A Step Towards Distributed Control of Massive-scale Networks

João Miguel Oliveira Domingos

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. João Manuel de Freitas Xavier

Examination Committee
Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. João Manuel de Freitas Xavier
Member of the Committee: Prof. João Manuel Lage de Miranda Lemos

June 2018
“A basic rule of mathematical life: if the universe hands you a hard problem, try to solve an easier one instead, and hope the simple version is close enough to the original problem that the universe doesn’t object.” — Jordan Ellenberg
Declaration

Declaration I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.
Acknowledgments

Foremost, I would like to highlight Prof. João Xavier for his outstanding teaching skills, noticeable ability to clarify any doubt (no matter how dumb) and availability to answer my (near uncountable) questions. Without his patience, insightful comments, vision and clarity none of this would have been possible.

Thank you, João for proof reading some of the early sections, Kishan for the ongoing discussions and exciting chess matches, Ricardo for the all the, clearly non academic, interesting debates we had and, Beatriz for enduring my peculiar personality and character.

Last but not least, a special thanks to my family: to my father for always being available to help me in any situation, my mother for making my life (a lot) easier, my sister for her support and concern and (yes) my dog for giving me reasons to take small work breaks.

As promised, this thesis is dedicated to my uncle, Amândio, who always believed in me and, for an early age, deeply respected and encouraged my academic path.
Resumo

A estabilização de um sistema dinâmico implica conduzi-lo a um ponto de operação pretendido, após ter sofrido uma perturbação momentânea. O sistema é estabilizado pelo controlador: um dispositivo que, através de sensores observa o estado do sistema e, através de actuadores age sobre o mesmo. No caso centralizado, o controlador tem acesso ao estado completo. Esta hipótese é razoável para sistemas pouco complexos mas nas aplicações actuais - rede energética, tráfego urbano, controlo de epidemias, veículos robóticos - deixa de se verificar. Assim sendo, no caso distribuído o controlador deve agir tendo em conta informação parcial. Esta mudança, reflecte-se matematicamente num padrão de esparsidade imposto no controlador. No caso centralizado o problema pode ser resolvido usando optimização convexa contudo, considerando esparsidade o problema torna-se não convexo e NP difícil.

Nesta dissertação estudam-se três problemas: (1) desenho centralizado de controladores esparsos e estabilizadores, usando heurísticas para o problema não convexo; (2) desenho de esparsidades que aumentam a estabilidade do sistema em malha fechada; (3) desenho distribuído de controladores esparsos e estabilizadores. Para (1) os nossos métodos conseguem estabilizar 40% – 77% de sistemas com esparsidades de 50%, isto é, com metade dos graus de liberdade. Relativo a (2) conseguimos desenhar esparsidades que aumentam em 28% a percentagem de estabilização, face a um desenho puramente aleatório. Referente a (3), desenhámos um algoritmo distribuído com garantias teóricas de performance, à custa de considerarmos sistemas puramente positivos.

Palavras-chave: Estabilização Esparsa, Estabilização Distribuída, Desenho de Esparsidade, Desigualdades matriciais lineares, Desigualdades matriciais bilineares
Abstract

Stabilizing a dynamical system implies conducting it to an intended operating point, after a momentary disturbance. The system is stabilized by the controller: a device that observes the state through sensors and acts on it through actuators. In many centralized cases, the controller has access to the complete state. This hypothesis is reasonable for low complexity systems, but in current applications - power grid, urban traffic, epidemic control, robotic vehicles - this no longer holds true. Therefore, in the distributed case the controller must act taking into account partial information. Mathematically a sparsity pattern is imposed on the controller. In the centralized case the problem can be solved using convex optimization; however, to handle the sparsity constraints the problem becomes nonconvex and \( \mathcal{NP} \) hard.

In this thesis we study three problems: (1) centralized design of stabilizing sparse controllers, based on heuristics for the non convex problem; (2) design of sparsity patterns that lead to the most stable closed loop system; (3) decentralized design of stabilizing sparse controllers. For (1) our methods can stabilize 40% – 77% of systems with sparsities of 50%, i.e., with half the degrees of freedom. Relative to (2) we were able to draw sparsities that increase the stabilization percentage by 28%, comparing against a purely randomized design. Referring to (3), we have designed a distributed algorithm with theoretical guarantees of performance, at the expense of considering purely positive systems.

**Keywords:** Sparse Stabilization, Decentralized Stabilization, Sparsity Design, Linear Matrix Inequalities, Bilinear Matrix Inequalities
# Contents

Declaration ................................................................. v  
Acknowledgments .......................................................... vii 
Resumo .............................................................................. ix  
Abstract ........................................................................... xi  
List of Tables ................................................................. xv  
List of Figures ............................................................... xvii  
Acronyms .......................................................................... xix  

## 1 Introduction
  1.1 Motivation for Stabilization .......................................... 1  
  1.2 Massive Scale Disclaimer .............................................. 1  
  1.3 Structure of the Document ............................................ 2  

## 2 Centralized Computation of Decentralized Controllers
  2.1 Introduction ............................................................... 3  
  2.2 Centralized Background ............................................... 4  
  2.3 Decentralized Formulation .......................................... 9  
    2.3.1 First Motivation .................................................. 10  
    2.3.2 Second Motivation ............................................. 12  
  2.4 Decentralized Heuristics ............................................. 13  
    2.4.1 State of the Art ................................................. 14  
    2.4.2 Big Picture ....................................................... 20  
    2.4.3 Incremental Inverse Algorithm ............................. 21  
    2.4.4 Decentralized Convex-Concave Approach ............... 26  
    2.4.5 Incremental PK and Variant ............................... 34  
  2.5 Numerical Section ..................................................... 38  
    2.5.1 Reduction Mode vs Original CCP ......................... 39  
    2.5.2 Numerical Comparison of Heuristics ..................... 40  

## 3 Optimal Sparsity Patterns
  3.1 Introduction ............................................................. 43  
  3.2 Problem Formulation ................................................... 43
List of Tables

2.1 Comparison of heuristics: Standard Gaussian Generation. . . . . . . . . . . . . . . . . . . 41

3.1 Comparison of Oracles: Standard Gaussian Generation with budget $= [8, 6, 3]^T$. . . . . . . . . . 59

3.2 Comparison of oracle $O_{ij}$: different $N_{oracle}$ values. . . . . . . . . . . . . . . . . . . . . . . 60

3.3 $N$ as a function of $(n, m)$ for a $n - 1$ control budget. . . . . . . . . . . . . . . . . . . . . . . 61

3.4 Developed oracles vs Exhaustive search table. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62

4.1 Centralized vs Decentralized controller Synthesizes: data of figures 4.2 and 4.3. . . . . . 75
List of Figures

2.1 Block diagram of a LTI system. .................................................. 4
2.2 Block diagram of a LTI controller. .................................................. 4
2.3 Bisection algorithm for \( \delta = 0.1 \). .............................................. 8
2.4 Home temperature example: centralized case with set \( \mathcal{K}_f \equiv \mathbb{R}^{m \times n} \). .................................................. 11
2.5 Home temperature example: decentralized case with set \( \mathcal{K}_f \). .................................................. 12
2.6 Schematic representation of ideas and algorithms for decentralized stability. .................................................. 21
2.7 Geometrical interpretation of the differential, page 93 of [35]. .................................................. 28
2.8 CCP vs Reduction strategy. .................................................. 40
2.9 Standard Gaussian Generation and \( L = 10 \). .................................................. 41
2.10 Standard Gaussian Generation and \( L = 15 \). .................................................. 41
3.1 Standard Gaussian Generation with budget = \([8, 6, 3]^T\). .................................................. 58
3.2 Sparsity patterns for generation 51. .................................................. 59
3.3 Same Generation and budget, now with \( N_{oracle} = 3 \). .................................................. 60
3.4 Developed oracles vs Exhaustive search. .................................................. 62
4.1 Decentralized design of decentralized controllers. \( d_1/d_2 \) are the mentioned delay types. .................................................. 65
4.2 Achieved results with \( (N_{DPDS}, N_c, n, m, a, b, p_{edge}) = (100, 20, 5, 2, 0, 0.5, 0.5) \). .................................................. 74
4.3 Achieved results with \( (N_{DPDS}, N_c, n, m, a, b, p_{edge}) = (600, 20, 4, 3, -0.5, 0, 0.4) \). .................................................. 75
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMM</td>
<td>Alternating Direction Method of Multipliers</td>
</tr>
<tr>
<td>BI-PK</td>
<td>Boosted Incremental PK heuristic</td>
</tr>
<tr>
<td>BMI</td>
<td>Bilinear Matrix Inequality</td>
</tr>
<tr>
<td>CCP</td>
<td>Convex Concave Procedure</td>
</tr>
<tr>
<td>DCC</td>
<td>Decentralized Convex Concave heuristic</td>
</tr>
<tr>
<td>DC</td>
<td>Difference of Convex</td>
</tr>
<tr>
<td>DOF</td>
<td>Degrees of Freedom</td>
</tr>
<tr>
<td>DPDSC</td>
<td>Decentralized Primal Dual Synthesizer with Consensus</td>
</tr>
<tr>
<td>DPDS</td>
<td>Decentralized Primal Dual Synthesizer</td>
</tr>
<tr>
<td>HIFOO</td>
<td>$H_{\infty}-H_2$ Fixed Order Optimization</td>
</tr>
<tr>
<td>I-PK</td>
<td>Incremental PK heuristic</td>
</tr>
<tr>
<td>II-Bisection</td>
<td>Incremental Inverse heuristic with Bisection</td>
</tr>
<tr>
<td>II-Fixed</td>
<td>Incremental Inverse heuristic with Fixed bound</td>
</tr>
<tr>
<td>II-Reduction</td>
<td>Incremental Inverse heuristic with Reduction strategy</td>
</tr>
<tr>
<td>LMI</td>
<td>Linear Matrix Inequality</td>
</tr>
<tr>
<td>LQR</td>
<td>Linear Quadratic Regulator</td>
</tr>
<tr>
<td>LTI</td>
<td>Linear Time Invariant</td>
</tr>
<tr>
<td>NP</td>
<td>Non-deterministic Polynomial time</td>
</tr>
<tr>
<td>PBH</td>
<td>Popov–Belevitch–Hautus</td>
</tr>
<tr>
<td>SDP</td>
<td>Semidefinite Program</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation for Stabilization

The stabilization problem, alone, has little interest for the control community. The most interesting problems are mostly related with reference tracking (integral control), stochastic disturbance rejection ($H_2$, $H_\infty$, mixed $H_2/H_\infty$ norms), minimization of control/state efforts (LQR - Linear Quadratic Regulator), robust control (polytopic and feedback-uncertain systems) and the generalization of standard centralized theory to a distributed environment. All these approaches require a stabilizing controller that is further designed to achieve other control objectives. Stabilization could be seen as the most simple control problem. However, decentralized stabilization is $NP$ hard; hence there is a need to first study this problem to then evolve to most appealing/realistic applications. In [1], stabilization with decentralized static controllers that receive output measurements, was mentioned as one of the three major open problems in systems and control theory. In this dissertation, we approach some variants of this problem using a state space model of the underlying dynamical system. Regarding terminology, the terms decentralized and distributed are used interchangeably.

1.2 Massive Scale Disclaimer

Before proceeding it is important to warn the reader regarding the “massive scale” aspect of this thesis. This term indicates that we are considering network systems for which no central entity exists and, hence, there is an intrinsic need for decentralized control laws, i.e., strategies computed from partial information. This feature is of course present in applications like traffic control of an entire city, where there exist an enormous number of involved variables. Our methods are designed to deal with the partial information aspect of such systems. However, our numerical experiments are executed in very low dimensional problems such that they are computational traceable by standard software and personal computers. We will not simulate a network with 1000 nodes and, hence, make no claims regarding the massive scale scalability of the developed ideas. This justifies the title of the thesis: “A Step Towards Distributed Control of Massive-scale Networks”.

1
1.3 Structure of the Document

This thesis has a somewhat unusual structure. Instead of dividing our work in (1) state of the art (2) theoretical analysis and (3) results/discussion, we have decided to create three main chapters each with the enunciated items. This option arose naturally, since we approach three different problems each with its own literature and ideas. From our perspective, this allows a more insightful reading experience. The three main chapters are:

i) Chapter 2 - Centralized Computation of Decentralized Controllers: Given an assignment of state variables to control components, how to design decentralized stabilizing controllers in a centralized manner, i.e., by centralized computations.

ii) Chapter 3 - Optimal Sparsity Patterns: How to compute an optimal assignment of state variables to control components.

iii) Chapter 4 - Decentralized Computation of Decentralized Controllers: Given an assignment of state variables to control components, how to design decentralized stabilizing controllers in a decentralized manner, i.e., by decentralized computations.

In chapter 4, we do not present a state of the art section since no relatable literature was found. Chapter 5 drafts the most important conclusions together with some contributions and future research directions, for the field of decentralized control. The interested reader is referred to appendixes A, B and C where some useful extensions are presented and we prove convergence of DPDS: the major step towards the decentralized stabilizing algorithm presented in chapter 4 (DPDSC).
Chapter 2

Centralized Computation of
Decentralized Controllers

2.1 Introduction

This chapter tackles the decentralized stabilization problem, in which we want to reject non-zero initial
conditions optimally, i.e., in the minimal amount of time while having to compute the control signal from
partial information. This signal will be generated by the controller, which is computed from a centralized
strategy, i.e., from a single computational entity. This problem is known to be computationally challenging
[2], [3] hence there is a need for suitable heuristics. Using Linear Matrix Inequalities we derive several
computationally efficient algorithms, that can be employed to search for optimal static controllers, under
any sparsity pattern and considering the feedback of state variables. In reality all derived approaches
can be employed to search for controllers in any convex set, i.e., the decentralized setup is only a pos-
sible application of the ideas presented here.

The remainder of the chapter is organized as follows: in section 2.2 we provide the necessary back-
ground regarding the centralized stabilization problem and restrict ourselves to a specific framework -
state feedback static controllers. This allows us to explain all major heuristics and the extension to a
more general setup is done in appendix A.1, where we design output feedback dynamic controllers. The
decentralized formalism is introduced in section 2.3, where we present two concrete motivating exam-
pies for partial information control laws. In section 2.4 some literature is reviewed and five major ideas
are explored, leading to the derivation of several algorithms for decentralized stabilization. Numerical
results are presented in section 2.5 where we compare the performance of all derived heuristics and
further investigate some questions that arose from the theoretical discussion of the previous sections.
2.2 Centralized Background

Consider a linear, time invariant system described by the following matrix difference equations:

\[
\begin{align*}
    x(k+1) &= Ax(k) + Bu(k) \\
    y(k) &=Cx(k)
\end{align*}
\]  

(2.1)

where \( A \in \mathbb{R}^{n \times n}, \ B \in \mathbb{R}^{n \times m} \) and \( C \in \mathbb{R}^{p \times n} \). This model comes from state space theory where \( x(k) \in \mathbb{R}^n \) represents the state variables at instant \( k \), \( u(k) \in \mathbb{R}^m \) is the \( m \) dimensional control signal and \( y(k) \in \mathbb{R}^p \) denotes the output signal. For a more complete understanding of this formalism consult, for example, chapter 5 of [4]. Given some initial condition \( x(0) \in \mathbb{R}^n \) the set of equations (2.1) describes a unique relation between the input sequence \( \{u(k)\}_{k \geq 0} \) and the output signal \( \{y(k)\}_{k \geq 0} \). This dependency is illustrated in figure 2.1 where the plant is fully characterized by the matrices \( (A,B,C) \) together with an initial condition \( x(0) \). The term plant, comes from basic control theory to designate the physical system to be acted upon, i.e., the underlying process which we want to influence in some manner.

![Figure 2.1: Block diagram of a LTI system.](image)

The control signal \( u(k) \) will be computed by a dynamic linear controller of the following type

\[
\begin{align*}
    x_K(k+1) &= A_Kx_K(k) + B_Ky(k) \\
    u(k) &= C_Kx_K(k) + D_Ky(k) \\
\end{align*}
\]  

(2.2)

The controller is also a LTI system, that receives input measurements \( y(k) \) and computes the control signal \( u(k) \). The order of the controller is given by \( n_k \) which is the dimension of \( x_K(k) \in \mathbb{R}^{n_k} \). Note that if \( (A_K,B_K,C_K) = (0_{n_k \times n_k}, 0_{n_k \times p}, 0_{m \times n_k}) \) then \( u(k) = D_Ky(k) \), i.e., we are considering a static control law that is completely defined by the matrix \( D_K \in \mathbb{R}^{m \times p} \). In the general case the controller is described by the matrices \( (A_K,B_K,C_K,D_K) \) together with an initial condition \( x_K(0) \), as seen in figure 2.2. Combining the controller and the plant in a single system (equations (2.1) and (2.2)) we conclude

![Figure 2.2: Block diagram of a LTI controller.](image)
that:

\[
\begin{bmatrix}
    x(k+1) \\
    x_K(k+1)
\end{bmatrix}
= \begin{bmatrix}
    A + BD_KC & BC_K \\
    BK_C & A_K
\end{bmatrix}
\begin{bmatrix}
    x(k) \\
    x_K(k)
\end{bmatrix},
\]

(2.3)

For the class of controllers considered, the classical stabilization problem consists in designing the matrices \((A_K, B_K, C_K, D_K)\) such that \([x(k), x_K(k)]^T \to 0\) for any initial conditions \(x(0)\) and \(x_K(0)\).

From now on assume that (a) \(y(k) = x(k)\) and (b) we are considering static control laws. This will allow to introduce all major ideas and the general setup will be revisited in appendix A.1. Under these assumptions equation (2.1) reduces to:

\[
x(k+1) = Ax(k) + Bu(k)
\]

(2.4)

and we are considering control laws of the following type \(u(k) = Kx(k)\). The stabilization problem consists in designing \(K \in \mathbb{R}^{n \times m}\) such that \(x(k) \to 0\) for any initial condition \(x(0)\). This is the state feedback setup considering linear static control laws.

The problem may seem abstract, but consider that we desired to drive \(x(k)\) not to zero, but to a fixed reference \(r \in \mathbb{R}^n\). This is the classical tracking problem in which we want to design \(\{u(k)\}_{k \geq 0}\) such that \(x(k) \to r\). This chapter is not concerned with this problem in particular, but we can say that in order to solve the tracking problem one must first tackle the stabilization issue. This happens, for example, when an integral control law is considered to achieve the desired reference \(r\) more robustly [5] (chapter 6.4). Robustness with regard to weak models of the underlying physical process, i.e., when the model \((A, B)\) is very different from the real dynamical system.

One important question is the following: assuming that \(u(k)\) can be a generic vector in \(\mathbb{R}^n\), is it possible to stabilize any discrete system described by (2.4) for any \(x(0)\)? It is clear that the answer will simply depend on the matrix pair \((A, B)\).

A system \((A, B)\) is stabilizable iff for any initial condition \(x(0)\) there exists a control sequence \(\{u(k)\}_{k \geq 0}\) such that \(x(k) \to 0\) where \(x(k+1) = Ax(k) + Bu(k)\). This condition may seem hard to certify, however there are several tests to verify if \((A, B)\) is stabilizable. Perhaps the most sophisticated procedure, is the PHB test which we state next:

**Theorem 2.2.1.** A system \((A, B) \in \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times m}\) is stabilizable iff for any \(\lambda \in \mathbb{C}\) such that \(|\lambda| \geq 1\), the following scalar equality holds

\[
\text{rank}
\left[
\begin{array}{cc}
\lambda I_n - A & B
\end{array}
\right]
= n.
\]

(2.5)

Theorem 2.2.1 can be found, for example, in page 172 of [6] (theorem 6.6) since definition 6.3 (of
page 172) is equivalent to the one given for a stabilizable pair \((A, B)\). Regarding theorem 2.2.1 it is trivial to verify that if \(\lambda\) is not an eigenvalue of \(A\), then (2.5) will be true since \(\operatorname{rank}(\lambda I_n - A) = n\). Hence, in practice, we only need to verify (2.5) for eigenvalues of \(A\) that are not in the interior of the unit disk, i.e., for \(\lambda = \lambda_i(A)\) where \(i = 1, \ldots, n\) and \(|\lambda_i(A)| \geq 1\). In fact, it is possible to substitute the rank condition of (2.2.1) by LMIs [6] (pages 172-174), but the important point is that there exist computational efficient methods to check if a pair \((A, B)\) is stabilizable. During the remaining of this thesis we will restrict ourselves to such pairs since, otherwise there may exist initial conditions \(x(0)\) such that the state \(x(k)\) does not decay to zero, no matter the control signal \(u(k)\).

Assume we have a discrete time system described by a stabilizable pair \((A, B)\). The question now arises: under what conditions do we have \(x(k) \to 0\)?

**Theorem 2.2.2.** Let \(x(k) \in \mathbb{R}^n\) such that \(x(k + 1) = (A + BK)x(k)\) for some stabilizable pair \((A, B)\) \(\in \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times m}\), \(K \in \mathbb{R}^{m \times n}\) and for \(k \geq 1\). Then \(x(k) \to 0\) iff \(\rho(A + BK) < 1\) where \(\rho(\cdot)\) denotes the spectral radius. In fact \(x(k) = (A + BK)^kx(0)\) where \(x(0)\) denotes the initial state condition, so the decay is exponential.

Theorem 2.2.2 follows easily from [7] (Theorem 5.6.12) and has, at least, two valuable consequences: it gives necessary and sufficient conditions for stability in terms of the spectral radius and, allows to quantify the decaying rate of some stabilizable physical system \((A, B)\) when controller \(K\) is employed.

In light of theorem 2.2.2 assume that, for a stabilizable system, we have two controllers \(K_1\) and \(K_2\) and further define \(\Theta_1 \triangleq A + BK_1\) and \(\Theta_2 \triangleq A + BK_2\). If \(\rho(\Theta_1) < \rho(\Theta_2)\) the first controller leads to a “faster” decaying dynamics. As an example imagine that our state variables \(x \in \mathbb{R}^n\) represent the temperature in each division of a house. We know how the temperature evolves through (discrete) time so we have \((A, B)\) and our goal is to design a controller \(K\) such that the temperature is kept at a desired value, say 20°C for every room. If we have two controllers that achieve the desired temperatures, we would prefer the “faster” one, since this implies we have to wait less time for the temperature to be “squeezed” in a \(\epsilon\) margin of 20°C, where \(\epsilon\) denotes any positive constant. This motivates the search for optimal controllers in a spectral radius sense, i.e., controllers with a maximum damping or decay rate.

From the former discussion we would like to find the “fastest” controller \(K^*\), i.e.,

\[
K^* \in \arg \min_{K \in \mathbb{R}^{m \times n}} \rho(A + BK).
\] (2.6)

However, problem (2.6) appears to be challenging since \(\rho(\cdot)\) is a non convex function. This can be easily proved by considering \(\Theta_1 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}\), \(\Theta_2 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}\) and concluding that \(\rho(0.5\Theta_1 + 0.5\Theta_2) = 0.5 > 0 = 0.5\rho(\Theta_1) + 0.5\rho(\Theta_2)\).
If $K$ does not exhibit any constraints, i.e., $K$ can be any matrix in $\mathbb{R}^{m \times n}$, then problem (2.6) can be solved, within any given precision, using some manipulations of the following theorem:

**Theorem 2.2.3.** For any $\Sigma \in \mathbb{R}^{n \times n}$ and $\gamma > 0$ we have that $\rho(\Sigma) < \gamma$ if and only if there exists $P > 0$ such that $\Sigma^T P \Sigma < \gamma^2 P$, where $P > 0$ means that $P$ is a symmetric positive definite matrix.

The proof of theorem 2.2.3 can be consulted in [8] (Theorem 3.5.4). Applying theorem 2.2.3 to $(A+BK)$ we get the following corollary:

**Corollary 2.2.3.1.** For any stabilizable pair $(A,B) \in \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times m}$ and $\gamma > 0$ there exists a controller $K \in \mathbb{R}^{m \times n}$ such that $\rho(A + BK) < \gamma$ if and only if there exists a $Q \succ 0$ and $G \in \mathbb{R}^{m \times n}$ such that

$$
\begin{bmatrix}
\gamma^2 Q & (AQ + BG)^T \\
AQ + BG & Q
\end{bmatrix} > 0,
$$

where $G = KQ$.

**Proof.**

1) By theorem 2.2.3 it is clear that $\rho(A + BK) < \gamma \iff \exists P > 0 : (A + BK)^T P (A + BK) \prec \gamma^2 P$

2) Since $P > 0$ we get that $(A + BK)^T P (A + BK) \prec \gamma^2 P \iff \begin{bmatrix}
\gamma^2 P & (A + BK)^T \\
A + BK & P^{-1}
\end{bmatrix} > 0$ using Schur’s complement [9] (Appendix C.4).

3) $\begin{bmatrix}
\gamma^2 P & (A + BK)^T \\
A + BK & P^{-1}
\end{bmatrix} > 0 \iff \Omega \triangleq \begin{bmatrix}
\gamma^2 P^{-1} & (A + BK) P^{-1} \\
(A + BK) P^{-1} & P^{-1}
\end{bmatrix} > 0$ since

$$
\Omega = \begin{bmatrix}
P^{-1} & 0_{n \times n} \\
0_{n \times n} & I_n
\end{bmatrix} \begin{bmatrix}
\gamma^2 P & (A + BK)^T \\
(A + BK) & P^{-1}
\end{bmatrix} \begin{bmatrix}
P^{-1} & 0_{n \times n} \\
0_{n \times n} & I_n
\end{bmatrix} \text{ and } \begin{bmatrix}
P^{-1} & 0_{n \times n} \\
0_{n \times n} & I_n
\end{bmatrix}
$$

is invertible.

4) Defining $Q \triangleq P^{-1}$ and $G = KQ$ we get the desired LMI, where the change of variables $G = KQ$ is valid since $Q \succ 0$ is invertible.

$\square$

Corollary 2.2.3.1 gives necessary and sufficient conditions for efficiently computing a gain matrix $K$ such that $\rho(A + BK) < \gamma$ for any $\gamma > 0$. Suppose we run the following experiment:

(1) Given some stabilizable pair $(A,B)$ we want to solve problem (2.6), in particular the solution $\rho(A + BK^*)$ should be less than 1 such that the system is stable. With this in mind, we start in the interval $[0, 1]$ and ask whether there exists a controller $K_1$ such that $\rho(A + BK_1) < (1 + 0)/2 = 0.5$. The answer is affirmative and we find that $\rho(A + BK_1) = 0.1989$.

(2) After (1) it is clear that $\rho(A + BK^*)$ must belong to $[0, 0.5]$ so we repeat the procedure and ask whether there exists a controller $K_2$ such that $\rho(A + BK_2) < (0 + 0.5)/2 = 0.25$. By (1) this is true, since $K_2 = K_1$ is a valid solution.
In an analogous way $\rho(A+BK^*)$ must belong to $[0,0.25]$ so we repeat the procedure and ask whether there exists a controller $K_3$ such that $\rho(A+BK_3) < (0+0.25)/2 = 0.125$. Now the answer is negative so we can draw two conclusions: $\rho(A+BK^*)$ must belong to $[0.125,0.25]$ and $\rho(A+BK_1) = 0.1989$ is still our best found solution.

Since $\rho(A+BK^*)$ must belong to $[0.125,0.25]$ we ask whether there exists a controller $K_4$ such that $\rho(A+BK_4) < (0.125 + 0.25)/2 = 0.1875$. We find an $K_4$ such that $\rho(A+BK_4) = 0.1314$ so it is true that $\rho(A+BK^*)$ must belong to $[0.125,0.1875]$. Conclusion: given any tolerance $\delta > 0$, if we keep repeating this procedure we can bound $\rho(A+BK^*)$ in an interval of Lebesgue measure (length) not greater than $\delta$. Figure 2.3 summarizes this construction for $\delta = 0.1$ where blue dots represent the middle points of the intervals, green dots indicate that the underlying question had an affirmative answer and red dots indicate the contrary, using the previous best found solution. The experiment stopped at the fifth iteration since $0.1875 - 0.125 = 0.0625 < 0.1 = \delta$ so if we use the controller $K_4$ it is clear that $|\rho(A+BK_4) - \rho(A+BK^*)| \leq 0.1 \neq \delta$.

![Figure 2.3: Bisection algorithm for $\delta = 0.1$.](image)

The previous experiment actually replicates the idea of a bisection algorithm where given an initial interval $[a,b]$ we bound $\rho(A+BK^*)$ in intervals $[a_n,b_n]$ such that $b_n - a_n = (b-a)/(2^n-1)$ for $n \geq 1$ with $[a_1,b_1] = [a,b]$. As already concluded, given a stabilizable pair $(A,B)$ and a finite precision $\delta > 0$, we can find a stabilizing controller $K$ such that $|\rho(A+BK) - \rho(A+BK^*)| \leq \delta$ by setting $[a,b] = [0,1]$, i.e., we can produce an optimal stabilizing controller for (2.6) within any $\delta > 0$. Since $b_n - a_n = (b-a)/(2^n-1)$ we know that the desired tolerance will be achieved after $\lceil \log_2(\frac{b-a}{\delta}) \rceil + 1$ iterations, where $\lceil \cdot \rceil$ denotes the ceiling operator. This checks for the example of figure 2.3 since after $\lceil \log_2(\frac{1}{0.1}) \rceil + 1 = 5$ iterations the procedure stopped. Algorithm 1 describes the precise routine.

The last conditional statement of algorithm 1 only makes sure that a non-zero controller is produced, if the above while loop achieves the desired tolerance $\delta$ before being able to generate any feasible controller. For example assuming $\rho(A) > 1$, $u_{init} = 1$ and $\delta = 1$ the condition of line 5 will automatically be false and if $(A,B)$ is stabilizable then a feasible controller will be produced by the optimization problem of line 18.
Algorithm 1 Centralized bisection algorithm

1: procedure CENTRALIZED BISECTION(A, B, δ, u_{init})
2: \( l = 0 \)
3: \( u = u_{init} \)
4: \( K_{best} = 0 \in \mathbb{R}^{m \times n} \)
5: while \( |u - l| > δ \) do
6: \( m = (u + l)/2 \)
7: if \( ρ(A + BK_{best}) < m \) then
8: \( u \leftarrow m \)
9: continue
10: Find \( G, Q \) such that
11: \[
\begin{bmatrix}
    m^2Q \\
    AQ + BG
\end{bmatrix}
\begin{bmatrix}
    (AQ + BG)^T \\
    Q
\end{bmatrix} > 0
\]
12: if (10 is feasible) then
13: \( K = QQ^{-1} \)
14: \( K_{best} \in \arg \min \{ρ(A + BK_{best}), ρ(A + BK)\} \)
15: else
16: \( u \leftarrow m \)
17: if \( ρ(A + BK_{best}) \geq u_{init} \) then
18: Find \( G, Q \) such that
19: \[
\begin{bmatrix}
    u_{init}^2Q \\
    AQ + BG
\end{bmatrix}
\begin{bmatrix}
    (AQ + BG)^T \\
    Q
\end{bmatrix} > 0
\]
20: if (18 is feasible) then
21: \( K_{best} = QQ^{-1} \)
22: return \( K_{best} \)

Note that, in line 7 before we face the feasibility problem for the current \( u \), we investigate the best found controller \( K_{best} \) and verify if \( ρ(A + BK_{best}) < m \). If this is the case a feasible controller already exists and the LMI can be skipped. This is merely a speed-up technique, that can improve the time performance of the classical bisection algorithm since, generally, fewer feasibility problems will be solved.

To conclude this section, when \( K \) can be any matrix in \( \mathbb{R}^{m \times n} \) algorithm 1 gives the optimal solution of problem (2.6), within any finite precision \( δ \).

