right\}, \tag{4.2}$$

with, for all $p \in [\![1, N]\!]$, $x_p \in \mathbb{R}^{n_p}$ and $u_p \in \mathbb{R}^{m_p}$ such that

$$\sum_{p=1}^{N} n_p \geq n \tag{4.3a}$$

$$\sum_{p=1}^{N} m_p = m. \tag{4.3b}$$

The weak interactions partitioning problem consists of minimizing the magnitude of the right-hand side sum in (4.2) for the subsystems while keeping each of them controllable. An ideal partitioning of the system (4.1) would yield completely decoupled subsystems, consequently the right-hand side sum in (4.2) would always be equal to zero regardless of the state and input variables value. A non-overlapping condition for the states and the inputs is imposed by (4.3) in the case of the equality for equation (4.3a) and a state overlapping condition is allowed in the case of the strict inequality, providing that all the state variables are used in the partitioning. The next section presents the decision variables, the constraints as well as the subsystem to subsystem interaction metric necessary in order to formulate the weak interactions partitioning optimization problem.

4.3 Weak Interactions Problem Formulation

4.3.1 Decision Variables

A decision variable is associated with the couples formed by a group p and a state i as well as a group p and an input j. All the decision variables are binary variables. They are organized in two grouping matrices, the state grouping matrix $\alpha \in [\![0, 1]\!]^{N\times n}$ and the input grouping matrix $\beta \in [\![0, 1]\!]^{N\times m}$. Therefore, the rows of α and β represent the N groups and the columns represent the n states and the m inputs respectively. For example, the partitioning of a system with $n = 5$ states and $m = 3$ inputs into $N = 2$ subsystems could be given by the non-overlapping state and input grouping

matrices (4.4).

$$\alpha = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \beta = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.4}$$

In this example, the first three states belong to the first group (red subsystem) and the fourth and fifth states compose the second group (blue subsystem) (4.5). Concerning the input matrix, the first subsystem (red subsystem) includes the first and third inputs to control the first, second and third states, whereas the second subsystem (blue subsystem) is composed of the second input in order to control the fourth and fifth states as presented respectively in equation (4.5) and equation (4.6).

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} \tag{4.5}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \\ b_{51} & b_{52} & b_{53} \end{bmatrix} \tag{4.6}$$

All the matrix elements not included in one of the subsystem models represent the state and input interactions between subsystems, respectively within the state and input matrices. The partitioning of A relies only on the grouping matrix α whereas the partitioning of B relies on both α and β grouping matrices. Without loss of generality, for a non-overlapping system partitioning the subsystem models can always be represented by block matrices along the state matrix diagonal after permutation of the states order and by disjoint block matrices composing the input matrix after possible permutation of the inputs. An example of a state overlapping partitioning is provided equation (4.7), in this case a state variable can be shared by multiple subsystems as it is presented equation (4.8) with the third state variable.

$$\alpha = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \beta = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.7}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} \tag{4.8}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \\ b_{51} & b_{52} & b_{53} \end{bmatrix} \tag{4.9}$$

Hence, a specific system model partitioning is represented by a pair of state and input non-overlapping or overlapping grouping matrices, even if this representation is not unique. Indeed, any new grouping matrices obtained by simultaneous permutation of the rows of α and β simply represent the same partitioning. Such a permutation is allowed because the subsystem models are not ordered or labelled and consequently it would correspond to changing the subsystem labels. More specifically, the columns of the grouping matrices are composed of zeros and a single one in the non-overlapping case. The one is positioned in the row representing the group where the state or input belongs respectively for a state and an input non-overlapping grouping matrix. Multiple ones can be located in the same column of a state grouping matrix in the case of state overlapping partitioning. The next subsection presents the linear constraints restricting the decision variables α and β in the integer optimization problem formulation.

4.3.2 Partitioning Constraints

The formulation of constraints on the decision variables is necessary in order for the algorithm to return a solution complying either with the rules defining non-overlapping subsystem models or state overlapping system models. The partitioning rules along with their mathematical equivalence as linear constraints are expressed as per equation (4.10) and equation (4.11), respectively for the non-overlapping and overlapping system model partitionings.

Non-overlapping Partitioning Constraints

1. Each state group contains at least a state, hence, no state group can be empty and the partitioning has the correct number of state groups

$$\forall p \in [\![1, N]\!], \sum_{i=1}^{n} \alpha_{pi} \geq 1 \tag{4.10a}$$

2. Each input group contains at least an input, hence, no input group can be empty and the partitioning has the correct number of input groups

$$\forall p \in [\![1, N]\!], \sum_{i=1}^{m} \beta_{pi} \geq 1 \tag{4.10b}$$

3. A state can be in only one state group, therefore, the multiple use of a state is prevented and the non-overlapping state requirement is respected

$$\forall i \in [\![1, n]\!], \sum_{p=1}^{N} \alpha_{pi} \leq 1 \tag{4.10c}$$

4. An input can be in only one input group, therefore, the multiple use of an input is prevented and the non-overlapping input requirement is respected

$$\forall i \in [\![1, m]\!], \sum_{p=1}^{N} \beta_{pi} \leq 1 \tag{4.10d}$$

5. Each state must belong to a state group, consequently, no state is left out of the optimization problem

$$\forall i \in [\![1, n]\!], \sum_{p=1}^{N} \alpha_{pi} \geq 1 \tag{4.10e}$$

6. Each input must belong to an input group, consequently, no input is left out of the optimization problem

$$\forall i \in [\![1, m]\!], \sum_{p=1}^{N} \beta_{pi} \geq 1 \tag{4.10f}$$

Overlapping Partitioning Constraints

1. Each state group contains at least a state, hence, no state group can be empty and the partitioning has the correct number of state groups

$$\forall p \in [\![1, N]\!], \sum_{i=1}^{n} \alpha_{pi} \geq 1 \tag{4.11a}$$

2. Each input group contains at least an input, hence, no input group can be empty and the partitioning has the correct number of input groups

$$\forall p \in [\![1, N]\!], \sum_{i=1}^{m} \beta_{pi} \geq 1 \tag{4.11b}$$

3. The total number of state variables used is set to $n + q$ to allow for q state variables to overlap

$$\sum_{p=1}^{N} \sum_{i=1}^{n} \alpha_{pi} = n + q \tag{4.11c}$$

4. An input can be in only one input group, therefore, the multiple use of an input is prevented and the non-overlapping input requirement is respected

$$\forall i \in [\![1, m]\!], \sum_{p=1}^{N} \beta_{pi} \leq 1 \tag{4.11d}$$

5. Each state must belong to at lest one state group, consequently, no state is left out of the optimization problem

$$\forall i \in [\![1, n]\!], \sum_{p=1}^{N} \alpha_{pi} \geq 1 \tag{4.11e}$$

6. Each input must belong to an input group, consequently, no input is left out of the optimization problem

$$\forall i \in [\![1, m]\!], \sum_{p=1}^{N} \beta_{pi} \geq 1 \tag{4.11f}$$

It can be noted that some pairs of inequality constraints, in both the non-overlapping and the overlapping case could be expressed as one equality constraint, without affecting the formulation of the partitioning problem. The constraints are

expressed for the two grouping matrices, however only three different sets of constraints concern each type of grouping matrix. Because α and β are arrays of binary variables a natural implicit constraint links n, m and N.

$$1 < N \leq \min(n, m) \tag{4.12}$$

Subsystem interactions can come from the state matrices or the input matrices, the next subsection presents how these interactions can be formulated firstly using the block matrix form and secondly using the state space model entries.

4.3.3 Objective: Minimizing Subsystem Interactions

The first part of the interactions comes from the state matrices. The subsystem model (4.2) presents the couplings with the other subsystems in the form of a sum, this sum can be split into the state interactions and the input interactions. For a given number of partitions $N \in [\![2, \min(n, m)]\!]$ and for any subsystem $p \in [\![1, N]\!]$ the state interactions can be expressed by,

$$J_p^{state} = \sum_{\substack{j=1 \\ j \neq p}}^{N} \left\| \text{vec}\left(A_{pj}\right) \right\|_1 . \tag{4.13}$$

The expression written in block matrix form can also be represented using the state matrix elements as well as the state grouping matrix elements as follows,

$$J_p^{state} = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{pi} |a_{ij}| \left(1 - \alpha_{pj}\right) . \tag{4.14}$$

The elements from the state grouping matrix are used here as boolean tests to take into account only the interactions acting on the subsystem p and coming from the other subsystem states. A similar reasoning is applied to quantify the interactions coming from the input matrices,

$$J_p^{input} = \sum_{\substack{j=1 \\ j \neq p}}^{N} \left\| \text{vec}\left(B_{pj}\right) \right\|_1 . \tag{4.15}$$

In a similar fashion, equation (4.15) can also be represented using the input matrix elements as well as the elements of the state grouping matrix combined with the elements of the input grouping matrix, such that,

$$J_p^{input} = \sum_{i=1}^{n} \sum_{k=1}^{m} \alpha_{pi} |b_{ik}| \left(1 - \beta_{pk}\right) . \tag{4.16}$$

Likewise, the elements from the state grouping matrix combined with the elements of the input grouping matrix are used as boolean tests to take into account only the interactions acting on the subsystem p and coming from the other subsystems inputs. After having defined the two types of interactions, the full interaction metric can be calculated. Consequently, the last step is to pose the weak interactions optimization problem like it is presented within the next subsection.

4.3.4 Weak Interactions Optimization Problem

The overall formulation of the weak interactions optimization problem is obtained by summing the interactions coming from the states and the inputs over the N subsystems as follows,

$$J^{interaction} = \sum_{p=1}^{N} \left\{ J_p^{state} + J_p^{input} \right\}. \tag{4.17}$$

Hence, the integer linear optimization problem can be formulated using the same notation as the one employed in equation (4.14) and equation (4.16) and is expressed by the optimization problem provided in equation (4.18).

$$\begin{aligned}
\underset{\alpha,\beta}{\text{minimize}} \quad & \sum_{p=1}^{N}\sum_{i=1}^{n} \left\{ \alpha_{pi} \left[\sum_{j=1}^{n} \left\{ |a_{ij}| \left(1-\alpha_{pj}\right) \right\} + \sum_{k=1}^{m} \left\{ |b_{ik}| \left(1-\beta_{pk}\right) \right\} \right] \right\} \\
\text{subject to} \quad & \forall p \in [\![1, N]\!], \sum_{i=1}^{n} \alpha_{pi} \geq 1, \sum_{i=1}^{m} \beta_{pi} \geq 1 \\
& \forall i \in [\![1, n]\!], \sum_{p=1}^{N} \alpha_{pi} \leq 1, \sum_{p=1}^{N} \alpha_{pi} \geq 1 \\
& \forall j \in [\![1, m]\!], \sum_{p=1}^{N} \beta_{pj} \leq 1, \sum_{p=1}^{N} \beta_{pj} \geq 1
\end{aligned} \tag{4.18}$$

The optimization problem presented in (4.18) has an equivalent matrix formulation using the state space model matrices as well as the state and input grouping matrices. This representation will be used later in order to display the subsystem

to subsystem interactions in the form of a weighted directed graph.

$$\begin{aligned} \underset{\alpha,\beta}{\text{minimize}} \quad & \operatorname{tr}(\alpha|A|(1-\alpha)^\top + \alpha|B|(1-\beta)^\top) \\ \text{subject to} \quad & \forall p \in [\![1, N]\!], \sum_{i=1}^{n} \alpha_{pi} \geq 1, \sum_{i=1}^{m} \beta_{pi} \geq 1 \\ & \forall i \in [\![1, n]\!], \sum_{p=1}^{N} \alpha_{pi} \leq 1, \sum_{p=1}^{N} \alpha_{pi} \geq 1 \\ & \forall j \in [\![1, m]\!], \sum_{p=1}^{N} \beta_{pj} \leq 1, \sum_{p=1}^{N} \beta_{pj} \geq 1 \end{aligned} \tag{4.19}$$

The grouping matrices $(1-\alpha)$ and $(1-\beta)$ are the conjugate of α and β respectively, they are obtained by changing the ones into zeros and vice versa. The magnitude operator applied to a matrix yields a matrix where each component is the magnitude of the initial matrix. The pre-multiplication of $|A|$ by α adds the selected rows from each subsystem together into a single row, then the post-multiplication by the transpose of the conjugate of α adds the columns not belonging to the same subsystem together. Consequently, the square matrix obtained is of dimension N and contains the subsystem to subsystem interactions on its diagonal. More specifically, the diagonal element with row and column indexes equal to i represents the sum of the interactions coming from all the subsystems with the subsystem i. In a similar fashion, replacing the constraints of the optimization problems (4.18) and (4.19) by the overlapping partitioning constraints yields the equivalent overlapping problem formulation. As it was demonstrated previously within this section the partitioning problem can be expressed as an integer optimization problem. However the cost function representing the interaction metric is non-linear as well as non-convex, therefore the problem can be intractable. As it is presented within the next section a linear relaxation of the optimization problem (4.18) is made possible throughout the use of auxiliary variables.

4.4 Linear Relaxation of the Weak Interactions Problem

The weak interactions optimization problem formulated previously in (4.18) can be turned into an Integer Linear Programming (ILP), this is made possible due to the introduction of auxiliary variables. Replacing a product of binary variables by an auxiliary variable is a well known technique that requires the use of new linear constraints (Cavalier et al., 1990; Bemporad and Morari, 1999; Williams, 2013).

Two auxiliary binary variables are created along with their linear constraints, a state auxiliary variable γ used to take into account the interactions coming from the state matrix, and an input auxiliary variable δ used to account for the interactions coming from the input matrix.

4.4.1 State Auxiliary Variable

As it is presented in Table 4.1 and in equation (4.20), γ is linked to α throughout four constraints. Indeed, four inequalities are necessary because of the four possible outcomes for the binary product $\alpha_{pi}(1-\alpha_{pj})$ in equation (4.14). From top to bottom within Table 4.1, the four different cases are, first when no states belong to the group p then no interaction has to be accounted for. If the state i is not in the group p but the state j is, then no interaction is accounted for as this will be taken into account in the symmetrical case. If the state i belongs to the group p and the state j does not then an interaction is accounted for. The last possible case is when the two states i and j both belong to the group p, in this last scenario no interaction subsists as they are both in the same group.

Table 4.1: Auxiliary variable γ.

Primary			**Auxiliary**
α_{pi}	α_{pj}	$\alpha_{pi}\left(1-\alpha_{pj}\right)$	γ_{pij}
0	0	0	0
0	1	0	0
1	0	1	1
1	1	0	0

The four linear constraints for the state auxiliary variable γ are the following,

$$\forall(p,i,j) \in [\![1,N]\!] \times [\![1,n]\!] \times [\![1,n]\!],$$

$$\gamma_{pij} \leq \alpha_{pi} + \alpha_{pj} \tag{4.20a}$$

$$\gamma_{pij} \leq 1 + \alpha_{pi} - \alpha_{pj} \tag{4.20b}$$

$$\gamma_{pij} \geq \alpha_{pi} - \alpha_{pj} \tag{4.20c}$$

$$\gamma_{pij} \leq 2 - \alpha_{pi} - \alpha_{pj}. \tag{4.20d}$$

4.4.2 Input Auxiliary Variable

In a similar way, δ is the auxiliary binary variable taking into account the interactions coming from the input matrix. This time the constraints are formed using α as well as β because the input interactions are also state dependent. The same reasoning

applies to formulate the four inequalities arising from the binary product $\alpha_{pi}(1-\beta_{pk})$ in equation (4.16). All the cases are gathered within Table 4.2.

Table 4.2: Auxiliary variable δ.

Primary			**Auxiliary**
α_{pi}	β_{pk}	$\alpha_{pi}\left(1-\beta_{pk}\right)$	δ_{pik}
0	0	0	0
0	1	0	0
1	0	1	1
1	1	0	0

The four linear constraints associated to δ are represented in (4.21).

$$\forall (p,i,k) \in [\![1,N]\!] \times [\![1,n]\!] \times [\![1,m]\!],$$

$$\delta_{pik} \leq \alpha_{pi} + \beta_{pk} \tag{4.21a}$$

$$\delta_{pik} \leq 1 + \alpha_{pi} - \beta_{pk} \tag{4.21b}$$

$$\delta_{pik} \geq \alpha_{pi} - \beta_{pk} \tag{4.21c}$$

$$\delta_{pik} \leq 2 - \alpha_{pi} - \beta_{pk} \tag{4.21d}$$

In summary, the required behaviours induced by the state and input auxiliary variables are equivalent to the logical conditions, state i in group p and state j not in group p as well as state i in group p and input k not in group p, respectively for γ_{pij} and δ_{pik}. Both auxiliary variables have three indexes and can be represented by cubic arrays of binary variables, their respective sizes are $N \times n \times n$ for the state auxiliary variable γ and $N \times n \times m$ for the input auxiliary variable δ. In addition to the linear constraints presented in equation (4.20) and equation (4.21), both auxiliary variables have to be composed only of binaries. Similarly, the same auxiliary variables with the same constraints are used to linearize the cost function of the overlapping partitioning problem. Indeed, the overlapping case is only a generalization of the non-overlapping case that is achieved by a modification of the linear constraints affecting the primary variables. The next subsection presents the formulation of the linearized weak interactions partitioning problem based on the state and input auxiliary variables.

4.4.3 Weak Interactions Optimization Linear Problem

The optimization problem presented in (4.18) is reformulated into the linear integer optimization problem presented in (4.22), obtained by replacing the binary products

by the auxiliary variables along with their linear constraints. The new optimization problem obtained has a linear cost function and linear constraints, however, all the decision variables are binaries. Therefore, the linearized optimization problem (4.22) is a Binary Integer Linear Programming (BILP). The linear relaxation provided by the introduction of auxiliary variables is tight and equivalent to the initial partitioning optimization problem (Williams, 2013). A similar formulation can be achieved in the overlapping case by replacing the set of constraints (4.10) by the constraints (4.11).

$$
\begin{aligned}
\underset{\alpha,\beta,\gamma,\delta}{\text{minimize}} \quad & \sum_{p=1}^{N}\sum_{i=1}^{n}\left\{\sum_{j=1}^{n}\left\{\gamma_{pij}|a_{ij}|\right\}+\sum_{k=1}^{m}\left\{\delta_{pik}|b_{ik}|\right\}\right\} \\
\text{subject to} \quad & \forall p \in [\![1,N]\!],\ \sum_{i=1}^{n}\alpha_{pi}\geq 1,\ \sum_{i=1}^{m}\beta_{pi}\geq 1 \\
& \forall i \in [\![1,n]\!],\ \sum_{p=1}^{N}\alpha_{pi}\leq 1,\ \sum_{p=1}^{N}\alpha_{pi}\geq 1 \\
& \forall i \in [\![1,m]\!],\ \sum_{p=1}^{N}\beta_{pi}\leq 1,\ \sum_{p=1}^{N}\beta_{pi}\geq 1 \\
& \forall (p,i,j,k) \in [\![1,N]\!]\times[\![1,n]\!]\times[\![1,n]\!]\times[\![1,m]\!] \\
& \gamma_{pij}\leq \alpha_{pi}+\alpha_{pj} \\
& \gamma_{pij}\leq 1+\alpha_{pi}-\alpha_{pj} \\
& \gamma_{pij}\geq \alpha_{pi}-\alpha_{pj} \\
& \gamma_{pij}\leq 2-\alpha_{pi}-\alpha_{pj} \\
& \delta_{pik}\leq \alpha_{pi}+\beta_{pk} \\
& \delta_{pik}\leq 1+\alpha_{pi}-\beta_{pk} \\
& \delta_{pik}\geq \alpha_{pi}-\beta_{pk} \\
& \delta_{pik}\leq 2-\alpha_{pi}-\beta_{pk}
\end{aligned}
\tag{4.22}
$$

As it was presented within this section, the use of two auxiliary variables enables the linearization of the optimization problem. The minimization problem presented in (4.22) because of the use of primary and auxiliary binary variables allows to trade non-linearities for an increase in decision variable dimension. Nonetheless, the complexity of the optimization problem can be reduced by exploiting the structure of the plant model and by creating only the auxiliary variables where the state space model elements are not equal to zero. For instance, if $a_{ij} = 0$ there is no need to create $(\gamma_{pij})_{p\in[\![1,N]\!]}$, similarly if $b_{ik} = 0$ with $(\delta_{pik})_{p\in[\![1,N]\!]}$. The next section presents the notion of partitioning cut reducing the search space in order to obtain only

controllable subsystems.

4.5 Partitioning Cuts

Running the previous optimization problem will yield N subsystems presenting the least amount of interactions, unfortunately no information is given concerning their controllability. The state space model of any subsystem p, represented without the couplings coming from the other subsystems can be rewritten from equation (4.2) as follows,

$$\forall p \in [\![1, N]\!],\ \dot{x}_p = A_{pp}x_p + B_{pp}u_p. \tag{4.23}$$

The controllability of any given subsystem model p yielded by the optimization problem can be checked by verifying that the controllability matrix C_p defined in (4.24) has full row rank.

$$\forall p \in [\![1, N]\!],\ C_p = \left[B_{pp} \,\middle|\, A_{pp}B_{pp} \,\middle|\, \dots \,\middle|\, A_{pp}^{n_p-1}B_{pp}\right] \tag{4.24}$$

Therefore, at the end of the optimization process, a controllability test is performed for each subsystem model, testing that the set of equalities given in (4.25) holds.

$$\forall p \in [\![1, N]\!],\ \text{rank}\left(C_p\right) = n_p \tag{4.25}$$

As one can see the controllability matrices C_p as well as the integers n_p representing the number of state variables in subsystem p are obtained as a result of solving the optimization problem and are not known a priori. Therefore, implementing constraints within the linear integer optimization problem in order to restrain the solutions to the set of controllable subsystems is a tremendously difficult task. However, applying controllability cuts to the search space recursively and a posteriori is possible. Every time a non-controllable partitioning is achieved new linear constraints are added to the existing ones in order to reduce the search space by cutting the non-controllable partitionings out with an affine hyperplane cut. The principle of cutting solutions out of the search space is similar to the Gomory cuts (Gomory, 1958) where cuts are used to discard solutions that are not integer. Controllability cuts are applied from the root node and are valid for the entire search tree, hence cuts lifting methods are not necessary in this case (Balas et al., 1996).

4.5.1 Non-overlapping Partitioning Cuts

The non-overlapping grouping matrices can be seen as a concatenation of basis vectors, such that

$$\alpha = \left[e_{i_1} \,\middle|\, e_{i_2} \,\middle|\, \ldots \,\middle|\, e_{i_k} \,\middle|\, \ldots \,\middle|\, e_{i_n}\right]_{i_k \in [\![1,N]\!]} \tag{4.26a}$$

$$\beta = \left[e_{i_1} \,\middle|\, e_{i_2} \,\middle|\, \ldots \,\middle|\, e_{i_k} \,\middle|\, \ldots \,\middle|\, e_{i_m}\right]_{i_k \in [\![1,N]\!]}, \tag{4.26b}$$

with $(e_{i_k})_{i_k \in [\![1,N]\!]}$ the canonical orthonormal basis of $\mathbb{R}^N$. Subsequently, the square of their l_2-norm is equal to n and m respectively and is calculated in equation (4.27) and equation (4.28).

$$\|\alpha\|_2^2 = \mathrm{tr}\left(\alpha^\top \alpha\right) = \sum_{k=1}^{n} e_{i_k}^\top . e_{i_k} = \sum_{k=1}^{n} \delta_{i_k i_k} = n \tag{4.27}$$

$$\|\beta\|_2^2 = \mathrm{tr}\left(\beta^\top \beta\right) = \sum_{k=1}^{m} e_{i_k}^\top . e_{i_k} = \sum_{k=1}^{m} \delta_{i_k i_k} = m \tag{4.28}$$

In the more general case of overlapping grouping matrices having q state overlaps, only the l_2-norm of α is changed as follows,

$$\|\alpha\|_2^2 = \mathrm{tr}\left(\alpha^\top \alpha\right) = \sum_{i=1}^{N}\sum_{j=1}^{n} \alpha_{ij}^2 = n + q. \tag{4.29}$$

The set of non-overlapping grouping matrices of size $N \times n$ respecting the constraints (4.10a), (4.10c) and (4.10e) will be referred to as $\mathbb{G}_{Nn}$ with $\mathbb{G}_{Nn} \subset [\![0,1]\!]^{N \times n}$. Whereas the set of overlapping grouping matrices of size $N \times n$ with q overlaps, respecting the constraints (4.11a), (4.11c) and (4.11e) will be referred to as $\mathbb{G}_{Nn}^q$ with $\mathbb{G}_{Nn}^q \subset [\![0,1]\!]^{N \times n}$. The optimal non-controllable partitionings can be cut out of the search space by applying the following cut after at least one non-controllable subsystem is obtained,

$$\begin{aligned}
&\forall\, (\alpha,\beta) \in \mathbb{G}_{Nn} \times \mathbb{G}_{Nm},\ (\alpha,\beta) \neq \left(\alpha^{nc^\star}, \beta^{nc^\star}\right) \\
&\Leftrightarrow \mathrm{tr}\left(\alpha^\top \alpha^{nc^\star}\right) + \mathrm{tr}\left(\beta^\top \beta^{nc^\star}\right) \leq n + m - 1 && \text{(4.30a)} \\
&\Leftrightarrow \sum_{p=1}^{N}\left\{\sum_{i=1}^{n}\left\{\alpha_{pi}\alpha_{pi}^{nc^\star}\right\} + \sum_{k=1}^{m}\left\{\beta_{pk}\beta_{pk}^{nc^\star}\right\}\right\} \leq n + m - 1. && \text{(4.30b)}
\end{aligned}$$

As proved by Lemma 4.1 as well as Lemma 4.2, cuts can be added to re-

move the non-overlapping or overlapping partitionings that include at least one non-controllable subsystem.

Lemma 4.1. *A non-controllable non-overlapping partitioning defined by a pair of grouping matrices* $(\alpha^{nc^\star}, \beta^{nc^\star})$ *is removed from the set of feasible solutions by applying the controllability cut* (4.30).

Proof. For a given non-controllable optimal non-overlapping partitioning denoted by $\alpha^{nc^\star}$ and $\beta^{nc^\star}$, and for any couple of non-overlapping grouping matrices α and β, the inequalities (4.31) hold for the state grouping matrices.

$$\forall \left(\alpha^{nc^\star}, \alpha\right) \in \mathbb{G}_{Nn}^2,\ \mathrm{tr}\left(\alpha^\top \alpha^{nc^\star}\right) = \sum_{k=1}^{n} e_{i_k}^\top . e_{i_k}^{nc^\star} = \sum_{k=1}^{n} \delta_{i_k i_k^{nc^\star}} \leq n \tag{4.31}$$

Similarly, for the input grouping matrices the inequality (4.32) holds.

$$\forall \left(\beta^{nc^\star}, \beta\right) \in \mathbb{G}_{Nm}^2,\ \mathrm{tr}\left(\beta^\top \beta^{nc^\star}\right) = \sum_{k=1}^{m} e_{i_k}^\top . e_{i_k}^{nc^\star} = \sum_{k=1}^{m} \delta_{i_k i_k^{nc^\star}} \leq m \tag{4.32}$$

Therefore, because of the sum of Kronecker operators, the upper bounds are only reached in equation (4.31) when $\alpha = \alpha^{nc^\star}$ and in equation (4.32) when $\beta = \beta^{nc^\star}$. Consequently, there exists a natural way of constructing hyperplane cuts when a non-controllable optimal partitioning $(\alpha^{nc^\star}, \beta^{nc^\star})$ is obtained. The hyperplane cut is as presented in equation (4.30) which completes the proof. □

4.5.2 Overlapping Partitioning Cuts

In a similar way, a non-controllable optimal overlapping partitioning $(\alpha^{nc^\star}, \beta^{nc^\star})$ can be cut out of the search space by adding a similar controllability cut, as introduced in equation (4.33), and proved by Lemma 4.2.

$$\begin{aligned}
&\forall (\alpha, \beta) \in \mathbb{G}_{Nn}^q \times \mathbb{G}_{Nm},\ (\alpha, \beta) \neq \left(\alpha^{nc^\star}, \beta^{nc^\star}\right) \\
&\Leftrightarrow \mathrm{tr}\left(\alpha^\top \alpha^{nc^\star}\right) + \mathrm{tr}\left(\beta^\top \beta^{nc^\star}\right) \leq n + m + q - 1 && \text{(4.33a)} \\
&\Leftrightarrow \sum_{p=1}^{N} \left\{ \sum_{i=1}^{n} \left\{ \alpha_{pi} \alpha_{pi}^{nc^\star} \right\} + \sum_{k=1}^{m} \left\{ \beta_{pk} \beta_{pk}^{nc^\star} \right\} \right\} \leq n + m + q - 1. && \text{(4.33b)}
\end{aligned}$$

Lemma 4.2. *A non-controllable overlapping partitioning defined by a pair of grouping matrices* $(\alpha^{nc^\star}, \beta^{nc^\star})$ *is removed form the set of feasible solutions by applying the*

controllability cut (4.33).

Proof. For a given non-controllable optimal overlapping partitioning denoted by $\alpha^{nc^\star}$ and $\beta^{nc^\star}$, and for any couple of overlapping grouping matrices α and β, the inequalities (4.34) hold for the state grouping matrices.

$$\begin{aligned} &\forall \left(\alpha^{nc^\star}, \alpha\right) \in \mathbb{G}_{Nn}^{q} \times \mathbb{G}_{Nn}^{q}, \\ &\operatorname{tr}\left(\alpha^\top \alpha^{nc^\star}\right) = \sum_{i=1}^{N}\sum_{j=1}^{n} \alpha_{ij}\alpha_{ij}^{nc^\star} \leq \sqrt{\|\alpha\|_2^2 \times \|\alpha^{nc^\star}\|_2^2} = n + q \end{aligned} \tag{4.34}$$

The inner product of any overlapping state grouping matrix α with the non-controllable optimal grouping matrix $\alpha^{nc^\star}$ has an upper bound according to the Cauchy-Schwartz inequality, given by equation (4.34). Moreover, this upper bound is achieved only when the two state overlapping grouping matrices are equal. Also, the previous inequality provided in equation (4.32) still holds in the overlapping case. Therefore, the upper bounds are only reached in equation (4.33) when $\alpha = \alpha^{nc^\star}$ and when $\beta = \beta^{nc^\star}$. Consequently, there exist a hyperplane cut, ruling out a non-controllable optimal overlapping partitioning $(\alpha^{nc^\star}, \beta^{nc^\star})$ from the search space. This cut corresponds to the inequality presented equation (4.33), which concludes the proof. □

Every time a non-controllable partitioning is obtained a controllability cut (4.30) or (4.33) is added to the linear constraint set before the optimization is performed again, respectively in the non-overlapping and overlapping cases. Therefore, the previous non-controllable optimal partitioning can no longer be reached as it is now excluded from the search space. However, because the groups are not ordered a similar partitioning can be achieved again simply by swapping simultaneously the rows of α and β, leading to another representation of the same system partition. Indeed, without any order constraints on the groups, $N!$ identical representations of a single partition are possible. Different techniques can be employed to make the optimization more efficient. First of all, it would be possible to constrain the grouping matrices in order to rank the different groups as it has been done with move blocking matrices (Cagienard et al., 2007). In this case only one representation per partitioning would be possible. The second solution would be to perform $N!$ controllability cuts every time a non-controllable optimal solution is obtained. Therefore, all the possible non-controllable representations of a given partitioning would be removed from the search space simultaneously. It is the latter solution that has been implemented in the weak interactions partitioning algorithm. Every time an optimal non-controllable solution is encountered $N!$ linear constraints corresponding to a controllability cut are added to the set of existing constraints. More

details on the complexity of the partitioning algorithm are provided in a later section. The optimization is then ran iteratively until the least interacting controllable partitioning is obtained. It can be noted that the duality between the controllability and the observability of a system can be used in order to partition a system model composed of a state matrix A as well as an output matrix C into observable subsystem models. The next section presents how the partitioning algorithm is built using the linear relaxation as well as the controllability cut technique.

4.6 Weak Interactions Partitioning Algorithm

This section presents the main weak interaction partitioning algorithm along with one of the auxiliary algorithm. Since the partitioning algorithm has to compute the controllability matrices for all the subsystems, one of the steps of the partitioning algorithm is to extract the subsystem state space models based on the value of the grouping matrices obtained. Subsequently, a second algorithm is implemented to extract the subsystem models relying on a technique using masking matrices and presented within the next subsection. This auxiliary algorithm uses the grouping matrices in order to compute the two sets of N state and N input masking matrices.

4.6.1 Extraction of the Subsystem Models

After each iteration of the partitioning algorithm a controllability check is performed on the subsystem state space models obtained. Therefore, one of the step for the weak interaction partitioning algorithm is to extract the subsystem models form the main system state space model. This step is achieved throughout the use of a set of state masking matrices $T_{\alpha,p}$ as well as a set of input masking matrices $T_{\beta,p}$, where p represents a subsystem index such that $p \in [\![1, N]\!]$. The main system state space model matrices are pre-multiplied and post-multiplied by the masking matrices in order to keep only the appropriate rows and columns for a given subsystem. The subsystem state space model of any subsystem indexed by $p \in [\![1, N]\!]$ is extracted as per equation (4.35).

$$\forall p \in [\![1, N]\!],$$

$$A_{pp} = T_{\alpha,p}^{\top} A T_{\alpha,p} \tag{4.35a}$$

$$B_{pp} = T_{\alpha,p}^{\top} B T_{\beta,p} \tag{4.35b}$$

The pre-multiplication of a state space matrix by the transpose of a masking matrix will mask the appropriate rows and the post-multiplication by a masking

matrix will mask the appropriate columns. Consequently, after pre-multiplications and post-multiplications by the state and input masking matrices the subsystem state space model are extracted from the main system model. The columns of the state and input masking matrices are composed of the unit vectors of the canonical basis of $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively for the state and input masking matrices. Subsequently, the dimensions of the masking matrices are only linked to the dimensions of the grouping matrices as well as the value of their entries. For a given subsystem of index p, the state masking matrix $T_{\alpha,p}$ is an element of the set $[\![0,1]\!]^{n\times n_p}$, similarly the input masking matrix belongs to the set $[\![0,1]\!]^{m\times m_p}$. Finally, for any indexes $(i,j)\in[\![1,N]\!]^2$ the block matrices representing the interactions between two distinct non-overlapping subsystems in the state space model (4.2) can be extracted with the masking matrices as per equation (4.36).

$$\begin{aligned}&\forall(i,j)\in[\![1,N]\!]^2,\\ &A_{ij}=T_{\alpha,i}^{\top}AT_{\alpha,j} &\text{(4.36a)}\\ &B_{ij}=T_{\alpha,i}^{\top}BT_{\beta,j}. &\text{(4.36b)}\end{aligned}$$

Note that the extraction of interaction models in the case of state overlapping subsystems is different for the state matrices and therefore relies on another set of state masking matrices. The Algorithm 4.1 presented within this subsection, explains how to compute the sets of state and input masking matrices.

Each state masking matrix is created by concatenating the unit vectors composing the canonical basis of $\mathbb{R}^n$ having a specific index. At the start, the algorithm initializes each masking matrix with a zero matrix of appropriate dimension. The masking matrices are built by concatenating vectors as follows, for a given subsystem index $p\in[\![1,N]\!]$, if a one is located at the intersection of the p-th row and the i-th column of the state grouping matrix α then the canonical unit vector $e_i\in\mathbb{R}^n$ is concatenated to the right side of $T_{\alpha,p}$, based on the value of the column index Col, that is then incremented. Consequently, the state masking matrix indexed by p is built by concatenating n_p vectors horizontally from the right. The same process is applied to build the set of input masking matrices. However, in this case the canonical unit vectors e_j are taken from the canonical basis of $\mathbb{R}^m$ and the equality condition is applied to the entries of the input grouping matrix β.

4.6.2 Partitioning Algorithm

The algorithm implemented to perform the weak interactions system partitioning takes the main state space model (A,B) as well as the subsystem number N as

Algorithm 4.1: Masking matrices computation algorithm.

Input : α, β
Output : $\forall p \in [\![1, N]\!]$, $\{T_{\alpha,p}, T_{\beta,p}\}$
Initialization: $\forall p \in [\![1, N]\!]$, $T_{\alpha,p} = 0_{n \times n_p}$, $T_{\beta,p} = 0_{m \times m_p}$

```
for p = 1 to N do
    Col = 1
    for i = 1 to n do
        if α_pi = 1 then
            T_α,p(:, Col) = e_i
            Col = Col + 1
        end
    end
end
for p = 1 to N do
    Col = 1
    for j = 1 to m do
        if β_pj = 1 then
            T_β,p(:, Col) = e_j
            Col = Col + 1
        end
    end
end
```

inputs. It returns the state and input grouping matrices α and β once one of the least interacting controllable partitioning is reached. The algorithm can be described step by step as it is implemented. The first step is to build the initial linear constraints that will be used for the primary and auxiliary variables. Then the optimization is performed yielding the first pair of state and input grouping matrices α and β. The subsystem models are extracted from the main system model based on the masking matrices computed by Algorithm 4.1, therefore, the subsystem controllability matrices can be computed and their rank can be evaluated. The last step of the algorithm is to add the appropriate set of controllability cuts if the current optimal partitioning presents at least one uncontrollable subsystem as well as to return to the previous step to run the optimization again with the new set of linear constraints. Otherwise, the algorithm terminates and returns the least interacting controllable partitioning as a final result when the optimal partitioning obtained has all its subsystem models controllable. The weak interaction partitioning problem is a $0-1$ integer linear programming problem also known as BILP. The weak interactions partitioning algorithm is presented below in Algorithm 4.2.

On the very first loop iteration, no solution exists, therefore, the optimization is performed and the first grouping matrices are obtained. The next step is to check

Algorithm 4.2: Weak interactions partitioning algorithm.

Input : A, B, N
Output: α, β
while $\exists p \in [\![1, N]\!]$, $\text{rank}(C_p) \neq n_p$ **do**
  if $\exists(\alpha^{nc^\star}, \beta^{nc^\star})$ **then**
    Add controllability cuts (4.33)
    Run the optimization problem (4.22) subject to cuts (4.33)
    Extract the subsystem state space models using Algorithm 4.1 and compute: $\forall p \in [\![1, N]\!]$, C_p
  else
    Run the optimization problem (4.22)
    Extract the subsystem state space models using Algorithm 4.1 and compute: $\forall p \in [\![1, N]\!]$, C_p
  end
end
return α, β

that every subsystem is controllable by verifying that equation (4.25) holds. If at least one of the subsystems obtained is not controllable then $N!$ controllability cuts (4.33) are added to the set of linear constraints and the optimization can start again using the reduced search space. It can be noticed that the non-overlapping case is subsumed by the overlapping case when the overlapping parameter q is set to zero. The algorithm finishes when the least interacting controllable partitioning comprising N subsystems is found, or when no controllable partitioning can be established. In the latter, the optimization problem becomes infeasible as the search space reduces to the empty set. Evaluating the controllability of the subsystems obtained is performed by using the auxiliary Algorithm 4.1 to extract the subsystem models, so that the controllability matrices can be computed. A variant of the weak interaction partitioning algorithm can be implemented if the stopping criteria is modified so that the algorithm does not terminate when the least interacting controllable subsystems are obtained but when all the possible controllable subsystems having the same interaction metric are found. Indeed, some examples can be computed where different partitionings have the same interaction cost and are all composed of controllable subsystems. Such a modification of the algorithm would allow to use the extra degree of freedom provided and compare all the partitionings obtained and chose the more suitable, according to other design criteria. The next subsection presents the combinatorial complexity of the weak interactions partitioning algorithm as well as the search space dimension.

4.7 Complexity of the Partitioning Problem

The weak interactions partitioning algorithm has the complexity of a binary integer linear program, it is a well known fact that such combinatorial problems have Non-deterministic Polynomial (NP) complexity (Karp, 1972). This section firstly explains the size of the discrete feasible search space by enumerating the decision variables and the constraints and secondly discusses the complexity of the binary integer linear programming problem.

4.7.1 Search Space Dimension

Partitioning a state space model into overlapping or non-overlapping weakly coupled subsystems consists of solving a combinatorial optimization problem. For a model composed of n states and m inputs, the set of feasible, non-empty, non-overlapping or overlapping partitions with N subsystems has to include a certain number of decision variables and to comply with the set of linear constraints presented previously.

Number of Decision Variables

Performing the binary linear optimization requires primary and auxiliary variables, the primary variables α and β are composed respectively of $N \times n$ state grouping binaries and $N \times m$ input grouping binaries. Finally, in order to linearize the optimization problem, two auxiliary variables γ and δ are added to the optimization, their respective dimensions are $N \times n \times n$ and $N \times n \times m$. Consequently, the integer linear problem (4.22) has a total number of decision variables d such that,

$$d = Nn + Nm + Nn^2 + Nnm = N(n+m)(n+1). \tag{4.37}$$

Number of Constraints

In order for the optimization problem to comply with the overlapping or non-overlapping problem requirements, some constraints have to be applied to the primary and auxiliary decision variables. As it can be seen from the definition of the problem (4.22), the number of linear constraints initially applied to the primary variables is $2(N+n+m)$. The auxiliary variables are subject to $4(Nn^2 + Nnm)$ constraints in the initial optimization problem. Therefore, the total initial number of constraints c is as follows,

$$c = 2(N+n+m) + 4Nn^2 + 4Nnm = 2(N+n+m) + Nn(n+m). \tag{4.38}$$

The total number of linear constraints is increased by $N!$ each time a controllability cut is added to the search space.

Non-overlapping Partitioning

The set of all feasible non-overlapping possible partitionings is a discrete set that grows exponentially based on the dimensions of the system model as well as the number of subsystems. The problem of partitioning the state and the input variables into a fixed number of non-overlapping groups is equivalent to the number of ways to partition a set of n states into N non-empty non-overlapping subsets. This combinatorial number is defined by the Stirling number of the second kind as well as the factorial operator as follows,

$$\operatorname{card} \mathbb{G}_{Nn} = N! \left\{ {n \atop N} \right\}. \tag{4.39}$$

Similarly, the number of ways to partition the input variables into N subsets is computed by replacing n by m in equation (4.39). Therefore, the cardinality of the set of non-overlapping partitions for a system composed of n state variables and m input variables is defined by the cardinality of the Cartesian product of the two discrete sets $\mathbb{G}_{Nn}$ and $\mathbb{G}_{Nm}$ such that,

$$\operatorname{card} \{\mathbb{G}_{Nn} \times \mathbb{G}_{Nm}\} = (N!)^2 \left\{ {n \atop N} \right\} \left\{ {m \atop N} \right\}. \tag{4.40}$$

The cardinality of the product of the discrete sets is the product of the cardinality of the two initial sets. The expression of the cardinality expressed in equation (4.39) correspond to the number of distinct system partition representations, nonetheless, the same partitioning can be represented $N!$ times by swapping simultaneously the rows of α and β. Consequently, the total number of distinct system partitionings is given by the following number,

$$N! \left\{ {n \atop N} \right\} \left\{ {m \atop N} \right\}. \tag{4.41}$$

Overlapping Partitioning

The set of all feasible overlapping partitionings is computed in two steps. The first step is similar to the set of non-overlapping partitionings, the extra step can be seen as performing q state overlaps. Thus, it is similar to choosing q state variables amongst $n(N-1)$ remaining states, in order to be used again in the state grouping

matrix. Therefore, the cardinality of the set $\mathbb{G}_{Nn}^{q}$ is defined by,

$$\operatorname{card} \mathbb{G}_{Nn}^{q} = N! \left\{ {n \atop N} \right\} \binom{n(N-1)}{q}. \tag{4.42}$$

Subsequently, the cardinality of the set of overlapping partitions for a system composed of n state variables and m input variables with a number of state overlaps set to q is defined by the cardinality of the Cartesian product of the two discrete sets $\mathbb{G}_{Nn}^{q}$ and $\mathbb{G}_{Nm}$ given in equation (4.43).

$$\operatorname{card} \left\{ \mathbb{G}_{Nn}^{q} \times \mathbb{G}_{Nm} \right\} = (N!)^2 \binom{n(N-1)}{q} \left\{ {n \atop N} \right\} \left\{ {m \atop N} \right\}. \tag{4.43}$$

Similarly as before, the cardinality number presented equation (4.43) is linked to the number of representations for an overlapping partitioning. However, since each partitioning has $N!$ distinct representations, the number of distinct state overlapping system partitionings is defined as follows,

$$N! \binom{n(N-1)}{q} \left\{ {n \atop N} \right\} \left\{ {m \atop N} \right\}. \tag{4.44}$$

It is possible to see that setting q to zero, correspond to the non-overlapping case since the overlapping case subsumes it.

4.7.2 Algorithm Complexity

The complexity of a binary integer linear programming is in general NP-complete and only some special cases have a deterministic polynomial time algorithm complexity (Papadimitriou, 1981; Papadimitriou and Steiglitz, 1998). In the worst case scenario solving a $0-1$ integer linear program requires to perform an exhaustive search with exponential complexity $\mathcal{O}(2^{d}\,\mathrm{poly}(d,c))$, where $\mathrm{poly}(\cdot)$ denotes a polynomial function, d represents the number of binary decision variables and c is the number of linear constraints respectively as per equation (4.37) and equation (4.38). As it has been mentioned previously, knowledge of the structure of the system matrices A and B can be exploited to reduce the number of auxiliary variables, hence the number of constraints. Nonetheless, it is in general faster than the worst case and parallel computing can be used in order to speed up the $0-1$ integer linear program. Also, since the algorithm is used offline and before performing the control design, the time sensitivity of the partitioning algorithm is relatively low.

4.8 Graph Representations

Graphs are mathematical objects used to represent the topology and structure between discrete finite sets of elements. Therefore, they are useful to represent the dynamic relations within a system or between multiple subsystems. Thus, the weak interaction partitioning problem can be compared to a graph partitioning problem. This section presents the relations between system and subsystem dynamical models and mathematical graphs, and highlights the similarity between the system and graph partitioning problems.

4.8.1 System Graph Representation

Linear systems can be represented by weighted directed graphs (Šiljak, 1991; Lunze, 1992), and graph theory has been used as a system partitioning technique previously in the literature (Ocampo-Martinez et al., 2011). The graph representation of a system is based on the system state space model in order to define the adjacency matrix of the system directed weighted graph. The adjacency matrix is built from the system state space model as follows,

$$\mathcal{A} = \begin{bmatrix} A^\top & 0 \\ B^\top & 0 \end{bmatrix}. \tag{4.45}$$

The adjacency matrix $\mathcal{A}$ is square of dimension $n+m$ and represents the oriented dynamical connection between the state and input variables of a system. Also, weights are associated to each of the edges within the directed graph to account for the dynamical interactions between state and input variables. The system (4.46) is mapped into the directed weighted graph represented Figure 4.1.

$$\dot{x} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} u \tag{4.46}$$

The adjacency matrix $\mathcal{A}$ corresponding to the system model (4.46) is therefore given as per equation (4.47).

$$\mathcal{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.47}$$

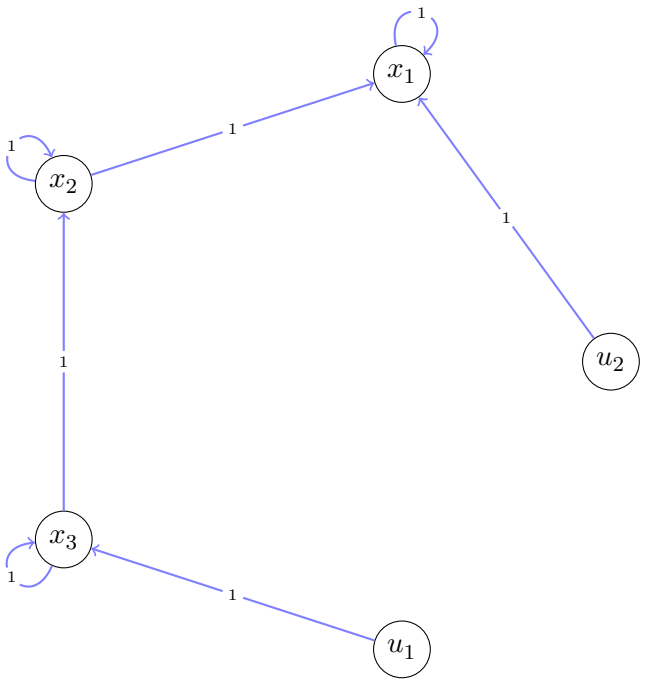

Figure 4.1: Weighted digraph representation of the system model (4.46).

4.8.2 Subsystem Graph Representation

Similarly, the dynamical interactions between different subsystems can be represented using weighted directed graphs. Therefore, the problem of computing the least interacting and controllable subsystem partitionings for a given system model is equivalent to a graph partitioning problem. The system (4.46) can be partitioned into two non-overlapping controllable subsystems with the minimum amount of interactions as presented Figure 4.2. The first subsystem (red subsystem) includes the state x_2 and x_3 as well as the input u_1 and the second subsystem (black subsystem) includes the rest of the system variables.

More precisely, the weak interactions system partitioning problem is equivalent to the minimum k-cut algorithm for graphs, where one graph is cut into k subgraphs such that the sum of the positive weights of the edges shared between all the subgraphs is minimized (Burlet and Goldschmidt, 1997). The distinction here resides in the fact that each subsystem has to remain controllable after the partitioning is performed. A subsystem representation of the system (4.46) is given Figure 4.3, where each node represents a given subsystem and the edges directions and weights respectively denote the interaction magnitudes and directions. The partitioning of the system (4.46) can be represented by the state and input grouping matrices given

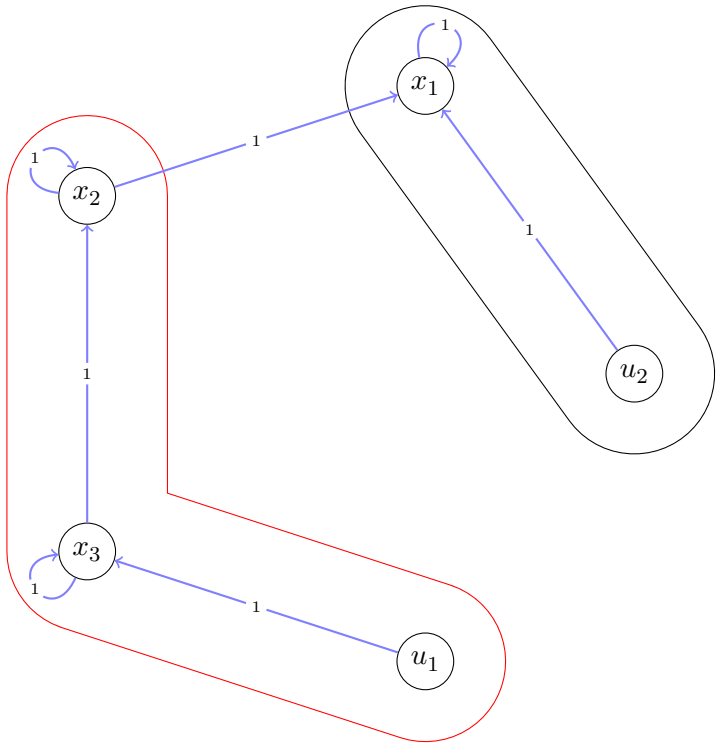

Figure 4.2: Weighted digraph representation of the controllable partitions of the system model (4.46) in two non-overlapping subsystems.

in equation (4.48).

$$\alpha = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \beta = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.48}$$

The representation of the subsystem graph for the system model (4.46), is linked to the grouping matrices presented previously through the definition of the adjacency matrix $\mathcal{A}$ defined as follows,

$$\mathcal{A}^\top = \alpha|A|\alpha^\top + \alpha|B|\beta^\top. \tag{4.49}$$

Defining the adjacency matrix of a system partitioning as per equation (4.49), provides a graph with N vertices and a maximum of N^2 edges including possible loops. Each vertex represents a subsystem and their associated weight denotes the interaction strength between two given subsystems. The interactions amongst the state and input variables of a subsystem is defined by a diagonal element and is represented by a loop in the subsystem graph. The subsystem graph representation of the system (4.46) is provided Figure 4.3, according to the adjacency matrix $\mathcal{A}$ computed in equation (4.50).

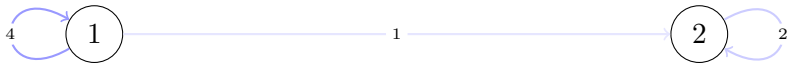

Figure 4.3: Subsystem graph representation of the subsystem (4.46).

The adjacency matrix $\mathcal{A}$ defined for the system model example is as follows,

$$\mathcal{A}^\top = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}^\top + \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^\top = \begin{bmatrix} 4 & 0 \\ 1 & 2 \end{bmatrix} \tag{4.50}$$

4.9 Numerical Examples

This section presents some numerical examples that were used to test the weak interactions partitioning algorithm. The first example presented was used only to test the binary integer linear optimization and does not require any controllability cuts. The second example was used specifically to demonstrate the controllability checks as well as the controllability cut technique. Finally, the third example highlights the state overlapping partitioning capability, as a generalization of the

weak interactions partitioning problem. These three examples have been solved in Matlab using CPLEX as the optimization solver (IBM corp., 2016). The algorithm used by CPLEX in order to solve BILP is a branch and cut technique, where the initial problem is relaxed into a Linear Programming (LP) problem and then cuts and branches are created until the solution of the relaxed LP is complies with the binary requirement.

4.9.1 Example without Controllability Cuts

The first example tested is the state space model of a military engine, the Pratt and Whitney F100 taken from (Jaw and Mattingly, 2009). The algorithm was used with the parameters given in (4.51). The reader can see that the whole system is already controllable even before performing any kind of partitioning.

$$A = \begin{bmatrix} -0.3245 \times 10^{1} & -0.2158 \times 10^{1} & -0.9155 \times 10^{3} & 0.5731 \times 10^{0} & 0.1342 \times 10^{3} \\ 0.1642 \times 10^{1} & -0.5941 \times 10^{1} & -0.2816 \times 10^{3} & 0.1897 \times 10^{0} & 0.5705 \times 10^{2} \\ 0.1685 \times 10^{-1} & -0.2554 \times 10^{-1} & -0.1003 \times 10^{2} & 0.7994 \times 10^{-2} & 0.5807 \times 10^{0} \\ 0 & 0 & 0 & -0.1 \times 10^{2} & 0 \\ -0.2163 \times 10^{1} & 0.6862 \times 10^{1} & 0.7405 \times 10^{3} & 0.1195 \times 10^{1} & -0.1715 \times 10^{3} \end{bmatrix} \tag{4.51a}$$

$$B = \begin{bmatrix} 0.1432 \times 10^{-1} & -0.3553 \times 10^{3} & -0.9906 \times 10^{2} & -0.1549 \times 10^{2} & 0.222 \times 10^{5} \\ 0.2871 \times 10^{0} & 0.7286 \times 10^{3} & 0.2514 \times 10^{2} & -0.6487 \times 10^{2} & 0.8122 \times 10^{4} \\ -0.2469 \times 10^{-2} & -0.103 \times 10^{3} & 0.6333 \times 10^{0} & -0.3213 \times 10^{0} & -0.7418 \times 10^{2} \\ 0.1 \times 10^{2} & 0 & 0 & 0 & 0 \\ -0.1311 \times 10^{0} & 0.3295 \times 10^{3} & -0.25 \times 10^{2} & 0.6257 \times 10^{2} & -0.6445 \times 10^{5} \end{bmatrix} \tag{4.51b}$$

$$N = 2 \tag{4.51c}$$

The partitioning obtained can be guessed due to the presence of zeros but also because of the presence of large elements in the matrices. The two grouping matrices resulting from the optimization are given in equation (4.52). It is important to notice that this first example does not need any controllability cut.

$$\alpha = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{4.52a}$$

$$\beta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.52b}$$

This example has been run on a standard desktop computer with an execution time of $0.75s$. The partitioning algorithm encounters multiple integer incumbents during the branch and cut search, before terminating at the minimum value. The value of the best incumbents stored during the branch and cut search are presented in Table 4.3 and plotted in Figure 4.4. The global minimum concerning the subsystem interactions is obtained during the very first iteration of the algorithm, thus the two subsystems are controllable and no controllability cuts are required.

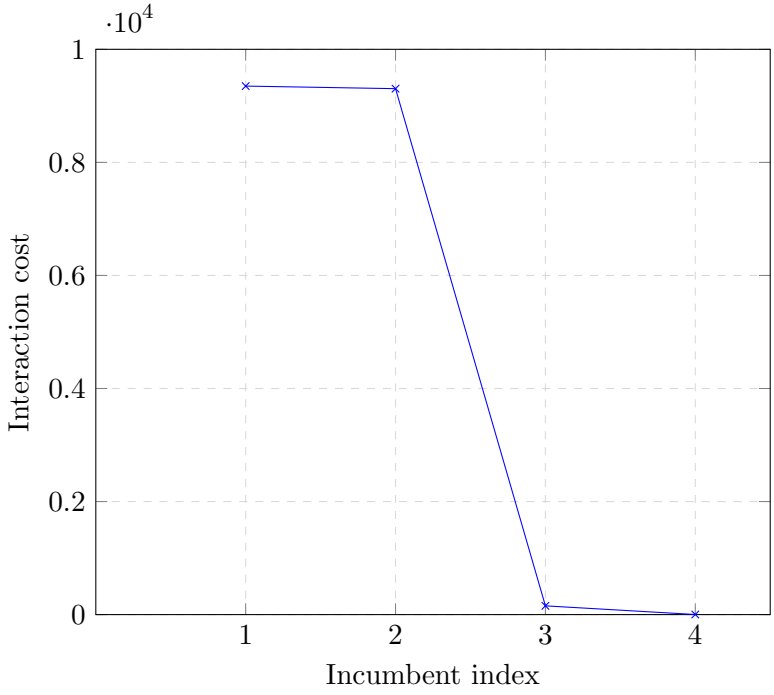

Figure 4.4: Interaction costs of the sequence of best integer incumbents for the partitioning of the system (4.51).

Table 4.3: Best integer incumbent costs for the partitioning of the system (4.51).

Incumbent index	**Interaction cost**
1	9349.62
2	9303.93
3	155.21
4	2.40

The total number of ways to partition the system (4.51) into two controllable subsystems without any overlaps is given as follows,

$$2!\left\{ {5 \atop 2} \right\}^2 = 450. \tag{4.53}$$

4.9.2 Example Involving Controllability Cuts

The second example is a five order system defined such that,

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{4.54a}$$

$$B = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.54b}$$

$$N = 3 \tag{4.54c}$$

The first 20 iterations of the algorithm result in non-controllable partitionings. After performing 120 controllability cuts, being $20 \times 3!$, one of the least interacting controllable partitioning is obtained as presented equation (4.55).

$$\alpha = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.55a}$$

$$\beta = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.55b}$$

For three groups each controllability cut has to be performed 6 times in order to take into account all the possible permutations of the grouping matrices. On a standard desktop computer the total run time was $15.5s$.

The Figure 4.5 presents the cost of the best integer incumbents encountered during the successive optimizations. Each new color represents a new partitioning optimization after a controllability cut has been performed. It can be noticed that the original system can be partitioned into three completely decoupled subsystems, however, these subsystems are not controllable. Naturally, applying cuts to the search space tends to increase the minimum interaction metric. The weighted digraph representation of the subsystems interactions is given Figure 4.6, it is possible

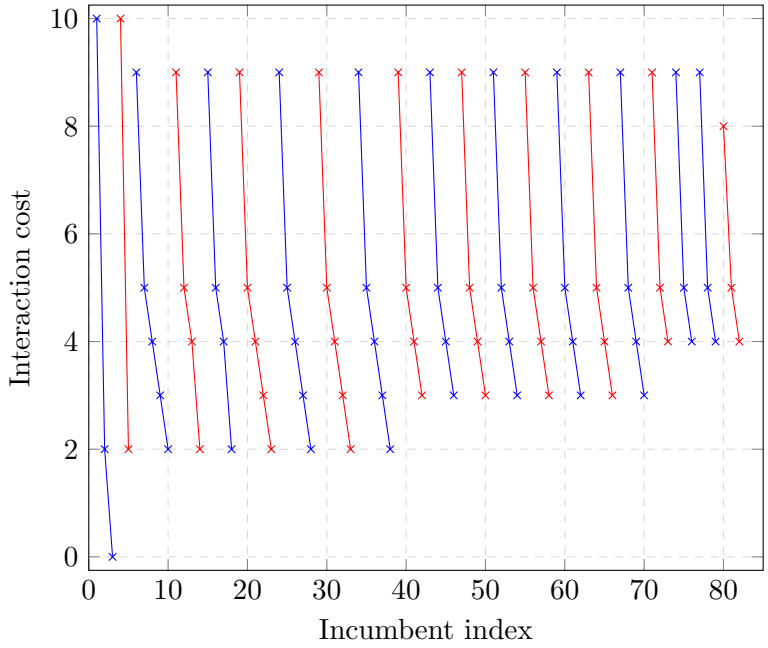

Figure 4.5: Interaction costs of the sequence of best integer incumbents encountered during the non-overlapping partitioning of the system (4.54). Each new (blue and red) line represents a new optimization performed after a controllability cut is added.

to notice that subsystem 1 is completely decoupled from the other two subsystems.

In this example the total number of distinct ways to partition the system (4.54) in two non-overlapping subsystems is as follows,

$$3!\left\{ {5 \atop 3} \right\}^2 = 3750 \tag{4.56}$$

4.9.3 Example with State Overlapping

Finally, this subsection includes an example presenting a system partitioning including a one state overlap between two subsystems.

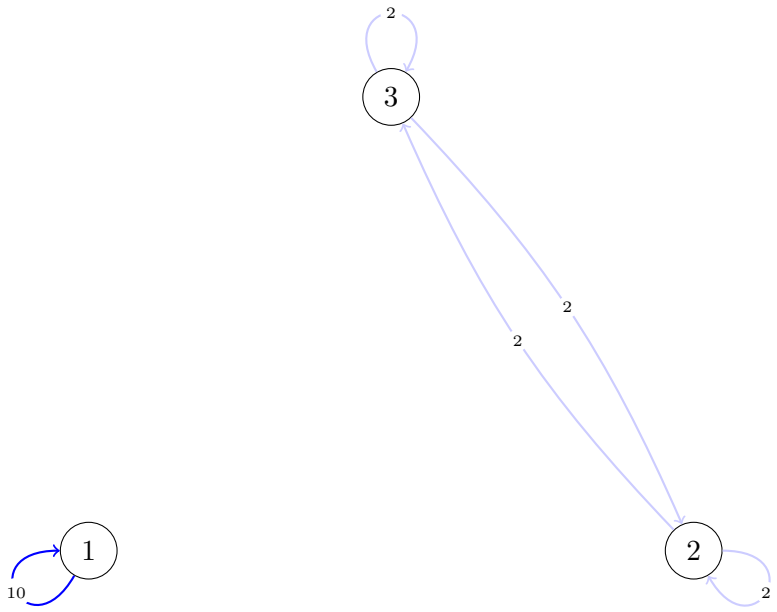

Figure 4.6: Subsystem graph representation of the least interacting subsystems of the system (4.54).

$$A = \begin{bmatrix} 1 & -24 & 1 \\ 1 & -20 & 1 \\ 1 & -24 & 1 \end{bmatrix} \tag{4.57a}$$

$$B = \begin{bmatrix} 2 & 0 \\ 0 & 0 \\ 0 & 2 \end{bmatrix} \tag{4.57b}$$

$$N = 2 \tag{4.57c}$$

$$q = 1 \tag{4.57d}$$

This example has been taken from (Šiljak, 1991), and as it can be seen from the optimization results, after 2 controllability cuts, two state overlapping subsystems are obtained. This optimization for this system has been performed in $1.75s$ on a standard desktop computer.

$$\alpha = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \beta = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.58}$$

The interaction costs linked to the best integer incumbents encountered during

the branch and cut system partitioning are presented in Table 4.4.

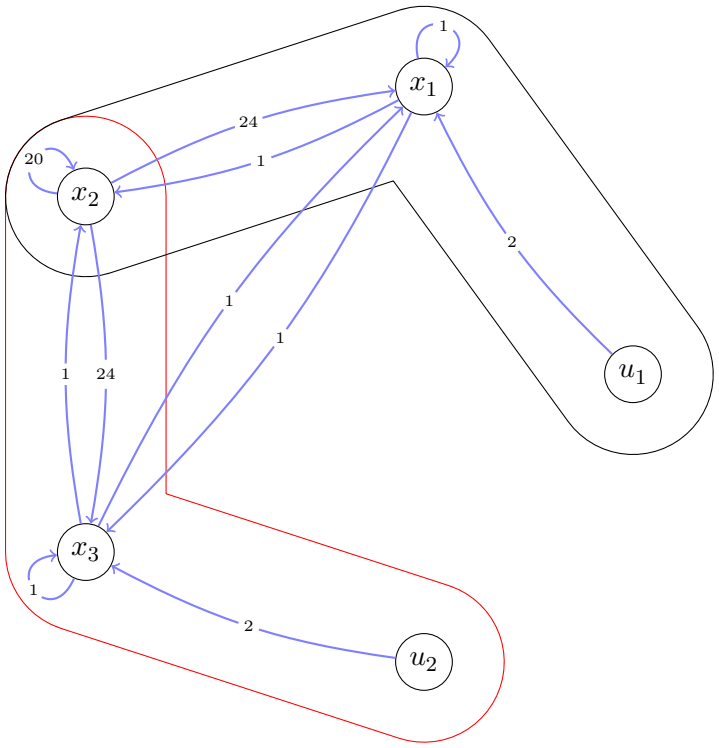

Figure 4.7: Weighted digraph representation of the two controllable subsystems for the system model (4.57) with one state variable overlap.

Table 4.4: Best integer incumbent costs for the partitioning of the system (4.57).

Incumbent index	**Cut index**	**Interaction cost**
1	0	27
2	0	4
3	1	29
4	1	4
5	2	29
6	2	27
7	2	8
8	2	4

4.10 Conclusion

A binary integer linear programming approached has been presented within this chapter to tackle the problem of partitioning a system model into subsystems models. The system model is partitioned into state overlapping or non-overlapping

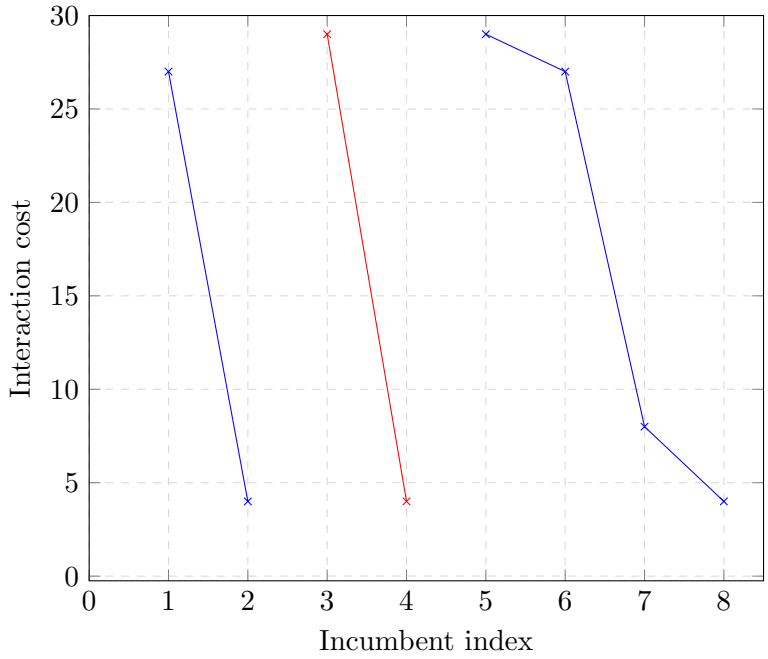

Figure 4.8: Interaction costs of the sequence of best integer incumbents encountered during the non-overlapping partitioning of the system (4.57). Each new (blue and red) line represents a new optimization performed after a controllability cut is added.

controllable subsystem models presenting the least amount of interactions. Firstly, the problem has been formulated as a non-linear integer optimization. Then, it has been demonstrated that auxiliary binary variables could be introduced in order to linearize the objective function and therefore yield a binary integer linear program. Finally, a method similar to the Gomory cut technique has been implemented in order to rule out the least interacting partitionings including at least one non-controllable subsystem model. Based on the duality between the controllability and observability properties, a similar partitioning technique can be implemented in order to partition a system into least interacting but observable subsystems. Since the partitioning problem is a combinatorial problem, the size of the search space increases very rapidly with the size of the system and the number of subsystems. In addition to this, binary integer linear programs have an exponential worst case complexity even if they are most of the time tractable in practice. Subsequently, the computational cost can be important for large-scale systems. Nonetheless, because the architecture optimization is performed offline, as a preliminary step before designing the control system, more time and computing power can be used if necessary to reach an optimal solution. Future research directions could provide an extension to the weak interaction partitioning problem to a full system state space model. Also, there is some interest in finding a way to speed up the partitioning algorithm,

possibly based on a warm start technique after partitioning cuts are applied.

Chapter 5

Supervised-distributed Control with Joint Performance and Communication Optimization

5.1 Introduction

Decentralized control architectures have attracted attention since the 1960s (Lunze, 1992). In non-centralized architectures, each subsystem is equipped with a local controller that may or may not share information with the other local controllers. Distributing the control actions brings different advantages, for instance it allows to decrease the weight of a system and also provides an increase in its modularity. Nowadays, systems are growing in size and complexity and they are most of the time composed of smaller interacting subsystems, integrated together. Control engineers design controllers using a specific control architecture adapted to the subsystem interactions (Scattolini, 2009). Therefore, the first problem to tackle is to select a controller architecture suitable for the system. This task is often achieved by minimizing the coupling between subsystems while keeping them controllable (Bristol, 1966; Kariwala et al., 2003). Another common technique consists of finding the input and output pair having the highest sensitivity (Manousiouthakis et al., 1986). For distributed control architectures, a communication scheme has to be selected and implemented (Maestre et al., 2009; Gross and Stursberg, 2016). Simply broadcasting the state variables between the local controllers at every time step will increase the overall performance of the control system but this will be achieved at a high communication cost. For instance, if the sensors are transmitting using a wireless communication medium, more energy will be required in order to transmit and receive the data packets (Ye and Heidemann, 2002). Also, for security reasons as well

as for privacy, communicating less often and only when the control performance is at stake, decreases the risk of network tapping and hacking from external entities. Distributed control architectures are relevant for future gas turbine engines relying on wireless sensor networks. In such systems, most sensors will be battery powered and the electrical energy they will use will be harvested from engine vibrations or thermal differences between engine components. Consequently, the main motivation of such a control system technology for next generations gas turbine engines lies in optimizing the sensors energy consumption. Therefore, one of the major challenges for distributed control systems is to solve the joint optimization problem balancing the communication effort between local subsystem controllers along with the system wide performance. The principal application of such a control method would be for steady state disturbance rejection. In this case, the control law is updated online based on the value of the state variable, and the controller switches between different communication topologies as a function of the system predicted performance. Finally, such a control method could be used to homogenize the control system performance among a set of similar systems with slightly different dynamics. The performance between the systems is balanced based on different control laws relying on distinct communication topologies. In the first case, the optimization is performed online and yields a sequence of control modes, in the second case the optimization is performed offline over a set of systems with similar dynamics.

Therefore, the next problem of importance is the controller design itself. Some approaches have focused on the implementation of fully decentralized controllers (Šiljak, 1991; Bakule, 2008). The synthesis of decentralized control laws has become an important research topic for large-scale systems (Lunze, 1992). Decentralized control systems answer the need for decreasing the shared information between the subsystems and therefore the number of communication links between local controllers. For instance, an efficient communication scheme based on game theory has been developed (Maestre et al., 2014). It has been used for systems coupled through their inputs after the system partitioning is performed. Following the work on decentralized control design, more research approaches have addressed the problem of structurally constrained control law synthesis.

It is well known that in most cases, the computation of structurally constrained controllers is a difficult problem, however, some particular systems complying with a quadratic invariance condition are tractable and can be formulated as convex optimization problems (Rotkowitz and Lall, 2006). Some approaches have performed the optimal design of decentralized controller in a framework such that the feedback gain obtained is as sparse as possible, while keeping acceptable system wide performance (Schuler et al., 2014). Most of these research approaches are based on

compressed sensing techniques developed to deal with sparsity (Candès and Tao, 2005). For instance, common convex relaxations of the l_0-norm such as the l_1-norm (Candès, 2008), or reweighed l_1-norm are frequently implemented as sparsity inducing costs (Candès et al., 2008). Other approaches have focused on row or column sparsity, in order to minimize respectively the number of sensors and actuators in use (Polyak et al., 2013). Finally, some approaches have managed to promote sparsity in the controller while minimizing the $\mathcal{H}_2$ performance index (Babazadeh and Nobakhti, 2016), or to induce a desired structure on the feedback gain using sufficient conditions (Crusius and Trofino, 1999).