2.3 Decentralized Formulation

From the previous section, the centralized stabilization problem is to design a controller \( K \in \mathbb{R}^{m \times n} \) such that the state variables tend asymptotically to zero. The control sequence \( \{u(k)\}_k \) will be given by \( u(k) = Kx(k) \) so considering the \( i \)-th component of \( u \) it is clear that:

\[
u_i(k) = \sum_{j=1}^{n} x_j(k)K_{i,j}.
\] (2.7)

Algorithm 1 will produce the optimal controller \( K^* \in \mathbb{R}^{m \times n} \) and since \( (A, B) \) forms a stabilizable pair, corollary 2.2.3.1 insures that if update (2.7) is executed with \( K = K^* \), then the closed loop system will be stable. But what happens if \( u_i(k) \) cannot access all state variables \( x_i(k) \)? In this scenario, the control signal has to be computed using only some state variables.
To formalize this idea of partial information lets define the indicator function $I(i,j)$ as:

$$I(i,j): \{1, \ldots, n\} \times \{1, \ldots, m\} \rightarrow \{0, 1\}$$

(2.8)

$$I(i,j) \mapsto \begin{cases} 1 & \text{if } i \sim j \\ 0 & \text{otherwise} \end{cases}$$

where $i \sim j$ means that the control component $u_j(k) \in \mathbb{R}$ can be computed using $x_i(k) \in \mathbb{R}$ for any time instant $k \geq 1$. The set of distributed controllers $\mathcal{K}_I$ is given by:

$$\mathcal{K}_I = \{ K \in \mathbb{R}^{m \times n} : K_{i,j} = 0 \text{ if } I(i,j) = 0 \},$$

(2.9)

since if $K \in \mathcal{K}_I$ then the computation of $u_j(k)$ involves only the $j$-th row of controller $K$ and state variables $x_i(k)$ for $I(i,j) = 1$. The set $\mathcal{K}_I$ is convex since it is the intersection of a finite number of linear constraints. Note that the centralized case can be recovered by defining $I(i,j) = 1$ for any valid pair $(i,j)$, i.e., the function $I$ dictates how sparse the controller must be. Using the same motivation as before, the optimal decentralized stabilization problem is the following:

$$\min_{K \in \mathcal{K}_I} \rho(\mathbf{A} + BK).$$

(2.10)

Note that the LMI given in corollary 2.2.3.1 is no longer valid for the decentralized problem (2.10) since the controller produced by algorithm 1 will be an arbitrary matrix in $\mathbb{R}^{m \times n}$, i.e., there are no sparsity guarantees. The naive solution of using algorithm 1 and then imposing the desired sparsity does not generally work, since a single zero entry, in $K$, can make the entire system unstable. Section 2.4 reviews some literature on this problem and introduces several computationally efficient heuristics. The next section motivates the decentralized formalism by means of two examples.

### 2.3.1 First Motivation

As motivation, consider the house temperature example where each variable $x_i$ represents the temperature of a division. We can model this system as a graph $G = (V, E)$ where each node represents a state variable $x_i$. Assume that the divisions of the house are numbered 1 through $n$. The edges of $G$ reflect the house architecture since if division $i_1$ is adjacent to division $i_2$ then node $x_1$ is a neighbour of node $x_2$. The shorthand notation $x_1 \sim x_2$ means that $x_1$ and $x_2$ are neighbouring nodes. We want to design a controller $K$ such that the temperature is kept at a desired reference. The actuators can be modelled as windows, heaters or air conditioners and the sensors will be thermometers. In the centralized case, we assume the existence of a central entity (say a remote computer) that computes $u(k)$ using the information of the entire state vector $x(k)$, i.e., update (2.7) can be executed. Physically, each sensor measures the corresponding temperature and then transmits it to the remote computer. After $u(k)$ is computed, the signal is routed to the actuators and this process is repeated iteratively. Communications are made through wireless channels. Figure 2.4 displays a possible graph for this example, where we
have actuators in the living room (node 1), the bedroom (node 3) and in the porch (node 9). Nodes with actuators are coloured cyan and a actuator is associated with a single control component (for simplicity). Nodes 2, 5, 4, 6 represent the kitchen, the hall and both bathrooms, respectively. Nodes 7 and 8 represent bedrooms like node 3. In figure 2.4 all nodes will send their state variable to the remote computer, i.e., we did not draw all dashed lines simply for visualization purposes. The edge between node 8 and the remote computer has a single arrowhead since this node has no actuator, hence it will receive no component of the control signal.

\[ u(k) = Kx(k) \]

Figure 2.4: Home temperature example: centralized case with set \( \mathcal{K}_T = \mathbb{R}^{m \times n} \).

Assume now that no central entity exists and instead we have \( N \) computational entities (say microprocessors) that receive some state variables (not all) and compute \( u_j(k) \) such that \( j = 1, \ldots, N \) and \( m_1 + \ldots + m_N = m \). The micro-controllers are placed in some division of the house. If micro controller \( j \) is placed in node \( i \) then, let we assume \( u_j(k) \) can only be computed using information from neighbouring nodes. This is reasonable if we think that the communication quality degrades with the distance between two nodes, hence it is assumed that neighbouring nodes in \( G \) are “near” each other and that the only reliable information is the one given by the neighbourhood of order 1, i.e., by direct neighbours. The main motivation is that we would like to keep the number of communication channels to a minimum hence only direct neighbours can communicate with each other, by some underlying wireless network. For simplicity consider \( N = m \) and \( m_j = 1 \) for any feasible \( j \), i.e., each microprocessors computes a single control component.

Considering equation (2.7), controller \( K \) will exhibit some structural constraints since if each microprocessor \( j \) is placed at node \( i \) then \( u_j(k) \) will be computed using only local information, i.e., \( x_j(k) \) for \( x_l \sim x_i \) (neighbouring nodes). Figure 2.5 displays the same example but now the controller has some
structural constraints. One might wonder why there are no dashed lines in figure 2.5, like in figure 2.4. We did not draw these lines because they would coincide with the links of the house, by the direct neighbours assumption. However it is important to understand their purpose: dashed lines would represent directed or undirected communications channels that allow to exchange information, in this case state variables $x_i$, between source and sink nodes of graph $G$. Consider for example, the microprocessor at node 1. This entity will compute $u_1(k) \in \mathbb{R}$ using only $x_1(k), x_5(k)$ which justifies the first row of matrix $K$. The zeros indicate that all other state variables are unreachable, i.e., $u_1(k) = x_1(k)K_{1,1} + x_5(k)K_{1,5}$. For figure 2.5 it is evident that $I(1,1) = I(5,1) = I(2,2) = I(3,2) = I(7,3) = I(8,3) = I(9,3) = 1$ and $I(i,j) = 0$ for any other possible combination of $i = 1, \ldots, 9$ and $j = 1, \ldots, 3$.

For figure 2.5 it is evident that $I(1,1) = I(5,1) = I(2,2) = I(3,2) = I(7,3) = I(8,3) = I(9,3) = 1$ and $I(i,j) = 0$ for any other possible combination of $i = 1, \ldots, 9$ and $j = 1, \ldots, 3$.

Just a side note, we are not considering that the dynamics of a node is only influenced by direct neighbours, i.e., the temperature at node 1 can potentially influence the temperature at node 9 since no sparsity is considered in the state matrices $(A, B)$. The sparsity in $K$ is modelling a different phenomena: it assumes nothing on how the system behaves only on what information is accessible to compute the components of the control signal.

This example gives a tangible idea on a concrete case where one needs decentralized controllers. If instead of a house with 9 divisions we consider a hotel with 500 rooms or a city with 5000 houses one can easily conclude that the centralized solution becomes unpractical due to computational resources, bottlenecks, fault tolerance, security, etc.

### 2.3.2 Second Motivation

The previous examples motivates decentralized control by assigning physical locations to state variables and control signals. However there is an even simpler example that motivates the decentralized approach: consider that we are in an industrial setup such that the pair $(A, B)$ describes the dynamics
of a complex machine. As engineers, we want to produce a control sequence \( \{u(k)\}_k \) such that the machine achieves a desired operating point, i.e., a predefined configuration of state variable specified by a reference signal \( r \). Our goal is to produce a controller \( K \) that makes sure that the state variables converge to the desired operating point, as fast as possible. Assume that most state variables have a physical existence. Hence there exist sensors to measure the value of \( x_i \). Sensors are physical devices that have associated costs: monetary, instalment, repairs, calibration, etc. In the centralized case, all sensors have to be operational since \( u(k) \) is computed using all state variables \( x(k) \). In the decentralized setup we can try to eliminate overpriced sensors by zeroing some entries of \( K \) thus saving resources.

For example for \( n = 20 \) and \( m = 5 \), imagine that state variable \( x_5 \) has a large sensing cost. The decentralized formulation allows to search for a controller \( K \in \mathcal{K}_I \) such that his fifth column is equal to the zero vector, i.e., \( I(5,j) = 0 \) for any \( j = 1,\ldots,5 \) and \( I(i,j) = 1 \) for any other feasible pair \( (i,j) \). Conclusion: we would save resources by choosing a decentralized approach. The exact same argument can be made if we consider defective sensors that can fail to measure some state component \( x_i(k) \). In this setup, the decentralized approach allows us to have a more robust control law detached of faulty sensors by properly choosing the indicator function \( I \) hence defining the set \( \mathcal{K}_I \).

### 2.4 Decentralized Heuristics

Before introducing the heuristics, it is important to revisit problem (2.10) and make some remarks. By theorem 2.2.3 it is trivial that problem (2.10) is equivalent to:

\[
\min_{P,K,\phi} \sqrt{\phi} \quad \phi P > (A + BK)^T P (A + BK) \\
P > 0, \quad \phi > 0, \quad K \in \mathcal{K}_I
\]  

(2.11)

where only a change of variables \( \phi = \gamma^2 \) was considered. Note that this manipulation is valid since \( \gamma \), by definition, is a positive scalar. Analysing the feasible set of (2.11), one easily concludes that the matrix inequality involving \( (P, K, \phi) \) is, in fact, a Bilinear Matrix Inequality (BMI): a matrix inequality which is linear with respect to each individual optimization variable. Checking the solvability of a general Bilinear Matrix Inequality is an \( \mathbf{NP} \) hard problem [2] (Theorem 1), so it is still not known how to solve problem (2.11) efficiently (polynomial time).

After exploring some state of the art approaches (section 2.4.1), we design algorithms that try to find a local solution of (2.11), namely we want to find a feasible controller, \( K \in \mathcal{K}_I \), such that \( \rho(A + BK) \) is, at least, lower than one. Clearly this can be impossible. The trivial example is to impose that \( \mathcal{K}_I = \{0\} \), which physically means that we have zero actuators at our disposal, so \( K = 0 \) and \( \rho(A + BK) = \rho(A) \) which, in general, is bigger than unit since we are trying to stabilize unstable systems. If our heuristic finds a controller such that \( \rho(A + BK) \) is, at least, lower than one we can guarantee the decentralized
stability of the system. Otherwise we cannot conclude that the problem of finding a $K \in K_I$ such that
\[ \rho(A + BK) < 1 \] is infeasible: our heuristic failed but it is possible for others to succeed. This is one of
the consequences of dealing with non-convex, $\mathcal{NP}$ hard problems: the following methods present only
sufficient conditions for the decentralized stabilization problem.

### 2.4.1 State of the Art

This section will review some of the most significant advances in the decentralized stability of LTI sys-
tems described by equation (2.1). In a recent paper [10], Alavian and Rotkowitz present a randomized
constructive algorithm to deal with the stabilization problem, under information constraints assuming a
dynamic controller type. The concept of fixed modes was extended to a decentralized setup: in the
centralized case the fixed modes of a plant, with a dynamic controller, correspond to the eigenval-
ues of the closed loop matrix, equation (2.3), that remain unchanged no matter the controller matrices
$(A_K, B_K, C_K, D_K)$. In the decentralized case, the exact same definition applies but the controller has to
respect, some pre-defined information constraints about which measurements are available to compute
each control component (Definition 1) [10]. Given some constraints, if the fixed modes of the plant are
all stable (inside the unit circle in $\mathbb{C}$) then algorithm 1 of [10] is guaranteed to find a stabilizing dynamic
controller and, in fact, the non fixed modes can be moved to any chosen location with arbitrary precision.

The major insight of [10] consists in constructing a dynamic controller that can reduce the number of
unfixed unstable modes, while leaving all fixed modes stable. This idea is then applied recursively, until
all unfixed unstable modes become stable. This result is strong since it is based on necessary and suf-
cient conditions for decentralized stability and it holds both for continuous and discrete time. However
some remarks have to be made when comparing the results of [10] with our framework:

1) The results of paper [10] are only valid for dynamic controllers, i.e., they do not produce a decentral-
ized stabilizing controller in the static case. Further more, the order of controller $n_K$ is not specified
by the user: it is generated by the underlying algorithm. This can be problematic if only static control
laws can be considered and the dynamic option is too computationally expensive. The heuristics of
this chapter, although only based on sufficient conditions, are able to produce either static or dynamic
controllers and for the dynamic case, the user can specify the order of the controller allowing a low
order search procedure, i.e., trying different $n_K$ values and choosing the lowest one that guarantees
decentralized stability.

2) The most relevant remark is that the authors assume that the dynamics of the controller is centralized,
i.e., the controller is a dynamical system which receives some available measurements and computes
the control signal in a centralized manner: all state variables of the controller, are potentially needed
for the computation of the control signal. Imagine that we have a full order controller, i.e., $n_K = n$.
The dynamic controller of [10] will potentially use all $n_K$ state variables to compute each control
component $\{u_i\}^m_i$. One can easily understand why this becomes unpracticable for massive scale
systems: no central entity can physically process all the information. The heuristics of this chapter,
allow to search for dynamic controllers that only use available measurements, but have decentralized
dynamics i.e., the matrices $(A_K, B_K, C_K, D_K)$ can exhibit some desired sparsity pattern allowing the
controller to be employed in a distributed fashion (section A.1).

In [11], Sadabadi and Peaucelle present a complete review on the most successful heuristics for the
problem of structured stabilization considering static output feedback controllers, while highlighting the
special cases where the problem becomes convex and hence computationally tractable. In this setup,
the controller receives output measurements to compute the control signal, hence
\[ u(k) = K_y(k) = K C x(k), \]
where the last equality comes from equation (2.1). Decentralized stability is a structural con-
straint where one imposes some sparsity pattern in the controller, i.e., $K \in \mathcal{K}_I$. Other examples include
transmission delays (composition constraints) and symmetric controllers (scaling constraints) [12],[13]
and [14]. However the survey of [11] assumes a continuous time state space formulation: the literature
is vast for continuous time but poor for discrete time models. This serves as a motivation for assuming
a discrete time formalism. In both cases synthesizing a decentralized controller boils down to solving a
Bilinear Matrix Inequality. However the problems are not equivalent: in discrete time we want to confine
the closed loop eigenvalues to the interior of the unit circle while, in continuous time, a system is stable
iff all closed loop eigenvalues have a negative real part. The continuous time BMI, for output feedback,
is given by:

\[
(A + BKC)^T P + P (A + BKC) \prec 2 \beta P
\]
\[
P > 0,
\]
(2.12)

where the optimization variables are $(K, P)$ and feasibility of (2.12) implies that the spectral abscissa
of $A + BKC$ is lower then $\beta$. The spectral abscissa, $\alpha(A + BKC)$, denotes the maximum real part
of the eigenvalues of $A + BKC$ and, in particular, the underlying continuous time system is stable iff
$\alpha(A + BK) < 0$. The main approaches of [11] that allow to impose that $K \in \mathcal{K}_I$ can be summarized as
follows:

1) Iterative LMI heuristics: this type of approaches deals directly with problem (2.12) trying to find vari-
ables $P > 0$ and $K \in \mathcal{K}_I$ such that the corresponding BMI holds. Most of them use one fairly simple
idea: transform the underlying BMI into an LMI, by either discarding non-convex terms or bounding
them by convex alternatives. In [15] Hassibi, How and Boyd use a path following method where given
some initial matrices $(\hat{P}, \hat{K}, \hat{\beta})$ such that (2.12) is feasible, a iterative procedure is employed in order
to improve $\hat{\beta}$, at least by $\delta \beta$, by doing incremental updates in the Lyapunov certificate $P = \hat{P} + \Delta P$
and the controller $K = \hat{K} + \Delta K$ and discarding second order information, i.e., the non convex cross
terms which involve the incremental variables. Some extra constraints guarantee that the discarded
terms have “little” influence and hence this first order approximation is not too crude. The basis for
this idea is further explored in section 2.4.5 where we tackle the discrete case and highlight the dif-
fferences with respect to [15].

Instead of discarding non convex terms another option is to use Majorization Minimization algorithms
namely convex concave procedures [16] where the BMI in (2.12) is decomposed as the difference of two matrix convex functions:

\[ G(P, K, \beta) - F(P, K, \beta) \]

and the concave term, \(-F(P, K, \beta)\), is linearized around some point \((\hat{P}, \hat{K}, \hat{\beta})\). After this linearization, the BMI is transformed into a LMI which then can be optimized with respect to the variables \((P, K, \beta)\) and, in particular, one can impose that \(K \in \mathcal{K}_I\) and minimize \(\beta\). This procedure was successfully applied to a wide range of continuous time problems [17] and the idea dates back to [18], where Oliveira, Camino and Skelton define the concept of convexifying potentials: an appropriate first order function that can be used to transform non-convex matrix inequalities into LMIs (Definitions 1 and 2) [18]. This idea will be further explored in section 2.4.4, where we introduce the concepts of matrix convex functions, linearizations, convex concave procedures and highlight our own contribution to this technique, in the context of discrete time decentralized stabilization.

2) Decoupled Lyapunov matrix methods: from formulation (2.12) one can easily conclude that the matrices \(A + BK C\) and \(P\) are coupled, in the sense that they exhibit a non-linear product interaction. However there exist equivalent formulations that, although not convex in all variables, allow to decouple the Lyapunov matrix \(P\) and closed loop system \(A + BK C\) [19],[20],[21]. One might wonder: why is that useful? Why trade one non-convex formulation by another? Consider the state feedback case, \(C = I_n\), such that we match the framework of [19], where Oliveira, Bernussoub and Geromel introduced this idea of decoupling, for the problem of testing stability over a uncertainty domain in discrete time. Controller synthesis follows from an extension of the developed stability condition. In [19] the authors assume that the state space matrices \((A, B)\) are not deterministic; instead they belong to a convex polytopic set \(A \times B\) of the following type:

\[
A = \{A \in \mathbb{R}^{n \times n} : A = \sum_{n=1}^{N} \alpha_i A_i, \sum_{n=1}^{N} \alpha_i = 1, \ \alpha_i \geq 0\} \\
B = \{B \in \mathbb{R}^{n \times m} : B = \sum_{n=1}^{N} \beta_i B_i, \sum_{n=1}^{N} \beta_i = 1, \ \beta_i \geq 0\} \quad (2.13)
\]

where the vertices \(\{(A_i, B_i)\}_{i}^{N}\) are known beforehand. One can easily conclude that a sufficient condition for stability over (2.13) is to use the LMI of corollary 2.2.3.1 for every pair \(\{(A_i, B_i)\}_{i}^{N}\), i.e., searching for a Lyapunov matrix \(P\) such that \(N\) corresponding LMIs hold. In practice this approach is too conservative since a single Lyapunov matrix will generally make the underlying problem unfeasible. The major contribution of [19] is to introduce a decoupled LMI which is equivalent to the one of theorem 2.2.3 and allows to compute a Lyapunov certificate for each vertex of the polytope (Theorem 2) by introducing extra \(n^2\) optimization variables.

Stability under uncertainty was the main motivation for this class of decoupled formulations. However one can apply these ideas for decentralized control design where we, again, face a bilinear matrix inequality, now with \(n^2\) extra optimization variables. More variables tend to be beneficial, from
an optimization point of view, since they enlarge the search space and hence, in general, are less conservative. More variables also implies an extra computational demand however the variables introduced in [19] do not increase the size or number of linear matrix inequalities, so there is a minor computational difference. In section 2.4.5 we introduce an idea for decentralized stability and develop two algorithms using the two types of formulations: the standard Lyapunov conditions of theorem 2.2.3 and the decoupled condition of [19].

3) Non-Lyapunov-based approaches: this type of algorithms tend to “avoid” the BMI (2.12), in the sense that the stabilization problem is seen from a pure optimization viewpoint:

\[
\text{minimize } K \in K \quad \alpha(A + BK).
\] (2.14)

This is the continuous time version of problem (2.10) and the spectral abscissa is a non convex, non smooth function which makes the optimization procedure challenging. The survey [11] highlights HIFOO (H\text{\$\$}_{\infty} - H\text{\$\$}_{2} Fixed Order Optimization) as one of the best non-Lyapunov based approaches: a free MATLAB package [22], [23] for a variety of continuous time control problems, including finding a decentralized stabilizing controller for formulation (2.14). This package is mainly based on quasi-Newton updating and gradient sampling algorithms [24],[25]. There is an extension of HIFOO to the discrete time case [26], where it is shown how one can compute the gradient of the spectral radius function and use the main ideas of the original HIFOO package. However the available software cannot be used to solve problem (2.10), i.e., the developed techniques are only used to search for stabilizing controllers and then use this solution to further improve some underlying control objective function. As an user, one cannot use HIFOO software to minimize the spectral radius under control sparsity constraints.

If one admits specific sparsity patterns, then there exist classical sufficient conditions for decentralized stability. In [27] Geromel, Bernussou and Peres studied the state feedback case assuming a static controller with the following block diagonal structure:

\[
K_D = \begin{bmatrix} K_1 & \cdots & K_M \end{bmatrix} \in \mathbb{R}^{m \times n} \quad (2.15)
\]

such that the matrices \( \{K_i\}_{i=1}^M \) can have arbitrary dimensions \( K_i \in \mathbb{R}^{m_i \times n} \) given that \( \sum_{i=1}^{M} n_i = n \) and \( \sum_{i=1}^{M} m_i = m \). The subscript D indicates the block diagonal structure of the controller, using the notation of [27]. One can easily show that the matrix pair \((A, B)\) is stabilizable iff the following augmented matrices
$\hat{A}, \hat{B}$ are also:

\[
\hat{A} = \begin{bmatrix} A & B \\ 0_{m \times n} & 0_{m \times m} \end{bmatrix} \in \mathbb{R}^{(m+n) \times (m+n)} \tag{2.16}
\]

\[
\hat{B} = \begin{bmatrix} 0_{(n+m) \times m} \\ I_m \end{bmatrix} \in \mathbb{R}^{(m+n) \times m}.
\]

The major insight of [27] consists in guaranteeing the block diagonal structure (2.15), by adding extra constraints on a Lyapunov certificate $W$. Extra constraints immediately imply loss of necessity, as mentioned in the fourth page of [27]. More specifically, the block diagonal controller $K_D$ can be computed from the following problem:

\[
\min_W 0 \quad 0
\]

\[
\begin{bmatrix} I_n & 0_{n \times m} \\ 0_{n \times n} & 0_{m \times m} \end{bmatrix} \{\hat{A}W\hat{A}^T - W\} \begin{bmatrix} I_n & 0_{n \times m} \\ 0_{m \times n} & 0_{m \times m} \end{bmatrix}^T < 0 \tag{2.17}
\]

\[
W = \begin{bmatrix} W_1 & W_2 \\ W_2^T & W_3 \end{bmatrix} \succeq 0, \quad W_1 > 0
\]

\[
W_1 = W_{1,D}, \quad W_2 = W_{2,D}
\]

by noticing that the feasibility of (2.17) implies that $K_D = W_{2,D}^T W_{1,D}^{-1}$ is a block diagonal stabilizing controller for the original problem (Theorem 5) [27]. Again the notation $W_{i,D}$ denotes block diagonal matrices. By using an augmented representation, the authors were able to understand that block diagonal constraints are preserved through a suitable change of variables, namely if $W_1 = W_{1,D}$ and $W_2 = W_{2,D}$ then $K_D = W_{2,D}^T W_{1,D}^{-1}$ will also exhibit this structure. This idea is then applied to a wide range of control problems, both in continuous and discrete time.

In [28] Crusius and Trofino consider static control laws and present sufficient LMI conditions for output stability. The decentralized aspect of the paper comes from an application of such conditions to a specific type of decentralized structure: lower triangular controllers defined by the following constraints

\[
F_o = \begin{bmatrix} 0 \\ \text{no restrictions} \end{bmatrix} \in \mathbb{R}^{m \times p} \tag{2.18}
\]

where ■ means no restrictions on the particular matrix entry, using the notation of [28]. Assuming the usual state space model, equation (2.1), the authors provide two convex feasibility problems that, if feasible, provide a decentralized controller $F_o$ through a suitable manipulation of the feasible variables. The two formulations are similar but have underlying rank assumptions regarding the state space matrices $C$ or $B$. We will present only the so called Discrete $W$ problem, Definition 4 of [28], to illustrate our point.
If the state space matrix $C \in \mathbb{R}^{p \times n}$ is full row rank then one encounters the following convex problem:

$$
\begin{equation}
\min_{W,M,N} \begin{bmatrix}
-W & WA^T - C^T N^T B^T \\
AW - BNC & -W
\end{bmatrix} \prec 0
\end{equation}
$$

where the lower triangular controller is given by $F_o = NM^{-1}$ if (2.19) is feasible. One can easily understand how this approach relates to that of [27]: the specific structure of the controller (2.18) is guaranteed, indirectly, by adding extra constraints on the optimization variables that allow to reconstruct $F_o$ if problem (2.19) is feasible. Note that if $(N,M)$ are both required to be block diagonal one can also impose block diagonal structures for the controller $F_o$. The ideas presented in [27] and [28] work only for a restricted set of decentralized structures: block diagonal and/or lower triangular matrices. However this concept can be generalized for arbitrary sparsity patterns: consider corollary 2.2.3.1 and suppose we want to make sure that $GQ^{-1} \in K_I$ by imposing some structure on variables $Q$ and $G$. An observation explored in [29] (but in continuous time) consists in imposing a diagonal structure on the positive definite matrix $Q$ such that:

$$
GQ^{-1} = \begin{bmatrix}
Q_{1,1} & \cdots & Q_{1,n} \\
\vdots & \ddots & \vdots \\
Q_{n,1} & \cdots & Q_{n,n}
\end{bmatrix} = \begin{bmatrix}
Q_{1,1}g_1 & \cdots & Q_{n,n}g_n
\end{bmatrix}
$$

where $g_i \in \mathbb{R}^m$ and $Q_{i,j} \in \mathbb{R}_{++}$ since $Q$ is a positive definite matrix. Since $Q_{i,j} \in \mathbb{R}_{++}$ we conclude that:

$$
GQ^{-1} \in K_I \iff G \in K_I;
$$

hence, by imposing the convex constraint $G \in K_I$ one has sufficient conditions for decentralized stability under arbitrary sparsity patterns. In the context of decentralized stability, this results can be seen as a generalization of [27] or [28] since it can be applied for any set $K_I$, while the former papers focused only on specific patterns. The issue is that these conditions can be too conservative since they tend to be too greedy, trying to solve the problem in one shot. For these cases, there can be a considerable distance between sufficiency and necessity.

Actually, if we impose extra conditions of the closed loop matrix, the diagonal sufficient condition can also be necessary. From [30] (Proposition 2) we find that for any $\Sigma \in \mathbb{R}^{n \times n}_+$, i.e., a $n$ by $n$ matrix with
non-negative entries, the following equivalence holds:

\[ \rho(\Sigma) < \gamma \iff \exists \ P = \begin{bmatrix} P_{1,1} & \cdots & \cdots & P_{n,n} \end{bmatrix} > 0 : \Sigma^T P \Sigma < \gamma^2 P \]  \hspace{1cm} (2.21)

where \( \gamma \) is some positive scalar. Note that result (2.21) is valid for \( \Sigma \in \mathbb{R}^{n \times n}_+ \), however in output feedback controller synthesis \( \Sigma = A + BC \) and \( K \) is an optimization variable so the sufficient and necessary conditions are only valid if one restricts the closed loop matrix to be non-negative. For an arbitrary closed loop matrix, the option of selecting a diagonal Lyapunov certificate only ensures sufficiency. We highlight this result to remind the reader that we make no assumption on the pair \((A, B)\). Only that it is stabilizable, otherwise the problem is impossible to solve. Our algorithms are developed for general systems and arbitrary sparsity patterns. If one imposes extra restrictions, like non-negativeness of the closed loop matrix, the \( \mathcal{NP} \) hard problem can become convex and hence computationally solvable by standard methods. Section 4.3 will make great use of this fact.

### 2.4.2 Big Picture

Before explaining our heuristics for decentralized stability, we present figure 2.6 that presents an high level notion on the main insights of this chapter:

1. The reduction strategy is a simple idea that consists in starting with a “big” bound on the spectral radius of \( A + BK \) and, incrementally, decreasing this value by a “small” amount. This idea is properly introduced in section 2.4.4, and its value consists in transforming sufficient conditions for decentralized stability, into a suitable algorithmic procedure. Consult sections 2.4.4 and 2.4.5.

2. The Bisection method was already explained in section 2.2, where it was used for computing the optimal centralized controller, assuming a state feedback control law. In the decentralized case, the same idea can be used to improve some heuristic that is employed for a fixed bound \( \gamma \). By running the heuristic, for several \( \gamma \) values generated by a bisection search, the performance of the method can only improve.

3. The rank constrained formulation, basically allows to convert the involved non-convex feasibility problem to that off minimizing a non-convex functional over a convex set. This is the basis of section 2.4.3.

4. The linearization technique consists in doing a first order approximation of a convex function and utilizing some useful global bounds, that follow trivially from convexity.

5. Finally, the incremental definition corresponds to path following methods (homotopy) where a given variable receives incremental updates around some nominal value. When applying this technique to the underlying BMI, the non-convex incremental terms can be discarded or replaced by proper
convex bounds. The main insight is that this approximations are less crude since we are only considering incremental terms, i.e., we already have some information about the nominal value.

### 2.4.3 Incremental Inverse Algorithm

The incremental inverse algorithm, is based on theorem 2.4.1 which gives an equivalent condition for the generalized inequality of theorem 2.2.3 in terms of $P$ and a new variable $Q$:

**Theorem 2.4.1.** For any $\Sigma \in \mathbb{R}^{n \times n}$ and $\gamma > 0$ we have that $\rho(\Sigma) < \gamma$ iff there exists a $Q, P > 0$ such that

\[
\begin{bmatrix}
\gamma^2 P & \Sigma^T \\
\Sigma & Q
\end{bmatrix} > 0, \quad \begin{bmatrix}
P & I_n \\
I_n & Q
\end{bmatrix} \succeq 0 \text{ and } \text{trace}(PQ) = n.
\]

**Proof.**

1) By theorem 2.2.3 it is clear that $\rho(\Sigma) < \gamma \iff \exists \ P > 0 : \Sigma^T \Sigma < \gamma^2 P$

2) Since $P > 0$ we get that $\Sigma^T \Sigma = \gamma^2 P \iff \begin{bmatrix}
\gamma^2 P & \Sigma^T \\
\Sigma & P^{-1}
\end{bmatrix} > 0$ using Schur's complement.

3) Defining $Q \triangleq P^{-1}$ our objective is to show that $Q = P^{-1} \iff \begin{bmatrix}
P & I_n \\
I_n & Q
\end{bmatrix} \succeq 0$ and $\text{trace}(PQ) = n$. It is clear that $Q > 0$, so $Q$ is a *symmetric* positive definite matrix.