This chapter provides a solution to the joint problem of optimizing communication and system performance in a distributed framework. A control technique where a supervisory agent recomputes online a state feedback controller in order to minimize an objective combining the infinite horizon control cost with a communication metric is proposed. The control law is updated online based on the value of the state variables and then sent to the local controllers in order to be implemented. Consequently, an optimization problem is solved online in order to decrease the joint communication and performance metrics and provide a sparse control law based on the value of the state variables. This type of control method can be used to provide efficient system regulation, where the controller relies on communication channels only to tackle more efficiently the disturbances. Also, convex constraints can be added to the problem in order to tackle constraints on the state and input variables as well as to provide robustness to system model uncertainty.

The work presented within this chapter shows how to compute a new linear state feedback gain optimally with regards to the control performance cost as well as a communication metric. The optimization problem is formulated as a Semi-definite Programming (SDP) problem including a Bilinear Matrix Inequality (BMI). This optimization problem is then relaxed using an alternate convex search method, also convergence of such a technique for this type of problem is proved. In addition to this, it is demonstrated that a feasible stable solution can always be computed, even after early termination of the algorithm. The asymptotic stability of the closed-loop system is guaranteed based on a dwell time requirement for switching between the different control modes generated. Finally, recursive feasibility of the supervised-distributed control algorithm is demonstrated.

The content presented within this chapter has been submitted for publication (Guicherd et al., 2019). The chapter is organised as follows, section 5.2 states the problem and section 5.3 introduces the required notations for the constraints and the objective function, leading to the formulation of the optimization problem. Section 5.4 demonstrates the stability and recursive feasibility of the supervised-distributed

control technique. In section 5.5, the control algorithm is presented along with a proof of its convergence. Also it is shown that extra constraints can be added to the optimization problem in order to include physical system constraints as well as system model uncertainty. Section 5.6 analyses the control algorithm complexity. In order to illustrate the control algorithm efficacy, section 5.7 includes two numerical examples, finally section 5.8 concludes the chapter.

5.2 Problem Statement

5.2.1 Distributed System Dynamics

Consider a discrete time, linear, time invariant system (5.1a) already partitioned into $N \in \mathbb{N}^*$ stabilizable subsystems. Such that, for all $p \in [\![1, N]\!]$ the subsystem indexed by p is modelled as per equation (5.1b).

$$x^+ = Ax + Bu \tag{5.1a}$$

$$x_p^+ = A_{pp}x_p + B_{pp}u_p + \sum_{\substack{j=1 \\ j\neq p}}^{N} \Big\{ A_{pj}x_j + B_{pj}u_j \Big\} \tag{5.1b}$$

with $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are respectively the system state and input variables, the vector x^+ denotes the successor system state. The matrices A and B are of appropriate dimensions and denote the system state space model. For all $p \in [\![1, N]\!]$, $x_p \in \mathbb{R}^{n_p}$ and $u_p \in \mathbb{R}^{m_p}$ are the state and input variables of the subsystem p, the vector x_p^+ denotes the successor subsystem state. The matrices A_{pp} and B_{pp} are of appropriate dimensions representing the subsystem model indexed by p, such that

$$\sum_{p=1}^{N} n_p \geq n \tag{5.2a}$$

$$\sum_{p=1}^{N} m_p = m. \tag{5.2b}$$

The right hand side sum in (5.1b) represents the subsystem couplings. The equation (5.2a) defines a possible overlapping condition for the subsystem state variables, whereas the equation (5.2b) denotes a condition of non-overlapping for the subsystem input variables. In other words, a state variable and an input variable can respectively be shared by multiple subsystems or belongs to a unique subsystem.

The following assumption is made with regards to the overall system as well as the subsystems.

Assumption 5.1. *The system* (A, B) *as well as, for all* $p \in [\![1, N]\!]$ *the subsystem* (A_{pp}, B_{pp}) *are stabilizable.*

Assumption 5.1 implies that a linear stabilizing state feedback controller $F \in \mathbb{R}^{m \times n}$ exists along with a decentralized controller. The next subsection formulates the optimal distributed control problem.

5.2.2 Control Problem

The aim is to synthesize online and periodically an optimal state feedback control law $u(k) = f(x, k)$, that simultaneously minimizes the classical Linear Quadratic Regulator (LQR) cost as well as a subsystem to subsystem communication metric, defined within the next section of this chapter. In particular, when the control law is linear (i.e. $u_k = Fx_k$) and designed as the optimal linear quadratic regulator gain F, then F is state invariant and fully centralized in general. The system (5.1) in closed-loop control with the LQR control law F is then represented as follows,

$$x^+ = (A + BF)x. \tag{5.3}$$

Without loss of generality, the state and input matrices $A = (A_{ij})_{(i,j)\in[\![1,N]\!]^2}$ and $B = (B_{ij})_{(i,j)\in[\![1,N]\!]^2}$ represent the system state space model along with its subsystem block decomposition. This decomposition is always achievable throughout the reorganization of the system state and input variables. As a consequence, the block matrices composing the matrix F represent linear feedback gains from a subsystem state variable to a subsystem input variable. More specifically, diagonal blocks represent feedback gains within a subsystem, whereas off-diagonal blocks represent gains between two distinct subsystems. In a similar fashion, F can be decomposed into N^2 block matrices as follows,

$$\forall (i, j) \in [\![1, N]\!]^2,\ F_{ij} \in \mathbb{R}^{m_i \times n_j},\ F = (F_{ij})_{(i,j)\in[\![1,N]\!]^2}. \tag{5.4}$$

Therefore, if the block matrix F_{ij} is different from the zero matrix, it implies communication from the subsystem j to the subsystem i. The supervised-distributed control framework is presented in Figure 5.1 and detailed in the Example 5.1.

Example 5.1 (Supervised-distributed framework). *This example presents the idea behind the supervised-distributed control technique. The gain matrix* F *is broadcast to the system composed of two non-overlapping subsystems, indexed by* 1 *and* 2*. The supervisory unit in Figure 5.1 periodically updates the control law* F *used*

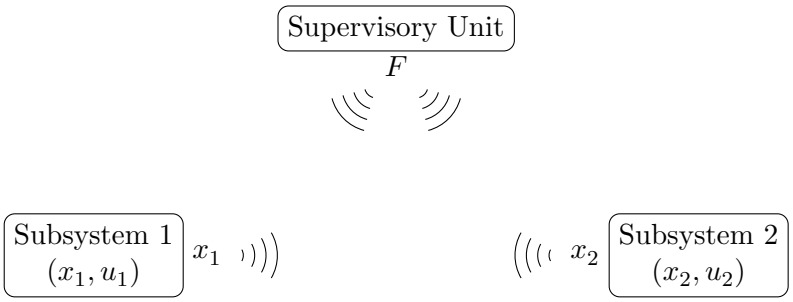

Figure 5.1: Representation of the information communicated between the subsystem 1, the subsystem 2 and the supervisory unit.

by the entire system, it can be noted that the communication structure between the subsystems is also included in the block structure of F. The gain matrix F can be decomposed into blocks as follows,

$$F = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}. \tag{5.5}$$

Therefore, the subsystems control inputs are calculated based on the following relations,

$$\begin{cases} u_1 = F_{11}x_1 + F_{12}x_2 \\ u_2 = F_{21}x_1 + F_{22}x_2 \end{cases} \tag{5.6}$$

In this example, when F_{12} is equal to the zero matrix, then the subsystem 2 does not have to communicate its state variable to the subsystem 1. Similarly, when the F_{21} with the subsystem 1. Therefore, in this case a block diagonal F corresponds to the decentralized control of the system, whereas a non-sparse F correspond to a full use of the subsystem to subsystem communication.

The objective to be minimized online by the supervised-distributed control algorithm is the quadratic cost to infinity under the linear control law F, as well as the associated communication effort. The latter being defined by the off-diagonal block sparsity of the control law F. The quadratic control cost is predicted based on the use of the control mode F, from the current time step onwards. The optimization problem is solved online periodically and in a receding horizon manner. The next section presents the formulation of the two parts composing the objective function as well as the constraints associated with the optimization problem, thus defining the approach taken.

5.3 Proposed Approach

The cost function formulated here includes a part linked to the control performance, representing the effort to steer the states to a reference using a controller F, as well as a penalty on the state variable discrepancy. The second part of the cost is a metric associated to the number of subsystem to subsystem communication links in use in the control law F.

5.3.1 Infinite Horizon Control Cost

The infinite horizon control quadratic cost is given by an infinite sum on the state and input variables (5.7). In the linear case, it only depends on the initial state, the state space model and the control law implemented (Zhou et al., 1996).

$$J_{ctrl} = \sum_{k=0}^{+\infty} \left\{ x_k^\top Q x_k + u_k^\top R u_k \right\} = x_0^\top P x_0 \tag{5.7}$$

where, $(Q, R) \in \mathbb{S}_+^n \times \mathbb{S}_{++}^m$ are the weighting matrices used in order to penalize adequately the state and input discrepancies respectively. The vectors x_k and u_k represent the system state and input variables at time step k. The matrix $P \in \mathbb{S}_{++}^n$ is the unique positive definite solution of the following discrete time Lyapunov equation

$$(A + BF)^\top P (A + BF) - P + Q + F^\top R F = 0. \tag{5.8}$$

The optimal feedback control law F can be computed by solving the discrete time algebraic Riccati equation (5.9a). It provides the optimal feedback gain (5.9b) that minimizes the infinite horizon control cost (5.7).

$$P = A^\top P A - (A^\top P B)(R + B^\top P B)^{-1}(B^\top P A) + Q \tag{5.9a}$$

$$F = -(R + B^\top P B)^{-1} B^\top P A \tag{5.9b}$$

These algebraic equations can be solved in many different ways (Laub, 1979; Zhou et al., 1996), including a semi-definite programming technique (Boyd et al., 1994; Kothare et al., 1996) used and explained in more details latter in this chapter. The next subsection presents the formulation of the communication cost that composes the second part of the distributed control algorithm objective function.

5.3.2 Communication Cost

The cost linked to the subsystem to subsystem communication is based on the sparseness of the off-diagonal blocks of the state feedback gain F. The corresponding entries of the feedback gain matrix rely on communication between local controllers and subsystem sensors. However, this communication metric does not evaluate the communication exchange information rate, it only represents the need for a communication link. Previously, a study on the effect of the communication bit rate allocation has already been performed (Xiao et al., 2003). The communication cost promoting sparsity employs the l_0-norm applied to the vectorized form of the feedback gain such that,

$$J_{comm} = \left\| \operatorname{vec}(W \circ F) \right\|_0 \tag{5.10}$$

where, $W \in [\![0,1]\!]^{m \times n}$ is a binary masking matrix used to select the entries of F relying on communication and to penalize. Sparseness of a vector or a matrix is penalized using the l_0-norm which associates a binary element to each of the entries based on their value. Therefore, the l_0-norm corresponds to the number of non-zero entries in a matrix or vector. Finding sparse solutions represents a combinatorial problem and is in general Non-deterministic Polynomial (NP) hard, usually requiring exhaustive search with exponential complexity. Since the l_0-norm does not comply with all the norm properties (Hurley and Rickard, 2009), and is therefore not a true norm, a common convex relaxation is to use the l_1-norm. Indeed, the l_1-norm is a well known sparsity promoting penalty. This heuristic has been used in the past in different fields such as optimal control, compressed sensing and signal reconstruction (Candès and Tao, 2005). Necessary and sufficient conditions have been established for this convex relaxation to be tight (Candès et al., 2008). Therefore, as it has been done in previous research approaches, sparsity is promoted using the l_1-norm as a convex relaxation in the following communication cost,

$$J_{comm} = \left\| \operatorname{vec}(W \circ F) \right\|_1. \tag{5.11}$$

The next subsection explains how the gain masking matrix W is computed from the definition of the distributed control system architecture.

5.3.3 Control Gain Masking Matrix

In order to account for the communication cost, a penalty on the entries of F representing subsystem to subsystem communication has to be accounted for. This is achieved using a matrix W composed of binaries and used as control gain masking

matrix. The control masking matrix $W \in [\![0,1]\!]^{m \times n}$ is multiplied element-wise with the feedback control gain in order to add a zero weight to the entries of F when they do not represent subsystem to subsystem communication and a weight of one to the other entries of F, the ones relying on subsystem to subsystem communication. Subsequently, the control masking matrix W can be computed from the control system architecture information. More specifically, computing W requires to know the set of state variables and input variables belonging to each of the subsystems, in order to flag the entries used to compute an input variable based on state variables coming from different subsystems. Therefore, the control masking matrix W can be calculated with the following matrix product,

$$W = \beta^{\top}(1-\alpha) = (1-\beta)^{\top}\alpha \tag{5.12}$$

where, $\alpha \in [\![0,1]\!]^{N \times n}$ and $\beta \in [\![0,1]\!]^{N \times m}$ are respectively the state and input grouping matrices representing the subsystem partitioning (5.1b) of the system (5.1a). The matrices $(1-\alpha)$ and $(1-\beta)$ represent respectively the conjugates of the state and input grouping matrices, and are obtained by replacing the 1 with 0 and vice versa (Guicherd et al., 2017). The masking matrix W is built by multiplying the transpose of the input grouping matrix with the conjugate of the state grouping matrix, hence, ones are located only when an input variable does not belong to the same group as a state variable. It can be noted that it is possible for α to be an overlapping state grouping matrix in order to allow for one or multiple state variables to be shared amongst different subsystems. Finally, the communication cost (5.11) could be computed using the mathematical trace operator in the same way as in Chapter 4 when it has been used to compute the subsystem state and input interaction metrics.

$$J_{comm} = \mathrm{tr}\left((1-\beta)|F|\alpha^{\top}\right) = \mathrm{tr}\left(\beta|F|(1-\alpha)^{\top}\right) \tag{5.13}$$

The square matrices computed within the trace operators in equation (5.13) compute on their diagonal entries the sum of the magnitudes of the gains resulting from subsystem communication with an index different from the diagonal element. The trace operator is then used to sum these gain magnitudes representing the communication effort in the same way as previously in equation (5.11). The next subsection presents the optimization problem to be solved recursively in order to determine what communication links are needed based on the system performance.

5.3.4 Minimization of the Control and Communication Costs

In this subsection, the optimization problem dealing with the infinite control cost and the communication metric is formulated using Linear Matrix Inequalities (LMIs) as well as a bilinear matrix equality constraint. The objective to minimize is the convex performance index taking into account both the control and communication costs and given in (5.14).

$$J = \lambda J_{ctrl} + (1-\lambda) J_{comm} \tag{5.14a}$$

$$\Leftrightarrow J = \lambda x_0^\top P x_0 + (1-\lambda) \left\| \operatorname{vec}(W \circ F) \right\|_1 \tag{5.14b}$$

where $\lambda \in]0,1]$ is a design tuning parameter implemented in order to balance the control (5.7) and communication (5.11) objectives. The constraints formulated for the optimization problem are LMI constraints as implemented in the optimization problem (5.15). They are implemented in order to minimize an upper bound on the infinite horizon control cost. A bilinear matrix equality is added to the constraint set so that the sparsity promoting objective can be expressed within the cost function. The optimization problem (5.15) presented below articulates the trade-off between the control and communication objectives

Remark 5.1. *Strictly speaking the optimization variables should be indexed with the control mode index to emphasize the fact that these modes are recomputed online and periodically. However, for the sake of conciseness, these indexes are used only for the asymptotic and recursive feasibility proofs.*

$$\begin{aligned}
\underset{\rho,F,X,K}{\text{minimize}} \quad & \lambda\rho + (1-\lambda) \left\| \operatorname{vec}(W \circ F) \right\|_1 \\
\text{subject to} \quad & \begin{bmatrix} X & XA^\top + K^\top B^\top & XQ^{\frac{1}{2}} & K^\top R^{\frac{1}{2}} \\ AX + BK & X & 0 & 0 \\ Q^{\frac{1}{2}} X & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}} K & 0 & 0 & \rho I_m \end{bmatrix} \succeq 0 \\
& \begin{bmatrix} 1 & x_0^\top \\ x_0 & X \end{bmatrix} \succeq 0 \\
& FX = K
\end{aligned} \tag{5.15}$$

Theorem 5.1. *Any feasible solution of the optimization problem* (5.15) *constitutes a stable feedback control law, stabilizing for the system* (5.1a).

Proof. Applying the Schur complement to both linear matrix inequality constraints in the optimization problem (5.15) yields the relations (5.16) and (5.17) respectively.

$$\begin{bmatrix} X & XA^\top + K^\top B^\top & XQ^{\frac{1}{2}} & K^\top R^{\frac{1}{2}} \\ AX + BK & X & 0 & 0 \\ Q^{\frac{1}{2}}X & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}}K & 0 & 0 & \rho I_m \end{bmatrix} \succeq 0 \tag{5.16a}$$

$$\Leftrightarrow \begin{cases} \rho \geq 0 \\ X \succeq 0 \\ X - (AX + BK)^\top X^{-1}(AX + BK) \succeq \rho^{-1}(XQX + K^\top RK) \end{cases} \tag{5.16b}$$

$$\Leftrightarrow \begin{cases} \rho \geq 0 \\ X \succeq 0 \\ \rho X^{-1} - (A + BKX^{-1})^\top \rho X^{-1}(A + BKX^{-1}) \succeq Q + X^{-1}K^\top RKX^{-1} \end{cases} \tag{5.16c}$$

$$\Leftrightarrow \begin{cases} \rho \geq 0 \\ X \succeq 0 \\ (A + BF)^\top P(A + BF) - P \preceq -(Q + F^\top RF) \end{cases} \tag{5.16d}$$

The equation (5.16) is equivalent to the Lyapunov equation (5.8) with a new parametrization for P and F, respectively the solution of the Lyapunov equation and the state feedback control law. The Lyapunov equation obtained is relaxed with an inequality instead of the equality of equation (5.8). The other constraint is equivalent to an invariant ellipsoid set as per equation (5.17).

$$\begin{bmatrix} 1 & x_0^\top \\ x_0 & X \end{bmatrix} \succeq 0 \tag{5.17a}$$

$$\Leftrightarrow \begin{cases} X \succeq 0 \\ 1 \geq x_0^\top X^{-1} x_0 \end{cases} \tag{5.17b}$$

$$\Leftrightarrow \begin{cases} X \succeq 0 \\ \rho \geq x_0^\top P x_0 \end{cases} \tag{5.17c}$$

The equation (5.17) provides an invariant ellipsoid set condition for the system state variable due to the strict decrease of the Lyapunov cost function. Thus, the system state variable is known to belong to the ellipsoid $\mathcal{E}$, defined by the inequality (5.17). The variable ρ represents an upper bound on the infinite horizon control cost that is minimized by the optimization problem (5.15). Hence, any feasible solutions of

the optimization problem (5.15) provides a stable control mode, which completes the proof. □

Note that the optimization problem presented in (5.15) is biconvex. Fixing the variable F or X renders convex optimization problems, respectively in the set of variables $\{\rho, X\}$ and $\{\rho, F\}$. With the parametrization of the problem (5.15), the Lyapunov solution P and the state feedback gain F are computed as follows,

$$P = \rho X^{-1} \tag{5.18a}$$

$$F = KX^{-1}. \tag{5.18b}$$

The optimization problem (5.15) minimizes a dual objective composed of a control and a communication metric, therefore it is in general different from a standard optimal linear quadratic control design problem. The next subsection explains why the optimization problem is different from a standard LQR optimal control problem and how it is affected by the communication objective.

5.3.5 Time-varying Control Modes

When the parameter λ is set to 1, the standard LQR feedback gain is returned by the optimization as the global optimum. Consequently, the optimization problem (5.15) without the sparsity promoting objective is state invariant. The minimum is reached for the solution of the discrete time algebraic Riccati equation (5.9a). In the case of (5.15), different degrees of sparsity of F will affect the infinite horizon control cost throughout the dynamical couplings between the subsystems. Indeed, completely dynamically decoupled subsystems would not require subsystem to subsystem communication and a sparse feedback control law would be optimal with regard to the performance as well as the communication. Also, in the case of (5.15), the sparsity level of F is balanced with the infinite horizon control cost in a semi-definite programming problem and consequently requires the inclusion of a bilinear matrix equality. It is well known that bilinear matrix equalities are usually non-convex and therefore constitute difficult problems to solve. In the literature different techniques have been used in order to solve these types of optimization problems. For instance, one approach implemented a sequential method where a penalty on the bilinear equality gap was added to the objective function (Doelman and Verhaegen, 2016). A convergence guarantee has been proven, however there is no insurance that the optimum reached will be the global optimum. Another technique called the alternate convex search uses iterative convex relaxations of the biconvex problem

(Gorski et al., 2007). The alternate convex search also known as alternate SDPs method (Fukuda and Kojima, 2001), works by alternately fixing one of the variables in the bilinear matrix equality constraint. The two convex optimization problems are solved alternately until a stopping criteria is met. Other techniques, relying on a branch-and-cut strategy have been developed in order to reach the global optimum value for a bilinear matrix inequality optimization problem (Goh et al., 1995; Fukuda and Kojima, 2001). However, the technique mentioned above requires a lot of time and computational power which makes any online implementation very difficult. The next section introduces a new set of convex constraints that are added to the optimization problem (5.15) in order to ensure global asymptotic stability of the switched closed-loop control system. Also, the recursive feasibility of the supervised-distributed control method is demonstrated.

5.4 Stability and Feasibility of the Distributed Controller

This section establishes two very important properties of the online optimization controller, the stability as well as the recursive feasibility. These two features of the distributed control technique are proved and a theorem gathers the overall result for the supervised-distributed control method.

5.4.1 Controller Stability

Since, the distributed controller minimizes not only the infinite horizon control cost but also the communication effort, and that no constraints are added to ensure the stability of the switched control system, instability can be triggered by switching between different control modes inadequately. For example, the system presented (5.19) is stable and can be stabilized by any of the two feedback control laws given in (5.20). Also, the two closed-loop systems obtained by implementing the controller F_0 and F_1 belong to the set of feasible solutions of the optimization problem (5.15), as stabilizing control laws. Nonetheless, instability can still be triggered by a rapid switching between these two control laws, as shown by the Example 5.2.

Example 5.2 (Switching system instability). *The Linear Time-invariant (LTI) discrete time system given by the equation* (5.19)*, can become unstable by an inap-*

propriate switching between the two control laws F_0 and F_1 (5.20).

$$A = \begin{bmatrix} -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{5.19}$$

$$F_0 = \begin{bmatrix} 2 & 0 \\ 0 & -\frac{1}{2} \end{bmatrix}, \qquad F_1 = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & -2 \end{bmatrix}. \tag{5.20}$$

Since the two control gains are stabilizing for the system (5.19), their Lyapunov functions can be computed in order to evaluate their infinite horizon control costs. The quadratic Lyapunov functions associated to the two closed-loop systems are respectively given by the following matrices,

$$P_0 = \begin{bmatrix} 25.71 & 5.71 \\ 5.71 & 6.43 \end{bmatrix}, P_1 = \begin{bmatrix} 6.43 & 5.71 \\ 5.71 & 25.71 \end{bmatrix}. \tag{5.21}$$

A direct switching between the control modes F_0 and F_1 brings instability to the switched controlled system. Indeed, this is due to the fact that for $\Delta = 1$, the following two inequalities presented below are not verified,

$$(A + BF_1)^{\Delta\top} P_0 (A + BF_1)^{\Delta} \prec P_1 \tag{5.22a}$$

$$(A + BF_0)^{\Delta\top} P_1 (A + BF_0)^{\Delta} \prec P_0. \tag{5.22b}$$

The phase portraits of the direct switching sequences between F_0 and F_1 are given Figure 5.2. Such switching sequences yield unstable switched closed-loop systems diverging away from the origin. The red curve shows a direct switching sequence starting with the control law F_0, whereas the blue curve pictures a direct switching sequence initialized with the control mode F_1. Both phase portraits for these two switching sequences start from an initial state given by $x_0 = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix}^\top$. It is well known that switching between stable control laws can be achieved in a stable fashion if the switching is performed slowly enough (Liberzon, 2003). The time delay before a switch is triggered is called the dwell time and is parametrized by $\Delta \in \mathbb{N}^*$ in the case of a discrete time system. Subsequently, if the dwell time parameter Δ is not chosen carefully, a switching sequence between the two previous control laws can provide a control cost increase and therefore destabilize the switched system, even if the two control modes are stable. An analysis of the dwell time effect on the inequalities (5.22) shows that only certain values of Δ will provide a stable switching sequence between these two control modes. However, as it will be proved later on within this section, since the two control laws are stabilizing for the closed-loop system, when the switching dwell time Δ is larger than a minimum value the control cost to infinity

decreases and thus, the switched controlled system remains asymptotically stable.

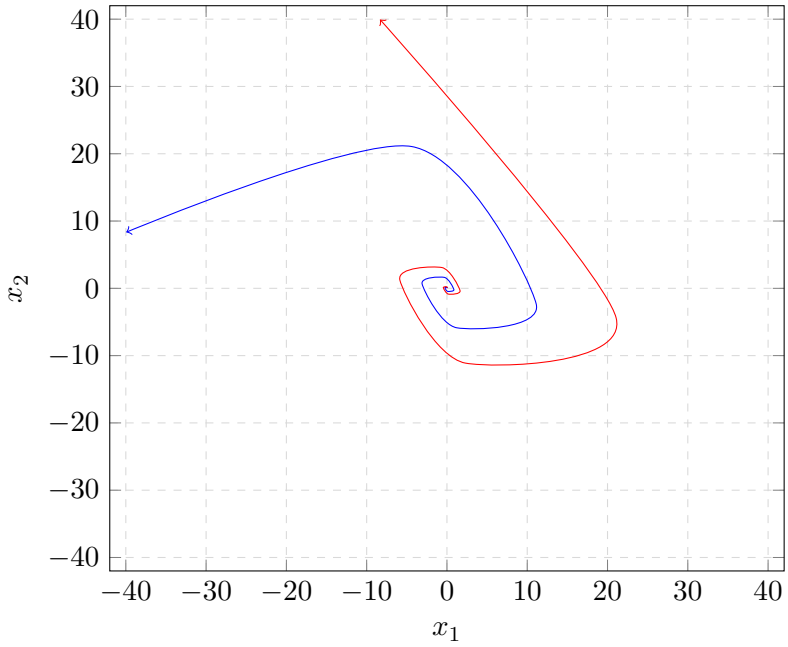

Figure 5.2: Phase portraits of the two direct switching sequences between the control laws F_0 and F_1.

In order to ensure a Lyapunov cost decrease for the switched controlled system, the dwell time Δ has to be chosen carefully so that the inequality (5.22a) *is satisfied, when a switch from F_1 to F_0 occurs, and the inequality* (5.22b) *has to be satisfied for a switch from F_0 to F_1.*

$$M_\Delta = P_1 - (A + BF_1)^{\Delta\top} P_0 (A + BF_1)^\Delta \tag{5.23a}$$
$$N_\Delta = P_0 - (A + BF_0)^{\Delta\top} P_1 (A + BF_0)^\Delta \tag{5.23b}$$

Satisfying these inequalities is equivalent to finding the values of Δ such that $M_\Delta \in \mathbb{S}^2_{++}$ and $N_\Delta \in \mathbb{S}^2_{++}$ respectively. The eigenvalues of these two matrices for the first few values of Δ are given Table 5.1, in this example these values are equal because of the system symmetry. It can be seen that for a value of Δ greater or equal to 6*, switching between the two control laws provides a stable switched control system. Intuitively, since the spectral radii of $(A+BF_0)$ and $(A+BF_1)$ are strictly less than one, it implies that the matrices M_Δ and N_Δ will converge to the matrices P_1 and P_0 respectively. Since M_Δ and N_Δ converge with increasing values of Δ, it implies that there exists a value $\Delta_{min} \in \mathbb{N}^*$ possibly different for M_Δ and N_Δ guaranteeing that any higher value of Δ will ensure a stable switch for the control system. Computing*

the value of Δ_{min} is usually a complex problem to solve, especially when the set of control modes is large. The eigenvalues of M_Δ and N_Δ are plotted on the Figure 5.3, it is possible to see that they oscillate while converging towards the eigenvalues of M_Δ and N_Δ respectively. Although, for any value of Δ greater than 6 both eigenvalues are strictly positive.

Table 5.1: Eigenvalues of M_Δ and N_Δ for different values of switching dwell times Δ.

Δ	$\lambda_{max}(M_\Delta)$	$\lambda_{min}(M_\Delta)$	$\lambda_{max}(N_\Delta)$	$\lambda_{min}(N_\Delta)$
1	4.176189	-70.247618	4.176189	-70.247618
2	22.304213	-8.241713	22.304213	-8.241713
3	4.535292	-27.637971	4.535292	-27.637971
4	24.267681	-2.295025	24.267681	-2.295025
5	4.842501	-3.775258	4.842501	-3.775258
6	25.503048	0.919072	25.503048	0.919072
7	10.130645	4.532180	10.130645	4.532180
8	26.251318	2.673624	26.251318	2.673624
9	17.570556	4.739783	17.570556	4.739783
10	26.691630	3.641150	26.691630	3.641150
11	21.812118	4.799947	21.812118	4.799947
12	26.945929	4.178760	26.945929	4.178760
13	24.203081	4.828706	24.203081	4.828706
14	27.091156	4.478981	27.091156	4.478981
15	25.548994	4.843886	25.548994	4.843886
16	27.173554	4.647148	27.173554	4.647148
17	26.306320	4.852175	26.306320	4.852175
18	27.220129	4.741516	27.220129	4.741516
19	26.732386	4.856767	26.732386	4.856767
20	27.246400	4.794525	27.246400	4.794525

The Example 5.2 introduced the concept of switched control systems along with a technique used in order to verify and therefore prevent system instability when control mode switching occurs. Indeed, the switching instability question can be tackled throughout the use of a dwell time constraint added to the optimization problem (5.15). This extra constraint enforces an asymptotic decrease of the Lyapunov cost function and is translated into a linear matrix inequality. The Figure 5.4 presents the possible evolution of $V_l(x_k)$ representing the switched Lyapunov cost function for the closed-loop system, where k is the time step index and l is the control mode index. The Lyapunov cost function V_l can increase during a switch as long as the increase of its value is maintained strictly below what the initial value was at the start of the previous control mode. In the past different approaches have been used in order to prove the stability of switched control systems, and this topic has seen the

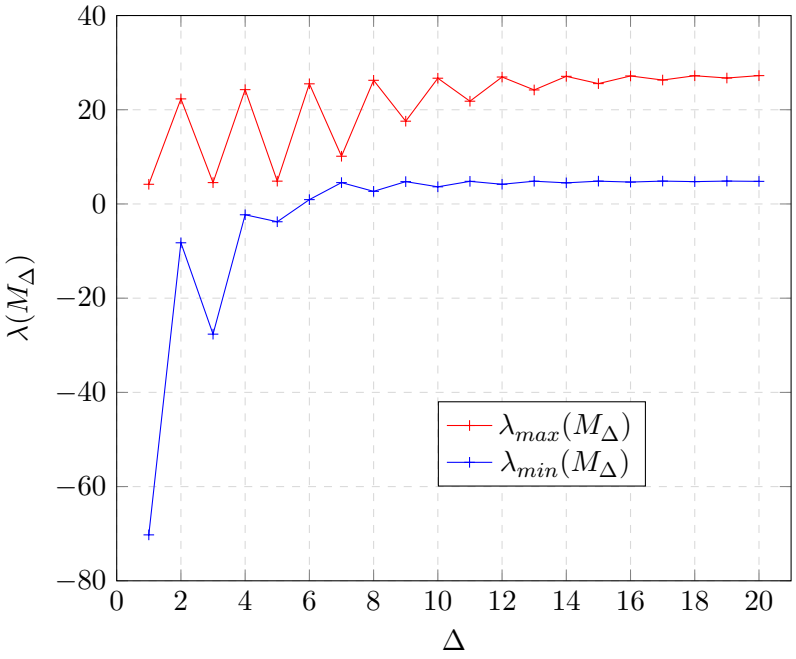

Figure 5.3: Eigenvalues of M_Δ against Δ.

development of multiple mathematical tools (Branicky, 1998). For example, switching amongst a set of systems yields a stable system if the switching is slow enough or is at least slow on average (Hespanha and Morse, 1999). This refers to the use of a minimum dwell time parameter or allows for rapid switching if and only if the switching is slow enough on average, slower than an average dwell time value. Other techniques have considered the use of a Lyapunov function common to the entire set of systems (Lin and Antsaklis, 2009). In the case of the supervised-distributed controller, every time a new feedback gain is broadcast to the plant subsystems, the entire plant becomes a switched system (Liberzon, 2003). Consequently, it is important to ensure that by design switching between two consecutive feedback gains is done so that one can guarantee global asymptotic stability of the closed-loop system (Geromel and Colaneri, 2006). In order to prevent any unstable behaviour from happening, an extra LMI constraint is added to the optimization problem (5.15) enforcing the switched asymptotic stability for the closed-loop system and leading to the formulation given in (5.24). This stability constraint is formulated using the current Lyapunov function P_l and control mode F_l in addition to a dwell time design parameter Δ. The control mode dwell time parameter Δ, with $\Delta \in \mathbb{N}^*$ guarantees that instability will not be triggered during control mode switching, if the switching is performed after at least Δ time steps. Also, the dwell time constraint allows enough time to perform the biconvex online optimization. As mentioned previously

in remark 5.1, the decision variables should all be indexed with the control mode indexes, however for simplicity the indexes are only used for the proofs. The Lemma 5.1 proves that the extra constraint added in (5.24) enforces system stability under switching before the recursive feasibility of the control algorithm is demonstrated in Lemma 5.2.

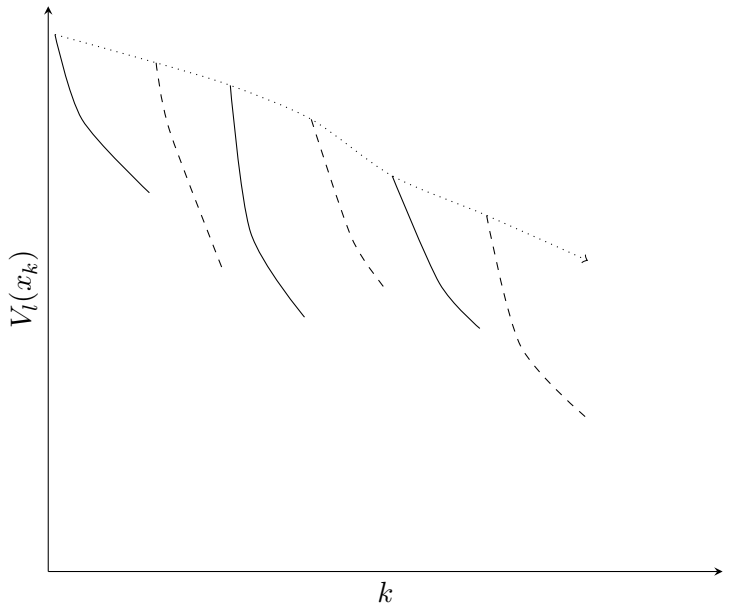

Figure 5.4: Example of a Lyapunov cost function for a stable switched system.

$$
\begin{aligned}
\underset{\rho,F,X,K}{\text{minimize}} \quad & \lambda\rho + (1-\lambda)\left\| \operatorname{vec}(W \circ F)\right\|_1 \\
\text{subject to} \quad & \begin{bmatrix} X & XA^\top + K^\top B^\top & XQ^{\frac{1}{2}} & K^\top R^{\frac{1}{2}} \\ AX + BK & X & 0 & 0 \\ Q^{\frac{1}{2}}X & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}}K & 0 & 0 & \rho I_m \end{bmatrix} \succeq 0 \\
& \begin{bmatrix} \rho P_l & \rho(A+BF_l)^{\Delta\top} & \rho\varepsilon I_n \\ \rho(A+BF_l)^{\Delta} & X & 0 \\ \rho\varepsilon I_n & 0 & \rho\varepsilon I_n \end{bmatrix} \succeq 0 \\
& \begin{bmatrix} 1 & x_0^\top \\ x_0 & X \end{bmatrix} \succeq 0 \\
& FX = K
\end{aligned}
\tag{5.24}
$$

Lemma 5.1. *Consecutive feasible solutions of the distributed control optimization*

problem (5.24) *are globally asymptotically stabilizing for the switched closed-loop system.*

Proof. The constraint added to the problem (5.24) corresponds to a stable switch from the control mode l to the next control mode $l+1$, after Δ time steps. This is shown by applying the Schur complement to the extra constraint in equation (5.25), with the previous parametrization (5.18a) of the optimization problem (i.e. $P_{l+1} = \rho X^{-1}$).

$$\begin{bmatrix} \rho P_l & \rho(A+BF_l)^{\Delta\top} & \rho\varepsilon I_n \\ \rho(A+BF_l)^{\Delta} & X & 0 \\ \rho\varepsilon I_n & 0 & \rho\varepsilon I_n \end{bmatrix} \succeq 0 \tag{5.25a}$$

$$\Leftrightarrow \begin{cases} \varepsilon\rho \geq 0 \\ X \succeq 0 \\ \rho P_l - (A+BF_l)^{\Delta\top}\rho^2 X^{-1}(A+BF_l)^{\Delta} \succeq \rho\varepsilon I_n \end{cases} \tag{5.25b}$$

$$\Leftrightarrow \begin{cases} \rho\varepsilon \geq 0 \\ X \succeq 0 \\ (A+BF_l)^{\Delta\top} P_{l+1}(A+BF_l)^{\Delta} - P_l \preceq -\varepsilon I_n \end{cases} \tag{5.25c}$$

The constraint (5.25) added to the optimization (5.24) yields the following relation when pre- and post-multiplied by $x_k^\top$ and x_k respectively

$$x_k^\top \left((A+BF_l)^{\Delta\top} P_{l+1}(A+BF_l)^{\Delta} - P_l\right) x_k < 0 \tag{5.26a}$$

$$\Leftrightarrow \left[(A+BF_l)^{\Delta} x_k\right]^\top P_{l+1}(A+BF_l)^{\Delta} x_k < x_k^\top P_l x_k \tag{5.26b}$$

$$\Leftrightarrow x_{k+\Delta}^\top P_{l+1} x_{k+\Delta} < x_k^\top P_l x_k, \tag{5.26c}$$

where x_k is the system full state vector at time step k, different from the null vector. Therefore, this ensures a strict decrease of the Lyapunov cost function for the switched control system. □

Remark 5.2. *The dwell time $\Delta \in \mathbb{N}^*$ is a designer tuning parameter. It will condition the time allocated to the optimization algorithm as well as the control feedback gain refreshing rate.*

It can be noted that the dwell time parameter could be changed online before the optimization is performed. This kind of feature could be useful in the case where some a priori knowledge on the system is known. For example, a reduction of the

dwell time can be used if a large disturbance affects the system whereas an increase of the dwell time parameter could be applied on a time window when it is known that no major disturbance will affect the system. Finally, the dwell time parameter Δ also affects the size of the feasibility set with regards to the set of control modes that can be used after Δ time steps have elapsed, this property is proved by Proposition 5.1.

Proposition 5.1. *For a given stable control mode $l \in \mathbb{N}$, the convex set*

$$\Pi_l^{\Delta} = \left\{ (\rho, X) \in \mathbb{R}_+^* \times \mathbb{S}_{++}^n \;\middle|\; \begin{bmatrix} \rho P_l & \rho(A+BF_l)^{\Delta\top} \\ \rho(A+BF_l)^{\Delta} & X \end{bmatrix} \succ 0 \right\}$$

is non-decreasing with $\Delta \in \mathbb{N}^*$ with respect to set inclusion, i.e. $\Pi_l^{\Delta} \subseteq \Pi_l^{\Delta+1}$.

Proof. For a given stable control mode $l \in \mathbb{N}$ defined by a feedback gain F_l as well as a Lyapunov function P_l, using the previous parametrization $P_{l+1} = \rho X^{-1}$, the following set inclusion arises,

$$\begin{aligned}
&\forall \Delta \in \mathbb{N}^*, \qquad (5.27)\\
&(\rho, X) \in \Pi_l^{\Delta} \Leftrightarrow (A+BF_l)^{\Delta\top} P_{l+1} (A+BF_l)^{\Delta} \prec P_l \\
&(\rho, X) \in \Pi_l^{\Delta} \Rightarrow \left[(A+BF_l)^{\Delta+1}\right]^{\top} P_{l+1} (A+BF_l)^{\Delta+1} \preceq (A+BF_l)^{\top} P_l (A+BF_l) \\
&(\rho, X) \in \Pi_l^{\Delta} \Rightarrow \left[(A+BF_l)^{\Delta+1}\right]^{\top} P_{l+1} (A+BF_l)^{\Delta+1} \prec P_l \\
&(\rho, X) \in \Pi_l^{\Delta} \Rightarrow (\rho, X) \in \Pi_l^{\Delta+1}.
\end{aligned}$$

This is due to the fact that P_l is the Lyapunov function linked to the control mode F_l, and consequently complies with the strict Lyapunov inequality. Therefore, for all $\Delta \in \mathbb{N}^*$ and for any stable control mode $l \in \mathbb{N}$, the set of stable control laws under switching is non-decreasing with Δ, $\Pi_l^{\Delta} \subseteq \Pi_l^{\Delta+1}$. □

In order to guarantee a strict decrease of the switched system Lyapunov functions after Δ time steps, the equation (5.26) must be a strict inequality. However, for a practical numerical implementation, the strict inequality is replaced by an inequality with a given ε precision, where $\varepsilon \in \mathbb{R}_+^*$ is a small but strictly positive real number. Consequently, if a solution to the optimization problem (5.24) exists, it makes the switched closed-loop system globally asymptotically stable. The next logical step is therefore to prove that the optimization problem stays recursively feasible from one control mode to the next. This property is presented within the next subsection.

5.4.2 Controller Feasibility

The global asymptomatic stability of the controller is given by the previous LMI constraint (5.15), and the stability of the control modes under switching is ensured by adding extra constraints as presented in the optimization problem (5.24). Therefore, the next important point is to make sure that this new set of constraints will not trigger the infeasibility of the optimization problem from one control mode to the next.

Lemma 5.2. *If there exists an initial asymptotically stabilizing control law for the system* (5.1)*, then the optimization problem* (5.24) *is recursively feasible.*

Proof. Consider the system given by the state space model (5.1). Due to system stabilizability, an initial stabilizing state feedback controller F_0 can be designed. This initial control mode can be computed by solving the discrete algebraic Riccati equation (5.9). In this case, F_0 is obtained by minimizing the infinite horizon quadratic cost. Consequently, there exists at least one feasible asymptotically stabilizing mode F_l, with $l \in \mathbb{N}$ for the system (5.1). Therefore, the control mode l must be feasible for the optimization time step $l+1$ with the lowest possible dwell time of 1. The control law F_l along with the Lyapunov solution P_l are feasible candidate solutions for the next optimization time step. Thus, even if the control mode F_l is not optimal for the optimization step $l+1$, it still constitutes a feasible solution. Hence, by induction, the optimization problem (5.24) remains recursively feasible. □

Following the proofs regarding the stability as well as the recursive feasibility of the supervised-distributed control algorithm, the main results are summarized in Theorem 5.2. The next subsection formulates the Theorem 5.2.

5.4.3 Overall Controller Feasibility and Stability

This subsection gathers the results on asymptotic stability and recursive feasibility developed in Lemma 5.1 and Lemma 5.2 respectively. Theorem 5.2 summarizes the properties of the optimization problem (5.24) when applied to the system given in (5.1) and complying with Assumption 5.1.

Theorem 5.2. *If there exists an initial asymptotically stabilizing control law for the system* (5.1)*, then the optimization problem* (5.24) *is recursively feasible and its consecutive solutions are globally asymptotically stabilizing control modes for the switched closed-loop system.*

Proof. The proof of this theorem is straight forward, and follows directly from Lemma 5.1 and Lemma 5.2. □

After the discussions on the stability and recursive feasibility of the supervised-distributed controller, the control system algorithm will be presented. The next section explains how the algorithm proceeds and sufficient conditions will be introduced in order to include physical system constraints as well as to ensure robustness to model uncertainty. Finally, a proof of convergence of the control algorithm is provided.

5.5 Supervised-distributed Control Algorithm

5.5.1 Unconstrained Supervised-distributed Controller

The optimization problem (5.24) yielding the distributed control modes is non-convex. Therefore, it is difficult to find a global optima or even to verify that a certain value is a global optima. However, since the problem is biconvex, including a bilinear matrix equality, known techniques exist to reach a local optimum. For instance, the alternate convex search technique is used to relax the problem into two well defined convex optimization problems that are SDP problems (Boyd and Vandenberghe, 2010). The Algorithm 5.1 presents this technique applied to the optimization problem (5.24). According to Theorem 5.3, the structure of the optimization problem, makes it possible to ensure convergence of the alternate convex search.

Theorem 5.3. *The optimization problem* (5.24) *is biconvex, more specifically convex when the sets of decision variables* (ρ, F) *and* (ρ, X) *are considered separately. Also, the objective function is always positive, subsequently bounded from below by zero. The optimization problem* (5.24) *is solvable in each set of decision variables, therefore the sequence of solutions generated by the alternate convex search converges to a stationary point that is a local optimum.*

Proof. Since the sequence of solutions generated by the alternate convex search is monotonically decreasing and since the objective function is bounded from below, subsequently the sequence of solutions converges to a limit value (Gorski et al., 2007). □

The optimization problem (5.24) is denoted by $\mathcal{P}_{x_0,X}(\rho, F)$ and $\mathcal{P}_{x_0,F}(\rho, X)$ respectively when (i) ρ, F are decision variables and x_0, X are fixed parameters, (ii) ρ, X are decision variables and x_0, F are fixed parameters. The supervised-distributed controller presented Algorithm 5.1, uses the predicted system state variable $\overline{x}_{k+\Delta|k}$ after Δ time steps under the current control mode F_l, in order to generate the next control law by alternate convex search.

Algorithm 5.1: Supervised-distributed control algorithm.

Inputs : x_k, F_l, P_l
Outputs : F_{l+1}, P_{l+1}
Parameters : Δ, λ, W, ε_F, ε_X
Initialization: $q = 0$, $F_0 = F_l$
Compute state prediction: $\overline{x}_{k+\Delta|k} = (A + BF_l)^{\Delta} x_k$
while $\|F_{q+1} - F_q\|_2 > \varepsilon_F$ or $\|X_{q+1} - X_q\|_2 > \varepsilon_X$
do
  Solve optimization problem: $\mathcal{P}_{\overline{x}_{k+\Delta|k},F_q}(\rho, X)$,
  $(\rho, X_{q+1}) = \text{argmin}(\mathcal{P}_{\overline{x}_{k+\Delta|k},F_q}(\rho, X))$.
  Solve optimization problem: $\mathcal{P}_{\overline{x}_{k+\Delta|k},X_{q+1}}(\rho, F)$,
  $(\rho, F_{q+1}) = \text{argmin}(\mathcal{P}_{\overline{x}_{k+\Delta|k},X_{q+1}}(\rho, F))$.
  $q = q + 1$
end
return
$F_{l+1} = F_q$
$P_{l+1} = \rho X_q^{-1}$

The alternate convex search is performed with different user defined parameters. Indeed, the dwell time parameter Δ, the masking matrix W, the trade-off parameter λ and the termination criteria have to be set before the Algorithm 5.1 can run. The first control mode F_0 used to initialize the control algorithm can be taken equal to any stable controller regardless of its communication requirements. One can notice that the order of the alternate convex search implemented in Algorithm 5.1 can be changed. In other words, X_q can be fixed first, before the optimization is performed with F_q fixed. Indeed, the control mode l has been shown to be a suboptimal but feasible solution of the succeeding optimization problem, at time step $l + 1$, and therefore any of the previous decision variables can be used to start the alternate convex search.

Theorem 5.4. *For a discrete time, linear, time invariant state space model, any convex combinations of asymptotically stabilizing linear feedback gains is stabilizing. Providing that each of their closed-loop spectral norm is strictly less than one.*

Proof. Consider the discrete set composed of N stable control laws such that, for all $N \in \mathbb{N}^*, (F_k)_{k\in[\![1,N]\!]} \in (\mathbb{R}^{m\times n})^N$, with $\forall k \in [\![1, N]\!]$, $\sigma_{max}(A + BF_k) < 1$, and $\theta_k \geq 0$,

with $\sum_{k=1}^{N} \theta_k = 1$.

$$\left\| A + B\Big(\sum_{k=1}^{N} \theta_k F_k\Big) \right\|_2 = \left\| \sum_{k=1}^{N} \Big(\theta_k (A + BF_k)\Big) \right\|_2 \tag{5.28a}$$

$$\Rightarrow \left\| \sum_{k=1}^{N} \Big(\theta_k (A + BF_k)\Big) \right\|_2 \leq \sum_{k=1}^{N} \left\| \theta_k (A + BF_k) \right\|_2 \tag{5.28b}$$

$$\Rightarrow \sum_{k=1}^{N} \left\| \theta_k (A + BF_k) \right\|_2 = \sum_{k=1}^{N} \left(\theta_k \left\| A + BF_k \right\|_2 \right) \tag{5.28c}$$

$$\Rightarrow \sum_{k=1}^{N} \left(\theta_k \left\| A + BF_k \right\|_2 \right) < \sum_{k=1}^{N} \theta_k = 1 \tag{5.28d}$$

The inequality (5.28b) is derived using the triangle inequality yielding an upper bound on the spectral radius of the convex combination of control laws, which concludes the proof. □

According to Theorem 5.4, for a given linear system, one only needs to know at least two stabilizing linear feedback gains complying with the spectral norm assumption in order to be able to compute an infinite amount of stabilizing controllers. Therefore, only a couple of gains are necessary in order to have access to an infinite amount of feasible initial control laws for the distributed controller. In the case where these control laws have different communication requirements, it would be possible to artificially create an initial control mode with a desired communication structure. Indeed, according to the Lyapunov stability theory, a symmetric positive definite solution exists if and only if the linear closed-loop system obtained is globally asymptotically stable. Therefore, many different gains with different communication requirements can be computed using a convex combination of existing controllers with different initial communication requirement. However, the spectral norm is an upper bound for the spectral radius (Goldberg and Zwas, 1974), hence Theorem 5.4 presents only a sufficient condition. Unfortunately, it is possible to show that the set composed of the asymptotically stabilizing feedback gains is not convex in general. This is due to the fact that the spectral radius does not define a norm on the set of square matrices. In particular, the triangle inequality does not apply to the spectral radius, therefore, a convex combination of stabilizing gains can yield an unstable controller.

As it is presented on Figure 5.5, at time step k the control law F_l is received by the subsystems and used for the next Δ time steps. In exchange, the subsystems broadcast their state variables so the optimization can be performed in order to compute the next control mode F_{l+1}. The optimization problem is solved based on

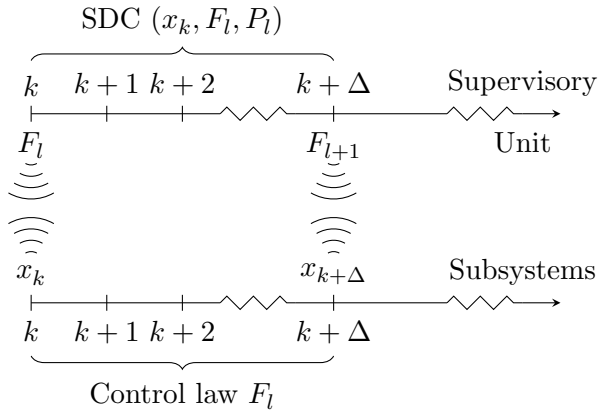

Figure 5.5: Supervised-distributed controller time line.

the current control mode as well as a prediction of the system state $\overline{x}_{k+\Delta|k}$, after Δ time steps. The next control law is obtained when one of the convergence stopping criteria is met. Although, the optimization can be terminated if the alternate convex search has not converged within the allocated time, yielding a feasible but suboptimal solution as proved in Corollary 5.1. Once the next control mode F_{l+1} is obtained, all the gains relying on a communication channel are set to zero if their values are below a certain threshold. Then, the control mode with definite zeros is tested against the optimization constraints for feasibility and sent to the subsystems if it passes the test, otherwise the gain computed originally is returned. The extra threshold step has to be completed due to the fact that the l_1-norm is used as a convex relaxation of the l_0-norm. Consequently, the entries of the feedback gain will be decreased to a small value but will not be set to a definite zero value. Another optimization strategy developed in the past in order to induce sparsity and that could be implemented in Algorithm 5.1 is to update the weights used for the l_1-norm of each gain, this framework is known as reweighted l_1-norm minimization (Candès et al., 2008). The final step is to broadcast the solution to the subsystems so that the control law F_{l+1} can be implemented for the following Δ time steps. Then, this process is applied recursively in a receding horizon fashion.

Corollary 5.1. *A feasible asymptotically stabilizable solution of the optimization problem* (5.24) *can be obtained by terminating Algorithm 5.1 after any iteration.*

Proof. According to Theorem 5.2, the optimization problem (5.24) stays recursively feasible and a suboptimal solution $(F_l,\, X_l)$ is known at every time step from the previous optimization solution. Subsequently, for all $q \in \mathbb{N}$, Algorithm 5.1 uses a feasible solution $(F_q,\, X_q)$ in order to solve alternately the two following convex semi-definite programming problems $\mathcal{P}_{\overline{x}_{k+\Delta|k},F_q}(\rho, X)$ which then generates $(F_q,\, X_{q+1})$

and $\mathcal{P}_{\overline{x}_{k+\Delta|k},X_{q+1}}(\rho, F)$ which generates (F_{q+1}, X_{q+1}). Hence, by induction the Algorithm 5.1 produces a new feasible solution after every iteration which concludes the proof. □

The periodical online computation of structured control modes uses ellipsoid invariant sets based on the Lyapunov solutions obtained, this ensures that the predicted state variable remains within an ellipsoid set. According to Theorem 5.2, the Lyapunov solution obtained at mode l remains a feasible solution for mode $l+1$, hence the initial ellipsoid set remains an invariant set for the following control modes. Therefore, using invariant set theory it is possible to enforce sufficient constraint conditions on the input, the output and the state variables for the future modes (Boyd et al., 1994). Because the control modes rely on the previous modes computed, it can be proved by induction that only the first optimization problem has to be feasible for the set of system constraints. The next subsection presents a way to introduce sufficient conditions in order for the control modes to comply with given physical system constraints, on the state, the input and the output variables.

5.5.2 Constrained Supervised-distributed Controller

Without changing the results presented previously within this chapter, it is possible to introduce constraints on the input, the output as well as the state variables in the optimization problem (5.24). These variables can be constrained in two different ways, based on sufficient LMI conditions. Firstly, their Euclidean norm can be bounded and secondly the maximum magnitude of each entry of these variables can be limited. These techniques have been studied previously and can be added to the supervised-distributed controller technique as presented within this subsection (Kothare et al., 1996). This subsection considers the system (5.29), this new system definition is similar to the system (5.1) and still complies with the Assumption 5.1 but differs from the initial system by the representation of the output variable.

$$x_+ = Ax + Bu \tag{5.29a}$$

$$y = Cx, \tag{5.29b}$$

where $y \in \mathbb{R}^p$ is the output variable, and C is the output matrix of appropriate dimension.

Inputs Constraints

An extra LMI constraint can be added in order to ensure that the Euclidean norm of the input variable remains within some boundaries (Boyd et al., 1994). This condition is a sufficient constraint and is expressed as follows,

$$\begin{bmatrix} u_{max}^2 I_m & K \\ K^\top & X \end{bmatrix} \succeq 0. \tag{5.30}$$

Adding the constraint (5.30) to the optimization problem (5.24) is a sufficient condition to provide the following peak constraint on the Euclidean norm of u_k,

$$\forall k \in \mathbb{N},\ \|u_k\|_2 \leq u_{max}, \tag{5.31}$$

where u_k is the system input variable at time step k. This result is proved below in Lemma 5.3.

Lemma 5.3. *Adding the constraint* (5.30) *to the optimization problem* (5.24) *ensures that the Euclidean norm of the input variable is bounded by* $u_{max} \in \mathbb{R}_+^*$.

Proof. According to Theorem 5.1, the system state variable x_k belongs to the invariant ellipsoid $\mathcal{E}$ defined such that $\mathcal{E} = \left\{ x \in \mathbb{R}^n \,\middle|\, x^\top X^{-1} x \leq 1,\ X \in \mathbb{S}_{++}^n \right\}$ for all $k \in \mathbb{N}$, consequently,

$$\max_{k\in\mathbb{N}} \|u_k\|_2^2 = \max_{k\in\mathbb{N}} \|KX^{-1}x_k\|_2^2 \tag{5.32a}$$

$$\leq \max_{x\in\mathcal{E}} \|KX^{-1}x\|_2^2 \tag{5.32b}$$

$$= \lambda_{max}\left(X^{-\frac{1}{2}} K^\top K X^{-\frac{1}{2}}\right). \tag{5.32c}$$

Hence, applying the Schur complement to (5.30) shows that there exists a boundary on the value $\lambda_{max}\left(X^{-\frac{1}{2}} K^\top K X^{-\frac{1}{2}}\right)$ that is u_{max}^2, therefore concluding the proof. □

The case presented previously can be extended in order to implement a magnitude boundary on all the entries of the input variable. This is achieved by adding the linear matrix inequality given in (5.33).

$$\begin{bmatrix} U^{max} & K \\ K^\top & X \end{bmatrix} \succeq 0 \tag{5.33}$$

where, for all $j \in [\![1, m]\!]$, $U_{jj}^{max} \leq u_{j,max}^2$. This constraint enforces the following

magnitude constraint on all the entries of u_k,

$$\forall k \in \mathbb{N},\, \forall j \in [\![1, m]\!],\, \left|u_{j,k}\right| \leq u_{j,max} \tag{5.34}$$

where, $u_{j,k}$ is the j-th component of the input vector at time step $k \in \mathbb{N}$.

Lemma 5.4. *Adding the constraint* (5.33) *to the optimization problem* (5.24) *ensures that the magnitude of the j-th entry of the input variable is bounded by $u_{j,max}$.*

Proof. The system state variable x_k belongs to the invariant ellipsoid $\mathcal{E}$ defined such that $\mathcal{E} = \left\{ x \in \mathbb{R}^n \,\middle|\, x^\top X^{-1} x \leq 1,\, X \in \mathbb{S}^n_{++} \right\}$ for all $k \in \mathbb{N}$, subsequently,

$$\begin{aligned}
\max_{k\in\mathbb{N}} \left|u_{j,k}\right|^2 &= \max_{k\in\mathbb{N}} \left|(KX^{-1}x_k)_j\right|^2 && \text{(5.35a)}\\
&\leq \max_{x\in\mathcal{E}} \left|(KX^{-1}x)_j\right|^2 && \text{(5.35b)}\\
&\leq \left\|(KX^{-\frac{1}{2}})_j\right\|_2^2 && \text{(5.35c)}\\
&= \left(KX^{-1}K^\top\right)_{jj}, && \text{(5.35d)}
\end{aligned}$$

where the subscript $j \in [\![1, m]\!]$ denotes the j-th entry or row when applied to a vector or a matrix respectively. The inequality between (5.35b) and (5.35c) is obtain by using the Cauchy-Schwartz inequality. Consequently, the existence of a symmetric matrix U^{max}, such that for all $j \in [\![1, m]\!]$, $U^{max}_{jj} \leq u^2_{j,max}$ ensures that the magnitude of $u_{j,k}$ is bounded by $u_{j,max}$. □

Outputs Constraints

In a similar fashion, sufficient LMI conditions can be used to implement a bound on the Euclidean norm of the output variable, this constraint is presented in (5.36).

$$\begin{bmatrix} X & (AX + BK)^\top C^\top \\ C(AX + BK) & y^2_{max} I_p \end{bmatrix} \succeq 0 \tag{5.36}$$

where C represents the system output matrix. Similarly, by using only the appropriate row of C sufficient conditions can be established to bound the magnitude of a single entry of the output variable. Hence, constraints can be implemented on the vector Euclidean norm as well as on the peak magnitude of the entries of the output vector such that,

$$\forall k \in \mathbb{N}^*, \|y_k\|_2 \leq y_{max}, \tag{5.37a}$$

$$\forall k \in \mathbb{N}^*,\, \forall j \in [\![1, p]\!], |y_{j,k}| \leq y_{j,max} \tag{5.37b}$$

where, $y_{j,k}$ is the j-th component of the output vector at time step $k \in \mathbb{N}$.

Lemma 5.5. *Adding the constraint* (5.36) *to the optimization problem* (5.24) *ensures that the Euclidean norm of the output variable is bounded by* $y_{max} \in \mathbb{R}_+^*$.

Proof. Since the system state variable x_k belongs to the invariant ellipsoid $\mathcal{E}$ defined such that $\mathcal{E} = \left\{ x \in \mathbb{R}^n \,\middle|\, x^\top X^{-1} x \leq 1,\ X \in \mathbb{S}_{++}^n \right\}$ for all $k \in \mathbb{N}$, subsequently,

$$\max_{k \in \mathbb{N}^*} \left\| y_k \right\|_2 = \max_{k \in \mathbb{N}} \left\| C(A + BF) x_k \right\|_2 \tag{5.38a}$$

$$\leq \max_{x \in \mathcal{E}} \left\| C(A + BF) x_k \right\|_2 \tag{5.38b}$$

$$= \sigma_{max} \left(C(A + BF) X^{\frac{1}{2}} \right). \tag{5.38c}$$

Therefore, the condition $\left\| y_k \right\|_2 \leq y_{max}$ is verified if the maximum singular value of the matrix $C(A + BF)X^{\frac{1}{2}}$ is less or equal to y_{max}. This condition can be turned into an LMI condition as follows,

$$\sigma_{max} \left(C(A + BF) X^{\frac{1}{2}} \right) \leq y_{max} \tag{5.39a}$$

$$\Leftrightarrow X^{\frac{1}{2}} (A + BF)^\top C^\top C (A + BF) X^{\frac{1}{2}} \preceq y_{max}^2 I_n \tag{5.39b}$$

$$\Leftrightarrow \begin{bmatrix} X & (AX + BK)^\top C^\top \\ C(AX + BK) & y_{max}^2 I_p \end{bmatrix} \succeq 0 \tag{5.39c}$$

The equivalent condition is obtained by pre- and post-multiplying (5.39b) by $X^{\frac{1}{2}}$ and by forming the Schur complement, which leads to the equation (5.39c) and concludes the proof. □

States Constraints

In the same way as before, replacing the output matrix C by an appropriate matrix or vector in the constraint (5.36) developed before will allow to constrain the Euclidean norm of the state variable as well as the magnitude of its entries. Limiting the Euclidean norm of the state variable by $x_{max} \in \mathbb{R}_+^*$ is achieved by replacing the matrix C by the identity matrix I_n.

$$\forall k \in \mathbb{N}^*,\ \|x_k\|_2 \leq x_{max} \tag{5.40}$$

Finally, replacing the matrix C with the transpose of the canonical unit vectors $e_i \in \mathbb{R}^n$ allows to enforce constraints on the magnitude of the i-th entry of the state variable as presented in equation (5.41).

$$\forall k \in \mathbb{N}^*,\ \forall i \in [\![1, n]\!],\ |x_{i,k}| \leq x_{i,max} \tag{5.41}$$

The supervised-distributed control technique can be made robust to system model uncertainty by adding more constraints to the optimization problem developed previously. The next subsection presents how to include the model uncertainty within the supervised-distributed control algorithm.

5.5.3 Supervised-distributed Controller with Model Uncertainty

The exact state space model of a system is rarely known perfectly, however the dynamic model is often known to evolve within some known boundaries. This can be due to two main paradigms. First of all, it can come from the fact that identifying the exact linear system model is complex, or it can be due to the fact that the system is not linear but can still be modelled by a set of linear state space models. Control techniques robust to model uncertainty have been developed in the past to answer the model uncertainty paradigms and to provide an optimal control to such systems (Kothare et al., 1996; Zhou et al., 1996). These techniques minimize the upper bound on the robust performance cost, which is equivalent to minimizing the cost in the worst case scenario. This section considers system model uncertainty that can be represented by a set of linear state space models, where the actual state space model possibly time-varying is known to belong.

$$\forall k \in \mathbb{N},\ \Big[A_k \,|\, B_k\Big] \in \Omega, \tag{5.42}$$

where A_k and B_k are respectively the state and input matrices at time step k. Only polytopic model uncertainty are considered within this section, therefore, the set Ω is a polytopic set defined by a finite number of vertices. An example of such a set is the set represented Figure 5.6 and defined by five distinct vertices. In this example each vertex is a discrete time state space model whose value can be achieved by the actual system dynamics.

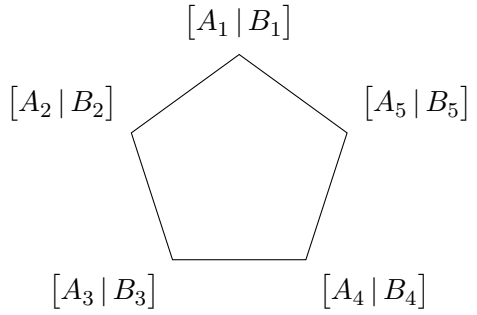

Figure 5.6: Representation of a polytopic uncertainty set with five vertices.

The polytopic uncertainty set Ω defined by $s \in \mathbb{N}^*$ vertices is the convex hull of

the set of vertices and is defined as follows,

$$\Omega = \operatorname{co}\left\{ \left[A_i \,|\, B_i \right] \,\middle|\, i \in [\![1, s]\!] \right\}. \tag{5.43}$$

The optimization problem (5.24) can be made robust to polytopic model uncertainty by replacing its constraints by the set of constraints provided in the optimization problem (5.44). It can be noted that in the case where the number of vertices s is equal to one, the linear time-invariant case (5.24) is recovered with Δ equal to one.

$$\begin{aligned}
&\underset{\rho,F,X,K}{\text{minimize}} \quad \lambda\rho + (1-\lambda)\left\| \operatorname{vec}(W \circ F) \right\|_1 \\
&\text{subject to} \quad \forall i \in [\![1, s]\!], \\
&\qquad \begin{bmatrix} X & XA_i^\top + K^\top B_i^\top & XQ^{\frac{1}{2}} & K^\top R^{\frac{1}{2}} \\ A_iX + B_iK & X & 0 & 0 \\ Q^{\frac{1}{2}}X & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}}K & 0 & 0 & \rho I_m \end{bmatrix} \succeq 0 \\
&\qquad \begin{bmatrix} \rho P_l & \rho(A_i + B_iF_l)^\top & \rho\varepsilon I_n \\ \rho(A_i + B_iF_l) & X & 0 \\ \rho\varepsilon I_n & 0 & \rho\varepsilon I_n \end{bmatrix} \succeq 0 \\
&\qquad \begin{bmatrix} 1 & x_0^\top \\ x_0 & X \end{bmatrix} \succeq 0 \\
&\qquad FX = K
\end{aligned} \tag{5.44}$$

Theorem 5.5. *If there exists an initial robust asymptotically stabilizing control law for the system* (5.1) *with polytopic model uncertainty* (5.42)*, then the optimization problem* (5.44) *is recursively feasible and its consecutive solutions are globally asymptotically stabilizing control modes for the switched closed-loop system robust to polytopic model uncertainty.*

Proof. Since all the LMI constraints indexed by $i \in [\![1, s]\!]$ are satisfied, the following

relations hold,

$$\forall i \in [\![1,s]\!],\ \begin{bmatrix} X & XA_i^\top + K^\top B_i^\top & XQ^{\frac{1}{2}} & K^\top R^{\frac{1}{2}} \\ A_iX + B_iK & X & 0 & 0 \\ Q^{\frac{1}{2}}X & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}}K & 0 & 0 & \rho I_m \end{bmatrix} \succeq 0 \tag{5.45a}$$

$$\Rightarrow \forall k \in \mathbb{N},\ \sum_{i=1}^{s} \theta_{i,k} \begin{bmatrix} X & XA_i^\top + K^\top B_i^\top & XQ^{\frac{1}{2}} & K^\top R^{\frac{1}{2}} \\ A_iX + B_iK & X & 0 & 0 \\ Q^{\frac{1}{2}}X & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}}K & 0 & 0 & \rho I_m \end{bmatrix} \succeq 0 \tag{5.45b}$$

$$\Rightarrow \forall k \in \mathbb{N},\ \begin{bmatrix} X & XA_k^\top + K^\top B_k^\top & XQ^{\frac{1}{2}} & K^\top R^{\frac{1}{2}} \\ A_kX + B_kK & X & 0 & 0 \\ Q^{\frac{1}{2}}X & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}}K & 0 & 0 & \rho I_m \end{bmatrix} \succeq 0, \tag{5.45c}$$

where $\theta_{i,k}$ are positive numbers at time step k summing to one, such that the model of the system at any time step k is given by the equation (5.46).

$$\forall k \in \mathbb{N},\ \Big[A_k \,|\, B_k\Big] = \sum_{i=1}^{s} \theta_{i,k} \Big[A_i \,|\, B_i\Big] \tag{5.46}$$

In a similar way, multiplying the LMI dwell time constraints by $\theta_{i,k}$ and summing them together yields the following relation,

$$\forall i \in [\![1,s]\!],\ \begin{bmatrix} \rho P_l & \rho(A_i + B_iF_l)^\top & \rho\varepsilon I_n \\ \rho(A_i + B_iF_l) & X & 0 \\ \rho\varepsilon I_n & 0 & \rho\varepsilon I_n \end{bmatrix} \succeq 0 \tag{5.47a}$$

$$\Rightarrow \forall k \in \mathbb{N},\ \begin{bmatrix} \rho P_l & \rho(A_k + B_kF_l)^\top & \rho\varepsilon I_n \\ \rho(A_k + B_kF_l) & X & 0 \\ \rho\varepsilon I_n & 0 & \rho\varepsilon I_n \end{bmatrix} \succeq 0. \tag{5.47b}$$

Therefore it ensures that the control law is robust to polytopic model uncertainty as well as asymptotically stabilizing for the system under switching, thus concluding the proof. □

Since a convex combination of the dwell time constraints is used in order to guarantee that the robust control modes will be asymptotically stable under switching, the dwell time parameter Δ has to be set to one. Nonetheless, as it has been proved previously in Proposition 5.1, the control mode refreshing rate can be higher

than the dwell time parameter in order to allocate more time to perform the optimization. This feature can be useful since the optimization problem with model uncertainty includes more LMI constraints and therefore requires more time to be solved. Finally, the model uncertainty robustness and the system constraints could be combined within the optimization problem (5.24) without changing the results established previously. The next section provides an analysis of the complexity of the two convex optimization problems solved alternately in the unconstrained case of Algorithm 5.1.

5.6 Supervised-distributed Algorithm Complexity

The optimal supervised-distributed control algorithm solved online is computed by solving sequentially two distinct SDP problems. Some work has been done previously on the complexity of solving LMI problems (Gahinet et al., 1995). An upper bound on the number of Floating Point Operations Per Second (FLOPS) and therefore on the algorithm complexity, in order to compute a solution with an ε_0 accuracy can be evaluated. This bound is based on the size of the optimization problem (number of variables and number of constraints). The Algorithm 5.1 sequentially solves the problem (5.24) with modified LMI constraints. The first problem to be solved is $\mathcal{P}_{\overline{x}_{k+\Delta|k},F_q}(\rho, X)$ when the variable F_q is fixed such that,

$$
\begin{aligned}
\underset{\rho,X}{\text{minimize}} \quad & \lambda\rho + (1-\lambda)\left\| \operatorname{vec}(W \circ F_q)\right\|_1 \\
\text{subject to} \quad & \begin{bmatrix} X & X(A+F_qB)^\top & XQ^{\frac{1}{2}} & XF_q^\top R^{\frac{1}{2}} \\ (A+BF_q)X & X & 0 & 0 \\ Q^{\frac{1}{2}}X & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}}F_qX & 0 & 0 & \rho I_m \end{bmatrix} \succeq 0 \\
& \begin{bmatrix} \rho P_l & \rho(A+BF_l)^\top & \varepsilon\rho I_n \\ \rho(A+BF_l) & X & 0 \\ \varepsilon\rho I_n & 0 & \varepsilon\rho I_n \end{bmatrix} \succeq 0 \\
& \begin{bmatrix} 1 & \overline{x}_{k+\Delta|k}^\top \\ \overline{x}_{k+\Delta|k} & X \end{bmatrix} \succeq 0.
\end{aligned}
\tag{5.48}
$$

The second optimization problem of the alternate convex search when the variable X_q is fixed is $\mathcal{P}_{\overline{x}_{k+\Delta|k},X_q}(\rho, F)$ as follows,

$$\begin{aligned}
&\underset{\rho,F}{\text{minimize}} \quad \lambda\rho + (1-\lambda)\big\| \operatorname{vec}(W \circ F)\big\|_1 \\
&\text{subject to} \begin{bmatrix} X_q & X_q(A+BF)^\top & X_q Q^{\frac{1}{2}} & X_q F^\top R^{\frac{1}{2}} \\ (A+BF)X_q & X_q & 0 & 0 \\ Q^{\frac{1}{2}} X_q & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}} F X_q & 0 & 0 & \rho I_m \end{bmatrix} \succeq 0 \\
&\qquad\qquad \begin{bmatrix} \rho P_l & \rho(A+BF_l)^{\Delta\top} & \varepsilon\rho I_n \\ \rho(A+BF_l)^{\Delta} & X_q & 0 \\ \varepsilon\rho I_n & 0 & \varepsilon\rho I_n \end{bmatrix} \succeq 0.
\end{aligned} \tag{5.49}$$

The first SDP optimization problem solved has got the LMI constraints (5.48) with $7n + m + 1$ rows and $\frac{n(n+1)}{2} + 1$ decision variables. Therefore, the algorithmic complexity in FLOPS is less or equal to,

$$(7n + m + 1)\left(\frac{n(n+1)}{2} + 1\right)^3 \log\left(\frac{V}{\varepsilon_0}\right), \tag{5.50}$$

with V a data-dependent scaling factor. The second SDP optimization has the LMI constraint (5.49) with $6n + m$ rows and $nm + 1$ decision variables. Subsequently, the complexity in FLOPS is bounded by,

$$(6n + m)\Big(nm + 1\Big)^3 \log\left(\frac{V}{\varepsilon_0}\right). \tag{5.51}$$

The complexity of both problems solved by Algorithm 5.1 is polynomial, because SDP problems belong to convex programming problems and thus can be efficiently solved with interior-point methods (Nesterov and Nemirovskii, 1994). The algorithmic complexity of the two problems solved during the alternate convex search are $\mathcal{O}(n^6(n+m))$ and $\mathcal{O}(n^3m^3(n+m))$ respectively. The scalable complexity combined with the possibility to terminate the algorithm after a certain number of iterations implies that such a control technique is very tractable and can be implemented online. For instance, for a system model composed of $n = 10$ state variables and $m = 10$ input variables, the number of FLOPS required is of the order of magnitude of the giga-FLOPS. As a comparison, a standard desktop computer has a computational power of the order of magnitude tens to thousands of giga-FLOPS. The next section presents some numerical examples in order to compare the supervised-distributed control algorithm to other control methods as well as to show the trade-off between control system performance and subsystem to subsystem communication.