4) Assume that $Q = P^{-1}$ then

\[
\begin{bmatrix}
P & I_n \\
I_n & Q
\end{bmatrix} \succeq 0 \iff \begin{bmatrix}
P & I_n \\
I_n & P^{-1}
\end{bmatrix} \succeq 0 \iff P - P \succeq 0 \text{ which is true since } 0 \in \mathbb{R}^{n \times n} \text{ has all eigenvalues equal to 0. The last equivalence uses Schur's complement and the fact that } Q = P^{-1} \succeq 0. \text{ The second condition follows easily since } PQ = I_n \text{ so } \text{trace}(PQ) = n.
\]

5) Assume that

\[
\begin{bmatrix}
P & I_n \\
I_n & Q
\end{bmatrix} \succeq 0 \text{ and } \text{trace}(PQ) = n \text{ so } \begin{bmatrix}
P & I_n \\
I_n & Q
\end{bmatrix} \succeq 0 \iff \Gamma \triangleq P - Q^{-1} \succeq 0. \text{ We will now show that if } \Gamma \succeq 0 \text{ then } \text{trace}(PQ) = n \text{ implies that } \Gamma = 0 \iff P = Q^{-1}. \text{ We do this by reduction}
to absurdity. Assume all assumptions hold but $\Gamma \neq 0$.

$$\text{trace}(PQ) = \text{trace}(I_n) + \text{trace}(\Gamma Q) = n + \text{trace}(\Gamma Q)$$

Now $\text{trace}(\Gamma Q) = \text{trace}(\Gamma^{1/2}\Gamma^{1/2}Q^{1/2}Q^{1/2}) = \text{trace}(\{\Gamma^{1/2}Q^{1/2}\}r_{\Phi}^{T}\{\Gamma^{1/2}Q^{1/2}\}) = \sum_{i,j=1}^{n} \Phi_{i,j}^2 \geq 0$. Since we initially assumed that $\text{trace}(PQ) = n$ this implies that $\text{trace}(\Phi^{T}) = 0$ but if $\text{trace}(\Phi^{T}) = 0$ then $\Phi = 0$ by the last observation. $\Phi = \Gamma^{1/2}Q^{1/2} = 0 \Leftrightarrow \Gamma^{1/2} = 0 \Rightarrow \Gamma = 0$, where the last equivalence is a consequence of $Q^{1/2}$ being invertible. $\Gamma = 0$ is a contradiction hence the proof is done.

$\square$

Applying theorem 2.4.1 to $A + BK$ we find that both LMIs depend linearly on $K$, which implies that we can impose any convex constraints on $K$, namely $K \in \mathcal{K}_I$. The issue is that $\text{trace}(PQ) = n$ is a non convex constraint which makes the entire set of feasible points non convex. To see this consider $n = 1$ and the following set:

$$\Delta = \{(p,q,\sigma) \in \mathbb{R}^3 : p q = 1, \ p > 0, \ q > 0, \ p \geq \frac{1}{q}, \ p > \sigma^2 \}$$

which represents the set of feasible scalars such that the new conditions of theorem 2.4.1 hold with $\gamma = 1$. Considering the points $\gamma_1 = (1,1,0.5)$ and $\gamma_2 = (2,0.5,0.5)$ it is clear that both points belong to $\Delta$. However $\gamma_3 = 0.5\gamma_1 + 0.5\gamma_2 = (1.5,0.75,0.5)$ does not belong to $\Delta$ since $1.5 \times 0.75 = 1.125$, i.e., the condition $pq = 1$ fails. One can test whether $(P,Q,\Sigma) \in \Delta$ by evaluating the solution of an optimization problem:

**Theorem 2.4.2.** For any $\Sigma \in \mathbb{R}^{n \times n}$ and $\gamma > 0$ we have that there exists a $Q,P > 0$ such that

$$\begin{bmatrix} \gamma^2 P & \Sigma^T \\ \Sigma & Q \end{bmatrix} \succeq 0, \quad \begin{bmatrix} P & I_n \\ I_n & Q \end{bmatrix} \succeq 0 \quad \text{and trace}(PQ) = n$$

iff $f(\Sigma,\gamma) = n$ where the function $f$ is defined as:

$$f(\Sigma,\gamma) = \min_{P,Q} \text{trace}(PQ) \begin{bmatrix} \gamma^2 P & \Sigma^T \\ \Sigma & Q \end{bmatrix} \succeq 0, \quad \begin{bmatrix} P & I_n \\ I_n & Q \end{bmatrix} \succeq 0, \ P,Q > 0.$$

The proof of theorem 2.4.2 is trivial by noticing that $\begin{bmatrix} P & I_n \\ I_n & Q \end{bmatrix} \succeq 0 \Rightarrow \text{trace}(PQ) \geq n$. All previous arguments gave necessary and sufficient conditions to ensure that $\rho(A + BK) < \gamma$, but now we lose necessity and introduce the incremental inverse heuristic which does incremental updates in the variables $P$ and $Q$. Consider $P = \dot{P} + \Delta P > 0$ and $Q = \dot{Q} + \Delta Q > 0$ and note that, by theorem 2.4.2, the problem of finding a controller $K \in \mathcal{K}_I$ such that $\rho(A + BK) < \gamma$ can be transformed into optimizing the non convex function $\text{trace}(PQ) = \text{trace}(\dot{P}\dot{Q} + P\Delta Q + \Delta P\dot{Q} + \Delta P\Delta Q)$ in a convex set. Since the
increments will correspond to the optimization variables, the objective function is non convex due to the cross terms $\Delta P \Delta Q$ but, intuitively, since $\Delta P$ and $\Delta Q$ represent incremental variables their product should have a “low” trace hence, by assuming $\text{trace}(\Delta P \Delta Q) \approx 0$ we drop this term and lose necessity. Given this construction, the basic idea is to start with a pair $(P_1, Q_1) = (\hat{P}, \hat{Q})$ and iterate the mentioned problem, until some termination criterion is met.

Note that this idea of testing whether $(P, Q, \Sigma) \in \Delta$ by evaluating the optimal value of a non-convex problem and then approximating the non-convex functional by a linear objective is well known in the literature. In [31] Ghaoui, Oustry and AitRami use this type of arguments for continuous type systems and denote the original non-convex problem as the cone complementarity problem. The setup of [31] is slightly different since, instead of assuming an incremental definition of the variables $(P, Q)$ and “discarding” the non-convex terms the authors use a linearization method where at a given $(P_0, Q_0)$ the following approximation is employed:

$$\text{trace}(PQ) \approx \text{trace}(PQ_0 + P_0Q) + \text{constant} \quad (2.23)$$

given that the constraint

$$\begin{bmatrix} P & I_n \\ I_n & Q \end{bmatrix} \succeq 0$$

holds. The algorithmic procedure of [31] (algorithm 1) can be summarized as follows:

$$(P_{k+1}, Q_{k+1}) \in \arg \min_{P, Q} \text{trace}(P_k Q + Q_k P)$$

subject to $$(P, Q) \in \mathcal{X}$$

$$\begin{bmatrix} P & I_n \\ I_n & Q \end{bmatrix} \succeq 0 \quad (2.24)$$

where $\mathcal{X}$ is an appropriate convex set. Note that this iterative process is very similar to the one described above, so one might wonder: why the need for this section?

The ideas of [31] and, in fact, all rank constrained approaches in survey [11] compute a stabilizing controller $K$, indirectly, by means of an alternative formulation that uses the well known elimination lemma [6], [32]. This is a lemma that allows to eliminate variables in an LMI with a certain structure, by making use of orthogonal complements. The set $\mathcal{X}$ in (2.24) is made of LMI conditions derived from this lemma such that the eliminated variable is the controller $K$. This technique can be useful, in the context of centralized stabilization, since the underlying BMI can be transformed into an equivalent LMI with an additional rank constraint such that the controller $K$ is eliminated from the problem, hence reducing the number of optimization variables. In [11] there exists a complete review of approaches that deal with the underlying rank constraint, one being the approach employed in [31] and present in theorem 2.4.2, i.e., this result is no novelty at all.

If the iterative procedure of (2.24) is successful the centralized controller can be reconstructed from some complex analytical formulas of [33] (Theorem 1). These results are not suitable for decentralized
control laws since it is difficult (if not impossible) to impose general sparsity patterns by relying on the highly non-linear analysis of [33]. This implies that the elimination of the controller, by use of the elimination/projection lemma like in [11], is not very suitable for general structured controller synthesis, in particular decentralized structures.

It is noticed in [11] that an alternative way to reconstruct the controller $K$ is to use the information from the reformulated problem, the variable $P$, and insert it in the original standard BMI, (2.12), such that the only optimization variable is $K \in K_I$. This is a one-shot procedure that tends to be greedy since there is no iterative computation of the controller $K$ and hence it will most likely fail. Another one-shot approach to reconstruct the centralized controller $K$ is proposed in [31] (Theorem 2.3). By doing some boundedness assumptions, the authors can prove that a centralized stabilizing controller exists if some more complex conditions are met. The same previous comment applies here: it is difficult to use these results for decentralized (non-greedy) controller synthesis. Having mentioned some work in the field, we denote our iterative approach as the incremental inverse heuristic:

**Algorithm 2** Incremental inverse heuristic with $\gamma > 0$

(i) Given some $\gamma > 0$ initialize $P_0, Q_0 > 0$ and further define $N_{\text{max}}$ as the maximum number of iterations and $\epsilon$ as a stopping tolerance.

(ii) $(K_{k+1}, P_{k+1}, Q_{k+1}) \in \arg \min_{P,Q,\Delta P, \Delta Q,K} \text{trace}(P_k \Delta Q + Q_k \Delta P + P_k Q_k)$ subject to

$$
\begin{bmatrix}
\gamma^2 P & (A + BK)^T \\
A + BK & Q
\end{bmatrix} > 0
$$

$$
\begin{bmatrix}
P & I_n \\
I_n & Q
\end{bmatrix} \succeq 0, \quad P,Q \succ 0, \quad K \in K_I
$$

$$
P = P_k + \Delta P, \quad Q = Q_k + \Delta Q
$$

(iii) Repeat (ii) until $\{\rho(A + BK_k) < \gamma\} \bigcup \{k = N_{\text{max}}\} \bigcup \{|\rho(A + BK_{k-1}) - \rho(A + BK_k)| < \epsilon\}$

Several remarks have to be made:

(1) Perhaps one immediate question is the following: why do we assume that $\text{trace}(\Delta P \Delta Q) \approx 0$ but have no constraint in (2.25) to actually enforce this feature?

The intuition to assume that $\text{trace}(\Delta P \Delta Q) \approx 0$ is clear from the previous discussion. However we do not materialize it since adding any extra constraint in (2.25) can decrease the set of feasible points hence discarding possible valid decentralized controllers $K \in K_I$. There is an obvious compromise: if we impose that $\text{trace}(\Delta P \Delta Q)$ is “small” then our objective function will become “close” to $\text{trace}(PQ)$ which is good, however the set of feasible points will decrease. We choose to have a “worst” approximation in the objective function, hence not discarding possible feasible decentralized controllers. In section 2.5 we will see that this works well in practice.

(2) The previous heuristic tries to find a controller $K \in K_I$ such that $\rho(A + BK) < \gamma$ for a fixed $\gamma$. 


However we can also employ a **bisection** framework, like in the centralized case. This implies that the **incremental inverse** heuristic can be executed in two different modes: for a fixed upper bound $\gamma$ or in a bisection framework using algorithm 1 where the feasibility problem of lines 10/18 is substituted by algorithm 2 with a suitable initialization. It is trivial that, by construction, the bisection mode gives a better controller, in the spectral radius sense. If one wants to at least stabilize the closed loop system $A + BK$, the bisection procedure should be started with $\gamma = 1$. In reality there exists a third possible mode to run this algorithm, however it will be highlight in the next section for consistency (it uses an idea explained there).

(3) Any initialization $P_0, Q_0$ can be employed. However different initializations will generally, lead to different solutions since the heuristic does incremental updates in the original pair $(P_0, Q_0)$. To prove that any pair $(P_0, Q_0)$ implies feasibility of problem (2.25) we assume, without loss of generality, that $A \neq 0 \in \mathbb{R}^{n \times n}$. If this was the case the stabilization problem would be trivial: choose $K = 0$ and achieve the optimal spectral radius of 0. Consider the following assignment of variables:

\[
\begin{align*}
P &= I_n \beta \\
Q &= I_n \alpha \\
\Delta P &= P - P_0 \\
\Delta Q &= Q - Q_0 \\
K &= 0_{m \times n}
\end{align*}
\]

where $\alpha$ and $\beta$ are positive scalars. Setting $K = 0$ is a feasible assignment regardless of the sparsity $K_I$ and, trivially, $P$ and $Q$ are positive definite matrices. To show that the two remaining matrix inequalities hold we have to find positive scalars $(\alpha, \beta)$ such that:

\[
\begin{bmatrix} P & I_n \\ I_n & Q \end{bmatrix} \succeq 0 \iff P - Q^{-1} \succeq 0 \iff \beta \geq \frac{1}{\alpha}
\]

\[
\begin{bmatrix} \gamma^2 P & (A + BK)^T \\ A + BK & Q \end{bmatrix} \succ 0 \iff \gamma^2 P - (A + BK)^T Q^{-1} (A + BK) \succ 0 \iff \gamma^2 I_n \beta - \frac{A^T A}{\alpha} \succ 0 \iff \beta \geq \frac{\lambda_{\text{max}}(A^T A)}{\gamma^2 \alpha}.
\]

For any $\alpha > 0$ consider this specific assignment of $\beta$:

\[
\begin{align*}
\beta &= \begin{cases} \\
\frac{1}{\alpha} + 0.1 & \text{if } \lambda_{\text{max}}(A^T A) \leq \gamma^2 \\
\frac{\lambda_{\text{max}}(A^T A)}{\gamma^2 \alpha} + 0.1 & \text{if } \lambda_{\text{max}}(A^T A) > \gamma^2
\end{cases}
\end{align*}
\]

One can easily conclude that (2.29) is a valid assignment that respects both (2.27) and (2.28) so any initialization $(P_0, Q_0)$ can be employed for algorithm 2.
An initialization that seems to work well in practice, is the following: for a target $\gamma$ use corollary 2.2.3.1 to find $Q \succ 0$ associated with the centralized controller and define $Q_0 = Q$ and $P_0 = Q^{-1}$. Intuitively, this makes sense since we first solve the centralized stabilization problem for the upper bound $\gamma$, and then "expect" that $(P, Q)$ are "similar" in the distributed case, for the same bound $\gamma$.

(4) The objective function $\text{trace}(P_k \Delta Q + Q_k \Delta P + P_k Q_k)$ can only improve from iteration $k$ to iteration $k + 1$. To prove this consider that, from iteration $k - 1$, resulted a feasible triplet $(P_k, Q_k, K_k)$ such that $\text{trace}(P_{k-1} \Delta Q + Q_{k-1} \Delta P + P_{k-1} Q_{k-1}) = \alpha_1$ where $P_k = P_{k-1} + \Delta P$ and $Q_k = Q_{k-1} + \Delta Q$. Note that the controller $K_k$ is not incrementally updated (there exists no $\Delta K$) since problem (2.25) only needs $P_k$ and $Q_k$ to compute $(P_{k+1}, Q_{k+1}, K_{k+1})$.

To compare the objective function between iterations, consider that in iteration $k$ we have $(P_k, Q_k, K_k)$ and problem (2.25) must be solved. If $\Delta P = \Delta Q = 0$ the objective function equals $\alpha_1$ since $P_{k+1} = P_k$, $Q_{k+1} = Q_k$ and $K_{k+1} = K_k$ and this set of variables is feasible, since it resulted from iteration $k - 1$. Conclusion: we found a set of variables $(\Delta P, \Delta Q, P, Q, K) = (0, 0, P_k, Q_k, K_k)$ such that all constraints are met and the objective function equals $\alpha_1$ hence the optimal value of $\text{trace}(P_k \Delta Q + Q_k \Delta P + P_k Q_k)$ can never be greater then $\alpha_1$. This justifies the need to impose a finite number of iterations $N_{\text{max}}$ or a stopping tolerance $\epsilon$ as a termination criteria.

(5) The constraint \[ \begin{bmatrix} P_{k+1} & I_n \\ I_n & Q_{k+1} \end{bmatrix} \geq 0 \] implies that $\text{trace}(P_{k+1} Q_{k+1}) \geq n$ so $\text{trace}(P_k \Delta Q + Q_k \Delta P + P_k Q_k) \geq n - \text{trace}(\Delta Q \Delta P)$ where equality means that the corresponding controller $K_k \in \mathcal{K}_G$ guarantees $\rho(A + B K_k) < \gamma$ by theorem 2.4.2. By (4) it is clear that $\text{trace}(P_k \Delta Q + Q_k \Delta P + P_k Q_k)$ describes a monotone tendency in the $(\Delta P, \Delta Q)$ plane, across iterations, trying to reach the lower bound $n - \text{trace}(\Delta Q \Delta P)$. If the lower bound is achieved then $\rho(A + B K_k) < \gamma$. In practice, we allow an early termination i.e., it can happen that we are in a $\epsilon$ margin of $n - \text{trace}(\Delta Q \Delta P)$ but we already have a feasible controller $K_k \in \mathcal{K}_I$ such that $\rho(A + B K_k) < \gamma$. In this scenario the heuristic finishes hence the early termination nomenclature. Note condition (iii) in algorithm 2.

2.4.4 Decentralized Convex-Concave Approach

This section introduces the convex-concave heuristic applied to problem (2.10): the decentralized convex-concave heuristic. The standard convex concave procedure (CCP) is a well known heuristic that converges to the set of stationary points of difference of convex (DC) optimization problems [16]. A DC optimization problem is defined as follows [16]:

\[
\min_{x \in \mathbb{R}^n} \ \tilde{f}_0(x) - \tilde{g}_0(x) \\
\tilde{f}_i(x) - \tilde{g}_i(x) \preceq_K 0, \ i = 1, \ldots, m
\] (2.30)
where \( \hat{f}_0, \hat{g}_0 : \mathbb{R}^n \rightarrow \mathbb{R} \) are convex functions, \( K \subseteq \mathbb{R}^p \) is a proper cone and \( \hat{f}_i, \hat{g}_i : \mathbb{R}^n \rightarrow \mathbb{R}^p \), for \( i = 1, \ldots, m \), are differentiable functions and convex w.r.t the generalized inequality in \( K \), i.e.,

\[
\forall \ \theta \in [0, 1] : \hat{f}_i(\theta x + (1 - \theta)y) \preceq_K \theta \hat{f}_i(x) + (1 - \theta)\hat{f}_i(y)
\] (2.31)

for any \( x, y \in \mathbb{R}^n \) [9], [34]. An equivalent definition of \( K \) convex functions [9] (section 3.6.2) is the following:

\[
\forall \ x, y \in \mathbb{R}^n : \hat{f}_i(y) \succeq_K \hat{f}_i(x) + D\hat{f}_i(x)(y - x)
\] (2.32)

where \( D\hat{f}_i(x) \) denotes the Jacobian of \( \hat{f}_i \) evaluated at point \( x \). The basic CCP is implemented as follows (algorithm 1.1)[16]:

(i) Given problem (2.30), linearize all functions \( \hat{g}_i \) around some point \( x_k \) for \( i = 1, \ldots, m \). The linearization corresponds to the right hand side of (2.32) with \( x = x_k \). In problem (2.30) substitute each function \( \hat{g}_i \) by the corresponding linearized version and note that this is now a convex problem: minimizing of a convex function under a convex set.

(ii) Solve the convex problem from (i) and use the found solution \( x^* \) to update the linearization point, i.e., \( x_{k+1} = x^* \).

(iii) Iterate the previous procedure.

In [16] Boyd and Lipp further explain the previous procedure, prove convergence, add some generalizations and provide some numerical examples that show that this idea generally works well, even on hard problem like Boolean satisfiability. Suppose we want to apply the previous results to \( S^n_+ \), i.e., the set of positive semidefinite matrices which is a proper cone in \( S^n \) [9] (page 43). The immediate issue is that (2.30), (2.31) and (2.32) are defined for proper cones in \( \mathbb{R}^p \). However \( S^n_+ \) is defined in \( S^n \), so there is a dimensionality problem. In [9] (page 43, example 2.15) the authors explicitly say that inequality \( X - Y \succeq_K 0 \) should be interpreted as the matrix \( X - Y \) being positive semidefinite if \( K = S^n_+ \). So it is clear how to generalize (2.30) if \( K = S^n_+ \). In page 110 of [9] (example 3.48) the authors also defined how (2.31) should be interpreted for \( K = S^n_+ \) by extending the concept of convexity to matrix convexity. A function \( f_i : \mathbb{R}^n \rightarrow S^m \) is matrix convex iff:

\[
\forall \ \theta \in [0, 1] : f_i(\theta x + (1 - \theta)y) \preceq_K \theta f_i(x) + (1 - \theta)f_i(y)
\] (2.33)

for any \( (x, y) \in \mathbb{R}^n \times \mathbb{R}^n \). To generalize property (2.32) with \( K = S^n_+ \) we make use of [17], where the derivative of a matrix value mapping \( G : \mathbb{R}^n \rightarrow S^p \) at a point \( x \in \mathbb{R}^n \) is defined as:

\[
DG(x)h = \sum_{i=1}^{n} h_i \frac{\partial G}{\partial x_i}(x), \ \forall h \in \mathbb{R}^n.
\] (2.34)

Note that \( DG(x)h : \mathbb{R}^n \rightarrow \mathbb{R}^p \) is a linear function of \( h \). The function \( G \) is said to be differentiable, on a domain \( \mathcal{G} \) if and only if \( DG(x)h \) is well defined for any \( x \in \mathcal{G} \). The matrix version of (2.32) is given by
part b) of lemma 2.1[17], which we rewrite as:

**Theorem 2.4.3.** Let $G : G \mapsto S^p$ denote a differentiable matrix function in $G$. Then $G$ is matrix convex iff for any $x, x + \Delta \in G$ the following matrix inequality holds:

$$G(x + \Delta) \succeq G(x) + DG(x)\Delta.$$  

Note that the matrix inequality in theorem 2.4.3 is well defined since all underlying matrices are symmetric in $R^{p\times p}$. As the notation suggests, theorem 2.4.3 has a simple interpretation: any differentiable matrix convex function can be, globally lower bounded by a linear functional of an increment $\Delta$.

The function $DG(x)h$ is sometimes called the differential of function $G$ at point $x$ with increment $h$ [35]. A simple geometric interpretation can be borrowed from [35]: consider a one dimensional scalar function $\phi : R \mapsto R$. The differential of $\phi$ at point $c$ with increment $u$, $d\phi_c(u)$ is defined as $d\phi_c(u) = u\phi'(c)$ where $\phi'(c)$ is the usual derivative of $\phi$ at point $c$. Using Taylor’s series it is easy to conclude that $\phi(c + u) \approx \phi(c) + u\phi'(x) = \phi(c) + d\phi_c(u)$ so $d\phi_c(u)$ represents the linear part of $\phi(c + u) - \phi(c)$. Figure 2.7 illustrates the differential concept, for a concave function $\phi$. It follows trivially that the convex function $-\phi$ is, globally lower bounded by the linear approximation represented by the straight line. Theorem 2.4.3 simply generalizes this geometric result for matrix-valued convex differentiable functions.

![Figure 2.7: Geometrical interpretation of the differential, page 93 of [35.](image)

Although problem (2.10) is not trivially formulated as a DC problem, it is possible to construct a convex concave algorithm in a decentralized stabilization framework. The following theorem is the main reason why the decentralized control design task can be interpreted as a matrix value DC optimization problem in $S^+_n$:

**Theorem 2.4.4.** The function $f(\Sigma, P) = \Sigma^TP^{-1}\Sigma$ is matrix convex where $(\Sigma, P)$ is defined in $R^{n\times n} \times S^+_n$.

**Proof.**
1) First note that matrix convexity is equivalent to show that for any \( x \in \mathbb{R}^n \) we have that \( g_x(\Sigma, P) \overset{\Delta}{=} x^T f(\Sigma, P) x \) is convex for any \( (\Sigma, P) \) defined in \( \mathbb{R}^{n \times n} \times S_{++}^n \). Now \( x^T f(\Sigma, P) x = y^T P^{-1} y \) where \( y \in \mathbb{R}^n \) is given by \( y = \Sigma x \) (a linear function of \( \Sigma \)).

2) From [9] (example 3.4, page 76) it is easy to show that \( h(y, P) = y^T P^{-1} y \) defined on \( \mathbb{R}^{n \times n} \times S_{++}^n \) is convex.

3) \( g_x(\Sigma, P) = h(\Sigma x, P) \) is the composition of a linear function with a convex one, hence \( g_x \) is convex for any \( x \) which implies that \( f(\Sigma, P) \) is matrix convex.

As a corollary of theorem 2.4.4 it is trivial to show that \( P^{-1} \) is matrix convex in \( S_{++}^n \) simply by taking \( \Sigma = I_n \). By theorem 2.2.3 there exists a \( K \in K_I \) such that \( \rho(A + BK) < \gamma \) if and only if

\[
\exists \ P^{-1} > 0 : (A + BK)^T P^{-1} (A + BK) \prec \gamma^2 P^{-1}
\] (2.35)

since the change of variables \( P \) to \( P^{-1} \) is always valid, for invertible matrices. Conclude that \( (A + BK)^T P^{-1} (A + BK) \) is matrix convex since it is the composition of an affine map \( A + BK \) with a convex function \( \Sigma^T P^{-1} \Sigma \) where \( P^{-1} > 0 \). Hence for a fixed \( \gamma > 0 \) the controller design problem is the following

\[
\min_{K, P} 0 \\
(A + BK)^T P^{-1} (A + BK) - \gamma^2 P^{-1} \prec 0 \quad P > 0, \ K \in K_I.
\] (2.36)

For a fixed bound \( \gamma \), problem (2.36) is a matrix DC problem, in particular a DC feasibility problem where the objective function can only take two values: 0 if all constraints are met and \( +\infty \) otherwise, which is a standard way of defining feasibility problems [9]. In continuous time one can also decompose (2.12) as the difference of two convex matrix functions [17]. The realization that conventional control theory BMIs can be interpreted using the results of DC problems has already occurred in [17] where several continuous time problems are considered using static output feedback controllers. Once again, this paper is only concerned with continuous time problems which motivates our discrete time setup. After explaining our algorithmic approach, we will highlight one extra insight that is not present in [17] but tends to give better overall numerical results.

From theorem 3 [35] (page 171) we get that the differential of \( G(P) = P^{-1} \) at point \( \hat{P} \) with increment \( \Delta \) is given by:

\[
DG(\hat{P}) \Delta = -\hat{P}^{-1} \Delta \hat{P}^{-1}.
\] (2.37)

This generalizes the scalar differential of \( g(p) = 1/p \) at \( \hat{p} \) with increment \( \delta \), which equals

\[
df(\hat{p})\delta = \delta g'(\hat{p}) = -\delta/\hat{p}^2.
\] (2.38)
Corollary 2.4.4.1 results from the application of theorem 2.4.3 to the convex differentiable function $G(P) = P^{-1}$:

**Corollary 2.4.4.1.** Let $F(P) = P^{-1}$ be a matrix function defined in $S_{++}^n$. If $\hat{P} \in S_{++}^n$ then

$$P^{-1} \succeq \hat{P}^{-1} - \hat{P}^{-1}(P - \hat{P})\hat{P}^{-1}$$

so $\hat{P}^{-1} - \hat{P}^{-1}(P - \hat{P})\hat{P}^{-1}$ denotes the first order approximation of function $F$ at point $\hat{P}$.

Note that the function $F(P) = P^{-1}$ can also be interpreted as a function from $\mathbb{R}^{n^2}$ to $S_{++}^n$, by considering the vec operator that stacks the columns of a matrix. This is just to say that the application of theorem 2.4.3 in corollary 2.4.4.1 is indeed valid. Using corollary 2.4.4.1 and formulation (2.35) we present the decentralized convex concave heuristic, which builds upon the matrix version of the classical CCP algorithm:

**Algorithm 3** Decentralized convex-concave heuristic

(i) Find some $P_0 \succ 0$ and $K_0 \in K_I$ such that $(A + BK_0)^T P_0^{-1} (A + BK_0) \prec r_0^2 P_0^{-1}$ for some $r_0 > 0$. Further define $N_{\text{max}}$ as the maximum number of iterations, $\epsilon$ as a stopping tolerance and $\beta > 0$ as the shrinking factor, i.e., $\beta < 1$. Set $r_0 = \beta r_0$.

(ii) Find $P_{k+1} = P > 0$ and $K_{k+1} = K \in K_I$ such that

$$(A + BK)^T P^{-1} (A + BK) \prec r_k^2 \{P_k^{-1} - P_{k-1}^{-1}(P - P_k)P_k^{-1}\}$$

(2.39)

where $P_k^{-1} - P_{k-1}^{-1}(P - P_{k-1})P_{k-1}^{-1}$ denotes the first order approximation of $P^{-1}$ (2.4.4.1). Note that (2.39) is an LMI in the optimization variables $P$ and $K$.

(iii) If $\rho(A + BK_k) < r_k$ then $r_{k+1} = \beta r_k$; otherwise go to (v).

(iv) Repeat until \( \left\{ k = N_{\text{max}} \right\} \cup \left\{ \left| \rho(A + BK_{k-1}) - \rho(A + BK_k) \right| < \epsilon \right\} \)

(v) Return $K^* \in \arg \min_{K \in K_I} \{\rho(A + BK_0), \ldots, \rho(A + BK_{\text{end}})\}$, where $K_{\text{end}}$ denotes the controller associated with the last iteration of the algorithm. At most, the heuristic runs for $N_{\text{max}}$ iterations.

Several remarks have to be made:

1. The previous algorithm needs a feasible triplet $(P_0, K_0, r_0)$ to start the iterative procedure. Any initialization can be employed such that $P_0 > 0$, $K_0 \in K_I$ and $(A + BK_0)^T P_0^{-1} (A + BK_0) \prec r_0^2 P_0^{-1}$. A initialization that seems to work well in practice, is the following: given some set $K_I$ first find a controller $K_0 = K \in K_I$ that minimizes the spectral norm of $A + BK$, i.e.,

$$K_0 \in \arg \min_{K \in K_I} ||A + BK||$$

(2.40)
where \( ||.|| \) denotes the spectral norm. Since any norm is a convex function and \( \mathcal{K}_I \) is a convex set, problem (2.40) is convex. Then set \( r_0 = \rho(A + B K_0) + \delta \) with \( \delta > 0 \) and find \( P_0 = P \) such that
\[
P > 0, \hspace{1em} (A + B K_0)^T P (A + B K_0) < r_0^2 P,
\]
which is always feasible, by theorem 2.2.3, since \( r_0 = \rho(A + B K_0) + \delta \) and \( \delta > 0 \). One can easily conclude that for any matrix \( \Phi \in \mathbb{R}^{n \times n} \) the spectral norm is an upper bound for the spectral radius, i.e., \( \rho(\Phi) \leq ||\Phi|| \), so this initialization finds a feasible set of variables such that a convex upper bound of \( \rho \) is minimized. In particular, if \( ||A + B K_0|| < 1 \) then \( \rho(A + B K_0) < 1 \) so the naive approach of solving problem (2.40) can lead to a stable closed loop system \( A + B K \).

(2) The idea of doing a first order approximation of the function \( P^{-1} \) such that problem (2.39) is convex and iteratively updating the linearization point is no novelty since this is the core of the generic convex concave procedure [16]. The contribution comes from incrementally decreasing \( r_k \) by a factor \( \beta \) if \( \rho(A + B K_k) < r_k \). This strategy will be denoted as a reduction strategy, so in the fact algorithm 3 should be named decentralized convex concave heuristic in a reduction mode. This strategy could be thought as a path following method (homotopy) where the design objective is iteratively improved, by at least \( \beta \), leading to a “path” of LMIs that try to find a local solution to problem (2.10) [15]. Algorithm 3 combines ideas from DC programming with path following methods, in the context of discrete time decentralized stability.

Intuitively, \( \beta \) should be a scalar close to 1 since, if in iteration \( k \), problem (2.39) is solved for \( r_k \), then it should not be “to hard” to find a feasible set of variables such that the same problem is solved for \( r_{k+1} \) that is “slightly” smaller than \( r_k \). Define \( S_k \) as a convex set in \( \mathbb{R}^{m \times n} \times S_+^n \times \mathbb{R}^+ \) such that problem (2.39) is feasible for that value of \((K, P, r_k)\). In broad terms this heuristic assumes that if \( r_k \approx r_{k+1} \) then the distance between the two sets \( S_k \) and \( S_{k+1} \) is “small”. Formalizing this idea is a bit tricky, so we give only a intuitive argument.