5.7 Numerical Examples

Two examples have been used to demonstrate the trade-off offered by the optimal distributed state feedback control. These numerical examples have been solved using YALMIP (Löfberg, 2004) along with the SeDuMi (Sturm, 1999) and Mosek (MOSEK ApS, 2017) SDP solvers.

5.7.1 Small-scale System

The first example used is a simple second order plant, where the interaction comes from both the state and the input matrices. The example consists of a simple second-order plant (5.52), where the subsystem interaction comes from both the state matrix and the input matrix. All the simulations start from the initial state $x_0^\top = \begin{bmatrix} 1 & -1 \end{bmatrix}$ and include multiple additive disturbances added at time steps 24, 26, 32 and 34 only affecting the state variable x_2.

$$x^+ = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} x + \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix} u \tag{5.52}$$

The weighting matrices Q and R used to evaluate the control performance are both taken equal to identity. The system is partitioned into two subsystems each including the state and input having the same index. In order to compare the performance of the distributed controller, two benchmark controllers are designed F_C and F_D respectively the centralized and the decentralized linear feedback controllers.

$$F_C = \begin{bmatrix} -0.8709 & -0.6210 \\ 0.6069 & -0.5330 \end{bmatrix} \tag{5.53a}$$

$$F_D = \begin{bmatrix} -0.7551 & 0 \\ 0 & -0.9078 \end{bmatrix} \tag{5.53b}$$

The spectral radius of the closed-loop plants with these two controllers are respectively equal to 0.4012 and 0.8803.

The Figure 5.7 shows the behaviour of the plant when the supervised-distributed controller is implemented. The supervisory unit computes a trade-off between the communication and the system performance infinite horizon cost. In this example, a dwell time parameter of five has been implemented, the value of lambda has been set to 0.99 and the maximum number of iterations has been limited to ten. Therefore, every five time steps, a new control law is computed and broadcast to the local controllers in order to be implemented for the next five time steps. The supervisory unit is initialized with a centralized controller and as soon as the system reaches

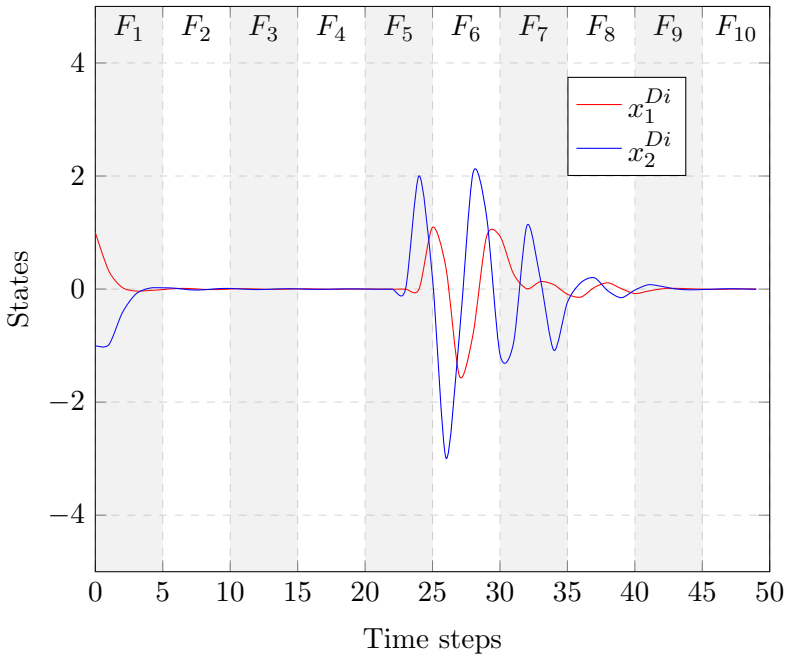

Figure 5.7: Supervised-distributed controller applied to the small-scale plant (5.52).

its steady state the supervisory unit outputs a decentralized state feedback gain. However, if some disturbance pushes the system out of its steady state value, the cost balance changes, hence the next control laws computed rely on communication again and could be even fully centralized. As it can be seen form Table 5.2, the structure of the control law is optimized in real time and allows to rely on communication only to improve the disturbance rejection performance. Because of the precision of the SDP solver, the value of the off-diagonal elements is not strictly zero, therefore some threshold has to be applied before a communication channel can be completely switched off.

Figure 5.8 presents the response to the second order system when controlled with the centralized state feedback controller F_C. This control law relies on all the available subsystem to subsystem communication, however it offers the best system performance and disturbance rejection capabilities as it can be seen from Figure 5.8 as well as from Table 5.3.

Finally, Figure 5.9 shows the response of the system when the decentralized control law F_D is implemented. This automatic control system does not rely on communication but offers the worst performance and disturbance rejection capabilities when compared to the two other system behaviours.

Figure 5.10 presents the trade-off between the communication metric and the control system performance for the small-scale system (5.52). This Pareto front has

Table 5.2: State feedback gains computed by the supervised-distributed controller in Figure 5.7.

State feedback gains				
$\text{vec}(F_1)^\top$	$[-0.8708$	$+0.6069$	-0.6210	$-0.5330]$
$\text{vec}(F_2)^\top$	$[-0.8083$	$+0.0374$	$+0.0000$	$-0.9484]$
$\text{vec}(F_3)^\top$	$[-0.7551$	$+0.0000$	$+0.0000$	$-0.9078]$
$\text{vec}(F_4)^\top$	$[-0.7551$	$+0.0000$	$+0.0000$	$-0.9078]$
$\text{vec}(F_5)^\top$	$[-0.7551$	$+0.0000$	$+0.0000$	$-0.9078]$
$\text{vec}(F_6)^\top$	$[-0.7551$	$+0.0000$	$+0.0000$	$-0.9078]$
$\text{vec}(F_7)^\top$	$[-0.8687$	$+0.6015$	-0.6170	$-0.5348]$
$\text{vec}(F_8)^\top$	$[-0.8108$	$+0.0397$	-0.0008	$-0.9497]$
$\text{vec}(F_9)^\top$	$[-0.8028$	$+0.3953$	$+0.4510$	$-0.6194]$
$\text{vec}(F_{10})^\top$	$[-0.7551$	$+0.0000$	$+0.0000$	$-0.9078]$

Table 5.3: Comparison of the cumulative control and communication costs for different control methods applied to the system (5.52).

Costs	**C-LQR**	**SDC**	**D-LQR**
Control	23.1	58.8	110.9
Communication	61.4	16.8	0
Total	**84.5**	**75.6**	**110.9**

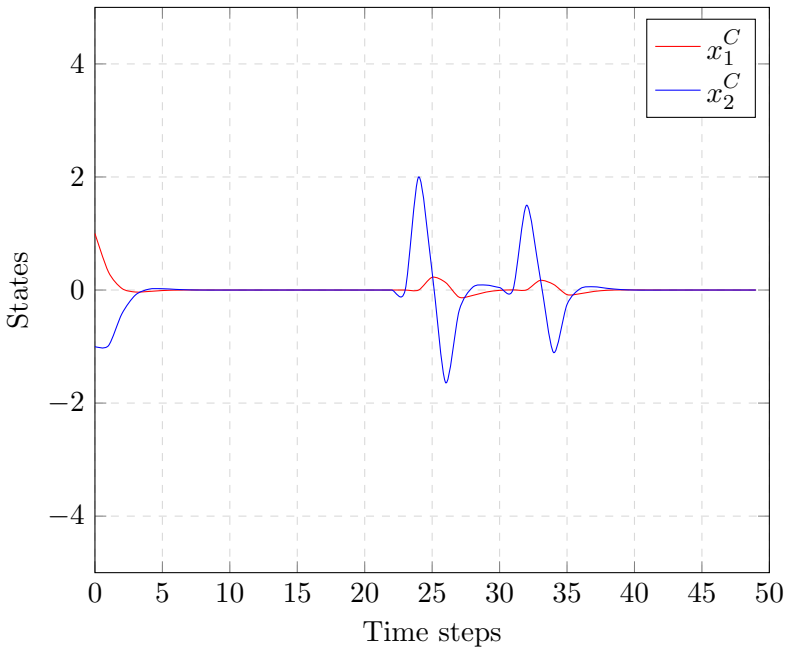

Figure 5.8: Centralized controller applied to the small-scale plant (5.52).

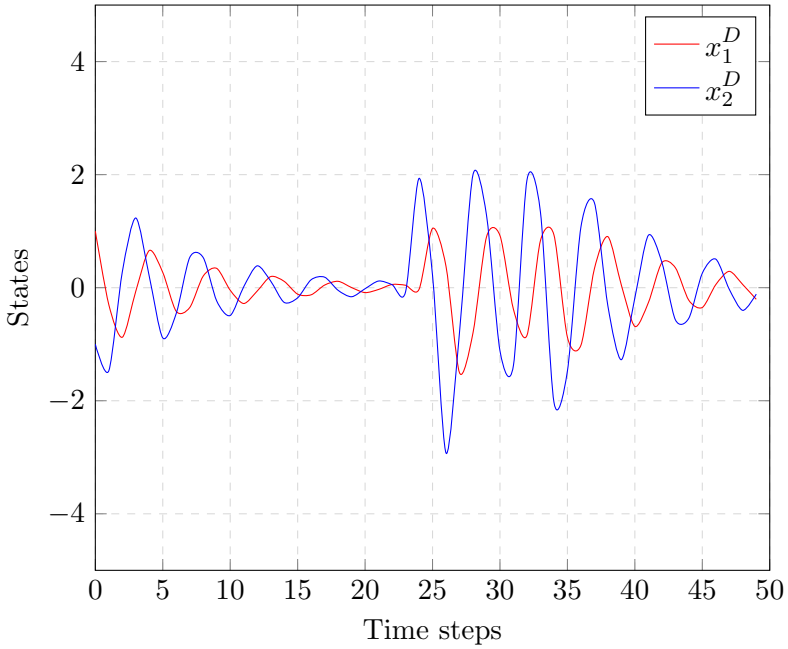

Figure 5.9: Decentralized controller applied to the small-scale plant (5.52).

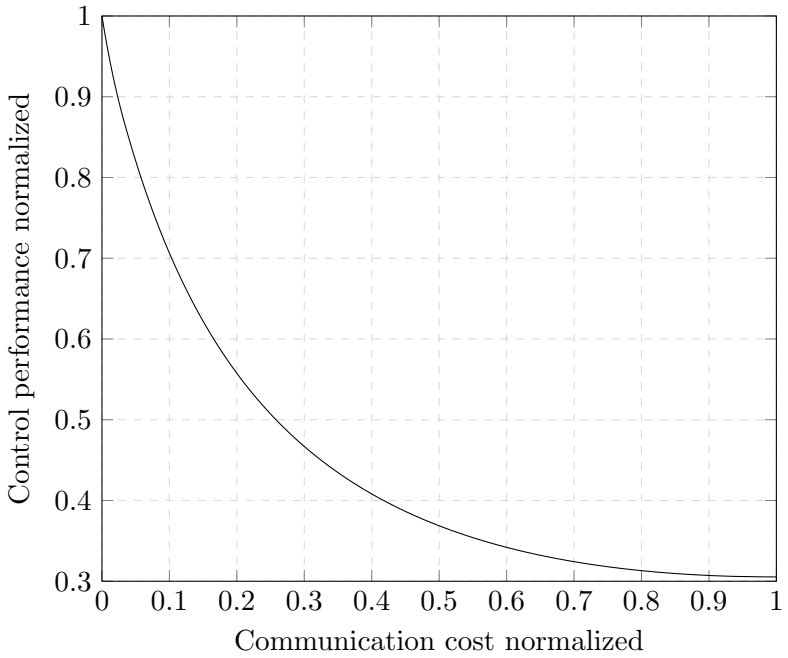

Figure 5.10: Trade-off between control performance and communication cost for the system (5.52).

been obtained by modifying the value of lambda from 0.85 to 1 within Algorithm 5.1 for a given fixed initial state. It can be noticed that because the two subsystems composing the original subsystems are coupled, different level of communication within the control law affect the performance of the system greatly. A performance cost decrease of almost 70% between the fully decentralized and the fully centralized control architectures is observed in this particular example.

5.7.2 Medium-scale System

The medium-scale example used here is the Pratt & Whitney F100 after-burning turbofan engine (Jaw and Mattingly, 2009), discretized with a sampling time of $0.1s$. This example has been chosen to show that there is a trade-off between the amount of communication between the subsystems and the system wide performance even when the subsystems are weakly coupled. This gas turbine system is composed of five state variables as well as five input variables partitioned according to the weak interactions technique given in Chapter 4 (Guicherd et al., 2017). The state variables respectively represent the fan speed, the compressor speed, the afterburner pressure, the main burner fuel metering valve position and the compressor discharge pressure. The five input variables of the gas turbine are the main burner fuel flow, the nozzle jet area, the compressor inlet guide vane position, the high variable stator position and the custom compressor bleed flow. The full state space model of the gas turbine engine is given in equation (5.54). The Figure 5.11 shows the trade-off between the normalized control performance and the normalized infinite horizon control cost, the normalized communication performance is given by the l_1-norm of the control law communication elements.

$$A = \begin{bmatrix} -0.3245 \times 10^{1} & -0.2158 \times 10^{1} & -0.9155 \times 10^{3} & 0.5731 \times 10^{0} & 0.1342 \times 10^{3} \\ 0.1642 \times 10^{1} & -0.5941 \times 10^{1} & -0.2816 \times 10^{3} & 0.1897 \times 10^{0} & 0.5705 \times 10^{2} \\ 0.1685 \times 10^{-1} & -0.2554 \times 10^{-1} & -0.1003 \times 10^{2} & 0.7994 \times 10^{-2} & 0.5807 \times 10^{0} \\ 0 & 0 & 0 & -0.1 \times 10^{2} & 0 \\ -0.2163 \times 10^{1} & 0.6862 \times 10^{1} & 0.7405 \times 10^{3} & 0.1195 \times 10^{1} & -0.1715 \times 10^{3} \end{bmatrix} \tag{5.54a}$$

$$B = \begin{bmatrix} 0.1432 \times 10^{-1} & -0.3553 \times 10^{3} & -0.9906 \times 10^{2} & -0.1549 \times 10^{2} & 0.222 \times 10^{5} \\ 0.2871 \times 10^{0} & 0.7286 \times 10^{3} & 0.2514 \times 10^{2} & -0.6487 \times 10^{2} & 0.8122 \times 10^{4} \\ -0.2469 \times 10^{-2} & -0.103 \times 10^{3} & 0.6333 \times 10^{0} & -0.3213 \times 10^{0} & -0.7418 \times 10^{2} \\ 0.1 \times 10^{2} & 0 & 0 & 0 & 0 \\ -0.1311 \times 10^{0} & 0.3295 \times 10^{3} & -0.25 \times 10^{2} & 0.6257 \times 10^{2} & -0.6445 \times 10^{5} \end{bmatrix} \tag{5.54b}$$

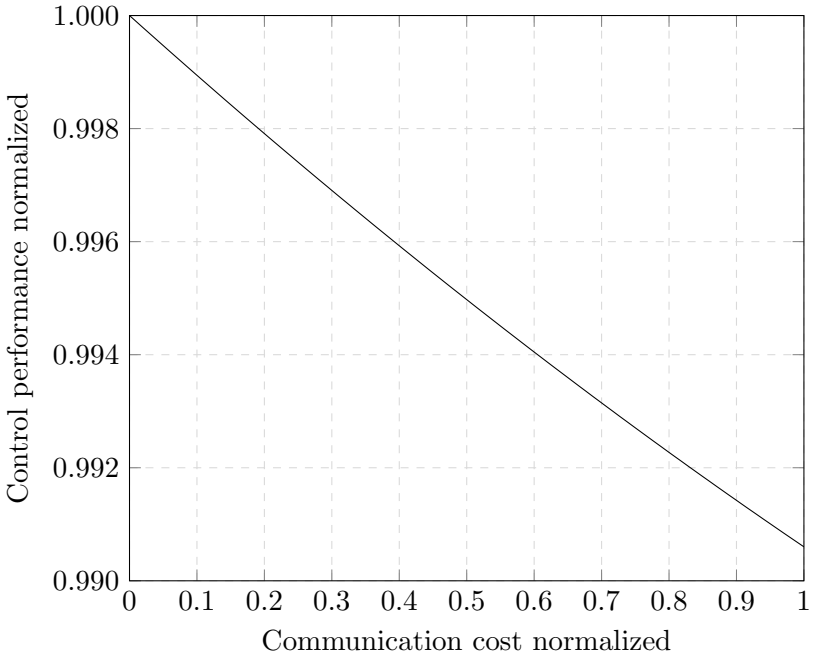

Figure 5.11: Trade-off between control performance and communication cost for the Pratt & Whitney F100 (5.54).

The sparsity within the control law is changed linearly from fully centralized to completely decentralized. Figure 5.11 shows the trade-off offered by the different sparsity levels in the control feedback gain and it can be seen that the control cost decreases by about 1% from the decentralized control architecture to the centralized control architecture. Consequently, the centralized and the decentralized systems perform very similarly. This is due to the fact that the subsystems were obtained by weak partitioning of the original system, thus, the need for communication is decreased for this particular architecture.

5.8 Conclusion

A supervised-distributed model-based control scheme has been presented within this chapter. It has been shown that this problem can be formulated as a biconvex optimization problem cast as a semi-definite programming problem including a bilinear matrix equality. Therefore, solvable using an alternate convex search technique where two convex programming problems are solved alternately until a stopping criteria is achieved. Also, the particular structure of the optimization problem allows to guarantee convergence of the optimal supervised-distributed control algorithm. In addition to this property, the supervised-distributed control technique is proved to be globally asymptotically stabilizing for the closed-loop system under switching

based on dwell time requirements. The dwell time parameter not only ensures the stability of the switched closed-loop controlled system but also provides enough time to perform the optimization online. In addition to this, the recursive feasibility of the optimization problem has been established, and it has been demonstrated that a stable feasible control mode could be obtained even after early termination of the algorithm. Finally, some extra constraints developed and used in the literature in the past can be added to the optimization problem in order to tackle physical system constraints as well as system model uncertainty, making the distributed control system robust although more conservative. The problem of jointly optimizing the communication and system performance is a complex non-convex problem. Therefore, because the optimization problem is not convex, there cannot be any guarantee that the solutions obtained are globally optimal. However, even local optima have been shown to perform better that both the centralized and the decentralized controllers in numerical simulations. Finally, it has been shown numerically that there exists a trade-off between the amount of subsystem to subsystem communication and the system performance when the subsystems are coupled. Future research directions regarding the joint communication and performance optimization should be looking at optimally selecting the designer tuning parameters λ and Δ. An investigation dealing with how to modify these two parameters online would also bring some improvements to the supervised-distributed control technique.

Chapter 6

Distributed Control for Linear Parameter-varying Systems with Joint Performance and Communication Optimization

6.1 Introduction

Physical systems are inherently non-linear, therefore performing system modelling as well as designing a controller can be very complex in the non-linear case. On the other hand, linear dynamical system models have a lot of useful properties but are applicable only over a small operating range (Bay, 1998). Subsequently, non-linear systems are linearized at multiple operating points, and the linear models obtained are patched together based on the value of a scheduling parameter (Rugh and Shamma, 2000), the new model obtained is named Linear Parameter-varying (LPV). Following this, controllers are designed for each linearization point and then blended together based on a technique similar to the one used for the linear models (Lawrence and Rugh, 1995). This divide and conquer strategy is a well known control technique named gain-scheduling. It has become one of the standard practical strategies to perform non-linear control design, this technique has arisen and then has been applied widely in the aerospace sector (Leith and Leithead, 2000; Balas, 2002; Gilbert et al., 2010). The design of gain scheduling feedback control laws relies on a LPV system model, and therefore enables the use of well established linear design techniques in the realm of non-linear systems. In other words, a non-linear system is approximated by a set of linear models varying with regards to an exogenous scheduling parameter, not controllable, however measurable at every

time step. Subsequently, the model of the system is known to evolve in a compact set with established boundaries. Hence, one of the control technique applicable to LPV systems would be to design a static state feedback gain ensuring robustness to plant model uncertainties over the entire compact set of dynamical models (Kothare et al., 1996). This technique can be quite conservative when the system model is known to vary within a large dynamical range. This is due to the fact that one control law has to be applicable over the entire compact set of system dynamics, and in some cases, such a robust controller does not even exist. Another approach is the gain scheduling technique, as mentioned previously, it relies on a gain that will vary based on the value of the exogenous scheduling parameter, in order to adapt to the varying system dynamics over the entire achievable set of dynamics (Wada et al., 2006; Emedi and Karimi, 2016). In the case of distributed systems, the design of structured LPV controllers can be performed to optimize the system-wide performance as well as the communication burden.

Designing a structured LPV controller has been considered in the decentralized case when applied to power systems (Qiu et al., 2004). Other research approaches have considered the design of structured LPV controllers for continuous and discrete time systems (Veselý et al., 2013). The main approach used in the literature is to design a family of control laws without any structure, on local sets of dynamics in order to improve the overall system performance (Azadi Yazdi and Nagamune, 2011; Hanifzadegan and Nagamune, 2014; Zhao and Nagamune, 2018). In the case where multiple control modes share a single Lyapunov function, then the controlled system remains stable under switching, regardless of the switching sequence. The switches between the different LPV control modes have to be performed more carefully when they do not share a single Lyapunov function. Nonetheless, the theory linked to switched systems is a well developed field (Liberzon, 2003), the switching techniques rely on smooth switching, bumpless switching, hysteresis or dwell time requirements (Lu and Wu, 2004; Yang et al., 2018). The case where all the control modes have a single Lyapunov function can be restrictive, therefore, a common technique used for the design of LPV controllers is to rely on a parameter-dependent Lyapunov function (Daafouz and Bernussou, 2001; Mason et al., 2007).

This chapter provides a solution to the distributed control of non-linear systems, modelled by LPV discrete time state space models. A distributed control technique where a supervisory agent selects a control law amongst a finite set of LPV controllers, minimizing a cost function combining the predicted upper bound on the infinite horizon control cost with a communication penalty is proposed. The control modes are selected online, periodically, based on the value of the system state variable, before being transmitted to the local controllers. Therefore, this control

technique rely on the offline synthesis of a set of LPV control modes having different communication requirements, as well as the online selection of the most appropriate control law. Such a control method can be used to provide efficient performance for system regulation, where the subsystems rely on wireless communication channels, only to tackle more efficiently the potential disturbances. Without changing the approach taken, convex constraints can be added to the offline control law synthesis in order to tackle constraints on the state and input variables (Wada et al., 2006).

The design approach taken within this chapter proposes a new distributed LPV control technique suited for LPV systems, where the system is already partitioned into non-overlapping or overlapping subsystems. The LPV control mode is selected online based on the previous exogenous scheduling parameter measurements as well as the current state variable value, in order to improve the system-wide performance as well as a communication metric. The stability of the switched LPV control modes is ensured online based on a sufficient control cost decrease condition, equivalent to verifying the feasibility of a set of Linear Matrix Inequalities (LMIs). The recursive feasibility is ensured by selecting the same LPV control mode as the one currently implemented by the subsystems, thus not triggering any control switches. The decentralized offline control mode can be used as a fail safe mode in case where some or all the communication channels are failing. The offline control synthesis problems are cast as Semi-definite Programming (SDP) optimization problems, including possible Bilinear Matrix Inequality (BMI) constraints in order to induce some sparsity structure in the control gain matrices.

This chapter is organized as follows, section 6.2 presents the problem statement, introducing the required notations and assumptions on the system model. In section 6.3, the design of the non-structured offline feedback control law is formulated as a SDP optimization problem, section 6.4 explains how to formulate the control synthesis optimization problems in order to compute structured LPV controllers. In section 6.5, stability conditions are presented and the recursive feasibility of the control technique is explained. Section 6.6 explains the LPV controller algorithm and section 6.7 includes a numerical example. Finally, section 6.8 concludes this chapter.

6.2 Problem Statement

6.2.1 Linear Parameter-varying System Dynamics

This chapter is concerned with the optimal distributed control of linear parameter-varying discrete time systems, such that the entire system can be represented by the

following LPV system model,

$$x_{k+1} = A(\theta_k)x_k + Bu_k, \tag{6.1}$$

where, $k \in \mathbb{N}$ represents the discrete time step index, $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^m$ are respectively the system state and input variables. For all $k \in \mathbb{N}$, $\theta_k \in \mathbb{R}^s$ is a measurable exogenous scheduling parameter and the state and input matrices $A(\theta_k)$ and B are of appropriate dimensions. It is assumed that for any given time step $k \in \mathbb{N}$, $A(\theta_k)$ is linear with respect to the entries of the vector θ_k, and that the scheduling parameter θ_k belongs to the following unit simplex,

$$\forall k \in \mathbb{N},\ \theta_k = \left[\theta_{1,k}, \ldots, \theta_{s,k}\right]^\top \in \Lambda_s, \tag{6.2a}$$

$$\Lambda_s = \left\{ \theta_k \in \mathbb{R}^s \,\middle|\, \sum_{i=1}^{s} \theta_{i,k} = 1,\ \forall i \in [\![1, s]\!],\ \theta_{i,k} \geq 0 \right\}. \tag{6.2b}$$

For all $k \in \mathbb{N}$, the system matrix $A(\theta_k)$ linearly depends on the entries of the exogenous parameter θ_k as follows,

$$\forall k \in \mathbb{N},\ \forall \theta_k \in \Lambda_s,\ A(\theta_k) = \sum_{i=1}^{s} \theta_{i,k} A_i. \tag{6.3}$$

Since the state matrix $A(\theta_k)$ is parametrized by a convex combination of a set of matrices, it implies that, for all $k \in \mathbb{N}$, $A(\theta_k)$ belongs to a matrix polytope, denoted Ω_s and defined such that,

$$\forall k \in \mathbb{N},\ A(\theta_k) \in \Omega_s, \tag{6.4a}$$

$$\Omega_s = \operatorname{co}\left\{A_i \,|\, i \in [\![1, s]\!]\right\}. \tag{6.4b}$$

Since for all values of k, θ_k is in the compact set Λ_s, it implies that θ_k as well as all of its entries are bounded. Therefore, the set Ω_s is also a compact set as the image of a compact set by a linear transformation. The next subsection introduces the LPV system decomposition into coupled LPV subsystem models.

6.2.2 Distributed Linear Parameter-varying System Dynamics

The discrete time LPV system provided in equation (6.1) is partitioned into $N \in \mathbb{N}^*$ coupled subsystems, such that for all $p \in [\![1, N]\!]$ the subsystem indexed by p is

modelled as follows,

$$x_p^+ = A_{pp}(\theta_k)x_p + B_{pp}u_p + \sum_{\substack{j=1 \\ j\neq p}}^{N} \left\{ A_{pj}(\theta_k)x_j + B_{pj}u_j \right\}, \tag{6.5}$$

where $x_p \in \mathbb{R}^{n_p}$ and $u_p \in \mathbb{R}^{m_p}$ are respectively the state and input variables of subsystem p. The right hand side sum in equation (6.5) represents the subsystem to subsystem interactions. The vector x_p^+ denotes the successor state for the subsystem p, and $A_{pp}(\theta_k)$ and B_{pp} are matrices of appropriate dimensions representing the block partitioning of the original state and input matrices of system (6.1). The dimensions of the block matrices are such that,

$$\sum_{p=1}^{N} n_p \geq n, \tag{6.6a}$$

$$\sum_{p=1}^{N} m_p = m. \tag{6.6b}$$

The equation (6.6a) defines a possible overlapping condition for the subsystem state variables, whereas the equation (6.6b) denotes a non-overlapping condition for the subsystem input variables. Therefore, a state variable and an input variable can respectively be shared by multiple subsystems or belongs to a unique subsystem. The following subsection presents the notation used for the LPV control modes as well as the assumption made on the overall LPV system (6.1).

6.2.3 Linear Parameter-varying State Feedback Control Law

In order to ensure that the LPV system (6.1) can be stabilized by an LPV controller, and therefore, that a parameter-dependent Lyapunov function exists (De Oliveira et al., 1999; Daafouz and Bernussou, 2001), the system has to comply with the definition of poly-quadratic stability given in Definition 6.1.

Definition 6.1. *A system is said to be poly-quadratically stabilizable if and only if there exist, for all* $i \in [\![1, s]\!]$, $X_i \in \mathbb{S}^n_{++}$, *and* K_i, G_i *of appropriate dimensions, solution of the following LMIs, for all* $(i, j) \in [\![1, s]\!]^2$,

$$\begin{bmatrix} G_i + G_i^\top - X_i & (A_iG_i + BK_i)^\top \\ A_iG_i + BK_i & X_j \end{bmatrix} \succ 0. \tag{6.7}$$

Assumption 6.1. *The LPV system* (6.1) *as well as the subsystems* (6.5) *are poly-quadratically stabilizable.*

The assumption 6.1 implies that the system (6.1) can be stabilized by a LPV control law F_0 and that a corresponding parameter-dependent Lyapunov function P_0 exists. In addition to this, the existence of a decentralized LPV controller is ensured, due to the poly-quadratic stabilizability of the subsystems (6.5). The LPV control law F_0 is denoted by the following parametrization,

$$\forall k \in \mathbb{N},\, \forall \theta_k \in \Lambda_s,\, F_0(\theta_k) = \sum_{i=1}^{s} \theta_{i,k} F_{i,0}. \tag{6.8}$$

Applying the LPV control law F_0 stabilizes the system over the entire convex hull Ω_s. Consequently, when the LPV system (6.1) is controlled using F_0, the control inputs u_k are calculated as follows,

$$\forall k \in \mathbb{N},\, \forall \theta_k \in \Lambda_s,\, u_k = F_0(\theta_k) x_k. \tag{6.9}$$

Therefore, the system (6.1) in closed-loop control with the control mode F_0 becomes an autonomous discrete time LPV system defined such that,

$$\begin{aligned} &\forall k \in \mathbb{N},\, \forall \theta_k \in \Lambda_s, \\ &\qquad x_{k+1} = A(\theta_k) + BF_0(\theta_k) x_k && \text{(6.10a)} \\ &\qquad\quad = \left[A(\theta_k) + BF_0(\theta_k)\right] x_k && \text{(6.10b)} \\ &\qquad\quad = A_{cl_0}(\theta_k) x_k. && \text{(6.10c)} \end{aligned}$$

The dynamics of the closed-loop system using the LPV control law F_0 are denoted by the LPV system A_{cl_0}. More generally, for any LPV control law F_i indexed by $i \in \mathbb{N}$, the closed-loop LPV system is denoted as follows,

$$\forall (i,k) \in \mathbb{N}^2,\, \forall \theta_k \in \Lambda_s,\, A_{cl_i}(\theta_k) = A(\theta_k) + BF_i(\theta_k). \tag{6.11}$$

In the remainder of this chapter, F_0 will be used to refer to the optimal unstructured LPV control law. Subsequently, for all $i \in [\![1, s]\!]$, the gains $F_{i,0}$ defined as per equation (6.8) to form F_0 by convex combination are all fully centralized in general. In the particular case where all or part of the subsystems are completely decoupled, and that over the entire range of system dynamics, then all the $F_{i,0}$ will be having the same decentralized structure of the system. The next subsection formulates the distributed LPV control problem treated within this chapter.

6.2.4 Control Problem

The aim is to synthesize a set or alphabet of asymptotically stabilizing LPV control modes offline, optimally, all relying on different communication topologies. Then, a supervisory unit selects online the best LPV control law candidate from the finite set of LPV controllers, based on the value of the system state variable as well as the past history of the system dynamics. The optimal controller selection is performed periodically, online, in order to minimize a combined cost composed of a prediction of the upper bound on the infinite horizon quadratic control cost combined with the subsystem to subsystem communication burden. Example 6.1 presents the main idea behind the online optimal selection of an LPV control mode, Figure 6.2 pictures the exchange of communication packets between the LPV subsystems and the supervisory unit.

Example 6.1 (Distributed LPV framework). *This example presents the main functioning principle behind the distributed control technique applied to the LPV subsystems. The linear-parameter varying control law F_i is broadcast to the system composed of two non-overlapping subsystems, respectively indexed by* 1 *and* 2*. The supervisory unit in Figure 6.1 periodically updates the control mode F_i used by the system. It can be noted that the communication structure between the subsystems is provided by the block structure of F_i. The LPV controller F_i can be decomposed into block matrices as follows,*

$$F_i(\theta_k) = \begin{bmatrix} F_{i,11}(\theta_k) & F_{i,12}(\theta_k) \\ F_{i,21}(\theta_k) & F_{i,22}(\theta_k) \end{bmatrix}. \tag{6.12}$$

Therefore, the subsystems control inputs are calculated based on the following relations,

$$\begin{cases} u_1 = F_{i,11}(\theta_k)x_1 + F_{i,12}(\theta_k)x_2 \\ u_2 = F_{i,21}(\theta_k)x_1 + F_{i,22}(\theta_k)x_2 \end{cases} \tag{6.13}$$

In this example, when $F_{i,12}$ is equal to the zero matrix, then the subsystem 2 *does not have to communicate its state variable to the subsystem* 1*, regardless of the value of θ_k. Similarly, with $F_{i,21}$ concerning the subsystem* 1*. Therefore, in this case a block diagonal F_i will correspond to the decentralized control of the LPV system, whereas a non-sparse F_i corresponds to a full use of the subsystem to subsystem communication channels.*

The next section describes how to compute the unstructured LPV control mode F_0 based on the solution of a SDP optimization problem.

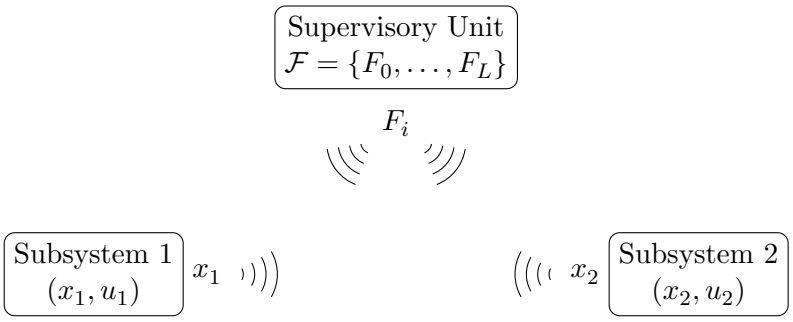

Figure 6.1: Representation of the information communicated between the LPV subsystem 1 and 2 as well as the supervisory unit.

6.3 Linear Parameter-varying Control Law Synthesis

The first step of the distributed LPV controller design is to compute the offline LPV control laws that will ensure the stability of the system over the entire convex hull Ω_s. The offline LPV controllers are designed with regards to the following weighting matrices $(Q, R) \in \mathbb{S}_+^n \times \mathbb{S}_{++}^n$, respectively the state and input weights. The optimization problem (6.14) is a standard SDP optimization problem (Boyd et al., 1994; Boyd and Vandenberghe, 2010), solvable to global optimum, efficiently and in polynomial time (Nesterov and Nemirovskii, 1994).

$$
\begin{aligned}
&\underset{\rho, G_i, K_i, X_i}{\text{minimize}} \quad \rho \\
&\text{subject to} \quad \forall (i,j) \in [\![1,s]\!]^2, \\
&\qquad \begin{bmatrix} G_i + G_i^\top - X_i & (A_i G_i + B K_i)^\top & G_i^\top Q^{\frac{1}{2}} & K_i^\top R^{\frac{1}{2}} \\ A_i G_i + B K_i & X_j & 0 & 0 \\ Q^{\frac{1}{2}} G_i & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}} K_i & 0 & 0 & \rho I_m \end{bmatrix} \succ 0
\end{aligned}
\tag{6.14}
$$

The LPV control law F_0 is computed from the solution of the optimization problem (6.14) as follows,

$$
\forall k \in \mathbb{N},\ \forall \theta_k \in \Lambda_s,\ F_0(\theta_k) = \sum_{i=1}^{s} \theta_{i,k} F_{i,0}, \tag{6.15}
$$

where, for all $i \in [\![1,s]\!]$, the gain matrices $F_{i,0}$ are computed such that,

$$
\forall i \in [\![1,s]\!],\ F_{i,0} = K_i G_i^{-1}. \tag{6.16}
$$

Note that the matrices G_i are all non-singular since the following relations hold,

$$\forall i \in [\![1, s]\!],\ G_i + G_i^\top \succ X_i \succ 0. \tag{6.17}$$

Solving the optimization problem (6.14) also yields a parameter-dependent Lyapunov function (El Ghaoui and Niculescu, 2000; De Oliveira et al., 1999), provided such that,

$$\forall k \in \mathbb{N},\ \forall \theta_k \in \Lambda_s,\ P_0(\theta_k) = \sum_{i=1}^{s} \theta_{i,k} P_{i,0}, \tag{6.18}$$

where, for all $i \in [\![1, s]\!]$, the matrices $P_{i,0}$ are computed as follows,

$$\forall i \in [\![1, s]\!],\ P_{i,0} = \rho X_i^{-1}. \tag{6.19}$$

The optimization problem (6.14) is known to be feasible, based on the Assumption 6.1 made on the overall LPV system. Solving the problem (6.14) offline provides the first LPV control law that will be used to populate the set of feasible control modes $\mathcal{F}$. Theorem 6.1, establishes that a feasible solution for the optimization problem (6.14), is stabilizing for the LPV system (6.1).

Theorem 6.1. *Any feasible solution of the optimization problem* (6.14) *constitutes a stable LPV control law, poly-quadratically stabilizing for the system* (6.1).

Proof. The set LMIs provided in equation (6.14) implies the following inequalities,

$$\forall i \in [\![1, s]\!], \tag{6.20a}$$
$$G_i + G_i^\top - X_i \succ 0 \tag{6.20b}$$
$$X_i \succ 0. \tag{6.20c}$$

Subsequently, since X_i is strictly positive definite, G_i is non-singular and the following conditions hold,

$$\forall i \in [\![1, s]\!],\ (X_i - G_i)^\top (X_i)^{-1} (X_i - G_i) \succeq 0. \tag{6.21}$$

The equation (6.21) can be expanded into the equivalent following condition, for all $i \in [\![1, s]\!]$,

$$G_i^\top X_i^{-1} G_i \succeq G_i + G_i^\top - X_i. \tag{6.22}$$

Subsequently, using the equation (6.22) in the LMI constraint of the optimization

problem (6.14), implies that,

$$\forall (i,j) \in [\![1,s]\!]^2, \begin{bmatrix} G_i^\top X_i^{-1} G_i & (A_i G_i + B K_i)^\top & G_i^\top Q^{\frac{1}{2}} & K_i^\top R^{\frac{1}{2}} \\ A_i G_i + B K_i & X_j & 0 & 0 \\ Q^{\frac{1}{2}} G_i & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}} K_i & 0 & 0 & \rho I_m \end{bmatrix} \succ 0. \quad (6.23)$$

The LMI constraints (6.22) can be changed into equivalent LMI constraints by using a congruence transformation. This transformation uses the matrix (6.24) as well as its transpose.

$$\begin{bmatrix} G_i^{-1} & 0 & 0 & 0 \\ 0 & X_j^{-1} & 0 & 0 \\ 0 & 0 & I_n & 0 \\ 0 & 0 & 0 & I_m \end{bmatrix} \quad (6.24)$$

The congruence transformation is achieved by pre and post-multiplying the LMI constraints (6.23) respectively by the transpose and the matrix (6.24), then using the change of variables $K_i = F_{i,0} G_i$, yields the following LMI condition,

$$\forall (i,j) \in [\![1,s]\!]^2, \begin{bmatrix} X_i^{-1} & \left(A_i + B F_{i,0}\right)^\top X_i^{-1} & Q^{\frac{1}{2}} & F_{i,0}^\top R^{\frac{1}{2}} \\ X_j^{-1}\left(A_i + B F_{i,0}\right) & X_j^{-1} & 0 & 0 \\ Q^{\frac{1}{2}} & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}} F_{i,0} & 0 & 0 & \rho I_m \end{bmatrix} \succ 0. \quad (6.25)$$

Applying the change of variable, $P_{i,0} = \rho X_i^{-1}$ to the equation (6.25), yields the following LMI constraints,

$$\forall (i,j) \in [\![1,s]\!]^2, \begin{bmatrix} \frac{P_{i,0}}{\rho} & \left(A_i + B F_{i,0}\right)^\top \frac{P_{i,0}}{\rho} & Q^{\frac{1}{2}} & F_{i,0}^\top R^{\frac{1}{2}} \\ \frac{P_{j,0}}{\rho}\left(A_i + B F_{i,0}\right) & \frac{P_{j,0}}{\rho} & 0 & 0 \\ Q^{\frac{1}{2}} & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}} F_{i,0} & 0 & 0 & \rho I_m \end{bmatrix} \succ 0. \quad (6.26)$$

Then, multiplying the LMI constraints by $\theta_{i,k}$ and summing and repeating this process with $\theta_{j,k+1}$ and summing the constraints again, provides the new following

constraints

$$\begin{bmatrix} \frac{P_0(\theta_k)}{\rho} & \left(A(\theta_k)+BF_0(\theta_k)\right)^\top \frac{P_0(\theta_{k+1})}{\rho} & Q^{\frac{1}{2}} & \left(F_0(\theta_k)\right)^\top R^{\frac{1}{2}} \\ \frac{P_0(\theta_{k+1})}{\rho}\left(A(\theta_k)+BF_0(\theta_k)\right) & \frac{P_0(\theta_{k+1})}{\rho} & 0 & 0 \\ Q^{\frac{1}{2}} & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}}F_0(\theta_k) & 0 & 0 & \rho I_m \end{bmatrix} \succ 0. \tag{6.27}$$

Finally, applying the Schur complement to the LMI constraint (6.27) gives a parameter-dependent discrete time Lyapunov equation as follows,

$$\begin{aligned} \left(A(\theta_k)+BF_0(\theta_k)\right)^\top P_0(\theta_{k+1})\left(A(\theta_k)+BF_0(\theta_k)\right) - P_0(\theta_k) \prec -Q \\ -\left(F_0(\theta_k)\right)^\top RF_0(\theta_k). \end{aligned} \tag{6.28}$$

Therefore, the optimization problem (6.14) minimizes the sum of the eigenvalues of the parameter-dependent Lyapunov function P_0, solution of the Lyapunov function (6.28), which concludes the proof. □

According to Assumption 6.1 as well as Theorem 6.1, there exist a centralized LPV control mode F_0 stabilizing for the system (6.1). Subsequently, the set of possible LPV control modes contains at least one control law, and can be initialized with F_0 such that $\mathcal{F} = \{F_0\}$. The next section presents how to perform the synthesis of other structured LPV control modes, in both the non-overlapping and overlapping partitioning cases, to populate the set of control modes $\mathcal{F}$.

6.4 Structured LPV Control Modes

In the past, some research work has been performed to compute structured controllers (Veselý et al., 2013; Schuler et al., 2014; Babazadeh and Nobakhti, 2016). However, structured control laws have rarely been achieved in a LPV framework. This section is organized in two subsections presenting respectively design procedures for the non-overlapping and the overlapping structured LPV control laws. The design of the LPV control law F_i with $i \in [\![1, L]\!]$ will ensure the stability of the system over the entire convex hull of system dynamics Ω_s, while relying on a specific communication topology.

6.4.1 Non-overlapping LPV Control Modes

Sufficient conditions can be formulated in order to induce structure in the LPV feedback control gains. In the case where no state overlaps are allowed between the subsystems the controller structure will always be represented by non-overlapping block matrices. This is always achievable after a possible permutation of the state and input variables. In the same way as it has been done previously concerning the unstructured control mode F_0, the structured LPV controllers are designed with regards to two weighting matrices $(Q, R) \in \mathbb{S}^n_+ \times \mathbb{S}^n_{++}$, respectively the state and input weights. A sufficient condition to obtain LPV control laws F_l with a desired block sparsity pattern is to constrain the gain matrices $F_{i,l}$, for all $i \in [\![1, s]\!]$, that compose the LPV control mode F_l, to have the same block sparsity pattern. This can be achieved by adding structural equality constraints to the SDP optimization problem (6.14), as presented in equation (6.29)

$$\forall i \in [\![1, s]\!],\ \forall (k, l) \in \mathbb{I}_G,\ G_i(k, l) = 0 \tag{6.29a}$$

$$\forall i \in [\![1, s]\!],\ \forall (k, l) \in \mathbb{I}_K,\ K_i(k, l) = 0. \tag{6.29b}$$

The sets $\mathbb{I}_G$ and $\mathbb{I}_K$ represent the pairs of row and column indexes where the matrix entries have to be set to zero. This induces some block structure to the matrices G_i, K_i as well as the gain matrices $F_{i,l}$ obtained by product (Crusius and Trofino, 1999). Therefore, the optimization problem (6.14) including the sufficient structural constraints is still a standard SDP optimization problem (Boyd et al., 1994; Boyd and Vandenberghe, 2010).

$$\begin{aligned}
&\underset{\rho, G_i, K_i, X_i}{\text{minimize}} && \rho \\
&\text{subject to} && \forall (i, j) \in [\![1, s]\!]^2, \\
&&& \begin{bmatrix} G_i + G_i^\top - X_i & (A_i G_i + B K_i)^\top & G_i^\top Q^{\frac{1}{2}} & K_i^\top R^{\frac{1}{2}} \\ A_i G_i + B K_i & X_j & 0 & 0 \\ Q^{\frac{1}{2}} G_i & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}} K_i & 0 & 0 & \rho I_n \end{bmatrix} \succ 0 \\
&&& \forall (k, l) \in \mathbb{I}_G,\ G_i(k, l) = 0 \\
&&& \forall (k, l) \in \mathbb{I}_K,\ K_i(k, l) = 0
\end{aligned} \tag{6.30}$$

After solving the optimization problem (6.30), the structured LPV control mode

F_l can be computed as follows,

$$\forall k \in \mathbb{N},\, \forall \theta_k \in \Lambda_s,\, F_l(\theta_k) = \sum_{i=1}^{s} \theta_{i,k} F_{i,l}, \tag{6.31}$$

where, for all $i \in [\![1, s]\!]$, the $F_{i,l}$ are computed such that,

$$\forall i \in [\![1, s]\!],\, F_{i,l} = K_i G_i^{-1}. \tag{6.32}$$

For all $i \in [\![1, s]\!]$, the matrices G_i are all non-singular since the previous matrix inequalities presented in equation (6.17) still holds.

Lemma 6.1. *If there exist block diagonal matrices G_i, solutions of the optimization problem* (6.30)*, then the control gain $F_{i,l}$ has the same block structure as the matrix K_i.*

Proof. The matrix G_i is non singular and since the inverse of a block diagonal matrix with $N \in \mathbb{N}^*$ is another block diagonal matrix with the same number of non-zero blocks, it implies that

$$\forall i \in [\![1, s]\!],\, G_i^{-1} = \operatorname{diag}\left(G_{i,jj}^{-1} \,\middle|\, j \in [\![1, N]\!] \right), \tag{6.33}$$

where the operator diag concatenates the elements on the matrix diagonal, and the matrix $G_{i,jj}$ denotes the j-th diagonal block of the matrix G_i. Therefore, the gain matrix $F_{i,l}$ is computed such that,

$$\begin{aligned} &\forall i \in [\![1, s]\!], \\ &F_{i,l,pq} = \sum_{k=1}^{N} K_{i,pk} G_{i,kq}^{-1} \\ &F_{i,l,pq} = K_{i,pq} G_{i,qq}^{-1}. \end{aligned} \tag{6.34}$$

Subsequently, the block structure of the gain matrices $F_{i,l}$ is the same as the block structure of the matrices K_i, which completes the proof. □

Then, the set $\mathbb{I}_K$ is modified in order to populate the set of control modes $\mathcal{F}$ with control modes having different communication topologies. It can be noted that the communication structure is enforced on the decision variables G_i and K_i only, therefore, the parameter-dependent Lyapunov function remains structurally unconstrained, and is not-sparse in general. The next subsection presents a technique applicable to the design of structured LPV control modes when the system partitioning includes some state variable overlaps.

6.4.2 Overlapping LPV Control Modes

The control synthesis method used previously does not apply when some state variables are shared between different subsystems. Indeed, the block diagonal structure is preserved after computing inverses or products of matrices, but the block structure is not preserved when the block matrices include overlaps. In this case, the bilinear matrix equality problem (6.35) has to be solved in order to induce the required sparsity pattern in the matrix gains $F_{i,l}$.

$$
\begin{aligned}
\underset{\rho,G_i,K_i,X_i,F_{i,l}}{\text{minimize}} \quad & \rho + \sum_{i=1}^{s} \left\| \operatorname{vec}(W_i \circ F_{i,l}) \right\|_1 \\
\text{subject to} \quad & \forall (i,j) \in [\![1,s]\!]^2, \\
& \begin{bmatrix} G_i + G_i^\top - X_i & (A_i G_i + B K_i)^\top & G_i^\top Q^{\frac{1}{2}} & K_i^\top R^{\frac{1}{2}} \\ A_i G_i + B K_i & X_j & o & 0 \\ Q^{\frac{1}{2}} G_i & 0 & \rho I_n & 0 \\ R^{\frac{1}{2}} K_i & 0 & 0 & \rho I_n \end{bmatrix} \succ 0 \\
& F_{i,l} G_i = K_i
\end{aligned}
\tag{6.35}
$$

For all $i \in [\![1,s]\!]$, $W_i \in \mathbb{R}^{m \times n}$ are weighting matrices used to penalize the elements of $F_{i,l}$ that should be set zero. The optimization problem (6.35) is non-convex, and can be solved by alternate convex search starting from the solution of (6.14) obtained without any structural constraints. Then, the matrices F_i and G_i are fixed alternately in order to use the bi-convexity property of the bilinear matrix equality in (6.35). Such an alternate convex search converges to a local minimum, however due to the non-convexity of the search space, obtaining a sparse solution can be complex. The implementation of the structured LPV control law F_l is stabilizing for the system (6.1), then other control modes can be computed by changing the weighing matrices W_i, in order to induce different sparsity patterns. The next section presents the conditions linked to the stability and recursive feasibility of the distributed control technique.

6.5 Stability and Recursive Feasibility of the LPV Distributed Controller

The online selection of an optimal LPV control mode is based on the predicted control performance of the closed-loop system as well as on the communication cost, accounting for the communication channels required by the controller selected.

Nonetheless, the periodical selection of distinct control laws will trigger control mode switching. Therefore, before selecting the best control mode, the supervisory unit has to test them for stability. This section develops a sufficient stability condition that ensures a stable switch between two control modes. In addition to this result, the recursive feasibility of the control technique is also explained. From this section, it is assumed that the set $\mathcal{F}$ is populated with an alphabet of LPV control modes, designed offline based on the optimization methods presented previously and having different communication structures such that,

$$\exists L \in \mathbb{N},\ \mathcal{F} = \{F_0, F_1, \ldots, F_L\} \tag{6.36}$$

where F_0 is the unstructured controller and F_l with $l \in \mathbb{N}^*$ corresponds to a control mode with a given communication structure.

6.5.1 Controller Stability

In order to make sure that the closed-loop system is stable, the parameter-dependent Lyapunov function has to decrease strictly along any trajectory within the polytope Ω_s (Daafouz et al., 2002). It has been shown previously in Theorem 6.1 that the optimal synthesis of LPV controllers uses a parameter-dependent Lyapunov function ensuring a strict decrease of the Lyapunov cost function. Therefore, each control mode of the set $\mathcal{F}$ is asymptotically stabilizing for the system (6.1). In order to ensure the asymptotic stability of the controlled system when a switch occurs, the Lyapunov function has to decrease strictly between the initial time instants of two consecutive control modes. This condition can be verified by measuring the scheduling parameter θ_k online in order to compute the state transition matrix defined as follows,

$$\mathcal{A}_{l|k\to k+\Delta} = \prod_{i=k}^{k+\Delta-1} A_{cl_l}(\theta_i) \tag{6.37}$$

where $l \in [\![1, L]\!]$ denotes to the control mode index, k and $k+\Delta$ represents respectively the starting and finishing time step indexes and the product operator corresponds to the matrix post-multiplication. Subsequently, $\mathcal{A}_{l|k\to k+\Delta}$ corresponds to the system closed-loop transition matrix from time step k to time step $k+\Delta$ when the control mode l is used, in the case where $\Delta = 0$ then $\mathcal{A}_{l|k\to k}$ is defined equal to I_n.

Lemma 6.2. *A switch from control mode l_1 to control mode l_2 is stable after Δ*

time steps from time step k, if the following LMIs are verified,

$$\forall i \in [\![1, s]\!],\ \mathcal{A}_{l_1|k\to k+\Delta}^{\top} P_{i,l_2} \mathcal{A}_{l_1|k\to k+\Delta} \prec P_{l_1}(\theta_k). \tag{6.38}$$

Proof. Multiplying the LMI constraints (6.38) by $\theta_{i,k+\Delta}$ and summing the s inequalities gives the following LMI,

$$\mathcal{A}_{l_1|k\to k+\Delta}^{\top} P_{l_2}(\theta_{k+\Delta}) \mathcal{A}_{l_1|k\to k+\Delta} \prec P_{l_1}(\theta_k). \tag{6.39}$$

Then, pre and post-multiplying the inequality given in equation (6.39) by the non-zero vectors $x_k^{\top}$ and x_k respectively, yields the following inequality,

$$\left[\mathcal{A}_{l_1|k\to k+\Delta} x_k\right]^{\top} P_{l_2}(\theta_{k+\Delta}) \mathcal{A}_{l_1|k\to k+\Delta} x_k < x_k^{\top} P_{l_1}(\theta_k) x_k. \tag{6.40}$$

Consequently, the inequality defined in equation (6.40) is equivalent to a strict control cost decrease, such that

$$x_{k+\Delta}^{\top} P_{l_2} x_{k+\Delta} < x_k^{\top} P_{l_1}(\theta_k) x_k \tag{6.41a}$$

$$\Leftrightarrow V_{l_2}(x_{k+\Delta}) < V_{l_1}(x_k). \tag{6.41b}$$

The quadratic Lyapunov functions obtained for each LPV control modes are constrained to be strictly positive definite by design as a convex combination of positive definite matrices. Subsequently, the Lyapunov matrix solutions associated to the control modes have strictly positive eigenvalues. Therefore, since the LMI condition (6.38) implies the inequalities presented in equation (6.41), it concludes the stability proof. □

The LMI condition established by Lemma 6.2 is only a sufficient condition for stability under switching. Subsequently, verifying that the LMIs (6.38) holds ensures that the control switch from the control mode l_1 to the control mode l_2 is stable. The next subsection presents the recursive feasibility of the distributed controller.

6.5.2 Controller Feasibility

Ensuring a decrease of the parameter-dependent Lyapunov function is a sufficient and necessary condition in order to ensure the asymptotic stability of the system under control mode switching. Since each control mode belonging to the set of control modes $\mathcal{F}$ satisfies the parameter-dependent Lyapunov equation (6.28), and that the right hand side of this equation is positive definite, it implies that the

following inequality is verified,

$$\forall (k, F_l) \in \mathbb{N} \times \mathcal{F},\ \mathcal{A}_{l|k\to k+1}^{\top} P_l(\theta_{k+1}) \mathcal{A}_{l|k\to k+1} - P_l(\theta_k) \prec 0. \tag{6.42}$$

The recursive feasibility of the online distributed controller is trivial and comes from the fact that remaining in the same control mode l asymptotically stabilizes the system, without triggering a control switch. Therefore, when the supervisory unit evaluates the LMIs conditions (6.38) defined previously, the set of achievable stable control modes will contain at least F_l according to (6.42).

Proposition 6.1. *If a switch between two control laws l_1 and l_2 is asymptotically stable after $\Delta^\star \in \mathbb{N}^*$ time steps, then the same control switch remains stable after any $\Delta \in \mathbb{N}^*$ time steps if $\Delta \geq \Delta^\star$.*

Proof. A stable switch from the control mode l_1 to the control mode l_2 performed after $\Delta^\star$ time steps implies that the following LMI constraint is verified for any value of time step k,

$$\forall k \in \mathbb{N},\ \mathcal{A}_{l_1|k\to k+\Delta^\star}^{\top} P_{l_2}(\theta_{k+\Delta^\star}) \mathcal{A}_{l_1|k\to k+\Delta^\star} \prec P_{l_1}(\theta_k). \tag{6.43}$$

In addition to this, since the control mode l_1 is stable, the following LMI constraint holds,

$$\forall k \in \mathbb{N},\ \mathcal{A}_{l_1|k\to k+1}^{\top} P_{l_1}(\theta_{k+1}) \mathcal{A}_{l_1|k\to k+1} - P_{l_1}(\theta_k) \prec 0. \tag{6.44}$$

Pre and post-multiplying the LMI condition (6.43) respectively by the transition matrices $\mathcal{A}_{l_1|k+\Delta^\star\to k+\Delta^\star+1}^{\top}$ and $\mathcal{A}_{l_1|k+\Delta^\star\to k+\Delta^\star+1}$ yields the following relation,

$$\begin{aligned} \forall k \in \mathbb{N},\ \mathcal{A}_{l_1|k\to k+\Delta^\star+1}^{\top} P_{l_2}(\theta_{k+\Delta^\star}) \mathcal{A}_{l_1|k\to k+\Delta^\star+1} \preceq \\ \mathcal{A}_{l_1|k+\Delta^\star\to k+\Delta^\star+1}^{\top} P_{l_1}(\theta_k) \mathcal{A}_{l_1|k+\Delta^\star\to k+\Delta^\star+1}. \end{aligned} \tag{6.45}$$

Finally, combining the LMI constraints (6.45) with the condition presented in equation (6.44) gives the following matrix inequality,

$$\begin{aligned} \forall k \in \mathbb{N},\ \mathcal{A}_{l_1|k\to k+\Delta^\star+1}^{\top} P_{l_2}(\theta_{k+\Delta^\star+1}) \mathcal{A}_{l_1|k\to k+\Delta^\star+1} \preceq \\ \mathcal{A}_{l_1|k+\Delta^\star\to k+\Delta^\star+1}^{\top} P_{l_1}(\theta_{k+1}) \mathcal{A}_{l_1|k+\Delta^\star\to k+\Delta^\star+1} \prec P_{l_1}(\theta_k) \end{aligned} \tag{6.46}$$

Noting that this condition holds for any value of time step k and applying it recursively completes the proof. □

According to Proposition 6.1, the set of control modes that can be used for switching increases with the value of Δ. Indeed, if $\Delta^\star_{l_i\to l_j} \in \mathbb{N}^*$ denotes the minimum dwell time from the control mode l_i to the control mode l_j, then the value $\Delta_{min} =$

$\max_{(i,j)\in[\![0,L]\!]^2} \left\{\Delta^{\star}_{l_i\to l_j}\right\}$ ensures that a control switch between any two control modes of the set $\mathcal{F}$ will be asymptotically stabilizing for the LPV system (6.1) (Liberzon, 2003). Therefore, the online selection of an LPV control mode is always feasible in order to ensure asymptotic stability. As presented in Figure 6.2, if the distributed controller is initialized with the unstructured control mode F_0, then F_0 constitutes a feasible solution as next control law. Similarly, when the control mode F_{l_1} is selected, the next control mode can be either the new candidate F_{l_2} or the same control mode F_{l_1}. The online distributed controller can be initialized with any control mode F_{l_0} from the alphabet $\mathcal{F}$ of control modes. A possible control switching sequence is as presented in Figure 6.2. Following the discussions about the stability and the recursive feasibility of the distributed LPV controller presented in Lemma 6.2 as well as Proposition 6.1, the main results are summarized in Theorem 6.2 formulated within the next subsection.

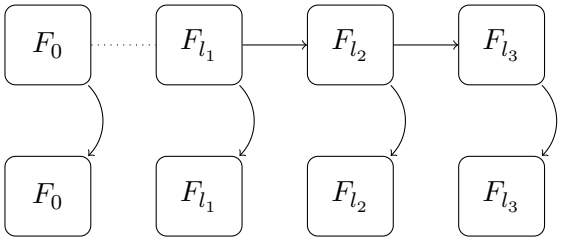

Figure 6.2: Switching sequence between the LPV control modes of the set $\mathcal{F}$.

6.5.3 Overall Controller Feasibility and Stability

This subsection gathers the results developed in the two previous subsections concerning the asymptotic stability as well as the recursive feasibility of the distributed LPV control technique. Theorem 6.2 summarizes the properties of the distributed LPV controller when applied to a LPV system complying with Assumption 6.1.

Theorem 6.2. *The distributed LPV controller obtained by selecting the best stable LPV control mode in the finite set $\mathcal{F}$ is recursively feasible and consecutive control modes are asymptotically stabilizing for the switched closed-loop LPV system* (6.1).

Proof. The proof of this theorem is straight forward, and follows directly from Lemma 6.2, Proposition 6.1 and the results developed and discussed previously. □

Following the discussion on the stability and recursive feasibility as well as the results provided about the synthesis of structured LPV control laws. The next section explains how the control algorithm proceeds to orchestrate stable control switches maximizing the system-wide performance while minimizing the communication cost, online.

6.6 Distributed Linear Parameter-varying Control Algorithm

This section presents how the online distributed LPV controller algorithm selects the most appropriate control mode candidate from the subset of feasible control modes $\mathcal{F}$. The predicted system performance is assessed based on the current measured state variable x_k, periodically, every $\Delta \in \mathbb{N}^*$ time steps. In order to select the most appropriate control mode, the first step is to reduce the set of controllers to the set of control modes that will preserve the asymptotic stability of the closed-loop system. This is achieved by verifying that the LMI condition (6.38) is verified. The set of all the control modes that are complying with this criteria forms the set denoted $\mathcal{F}_s$. As it has been proved previously, the set $\mathcal{F}_s$ is always different from the empty set due to the fact that that it always contains at least the current control mode.

Algorithm 6.1: Distributed LPV control algorithm

Inputs : θ_k
Output : F_l
Parameters : λ, Δ, $\mathcal{F} = \{F_0, F_1, \ldots, F_L\}$
Initialization: $k_0 = 0$, $k_1 = 0$, $l = 0$, $\mathcal{A}_{l|k_0 \to k_1} = I_n$
$\mathcal{A}_{l|k_0 \to k_1+1} = \mathcal{A}_{l|k_0 \to k_1} \times A_{cl_l}(\theta_k)$
$k_1 = k_1 + 1$
if $\mathrm{mod}(k_1 - k_0, \Delta) = 0$ **then**
  Check for switching stability conditions with LMIs in equation (6.38)
  Compute the set of stable control modes: $\mathcal{F}_s$
  Rank the control modes in the set $\mathcal{F}_s$
  Select best control mode: $F_{l^\star} \in \mathcal{F}_s$
  if $F_{l^\star} \neq F_l$ **then**
    $l = l^\star$
    $k_0 = k$
    $k_1 = k$
    $\mathcal{A}_{l|k_0 \to k_1} = I_n$
  end
end
return
F_l

Then, ranking the LPV control modes belonging to the set $\mathcal{F}_s$, is performed by computing a combined cost, composed of the predicted infinite horizon quadratic cost with a communication metric given in equation (6.47), as follows

$$\forall F_l \in \mathcal{F}_s,\ J_l = x_k^\top P_l(\theta_k) x_k + \lambda \|\mathrm{vec}\,(W \circ F_l)\|_0, \tag{6.47}$$

where $\lambda \in \mathbb{R}_+^*$ is a strictly positive weight constant, added to balance the control and communication costs, and $W \in [\![0,1]\!]^{m\times n}$ is a masking matrix used to account for the entries of F_l relying on subsystem to subsystem communication. In the unlikely case, where the best combined cost is achieved by more than one candidate, the control candidate providing the best control performance or the best communication cost can arbitrarily be implemented. The implementation of the distributed LPV control algorithm requires to set the tuning parameter λ, also since any control modes from the set $\mathcal{F}_s$ can be implemented at a given control switch instant, the value of λ can be adjusted online as a function of the system behaviour. Finally, some LMI constraints can be used during the design of the set $\mathcal{F}$ in order to enforce physical system constraints (Wada et al., 2006). The next section presents a numerical example to show the efficacy of the distributed algorithm.

6.7 Numerical Examples

This numerical examples have been solved using Yalmip (Löfberg, 2004) as well as the optimization solvers SeDuMi (Sturm, 1999) and Mosek (MOSEK ApS, 2017).

6.7.1 Example with Small Scheduling Parameter Dimension

This example is a modification of the benchmark example to evaluate robust controllers (Wie and Bernstein, 1992). The aim of the benchmark example is to control the position and velocity of two carts linked by a spring. The physical continuous system is discretized with a sampling time of $0.1s$, the masses m_1 and m_2 of the two carts are fixed and known, however the spring constant varies with time. The system is modelled by the discrete-time LPV system described in equation (6.48).

$$x_{k+1} = \begin{bmatrix} 1 & \frac{1}{10} & 0 & 0 \\ \frac{-K(\theta_k)}{10m_1} & 1 & \frac{K(\theta_k)}{10m_1} & 0 \\ 0 & 0 & 1 & \frac{1}{10} \\ \frac{K(\theta_k)}{10m_2} & 0 & \frac{-K(\theta_k)}{10m_2} & 1 \end{bmatrix} x_k + \begin{bmatrix} 0 & 0 \\ \frac{1}{10m_1} & 0 \\ 0 & 0 \\ 0 & \frac{1}{10m_2} \end{bmatrix} u_k \tag{6.48}$$

The state variables are the positions and velocities of the carts 1 and 2, respectively for the state variables index from 1 through 4. The control input variables are forces applied on the cart one and two. A representation of the system is given in Figure 6.3.

The continuous time state space model is discretized using Euler's first order approximation for the derivative with the sample time. The spring parameter $K(\theta_k)$ varies between $K_{min} = 1$ and $K_{max} = 5$, and the masses m_1 and m_2 are taken equal to 0.1, the weighting matrices Q and R are both equal to the identity matrices of

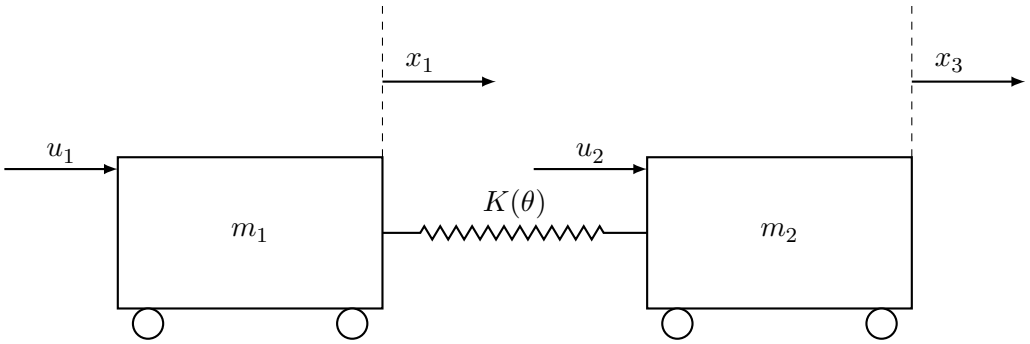

Figure 6.3: Two-mass-spring LPV system.

appropriate dimensions. Four different control modes with different communication requirements between the two carts are designed for this example, as follows,

$$\begin{aligned} F_0(\theta_k) = \theta_k &\begin{bmatrix} -4.12 & -1.42 & 2.80 & 0.46 \\ 2.80 & 0.46 & -4.12 & -1.42 \end{bmatrix} \\ &+ (1-\theta_k) \begin{bmatrix} -0.38 & -1.45 & -0.94 & 0.49 \\ -0.93 & 0.49 & -0.38 & -1.45 \end{bmatrix} \end{aligned} \tag{6.49}$$

$$\begin{aligned} F_1(\theta_k) = \theta_k &\begin{bmatrix} -7.66 & -1.88 & -0.38 & 0.07 \\ 0 & 0 & -3.83 & -1.43 \end{bmatrix} \\ &+ (1-\theta_k) \begin{bmatrix} -3.23 & -1.93 & -4.40 & 0.03 \\ 0 & 0 & -0.08 & -1.65 \end{bmatrix} \end{aligned} \tag{6.50}$$

$$\begin{aligned} F_2(\theta_k) = \theta_k &\begin{bmatrix} -3.83 & -1.43 & 0 & 0 \\ -0.38 & 0.07 & -7.66 & -1.88 \end{bmatrix} \\ &+ (1-\theta_k) \begin{bmatrix} -0.08 & -1.65 & 0 & 0 \\ -4.40 & 0.0287 & -3.23 & -1.93 \end{bmatrix} \end{aligned} \tag{6.51}$$

$$\begin{aligned} F_3(\theta_k) = \theta_k &\begin{bmatrix} -4.90 & -1.65 & 0 & 0 \\ 0 & 0 & -4.9030 & -1.6540 \end{bmatrix} \\ &+ (1-\theta_k) \begin{bmatrix} -2.78 & -1.86 & 0 & 0 \\ 0 & 0 & -2.78 & -1.86 \end{bmatrix} \end{aligned} \tag{6.52}$$

Therefore, for this example $\mathcal{F} = \{F_0, F_1, F_2, F_3\}$, the simulation is performed with the initial state $x_0^\top = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$. The aim of this simulation is to regulate the state variables to zero and a periodic switching time of $\Delta = 10$ time steps is used. The distributed controller relies on the fully centralized control law to tackle the transient behaviour, before disconnecting the subsystems communication channels without any major impact on the overall system performance. The control

costs and the number of communication packets transmitted are given in Table 6.1.

Table 6.1: Comparison of the cumulative control and communication costs for different control methods applied to the mass-spring-system (6.48).