One immediate question is the following: why is our decay factor constant through iterations? Why not set, for example, \( \beta_k = \rho(A + B K_{k-1}) \) such that, at iteration \( k \), we have only to improve the solution found at the previous iteration? This makes sense since if we find a feasible \( K \in \mathcal{K}_I \) such that \( \rho(A + B_k) = 0.8 \), our interest is to improve this value in consecutive iterations. Our numerical experiments indicate that a constant decay factor leads to better results since, for \( \beta_k = \rho(A + B K_{k-1}) \), the heuristic does not has enough time to evolve. This approach can be too greedy since, at iteration \( k \), we demand a strict improvement regarding the previous found controller.

The heuristic nature of the reduction strategy, will be concentrated in the empirical choice of \( \beta < 1 \). A value of \( \beta = 0.99 \) seems to work well in practice, hence all reduction mode algorithms will use this value.

(3) If \( \beta \) is a value “close” to 1, then \( r_k = (\beta)^k r_0 \) will have a slow decay, hence a “large” number of
iterations will be needed to, at least, get a stabilizing controller such that \( \rho(A + BK) < 1 \). So there is an obvious inverse proportional relation between the value of \( \beta \) and the maximum number of iterations \( N_{\text{max}} \) for reduction mode algorithms. Given \( \beta \) it is trivial to come up with an upper bound for \( N_{\text{max}} \) by considering the following argument:

(i) Suppose we want to solve problem (2.10) for some stabilizable pair \((A, B)\) and set \( K_I \). The solution of (2.10) is greater than or equal to \( \rho(A + BF) \) where \( F \) denotes the optimal centralized controller given by algorithm (1). This is obvious since problem (2.10) is more constrained than the central version (2.6).

(ii) Suppose we start algorithm 3 with an upper bound \( r_0 \). Using (i) it is clear that:

\[
N_{\text{max}} \leq \left\lceil \log_\beta \left( \frac{\rho(A + BF)}{r_0} \right) \right\rceil \quad (2.42)
\]

where \( \lceil \cdot \rceil \) denotes the ceil operator. This is simply finding the integer \( k \), such that \( r_k = (\beta)^k r_0 = \rho(A + BF) \).

Bound (2.42) has some interest considering low-medium size problems: for large scale systems no single entity can computed the full centralized controller, so our decentralized approach simply cannot depend on central results. For these cases a sufficient large \( N_{\text{max}} \) should be employed, unfortunately in an ad-hoc fashion.

(4) After analysing algorithm 3 one might ask: since \( r_k \) is being successively decreased through iterations, why do we need to return \( K^* \in \arg \min_{K \in K_j} \{ \rho(A + BK_0), \ldots, \rho(A + BK_{\text{end}}) \} \)? Isn’t \( K^* \) always the controller associated with the last feasible iteration?

Denote \( m \) as the last iteration such that \( \rho(A + BK_m) < r_m \), i.e., \( K_m = K_{\text{end} - 1} \). It is true that, since \( m \) is the last feasible iteration, then \( r_m \leq r_n \) for any \( n \in \{1, \ldots, m\} \). However remember that \( r_m \) is an upper bound, i.e., it only says that \( \rho(A + BK_m) < r_m \). It says nothing about the actual value \( \rho(A + BK_m) \), so it can happen that \( \rho(A + BK_m) > \rho(A + BK_n) \) for some \( n \neq m \). This justifies step five of algorithm 3.

(5) Note that this reduction strategy could also be applied to the incremental inverse algorithm of section 2.4.3. Using the initialization proposed in (1) we can find \( P_0, r_0 \) and then define \( Q_0 \triangleq P_0^{-1} \). The incremental inverse heuristic in a reduction mode will use the core structure of algorithm 3 where problem (2.39) is substituted by formulation (2.25). The incremental inverse hence can be employed in three modes: for a fixed upper bound \( \gamma \), in a bisection framework, or in a reduction strategy. For the decentralized convex concave procedure the reduction strategy has a valid justification. It is not really obvious why one would want to mix the reduction strategy together with the incremental inverse formulation, since the former already exhibits a functional to optimize hence guiding the iterative nature of the algorithm. Experimentally we verify that the reduction strategy is considerably faster (section 2.5), so this may serve as a motivation for this option.
(6) There is no need to provide an initial \( \gamma > 0 \), since the heuristic will continue running even if \( \rho(A + BK_k) < \gamma \). The reduction strategy can be thought of as a “global” method in the sense that it tries to find the optimal controller \( K^* \) and not just a controller \( K_k \) that guarantees that \( \rho(A + BK_k) < \gamma \). Observe that there is no stopping criterion that checks if \( \rho(A + BK_k) < \gamma \).

(7) After observation (2) one could ask: what happens if we run algorithm 3 without the reduction strategy and simply use a fixed \( \gamma > 0 \)?

From formulation (2.36) one can easily conclude that this heuristic is based on feasibility problems, i.e., there is no explicit objective function to optimize. The reduction strategy is really guiding the searching procedure by promoting the improvement of \( \rho(A + BK_k) \) through consecutive iterations, using the previous feasible variable \( P_{k-1} \). If we employ the heuristic for a fixed \( \gamma > 0 \), we are discarding the iterative nature of the algorithm hence the approach will generally be too greedy in the sense that it will try to find a feasible controller in one shot, i.e., without iterating.

(8) Consider (again) formulation (2.35). Our main motivation for introducing the reduction mode strategy, was based on the observation that there is no explicit way to minimize \( \alpha \). The incremental inverse heuristics exhibits it own functional to optimize hence guiding the searching procedure, but for algorithm 3 we had to promote improvement, indirectly, through the shrinking factor \( \beta \). Actually this is not entirely true since, by theorem 2.4.4, the term \( \gamma^2P^{-1} \) is actually matrix convex, i.e., we can consider \( \gamma \) as an optimization variable and actually arrive at the following problem:

\[
\begin{align*}
\min_{K, P, \gamma} & \quad \gamma \\
\text{s.t.} & \quad (A + BK)^TP^{-1}(A + BK) - \gamma^2P^{-1} \prec 0 \\
& \quad P > 0, \quad K \in K_I, \quad \gamma > 0.
\end{align*}
\]

This is a completely valid matrix CCP problem since the linear inequality, involving the controller \( K \), is a difference of two matrix convex functions. This raises the question: if this is the case, why use the reduction strategy? Why not simply linearize the term \( \gamma^2P^{-1} \) around some point \((\gamma_k, P_k)\) and iterate, like in the classical CCP algorithm? This would be easy to implement since the matrix convex function \( F : \mathbb{R}^+ \times \mathbb{S}^{n}_{++} \to \mathbb{S}^{n}_{++} \) defined as \( F(\gamma, P) = \gamma^2P^{-1} \) exhibits the following matrix inequality:

\[
F(\gamma, P) \succeq F(\gamma_k, P_k) + 2(\gamma - \gamma_k)P^{-1} - \gamma_k^2P^{-1}(P - P_k)P^{-1} \tag{2.44}
\]

where \((\gamma_k, P_k)\) represents some linearization point and the second term in (2.44) corresponds to the linear approximation of \( \gamma^2P^{-1} \), for a fixed \( P = P_k \), around \( \gamma_k \). The right hand side of (2.44) could be inserted in problem (2.43) such that a classical CCP iterative procedure was employed, so why not do that?

This question will be investigated in section 2.5.1. Our experiments indicate that the reduction mode strategy tends to numerically outperform the classical CCP, in a spectral radius sense. This is our
previous mentioned insight: the continuous time version of [17] uses the classical CCP approach, in
the sense that it minimizes the spectral abscissa, directly, by resorting to linearized versions of the
matrix concave terms in (2.12) like we did in (2.44).

2.4.5 Incremental PK and Variant

This section explores a new idea to solve problem (2.10), from which two heuristics arise. As the name
suggests, the incremental PK heuristic will do incremental updates on two variables: the controller
\( K \in K_I \) and the positive definite matrix \( P \) (also called the Lyapunov certificate). The variant version of
the algorithm will propagate more variables making use of the decoupled formulation [19] mentioned in
section 2.4.1. Although the name is similar, these approaches should not be confused with the PK itera-
tions of [15] and [11]: in [15] there is an incremental definition of the variables such that the non-convex
interactions are discarded and some extra conditions impose that this option is not too severe. Here, we
do not discard any terms but instead use convex bounds to deal with undesired structures.

Define \( P = \bar{P} + \Delta P \succ 0 \) and \( K = \bar{K} + \Delta K \) where \((\bar{P}, \bar{K})\) are constant matrices and \((\Delta P, \Delta K)\)
are the optimization variables. From theorem 2.2.3 it is clear that there exists a controller \( K \in K_I \) such
that \( \rho(A + BK) < \gamma \) iff there exists a certificate \( P \succ 0 \) such that the following matrix inequality holds:

\[
\gamma^2 P - (A + BK)^T PP^{-1} P(A + BK) \succ 0. \tag{2.45}
\]

Now, using Schur’s complement in (2.45) and the definition of \( P \) and \( K \), one gets the following equivalent
condition:

\[
\begin{bmatrix}
\gamma^2 \{ \hat{P} + \Delta P \} & (A + B\{ \hat{K} + \Delta K \})^T \{ \hat{P} + \Delta P \} \\
(A + B\{ \hat{K} + \Delta K \})(A + B\{ \hat{K} + \Delta K \}) & \hat{P} + \Delta P
\end{bmatrix} \succ 0. \tag{2.46}
\]

Expanding (2.46) one can easily conclude that this matrix inequality is non convex due to the cross terms
involving \( \Delta K \) and \( \Delta P \) that appear in the off diagonal blocks. All other terms are linear in the optimization
variables. The incremental PK heuristic is a way of dealing with this non convexity, by introducing a
convex lower bound of (2.46). Although trivial, the next theorem will be the key to produce such bound:

**Theorem 2.4.5.** For any \( L, D \in \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times n} \) the following matrix inequality holds:

\[
\begin{bmatrix}
0_{n \times n} & LT \ D \\
D^T \ L & 0_{n \times n} \\
\end{bmatrix} \succeq \begin{bmatrix}
L \ T \ L & 0_{n \times n} \\
0_{n \times n} & D^T \ D
\end{bmatrix}.
\]

The proof of this theorem is trivial by noticing that \( \begin{bmatrix} L^T \\ D^T \end{bmatrix} \begin{bmatrix} L & D \end{bmatrix} \succeq 0 \). The value of theorem
2.4.5 is that it allows to decouple the variables \((L, D)\). Applying this result to (2.46), setting \( L = B\Delta K \),

\[
\begin{bmatrix}
L^T \\ D^T
\end{bmatrix} \begin{bmatrix} L & D \end{bmatrix} \succeq 0.
\]
and $D = \Delta P$, one gets the following sufficient LMI:

$$
\begin{bmatrix}
\gamma^2 \{\hat{P} + \Delta P\} & (A + BK)^T \{\hat{P} + \Delta P\} + (B\Delta K)^T \hat{P} & (B\Delta K)^T \\
\hat{P}^T & \hat{P} + \Delta P & 0_{n \times n} \\
B\Delta K & 0_{n \times n} & I_n \\
0_{n \times n} & \Delta P & 0_{n \times n}
\end{bmatrix} \succ 0
\quad (2.47)
$$

where Schur’s complement was again applied by noticing that

$$
\begin{bmatrix}
(B\Delta K)^T (B\Delta K) & 0_{n \times n} \\
0_{n \times n} & (\Delta P)(\Delta P)
\end{bmatrix} =
\begin{bmatrix}
(B\Delta K)^T & 0_{n \times n} \\
0_{n \times n} & (\Delta P)
\end{bmatrix}
I_{2n}
\begin{bmatrix}
(B\Delta K) & 0_{n \times n} \\
0_{n \times n} & (\Delta P)
\end{bmatrix}.
\quad (2.48)
$$

From (2.47) it is trivial to construct the incremental PK heuristic using, again, a reduction strategy as a searching criteria for the optimal decentralized controller:

**Algorithm 4 Incremental PK heuristic**

(i) Find some $P_0 \succ 0$ and $K_0 \in K_I$ such that $(A + BK_0)^T P_0 (A + BK_0) \prec r_0^2 P_0$ for some $r_0 > 0$. Further define $N_{max}$ as the maximum number of iterations, $\epsilon$ as a stopping tolerance and $\beta > 0$ as the shrinking factor, i.e., $\beta < 1$. Set $r_0 = \beta r_0$.

(ii) Define $P_{k+1} = P_k + \Delta P$ and $K_{k+1} = K_k + \Delta K$. The problem is to find the increments $\Delta P, \Delta K$ such that (2.47) holds with $(r, \hat{P}, \hat{K}) = (r_k, P_k, K_k)$.

(iii) If $\rho(A + BK_k) < r_k$ then $r_{k+1} = \beta r_k$; otherwise go to (vi).

(iv) Repeat until \$k = N_{max}\$ \$\bigcup\{ |\rho(A + BK_{k-1}) - \rho(A + BK_k)| < \epsilon \}\$

(v) Return $K^* \in \arg \min_{K \in K_I} \{ \rho(A + BK_1), \ldots, \rho(A + BK_{end}) \}$

The initialization given in section 2.4.4 for the decentralized convex concave heuristic can be used in this context, since both algorithms run in a reduction mode, i.e., given some initial feasible triplet $(L_0, K_0, r_0)$ such that

$$
(A + BK_0)^T L_0 (A + BK_0) \prec r_0^2 L_0,
\quad (2.49)
$$

both heuristics will incrementally try to reduce the value of $r_k$, using the corresponding convex bounds that give sufficient conditions for the decentralized controller design. For algorithm 3 we consider $L_0 = P_0^{-1}$ while $L_0 = P_0$ in heuristic 4. Note that the skeleton of algorithm 4 is the same of algorithm 3 with differences only in the correspondent feasibility problems and the implicit choice of $L_0$. Denoting $F$ as the optimal centralized controller given by algorithm 1, we can set the maximum number
of iterations \( N_{\text{max}} = \left\lceil \log_{\beta} \left( \frac{\rho(A + BF)}{r_n} \right) \right\rceil \) since algorithm 4 uses a reduction strategy. This argument was already explained in section 2.4.4.

In [19] the authors provide a more general discrete-time stability condition, i.e., an equivalent condition that extends the one given in theorem 2.2.3 by decoupling the controller \( K \) and the Lyapunov certificate \( P \). After some reformulations of theorem 2.2.3 we derived the incremental PK algorithm, an heuristic that does incremental updates in the controller \( K_k \in \mathcal{K}_I \) and the Lyapunov certificate \( P_k \succ 0 \). The condition given in [19] introduces a new variable \( Q \) that could be used to guide the searching procedure. From [19] (theorem 1) it is clear that given some \( \gamma > 0 \) there exists a controller \( K \in \mathcal{K}_I \) such that \( \rho(A + BK) < \gamma \) if and only if:

\[
\exists P \succ 0, \ Q \in \mathbb{R}^{n \times n}, \ K \in \mathcal{K}_I : \begin{bmatrix} \gamma^2 P & \ast^T \\ Q(A + BK) & Q + Q^T - P \end{bmatrix} \succ 0. \tag{2.50}
\]

Comparing problems (2.50) and (2.46) one can easily conclude that if \( Q = P \) then both formulations are exactly the same, hence (2.46) implies (2.50) in terms of feasibility. The other implication is proved in [19]. This immediately motivates a new heuristic that tries to use the same type of arguments to find a decentralized controller \( K \in \mathcal{K}_I \): the boosted incremental PK heuristic. The prefix \( \textit{boosted} \) was chosen since there exists \( n^2 \) new unconstrained variables to guide the searching procedure. Unconstrained since \( Q \) can be any matrix in \( \mathbb{R}^{n \times n} \), i.e., it exhibits no special structure not even symmetry. The variable \( Q \) appears linearly in the right lower block of (2.50) and the dependency in the off-diagonal blocks is the one encountered in (2.46) for matrix \( P \). This just to say that if we revisit (2.45), (2.46) and (2.47) taking into account \( Q \), we easily arrive at the following convex lower bound of (2.50):

\[
\begin{bmatrix}
\gamma^2 (\hat{P} + \Delta P) & (A + B\hat{K})^T (\hat{Q} + \Delta Q)^T + (B\Delta K)^T \hat{Q}^T \\
\ast^T & (\hat{Q} + \Delta Q) + (\hat{Q} + \Delta Q)^T - (\hat{P} + \Delta P) \\
B\Delta K & 0_{n \times n} \\
0_{n \times n} & \Delta Q^T \\
\end{bmatrix} \succ 0 \tag{2.51}
\]

where \( (\hat{Q}, \hat{P}, \hat{K}) \) are constant matrices and \( (\Delta Q, \Delta P, \Delta K) \) are the optimization variables. Take \( \Delta Q = \Delta P \) and conclude that 2.51 is a generalization of 2.47 if the constant terms \( (P, K, r_k) \) are equal in both algorithms and \( \hat{Q} = \hat{P} \). The \textit{boosted incremental PK} heuristic goes as follows:
Algorithm 5 Boosted incremental PK heuristic

(i) Find $P_0 > 0$, $K_0 \in \mathcal{K}$ and $Q_0 \in \mathbb{R}^{n \times n}$ such that
\[
\begin{bmatrix}
  r_0^2 P_0 & S^T \\
  Q_0 (A + BK_0) & Q_0 + Q_0^* - P_0
\end{bmatrix} > 0
\]
for some $r_0 > 0$. Further define $N_{max}$ as the maximum number of iterations, $\epsilon$ as a stopping tolerance and $\beta > 0$ as the shrinking factor, i.e., $\beta < 1$. Set $r_0 = \beta r_0$.

(ii) Define $P_{k+1} = P_k + \Delta P$, $K_{k+1} = K_k + \Delta K$ and $Q_{k+1} = Q_k + \Delta Q$. The problem is to find the increments $\Delta P, \Delta K, \Delta Q$ such that (2.51) holds with $(\gamma, P, K, Q) = (\gamma, P_k, K_k, Q_k)$.

(iii) If $\rho(A + BK_k) < r_k$ then $r_{k+1} = \beta r_k$; otherwise go to (v).

(iv) Repeat until $\{k = N_{max}\} \bigvee \{|\rho(A + BK_{k-1}) - \rho(A + BK_k)| < \epsilon\}$

(v) Return $K^* \in \arg \min_{K \in \mathcal{K}_i} \{\rho(A + BK_1), \ldots, \rho(A + BK_{end})\}$

Several remarks have to be made:

(1) The previous algorithm needs a feasible triplet $(Q_0, P_0, K_0, r_0)$ to start the iterative procedure. In section 2.4.4 we have discussed an initialization that given some $(A, B, \mathcal{K})$ could generate a triplet $(P_0, K_0, r_0)$ such that
\[
(A + BK_0)^TP_0(A + BK_0) < r_0^2 P_0
\] (2.52)
by first discovering the controller $K_0 \in \mathcal{K}_C$ that minimizes the spectral norm of the closed loop system and then finding a Lyapunov certificate $P_0 > 0$ associated with $A + BK_0$ and $r_0 = \rho(A + BK_0) + \delta$ with $\delta > 0$. The exactly same procedure can be used to initialize the boosted incremental PK heuristic if we now define $Q_0 = P_0$. This is a consequence of a previous observation: problem (2.50) can be reduced to (2.46) by choosing $Q = P$, so $(Q_0, P_0, K_0, r_0)$ is a valid initialization if the previous scheme is followed.

(2) One immediate question is the following: should the boosted heuristic always yield better results if $(\beta, \epsilon, N_{max})$ are the same in both heuristics?

Denote the boosted algorithm as $\mathcal{B}$ and the normal version as $\mathcal{N}$. If the initialization of $\mathcal{N}$ is different from that of $\mathcal{B}$ anything can happen, since both algorithms have an incremental nature. As an extreme case, assume we have an unstable system $\rho(A) > 1$ and someone provides $(K^*, P^*, r^*)$ where $K^*$ is the global minimizer of problem (2.10), $r^* = \rho(A + BK^*) + \delta$ with $\delta > 0$ and $P^*$ is the corresponding Lyapunov certificate. Assume that the optimal solution is not given by $K^* = 0$, i.e., choosing no controller at all. If this was the case the controller design problem would become trivial. We choose to initialize $\mathcal{N}$ with $(K^*, P^*, r^*)$ and $\mathcal{B}$ with $\hat{K} = 0, \hat{r} = \rho(A) + \delta$ and $\hat{P} > 0$ as the corresponding certificate such that $A^T \hat{P} A < \hat{r}^2 \hat{P}$. For $\mathcal{N}$ and $\mathcal{B}$ define $\beta = \frac{r^*}{2(\rho(A) + \delta)} < 1$ since the optimal solution is non trivial, i.e., $r^* < \rho(A)$. Denote $r_{0,N}$ as the input upper bound on algorithm
It is clear that $\mathcal{N}$ terminates immediately because when we redefine:

$$r_{0,N} \leftarrow \beta r_{0,N} = \beta r^* < r^*, \quad (2.53)$$

problem (2.51) becomes infeasible since it is impossible to improve the global solution $r^*$. This implies that $\mathcal{N}$ returns the controller $K_{end} = K_0 = K^*$. Applying a similar reasoning to $B$ we conclude that

$$r_{0,B} \leftarrow \beta r_{0,B} = \beta \hat{r} = \frac{r^*}{2} < r^* \quad (2.54)$$

so $B$ returns $K_{end} = K_0 = 0$. Conclusion: $\mathcal{N}$ found the optimal solution while $B$ found the trivial solution of $K^* = 0$, i.e., $\mathcal{N}$ yields a better controller. If we start $B$ with $(K^*, r^*, P^*)$ and $\mathcal{N}$ with $(\hat{K}, \hat{r}, \hat{P})$ we find the exact reciprocal behaviour. This algorithmic argument only illustrates that different initializations of $\mathcal{N}$ and $B$ will, generally, imply different local solutions.

If $\mathcal{N}$ is initialized from some $(P_0, K_0, r_0)$ and $B$ starts from $(Q_0, P_0, K_0, r_0)$ with $Q_0 = P_0$ the answer will remain unchanged: anything can happen. This happens since problem (2.51) is a generalization of (2.47) only if $P_k, K_k$ and $r_k$ coincide in both algorithms and $Q_k = P_k$ in $B$. Assume that this is the case and problem (2.47) is solved. This implies that there exists a feasible set of variables $(\Delta P_N, \Delta K_N)$ where the subscript allows to identify the corresponding algorithms. It is true that choosing $(\Delta Q_B, \Delta P_B, \Delta K_B) = (\Delta P_N, \Delta P_N, \Delta K_N)$ solves problem (2.47). The point is that $P_k, K_k, r_k$ will generally only coincide in the first iteration $k = 0$ since we can choose to initialize $\mathcal{N}$ and $B$ with the same $(P_0, K_0, r_0)$ and $Q_0 = P_0$. This restriction will, generally never be fulfilled in further iterations. Conclusion: even if one chooses the same initialization it is very hard to theoretically discriminate the performance of both algorithms.

This section introduced several decentralized heuristics based on different formulations of the matrix inequality given in theorem 2.2.3. Five of those algorithms were based on a idea which we denoted reduction strategy: a strategy to guide the searching procedure when no explicit objective function exists or when better numerical results are required (section 2.5.1). All heuristics were based on convex lower bounds of non convex terms hence they give sufficient conditions for decentralized stability. We also discussed some theoretical properties of the algorithms and motivated some practical problems that could benefit from decentralized control strategies. The next section compares the performance of all heuristics and answers some, postponed numerical questions.

### 2.5 Numerical Section

All experiments were developed using the package CVX with MATLAB [36], [37]. In particular we have used version 1.34 of the solver SeDuMi in a computer with the following specifications:

- Processor: Intel(R) Core(TM) i7-2600 CPU @ 3.400GhZ
- Memory (RAM): 16.0 GB
• Operating System (OS): Windows 7 Ultimate - 64 bits

From the previous sections, the solver will mostly solve feasibility problems involving strict matrix inequalities. Under CVX if we impose that $P \succ 0$ we immediately get a warning saying that strict inequalities are not really suitable, as mentioned in page 31 of [38] (Strict Inequalities). As suggested in [38] we impose that $P \succeq \epsilon I_n \iff \lambda(P)_{\text{max}} \geq \epsilon$ where $\epsilon > 0$ is some small offset. But how “small” should $\epsilon$ be?

If $\epsilon$ is too low it becomes numerically irrelevant and is ignored by the solver. If $\epsilon$ is too high the problem can become unfeasible (if not already) since we are tightening the set of feasible variables. We found good results with $\epsilon = 10^{-7}$ so every encountered strict generalized inequality is transformed into a non-strict inequality with $\epsilon = 10^{-7}$. All reduction mode algorithms run with $\beta = 0.99$. For all heuristics the maximum number of iterations is set to $N_{\text{max}} = \left\lceil \log_\beta \left( \frac{\rho(A+BF)}{r_0} \right) \right\rceil$ and the stopping tolerance takes the value of $10^{-5}$. Note that $F$ is the optimal centralized controller given by algorithm 1 (with $u_{\text{init}} = 1$) and $r_0 = \rho(A + BK_0) + 0.1$ with $K_0$ given by equation (2.40). This a valid upper bound for all reduction mode algorithms. Since we simulate systems with a moderate size, there is no issue in using $F$ to compute $N_{\text{max}}$. All algorithms run with the proposed initializations.

2.5.1 Reduction Mode vs Original CCP

In this section, we will give numerical evidence that validates one of our previous statements: the reduction mode strategy tends to work better than a classical CCP approach, in the context of decentralized stability. Consider the following experimental setup:

1. Define $n = 5$, $m = 3$ such that the low dimensions of the problem allow to perform a significant number of experiments. The sparsity count of the controller $K$, $s \in \{0, \ldots, nm\}$, is interpreted as the total number of zero entries imposed in $K$.

2. For each sparsity count $s \in \{0, \ldots, nm\}$ generate $N = 50$ system realizations $A, B$ and randomly define $N$ indicators functions $I(.)$. The pair $(A, B)$ is generated according to a standard Gaussian distribution and the $N$ random indicator functions impose a total of $s$ zero entries in the controller $K$.

3. For each sparsity count $s \in \{0, \ldots, nm\}$ run both algorithms: algorithm 3, which uses the reduction mode strategy, and the classical CCP version using (2.44) and (2.43). The stopping criteria of the former approach is to meet the previously defined stopping tolerance of $10^{-5}$.

The results are displayed in figure 2.8 where, for each sparsity count, the solid lines correspond to the median of the $N$ corresponding trials and the dotted lines represent the 0% and 90% percentiles, i.e., 90% of all $N$ experiments are squeezed between the dotted curves.

Two immediate conclusions can be drawn: the reduction mode strategy tends to achieve, in median, lower spectral radius values while being regularly stable since the red percentiles present a much lower variation (spread) while compared to the green curves. For high sparsity counts, $s \geq 11$, both heuristics present similar performances since the classical CCP approach is only incrementally superior, in median terms, while having the same spread of results. For the data of figure 2.8 we find that, from the
In $N_{nm} = 750$ trials, the reduction mode strategy gave better spectral results in 69\% of the experiments while the classical CCP only prevailed in 25\% of the epochs. Both heuristics gave the same results in the last 50 experiments (6\%) since we are imposing a totally sparse controller, i.e., $K = 0$. In sum, for our experimental setup, the reduction strategy tends to give better spectral results and, when this is not the case, the difference between the algorithms is “small”.

### 2.5.2 Numerical Comparison of Heuristics

This section is devoted to the numerical comparison of all previously defined heuristics. The experimental setup is fairly simple:

(i) For $n = 10$ and $m = 3$ we randomly generate $N = 30$ pairs $(A, B)$ and define $L$ as total number of zero entries for a controller $K \in \mathcal{K}_I$. Given $L$, we generate $N$ random sparsity sets each with a total of $L$ zero constraints.

(ii) For each generation of (i) run all six heuristics: the incremental inverse algorithm is deployed for a fixed bound $\gamma = 1$, in a bisection setup starting from $\gamma$ and under a reduction strategy. We also consider the decentralized convex concave procedure and the two variants of the incremental PK approach.

In figures 2.9 and 2.10 we have considered the previous experiment for $L = 10, 15$ respectively, and under a standard Gaussian Generation for the pair $(A, B)$. Each circle represents an experiment, identified by the colour legend on the right of each plot. We have decided to only plot successful trials, i.e., experiments for which we could produce a decentralized stabilizing controller. The colour legend also informs the reader about the number of successful experiments, for each heuristic, and some statistics regarding the original open loop systems.
To process the information of figures 2.9 and 2.10 we present table 2.1 that compares all heuristics in terms of (a) mean spectral radius of stable experiments (b) mean computational time and (c) percentage of successful experiments. The two values, for each entry of table 2.1, correspond to the information regarding the experiments with $L = 10, 15$ respectively. The missing value in the sixth line of table 2.1, mean $\rho(A)$ for 30 experiments = 3.39, median $\rho(A)$ for 30 experiments = 3.36

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Mean $\rho(A+BK)$</th>
<th>Mean Comput. Time [min]</th>
<th>% Stable Exp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental PK</td>
<td>0.64 $\rightarrow$ 0.76</td>
<td>1.03 $\rightarrow$ 0.67</td>
<td>87 $\rightarrow$ 43</td>
</tr>
<tr>
<td>Boosted Incremental PK</td>
<td>0.55 $\rightarrow$ 0.68</td>
<td>1.36 $\rightarrow$ 0.82</td>
<td>93 $\rightarrow$ 40</td>
</tr>
<tr>
<td>Incr. Inverse fixed mode w/ $\gamma = 1$</td>
<td>0.99 $\rightarrow$ 0.99</td>
<td>0.23 $\rightarrow$ 1.58</td>
<td>100 $\rightarrow$ 73</td>
</tr>
<tr>
<td>Incr. Inverse bisection w/ $\gamma = 1$</td>
<td>0.40 $\rightarrow$ 0.78</td>
<td>28.9 $\rightarrow$ 33.2</td>
<td>100 $\rightarrow$ 77</td>
</tr>
<tr>
<td>Incr. Inverse reduction</td>
<td>0.65 $\rightarrow$ ---</td>
<td>0.92 $\rightarrow$ 0.39</td>
<td>53 $\rightarrow$ 00</td>
</tr>
<tr>
<td>Decentralized convex concave</td>
<td>0.69 $\rightarrow$ 0.86</td>
<td>1.40 $\rightarrow$ 1.12</td>
<td>97 $\rightarrow$ 43</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison of heuristics: Standard Gaussian Generation.
for \( L = 15 \), indicates that no stabilizing controller was encountered, as concluded from the corresponding 0 \% stability percentage. Analysing figure 2.10 one might wonder why the bisection method was able to stabilize 23 systems while the fixed approach achieved only 22 stabilizations. Theoretically this is a contradiction but, in practice, this type of hazards can happen due to numerical precision issues. In this specific case, the bisection incremental inverse method returned a stabilizing controller found with \( \gamma = 0.9258 \), but the fixed alternative failed to find a feasible controller with \( \gamma = 1 \). For low sparsity counts, \( L = 10 \), the bisection incremental inverse method was able to stabilize all 30 systems while exhibiting the better overall spectral results, as seen in figure 2.9 and table 2.1. The drawback is that this method is 20-30 times slower than any other reduction mode heuristic, hence there is a clear compromise. If we only want to impose closed loop stability, the same method with a constant bound \( \gamma = 1 \) is a fast alternative that has the drawback of providing a spectral radius close to unit. Note that for \( L = 10 \), the mean spectral radius is only at a 0.01 distance from margin stability, i.e., from \( \rho(A + BK) = 1 \).

For high sparsity levels, \( L = 15 \), the bisection incremental inverse was able to achieve the largest stability percentage (77\%) but the boosted incremental PK algorithm is a suitable alternative that, although with only 40 \% of stabilizations, was able to achieve considerable better spectral results while being 40 times faster. In terms of all reduction mode algorithms, we elect the second and last methods of table 2.1 as computational efficient alternatives for the bisection approach: although they decrease, 7-37 \% the stability percentages they present the better spectral results, among all reduction mode approaches, while exhibiting a low computational cost. Note, in figures 2.9 and 2.10, the low horizontal spread of reduction mode methods while compared with the bisection incremental inverse heuristic.