Costs	F_0	**Distributed LPV**	F_3
Control	43.55	43.74	142.92
Number of communications	100	20	0
Total	143.55	63.74	142.92

6.7.2 Example with Medium Scheduling Parameter Dimension

This second example represents a modification of the LPV model for the CMPASS-1 gas turbine engine (Richter, 2012). The two state variables represent the fan and core engine speed, and the two input variables represent the fuel flow and the variable stator vane respectively. The gas turbine engine model is parametrized with a scheduling parameter of dimension two representing the Mach number and the altitude for values between 0 and 0.85 as well as $0ft$ and $42000ft$ respectively. The gas turbine engine model is discretized using a sampling time of $0.1s$. The discrete-time LPV system model is described in equation (6.53).

$$\begin{aligned} x_{k+1} &= A(\theta_k)x_k + Bu_k \\ x_{k+1} &= \left(A_0 + M \times A_M + h \times A_h\right) x_k + Bu_k \end{aligned} \tag{6.53}$$

The scheduling parameter θ_k is a vector of dimension four composed of the combinations of engine Mach number and altitude extreme cases. Thus, the number of vertices used to compute the control modes is equal to four. The vertices for the discrete time model as well as the input matrix are given as follows,

$$A_1 = \begin{bmatrix} 1.0202 & 0.0444 \\ 0.0137 & 1.1102 \end{bmatrix} \tag{6.54}$$

$$A_2 = \begin{bmatrix} 0.9373 & 0.0476 \\ 0.0216 & 1.0883 \end{bmatrix} \tag{6.55}$$

$$A_3 = \begin{bmatrix} 1.2826 & -0.0824 \\ -0.0083 & 1.4660 \end{bmatrix} \tag{6.56}$$

$$A_4 = \begin{bmatrix} 1.3968 & -0.0945 \\ -0.0211 & 1.4963 \end{bmatrix} \tag{6.57}$$

$$B = \begin{bmatrix} 29.6492 & -4.2037 \\ 100.8654 & 13.2474 \end{bmatrix}. \tag{6.58}$$

The fuel flow input and the fan speed state compose the first subsystem, the variable stator vane input combined with the core speed state variable form the second subsystem. The weighting matrices Q and R used for the control design are both equal to the identity matrices of appropriate dimensions, and four different control modes with different communication requirements between the two subsystems are designed, as follows,

$$\begin{aligned} F_0(\theta_k) = \theta_{1,k} \begin{bmatrix} -0.0166 & -0.0064 \\ 0.1255 & -0.0348 \end{bmatrix} & +\theta_{2,k} \begin{bmatrix} -0.0153 & -0.0064 \\ 0.1149 & -0.0336 \end{bmatrix} \\ + \theta_{3,k} \begin{bmatrix} -0.0208 & -0.0062 \\ 0.1587 & -0.0634 \end{bmatrix} & +\theta_{4,k} \begin{bmatrix} -0.0225 & -0.0062 \\ 0.1732 & -0.0660 \end{bmatrix} \end{aligned} \tag{6.59}$$

$$\begin{aligned} F_1(\theta_k) = \theta_{1,k} \begin{bmatrix} -0.0210 & -0.0072 \\ 0.0000 & -0.0580 \end{bmatrix} & +\theta_{2,k} \begin{bmatrix} -0.0194 & -0.0071 \\ 0.0000 & -0.0548 \end{bmatrix} \\ + \theta_{3,k} \begin{bmatrix} -0.0263 & -0.0073 \\ 0.0000 & -0.0931 \end{bmatrix} & +\theta_{4,k} \begin{bmatrix} -0.0286 & -0.0073 \\ 0.0000 & -0.0985 \end{bmatrix} \end{aligned} \tag{6.60}$$

$$\begin{aligned} F_2(\theta_k) = \theta_{1,k} \begin{bmatrix} -0.0174 & 0.0000 \\ 0.1308 & -0.0795 \end{bmatrix} & +\theta_{2,k} \begin{bmatrix} -0.0161 & 0.0000 \\ 0.1201 & -0.0780 \end{bmatrix} \\ + \theta_{3,k} \begin{bmatrix} -0.0214 & 0.0000 \\ 0.1631 & -0.1067 \end{bmatrix} & +\theta_{4,k} \begin{bmatrix} -0.0232 & 0.0000 \\ 0.1777 & -0.1092 \end{bmatrix} \end{aligned} \tag{6.61}$$

$$\begin{aligned} F_3(\theta_k) = \theta_{1,k} \begin{bmatrix} -0.0358 & 0.0000 \\ 0.0000 & -0.0153 \end{bmatrix} & +\theta_{2,k} \begin{bmatrix} -0.0280 & 0.0000 \\ 0.0000 & -0.0150 \end{bmatrix} \\ + \theta_{3,k} \begin{bmatrix} -0.0418 & 0.0000 \\ 0.0000 & -0.0458 \end{bmatrix} & +\theta_{4,k} \begin{bmatrix} -0.0475 & 0.0000 \\ 0.0000 & -0.0485 \end{bmatrix} \end{aligned} \tag{6.62}$$

The experiments is conducted with the set of control modes $\mathcal{F} = \{F_0, F_1, F_2, F_3\}$ from the initial state $x_0^\top = \begin{bmatrix} 100 & -100 \end{bmatrix}$, which correspond to a higher fan speed and a lower core speed to regulate. The scheduling parameter evolves from an altitude of $0ft$ and a Mach number of 0 to the $42000ft$ cruise altitude and the 0.85 cruise Mach number. The aim of this simulation is to regulate the state variables to zero which

correspond to their steady state values, a periodic switching time of $\Delta = 10$ time steps is used throughout the simulation. A summary presenting the control costs as well as the number of communication packets transmitted are given in Table 6.2.

Table 6.2: Comparison of the cumulative control and communication costs for different control methods applied to the modified CMAPS-1 turbofan engine (6.53).

Costs	F_0	**Distributed LPV**	F_3
Control	20258.01	20258.01	378875.07
Number of communications	50	20	0
Total	20308.01	20278.01	378875.07

The centralized LPV controller F_0 is close to a dead beat controller, therefore, switching to a decentralized control mode after the first 10 time steps does not affect the overall control cost. The Figure 6.4 and 6.5, respectively present the behaviour of the turbofan during the simulation for the centralized and decentralized LPV controllers.

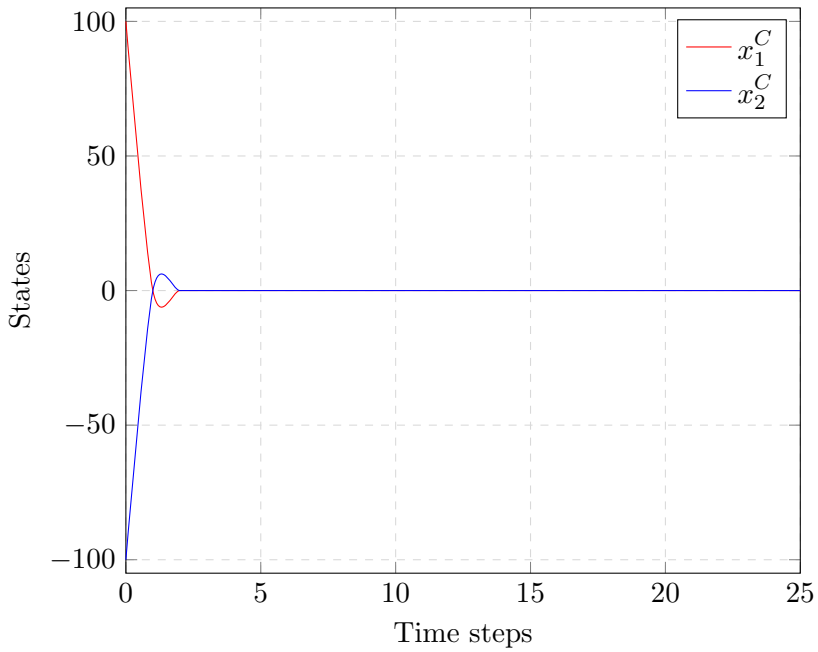

Figure 6.4: Centralized LPV controller applied to the CMAPS-1 turbofan engine (6.53).

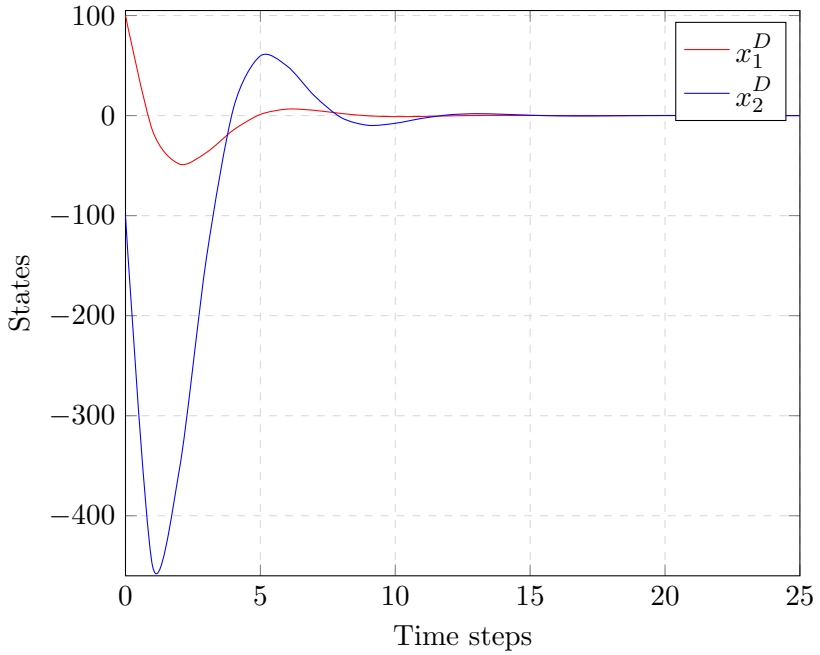

Figure 6.5: Decentralized LPV controller applied to the CMAPS-1 turbofan engine (6.53).

6.8 Conclusion

A distributed LPV control technique has been presented within this chapter, where a supervisory unit updates the LPV control mode by selecting a LPV controller amongst a finite set $\mathcal{F}$ of control laws. All the control laws composing the finite alphabet present different communication topologies and therefore, in the case where the LPV system is composed of coupled subsystems different control performance as well. The selection of the most appropriate control mode is performed by first selecting the stable controllers. Then, this subset of controllers is ranked based on a cost function including a prediction of the infinite horizon control performance combined with the number of communication channels in a weighted sum. The best control mode selected is broadcast to the subsystems and this process is repeated. It has been demonstrated that the synthesis of unstructured and structured LPV control laws can be achieved based on a SDP formulation, including BMI constraints in the case where some state variables are shared between multiple subsystems.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The contributions presented within this thesis are concerned with the optimal design of control system architectures as well as the online and offline synthesis of distributed control techniques, respectively with applications for linear system as well as non-linear system modelled as linear-parameter varying systems. In addition to this, the thesis has presented a literature review in control and system theory, introducing the concept of optimal control and has provided the necessary mathematical background in order for the research contributions to be motivated, and for the optimization techniques to be clearly introduced.

Systems and control engineering is a multidisciplinary field that affects a lot of different types of systems. Subsequently, a lot of theoretical properties have been developed in order to tailor the dynamical model to the physical systems. In addition to this, the chapter 2 has presented the work developed to show the stability of a system and to analyze the controllability and observability of a system. These properties have led naturally to the development of optimal control techniques. First, optimal control has been applied offline in the case of linear time invariant system models. Following this, online optimal control strategies such as model-based predictive control have been developed to provide an intuitive and simple way to deal with system constraints.

The chapter 3 introduced the background and the main mathematical notions and notations used throughout the thesis. The definition of convex sets as well as convex functions along with some of their fundamental properties has been presented, leading to the formulation of convex optimization problem. It has been shown that conic programming is a powerful optimization framework gathering most of the convex optimizations used in control and system theory. Even if convex optimization

is used extensively in control, the formulation of some problems lead to non-convex optimization. Therefore, this chapter also described the principal non-convex optimization problems encountered in the field of control systems. This chapter finished by presenting some of the algorithms and methods implemented to solve the optimization problems introduced previously.

Firstly, a control system architecture technique has been presented within chapter 4. The purpose of this technique is to partition a system model into non-overlapping or overlapping subsystem models, such that the interactions between subsystems are minimized. Partitioning a system model consists of partitioning the set of state variables as well as the set of input variables into paired subsets of state and input variables, where each subset pair represents one subsystem. The non-overlapping and overlapping partitionings are obtained by respectively preventing or allowing some state variables to be shared amongst multiple subsystems. Initially, this task has been achieved by expressing the partitioning minimization problem as a binary integer non-linear program, that has then been linearized based on the introduction of a state and input auxiliary variable. Solving this partitioning optimization problem offline yields the subsystem architecture having the least interacting but controllable subsystem models. It has been shown that the weak interactions partitioning problem is a combinatorial optimization problem. Therefore, it is a difficult problem to solve that in the worst case scenario leads to an exhaustive search, with the set of feasible solutions having a cardinality that grows exponentially with the size of the input parameters. However, this architectural optimization technique is important considering that partitioning a system into subsystems is the preliminary step that will condition the control system architecture, and therefore its design. Subsequently, the control architecture choice will directly influence the control system overall performance as well as the communication requirements when a non-centralized controller is implemented.

Secondly, in chapter 5, a supervised-distributed control algorithm has been developed in order to maximize the system-wide performance while minimizing the subsystem to subsystem communication burden. The supervised-distributed control technique uses a supervisory agent that periodically updates the subsystems control laws by solving an online optimization problem. This optimization problem has been formulated as a semi-definite program including a bilinear matrix equality. Subsequently, the optimization problem has been relaxed into two standard semi-definite programming problems solved alternately until a stopping criterion is reached. The asymptotic stability of the distributed controller is guaranteed based on a dwell time parameter, ensuring that switches between two consecutive control modes will not destabilize the system and will also provide enough time for the algorithm to

converge. Also, based on the biconvex structure of the optimization problem it has been proved that the algorithm converges to a stationary point that is a local optimum. It has been demonstrated that even if the algorithm does not converge within the allocated time, early termination will output a feasible control mode. Finally, recursive feasibility of the online optimization has been demonstrated. Variants of the supervised-distributed control algorithm can include physical system constraints as well as robustness to system model uncertainty, without affecting the results presented previously. The supervised-distributed control technique developed is relevant to wireless sensor networks, where the sensors of a subsystem will have to broadcast data wirelessly to other subsystems based on battery energy. Consequently, the control system can minimize the total amount of energy spent by the system by accounting for the communication in the control objective function.

Finally, an extension of the supervised-distributed control method applicable to non-linear systems modelled as linear parameter-varying systems has been presented within chapter 6. This control technique relies on the offline design of a set of linear-parameter varying control modes having different communication requirements. Following the offline design step, a supervisory agent can select the best control mode within a finite alphabet of control laws according to the predicted infinite horizon cost as well as the forecast communication cost. The selection and the implementation of a new control mode implies performing a switch from on controller to another. A switching sequence ensuring the asymptotic stability of the system can be guaranteed relying on sufficient linear matrix inequality conditions, checked online based on the history of the system dynamics. Trivially, the online control selection method remains feasible due to the fact that each control mode is asymptotically stable. It has been shown that the synthesis of linear-parameter varying control laws having a given sparsity structure can be performed optimally in the case of non-overlapping and overlapping subsystem decompositions. These optimization problems are formulated as semi-definite programming problems that include a set of bilinear matrix equalities when the controller sparsity pattern cannot be expressed using block matrices.

7.2 Future Work

Possible future work and research directions include points linked to the three contributions developed within this thesis, and are presented within the next three subsections.

7.2.1 Weak Interaction System Partitioning

The problem of partitioning a system into subsystems is a difficult combinatorial problem that scales up with poor performance. Therefore, the development of computational methods used to speed up the convergence of the system partitioning algorithm would be desirable. Another interesting aspect linked to the partitioning of a system model would be to be able to account for a multi-objective cost function that would include other physical data such as the distances between the sensors and actuators of a subsystem, as well as the total system weight or power consumption. The partitioning of linear-parameter varying system models would also be an interesting point to investigate. As it has been discussed previously, steering or constraining the partitioning algorithm to the controllable subsystems is complex, as it involves rank constraints with unknown matrix dimensions. However, the study of a partitioning algorithm including rank constraints would be very valuable.

7.2.2 Supervised-distributed Controller

Concerning the supervised-distributed algorithm, even if the control algorithm has a polynomial time complexity, the overall optimization of the control algorithm would be very relevant. Such a feature would potentially allow to apply the supervised-distributed control technique to very large-scale systems. Another algorithm complexity encountered was the need to include a bilinear matrix equality in the optimization problem, triggering the non-convexity of the feasible set. Research performed in the field of optimization, looking for new techniques applicable to problems involving bilinear matrix inequality is very active. The development of more efficient branch and bound methods or similar algorithms applicable to bilinear matrix problems would be very beneficial for this control technique. Last but not least, a study about the tuning of the control algorithm, with the trade-off parameter λ as well as the dwell time parameter Δ would be interesting.

7.2.3 Distributed Linear-parameter Varying Controller

Some future work regarding the distributed linear-parameter varying control technique could be investigating the online design of the control modes composing the control set $\mathcal{F}$. In order to be able to implement the optimal control law synthesis online, an improvement of the algorithm complexity would have to be performed, especially in the case of large and very large-scale systems. Future research directions could also be looking into the performance improvement of linear-parameter varying control laws based on the measurement of the real time system dynamics, based on the assumption that the rate of change of the exogenous scheduling parameter is

bounded. This would imply that the system dynamics are located within a subset of the achievable dynamics, and that this situation will remain for a certain amount of time steps. Finally, an extension to the linear-parameter varying problem can be considered where all the matrices composing the system model are varying with the exogenous scheduling parameter.

Appendix A

Stirling Numbers

A.1 Introduction

Number theory is a field of mathematics that studies integers, it subsumes the branch of combinatorics concerned with counting the number of combinations and permutations of finite sets (Laplace, 1820; Riordan, 1958). The Stirling numbers of first, second and third kind play an important role in combinatorial mathematics, they find applications for different analytic and combinatorial problems and they can all be used to express the coefficient of sequences of polynomials. All these numbers are linked by the fact that they can express the number of partitions of a set with n elements into k non-empty non-overlapping subsets. This appendix aims at introducing the Stirling numbers as well as how they can be computed. Also, a presentation of some of their properties as well as the link between them is also provided.

A.2 Stirling Numbers of the First Kind

The Stirling numbers of the first kind are linked to the number of cycles within a finite discrete set. These numbers can be unsigned or signed and are defined by the coefficients of the rising and falling factorials as follows,

$$x^{(n)} = \prod_{k=0}^{n-1}(x+k) = \sum_{k=0}^{n} \begin{bmatrix} n \\ k \end{bmatrix} x^k. \tag{A.1}$$

The equation (A.1) defines the rising factorial and its relation with the unsigned Stirling numbers of the first kind, whereas the equation (A.2) defines the falling factorial along with its relation with the unsigned Stirling numbers of the first kind. The signed Stirling numbers of the first kind are obtained by combining the sign in

the sum of (A.2) with the unsigned Stirling numbers of the first kind.

$$(x)_n = \prod_{k=0}^{n-1}(x-k) = \sum_{k=0}^{n}(-1)^{n-k}\begin{bmatrix} n \\ k \end{bmatrix} x^k \tag{A.2}$$

The unsigned Stirling numbers of the first kind correspond to the number of permutations of a set of n elements with k disjoint cycles. These numbers can also be computed using the following recurrence relation,

$$\forall (n,k) \in \mathbb{N}^* \times \mathbb{N}^*, \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 1, \begin{bmatrix} n \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ n \end{bmatrix} = 0, \tag{A.3a}$$

$$\begin{bmatrix} n+1 \\ k \end{bmatrix} = \begin{bmatrix} n \\ k-1 \end{bmatrix} + n\begin{bmatrix} n \\ k \end{bmatrix}. \tag{A.3b}$$

It is possible to show this recurrence relation using either the definition of the Stirling numbers of first kind based on the rising or falling factorials or by using their combinatorial definition based on permutations. The first few Stirling numbers of the first kind are presented within Table A.1.

Table A.1: Stirling numbers of the first kind.

$n \backslash k$	0	1	2	3	4	5
0	1					
1	0	1				
2	0	1	1			
3	0	2	3	1		
4	0	6	11	6	1	
5	0	24	50	35	10	1

For a fixed value of n, the sum of all the Stirling numbers of the first kind over k from 0 to n yields factorial n, which can be computed from the rising factorial with x equal 1.

$$\forall n \in \mathbb{N}, \sum_{k=0}^{n}\begin{bmatrix} n \\ k \end{bmatrix} = n! = \begin{bmatrix} n+1 \\ 1 \end{bmatrix} \tag{A.4}$$

Using the recursive relation (A.3), it is trivial to show that the right side equality of (A.4) holds.

Figure A.1 represents the different cycles linked to the Stirling number of the first kind with a set of $n = 3$ elements and $k = 2$ cycles. The total number of ways to partition the set is 3. In this case, all three partitions have a cycle composed of 2 elements and a cycle of a single element.

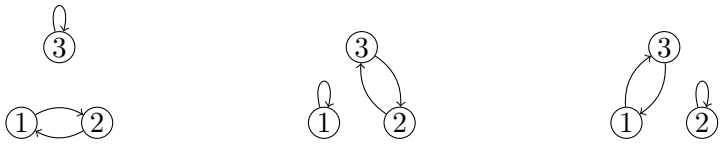

Figure A.1: Cycle representations of the Stirling number of the first kind with $n = 3$ and $k = 2$.

A.3 Stirling Numbers of the Second Kind

The Stirling numbers of the second kind represent the number of ways to partition a set of n elements into k non-empty non-overlapping subsets. These numbers are the inverse to the Stirling numbers of the first kind. Similarly to the Stirling number of the first kind, they can be computed based on the generating function consisting of the rising and falling factorials (A.5), such that,

$$\forall n \in \mathbb{N},\ x^n = \sum_{k=0}^{n} \left\{ {n \atop k} \right\} (x)_k = \sum_{k=0}^{n} (-1)^{n-k} \left\{ {n \atop k} \right\} x^{(k)} \tag{A.5}$$

It can be proved that there exists a recurrence relation between the Stirling number of the second kind based on their definition from the falling factorial. The recurrence relation is as follows,

$$\forall (n,k) \in \mathbb{N}^* \times \mathbb{N}^*, \left\{ {0 \atop 0} \right\} = 1,\ \left\{ {n \atop 0} \right\} = \left\{ {0 \atop n} \right\} = 0, \tag{A.6a}$$

$$\left\{ {n+1 \atop k} \right\} = k \left\{ {n \atop k} \right\} + \left\{ {n \atop k-1} \right\}. \tag{A.6b}$$

Table A.2 presents the first few Stirling numbers of the second kind which highlights the recurrence relation presented previously.

Table A.2: Stirling numbers of the second kind.

$n \backslash k$	0	1	2	3	4	5
0	1					
1	0	1				
2	0	1	1			
3	0	1	3	1		
4	0	1	7	6	1	
5	0	1	15	25	10	1

There is an explicit formula used to compute the Stirling numbers of the second

kind relying on factorials, this formula is as follows,

$$\left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} i^n. \tag{A.7}$$

The total number of ways to partition a set with n elements into non-empty non-overlapping subsets is defined by the Bell number. Consequently, the n-th Bell number B_n is equal to the sum of the Stirling numbers of the second kind such that,

$$\forall n \in \mathbb{N}, \sum_{k=0}^{n} \left\{ {n \atop k} \right\} = B_n. \tag{A.8}$$

In order to have an idea of the distinct ways to partition a set of n elements into k non-empty non-overlapping subsets, the Figure A.2 illustrates all the different partitioning ways with $n = 4$ and $k = 2$. In this case, there are 6 possible ways the partition the set of four elements, each one is composed of a subset composed of two elements as well as two subsets with a single element.

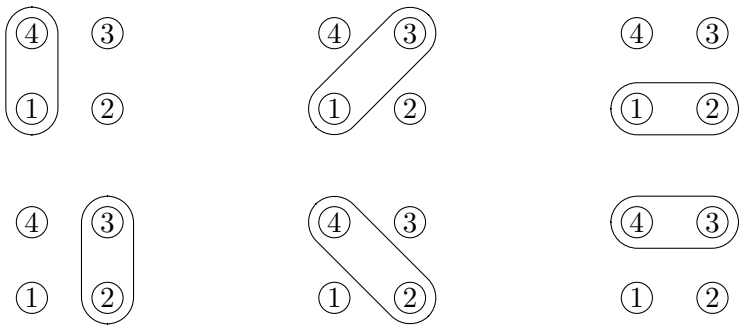

Figure A.2: Set partitions representing the Stirling number of the second kind with $n = 4$ and $k = 3$.

A.4 Lah Numbers

The Lah numbers, also sometimes called the Stirling numbers of the third kind, represent the number of ways to partition a set of n elements into k non-empty non-overlapping ordered subsets. Similarly to the Stirling number of the first and second kinds, the Lah numbers can be generated using the rising factorial as follows,

$$x^{(n)} = \sum_{k=1}^{n} \left[{n \atop k} \right] (x)_k. \tag{A.9}$$

In the same fashion as for the Stirling numbers of the first and second kinds, the

Lah numbers are linked to the falling factorial, the relation is as follows,

$$(x)_n = \sum_{k=1}^{n} (-1)^{n-k} \left\lfloor {n \atop k} \right\rfloor x^{(k)}. \tag{A.10}$$

The Lah numbers can be calculated by using the following recurrence relation,

$$\forall (n,k) \in \mathbb{N}^* \times \mathbb{N}^*, \left\lfloor {0 \atop 0} \right\rfloor = 1, \left\lfloor {n \atop 0} \right\rfloor = \left\lfloor {0 \atop n} \right\rfloor = 0, \tag{A.11a}$$

$$\left\lfloor {n+1 \atop k} \right\rfloor = (n+k) \left\lfloor {n \atop k} \right\rfloor + \left\lfloor {n \atop k-1} \right\rfloor. \tag{A.11b}$$

However, in this case, an explicit formula to compute these numbers exists based on binomial coefficients and factorials exists and is given by,

$$\left\lfloor {n \atop k} \right\rfloor = \binom{n-1}{k-1} \frac{n!}{k!}. \tag{A.12}$$

Some properties of the Lah numbers can be proved by recurrence, for example,

$$\forall n \in \mathbb{N}^*, \left\lfloor {n \atop 1} \right\rfloor = n!, \tag{A.13a}$$

$$\left\lfloor {n \atop n} \right\rfloor = 1, \tag{A.13b}$$

$$\left\lfloor {n \atop n-1} \right\rfloor = n(n-1). \tag{A.13c}$$

Finally, Table A.3 presents the first Lah numbers for n less or equal to 5. The distinct ways to partition a set of $n = 3$ elements into $k = 2$ non-empty non-overlapping ordered subsets is represented Figure A.3. In this case, there are 6 different partitions possible, every partition is composed of an ordered subset including two elements and a subset composed of a single element.

Table A.3: Lah numbers.

$n \backslash k$	0	1	2	3	4	5
0	1					
1	0	1				
2	0	2	1			
3	0	6	6	1		
4	0	24	36	12	1	
5	0	120	240	120	20	1

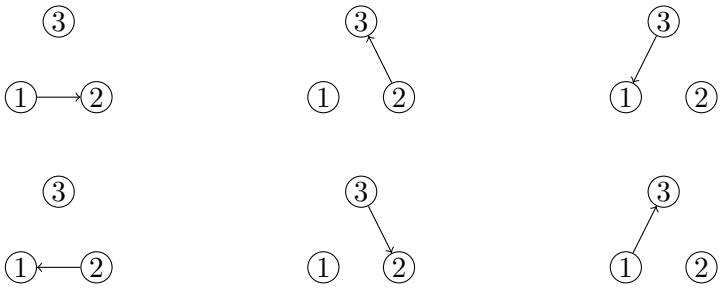

Figure A.3: Ordered subsets representing the Lah number with $n = 3$ and $k = 2$.

A.5 Relations Between the Stirling and Lah Numbers

The three different combinatorial numbers presented within the previous sections of this appendix are related. It can be seen that they represent a change in basis for polynomial functions as it is pictured by Figure A.4. A Polynomial function can be expressed uniquely with the canonical polynomial basis or with the polynomial basis generated from the rising and the falling factorials. The Figure A.4 presents how these numbers connect the three polynomial basis mentioned previously. Therefore, it means that the Stirling number of the first and second kind can be considered as inverses when they form lower triangular matrices whose entries are the Stirling numbers with corresponding row and column indexes. Similarly, as it can be seen Figure A.4, the Lah numbers and the Lah numbers multiplied by $(-1)^{n-k}$ can be seen as inverses when they compose the entries of lower triangular matrices. Finally, since these three combinatorial numbers represent different ways to partition a set of n elements into k subsets being respectively unordered, cyclically ordered and linearly ordered, the following inequalities naturally arise,

$$\forall (n,k) \in \mathbb{N} \times \mathbb{N}, \left\{ {n \atop k} \right\} \leq \left[{n \atop k} \right] \leq \left\lfloor {n \atop k} \right\rfloor \tag{A.14}$$

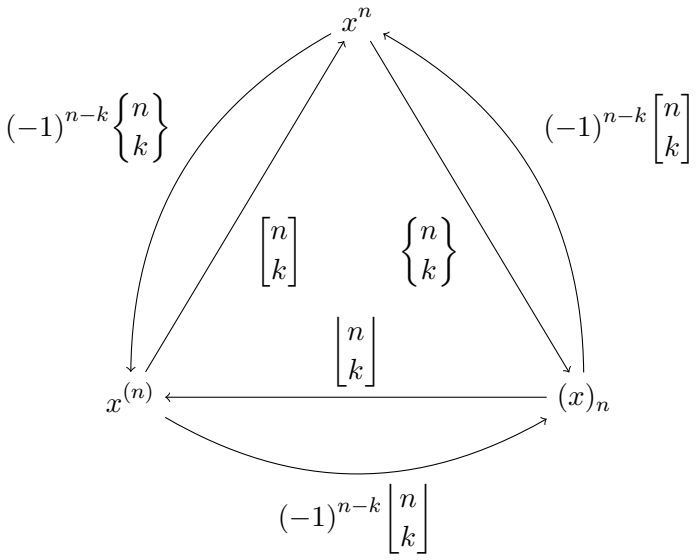

Figure A.4: Relations between the Stirling and Lah numbers.

Appendix B

Graph Theory

B.1 Introduction

Multiple real life situations can be modelled by a set of vertices connected together by edges, this representation is called a graph. Graph theory is a field of discrete mathematics that has been developed in order to provide the tools to analyse and solve problems involving graphs, subsequently providing answers to the real life situations they represent (Diestel, 2000; Bondy and Murty, 2008). Graphs can be undirected, directed, and even weighted based on the type of situation that is modelled. A given graph $\mathcal{G}$ is defined by an ordered pair of elements, including a set of vertices V and a set of edges E such that,

$$\mathcal{G} = (V, E). \tag{B.1}$$

The set of edges E is composed of pairs of vertices taken from the set V that can be directed and even weighted in some cases. Graphs can also be defined by their incidence or adjacency matrices. The incidence matrix of an undirected graph is a rectangle matrix whose rows represent the vertices and whose columns represent the edges, its entries are positive integers representing the number of times a vertex and an edge are incident. The adjacency matrix is a square matrix with rows and columns labelled after the vertices and composed of binaries. When an entry is equal to 1, it indicates the presence of an edge between the vertices corresponding to the row and column indexes, whereas a 0 entry means that there is no direct link between these two particular vertices. A graph is said to be disconnected if its vertices can be partitioned into two distinct subsets that do not share any edges, otherwise a graph is said to be connected.

B.2 Undirected Graphs

An undirected graph also called simple graph is a graph where the edges are not oriented, therefore, the edge $\{1, 2\}$ is the same as the edge $\{2, 1\}$. A direct consequence of this is that for an undirected graph with n vertices the maximum number of edges is $\frac{n(n-1)}{2}$ if loops are not considered and $\frac{n(n+1)}{2}$ with loops. Two vertices linked by an edge are said to be adjacent, and a vertex is said to be incident with an edge and vice versa. Subsequently, a graph can be defined by its incidence matrix or adjacency matrix. The adjacency matrix of a simple graph is a symmetric binary matrix, the Figure B.1 represents a simple undirected graph with 10 vertices.

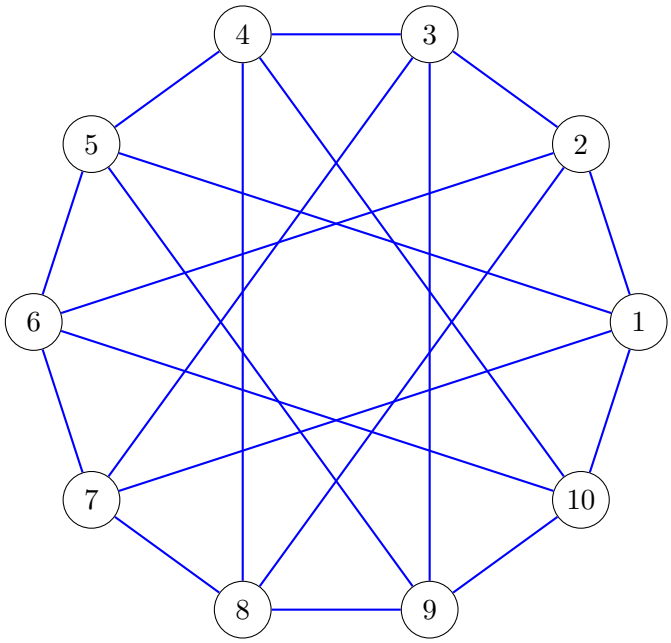

Figure B.1: Example of an undirected graph.

The adjacency matrix representation is not unique and any permutations of a row and a column with equal indexes will yield the same graph by changing the vertex labels. A possible adjacency matrix $\mathcal{A}$ for the graph presented in Figure B.1

is as follows,

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{B.2}$$

Similarly, a graph can be defined by its incidence matrix $\mathcal{I}$. This kind of matrix links each edge to the two vertices it connects. The incidence matrix of the graph B.1 is presented in equation (B.3). It can be noted that the adjacency matrix does not have to be symmetric or even square since the number of edges can be different from the number of vertices. In the same way as for the adjacency matrix, any permutations of the rows and the columns will represent the same graph after changing the labels of the vertices and of the edges respectively. Very often graphs have a lot more edges than vertices, subsequently in most cases the adjacency matrix constitutes a more compact way to store a graph than the incidence matrix. Hence,

it is the preferred representation in most cases.

$$\mathcal{I}^{\top} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.3}$$

Graphs are mathematical objects used to represent a given topology and consequently, the relative position of the vertices as well as the shape of the edges does not change the topological properties of a given graph. However, only plotting the undirected edges between a set of vertices can be insufficient sometimes, and in some cases associating an edge with a specific direction can carry some useful meaning. This property is achieved for the directed graphs and presented within the next section.

B.3 Directed Graphs

Each edge of a simple graph can be oriented from a vertex towards another in order to define a directed graph. An edge is called a loop if it connects a vertex to itself. A directed graph also called a digraph does not usually include any loops or parallel edges (multiple edges from and to the same vertex). The graph presented in Figure B.2 corresponds to the oriented adjacency matrix $\mathcal{A}$ given equation (B.4). In this case the adjacency matrix is still square and composed of binaries entries

but does not have to be symmetric any more. Indeed, the adjacency matrix of a digraph contains the information related to the incidence of the edges as well as their orientation. Any element of $\mathcal{A}$ indicates the number of edges starting from the vertex indexed by the row index and going to the vertex indexed by the column index. Directed graphs do not only inform on the topology but also on the direction of the edges between the vertices, therefore, they can be used to represent a succession of states or flows between vertices. In the example presented Figure B.2, each vertex is the initial vertex of two edges and the terminal vertex of two other edges.

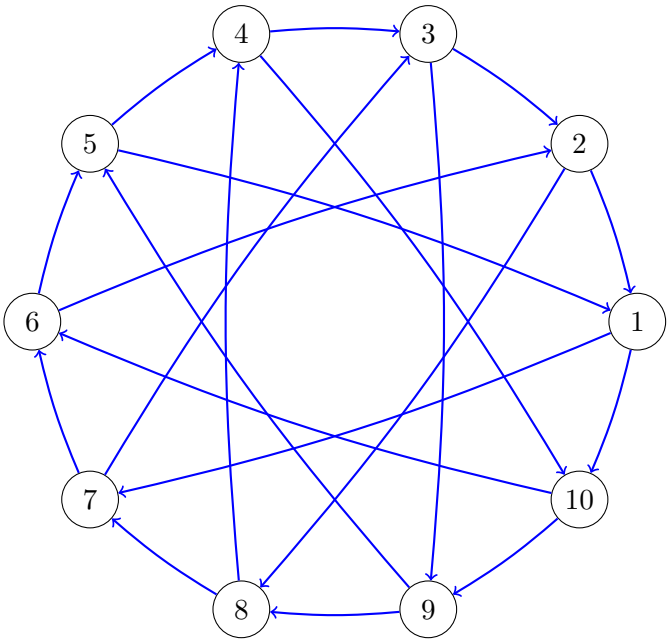

Figure B.2: Example of a directed graph.

Consequently, in this specific case the binary entries of the adjacency matrix $\mathcal{A}$ are positioned in such a way that each row and each column sum to two. A row sums to two for two edges leaving a vertex and a column sums to two for two edges

going to a vertex.

$$\mathcal{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{B.4}$$

The representation of a directed graph by its incidence matrix is done by adding signs to the matrix entries. An entry is set to -1 if an edge leaves a vertex and to 1 if an edge points towards a vertex, it is set to 0 otherwise.

B.4 Weighted Graphs

In some cases it is important to associate each oriented edge of a directed graph to a weight. Weighted graphs are used to define a certain distance closeness between two given vertices and are therefore essential to analyse the flow between vertices. In the same way as for the simple graphs and the directed graphs, weighted graphs could be defined by their incidence or adjacency matrix.

The adjacency matrix of a weighted graph has to contain the necessary information about the weights and the orientations of all the edges. A weight is located on the entry at the intersection of the row and the column, whose indexes are the index of the initial vertex and of the terminal vertex respectively. The entries of the incidence matrix are the weights of the edges with positive and negative signs, positive for an edge that is directed from a vertex and negative for an edge directed towards a vertex.

B.5 Multi-graphs

Finally, multi-graphs allow to have multiple weighted edges between two given vertices, also a given vertex can be connected to itself by a loop. Therefore, multi-graphs subsume the oriented and weighted graphs into one single type of graph. In the same way as before they can be represented by an adjacency matrix $\mathcal{A}$ as presented equation (B.5) or by an incidence matrix. The multi-graph linked to the

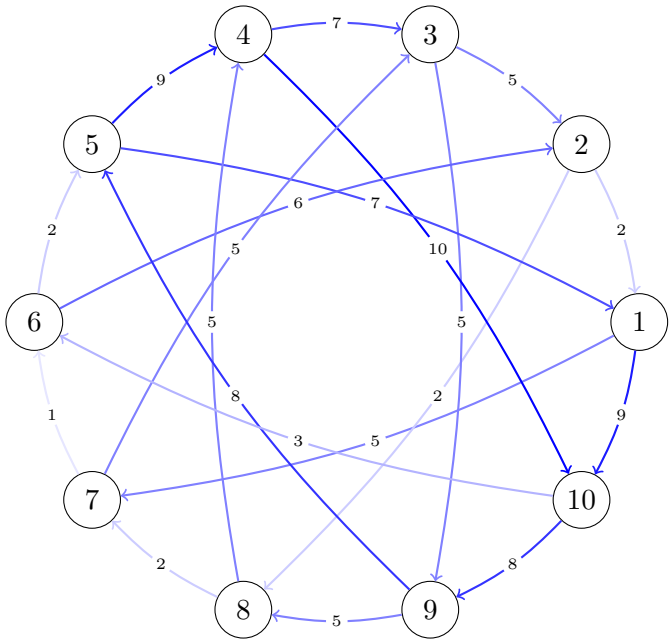

Figure B.3: Example of a weighted graph.

adjacency matrix $\mathcal{A}$ is shown in Figure B.4. The adjacency matrix contains the orientation of each edge as well as their respective weights, subsequently it is a square matrix linking the vertices with weights. The edges connect the vertices indexed by the row indexes to the ones indexed by the column indexes with the weight provided by the value of the matrix entry. The incidence matrix has the same definition as the incidence matrix for the weighted graphs, possibly including many parallel edges as well as loops.

$$\mathcal{A} = \begin{bmatrix} 1 & 5 & 0 & 0 & 1 & 0 & 5 & 0 & 0 & 5 \\ 2 & 2 & 1 & 0 & 0 & 5 & 0 & 2 & 0 & 0 \\ 0 & 5 & 3 & 2 & 0 & 0 & 2 & 0 & 5 & 0 \\ 0 & 0 & 7 & 4 & 5 & 0 & 0 & 2 & 0 & 5 \\ 7 & 0 & 0 & 7 & 5 & 5 & 0 & 0 & 1 & 0 \\ 0 & 5 & 0 & 0 & 2 & 6 & 5 & 0 & 0 & 1 \\ 2 & 0 & 5 & 0 & 0 & 1 & 7 & 5 & 0 & 0 \\ 0 & 7 & 0 & 5 & 0 & 0 & 2 & 8 & 1 & 0 \\ 0 & 0 & 5 & 0 & 7 & 0 & 0 & 5 & 9 & 1 \\ 5 & 0 & 0 & 5 & 0 & 1 & 0 & 0 & 1 & 10 \end{bmatrix} \tag{B.5}$$

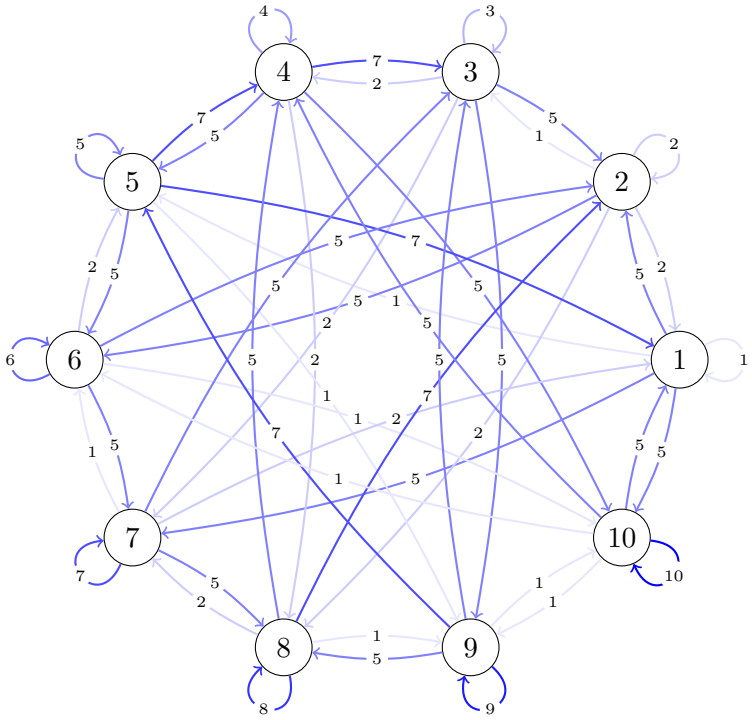

Figure B.4: Example of a weighted multi-graph.

Appendix C

Algorithmic Complexity

C.1 Introduction

The algorithmic complexity is measured by the time as well as the memory required in order to run an algorithm for an input of size n. The study of the complexity of an algorithm is important in order to evaluate the performance and be able to compare different algorithms (Garey and Johnson, 1979). The time complexity can be estimated based on the number of elementary operations that the algorithm has to perform for an inout of size n, relying on the assumption that the execution of each operation takes a fixed amount of time. On the other hand, the memory complexity also called the space complexity of an algorithm is the assessment of the space in memory required during the execution of an algorithm with an input of size n. The main point of interest in both the time and space complexity case is the asymptotic behaviour, in other words how does the complexity grow when the input size n tends to infinity. In reality, time and space complexity are also linked to the operating system, the programming language, the hardware, as well as many other parameters. However, since the study of complexity is focused on the algorithmic implementation based on the elementary operations, the complexity analysis become independent from the parameters mentioned previously. The study of algorithms and their complexity belongs to the field of computer science but they are relevant to the domains of control and systems engineering when one wants to assess the performance of a control algorithm. Finally, other aspects such as the algorithm power consumption, the bandwidth, the number of disk access could be considered as indicators of algorithm efficiency. Nonetheless, in general, only the time and space complexities are of interest.

C.2 Time Complexity

The time complexity of an algorithm is usually composed of a fixed part, completely independent of the input as well as a part directly linked to the size of the input denoted n. In most cases the fixed part is based on the initialization of variables and counters, these elementary operations have to be completed regardless of the input dimension. Analyzing the time complexity of an algorithm is performed based on the worst case scenario, when the algorithm has to complete the largest amount of elementary operations, before it can terminate. Different inputs of size n are likely to have different execution times, therefore the worst case scenario is assessed to provide an upper bound on the number of elementary operations required. Therefore, evaluating the time complexity of an algorithm is equivalent to expressing the number of elementary operations as a function of the input size, as follows,

$$t(n) = \mathcal{O}\left(f(n)\right), \tag{C.1}$$

where $t(n)$ denotes the number of elementary operations as a function of the input size n for the algorithm to execute in the worst case scenario. The function $f(n)$ describes the asymptotical behavior of $t(n)$. The notation $\mathcal{O}(\cdot)$ represents the Landau big O notation, and it is used to illustrate the complexity of the algorithm when n tends to infinity. For example, when $f(n) = n$ the complexity is said to be linear, and when $f(n) = n^k$ with $k \in \mathbb{N}^*$ the complexity is said to be polynomial. The evaluation of the time complexity of an algorithm is important in practice, especially in the case of online time-sensitive applications. An algorithm with poor performance could provide outdated outputs or use an important amount of computing resources. An algorithm with a time complexity at most polynomial is said to be tractable and fast, whereas a time complexity worst than polynomial is considered as slow. Different algorithm complexities are represented Figure C.1. Other less common time complexity measures consider the best and average cases, however these performance indexes are less used than the worst case analysis.

C.3 Space Complexity

The space complexity of an algorithm is another important performance index as it defines the amount of memory required in order to run the algorithm. Similarly to the time complexity, the space complexity is composed of a fixed part independent of the input as well as a part linked to the input size called the input space. The total amount of memory needed to execute an algorithm includes the instruction space to store the algorithm instructions, the environmental stack in the case where an

other routines are called, to store the variables while the other routines are executed and the data space that is used to store the variables. Subsequently, the assessment of the space complexity is achieved by finding the relation between the amount of memory needed and the input size, such that

$$s(n) = \mathcal{O}\left(g(n)\right), \tag{C.2}$$

where $s(n)$ denotes the amount of memory necessary to run the algorithm in the worst case scenario with an input of size n. The function $g(n)$ represents the asymptotical behaviour of the algorithm space complexity. In some cases, there is a possible trade-off between the time and the space complexities. In other words, the time complexity of an algorithm can be improved by using more space and vice versa. In practice, the space complexity is important to be able to select the appropriate hardware. Different asymptotical complexity behaviours are represented Figure C.1.

C.4 Polynomial versus Non-polynomial

The complexity of an algorithm is linked to the number of steps completed during the execution of the algorithm, in the case where the algorithm is used to solve a problem, the complexity of the best algorithm that exists can either be at most polynomial time or more complex. These problems are ranked into two main classes called Polynomial Deterministic (P) and Non-deterministic Polynomial (NP). Any problems belonging to the class P can be solved in polynomial time, and therefore, a given solution can also be verified with polynomial time complexity. A problem from the class NP cannot be solved easily, in polynomial time, however, a given solution can be checked in polynomial time. Consequently, any problems in the class P also belong to the class NP. Proving or disproving that the class P is equal to the class NP is still an open research question in the field of computer science (Cook, 1971). If P is different from NP, it would imply that there exists some problems that cannot be solved efficiently, in polynomial time, but whose solution can be checked quickly. On the contrary, if P and NP are equal, then new efficient algorithms can be developed in order to solve the problems that are believed to be currently difficult to tackle. The problems in NP can be solved with a complexity worst than polynomial, for example with an exponential or factorial complexity, often corresponding to the exhaustive search of the set of feasible solutions (Papadimitriou and Steiglitz, 1998). Some problems in NP have been shown to be at least as difficult to solve as the hardest problems in NP, these problems compose a new class called NP-complete. Finally, a larger class of problems called NP-hard gathers the problems that are at least as hard as the hardest problems in NP. By construction, any problem in NP reduces

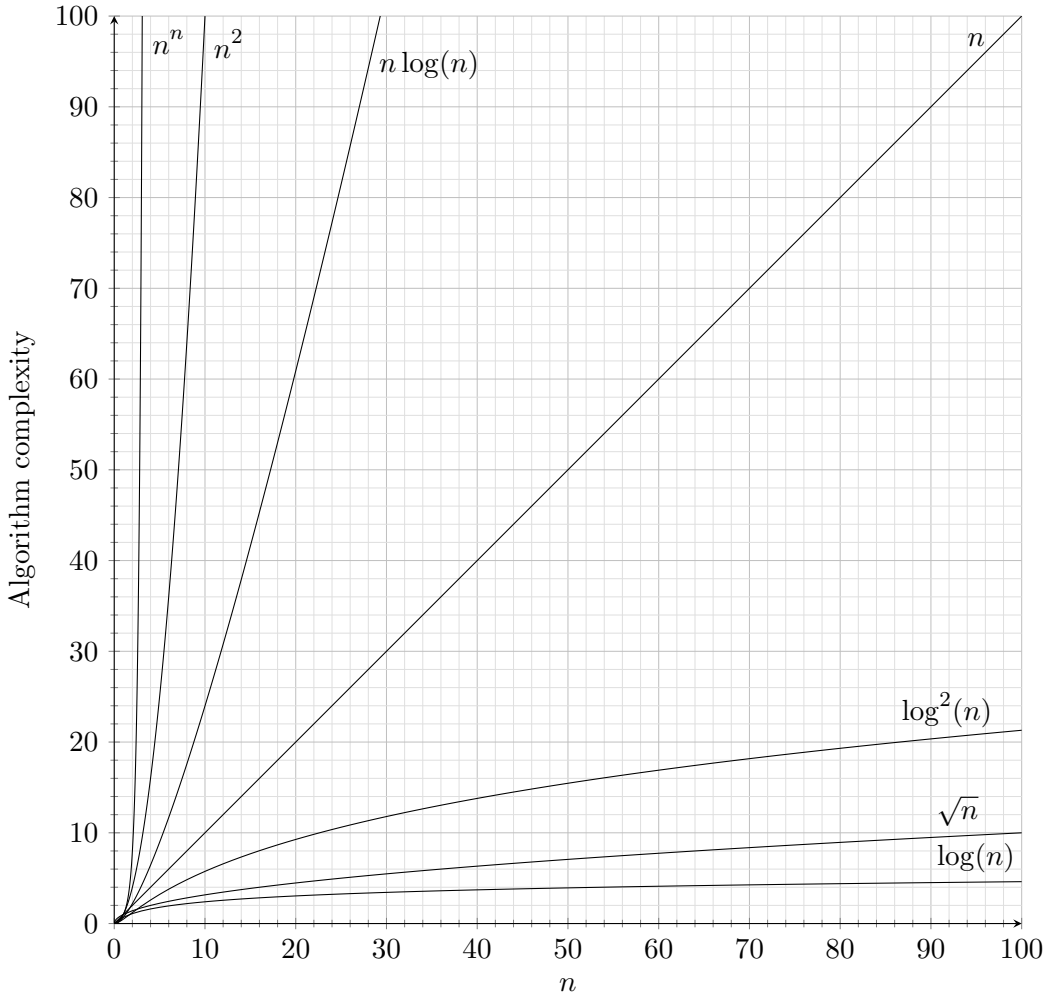

Figure C.1: Algorithm complexity against input size n.

in polynomial time to a NP-hard problem. Consequently, the class NP-complete can be perceived as the intersection between the class NP and the class NP-hard, since all NP-complete problems are in NP and are at least as difficult as the hardest problems belonging to NP.

References

Adelipour, S., Rastgar, M., and Haeri, M. (2017). Computational load reduction in model predictive control of nonlinear systems via decomposition. *Control, Instrumentation, and Automation (ICCIA), 2017 5th International Conference on*, pages 216–221.

Airbus (2014). Flying on demand 2014-2033, Global Market Forecast. Technical report, Airbus SE.

Al-gherwi, W., Budman, H., and Elkamel, A. (2010). Selection of control structure for distributed model predictive control in the presence of model errors. *Journal of Process Control*, 20:270–284.

Alessio, A., Barcelli, D., and Bemporad, A. (2011). Decentralized model predictive control of dynamically coupled linear systems. *Journal of Process Control*, 21:705–714.

Aström, K. J. and Kumar, P. R. (2014). Control: A perspective. *Automatica*, 50:3–43.

Austin Spang, H. and Brown, H. (1999). Control of jet engines. *Control Engineering Practice*, 7:1043–1059.

Azadi Yazdi, E. and Nagamune, R. (2011). A parameter set division and switching gain-scheduling controllers design method for time-varying plants. *Systems & Control Letters*, 60:1016–1023.

Babazadeh, M. and Nobakhti, A. (2016). Sparsity Promotion in State Feedback Controller Design. *IEEE Transactions on Automatic Control*, 62:4066–4072.

Bakule, L. (2008). Decentralized control: An overview. *Annual Reviews in Control*, 32:87–98.

Balas, E., Ceria, S., Cornuéjols, G., and Natraj, N. (1996). Gomory cuts revisited. *Operations Research Letters*, 19:1–9.

Balas, G. J. (2002). Linear, parameter-varying control and its application to a turbofan engine. *International Journal of Robust and Nonlinear Control*, 12:763–796.

Bartlett, R. A., Biegler, L. T., Backstrom, J., and Gopal, V. (2002). Quadratic programming algorithms for large-scale model predictive control. *Journal of Process Control*, 12:775–795.

Bay, J. S. (1998). *Fundamentals of linear state space systems.* McGraw-Hill, New York.

Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427.

Benito, A. and Alonso, G. (2018). *Energy Efficiency in Air Transportation.* Butterworth-Heinemann, Oxford.

Bondy, J. A. and Murty, U. S. R. (2008). *Graph Theory.* Springer-Verlag, London.

Boyd, S., El Ghaoui, L., Feron, E., and Balakrishnan, V. (1994). *Linear matrix inequalities in system and control theory.* SIAM, Philadelphia.

Boyd, S. and Vandenberghe, L. (2010). *Convex Optimization.* Cambridge University Press, Cambridge.

Branicky, M. (1998). Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43:475–482.

Brezhneva, O. A., Tret'yakov, A. A., and Wright, S. E. (2009). A simple and elementary proof of the Karush-Kuhn-Tucker theorem for inequality-constrained optimization. *Optimization Letters*, 3:7–10.

Bristol, E. H. (1966). On a new measure of interaction for multivariable process control. *IEEE Transactions on Automatic Control*, 11:133–134.

Browning, T. R. (2001). Applying the design structure matrix to system decomposition and integration problems: A review and new directions. *IEEE Transactions on Engineering Management*, 48:292–306.

Burlet, M. and Goldschmidt, O. (1997). A new and improved algorithm for the 3-cut problem. *Operations Research Letters*, 21:225–227.

Cagienard, R., Grieder, P., Kerrigan, E. C., and Morari, M. (2007). Move blocking strategies in receding horizon control. *Journal of Process Control*, 17:563–570.

Camacho, E. F. and Bordons, C. (2004). *Model predictive control*. Springer-Verlag, London.

Candès, E. J. (2008). The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathématique*, 346:589–592.

Candès, E. J. and Tao, T. (2005). Decoding by linear programming. *IEEE Transaction on Information Theory*, 51:4203–4215.

Candès, E. J., Wakin, M. B., and Boyd, S. P. (2008). Enhancing sparsity by reweighted l_1 minimization. *Journal of Fourier Analysis and Applications*, 14:877–905.

Cavalier, T. M., Pardalos, P. M., and Soyster, A. L. (1990). Modeling and integer programming techniques applied to propositional calculus. *Computers & Operations Research*, 17:561–570.

Chen, D. and Seborg, D. E. (2003). Design of decentralized PI control systems based on Nyquist stability analysis. *Journal of Process Control*, 13:27–39.

Christofides, P. D., Scattolini, R., Muñoz de la Peña, D., and Liu, J. (2013). Distributed model predictive control: A tutorial review and future research directions. *Computers & Chemical Engineering*, 51:21–41.

Cook, S. A. (1971). The complexity of theorem-proving procedures. *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158.

Crusius, C. A. R. and Trofino, A. (1999). Sufficient LMI conditions for output feedback control problems. *IEEE Transactions on Automatic Control*, 44:1053–1057.

Daafouz, J. and Bernussou, J. (2001). Parameter Dependent Lyapunov Functions for Discrete Time Systems With Time-Varying Parametric Uncertainties. *Systems & Control Letters*, 43:355–359.

Daafouz, J., Riedinger, P., and Iung, C. (2002). Stability analysis and control synthesis for switched systems: a switched Lyapunov function approach. *IEEE Transactions on Automatic Control*, 47:1883–1887.

Dantzig, G. B., Orden, A., and Wolfe, P. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5:183–195.

De Oliveira, M. C., Bernussou, J., and Geromel, J. C. (1999). A new discrete-time robust stability condition. *Systems & Control Letters*, 37:261–265.

Diestel, R. (2000). *Graph Theory*. Springer-Verlag, Heidelberg.

Doelman, R. and Verhaegen, M. (2016). Sequential convex relaxation for convex optimization with bilinear matrix equalities. *2016 European Control Conference*, pages 1946–1951.

El Ghaoui, L. and Niculescu, S.-I. (2000). *Advances in linear matrix inequality methods in control: advances in design and control*. SIAM, Philadelphia.

Emedi, Z. and Karimi, A. (2016). Fixed-structure LPV discrete-Time controller design with induced l 2-norm and performance. *International Journal of Control*, 89:494–505.

Eren, U., Prach, A., Koçer, B. B., Raković, S. V., Kayacan, E., and Açıkmeşe, B. (2017). Model Predictive Control in Aerospace Systems: Current State and Opportunities. *Journal of Guidance, Control, and Dynamics*, 40:1541–1566.

Fourier, J.-B. J. (1827). Analyse des travaux de l'Académie Royale des sciences, pendant l'année 1824, Partie Mathématique. *Histoire de l'Académie Royale des Sciences de l'institut de France*, 7:xlvii–lv.

Fukuda, M. and Kojima, M. (2001). Branch-and-cut algorithms for the bilinear matrix inequality eigenvalue problem. *Computational Optimization and Applications*, 19:79–105.

Gahinet, P. and Apkarian, P. (1994). A linear matrix inequality approach to H_∞ control. *International Journal of Robust and Nonlinear Control*, 4:421–448.

Gahinet, P., Nemirovski, A., Laub, A. J., and Chilali, M. (1995). *LMI Control Toolbox for Use with Matlab*. The MathWorks Inc., Natick.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability*. W. H. Freeman and Company, New York.

Geromel, J. C. and Colaneri, P. (2006). Stability and stabilization of discrete time switched systems. *International Journal of Control*, 79:719–728.

Gilbert, W., Henrion, D., Bernussou, J., and Boyer, D. (2010). Polynomial LPV synthesis applied to turbofan engines. *Control Engineering Practice*, 18:1077–1083.

Goh, K. C., Safonov, M. G., and Papavassilopoulos, G. P. (1995). Global optimization for the biaffine matrix inequality problem. *Journal of Global Optimization*, 7:365–380.

Goldberg, M. and Zwas, G. (1974). On matrices having equal spectral radius and spectral norm. *Linear Algebra and Its Applications*, 8:427–434.

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278.

Gorski, J., Pfeuffer, F., and Klamroth, K. (2007). Biconvex sets and optimization with biconvex functions: A survey and extensions. *Mathematical Methods of Operations Research*, 66:373–407.

Gross, D. and Stursberg, O. (2016). A cooperative distributed MPC algorithm with event-based communication and parallel optimization. *IEEE Transactions on Control of Network Systems*, 3:275–285.

Guicherd, R., Trodden, P. A., Mills, A. R., and Kadirkamanathan, V. (2017). Weak interactions based system partitioning using integer linear programming. *IFAC-PapersOnLine*, 50:3698–3704.

Guicherd, R., Trodden, P. A., Mills, A. R., and Kadirkamanathan, V. (2019). Supervised-distributed Control with Joint Performance and Communication Optimization. *International Journal of Control*. In Preparation.

Guicherd, R., Trodden, P. A., Mills, A. R., and Kadirkamanathan, V. (2020). Supervised-distributed Control with Joint Performance and Communication Optimization for LPV Systems. *Automatica*. In Preparation.

Hanifzadegan, M. and Nagamune, R. (2014). Smooth switching LPV controller design for LPV systems. *Automatica*, 50:1481–1488.

Hespanha, J. and Morse, A. (1999). Stability of switched systems with average dwell-time. *Proceedings of the 38th IEEE Conference on Decision and Control*, 3:2655–2660.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (2001). *Fundamentals of Convex Analysis*. Springer-Verlag, New York.

Hurley, N. and Rickard, S. (2009). Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55:4723–4741.

IBM corp. (2016). *IBM ILOG CPLEX Optimization Studio CPLEX User's Manual. Version 12.7*.

Jamoom, M. B., Feron, E., and McConley, M. W. (1998). Optimal distributed actuator control grouping scheme. *Proceedings of the* 37^{th} *IEEE Conference on Decision & Control*, 2:1900–1905.

Jaw, L. C. and Mattingly, J. D. (2009). *Aircraft Engine Controls: Design, System Analysis, and Health Monitoring*. AIAA, Reston.

Kalman, R. E. (1960). Contributions to the Theory of Optimal Control. *Bolentin de la Sociedad Matematica Mexicana*, 5:102–119.

Kariwala, V., Forbes, J. F., and Meadows, E. S. (2003). Block relative gain: properties and pairing rules. *Industrial & Engineering Chemistry Research*, 42:4564–4574.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer-Verlag.

Khachiyan, L. (1980). Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20:53 – 72.

Klee, V. and Minty, G. J. (1972). How good is the simplex algorithm? *Proceedings of the Third Symposium on Inequalities*, pages 159–179.

Kothare, M. V., Balakrishnan, V., and Morari, M. (1996). Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32:1361–1379.

Kwakernaak, H. (1993). Robust control and H_∞-optimization - Tutorial paper. *Automatica*, 29:255–273.

Land, A. H. and Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28:497–520.

Langson, W., Chryssochoos, I., Raković, S. V., and Mayne, D. Q. (2004). Robust model predictive control using tubes. *Automatica*, 40:125–133.

Laplace, P. S. (1820). *Théorie analytique des probabilités*. Courcier, Paris.

Laub, A. J. (1979). A Schur method for solving algebraic Riccati equations. *IEEE Transactions on Automatic Control*, 24:913–921.

Lawrence, D. A. and Rugh, W. J. (1995). Gain scheduling dynamic linear controllers for a nonlinear plant. *Automatica*, 31:381–390.

Leininger, G. G. (1979). Diagonal dominance for multivariable Nyquist array methods using function minimisation. *Automatica*, 15:339–345.

Leith, D. J. and Leithead, W. E. (2000). Survey of gain-scheduling analysis and design. *International Journal of Control*, 73:1001–1025.

Lepschy, A. M., Mian, G. A., and Viaro, U. (1992). Feedback Control in Ancient Water and Mechanical Clocks. *IEEE Transactions on Education*, 35:3–10.

Liberzon, D. (2003). *Switching in Systems and Control*. Birkhauser, Boston.

Lin, H. and Antsaklis, P. J. (2009). Stability and stabilizability of switched linear systems: A survey of recent results. *IEEE Transactions on Automatic Control*, 54:308–322.

Linke-Diesinger, A. (2008). *Systems of commercial turbofan engines: an introduction to systems functions*. Springer-Verlag, Berlin Heidelberg.

Löfberg, J. (2004). YALMIP: A toolbox for modeling and optimization in MATLAB. *2004 IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289.

Lu, B. and Wu, F. (2004). Switching LPV control designs using multiple parameter-dependent Lyapunov functions. *Automatica*, 40:1973–1980.

Lunze, J. (1992). *Feedback Control of Large-Scale Systems*. Prentice Hall, Hempstead.

Lyapunov, A. M. (1992). The general problem of stability of motion. *International Journal of Control*, 55:531–534.

Maciejowski, J. M. (2002). *Predictive control with constraints*. Prentice Hall, Harlow.

Maestre, J., Muñoz de la Peña, D., Camacho, E., and Alamo, T. (2011). Distributed model predictive control based on agent negotiation. *Journal of Process Control*, 21:685–697.

Maestre, J. M., Muñoz de la Peña, D., and Camacho, E. F. (2009). A distributed MPC scheme with low communication requirements. *Proceedings of the American Control Conference*, 1:2797–2802.

Maestre, J. M., Muñoz de la Peña, D., Jiménez Losada, A., Algaba, E., and Camacho, E. F. (2014). A coalitional control scheme with applications to cooperative game theory. *Optimal Control Applications and Methods*, 35:592–608.

Maestre, J. M. and Negenborn, R. R. (2013). *Distributed Model Predictive Control Made Easy*. Springer-Verlag, Dordrecht.

Manousiouthakis, V., Savage, R., and Arkun, Y. (1986). Synthesis of decentralized process control structures using the concept of block relative gain. *American Institute of Chemical Engineers*, 32:991–1003.

Mason, P., Sigalotti, M., and Daafouz, J. (2007). On stability analysis of linear discrete-time switched systems using quadratic Lyapunov functions. *2007 46th IEEE Conference on Decision and Control*, pages 5629–5633.

Mayne, D., Rawlings, J., Rao, C., and Scokaert, P. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814.

Mayne, D. Q. (2014). Model predictive control: Recent developments and future promise. *Automatica*, 50:2967–2986.

Mayr, O. (1970). *The origins of feedback control*. MIT Press, Cambridge.

Merrill, W., Kim, J.-H., Lall, S., Majerus, S., Howe, D., and Behbahani, A. R. (2010). Distributed Engine Control Design Considerations. *Proceedings of the 46th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, pages 1–13.

Mesarović, M. D., Macko, D., and Takahara, Y. (1970). *Theory of Hierarchical, Multilevel, Systems*. Academic Press, New York.

Morari, M. and Lee, J. H. (1999). Model predictive control : past , present and future. *Computers and Chemical Engineering*, 23:667–682.

Moroşan, P.-D., Bourdais, R., Dumur, D., and Buisson, J. (2010). Building temperature regulation using a distributed model predictive control. *Energy and Buildings*, 42:1445–1452.

MOSEK ApS (2017). *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*

Motee, N. and Sayyar-Rodsari, B. (2003). Optimal partitioning in distributed model predictive control. *Proceedings of the American Control Conference*, 6:5300–5305.

Negenborn, R. and Maestre, J. (2014). Distributed Model Predictive Control: An Overview and Roadmap of Future Research Opportunities. *IEEE Control Systems*, 34:87–97.

Nesterov, Y. and Nemirovskii, A. (1994). *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, Philadelphia.

Ocampo-Martinez, C., Bovo, S., and Puig, V. (2011). Partitioning approach oriented to the decentralised predictive control of large-scale systems. *Journal of Process Control*, 21:775–786.

Pannocchia, G., Rawlings, J. B., and Wright, S. J. (2007). Fast, large-scale model predictive control by partial enumeration. *Automatica*, 43:852–860.

Pannocchia, G., Rawlings, J. B., and Wright, S. J. (2011). Conditions under which suboptimal nonlinear MPC is inherently robust. *System & Control Letters*, 60:747–755.

Papadimitriou, C. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications Inc., New York.

Papadimitriou, C. H. (1981). On the complexity of integer programming. *Journal of the Association for Computing Machinery*, 28:765–768.

Polyak, B., Khlebnikov, M., and Shcherbakov, P. (2013). An LMI approach to structured sparse feedback design in linear control systems. *European Control Conference*, pages 833–838.

Potra, F. A. and Wright, S. J. (2000). Interior-point methods. *Journal of Computational and Applied Mathematics*, 124:281–302.

Qin, S. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764.

Qiu, W., Vittal, V., and Khammash, M. (2004). Decentralized Power System Stabilizer Design Using Linear Parameter Varying Approach. *IEEE Transactions on Power Systems*, 19:1951–1960.

Rawlings, J. B. and Mayne, D. Q. (2009). *Model Predictive Control : Theory and Design*. Nob Hill Publishing, Madison.

Recht, B., Fazel, M., and Parrilo, P. A. (2007). Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization. *SIAM Review*, 52:471–501.

Richalet, J. (1993a). Industrial applications of model based predictive control. *Automatica*, 29:1251–1274.

Richalet, J. (1993b). *Pratique de la commande prédictive*. Hermes, Cachan.

Richalet, J., Rault, A., Testud, J., and Papon, J. (1978). Model predictive heuristic control: applications to industrial processes. *Automatica*, 14:413–428.

Richards, A. and How, J. P. (2007). Robust distributed model predictive control. *International Journal of Control*, 80:1517–1531.

Richter, H. (2012). *Advanced control of turbofan engines*. Springer-Verlag, New York.

Riordan, J. (1958). *An introduction to combinatorial analysis*. Wiley, New York.

Rockafellar, R. T. (1970). *Convex Analysis*. Princeton University Press, New Jersey.

Rockafellar, R. T. (1993). Lagrange multipliers and optimality. *SIAM review*, 35:183–238.

Rossiter, J. (2003). *Model-based predictive control: a practical approach*. CRC Press, Boca Raton.

Rotkowitz, M. and Lall, S. (2006). A characterization of convex problems in decentralized control. *IEEE Transactions on Automatic Control*, 51:274–286.

Rugh, W. (1996). *Linear system theory*. Prentice-Hall, Upper Saddle River.

Rugh, W. J. and Shamma, J. S. (2000). Research on gain scheduling. *Automatica*, 36:1401–1425.

Scattolini, R. (2009). Architectures for distributed and hierarchical model predictive control - A review. *Journal of Process Control*, 19:723–731.

Schrijver, A. (1998). *Theory of linear and integer programming*. John Wiley & Sons, Chichester.

Schuler, S., Münz, U., and Allgöwer, F. (2014). Decentralized state feedback control for interconnected systems with application to power systems. *Journal of Process Control*, 24:379–388.

Scokaert, P. and Mayne, D. (1998). Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43:1136–1142.

Seok, J., Kolmanovsky, I., and Girard, A. (2017). Coordinated Model Predictive Control of Aircraft Gas Turbine Engine and Power System. *Journal of Guidance, Control, and Dynamics*, 10:2538–2555.

Shor, N. Z. (1977). Cut-off method with space extension in convex programming problems. *Cybernetics*, 13:94–96.

Šiljak, D. D. (1991). *Decentralized control of complex systems*. Academic Press, San Diego.

Sobey, A. and Suggs, A. (1963). *Control of aircraft and missile powerplants*. John Wiley & Sons, New York.

Sontag, E. (1998). *Mathematical Control Theory*. Springer-Verlag, New York.

Stewart, B. T., Venkat, A. N., Rawlings, J. B., Wright, S. J., and Pannocchia, G. (2010). Cooperative distributed model predictive control. *Systems & Control Letters*, 59:460–469.

Sturm, J. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653.

Thompson, H. (1992). *Parallel processing for jet engine control*. Springer-Verlag, London.

Thompson, H., Chipperfield, A., Fleming, P., and Legge, C. (1999). Distributed aero-engine control systems architecture selection using multi-objective optimisation. *Control Engineering Practice*, 7:655–664.

Trodden, P. A. (2014). Feasible parallel-update distributed MPC for uncertain linear systems sharing convex constraints. *Systems and Control Letters*, 74:98–107.

Trodden, P. A. and Richards, A. (2013). Cooperative distributed MPC of linear systems with coupled constraints. *Automatica*, 49:479–487.

VanAntwerp, J. G. and Braatz, R. D. (2000). Tutorial on linear and bilinear matrix inequalities. *Journal of Process Control*, 10:363–385.

Vandenberghe, L. and Boyd, S. (1996). Semidefinite Programming. *SIAM Review*, 38:49–95.

Veselý, V., Rosinová, D., and Ilka, A. (2013). Decentralized gain-scheduling controller design: Polytopic system approach. *IFAC-PapersOnline*, 13:401–406.

Wada, N., Saito, K., and Saeki, M. (2006). Model Predictive Control for Linear Parameter Varying Systems using Parameter Dependent Lyapunov Function. *IEEE Transaction on Circuits and Systems - II: Express Briefs*, 53:1446–1450.

Whittle, F. (1945). The first James Clayton lecture: the early history of the Whittle jet propulsion gas turbine. *Proceedings of the institution of mechanical engineers*, 152:419–435.

Wie, B. and Bernstein, D. S. (1992). A Benchmark Problem for Robust Control Design. *Journal of Guidance, Control, and Dynamics*, 15:1057–1059.

Williams, P. H. (2013). *Model building in mathematical programming*. Wiley, Chichester.

Xiao, L., Johansson, M., Hindi, H., Boyd, S., and Goldsmith, A. (2003). Joint optimization of communication rates and linear systems. *IEEE Transactions on Automatic Control*, 48:148–153.

Xie, L., Cai, X., Chen, J., and Su, H. (2016). GA based decomposition of large scale distributed model predictive control systems. *Control Engineering Practice*, 57:111–125.

Yang, D., Zhao, J., Yang, D., and Zhao, J. (2018). H_∞ bumpless transfer for switched LPV systems and its application. *International Journal of Control*, 91:1–14.

Ye, W. and Heidemann, J. (2002). An energy-efficient MAC protocol for wireless sensor networks. *Proceedings of the IEEE Infocom*, pages 1567–1576.

Zhao, P. and Nagamune, R. (2018). Switching linear parameter-varying control with improved local performance and optimized switching surfaces. *International Journal of Robust and Nonlinear Control*, 28:3403–3421.

Zhou, K., Doyle, J. C., and Glover, K. (1996). *Robust and Optimal Control*. Prentice Hall, Upper Saddle River.