We have also considered a different type of generation for the pair \((A, B)\). The figures/table are not presented due to lack of space. For a uniform \([0, 1]\) distribution, we find that all methods have a better performance since, for example, the bisection approach was able to stabilize \( N = 30 \) systems, each with a total of 20 zero entries. For this generation, considering \( L = 15, 20 \), the same set of conclusions applies: the bisection method achieves better stability percentages and mean spectral results with the drawback of being 20-70 times slower; and the most reliable reduction alternatives are still the second and last methods of table 2.1.

In the context of large scale systems the bisection option might be to computationally expensive; hence, algorithms 3 and 5 constitute fast alternatives that can be employed to verify if the resulting controller serves the design task at hand. If this is the case, one can achieve suitable spectral results with much less computational demands. Note that all algorithms were deployed independently, i.e., the iterations of approach A are completely independent from that of approach B. In this section, our goal was to provide a fair comparison of algorithms where no prior information or cooperation was considered. In specific applications, one can always use the solution of one method to initialize the other or, in the limit, run all algorithms sequentially using the previous best found solution as the initialization of the current approach.
Chapter 3

Optimal Sparsity Patterns

3.1 Introduction

In chapter 2, we addressed the problem of stabilizing LTI discrete time systems under information constraints: the $j$-th component of the control signal $u_j$, could only be computed using some of the measurements $x_i$. In particular we have defined an indicator function $I(.)$ that selects which measurements are available for each control component. The information constraints were mathematically reflected in a sparsity pattern $K_I$ for the controller. Given $K_I$ one can use the previous heuristics to search for optimal decentralized controllers. But what if the information constraints are not defined a priori? How can one build optimal sparsity patterns, in a spectral radius sense?

This chapter will address these questions by combining the heuristics of chapter 2 with some results from perturbation theory for optimization problems. The remainder of the chapter is organized as follows: section 3.2 defines the idea of optimal sparsity patterns and explores the combinatorial nature of the problem, i.e., optimizing over a discrete set of constrained controllers. In section 3.3 we mention some state of the art approaches for building unconstrained sparsity patterns, in a variety of related control problems. We found no methods that can be, directly applied to the problem formulation of section 3.2. After providing some background on sensitivity analysis in section 3.4, we apply these results for some heuristics of chapter 2 an derive two sparsity oracles: strategies that construct sparsity patterns. Although no optimality claims are made, some intuition is provided in section 3.5. The numerical performance of all developed methods is carried out in section 3.6 and some useful extensions are presented in appendix B.1.

3.2 Problem Formulation

Define a budget $\phi_j \in \{0, \ldots, p\}$ as the maximum number of sensor readings, that can be used to compute control component $u_j$, where $j = 1, \ldots, m$. Clearly the centralized setting can be recovered with $\phi_j = p$, for any valid $j$. The budget vector $\phi = (\phi_j)_{j=1}^m$ can be used to model the computational
resources available to compute each control component: if \( \phi_i = 1 \) then the computational entity that computes \( u_i \), can only process one output measurement. For simplicity, we will first consider static state feedback control laws

\[
u(k) = Kx(k).
\] (3.1)

The extension to a more general setup is carried out in appendix B.1.1. If controller \( K \) respects budget \( \Phi \), then the \( i \)-th row of \( K \) has a maximum of \( \phi_i \) non-zero entries. Define \( \mathcal{K}_\Phi \subseteq \mathbb{R}^{m \times n} \) as the set of controllers that respect a given budget \( \Phi \). The number of structural constraints associated with the \( i \)-th row of \( K \in \mathcal{K}_\Phi \) is given by

\[
N_i = \sum_{l=0}^{\phi_i} \binom{n}{l}
\]

where scalar \( N_i \) represents all possible combinations of choosing a maximum of \( \phi_i \) non-zero entries among \( n \) choices. Applying this reasoning for each row, the controller \( K \) can exhibit a total of \( N = \prod_{i=1}^{m} N_i \) sparsity patterns. It follows trivially that

\[
\mathcal{K}_\Phi = \bigcup_{i=1}^{N} \mathcal{K}^i_\Phi,
\]

where \( \{\mathcal{K}^i_\Phi\}_{i=1}^{N} \) denotes a partition of the feasible space. Under this framework we would like to find the controller \( K^* \in \mathcal{K}_\Phi \) that yields the stabllest close loop system, i.e.,

\[
\minimize_{K \in \mathcal{K}_\Phi} \rho(A + BK).
\] (3.4)

By (3.3), the set \( \mathcal{K}_\Phi \) is given by the union of \( N \) disjoint sets. This implies that problem (3.4) is actually combinatorial, i.e., in order to solve (3.4) we must solve \( N \) optimization problems imposing that \( K_i \in \mathcal{K}_\Phi^i \) for \( i = 1, \ldots, N \) and then pick the best found controller, in a spectral radius sense. Several remarks have to be made:

1. As seen in chapter 1, the problem of minimizing the spectral radius of an affine map of \( K \), such that \( K \in \mathcal{K}_i \), is NP hard;

2. Even if (1) could be solved efficiently, problem (3.4) has a combinatorial nature, i.e., it becomes numerically intractable to find \( K^* \in \mathcal{K}_\Phi \), with an increasing \( N \).

In chapter 2, we considered the fixed sparsity problem where several decentralized algorithms were developed and compared. We selected two approaches as time efficient methods for decentralized stability: the decentralized convex concave heuristic and the incremental PK approach. Both algorithms provide acceptable numerical results while having adequate running times. Suppose we choose the first method and denoted it as algorithm \( \mathcal{A} \). Our approach for problem (3.4) is two fold: we first design a sparsity pattern and then use \( \mathcal{A} \) to find a controller which matches this structure. Section 3.3 will develop strategies for pattern design, such that we avoid the combinatorial search, i.e., “solving” \( N \) optimal problems by means of \( \mathcal{A} \) and then choosing the best pattern/controller, in a spectral radius sense. We emphasized the “solving” aspect of \( \mathcal{A} \), since the original problem is NP hard and \( \mathcal{A} \) is merely
a heuristic, i.e., it has no guarantees of finding the global solution. The next section reviews some state of the art approaches in designing unconstrained sparsity patterns, for specific control objectives.

### 3.3 State of the Art

Whenever one combines the words *engineering* and *sparsity* in the same sentence, the expected outcome is to minimize some domain specific penalty function, with an added $l_1$ sparsity promoting regularization factor. This holds true for feature selection in machine learning, basis pursuit in compressed sensing and data fitting, portfolio optimization, etc. In [39] the authors refer to $l_1$ regularization as the “modern least squares” when pursuing sparsity. Hence it is not surprising that this approach is also embedded in controller synthesis [40]. In simple terms, the $l_1$ regularization boils down to

$$\min_{x \in \mathcal{X}} f(x) + \lambda \|x\|_1 \quad (3.5)$$

where $\lambda > 0$ is a regularization factor and the pair $(f, \mathcal{X})$ corresponds to a function and set, respectively, specific to the application at hand. When $\lambda$ is increased the term $\|x\|_1$ has increasing importance and, generally, sparser solutions can be found. Some standard analysis can be found in chapter 6 of [9] and in [40] (section 6) where the authors formalize the previous claim, in the context of synthesizing sparse controllers. This idea was applied in [17] (section 5.1) and [15] (section 4.1), for continuous-time systems. Formulation (5.1) of [17] constitutes the immediate application of $l_1$ regularization techniques to the problem of optimal stabilization:

$$\min_{K, \beta, P} \beta + \lambda \sum_{i,j} |K_{i,j}|$$

subject to

$$P \succ 0$$

$$P^{-1}(A + BK)^TP + P(A + BK) \prec 2\beta P.$$ 

It is clear that problem (3.6) is a specific case of (3.5), in the context of continuous-time decentralized control synthesis. Note however that problem (3.6) is non convex, due to the BMI constraint involving the controller $K$, the certificate $P$ and $\beta$. If the BMI is correctly modified, for discrete time systems, one can use all heuristics of chapter 1 to search for unstructured stable controllers that are encouraged to be sparse, for an increasing $\lambda$. However this approach has one downfall: this strategy is promoting sparsity but in an uncontrollable way, i.e., for a given $\Phi$ the best found controller has no guarantees of respecting the budget imposed by $\Phi$. In simple terms, the $l_1$ regularization will put some entries of controller $K$ to zero. However we do not want any zero entries. We have a budget $\Phi$ such that the $i$-th row of $K$ can only exhibit a maximum of $\phi_i$ non-zero entries. Our sparsity is structured and any valid controller should respect $\Phi$. This is the main reason why sparsity promoting techniques are not generally valid for this problem: we need sparsity ensuring methods. When reviewing the literature, one actually finds that most of sparsity design methods use some type of sparsity promoting technique. We will highlight some of the most relevant work in this context, to illustrate our point.
In [41] Polyak, Khlebnikov, and Shcherbakov propose a sparse feedback strategy, where sparsity in the controller is promoted, indirectly, through related variables. The authors make use of the following norms of a generic matrix $X \in \mathbb{R}^{n \times m}$:

$$
||X||_{r1} = \sum_{i=1}^{n} \max_{1 \leq j \leq m} |X_{i,j}|
$$

$$
||X||_{c1} = \sum_{j=1}^{m} \max_{1 \leq i \leq n} |X_{i,j}|.
$$

(3.7)

This first norm is used to induce row sparse controllers while the second promotes column sparsity. In our setup, we would be interested in row sparsity since this constitutes the motivation for considering a budget $\Phi$, of maximum number of measurements that a single control component can utilize. In [41] both problems are addressed via a fairly simple insight. As seen in section 2.2, if one consider centralized state feedback laws, there exist necessary and sufficient conditions for computing stabilizing controllers. In the context of corollary 2.2.3.1, a centralized controller can always be found by the following change of variables:

$$
K = GQ^{-1}
$$

(3.8)

where $G$ and $Q$ are optimization variables from the associated feasibility problem. One of the core ideas of [41] is that row sparsity is preserved through (3.8), i.e., if $G$ exhibits this sparsity then also will $K$:

$$
K = GQ^{-1} \Leftrightarrow \begin{bmatrix}
* & * & \ldots & * \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 0 \\
\end{bmatrix} = \begin{bmatrix}
* & * & \ldots & * \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 0 \\
\end{bmatrix} \begin{bmatrix}
* & * & \ldots & * \\
\vdots & \vdots & \ddots & \vdots \\
* & * & \ldots & * \\
\end{bmatrix}
$$

(3.9)

where the symbol $*$ represents arbitrary non-zero entries. Instead of solving a feasibility problem, like in corollary 2.2.3.1, the authors propose finding a solution $(G, Q)$ such that the corresponding row norm of $G$ is minimized, hence avoiding the BMI aspect of the problem through the typical change of variables.

The downside of this approach is that the resulting controller could be fully centralized, since there are no guarantees of finding a feasible pair $(G, Q)$ such that $G$ has, at least, one row of zeros. Our numerical experiments indicate that this simple approach does not seem to work for general systems. Even if this was the case, this method would not respect the budget $\Phi$. This can be easily seen as follows: assume that each control component can utilize $n - 1$ measurements. In this case:

$$
\Phi = \begin{bmatrix}
n - 1 \\
n - 1 \\
\vdots \\
n - 1 \\
\end{bmatrix} \in \mathbb{R}^{m}
$$

(3.10)
and the only feasible sparsity pattern generated by [41], would be $K = 0$. This happens since the controller has either fully sparse or fully filled rows: since budget (3.10) does not support centralized rows, all rows must be equal to zero such that the underlying controller is valid. Unstable systems are trivially not stabilized by $K = 0$; hence, the former idea failed for a fairly centralized budget vector, i.e., only a single zero entry, per row, is being imposed by (3.10). The idea of promoting sparsity, in rows and columns, is alternatively explored in [42], where the motivation is to design optimal actuator/sensor selection for the multichannel $H_2$ problem in continuous-time. It is not important to properly define this problem or to address the specific sparsity promotion strategies considered there. The main message is that, by penalizing the number of actuators and sensors into an control objective, the authors are able to derived an iterative process that induces some row/column sparsity in the controller, without taking into account specific row-budget constraints.

In [39] Candes, Wakin and Boyd present, justify and test a new alternative for sparsity promotion: a sequence of $l_1$ problems where the regularizer $||x||_1$ is changed to $||Wx||_1$, $W$ being a weighting matrix that, if properly designed, can lead to much sparser solutions. Instead of solving a single problem, this new approach is iterative and matrix $W$ is consecutively updated, being inversely proportional to the absolute value of individual entries of feedback gain in the previous step. This approach is referenced as reweighted $l_1$ minimization and tends to outperform the standard $l_1$ regularization. This results has already propagated to the control community, in concrete to the decentralized $H_2$ problem [43], [44] and, the previously mentioned multichannel setup with row/column sparsity promotion [42].

In [45], [46] Lin, Fardad and Jovanovic adopt a continuous-time formulation and study the problem of designing optimal sparsity patterns in the context of $H_2$ minimization. Their approach could still be classified as an $l_1$ regularization method; however, one major novelty is introduced: optimal sparsity patterns are designed through ADMM, a well known algorithm [47] to solve problems of the following type:

$$\begin{align*}
\min_{x, z} & \quad f(x) + g(z) \\
\text{subject to} & \quad Ax + Bz = c
\end{align*} (3.11)$$

where $x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$ and $(A, B, c)$ are constant matrices. The explicit algorithm can be consulted in [47] where, under some mild assumptions like convexity on $f$ and $g$, ADMM is guaranteed to find the optimal solution of (3.11). These results are crucial in distributed optimization since the structure of problem (3.11) matches that of the global consensus problem [47] (section 7).

To understand the work of Lin, Fardad and Jovanovic, in the context of sparsity design, it is not important to properly define the $H_2$ problem. One has just to known that this is a generalization of the stabilization problem, when random disturbances are considered. Assume that the $H_2$ norm represents a meaningful closed loop performance measure. In this problem, we wish to minimize this measure,
denoted as $J(\cdot)$, and design a sparsity pattern that is as sparse as possible:

$$\min_{F} J(F) + \lambda g(F)$$

(3.12)

where $\lambda > 0$ is the typical regularization factor, $g(\cdot)$ is a sparsity promoting function and $F$ is the controller. Several choices for $g(\cdot)$ are discussed in [45], [46] namely the standard $l_1$ norm and some variants of [39]. Problem (3.12) is non-convex due to the functional $J(F)$, [46] (equation 3). The author propose a two phase procedure:

(1) Identify the optimal sparsity pattern by using an ADMM approach in the following equivalent formulation of (3.12):

$$\min_{F,G} J(F) + \lambda g(G)$$

subject to $F = G$

(3.13)

where the constraint $F = G$ imposes the structure of problem (3.11), now with matrix variables. After ADMM is applied a typical sparse controller $F_1$ is generated and, from this the optimal sparsity set $S$ is identified. The optimal sparsity pattern $S$ corresponds to zero entries of $F_1$ or near zero entries, assuming some “small” thresholding value. The intrinsic non convexity implies that ADMM has no convergence guarantees; however, encouraging sparsity results where given in [45], [46].

(2) Once $S$ is identified, the regularization term is dropped from problem (3.12) and the structure of $S$ is imposed. Using some properties of controller $F_1$ the authors then use Newton’s method with a conjugate gradient to deal with the structured minimization of the functional $J(F)$. The first controller $F_1$ is essential in this phase, since it provides a good initial condition for the structured design.

This approach is hence two-fold: it simultaneously builds good sparsity patterns and finds a suitable controller, with respect to the measure $J$. Some examples show that this approach tends to be numerically effective since, by tuning $\lambda$, one can achieve sparse controllers that have small performance deterioration. Another nice feature of [45], [46] is that all source MATLAB code is available at http://people.ece.umn.edu/~mihailo/software/lqrsp. Note however, that no sparsity budget is considered in the work of Lin, Fardad and Jovanovic.

In [48] Bahavarnia, Somarakis, and Motee design sparsity patterns by an alternative method, that does not use $l_1$ regularization techniques. Their approach is based on the notion of non-fragile controllers, i.e., stabilizable controllers that preserve this property under “small” changes in controller coefficients. Given some stabilizable state feedback controller $F$, the authors propose a sparsification procedure, for $F$, such that the resulting controller can still insure closed loop stability. In practice one could start with a centralized controller or use this method as an initialization for other sparsity design strategies. The non-fragility of system $(A, B)$, with controller $F$, has a purely theoretical definition, equation 3 [48]; however, the authors propose two computable upper bounds for this quantity, which allow to derive a sparsification method for this controller. This method receives as input a desired sparsity pattern $S$, that
exhibits the following constraint:
\[ S \subseteq I_{\tilde{\rho}} = \{(i,j) : |F_{i,j}| \leq \tilde{\rho}\} \]

where \( \tilde{\rho} \) represents one upper bound for the true non-fragility quantity, and can be computed from the matrices \((A,B,F)\). For a given \( \tilde{\rho} \), one can impose sparsity patterns \( S \) that are a subset of \( I_{\tilde{\rho}} \) so, in a sense we would like for the entries of \( F \) to be small and for \( \tilde{\rho} \) to be large such that the maximal number of entries can be sparsified. This is not always possible since \( \tilde{\rho} \) is not a free parameter. It depends affinely on controller \( F \), as seen in equation 4 \([48]\). The work of Bahavarnia, Somarakis, and Motee is impressive since their method as the major advantage of being simple and computational efficient, allowing to sparsify controllers in large scale dynamical systems, say with 1000 state variables. One has restricted sparsity flexibility and, clearly, no budget \( \Phi \) can be considered. Nonetheless, the large scale aspect is very appealing, since most sparsity design approaches rely on \( l_1 \) techniques which can became computationally demanding with an increasing number of state variables. This method was compared with the work of Lin, Fardad and Jovanovic and encouraging results were found. Unfortunately, no discrete-time version of \([48]\) is available.

### 3.4 Background

In this section we summarize some results from perturbation and sensitivity analysis, applied to the problem of decentralized stabilization. Consult, for example, chapter 5.6 of \([9]\) for a more complete understanding. Let \( f : \mathbb{R}^{m \times n} \to \mathbb{R} \cup \{+\infty\} \) denote an extended value scalar convex function, and \((I,J)\) represents some index set of matrix \( K \), i.e., \( I \times J \subseteq \{1, \ldots, m\} \times \{1, \ldots, n\} \). The set \( \mathcal{X} \subseteq \mathbb{R}^{m \times n} \) is a convex set formed by \( N_1 \) linear constraints and \( N_2 \) generalized inequalities with respect to the cone of positive semi-definite matrices \( S_n^+ \), i.e.,

\[
\mathcal{X} = \{ K : A_i K B_i = C_i, \ D_j + (E_j K) + (E_j K)^T + F_j^T K F_j \succeq 0, \ i = 1, \ldots, N_1, \ j = 1, \ldots, N_2 \} \quad (3.14)
\]

where all matrices have appropriate dimensions and only \( D_j \) is required to be symmetric. Consider the following optimization problem:

\[
p^* \in \min_{K \in \mathcal{X}} \quad f(K)
\]

subject to \( K_{i,j} = 0, \ (i,j) \in I \times J \) \quad (3.15)

where \( p^* \) denotes the optimal value of the primal problem. Problem (3.15) is convex, since we are minimizing a convex function on the convex set \( \{ K : K \in \mathcal{X} \} \cap \{ K : K_{i,j} = 0, \ (i,j) \in I \times J \} \). The dual problem of (3.15) is defined as

\[
d^* \in \max_{\Lambda} \quad \inf_{K \in \mathcal{X}} \left\{ f(K) + \sum_{i \in I} \sum_{j \in J} \lambda_{i,j} K_{i,j} \right\}, \quad (3.16)
\]

49
where $\Lambda = (\lambda_{i,j})$ is the matrix of dual variables, $L(.)$ is the Lagrange dual function and $d^*$ represents the optimal value of the dual problem. Since problem (3.15) is convex we can use Slater’s condition: if there exists a strictly feasible point then strong duality holds meaning that $d^* = p^*$ and if $p^*$ is finite then problem (3.16) is solvable. A strictly feasible point $K_s$ is a point that belongs to $X$ such that all generalized inequalities are strictly satisfied, i.e., $D_j + (E_j K_s) + (E_j K_s)^T + F_j^T K_s F_j > 0$ for $j = 1, \ldots, N_2$. Problem (3.16) is solvable whenever there exists a dual variable $\Lambda^*$ such that the optimal value $d^*$ is attained, i.e., $L(\Lambda^*) = d^*$.

Consider now the perturbed version of problem (3.15) where the right hand side of the equality constraints is replaced by $\sigma_{i,j}$:

$$\min_{K} f(K)$$
$$\text{subject to} \ K_{i,j} = \sigma_{i,j}, \ (i,j) \in I \times J. \ (3.17)$$

Basic duality theory allows to relate the solution of (3.17) as a function of the perturbation matrix $\Sigma = (\sigma_{i,j})$. Let $|S|$ denote the cardinality of some set $S$. Define $h(\Sigma)$ as the optimal value of (3.17) given some perturbation matrix $\Sigma$:

$$h: \mathbb{R}^{|I| \times |J|} \to \mathbb{R} \cup \{+\infty\} \ (3.18)$$
$$\Sigma \mapsto \inf_{K \in X} \left\{ f(K) : K_{i,j} = \sigma_{i,j}, \ (i,j) \in I \times J \right\}$$

Assume that strong duality holds and $p^*$ is finite. From this one can easily conclude (section 5.6.2 [9]) that for any perturbation $\Sigma$, the following inequality holds:

$$h(\Sigma) \geq p^* - \text{trace}(\Sigma^T \Lambda^*) \ (3.19)$$
$$= p^* - \sum_{i \in I} \sum_{j \in J} \lambda^*_{i,j} \sigma_{i,j}$$

where $\Lambda^* = (\lambda^*_{i,j})$ denotes the matrix of optimal dual variables for problem (3.16). Note that $p^* = h(0)$ so the equality is attained for $\Sigma = 0$. If $h$ is convex and differentiable, we actually get that

$$\frac{\partial h(\Sigma)}{\partial \sigma_{i,j}} = -\lambda_{i,j}. \ (3.20)$$

The important aspect of this section is the interpretation of equation (3.20): the absolute value of the dual variable $\lambda^*_{i,j}$ actually allows to quantify how the corresponding constraint, $K_{i,j} = 0$, locally influences the optimal value of problem (3.15). Consider the following cases:

1. If $\lambda^*_{i,j} = 0$ then $\frac{\partial h(\Sigma)}{\partial \sigma_{i,j}} = 0$ which means that, for a small enough perturbation $\sigma_{i,j}$, $h(\Sigma) = p^*$. This basically tells us that the equality constraint $K_{i,j} = \sigma_{i,j}$, has no influence on $h(.)$, for a small enough $\sigma_{i,j}$.

2. If $|\lambda^*_{i,j}| \gg |\lambda^*_{k,l}|$ for any $k \neq i$ and $l \neq j$, we can conclude that the optimal value of (3.17) is much more reactive to changes in $\sigma_{i,j}$ then $\sigma_{k,l}$, for small enough perturbations. In particular if $\lambda^*_{i,j} > 0$
(<0) and \(\sigma_{i,j} < 0 (>0)\) then inequality (3.19) allows to guarantee that the optimal value of (3.17) will approximately increase by, at least, \(-\lambda^*_{i,j} \sigma_{i,j}\), i.e.,

\[
h(\Sigma) - p^* \geq - \sum_{i \in I} \sum_{j \in J} \lambda^*_{i,j} \sigma_{i,j}
\]

\[
\approx -\lambda^*_{i,j} \sigma_{i,j}
\]

\[
> 0.
\]

The right hand-side of inequality (3.19) corresponds to the first order approximation of function \(h(\Sigma)\) around the point \(\Sigma = 0 \in \mathbb{R}^{|I| \times |J|}\). The reasoning of (2) is valid for small enough perturbations since the high order terms of \(h(\Sigma)\) have little influence. If the perturbations are big enough it can happen that the dual variables with a bigger energy will not necessarily perturb the optimal solution of (3.17) the most.

Consider the following illustrative example: regarding formulation (3.15) take \(f([a,b,c]) = a + b^2 + c^2\), \(I \times J = \{(1,1),(1,2)\}\) and \(\mathcal{X} = \mathbb{R}^{1 \times 3}\). It is trivial to verify that \(f\) is convex, differentiable and that \(p^* = 0\) is a finite value, achieved by \((a^*, b^*, c^*) = (0,0,0)\). Slater’s condition is verified since \(\mathcal{X} = \mathbb{R}^{3 \times 1} = \{[a,b,c] : 1 \geq 0\}\), i.e., regarding the definition of \(\mathcal{X}\) we are considering no equality constraints and a single scalar inequality. Strict feasibility follows trivially hence we have strong duality for this setup. To build the dual problem (3.16), first we have to find the Lagrange dual function:

\[
L(\lambda_{1,1}, \lambda_{1,2}) = \inf_{[a,b,c]} \{ a + b^2 + c^2 - \lambda_{1,1} a - \lambda_{1,2} b \} \tag{3.22}
\]

\[
= \inf_a \left\{ a - \lambda_{1,1} a \right\} + \inf_b \left\{ b^2 - \lambda_{1,2} b \right\} + \inf_c \{c^2\}
\]

\[
= \begin{cases} 
0 & \text{if } \lambda_{1,1} = 1 \\
-\infty & \text{c.c.}
\end{cases} + \left( -\frac{\lambda_{1,2}^2}{2} \right) + (0)
\]

\[
= \begin{cases} 
-\frac{\lambda_{1,2}^2}{2} & \text{if } \lambda_{1,1} = 1 \\
-\infty & \text{otherwise}
\end{cases}
\]

From (3.22) it is trivial to verify that the solution of (3.16) is achieved by \(\lambda^*_{1,1} = 1\) and \(\lambda^*_{1,2} = 0\). In light of (2) we suspect that the constraint \(a = 0\), influences the most the optimal value of (3.17) since \(|\lambda^*_{1,1}| > |\lambda^*_{1,2}|\). Since this is the case, we take \(\sigma_{1,1} = \alpha > 0\) and \(\sigma_{1,2} = 0\) such that the constraint \(a = 0\) is perturbed by \(\alpha\), and \(b = 0\) is maintained. For this type of perturbations we get that \(h([\alpha,0]) = \alpha\) so it follows that \(|h([\alpha,0]) - p^*| = \alpha\). If instead we maintain the constraint \(a = 0\) and perturb \(b = 0\) to \(b = \alpha\) we find that \(h([0,\alpha]) = \alpha^2\) so \(|h([0,\alpha]) - p^*| = \alpha^2\).

Conclusion: for low enough perturbations \(\alpha < 1\) the reasoning of (2) was correct, and the dual variable with the largest absolute value \(\lambda^*_{1,1}\) yields the biggest difference between \(p^*\) and \(h(\Sigma)\). For perturbations \(\alpha > 1\) we actually get the opposite behaviour and \(b = 0\) is the more influential constraint. This illustrates that the reasoning of (2) is only a local sensitivity analysis.
This concludes our revision on perturbation theory applied to convex problems with a specific structure (3.15). The main message is that, under some assumptions, the dual variables of the unperturbed problem (3.15) contain valuable information regarding its perturbed counterpart (3.17), namely we can quantify how much a given constraint, \( K_{i,j} = 0 \), locally influences the optimal value of problem (3.15).

### 3.5 Sparsity Oracles

Given some budget vector \( \Phi \in \mathbb{R}^n \) we have assumed the existence of an algorithm \( \mathcal{A} \), that can be used to “solve” problem (3.4) under some fixed sparsity pattern \( \mathcal{K}_\Phi \). This chapter will develop oracles for optimal sparsity patterns: strategies that will try to find the optimal pattern for problem (3.4), by avoiding the exhaustive search over the partition \( \{ K_\Phi^i \}_{i=1}^{N} \). Our main motivation will be to use the results of section 3.4 regarding perturbation theory: an oracle \( \mathcal{O} \) will generate a feasible set \( \mathcal{K}_O \subseteq \mathcal{K}_\Phi \) by using the dual information of some optimization problems that matches the given framework. Our oracles are inspired by the ideas of chapter 2. Note that there is no dependency between oracles and algorithm \( \mathcal{A} \): the oracle will simply generate a feasible sparsity set. After this, we can employ any heuristic of chapter 2 to find the corresponding decentralized controller. In section 3.6.1 we will study the numerical performance of each oracle, for the previous selected algorithm \( \mathcal{A} \). The next section formalizes and motivates each oracle, by means of perturbation theory arguments.

#### 3.5.1 Incremental Inverse Oracle

In this section we will study a class of oracles, denoted as \( \mathcal{O}_{II} \), that use dual information of the iterative Incremental Inverse algorithm. Hence the \( \mathcal{II} \) nomenclature in \( \mathcal{O}_{II} \). Two approaches will be considered: static oracles, \( \mathcal{O}_{II}^s \), and iterative ones \( \mathcal{O}_{II}^i \). First we will introduce the static version and motivate the intuition for this method.

Given some budget vector \( \Phi \in \{0, \ldots, n\}^m \) a feasible sparsity structure can always be found by imposing that \( K = 0 \), i.e., considering a full sparse controller. Consider the following experiment: assume we deploy the incremental inverse heuristic, algorithm 2, for a fixed bound \( \gamma = 1 \) and imposing a full sparse controller. The heuristic will finish at the second iteration, since one of the stopping criteria is immediately met:

\[
|\rho(A + BK_1) - \rho(A + BK_2)| = |\rho(A) - \rho(A)| = 0 < \epsilon, \ \forall \ \epsilon > 0.
\]  

(3.23)

This might seem redundant since the underlying process served as a pass-all filter: given a full sparse controller it returned the same full sparse controller. However this is not entirely true, since the underlying method also returned valuable dual information, namely the dual variables \( \lambda_{*,j}^i \) associated with the constraints:

\[
\lambda_{*,j}^i: \ K^*(i,j) = 0, \ (i, j) \in \{1, \ldots, m\} \times \{1, \ldots, n\}.
\]  

(3.24)
The dual variable with the largest absolute value indicates which constraints have a greater influence on the optimal value of problem (2.25), assuming it matches the framework of section 3.4. The static oracle \( O_{II}^s \) uses one very simple idea: construct the set \( K_{O_{II}^s} \subseteq K_{\Phi} \) using the most influential dual variables corresponding to the last iteration of the incremental inverse method.

**Algorithm 6** Incremental Inverse static oracle \( O_{II}^s \)

(i) For any budget \( \Phi \), run the incremental inverse heuristic, algorithm 6, with \( \gamma = 1 \) and imposing a full sparse controller, i.e., \( K = 0 \).

(ii) Problem (2.25) will only be solved twice; hence, collect the corresponding matrix of dual variables \( \Lambda_{II}^s \):

\[
\Lambda_{II}^s \in \arg \max_{\Lambda} L(\Lambda)
\]

where \( L \) denotes the Lagrange Dual function of problem (2.25) with \((P_k, Q_k) = (P_1, Q_1)\). In general, there is no closed form expression for function \( L \).

(iii) The dual information \( \Lambda_{II}^s \) allows to construct the corresponding feasible set \( K_{O_{II}^s} \): the zero entries that defined the set \( K_{O_{II}^s} \) correspond to the elements of \( \Lambda_{II}^s \), with the lowest absolute value such that the entire budget \( \Phi \) is spent.

As an illustrative example of algorithm 6 consider \( m = 2, p = 3, B = [1, 2]^T \) and assume the following matrix of dual variables:

\[
\Lambda_{II}^s = \begin{bmatrix}
0.1 & 3 & -1.1 \\
-0.5 & 2.1 & 2.3
\end{bmatrix}
\]

(3.26)

The first control component \( u_1 \) can only be computed using one measurement since \( B_{1,1} = 1 \). The matrix \( \Lambda_{II}^s \) suggests that the most significant measurement, for this control component, is \( y_2 \) since the entry \((1,2)\) has the largest absolute value among the first row of \( \Lambda_{II}^s \). By a similar reasoning, the second control component \( u_2 \) should be computed using the second and third measurements, i.e., \( y_2 \) and \( y_3 \).

Under these decisions the oracle \( O_{II}^s \) outputs the following set

\[
K_{O_{II}^s} = \{ K \in \mathbb{R}^{2 \times 3} : K_{1,1} = K_{1,3} = K_{2,1} = 0 \}.
\]

This is the type of construction mentioned in point (iii) of algorithm 6. Several remarks have to be made:

1. To apply the results of section 3.4 we have to verify several properties regarding problem (2.25) with \((P_k, Q_k) = (P_1, Q_1)\). Namely (a) the problem is convex, (b) Slater’s conditions holds and (c) the primal solution is finite. The problem at hand is the following:

\[
\begin{align*}
\text{minimize} & \quad \text{trace}(P_1 \Delta Q + Q_1 \Delta P + P_1 Q_1) \\
\text{subject to} & \quad \begin{bmatrix}
\gamma^2 P & (A + BK)^T \\
A + BK & Q
\end{bmatrix} \succ 0 \\
& \quad \begin{bmatrix}
P & I_n \\
I_n & Q
\end{bmatrix} \succeq 0, \quad P, Q \succ 0, \quad K \in K_{II} \\
P = P_1 + \Delta P, \quad Q = Q_1 + \Delta Q.
\end{align*}
\]

(3.27)
This setup corresponds to the second iteration of algorithm 2; hence, we already have at our disposal a feasible triplet \((P_1, Q_1, L)\) where \(L\) denotes the controller from the first iteration. Condition (a) follows trivially since we are minimizing a linear functional over a convex set. To prove Slater’s condition we have to discover a strictly feasible point for (3.27). This might seem non trivial because of the non-strict inequality involving the variables \((P,Q)\). Given any \(\epsilon > 0\) use the following assignment of variables:

\[
\begin{align*}
P &= P_1 + \epsilon I_n \\
Q &= Q_1 \\
\Delta P &= \epsilon I_n \\
\Delta Q &= 0_{n \times n} \\
K &= L
\end{align*}
\]  

and verify that all inequalities of problem (3.27) hold strictly, namely:

\[
\begin{bmatrix} P & I_n \\ I_n & Q \end{bmatrix} \succeq 0 \iff P - Q^{-1} \succeq 0 \iff \epsilon I_n + (P_1 - Q_1^{-1}) \succ 0
\]  

\[
\begin{bmatrix} \gamma^2 P & (A + BK)^T \\ A + BK & Q \end{bmatrix} = \begin{bmatrix} \epsilon I_n & 0_{n \times n} \\ 0_{n \times n} & 0_{n \times n} \end{bmatrix} + \begin{bmatrix} \gamma^2 P_1 & (A + BL)^T \\ A + BL & Q_1 \end{bmatrix} \succ 0,
\]  

where the right most premisses are true since \((P_1, Q_1, L)\) is a feasible triplet that resulted from the first iteration of algorithm 2. The other two inequalities hold trivially, i.e., \(Q = Q_1 > 0\); and since \(\epsilon > 0\), the variable \(P = P_1 + \epsilon I_n\) is also positive definite. This proves strictly feasibility and hence strong duality holds for problem (3.27). Condition (c) can also be verified. Clearly \(p^* \neq +\infty\) since a feasible solution can be found by choosing \((\Delta P, \Delta Q) = (0, 0)\) and \((P_1, Q_1, L)\) is a feasible triplet from the first iteration. Remember that, in chapter 2, we have shown that any initialization \((P_0, Q_0)\) can be employed; hence, the existence of \((P_1, Q_1, L)\) is guaranteed. To show that \(p^* \neq -\infty\) we make use of Theorem 2.1 of [49], namely that the optimal value of

\[
\begin{align*}
\text{minimize}_{P Q K} & \quad \text{trace}(Q_1 P + P_1 Q) \\
\text{subject to} & \quad (P, Q, K) \in \mathcal{X}
\end{align*}
\]

is lower bounded by \(2n\), where \(\mathcal{X}\) denotes an arbitrary convex set and the pair \((P_1, Q_1)\) exhibits the following relationship:

\[
\begin{bmatrix} P_1 & I_n \\ I_n & Q_1 \end{bmatrix} \succeq 0.
\]  

To apply the result of [49] to problem (3.27) we simply have to increment the ambient space of \(\mathcal{X}\).
and define:

\[ X = \{ (P, \Delta P, Q, \Delta Q, K) : P > 0, \ Q > 0, \ K \in K_I, \ P = P_1 + \Delta P \\
Q = Q_1 + \Delta Q, \ \gamma^2 P - (A + BK)^T Q^{-1} (A + BK) > 0 \} \quad (3.33) \]

Applying definition (3.33) to problem (3.31) shows that the optimal value of (3.27) is finite and hence the results of section 3.4 apply.

(2) Perhaps the most pertinent question is the following: what is the connection between the dual information, \( \Sigma_{II} \), and the original problem (3.4)? How can the dual information of problem (3.27) allow us to infer on problem (3.4)?

This point summarizes the heuristic nature of our oracles. Algorithm 6 identifies the constraints which have the largest discriminative power to influence the optimal value of (3.27), at least locally. Given the dual information of (3.25), it is indeed difficult to theoretically infer on problem (3.4) simply because the problem are not equivalent: one is \( \mathcal{NP} \) hard while the other is convex. With this in mind, our reasoning underlies one fundamental assumption: “good” constraints for problem (3.27) are also “good” for the original problem. The intuition for this assumption is the following: by using the constraints selected by algorithm 6 we will perturb, the most, the optimal value of (3.27). If problem (3.27) achieves its finite lower bound, then the resulting controller \( K \in K_I \) insures that \( \rho(A + BK) < \gamma = 1 \) so, in a sense, we can give a high level probabilistic interpretation for algorithm 6: the “probability” that the underlying constraint influences the optimal value of (3.4) is given by the corresponding absolute value of the normalized dual variable. In simple terms, we assume that:

\[
\{ \text{Influence of measurement } j \text{ in control } i \} = \begin{cases} 
\frac{|(\Sigma_{II})_{i,j}|}{||\Sigma_{II}||_{max}}, & \text{if } \Sigma_{II} \neq 0 \\
0, & \text{otherwise}
\end{cases} \quad (3.34)
\]

where \( ||\Sigma||_{max} \) denotes the maximum norm, i.e., the largest absolute value of a generic matrix \( \Sigma \in \mathbb{R}^{p \times q} \). Again (3.34) gives only a high-level notion of probability allowing to justify our intuition for algorithm 6: if we have to choose between two constraints, we eliminate the one with the largest absolute dual variable, since variations on the optimal value of (3.27) may allow to infer on problem (3.4). In particular if (3.27) achieves the finite optimal value then the solution (3.4) is strictly upper bounded by \( \gamma = 1 \).

(3) One might wonder: in algorithm 6 point (iii), why do we need to spend the entire budget \( \Phi \)?

Going back to problem (3.4) our goal is to find the feasible assignment which is optimal in a spectral radius sense. This justifies our option: by eliminating constraints, the optimal value of (3.4) can only improve, so one should use all resources such that the final budget is respected. Note that the budget \( \Phi \) was defined taking into account physical constraints: it should reflect the computational
constraints of the underlying entity that receives output measurements and computes control signals. If entity $i$ has low capabilities it should have a corresponding low computational budget $\phi_i$. We emphasize that this full budget option is still sparse friendly: if one requires a sparser control law the budget $\Phi$ should be adjusted.

Algorithm 6 has a static nature, in the sense that a single call to the underlying heuristic served as a surrogate for constructing the set $K_{Oi} \in K_\Phi$. Another option is to use an iterative oracle that builds the same set using dual informations from several calls. We propose an iterative approach that instead of doing all assignments of $K_{Oi}$ in one, each assignment is associated with one run of the incremental inverse heuristic. The basic reasoning is to use the idea of algorithm 6 but in step (iii), we only choose the constraint associated with the largest absolute value of the underlying dual variable. We then impose this constraint and repeat the process, iteratively, until the entire budget vector is spent.

**Algorithm 7** Incremental Inverse iterative oracle $O_{II}$

(i) For any budget $\Phi$, run the incremental inverse heuristic, algorithm 2, with $\gamma = 1$ and imposing a full sparse controller, i.e., $K = 0$.

(ii) Problem (2.25) will only be solved twice; hence, collect the corresponding matrix of dual variables $\Lambda_{II}^1$:

$$\Lambda_{II}^1(1) \in \arg \max_{\Lambda} L(\Lambda)$$

where $L$ denotes the Lagrange Dual function of problem (2.25) with $(P_k, Q_k) = (P_1, Q_1)$. In general, there is no closed form expression for function $L$.

(iii) Identify the index set associated with the largest absolute value dual variable:

$$(l_1, m_1) \in \arg \max_{(l,m)} \left| \left\{ \Lambda_{II}^1(1) \right\}_{l,m} \right|$$

(iv) Use $(l_1, m_1)$ to eliminate the constraint:

$$K_{l_1,m_1} = 0$$

i.e., the controller $K \in \mathbb{R}^{m \times p}$ can now use measurement $m_1$ to compute control $l_1$. The set $K_{Oi}$ will be constructed after $\sum_k \phi_k$ eliminations/relaxations, when the entire budget $\Phi$ is used. Steps (i)-(iv) correspond to the first iteration, $n = 1$.

(v) In iteration $n > 1$ run the incremental inverse heuristic with $\gamma = 1$ and imposing the full sparse pattern with the previous $n - 1$ relaxations. If necessary, adjust the maximum number of iterations, $N_{oracle}$, of the incremental inverse heuristic to have a time-efficient oracle.

(vi) After (v), collect the dual variables $\Lambda_{II}^n$ associated with the last iteration of algorithm 2. For $n = 1$, algorithm 2 stopped after two iterations, since $K = 0$. This will generally never repeat since, after step (iv), the controller is not obliged to be fully sparse. Use equation (3.36) to define $(l_n, m_n)$ substituting $\Lambda_{II}^1$ by $\Lambda_{II}^n$.

(vii) Repeat (v) and (vi) until a feasible set is found, i.e., after doing all $\sum_k \phi_k$ assignment of measurements to controls.

Using the previous arguments, one can easily conclude that the results from section 3.4 still hold for algorithm 7: for any sparsity constraints, the last iteration of algorithm 2 constitutes a convex problem with a finite optimal value for which strong duality holds. The probabilistic interpretation gave for algorithm 6 still applies: the dual information encodes, at least locally, the constraints which are most likely to influ-
ence the optimal of problem (3.4), indirectly, by perturbing the optimal value of the their corresponding problem, i.e., the last feasible iteration from algorithm 2.

Comparing algorithm 6 and 7 is clear that the former is much more computationally demanding, since we have to run the incremental inverse heuristics $\sum_{k=1}^{m} \phi_k$ times, until a feasible sparsity set is generated. The first oracle only needs to solve two optimization problem, i.e., the first two iterations of the same heuristic. A practical suggestion is to adjust the maximum number of iterations for the incremental inverse heuristic, point (v) in oracle 7. This option will be considered in the numerical section 3.6.1.

Intuitively, it seems that the iterative oracle is more suitable, since it uses more insight than the static approach: to relax the $n$-th constraint we need to use the information regarding the previous $n - 1$ relaxations. Algorithm 6 can be too greedy, trying to do all assignments in one step. Assume the following scenario, where $m = 2, p = 3$ and $B = [1, 1]$:

$$\Lambda^{s}_{II} = \begin{bmatrix}
0.001 & 0.0016 & -0.001 \\
-0.07 & -0.0034 & 2.7
\end{bmatrix}.$$ (3.38)

The static oracle, $O^{s}_{II}$, would assign $y_2$ to $u_1$ and $y_3$ to $u_2$. The second choice is reasonable since there is a considerable difference between $(\Lambda^{s}_{II})_{2,3}$ and the other entries of the second row of matrix $\Lambda^{s}_{II}$. However, for the first control component, there is no clear distinction in the dual information so assigning $y_2$ to $u_1$ can be redundant. The iterative oracle, $O^{i}_{II}$, would only pick the reasonable assignment and refresh the dual information to deal with the redundancy observed for $u_1$. This is the basic idea behind algorithm 7. No optimality claims are made for any oracle but a numerical study will be carried out in section 3.6.

### 3.6 Numerical Section

In the subsequent sections, after a feasible sparsity set is generated, $K_O \subseteq K_\Phi$, all experiments use algorithm 3 from chapter 2 where the software configurations remain unchanged, i.e., processor, RAM, solver, precision values, stopping tolerance, initialization, etc. Consult section 2.5 to get access to the original computational setup.

#### 3.6.1 Comparison of Oracles

The goal of this section is to provide a numerical comparison of all previous oracles, in terms of computational time and resulting spectral radius. The decentralized convex concave heuristic is selected as algorithm $A$, as already mentioned in section 3.2. Consider the following experimental setup:

1. Given a budget vector $\Phi$ we generate 100 stabilizable pairs $(A, B)$ with $n = 10, m = 3$ under a standard Gaussian distribution.
(2) For each generation, the two oracles of section 3.5 are executed, generating two corresponding feasible sparsity sets. In the incremental inverse iterative oracle, algorithm 6, each call to the incremental inverse heuristic runs for a maximum of $N_{oracle} = 20$ iterations. This bound simply appears to limit the computational time of the oracle $O_{II}$, i.e., $N_{oracle}$ tends to be smaller than $N_{max}$ which is the maximum number of iterations defined in chapter 2 for any stabilizing algorithm $A$.

As an baseline, we also consider a purely randomized approach: given a budget vector $\Phi$ we randomly generate a feasible assignment of measurements to control components. This allows to investigate whether the extra computational burden, of each oracle, improves this naive approach.

(3) For each strategy of (2) we infer on the optimality of the previously generated sparsity sets, by executing algorithm $A$.

Figure 3.1 and table 3.1 present some results when one considers a control budget of $[8, 6, 3]^T$. Under this budget, the computational entity that computes $u_1$ can process almost every output measurement (80%), while $u_3$ can only be computed with 30% of the available observations. In table 3.1, $t_O$ denotes the amount of time associated with each oracle and $t_T$ is the total time needed to generate the underlying decentralized controller: oracle time + time of algorithm $A$. The abscissa in figure 3.1 corresponds to $t_T$.

The intuition of section 3.5 is experimentally verified: the oracle that uses incremental information tends to outperform the non incremental approaches. Note the stabilization percentages of $O_{II}$. Oracle $O_{II}$ is approximately 120 times slower than any other method. Again we find numerical evidences similar to that of chapter 2: the slowest method provides better numerical results. The iterative method also provides slightly better mean spectral results, as observed in the point clouds of figure 3.1 and the first column of table 3.1.
Table 3.1: Comparison of Oracles: Standard Gaussian Generation with budget = [8, 6, 3]T.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Mean $\rho(A + BK)$</th>
<th>Mean $t_\mathcal{O}$ [s]</th>
<th>Mean $t_T$ [s]</th>
<th>% Stable Exp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>II static oracle</td>
<td>0.82</td>
<td>1.24</td>
<td>72.88</td>
<td>59</td>
</tr>
<tr>
<td>II iterative oracle</td>
<td>0.79</td>
<td>152.10</td>
<td>231.23</td>
<td>73</td>
</tr>
<tr>
<td>Random oracle</td>
<td>0.83</td>
<td>$10^{-8}$</td>
<td>74.51</td>
<td>56</td>
</tr>
</tbody>
</table>

Figure 3.2 summarizes the information of generation 51, highlighting the sparsity patterns found by each oracle and the corresponding spectral radius once algorithm $\mathcal{A}$ is employed. For this particular run, all developed methods were able to stabilize the system while a random approach led to instability. The random generation 3.2(c) only differs from the best solution 3.2(a) by 6 displacements, i.e., by slightly changing the underlying pattern we got considerable spectral differences. This justifies the need for sparsity designing oracles: there is discriminative power in selecting the right assignment of measurements to control components. A right choice may render a suitable stable system 3.2(a), 3.2(b), while a wrong choice may lead to instability 3.2(c). The previous conclusions indicate that oracle $\mathcal{O}_{II}$ provides better numerical results while being considerably slower. The time performance of algorithm 6 can be easily improved: simply lower $N_{\text{oracle}}$ which is the maximum number of iterations for each incremental inverse call. This suggests an interesting question: in this scenario, will the performance of the oracle also decrease? If we make less dual improvement, in each iteration of algorithm 7, will we get a worst stabilizing controller after algorithm $\mathcal{A}$ is executed?

To answer this question, oracle $\mathcal{O}_{II}$ was re-executed with the previous experimental setup, i.e., for the same budget $\Phi$ and pairs $(A, B)$, algorithm 6 was deployed with $N_{\text{oracle}} = 3$ instead of $N_{\text{oracle}} = 20$. This reduction should be sufficient such that we, at least, match the time performance of the other approach.
The results are displayed in figure 3.2. Note that the stability percentages, improved from 73\% to 84\% while oracle $O_{II}$, with $N_{oracle} = 3$, is approximately seven times faster then with $N_{oracle} = 20$. This result is very non-intuitive, reason why it is truly remarkable: by decreasing $N_{oracle}$ we get more stable systems in a less amount of time. There is no apparent disadvantage.

![Figure 3.3: Same Generation and budget, now with $N_{oracle} = 3$.](image)

Table 3.2 compares the performance of the oracle $O_{II}$ for different $N_{oracle}$ values, considering the same algorithm $A$, budget $\Phi$ and pairs $(A, B)$. After the success of setting $N_{oracle} = 3$, the immediate impulse would be to further decrease this value. The third row of table 3.2 summarizes the results, with $N_{oracle} = 1$. Although the mean spectral radius was slightly improved (0.01), the computational time gained ($\approx 8s$) is not beneficial: the stability margins dropped 18\%. This lead us to following conclusion: although a reduction of $N_{oracle}$ is beneficial, the decrease should not be too drastic. In practice, one could try a small number of values, say $N_{oracle} \in \{3, 6, 9, \ldots, 18\}$, apply algorithm $A$ and then choose the most suitable controller. We found very good results with $N_{oracles} = 3$ as seen in figure 3.3 and table 3.2.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Mean $\rho(A + BK)$</th>
<th>Mean $t_O$ [s]</th>
<th>Mean $t_T$ [s]</th>
<th>% Stable Exp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>II iterative oracle w/ $N_{oracle} = 20$</td>
<td>0.79</td>
<td>152.10</td>
<td>231.23</td>
<td>73</td>
</tr>
<tr>
<td>II iterative oracle w/ $N_{oracle} = 3$</td>
<td>0.77</td>
<td>22.05</td>
<td>104.61</td>
<td>84</td>
</tr>
<tr>
<td>II iterative oracle w/ $N_{oracle} = 1$</td>
<td>0.76</td>
<td>13.88</td>
<td>93.35</td>
<td>66</td>
</tr>
</tbody>
</table>

Table 3.2: Comparison of oracle $O_{II}$: different $N_{oracle}$ values.

### 3.6.2 Oracles vs Exhaustive Search

In this section we will compare the previous developed oracles, with the exhaustive search over the entire partition $\{K^i\}_{i=1}^N$, where $N$ corresponds to the total number of feasible sparsity patterns, previously defined in section 3.2. This section is of great importance, since it provides another reliability test of the developed methods: assuming we have time to do an exhaustive search, how optimal are our oracles?
To really understand why the exhaustive search is numerically intractable we present table 3.2 which illustrates the evolution of $N$ assuming a $n-1$ control budget $\Phi$, i.e., each of the $m$ control components can be computed, potentially, from $n-1$ state variables:

$$\Phi = \begin{bmatrix} n-1 \\ n-1 \\ \vdots \\ n-1 \end{bmatrix} \in \mathbb{R}^m$$

<table>
<thead>
<tr>
<th>m/n</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>15</td>
<td>31</td>
<td>63</td>
<td>127</td>
<td>255</td>
<td>511</td>
<td>1023</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>9</td>
<td>49</td>
<td>225</td>
<td>961</td>
<td>3969</td>
<td>16129</td>
<td>65025</td>
<td>261121</td>
<td>1046529</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>27</td>
<td>343</td>
<td>3375</td>
<td>29791</td>
<td>250047</td>
<td>2048383</td>
<td>16581375</td>
<td>133432831</td>
<td>1070599167</td>
</tr>
</tbody>
</table>

Table 3.3: $N$ as a function of $(n, m)$ for a $n-1$ control budget.

The first column of table 3.3 is a consequence of having a $n-1$ control budget: for $n = 1$ only a full sparse controller will respect $\Phi$, regardless of the number of control components. Assume algorithm $\mathcal{A}$ takes $\approx 1$ minute to find a decentralized controller, given some sparsity pattern. Table 3.3 implies that, for low dimensions $(n, m) = (10, 3)$, the exhaustive search would take, approximately, 2037 years of computation to find the optimal sparsity pattern, i.e., the “solution” of problem (3.4) by means of $\mathcal{A}$. The greedy search explodes even for small dimension; hence, it is surely not applicable for large scale systems. With this in mind, our experimental setup will consider a tractable choice of state variables $n$ and control components $m$:

1. For $(n, m) = (3, 2)$ assume a $n-1$ control budget of $\Phi = [2, 2]^T$, and randomly generate 100 Gaussian pairs $(A, B)$.

2. For each system of (1), run all previous oracles together with the exhaustive method of evaluating all 49 sparsity possibilities and picking the one which gives the best outcome, in a spectral radius sense. Taking into account the results of section 3.6.1 we select $N_{\text{oracle}} = 3$ for $O_{11}$.

Figure 3.4 displays the cumulative probability of selecting a sparsity pattern among the $l$ best options, in a spectral radius sense, after algorithm $\mathcal{A}$ is applied. Note that $l \in \{1, \ldots, 49\}$ since, for each system $(A, B)$, there exist 49 feasible sparsity patterns for controller $K$. In simple terms, for a given number $l$ and oracle $\mathcal{O}$ the ordinate of figure 3.4 represents

$$\mathcal{P}\{\text{Oracle } \mathcal{O} \text{ selecting a pattern among the } l \text{ best options} \} = \frac{\# l \text{ optimal decisions}}{100}.$$  (3.39)

This justifies the unit step behaviour of the optimal oracle: this oracle always chooses the best sparsity pattern hence, the number of $l = 1$ optimal decisions is trivially equal to 100. The worst possible oracle would be represented, in figure 3.4, by a single unit Dirac at $l = 49$, i.e., this oracle would make the
worst possible decision, in all 100 trials. From figure 3.4 it is evident that the developed oracles tend
to outperform the naive approach of selecting a random sparsity pattern. In concrete, we observe that
our oracles where among the \( l = 10 \) optimal decisions in 50\% of the trials, while a randomized approach
is only at a \( l = 20 \) level, for the same cumulative probability value. In this experiments, both oracles
present similar cumulative probabilities curves but figure 3.4 demonstrates that \( O_{II}^{s} \) tends to outperform
strategy \( O_{II}^{i} \). We conjecture that this is due to the low dimensions of the problem since, as seen in
section 3.6.1, the incremental inverse iterative oracle is quite effective, in the context of decentralized
stability. Table 3.4 summarizes the information of figure 3.4 and shows that the optimal exhaustive oracle

\[ \begin{array}{cccc}
\text{Heuristic} & \text{Mean } \rho(A + BK) & \text{Mean } t_O [s] & \text{Mean } t_T [s] & \% \text{ Stable Exp.} \\
\hline
\text{II static oracle} & 0.04 & 0.74 & 93.08 & 99 \\
\text{II iterative oracle} & 0.06 & 2.67 & 92.95 & 100 \\
\text{Random oracle} & 0.32 & 10^{-8} & 54.32 & 66 \\
\text{Optimal oracle} & 0.01 & 3059.18 & 3059.18 & 100 \\
\end{array} \]

Table 3.4: Developed oracles vs Exhaustive search table.
Chapter 4

Decentralized Computation of Decentralized Controllers

4.1 Introduction

All ideas of chapter 2 produce decentralized controllers by centralized computations, i.e., a single computational entity was responsible for executing all operations that define the algorithmic procedure. This chapter tackles the decentralized computation of decentralized controllers, i.e., the computation of the controller is carried out by several agents that can communicate through an underlying network. The proof of convergence for DPDS can be found in appendix C.

First we define this idea of decentralized computations (section 4.2) and proceed to construct a provable decentralized stabilizing algorithm for a specific class of systems: closed-loop positive systems (section 4.3). In section 4.4 we illustrate the convergence properties of our method. We conclude this chapter with a particular application of the developed method: sharing the computational load in a network of computers (section 4.5).

4.2 Problem Formulation

For simplicity, consider static state feedback control laws. Given a stabilizable system \((A, B)\), let \(\mathcal{G} = (\mathcal{N}, \mathcal{E})\) denote an undirected graph, where each node represents an agent that can measure a state variable. The set of nodes is partitioned into control, \(\mathcal{N}_c\), and non-control nodes, \(\mathcal{N} \setminus \mathcal{N}_c\). Control node \(i \in \mathcal{N}_c\) has to compute some component of the control signal from local information, i.e., information from neighbouring nodes in graph \(\mathcal{G}\). The concept of information will be made clear further on. \(I_c(i)\) denotes the control component computed by node \(i \in \mathcal{N}_c\). An edge \((i, j) \in \mathcal{E}\) indicates that nodes \(i\) and \(j\) can exchange bidirectional information between themselves, and \(\mathcal{N}_i\) is the neighbouring set of node \(i\).
\[ i \in \mathcal{N}. \text{ More formally:} \]

\[ \mathcal{N}_c = \{i_1, \ldots, i_m\} \subseteq \{1, \ldots, n\} = \mathcal{N} \]

\[ I_c(i) : \mathcal{N}_c \rightarrow \{1, \ldots, m\} \]

\[ \mathcal{E} = \{(i_1, j_1), \ldots, (i_{|\mathcal{E}|}, j_{|\mathcal{E}|})\} \]

\[ \mathcal{N}_i = \{j \in \mathcal{N} : (i, j) \in \mathcal{E} \lor (j, i) \in \mathcal{E}\} \]

where \(|\mathcal{E}|\) denotes the total number of edges of graph \(G\). We will also assume that \(G\) is a connected graph, meaning that there exists a path between any two nodes, i.e., there is no isolated state variable. As before, our goal is to compute a decentralized controller \(K\) that stabilizes system \((A, B)\), where the sparsity of the controller is dictated by the topology of the network \(G\):

\[ \forall \; i \in \mathcal{N}_c : \; K_{I_c(i), j} = 0, \; \text{ for } j \not\in \mathcal{N}_i. \quad (4.2) \]

An example of this setup was given in section 2.3.1. In chapter 2, all derived heuristics had a centralized nature, i.e., a single computer/processor will compute \(K\) and the concept of graph \(G\) is absent in the design phase. If a valid controller is found, we can then implement it in a decentralized fashion i.e., respecting the underlying sparsity pattern. This justifies the title of the chapter “Centralized Computation of Decentralized Controllers”.

This chapter approaches a more challenging problem: the decentralized design of controller \(K\). Distributed optimization refers to the capability of solving an optimization problem by combining local computations (at each node) with some form of message passing (communication protocol) between neighbouring nodes [47]. To formalize our interpretation of decentralized design of stabilizing controllers, we present the following definition:

**Definition 4.2.1.** Given a stabilizable system \((A, B)\) and graph \(G\) assume that each node \(i \in \mathcal{N}\) is initialized with a non-trivial private variable \(V_i(0) \in \mathbb{R}^{l_i \times p_i}\). If \((i, j) \in \mathcal{E}\) then node \(i\) can receive \(V_j(k)\) and send \(V_i(k)\) from/to node \(j\), for any \(k \geq 0\). \(D\) is a decentralized design strategy if and only if:

(i) For every node \(i \in \mathcal{N}\), the variable \(V_i(k + 1)\) is updated from neighbouring past information:

\[ \{V_i(m), V_j(m)\}_{0 \leq m \leq k}, \; \text{ for } j \in \mathcal{N}_i. \quad (4.3) \]

(ii) Asymptotically, control node \(i \in \mathcal{N}_c\) computes the \(I_c(i)\)-th row of controller \(K\), as a function of private information \(V_i\):

\[ V_i := \lim_{k \to \infty} V_i(k). \quad (4.4) \]

This automatically implies that the update strategy of (i), most converge.

(iii) Asymptotically, the overall closed loop system is \(\gamma\) stable, i.e., \(\rho(A + BK) < \gamma < 1\), where \(K\) is given by (ii).
The non-trivial initialization, in definition 4.2.1, prevents the usage of centralized design strategies. If any initialization could be employed, consider the following assignment:

\[
\forall k \geq 0 : \ V_i(k) = \begin{cases} 
0, & i \in \mathcal{N} \setminus \mathcal{N}_c \\
\hat{k}_T^T(I_c(i)), & i \in \mathcal{N}_c
\end{cases}, \quad \hat{K} = \begin{bmatrix} 
\hat{k}_1^T \\
\vdots \\
\hat{k}_m^T
\end{bmatrix},
\tag{4.5}
\]

where \( \hat{K} \) denotes a \( \gamma \) stabilizing controller that respects constraint (4.2), and was computed using the heuristics of chapter 2. Assignment (4.5) respects all properties of definition 4.2.1; however, this is really not a decentralized design strategy since the controller was actually computed by a centralized strategy. Non-trivial initializations exclude these degenerate cases. In order to stabilize system \((A, B)\), we first run the decentralized design strategy \(D\) until a valid controller \(K\) is found. Afterwards, each control node \(i \in \mathcal{N}_c\) can implement the found controller component, i.e., the \(I_c(i)\)-th row of vector \(K\). Note that this operation does not require any central entity, i.e., each control node can perform an independent implementation. Regardless of the initial condition \(x(0)\), the control law:

\[
\begin{align*}
\ u_{I_c(i)}(k) &= \sum_{j \in \mathcal{N}_i} x_j(k)K_{I_c(i),j}
\end{align*}
\tag{4.6}
\]

is guaranteed to stabilize the closed loop system, \(A + BK\), provided that \(\rho(A + BK) < 1\) (theorem 2.2.2).

Hence, to stabilize a dynamical system one encounters two types of delay: (1) the delay associated with the decentralized computation of \(K\) and (2) the delay associated with the natural decay of state variables, once controller \(K\) is applied. A schematic representation is presented in figure 4.1, where control nodes are coloured cyan and \(I_c(1) = 1, \ I_c(6) = 2\).

![Decentralized design of decentralized controllers](image.png)

Figure 4.1: Decentralized design of decentralized controllers. \(d_1/d_2\) are the mentioned delay types.
4.3 Closed Loop Positive Systems

In chapter 2 we addressed the general problem of stabilizing a system \((A, B)\) under an arbitrary sparsity pattern \(K_I\), such that \((A, B)\) forms a stabilizable pair. As already mentioned, all previous ideas are centralized design strategies. In this section, we present a decentralized design algorithm for a specific class of systems: closed loop positive systems. A closed loop system, \(A + BK\), is called positive if \((A + BK) \in \mathbb{R}^n_{+}\times\mathbb{R}^n_{+}\), i.e., each entry of matrix \(A + BK\) is non-negative. The general interpretation of this condition is that the dynamics of the closed loop system are invariant to the nonnegative orthant, i.e.,

\[
\begin{align*}
    x(0) &\in \mathbb{R}^n_{+} \Rightarrow x(k) = (A + BK)^k x(0) \in \mathbb{R}^n_{+}, \forall \ k \geq 0.
\end{align*}
\]  

(4.7)

Note that the state space matrices, \((A, B)\), are not required to be nonnegative. If they are, the state is naturally nonnegative, hence the controller is designed such that this property is maintained. Example: when the state correspond to age, height, number of elements, Kelvin scale temperature, etc. If the pair \((A, B)\) has no definite sign, the interpretation is that we are enforcing nonnegativeness. Example: the state corresponds to the location of agents that can move freely in space, but we want to restrict movement to the nonnegative orthant.

We have chosen to restrict ourselves to this class of system since, as mentioned in section 2.4.1, the problem of synthesizing sparse stabilizing controllers actually becomes convex, when one restricts the closed loop system to be positive. Actually, one can formulate this problem as a linear or semidefinite program. In general, we would discard the SDP formulation and focus on the simpler linear problem. However, as will be explained later, the structure of the SDP allows to derive a decentralized design strategy that respects definition 4.2.1. The next theorem summarizes the centralized results of designing sparse stabilizing controllers, for closed loop positive systems, with a cone program. For completeness, we also consider additional linear constraints on controller \(K\):

\[
L_1 K \geq L_2
\]

(4.8)

where \((L_1, L_2) \in \mathbb{R}^{l \times m} \times \mathbb{R}^{l \times n}\). This extra constraints can be useful since, for example, by imposing

\[
K \geq 0 \lor K \leq 0
\]

(4.9)

one can achieve nonnegative / nonpositive control laws \(u(k) = Kx(k)\). This can be meaningful if the control signal also has a clear interpretation. For example: given \(m\) drugs and \(n\) patients, \(u(k)\) could represent the quantity, of each drug, to administrate to a particular patient in order to cure some disease.

**Theorem 4.3.1.** For any stabilizable pair \((A, B) \in \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times m}\) and \(\gamma > 0\) there exists a controller \(K \in K_I\) such that \(\rho(A + BK) < \gamma\), \(A + BK \succeq 0\), \(L_1 K \geq L_2\) iff there exists exists a diagonal matrix \(Q > 0\)
and $G \in K_I$ such that:

$$
\begin{bmatrix}
\gamma^2 Q & (AQ + BG)^T \\
AQ + BG & Q
\end{bmatrix} 
\succ 0
$$

(4.10)

$$
AQ + BG \geq 0, \quad L_1 G \geq L_2 Q
$$

where controller $K \in K_I$ is given by $K = GQ^{-1}$.

**Proof.**

1) From [30] (Proposition 2) there exists a controller $K \in K_I$ with $\rho(A + BK) < \gamma$, $A + BK \geq 0$, $L_1 K \geq L_2$ if there exists a $P \succ 0$ and $K \in K_I$ such that:

$$
P = \begin{bmatrix}
p_1 \\
\vdots \\
p_n
\end{bmatrix}, \quad
\begin{bmatrix}
\gamma^2 P & (A + BK)^T \\
A + BK & P^{-1}
\end{bmatrix} \succ 0
$$

(4.11)

$$
A + BK \geq 0, \quad L_1 K \geq L_2.
$$

2) Since $P \succ 0$ is a diagonal matrix we get that each diagonal entry $p_i$ is positive, hence all linear scalar inequalities can be right multiplied by $P^{-1}$:

$$
A + BK \geq 0 \iff AP^{-1} + BK P^{-1} \geq 0
$$

(4.12)

$$
L_1 K \geq L_2 \iff L_1 KP^{-1} \geq L_2 P^{-1}.
$$

3) From step 3) of corollary 2.2.3.1.

$$
\begin{bmatrix}
\gamma^2 P & (A + BK)^T \\
A + BK & P^{-1}
\end{bmatrix} \succ 0 \iff \begin{bmatrix}
\gamma^2 P^{-1} & (A + BK)^T P^{-1} \\
(A + BK)P^{-1} & P^{-1}
\end{bmatrix} \succ 0,
$$

4) Defining $Q = P^{-1}$ and $G = KQ$ we get the desired convex program, where this change of variables is valid since $P \succ 0$ is invertible.

Theorem 4.3.1 is of great importance since it gives convex, necessary and sufficient conditions, for the synthesis problem if one restricts the closed loop system to be positive. This automatically implies that a bisection approach can be considered to generate the optimal controller, in a spectral radius sense. More rigorously, consider algorithm 1 and substitute the feasibility problem of lines 10/18 by the convex program of theorem 4.3.1. This adapted bisection method will solve the following problem:

$$
\begin{align}
\text{minimize} & \quad \rho(A + BK) \\
\text{subject to} & \quad K \in K_I, \quad A + BK \geq 0, \quad L_1 K \geq L_2
\end{align}
$$

(4.13)
given any finite precision $\delta > 0$. The novelty of this chapter consists in deriving a decentralized synthesis strategy, using the SDP of theorem 4.3.1 for a fixed $\gamma$. We denote our approach as DPDSC - Decentralized Primal Dual Synthesizer with Consensus.

### 4.3.1 Decentralized Primal Dual Synthesizer with Consensus

In the context of decentralized design strategies, the sparsity pattern of theorem 4.3.1 is interpreted as follows:

$$\forall i \in \mathcal{N}_c, \ k_i \in \mathcal{S}_i := \{x \in \mathbb{R}^n : x_j = 0, \ for \ j \notin \mathcal{N}_i\} \tag{4.14}$$

where $k_i$ denotes the $i$-th column of controller $K$ transposed, i.e., $K^T = [k_1 \ldots k_m]$. Assuming the following decomposition of matrices $(B, L_1)$,

$$B = [b_1 \ldots b_m], \ L_1 = [l_1^1 \ldots l_m^1] \tag{4.15}$$

the next feasibility problem is equivalent to the LMI condition given in theorem 4.3.1:

$$\exists \left\{Q_i, g_i, \Lambda_i, \Sigma_i, \Gamma_i \right\}_{i=1}^m : \begin{array}{l}
Q_i > 0 \ diagonal, \ g_i \in \mathcal{S}_i, \ A_i \in \mathbb{R}^{n \times n}, \ \Sigma_i \in \mathbb{S}^{2n} \\
AQ_i + b_i g_i^T \geq \Lambda_i, \ \left[ \begin{array}{c}
\gamma Q_i \ \ \ \ \ \ \ u^T \\
Q_i \left[ AQ_i + b_i g_i^T \right] \\
\end{array} \right] \geq \Sigma_i, \ \ u^T g_i^T - L_2 Q_i \geq \Gamma_i \\
m \sum_{i=1}^m \Lambda_i \geq 0, \ \ m \sum_{i=1}^m \Gamma_i \geq 0, \ \ m \sum_{i=1}^m \Sigma_i \succ 0
\end{array} \tag{4.16}$$

To prove necessity assume that $(\hat{Q}, \hat{G})$ is feasible, according to theorem 4.3.1, and decompose $\hat{G}$ as $\hat{G}^T = [\hat{g}_1, \ldots, \hat{g}_m]$. Considering the following assignment of variables:

$$Q_i = \frac{\hat{Q}}{m}, \ g_i = \frac{\hat{g}_i}{m}, \ A_i = AQ_i + b_i g_i^T, \ \Lambda_i = \left[ \begin{array}{c}
\gamma Q_i \ \ \ \ \ \ \ u^T \\
Q_i \left[ AQ_i + b_i g_i^T \right] \\
\end{array} \right], \ \Gamma_i = \frac{1}{m} \sum_{i=1}^m \left( \frac{\hat{g}_i}{m} \right)^T - L_2 Q_i \tag{4.17}$$

we clearly see that problem (4.16) is also feasible. Sufficiency holds trivial by defining $\hat{Q} = \sum_{i=1}^m Q_i > 0$ and $\hat{G} = G$. Reformulation (4.16) simply uses a rank one sum decomposition of the products, $BG$ and $L_1 G$, together with the fact that positive definitiveness is preserved through sums. Note that problem (4.16) is homogeneous in the optimization variables, i.e., each term is coupled to an optimization variable. This automatically implies that all solutions can be scaled by any positive factor. More formally, given any $\alpha > 0$:

$$\left\{Q_i, g_i, \Lambda_i, \Sigma_i, \Gamma_i \right\}_{i=1}^m \text{ solves (4.16) } \iff \left\{\alpha Q_i, \alpha g_i, \alpha \Lambda_i, \alpha \Sigma_i, \alpha \Gamma_i \right\}_{i=1}^m \text{ solves (4.16).} \tag{4.18}$$

Given any $\epsilon_Q > 0$ and $\epsilon_\Sigma > 0$ consider the following positive $\alpha$:

$$\alpha = \max \left\{ \frac{\epsilon_Q}{\lambda_{\min}(Q_1)}, \frac{\epsilon_Q}{\lambda_{\min}(Q_2)}, \ldots, \frac{\epsilon_Q}{\lambda_{\min}(Q_m)}, \frac{\epsilon_\Sigma}{\lambda_{\min}(\sum_{i=1}^m \Sigma_i)} \right\} \tag{4.19}$$
and conclude that problem (4.16) is equivalent to

$$\exists \left\{ Q_i, g_i, \Lambda_i, \Sigma_i, \Gamma_i \right\}_{i=1}^m : Q_i \succeq \epsilon Q I_n \text{ diagonal, } g_i \in S^i, \Lambda_i \in \mathbb{R}^{n \times n}, \Sigma_i \in \mathbb{S}^{2n}$$

(4.20)

$$AQ_i + b_i g_i^T \succeq \Lambda_i, \quad \begin{bmatrix} \gamma^2 Q_i \\ AQ_i + b_i g_i^T \end{bmatrix} \succeq \Sigma_i, \quad l_i^T g_i - L_2 Q_i \geq \Gamma_i$$

$$\sum_{i=1}^m \Lambda_i \geq 0, \quad \sum_{i=1}^m \Gamma_i \geq 0, \quad \sum_{i=1}^m \Sigma_i \succeq \epsilon I_{2n}.$$  

Formulation (4.20) might seem unnecessary since the only difference is that the strict inequalities were replaced by non-strict ones. This transformation is important for two reasons. First, as highlighted in section 2.5, strict inequalities are not suitable for the underlying software and should be replaced whenever possible. But, more importantly, this step is of great importance for the theoretical analysis carried out in appendix C.1.

Note that problem (4.20) has a very particular structure: we have $m$ decoupled feasibility problems and three coupling inequalities that link the variables $\left\{ \Lambda_i, \Sigma_i, \Gamma_i \right\}_{i=1}^m$. This type of structure can be used to derive a decentralized synthesis algorithm, i.e., a strategy that respects definition 4.2.1. For simplicity, we first present the algorithm and convergence is proved in appendix C.1. The theoretical results mostly come from a recent paper of Aybat and Hamedani [50], where a decentralized algorithm is derived for certain decoupled optimization problems with coupling conic constraints. The major insight in [50] comes from a Primal Dual algorithm proposed in [51] and the application of such method to a distributed setting.

Our decentralized design strategy constitutes a trivial extension of the work of Aybat and Hamedani [50], in the context of designing sparse stabilizing controllers for closed loop positive systems. Consult appendix C.1 for further insight. Define the following operators that given an input vector $x$ or matrices $\{ X_i \}$, construct a diagonal or block diagonal matrix:

$$\forall \ x \in \mathbb{R}^n : \ diag(x) = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n \times n} \quad (4.21)$$

$$\forall \ \left\{ X_i \right\}_{i=1}^n : \ X_i \in \mathbb{R}^{n_i \times n_i}, \ \sum_{i=1}^n n_i = n : \ \text{Diag}(X_1, \ldots, X_n) = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} \in \mathbb{R}^{n \times n}. \quad (4.22)$$

We denote our approach as $\mathcal{DPDS}$ - Decentralized Primal Dual Synthesizer:
Algorithm 8 Decentralized Primal Dual Synthesizer - DPDS

(i) Given some stabilizable pair \((A, B)\), connected graph \(G\) and bound \(\gamma > 0\) define an arbitrary scalar \(\beta > 0\). Node \(i \in \mathcal{N}\) is initialized with:

\[
\begin{align*}
    &c_i \in \mathbb{R}_{++}, \\
    &\tau_i = 1/c_i, \\
    &\kappa_i = \frac{c_i}{2c_i\beta|\mathcal{N}_i| + 1}, \\
    &Y_i(0) = 0_{(2n+n^2+n)\times(2n+n^2+n)}, \\
    &S_i(0) = 2Y_i(0), \\
    &\mathbf{Q}_i(0) \succeq \epsilon Q I_n \text{ arbitrary diagonal}, \\
    &\mathbf{g}_i(0) \in \mathcal{S}_i \text{ arbitrary}, \\
    &\Lambda_i(0) = A\mathbf{Q}_i(0) + b\mathbf{g}_i(0)^T, \\
    &\Sigma_i(0) = \begin{bmatrix} \gamma^2 Q_i(0) & *^T \\ A\mathbf{Q}_i & \Lambda_i \end{bmatrix}, \\
    &\Gamma_i(0) = l_1^i g_i(0)^T - L_2 \mathbf{Q}_i(0)
\end{align*}
\]

where \(|\mathcal{N}_i|\) denotes the number of neighbours of node \(i \in \mathcal{N}\). Further define \(N_{\text{DPDS}}\) as the maximum number of iterations.

(ii) Update the primal variables, \(\{\mathbf{Q}_i(k+1), \Lambda_i(k+1), \Sigma_i(k+1), \Gamma_i(k+1), \mathbf{g}_i(k+1)\}\) of each control node. For each \(i \in \mathcal{N}_c\):

\[
\begin{align*}
    &\text{minimize}_{\mathbf{Q}_i, \Lambda_i, \Sigma_i, \mathbf{g}_i} \; f(Q_i, \Lambda_i, \Sigma_i, \Gamma_i, \mathbf{g}_i) \\
    &\text{subject to} \\
    &\gamma^2 Q_i + *^T \succeq \Sigma_i \\
    &g_i \in \mathcal{S}_i, \; Q_i \succeq \epsilon Q I_n \text{ diagonal} \\
    &AQ_i + b\mathbf{g}_i^T \succeq \Lambda_i, \; l_1^i g_i^T - L_2 \mathbf{Q}_i \succeq \Gamma_i
\end{align*}
\]

where \(f\) is a quadratic function defined as:

\[
\begin{align*}
    f(Q_i, \Lambda_i, \Sigma_i, g_i) = & \text{trace} \left( Y_i(k)^T \text{Diag}\left( \text{vec}(\Lambda_i), \text{vec}(\Gamma_i), \Sigma_i - \frac{\epsilon \Sigma}{n} I_{2n} \right) \right) \\
    &+ \frac{1}{2\tau_i} \left\{ ||Q_i - Q_i(k)||_F^2 + ||\Lambda_i - \Lambda_i(k)||_F^2 + ||\Sigma_i - \Sigma_i(k)||_F^2 \\
    &\quad + ||g_i - g_i(k)||_F^2 + ||\Gamma_i - \Gamma_i(k)||_F^2 \right\}
\end{align*}
\]
(iii) Augment the data set of each node, adding/updating the following mismatch variable:
\[
\forall \ i \in \mathcal{N} : \ E_i(k+1) = \sum_{j \in \mathcal{N}_i} \left( S_i(k) - S_j(k) \right).
\] (4.26)

(iv) Update the dual variable, \( \left\{ Y_i(k) \right\}_{i=1}^{n} \) of each node. For each \( i \in \mathcal{N} \):
\[
\begin{align*}
\minimize_{Y_i} & \quad g(Y_i) \\
\text{subject to} & \quad Y_i \preceq 0
\end{align*}
\] (4.27)

where \( g \) is a quadratic function defined as:
\[
g(Y_i) = \beta \text{trace} \left( Y_i(k)^T E_i(k+1) \right) + \frac{1}{2n_i} \| Y_i - Y_i(k) \|^2_F \\
- \text{trace} \left( Y_i^T \text{Diag} \left\{ \text{vec} \left\{ 2 \Lambda_i (k+1) - \Lambda_i(k) \right\} \right\} \right), \\
- \text{trace} \left( Y_i^T \text{Diag} \left\{ \text{vec} \left\{ 2 \Gamma_i (k+1) - \Gamma_i(k) \right\} \right\} \right), \\
2 \Sigma_i (k+1) - \Sigma_i(k) - \frac{\epsilon \Sigma}{n} I_{2n}.
\] (4.28)

(v) Update the variable the sum variable \( S_i(k) \):
\[
\forall \ i \in \mathcal{N} : \ S_i(k+1) = Y_i(k+1) + \sum_{l=0}^{k+1} Y_i(l).
\] (4.29)

(vi) Repeat until \( k = N_{DPDS} \).

Note that steps (ii) and (iii) can be processed in parallel since they depend on past data, i.e., variables from iteration \( k \). This is also true for (iii) and (iv): once node \( i \) computes \( Y_i \), update (4.29) can be executed. Instead of imposing a maximum number of iterations one can consider some type of distributed termination criteria. Regarding the control computation, for \( i,j \in \mathcal{N}_c \) with \( i \neq j \), control agent \( i \) does not needs to know how \( u_j(k) \) will affect the global dynamics. More rigorously, control agent \( i \in \mathcal{N}_c \) only needs to know: (1) its neighbouring set \( \mathcal{N}_i \), (2) the \( I_{c}(i) \)-th row of matrices \( (B, L_1) \) and (3) the entire matrices \( (A, L_2) \). Non control nodes only need information regarding \( \mathcal{N}_i \) since their single purpose is to update and exchange dual information i.e., variables \( Y_i(k) \) and \( S_i(k) \).

In simple terms, matrices \( (B, L_1) \) are distributed, \( (A, L_2) \) must be globally known and non control nodes do not compute any control component. The next theorem summarizes the convergence results of \( DPDS \):

**Theorem 4.3.2.** For any stabilizable pair \( (A, B) \in \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times m} \), bound \( \gamma > 0 \), connected graph \( \mathcal{G} \) and positive scalars \((\epsilon_P, \epsilon_\Sigma)\) assume problem (4.20) is feasible. As \( N_{DPDS} \to +\infty \) algorithm 8 converges
to a controller $K^*$ such that $k_i^* \in S_i$, $\rho(A + BK^*) < \gamma$, $A + BK \geq 0$ and $L_1 K^* \geq L_2$. Controller $K^T = \left( k_i^* \right)_{j=1}^m$ is given by:

$$\forall i \in \mathcal{N}, \ j \in \{1, \ldots, m\} : L_i(i) = j \Rightarrow k_j^T = g_i(N_{DPDS})^T \left\{ \sum_{i \in \mathcal{N}_c} Q_i(N_{DPDS}) \right\}^{-1}. \quad (4.30)$$

Details can be found in appendix C.1. Note however, that $DPDS$ does not respect definition 4.2.1 since, from theorem 4.3.2, each control node needs to have access to:

$$\sum_{i \in \mathcal{N}_c} Q_i(N_{DPDS}), \quad (4.31)$$

to compute its corresponding row of controller $K^*$. By iterating algorithm 8, node $i \in \mathcal{N}_c$ will only know $Q_i(N_{DPDS})$. How can sum (4.31) be computed in a decentralized way?

This is, perhaps, the most famous problem in network science: undirected consensus. Given some connected undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, assume each node is initialized with a variable $x_i \in \mathbb{R}$, i.e., $\theta_i(0) = x_i$. There is no central node. The undirected consensus problem consists in designing an algorithm such that:

$$\forall i = 1, \ldots, |\mathcal{N}| : \theta_i(k) \rightarrow \sum_{i \in \mathcal{N}} \frac{x_i}{|\mathcal{N}|}, \quad (4.32)$$

given that nodes $i$ and $j$ can only communicate if $i \in \mathcal{N}_j$ or $j \in \mathcal{N}_i$. A solution for this problem consists in employing the following iterative linear procedure:

$$\theta(k + 1) = W\theta(k), \quad (4.33)$$

where $\theta(k) = \left\{ \theta_i(k) \right\}_i$, and $W$ is an appropriate consensus matrix. In particular, update (4.33) solves the undirected consensus problem if matrix $W$ exhibits the following properties [52]:

$$(i) \quad W = W^T$$

$$(ii) \quad W_{i,j} = 0, \text{ if } i \notin \mathcal{N}_j \text{ or } j \notin \mathcal{N}_i$$

$$(iii) \quad \left\{ \lambda_1(W), \ v_1(W) \right\} = \left\{ 1, \left[ \frac{1}{|\mathcal{N}|} \right] \right\}$$

$$|\lambda_i(W)| < 1, \ i = 2, \ldots, |\mathcal{N}|,$$

where $\lambda_1(W) \leq \lambda_2(W) \leq \cdots \leq \lambda_{|\mathcal{N}|}(W)$ denote the eigenvalues of $W$ and $v_i(W)$ denotes one eigenvector associated with the eigenvalue $\lambda_i(W)$. Note that (ii) ensures that only valid communications can occur.

From this, one can easily understand how to transform $DPDS$ into a truly decentralized design strategy: after algorithm 8 simply perform $n$ parallel consensus rounds to construct the diagonal matrix (4.31). More rigorously, after $N_{DPDS}$ iterations of algorithm 8 each node $i \in \mathcal{N}$ will update $n$ scalar variables.
\[ \{Q_{j,j}(i,k)\}_{j=1}^{n}, \] where \( k \) denotes (discrete) time and \( j \) is indexing the \((j,j)\) entry of matrix (4.31):

\[
i \in \mathcal{N} \setminus \mathcal{N}_c : Q_{j,j}(i,0) = 0, \quad \forall \ j \in \mathcal{N} \tag{4.35}
\]

\[
i \in \mathcal{N}_c : Q_{j,j}(i,0) = n\{Q_i(N_{DPDS})\}_{j,j}, \quad \forall \ j \in \mathcal{N}.
\]

Non-control nodes are initialized with zero while control nodes are initialized with the scaled diagonal entries of matrix \( Q_i(N_{DPDS}) \), i.e., private information. By running \( n \) parallel rounds of (4.33) each node will asymptotically know (4.31); hence, the controller’s rows can be recovered by theorem 4.3.2. Note that the controller is not globally known: every node has access to (4.31), but only control nodes can compute their corresponding row of controller \( K^* \). This can be beneficial, in the context of providing a secure control law. For any \( \gamma < 1 \) the following procedure clearly respects definition 4.2.1:

\begin{algorithm}
\textbf{Algorithm 9} Decentralized Primal Dual Synthesizer with Consensus - DPDSC
\begin{itemize}
\item[(i)] Given some stabilizable pair \((A, B)\), connected graph \( \mathcal{G} \) and bound \( \gamma > 0 \) run algorithm 8 with constants \((\epsilon_Q, \epsilon_\Sigma)\) and for \( N_{DPDS} \) iterations.
\item[(ii)] Run \( n \) parallel consensus rounds, each for \( N_c \) iterations. Using initializations (4.35) perform:
\[
Q_{j,j}(k+1) = WQ_{j,j}(k), \quad Q_{j,j}(k) = \begin{bmatrix} Q_{j,j}(1,k) \\ \vdots \\ Q_{j,j}(n,k) \end{bmatrix} \tag{4.36}
\]
for any \( j \in \mathcal{N}, W \) respecting (4.34) and \( k \leq N_c \).
\end{itemize}
\end{algorithm}

\section{4.4 Numerical Section}

In this section we provide numerical confirmation of the convergence properties, stated in theorem 4.3.2. To this end, consider the following experimental setup:

(1) Given some pair \((n, m)\) we sample state space matrices \((A, B)\) from an uniform distribution in the interval \([a, b]\). The \( m \) control nodes are also chosen at random and matrices \((L_1, L_2)\) are defined latter.

(2) Graph \( \mathcal{G} \) is a connected Erdős-Rényi model. Specifically, given \( n \) edges the probability that any two edges connect is given by \( p_{\text{edge}} \in (0, 1) \). As \( p_{\text{edge}} \) decreases the graph becomes sparser. Non-connected graphs are discarded.

(3) Let \( K_c \) denote a sparse centralized controller computed from theorem 4.3.1 setting \( \gamma = 1 \).

(4) For the same bound, algorithm 9 is deployed for \((N_{DPDS}, N_c)\) iterations, imposing \( \epsilon_\Sigma = 10^{-7} \) and \( \epsilon_p = 1 \). We have selected \( W \) as the fastest Laplacian weight consensus matrix [52] (section 4.2, equation 23).
Figures 4.2 and 4.3 display some results where $K_d$ denotes the final controller produced by $\text{DPDSC}$ and $K_c$ is the centralized version. For the second experiment, note that the pair $(A, B)$ is entry wise non-positive, hence the controller is enforcing positiveness in the closed loop system. For figure 4.2 we simply preserve this property, since matrices $(A, B)$ are already non-negative. Regarding the consensus plots, we have chosen a random diagonal entry of matrix (4.31) to display the convergence properties, i.e., legend $Q_{i,i}$ corresponds to the $(i, i)$ entry of (4.31). For this sequence of figures, we impose an undifferentiated and non-positive control law, i.e.,

$$ (L_1, L_2) = (0_{1 \times n}, 0_{1 \times n}) $$

$$(L_1, L_2) = (-I_m, 0_{m \times n}).$$

In table 4.1 we compare the centralized and decentralized approaches for the generations of figures 4.2 and 4.3. The computational time associated with the decentralized method is given by the sum of two terms: the time of $\text{DPDS}$ plus the time of $n$ consensus rounds, respectively. Regarding figure 4.3, $\text{DPDSC}$ was able to compute a much stable controller, comparing with the centralized version. There is no contradiction since we simply require a stabilizing controller, i.e., the bound $\gamma$ was set to one. One could also use the bisection method, mentioned at the end of section 4.3, to find the solution of problem (4.13), and then employ $\text{DPDSC}$ for some target bound.

We have chosen low dimensional problems such that results can be computed in acceptable time,

Figure 4.2: Achieved results with $(N_{\text{DPDS}}, N_c, a, m, b, p_{\text{edge}}) = (100, 20, 5, 2, 0, 0.5, 0.5)$. We have chosen low dimensional problems such that results can be computed in acceptable time,
since \( \mathcal{DPDS} \) involves solving SDPs of large dimensions, i.e., in the order of \( n^2 \). This is reflected in the last line of table 4.1. In sections 5.1 and 5.3 we discuss possible solutions to address this issue. To finalize this section, note that \( \mathcal{DPDS} \) will guarantee closed loop stability long before ensuring positiveness. This might indicate that this method can be further explored to design decentralized controllers, for general closed loop systems, i.e., adapting \( \mathcal{DPDSC} \) without the positiveness restriction and hence presenting only sufficient conditions for decentralized stability.

<table>
<thead>
<tr>
<th>Controller</th>
<th>( K_c )</th>
<th>( K_d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \rho(A + BK) )</td>
<td>0.9413 → 0.7993</td>
<td>0.9433 → 0.5450</td>
</tr>
<tr>
<td>( \min_{(i,j)} { A + BK }_{i,j} )</td>
<td>0.0258 → 0.0255</td>
<td>0.0072 → 5.12011 ( \times 10^{-4} )</td>
</tr>
<tr>
<td>( \min_{(i,j)} { L_1 K - L_2 }_{i,j} )</td>
<td>0 → 0</td>
<td>0 → 0</td>
</tr>
<tr>
<td>Computational Time [min]</td>
<td>( { 0.2883 → 0.3930 } / 60 )</td>
<td>( 27 + \frac{0.0071}{60} \ → 91.84 + \frac{0.0058}{60} )</td>
</tr>
</tbody>
</table>

Table 4.1: Centralized vs Decentralized controller Synthesizes: data of figures 4.2 and 4.3.

---

**4.5 Application: Load Sharing in a Network of Computers**

In this section, we present a particular application that could benefit from the ideas of section 4.3. Consider a network formed by \( n \) computers, and let \( x_i(k) \in \mathbb{R}_+ \) denote the computational load in computer
At time \( k \). One can assume that (continuous) time is discretized in equal amounts, hence \( x_i(k) \) could represent the average number of processes, of computer \( i \), in the corresponding time interval. Assume the following dynamics, for the open loop system:

\[
x(k + 1) = Ax(k), \quad A = \begin{bmatrix}
1 - \delta_1 \\
\vdots \\
1 - \delta_n
\end{bmatrix}
\]

(4.38)

where \( \delta_i \in [0, 1] \) represents the computational capacity of computer \( i \). In simple terms, computer \( i \) can process \( \delta_i \% \) of the load, in a single time interval. System (4.38) is naturally positive, since \( x_i(k) \) represents the load of computer \( i \). The system is also completely decoupled and stable: computer \( i \) does not influence computer \( j \) and, given enough time, all processes will be executed. Our goal is to decrease this waiting time, by using the underlying network to share some of the computational load. In particular, we want to generate a sharing strategy which improves the standard performance and is computed in a decentralized fashion. More rigorously, system (4.38) is generalized to:

\[
x_i(k + 1) = (1 - \delta_i) \sum_{j \in N_i} \beta_{i,j} x_i(k) + \sum_{j \in N_i} \beta_{j,i} x_j(k)
\]

(4.39)

where \( \beta_{i,j} \in (0, 1) \) represents the load percentage, sent from computer \( i \) to computer \( j \). The actual amount is given by \( \beta_{i,j} x_i(k) \). In (4.39), we are adding/subtracting the flow send to/from node \( i \). Note that \( \{ (1 - \delta_i) - \sum_{j \in N_i} \beta_{i,j} \} \) should always be nonnegative, i.e., computer \( i \) cannot send more processes than those that it owns. Equation (4.39) corresponds to a closed loop system where \( B = I_n, A \) is given by equation (4.38) and the constraints of the problem are concentrated in controller \( K \). In particular controller \( K \) exhibits the following structure:

\[
K_{i,j} = \begin{cases}
- \sum_{j \in N_i} \beta_{i,j} & \text{if } i = j \\
\beta_{j,i} & \text{if } i \neq j \land j \in N_i \\
0 & \text{otherwise}
\end{cases}
\]

with \( \{ (1 - \delta_i) \geq \sum_{j \in N_i} \beta_{i,j} \} \).

(4.40)

This problem can be formulated using the formalism of section 4.3 since (1) we have a closed loop positive system (2) the controller must exhibit a fixed sparsity pattern (3) the controller exhibits linear constraints and (4) we want to design a decentralized strategy. One just has to construct matrices \( L_1, L_2 \) such that constraints (4.40) have the prescribed form, i.e., equation (4.8). The general construction is given by

\[
L_1 = \begin{bmatrix} 1^n \\ -1^n \\ I_n \\ -I_n \end{bmatrix}, \quad L_2 = \begin{bmatrix} 0_{2 \times n} \\ -A \\ D \end{bmatrix}, \quad D_{i,j} = \begin{cases} -1 & \text{if } j \in N_i \\
0 & \text{otherwise}
\end{cases}
\]

(4.41)
Chapter 5

Conclusions, Contributions and Future Work

5.1 Conclusion

Decentralized control is an exciting and emerging topic which combines ideas from linear algebra, optimization, distributed optimization and classical control theory. This dissertation advances this area, by studying three fundamental problems for the distributed control of massive-scale networks.

In chapter 2 we started by providing some background on the centralized design task, and proceeded to develop several methods for controller synthesizes, given a pre-defined sparsity pattern. Our methods are only heuristics since the original problem is \( \mathcal{NP} \) hard; however, we assess their performance through extensive numerical experiments. The decentralized convex concave method and the boosted incremental-PK approach were elected as the most suitable algorithms, achieving acceptable stabilization margins while being computational fast.

Using the ideas from chapter 2, we tackle the budget sparsity design problem in chapter 3. Our main insight was to use perturbation theory arguments, hence escaping the classical \( l_1 \) regularization option, which is not trivially applicable for the problem at hand. Both our oracles were shown to be effective, since they outperformed the randomized design option. Hence, there is a clear benefit in using our informed approaches.

Chapter 4 approaches an even more difficult problem: the decentralized computation of sparse stabilizing controllers. Having rigorously defined our understanding of the problem, we presented a truly decentralized design strategy, for the particular instance of closed loop positive systems. Although the positiveness assumption can be restrictive, \( \text{DPDSC} \) is based on necessary and sufficient conditions; hence, it is highly attractive for realistic applications that demand some kind of performance guarantee.
5.2 Contributions

The main contributions of the thesis are the following:

(1) We exploited some classical results from centralized stabilization theory to create suitable stabilizing heuristics for decentralized controllers. All methods of chapter 2 are fairly flexible, since generic convex constraints can be imposed on controller $K$: the sparsity aspect constitutes only a particular application of the methods presented there.

(2) We numerically compared all the proposed methods and found the most competitive ones.

(3) We provided numerical evidence that the reduction strategy can improve the performance of CCP, in the context of producing sparse stabilizing controllers. A surprising result, since this strategy is fairly simple and CCP is able to perform very well in a variety of hard problems.

(4) Using classical results from perturbation theory, we derived new approaches for controller sparsity design, with budget constraints. To the best of our knowledge, classical $l_1$ regularization cannot be employed for this problem, hence our method can be seen as a novelty in this field since no relatable literature was found.

(5) We provided several extensions of the developed methods, thus motivating further investigation, say to a $H_2$ control context.

(6) We created a decentralized design strategy for closed loop positive systems, subject to arbitrary linear constraints. This strategy is guaranteed to find a suitable controller, given that one exists.

5.3 Future Work

Regarding future work there are several research directions one could take. In fact, we already embarked in some alternatives but, due to space constraints, this work cannot be presented here. Hence, our suggestions can be categorized in two groups: (1) topics for which some preliminary results already exist and (2) completely new work. The next list summarize our vision regarding the future of decentralized control, including ideas from both groups:

(1) Preliminary work

   i) As mentioned in section 1.1 the stabilization problem, alone, has little interest in more realistic control applications. A natural idea is to utilize the foundations of this thesis and approach more exciting control problems like the $H_2$ problem, where one considers stochastic disturbances in the system and the control objective is to minimize the disturbance effect, in an adequate sense. This problem can be formulated as minimizing a linear functional over a non convex set, with a structure similar to that of theorem 2.2.3. This similarity can be explored to adapt some of the developed heuristics. An adaptation of the decentralized convex concave method presented encouraging results, when compared with other approaches in the literature.
ii) Regarding appendix B.1.2 the idea of discriminative oracles can be used to assign control components to state variables, in graphs. Considering the formalism of chapter 4.4, we assume that the sparsity pattern of controller $K$ is given. We know where all control components should be placed, i.e., in which node of graph $\mathcal{G}$. But what if this assignment is not defined a priori? If we can choose, what assignment should we do? Discriminative oracles allow to select an assignment that promotes the stability of the closed loop system. In simple terms, growing stable control laws.

iii) In section 4.3 we have restricted ourselves to closed loop positive systems and derived a decentralized control law that is able to compute a stabilizing controller in a fully distributed sense. As previously mentioned, one drawback of this approach is the high computational load that each node must endure. A naturally question is how to alleviate this burden, in order to promote scalability. As mentioned in 4.3, the same control problem can be formulated as a linear program. Computationally, a linear program can be solved much more efficiently than a SDP. This motivates the question: can we use the presented ideas and migrate them to the linear case? Without going into too much detail, the short answer is yes if one adds some extra constrains. Our early analysis indicates that, if matrix $A$ is non-negative, then the linear formulation allows to build an equivalent of DPDS where each control node would compute the corresponding column of controller $K$. This is not a decentralized control strategy since, to compute $u_i(k)$, control node $i \in \mathcal{N}_c$ needs to have access to the $i$-th row of controller $K$. However, one can combine the linear formulation with consensus ideas to, again, produce a decentralized design strategy. In particular the necessary number of consensus rounds would be equal to

$$\sum_{i \in \mathcal{N}_c} |\mathcal{N}_i|.$$  \hspace{1cm} (5.1)

An interesting research direction would compare the efficiency of both approaches: DPDS\textsuperscript{C} vs a linear version of DPDS with (5.1) consensus rounds. The second option can potentially lead to efficient implementations being able to scale with larger systems. Perhaps, the non-negativeness assumption on $A$ can be further relaxed.

(2) Completely new work

i) Produce decentralized design strategies for general closed loop systems, i.e., trying to migrate the ideas of chapter 2 to a fully decentralized setting. As mentioned in section 4.4, a promising idea would be to adapt DPDS by eliminating the positiveness restriction and hence losing necessity, for decentralized stability. This direction is very appealing, since DPDS can achieve closed loop stability very quickly, i.e., in few iterations.

ii) Improve scalability of all the developed heuristics. This is a very ambitious claim since the scalability issues follow directly from the general computational load required to solve an SDP, of moderate dimensions.
iii) Further explore the class of positive systems. When dealing with positive matrices, Perron Frobenius Theory allows to derive several very useful results. Maybe $DPDSC$ can be further extended or adapted to cope with more realistic closed loop positive systems.

iv) As with any distributed algorithm, generalize $DPDSC$ to a more realistic setting. For example considering time varying networks where edges can appear or disappear, at random. This is a challenging problem since one needs to redefine the control objective. For a fixed network, the sparsity of controller $K$ is dictated by the structure of graph $\mathcal{G}$. If $\mathcal{G}$ varies in time how do we define a stabilizing controller? What sparsity should it exhibit? Should it also change over time?
Bibliography


Appendix A

Extensions for chapter 2

A.1 Output Feedback Dynamic Controllers

Going back to section 2.2, we started by introducing the most general linear model for the plant and the controller, equations (2.1) and (2.2), and then proceed to discuss a variety of ideas for a restricted framework: considering static controllers and state feedback control laws. All those decentralized algorithms can be extended to cover the most general setup of the stabilization problem. In this setup the control component $u_j(k)$ is computed using some measurements $y_i(k)$ for $I(i,j) = 1$, i.e., the indicator function should be redefined in terms of measurements $y(k) \in \mathbb{R}^m$ and not state variables $x(k) \in \mathbb{R}^n$. This is a more realistic assumption, since in most applications we generally do not have access to state variables, only to some measurements of the underlying process.

No algorithm depends on any change of variables directly involving the controller $K \in \mathcal{K}_I$. This has one main consequence: all heuristics can be used to impose any convex constraint on the controller $K$, i.e., imposing that $K \in \mathcal{K}_I$ is only a restricted use of the algorithms. They can be used to solve problem (2.10) under any convex set, for example imposing that the controller is a symmetric matrix [12],[13]. This type of structure appears in vehicle formation applications [53] where, given a fictitious leader and follower, there are $N$ vehicles in a straight line formation that must be kept at some positions assuming a desired grid of regularly spaced points moving with a constant velocity. In absence of disturbances, this problem can be formulated as a stabilization problem where each vehicle must compute a control signal from local measurements of the relative distances with respect to their immediate neighbours, i.e., each control component is computed from a forward and backward gain. Under some assumptions on the dynamics of the vehicles, if the forward and backward gains are equal, the stabilization problem boils down to finding a specific structured symmetric controller that ensures closed loop stability.

From equation (2.3) we get that problem (2.6), in the general case, boils down to minimizing the spectral
radius of the following matrix:
\[
\begin{bmatrix}
A + BD_KC & BC_K \\
B_KC & A_K
\end{bmatrix}
\]

where \((A_K, B_K, C_K, D_K)\) are the optimization variables and \((A, B, C)\) are constant state space matrices. The decentralized case will minimize the spectral radius of the same matrix, but now there exist structural constraints associated with each matrix variable, i.e., \(A_K \in A_I, B_K \in B_I, C_K \in C_I\) and \(D_K \in D_I\). Each matrix variable exhibits its own indicator function \(I(.)\) that simply insures that the control signal \(u(k)\) is computed using available information only: measurements and control state variables. This was our main critic for the assumptions of Alavian and Rotkowitz in [10]. The authors assume that a decentralized controller can, potentially, utilize all state variables \(x_K(k)\), while processing only available measurements, i.e., only a decentralization of measurements is assumed.

To better grasp this concept, consider the state feedback example of figure 2.5 but now assume a full order LTI controller described by equation (2.2). Under this framework we have that \(n_k = n = p = 9\) and \(m = 3\). Since the controller is now dynamic, let us assume that each node \(x_i\) will also have access to the \(i\)-th component of \(x_K\), i.e., the state component associated with the dynamics of the controller. Under these assumptions the matrices \((A_K, B_K, C_K, D_K)\) will have the following structure:

\[
A_K, C_K = \begin{bmatrix}
* & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 \\
0 & * & * & 0 & * & 0 & 0 & 0 & 0 \\
0 & * & * & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & * & * & 0 & 0 & 0 & 0 \\
* & * & 0 & * & * & * & * & * & 0 \\
0 & 0 & 0 & 0 & * & * & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & * & 0 & 0 & * \\
0 & 0 & 0 & 0 & * & 0 & 0 & * & * \\
0 & 0 & 0 & 0 & 0 & 0 & * & * & * \\
\end{bmatrix}
\]

\[
B_K, D_K = \begin{bmatrix}
* & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 \\
0 & * & * & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & * & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & * & 0 & * \\
\end{bmatrix}
\]

For example, to compute \(u_1(k)\) agent \(x_1\) can only communicate with agent \(x_5\); this explains the zeros in the first row of (A.3). The structure of \(B_K, D_K\) is exactly the one observed in figure 2.5 for the static controller \(K\). Constraints (A.2) follows easily from the edges of figure 2.5. Note that the sets \(A_K/C_K\) and \(B_K/D_K\) exhibit the same structural constraints since we are assuming a state feedback example and each node has access to both state variables: of the underlying process and of the controller. The indicator functions \(I(.)\) follows directly for the sets \(A_I = C_I\) and \(B_I = D_I\).

Note that \(A_I \subseteq \mathbb{R}^{n_k \times n_k}\) where the order of the controller \(n_k\) can be less than the order of the sys-
item $n$. The provided heuristics can be employed for any $n_k$, i.e., they can be used to find low order stabilizing controllers. This can be very beneficial if we remember that the controller will be a physical device embedded with some computational capacities. Low order controllers imply low computational demands hence saving resources.

To illustrate how the previous ideas can be extended to cover a more general formalism, let us define the matrix of equation (A.1) as $\Omega_K(A_K, B_K, C_K, D_K)$. Note that $\Omega_K$ depends affinely on the matrix parameters $(A_K, B_K, C_K, D_K) \in A_I \times B_I \times C_I \times D_I$. Using theorem 2.2.3, there exist matrices $A_K \in A_I$, $B_K \in B_I$, $C_K \in C_I$ and $D_K \in D_I$ such that $\rho(\Omega_K(A_K, B_K, C_K, D_K)) < \gamma$ iff

$$\exists \ P \in S_{++}^{n+n_k}, \ A_K \in A_I, \ B_K \in B_I, \ C_K \in C_I, \ D_K \in D_I : \begin{bmatrix} \gamma^2 P & \Omega^T_K \\ \Omega_K & Q \end{bmatrix} \succ 0$$

(A.4)

where $\Omega_K(A_K, B_K, C_K, D_K)$ is given by equation (A.1). From formulation (A.4) it is trivial to generalize all previous heuristics since $\Omega_k$ will play the role of the previous closed loop matrix $A + BK$. Namely in algorithm 2 the linear matrix inequality (2.25) generalizes to:

minimize $\text{trace}(P_k \Delta Q + Q_k \Delta P + P_k Q_k)$

subject to

$$\begin{bmatrix} \gamma^2 P & \Omega^T_K \\ \Omega_K & Q \end{bmatrix} \succ 0, \ \Omega_K = \begin{bmatrix} A + BD_K C & BC_K \\ B_K C & A_K \end{bmatrix}$$

(A.5)

where the updates variables $P_k, Q_k$ come from the previous iteration or from the initialization $P_0, Q_0$. In algorithm 3 the feasibility problem (2.39) is generalized to the following LMI:

$$\Omega^T_K P^{-1} \Omega_K \prec \gamma^2_k \{P^{-1}_k - P^{-1}_k (P - P_k) P^{-1}_k\}$$

(A.6)

where $\Omega^T_K P^{-1} \Omega_K$ is matrix convex since it is the composition of an affine map $\Omega_K(A_K, B_K, C_K, D_K)$ with a convex function $\Sigma^T P^{-1} \Sigma$ where $P^{-1} \succ 0$, theorem 2.4.4. Again the optimization variables in (A.6) will be $P \succ 0$, $A_K \in A_I$, $B_K \in B_I$, $C_K \in C_I$, $D_K \in D_I$ and $\Omega_K$ is given by equation (A.1).

To extend the incremental PK algorithms we define $\Omega_K = \Theta_C + \Theta_L \Psi_K \Theta_R$ where $\Theta_C \in \mathbb{R}^{(n+n_k) \times (n+n_k)}$ denotes the constant term in $\Omega_K$, $\Psi_K \in \mathbb{R}^{(m+n_k) \times (p+n_k)}$ encapsulates the optimization variables and $\Theta_L \in \mathbb{R}^{(n+n_k) \times (m+n_k)}$, $\Theta_R \in \mathbb{R}^{(p+n_k) \times (n+n_k)}$ correspond to constant matrices that multiply $\Psi_K$ from the left and right, respectively. More rigorously, the matrix $\Omega_K \in \mathbb{R}^{(n+n_k) \times (n+n_k)}$ decomposes as fol-
The new terms should be interpreted as the matrix $P_\Theta$ that resulted from problem (A.8) in iteration $k$. Remember that $P_\Theta$ collects the matrices that define the dynamic controller. Using exactly the same type of arguments one can easily generalize the feasibility problem of algorithm 5 by simply introducing the new terms $Q_L$, $\Delta Q$ in the adequate positions. We do not present the explicit LMI, since it can be directly obtained from (A.8).
Appendix B

Extensions for chapter 3

B.1 Extensions

B.1.1 Output Feedback Oracles

In section 3.2, we have assumed static state feedback controllers for the problem of generating respectable control laws, with respect to budget $\Phi$, that insure closed loop stability. All previous oracles can be generalized. Consider a static output feedback control law:

$$ u(k) = Ky(k) $$

(B.1)

where $y(k) \in \mathbb{R}^p$ denotes the output signal. Note that $K$ is still a static controller, hence the budget interpretation remains unchanged: $\phi_i \in \{0, \ldots, p\}$ models the number of output measurements available to compute $u_i \in \mathbb{R}$. In this setup, problem (3.4) is trivially generalized to:

$$ \min_{K \in \mathcal{K}_{\Phi}} \rho(A + BKC) $$

(B.2)

where $C \in \mathbb{R}^{p \times n}$ denotes the usual state space matrix. The set $\mathcal{K}_{\Phi}$ is still given by equation (3.3). In section A.1 all heuristics, for decentralized stability, were generalized to cover the output feedback dynamic case. This has one major implication: when faced with problem (B.2), all ideas of chapter 3.5 are still valid heuristics. This should be immediate: in section 3.5 we developed sparsity oracles combining the heuristics of chapter 2, with standard results from perturbation theory. If the heuristics can be generalized and the oracles depend on the heuristics, a trivial generalization is due for the methods of section 3.5. For algorithms 6 and 7, one just has to re-interpreted the incremental inverse algorithm call: instead of considering the state feedback approach of section 2.4.3 assume the corresponding general version of section A.1.
B.1.2 Discriminative Oracles

When budget $\Phi$ was defined, an implicit assumption was made: output measurements are not discriminated, i.e., $u_i$ will be computed using, at most, any $\phi_i$ measurements. This will constrained the number of usable measurements. Given $\phi_i$, what happens if some measurements cannot be used for the computation of $u_i$? Are the developed oracles still valid? This question is relevant, since some measurements might simply be unreachable. Assume that state variables have different physical locations and measurements are sent through wireless links. It is not possible to transmit wireless information between far away physical locations, so $u_i$ can only be computed from some output measurements, i.e., it cannot depend on unreachable information.

Define $S_i$ as the index set of non available measurements for computing component $u_i$, i.e., for any $l \in \{1, \ldots, p\}$:

$$l \in S_i \iff I(l, i) = 0$$

where $I$ denotes the indicator function defined in (2.8). Equivalence (B.3) simply states that one cannot utilize $y_l$ to compute $u_i$. Under this setup, and after section B.1.1, problem (3.4) is transformed into:

$$\begin{align*}
\min_{K} & \quad \rho(A + BKC) \\
\text{subject to} & \quad K_{l,i} = 0 \quad \text{for } l \in S_i \\
& \quad K \in K_\Phi
\end{align*}$$

(B.4)

where constraints (B.3) were added to problem (B.2). All ideas of section 3.5 can still be applied to design sparsity oracles for problem (B.4). In simple terms, if $l \in S_i$ then constraint $K_{l,i} = 0$ cannot be relaxed. This constraint is easily reflected in algorithms 6 and 7: when evaluating any dual information, we just discard non available information, i.e., if $l \in S_i$ then the dual variable associated with $K_{l,i} = 0$ cannot be utilized since this constraint can never be eliminated. All ideas remain unchanged added that only available dual information is used to infer on which constraints, $K_{i,j} = 0$, should be relaxed/eliminated. Basically, the sparsity search space $K_\Phi$ is reduced by introducing the constraint of available information, but all previous ideas are still valid.
Appendix C

Proof of convergence for DPDS

C.1 Convergence

This section exists to make some theoretical claims regarding algorithm 8, namely to prove theorem 4.3.2. We start by noticing that problem (4.20) is equivalent to:

\[
\begin{align*}
\text{minimize} & \quad \xi \sum_{i \in \mathcal{N}} I_{\Phi_i}(\xi_i) \\
\text{s.t} & \quad i \in \mathcal{N}_c : \Phi_i = \left\{ \xi_i : \xi_i = \begin{bmatrix}
\text{diag}\{\text{vec}(A_i)\} & 0_{n^2 \times n} & 0_{n^2 \times 2n} \\
0_{n \times n^2} & \text{diag}\{\text{vec}(\Gamma_i)\} & 0_{n \times 2n} \\
g_i^T & 0_{1 \times n^2 - n} & \begin{bmatrix}
\Sigma_i \\
Q_i & 0_{n \times n^2 - n} & 0_{n \times 2n}
\end{bmatrix}
\end{bmatrix}, Q_i \succeq \epsilon Q \right\} \quad (C.2)
\end{align*}
\]

Formulation (C.1) simply concatenates all control local variables \(\{\Sigma_i, Q_i, g_i, \Lambda_i, \Gamma_i\}\) into \(\xi_i\) and writes the feasibility problem, (4.20), as the sum of indicator functions. Non control nodes are embedded with
the trivial set \( \{ \xi_i = 0 \} \) such that they do not disturb the last coupling sum constraint, i.e.,

\[
\sum_{i \in N} R_i \xi_i - r_i \succeq 0 \iff \sum_{i \in N} \Diag \left( \text{diag}\{ \text{vec}(A_i) \}, \text{diag}\{ \text{vec}(\Gamma_i) \} \right) \geq 0 \quad \text{(C.3)}
\]

\[
\iff \sum_{i \in N_c} A_i \geq 0, \sum_{i \in N_c} \Gamma_i \geq 0, \sum_{i \in N_c} \Sigma_i \succeq \frac{\epsilon \Sigma}{n} I_{2n}
\]

In [50] Aybat and Hamedani derive an distributed algorithm for the following class of optimization problems:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in \mathbb{R}} \rho_i(\xi_i) + f_i(\xi_i) \\
\text{subject to} & \quad \sum_{i \in N} R_i \xi_i - r_i \in \mathcal{K}_c
\end{align*}
\]

where \( \mathcal{K}_c \subseteq \mathbb{R}^m \) denotes a closed convex cone, \( \rho_i \) is a proper, closed, convex, extended value function and \( f_i \) is a smooth, convex functional with Lipschitz constant \( L_i \). Let \( \mathcal{K}_c^0 \) denote polar cone of \( \mathcal{K} \). In [50] each \( i \) corresponds to a node of an undirected connected graph \( G = (N, E) \). Consult [9], [34], [50] and [51] for further insight and definitions. Considering problem (C.4), if (1) strong duality holds and (2) a primal dual solution exists, the following distributed iterative procedure:

\[
\xi_i(k+1) \in \arg \min_{\xi_i} \phi_i(\xi_i) + f_i(\xi_i) + \left\langle \nabla f_i(\xi_i(k)), \xi_i - \xi_i(k) \right\rangle + \left\langle R_i \xi_i - r_i, y_i(k) \right\rangle + \frac{1}{2\tau_i} \|\xi_i - \xi_i(k)\|^2
\]

\[
p_i(k+1) = \sum_{j \in N_i} s_i(k) - s_j(k) \quad \text{(C.6)}
\]

\[
y_i(k+1) \in \arg \min_{y_i \in \mathcal{K}_c^0} \beta \left(y_i, p_i(k+1) \right) - \left\langle R_i \left( 2\xi_i(k+1) - \xi_i(k) \right) - r_i, y_i \right\rangle
\]

\[
+ \frac{1}{2\kappa_i} \|y_i - y_i(k)\|^2
\]

\[
s_i(k+1) = y_i(k+1) + \sum_{l=0}^{k+1} y_i(l) \quad \text{(C.8)}
\]

is guaranteed to converge to a solution of (C.4), if \( s_i(0) = 2y_i(0) \) and the constants \( \beta, \kappa, \tau \) are properly chosen (Lemma 2) [50]. The structure of problem (C.1) resembles that of (C.4), however an immediate difference is the ambient space: \( \xi \) is a vector but \( \xi \) is a matrix. Note that this difference is not simply a matter of reinterpreting matrix variables \( \xi \) as a “big” vectors \( \text{vec}(\xi) \): for example, it is really non trivial how to translate convex positive semidefinite constraints as convex scalar inequalities. If this was the case semidefinite programming could be cast as a linear program.

Algorithm 8 constitutes an application of (C.5) - (C.6) - (C.7) - (C.8) to problem (C.1), but considering matrix variables and generalizing some analysis of [50]. Note that variable \( p_i \), in equation (C.6), was renamed as \( E_i \) in algorithm 8, since \( P_i \) is already reserved for the Lyapunov certificate of each control node. The actual proofs of convergence can be found in [50] if one uses the general definitions, given here, and follows the exact same steps. We do not present the complete work here for 2 reasons:
(1) it would be, practically, a copy of [50] and (2) we have limited space. Using the notation of [50], we prove that problem (C.1) corresponds to a particular instance of (C.4) (with matrix variables) and, trivially, generalize some concepts:

(i) For any node \( i \in \mathcal{N} \), function \( I_{\Phi_i}(\xi_i) \) is decomposed as:

\[
I_{\Phi_i}(\xi_i) = \rho_i(\xi_i) + f_i(\xi_i)
\]  

(C.9)

with \( \rho_i \) being a proper, closed, convex, extended value function and \( f_i \) a smooth, convex functional with Lipschitz constant \( L_i \). Take \( f_i(\xi_i) = 0 \) and note that convexity and smoothness are naturally fulfilled and any \( L_i \geq 0 \) is a valid Lipschitz constant, i.e.,:

\[
\forall L_i \geq 0 : L_i ||\xi_i - \xi_j|| \geq |f(\xi_i) - f(\xi_j)|, \forall \xi_i, \xi_j
\]  

(C.10)

For simplicity take \( L_i = 0 \) for any node \( i \in \mathcal{N} \). Function \( I_{\Phi_i}(\xi_i) \) is convex, since it is the indicator function of a convex set. To prove that \( I_{\Phi_i} \) is proper we first note that, by definition:

\[
I_{\Phi_i}(\xi_i) \geq 0, \forall \xi_i \in \mathbb{R}^{3n+1+n^2+nl} \times (n^2+nl+2n)
\]  

(C.11)

we have that \( \hat{\xi}_i \in \Phi_i \) hence \( I_{\Phi_i} \) is a proper function. Finally, to show that \( I_{\Phi_i} \) is a closed function we simply have to show that, for any \( \alpha \in \mathbb{R} \), the level set \( \{\xi_i : I_{\Phi_i}(\xi_i) \leq \alpha\} \) is a closed set. Note that:

\[
\alpha < 0 : \{\xi_i : I_{\Phi_i}(\xi_i) \leq \alpha\} = \emptyset
\]  

(C.13)

\[
\alpha \geq 0 : \{\xi_i : I_{\Phi_i}(\xi_i) \leq \alpha\} = \Phi_i
\]

hence the closeness of the level set follows from the closeness of \( \Phi_i \), since the empty set is trivially close.

(ii) There exists a primal dual solution to problem (C.1) and strong duality holds. To prove strong duality we make of Slater’s condition, namely we have to find a feasible point \( \hat{\xi} \), such that all generalized
inequalities hold strictly, in formulation (C.1).

In general, point \( \hat{\xi} \) is not required to exist, since we are restricting the closed loop system to be positive and lie in a polytope. This criteria can be too restrictive, hence we have to restrict ourselves to problems for which these constraints are acceptable, i.e., for which there exists a point \( \hat{\xi} \), such that problem (C.1) is feasible. In practice, this just mean that we considering the subset of problems for which the centralized criteria is indeed suitable. Given \( \hat{\xi} \) and any positive scalar \( \eta > 1 \) consider the following construction:

\[
\begin{align*}
    i \in \mathcal{N} \setminus \mathcal{N}_c & : \: \hat{\xi}_i = 0 \\
    i \in \mathcal{N}_c & : \: \begin{cases}
        \hat{Q}_i = \eta \hat{Q}_i \\
        \hat{g}_i = \eta \hat{g}_i \\
        \hat{A}_i = \eta \hat{A}_i \\
        \hat{\Gamma}_i = \eta \hat{\Gamma}_i \\
        \hat{\Sigma}_i = (1 + \frac{\eta - 1}{2}) \hat{\Sigma}_i
    \end{cases}
\end{align*}
\]

and conclude that \( \hat{\xi} \) is a strict feasible point with respect to the general inequalities of formulation (C.1). This follows trivially since \( 1 < (1 + \frac{\eta - 1}{2}) < \eta \). If the optimal value of (C.1) is finite then, strong duality implies that a dual solution is attained. By assumption, the point \( \tilde{\xi} \) is feasible, hence the optimal value of (C.1) is zero and there exists a primal dual solution.

(iii) This step will generalize some definitions of [50] when one consider matrix input variables. Given some \( W \in \mathbb{R}^{[|E|(2n+n^2+nl) \times (2n+n^2+nl)]} \) define the matrix \( X = (\xi, W) \) and let \( Y = (Y_i)_{i=1}^n \) with \( Y_i \) an arbitrary matrix of dimension \( 2n + n^2 + nl \) by \( 2n + n^2 + nl \). For any constants \( \beta, \tau_i, \kappa_i > 0 \) construct the following diagonal matrices:

\[
D_\beta = \frac{I_{m_1}}{\beta}, \quad D_\tau = \text{Diag} \left( \left\{ \frac{I_{m_2}}{\tau_i} \right\}_{i=1}^n \right), \quad D_\kappa = \text{diag} \left( \left\{ \frac{I_{m_3}}{\kappa_i} \right\}_{i=1}^n \right)
\]

\[
m_1 = |E|(2n + n^2 + nl), \quad m_2 = 3n + 1 + n^2 + nl, \quad m_3 = 2n + n^2 + nl
\]

Given (C.15), the following functions are valid Bregman distance functions that generalize that of [50] (Definition 3):

\[
D_\beta(Y, \tilde{Y}) = \frac{1}{2}\|Y - \tilde{Y}\|_{F, D_\beta}^2, \quad D_\tau(X, \tilde{X}) = \frac{1}{2}\|X - \tilde{X}\|_{F, D_\tau}^2 + \frac{1}{2}\|W - \tilde{W}\|_{\tilde{F}, D_\beta}^2
\]

where the \( Q \) Frobenius norm is defined as \( \|X\|_{F, D} := \text{trace}(X^T DX) \frac{1}{2} = \|D^{1/2} X\|_F \) for any \( D \succ 0 \). One can easily verify that \( \|X\|_{F, D} \) is a norm for any \( D \succ 0 \). The Bregman property follows easily
instead of the euclidean norm we consider:

\[
\psi_x(X) = \frac{1}{2} \text{trace}(\xi^T D_x \xi) + \frac{1}{2} \text{trace}(W^T D_y W)
\]

(C.17)

\[
D_x(X, \hat{X}) = \psi_x(X) - \psi_x(\hat{X}) - (\nabla \psi_x(\hat{X}), X - \hat{X})
\]

\[
\psi_y(Y) = \frac{1}{2} \text{trace}(Y^T D_y Y)
\]

(C.18)

\[
D_x(Y, \hat{Y}) = \psi_y(Y) - \psi_y(\hat{Y}) - (\nabla \psi_y(\hat{Y}), Y - \hat{Y})
\]

since \(\psi_x, \psi_y\) are simply quadratic functions, hence continuously differentiable and strongly convex.

Applying definitions (C.15) on (C.16) is trivial that:

\[
D_x(Y, \hat{Y}) = \sum_{i \in \mathcal{N}} \frac{1}{2\kappa_i} \|Y_i - \hat{Y}_i\|^2_F
\]

(C.19)

\[
D_x(X, \hat{X}) = \sum_{i \in \mathcal{N}} \frac{1}{2\kappa_i} \|\xi_i - \hat{\xi}_i\|^2_F + \frac{1}{2\beta} \|W - \hat{W}\|^2_F
\]

(C.20)

Given any matrix \(W\), if both \((\xi, \hat{\xi})\) are feasible for problem (C.1), then \((\xi_i, \hat{\xi}_i) \in \Phi_i\) hence (C.20) is further simplified into:

\[
D_x(X, \hat{X}) = \sum_{i \in \mathcal{N}} \frac{1}{2\kappa_i} \left\{ \|\Lambda_i - \hat{\Lambda}_i\|^2_F + \|\Sigma_i - \hat{\Sigma}_i\|^2_F + \|g_i - \hat{g}_i\|^2_F \right. \]

\[+ \left. \|\Gamma_i - \hat{\Gamma}_i\|^2_F + \|Q_i - \hat{Q}_i\|^2_F \right\} + \frac{1}{2\beta} \|W - \hat{W}\|^2_F
\]

(C.21)

This follows easily from the fact that \(\hat{\xi}_i = \xi_i = 0\) for \(i \in \mathcal{N} \setminus \mathcal{N}_c\) and:

\[
\forall A \in \mathbb{R}^{l \times p}, \ n_1, n_2 \geq 1 : \left\| \begin{bmatrix} A \\ 0_{n_1 \times p} \end{bmatrix} \right\|^2_F = \left\| \begin{bmatrix} \text{vec}(A) \\ 0_{n \times n_2} \end{bmatrix} \right\|^2_F
\]

(C.22)

\[
= \left\| \begin{bmatrix} 0_{l \times n_1} A \\ 0_{l \times n_2} \end{bmatrix} \right\|^2_F
\]

\[
= \left\| A \right\|^2_F = \left\| \text{diag} \{\text{vec}(A)\} \right\|^2_F
\]

Using the previous facts one can easily generate \(\text{DPDS}\) simply by adapting (C.5) - (C.6) - (C.7) - (C.8), to problem (C.1). Vector inner products are generalized to the matrix case, \(\langle A, B \rangle = \text{trace}(A^T B)\), and instead of the euclidean norm we consider:

\[
\|\xi_i\|^2 \mapsto \|\Lambda_i\|^2_F + \|\Sigma_i\|^2_F + \|g_i\|^2_F + \|\Gamma_i\|^2_F
\]

(C.23)

\[
\|y_i\|^2 \mapsto \|\hat{Y}_i\|^2_F
\]

(C.24)

Regarding the choice of constants, \(\left\{ \kappa_i, \tau_i \right\}_{i=1}^n\) and \(\beta\), the first remark of section II [50], computes this values in closed form. Given any \(c_i, \beta > 0\) the following choice:

\[
\tau_i = \frac{1}{c_i + \bar{L}_i}, \quad \kappa_i = \frac{c_i}{2\beta c_i |\mathcal{N}_c| + \|R_i\|^2}
\]

(C.25)
insures convergence of (C.5) - (C.6) - (C.7) - (C.8) (Lemma 2)[50]. The same criteria applies for DPDS.

Matrix $R_i$ is given by (C.2) hence $||R_i||^2 = 1$ and the Lipschitz constant $L_i$ was already set to zero, for simplicity. From this one can easily understand the choice of parameters, $\tau_i, \kappa_i, \beta > 0$, in algorithm 8 and the results of theorem 4.3.2. To finish this section, we just highlight that theorem 4.3.2 can actually be refined by considering the ergodic variables:

$$
\xi^k := \frac{1}{k} \sum_{p=1}^{k} \xi(p), \quad Y^k := \frac{1}{k} \sum_{p=1}^{k} Y(p)
$$

where $k \geq 1$, denotes the iteration number. For this variables, one can compute some error bounds regarding the distance of $\sum_{i \in \mathcal{N}} R_i \xi^k_i - r_i$ to the cone of positive definite matrices and some consensus violation results, regarding the dual variables $Y^k$. Both quantities present an $O(1/k)$ convergence rate and, again, the analysis can be found in [50].