 Nested Lyapunov Ellipsoids
Implementation for Gain Scheduling
Techniques in Controller Design

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Mechanical and Aerospace Engineering

by

José Joaquín Mohedano Robles

Thesis Committee:
Prof. Faryar Jabbari, Chair
Prof. Kenneth Mease
Prof. James E. Bobrow

2005
The thesis of José Joaquín Mohedano Robles is approved:

University of California, Irvine
2005
A mi familia y a mis amigos, especialmente a mis abuelos Carmen y Juan R.
# Contents

List of Tables viii

List of Figures x

Abstract of the Thesis xiv

1 Introduction 1

2 Description of the basic equations 5
   2.0.1 First condition: Reachable set 6
   2.0.2 Second condition: Nonsaturation of the controller 8
   2.0.3 Third condition: Optimal performance of the system 9

2.1 Solving the system of matrix inequalities 12

2.2 Conservatism in the model and parameter \( \delta \) 15

3 Ellipsoids with \( P = \text{constant} \):
   First approach to Gain Scheduling 22
   3.1 General proceeding for gain scheduling 24
      3.1.1 Practical implementation guidelines 27
   3.2 Some test for the new method 29
      3.2.1 Importance of \( D \neq 0 \) over \( \epsilon_1 \) (considering \( \beta \)) 29
      3.2.2 Importance of \( D \neq 0 \) over \( \epsilon_1 \) (considering \( \gamma \)) 32
   3.3 First scheme of solution: \( P_{\text{sequential constant}} \) 35
   3.4 Second scheme of solution: \( P_{\text{global constant}} \) 39
   3.5 Limitations of both \( P_{\text{constant}} \) schemes 43
## Nested Sets: \( \dot{r} \leq 0 \)

### 4 New assumptions regarding \( \dot{P} \)

- **4.1** New assumptions regarding \( \dot{P} \)
- **4.2** Parametrization for \( P \)
  - **4.2.1** \( P \) as a Spline Function over \( r \)
  - **4.2.2** \( P(r) \) sensitivity
  - **4.2.3** Description of \( \frac{\delta r(t)}{\delta t} \)

### 4.3 New considerations on the set of LMI

- **4.3.1** Non-Saturation
- **4.3.2** Performance
- **4.3.3** Reachable set

### 4.4 Schemes of solution and practical implementation

- **4.4.1** Practical implementation of the controller
- **4.4.2** Quick facts about \( P_{\text{sequential}} \) method

## Comparing the four methods for the Nested Sets

### 5.1 Study of the unexpected result

### 5.2 Stepsize influence

## New \( P_{\text{sequential variable}} \) method

### 6.1 New strategy: Combined methods (CM)

- **6.1.1** First combined method (1-CM):
  - Divide the sequence on sections solved globally
- **6.1.2** Second Combined Method (2-CM):
  - Solve sequentially by small overlapped groups
- **6.1.3** Third Combined Method (3-CM):
  - Solve sequentially by small overlapped optimized groups

### 6.2 Speeding up the new method

### 6.3 Influence of the weights on the new method

## Input Scheduling criteria with \( \dot{r} \leq 0 \)

### 7.1 Rethinking our previous set of LMI

### 7.2 Modifications on the system of LMI

- **7.2.1** Spline description for the Reachable Set LMI

### 7.3 New scheduling areas on the State Space
7.3.1 Results of this new considerations and conclusions .......................... 119

8 Input Scheduling Criteria with $-\infty < \dot{r} \leq |d_M|$ .......................... 121

8.1 Incorporating $\dot{r} > 0$ to our set of LMI ............................................. 123
8.1.1 Reachable Set LMI ........................................................................ 124
8.1.2 Performance LMI ........................................................................ 128
8.1.3 Non-Saturation LMI and the Nesting Condition LMI ...................... 131
8.2 The LMI system to solve ....................................................................... 132
8.2.1 Strategy to solve the system: The 4 $-\text{CM}_3$ method .................... 135
8.3 Results from the new criteria ................................................................. 140

9 Real time simulations ............................................................................. 142

9.1 Description of the elements of the system ............................................. 145
9.2 Simulation processes considered ............................................................. 149
9.2.1 State Space Scheduling, for $P = \text{constant}$ ......................................... 149
9.2.2 State Space Scheduling, for $\dot{r} \leq 0$ .................................................. 152
9.2.3 Input Scheduling, for $\dot{r} \leq 0$ .......................................................... 155
9.2.4 Input Scheduling, for $-\infty < \dot{r} \leq |d_M|$ ............................................. 156
9.3 Disturbance signals and results ............................................................... 159
9.3.1 Disturbance signals ........................................................................ 159
9.3.2 Results for $\dot{r} \leq 0$, State Space Scheduling ...................................... 161
9.3.3 Results for $\dot{r} \leq 0$, Input Scheduling ............................................... 167
9.3.4 Results for $-\infty < \dot{r} \leq |d_M|$ simulations ............................................ 171
9.3.5 Final comparisons ............................................................................ 175

10 Conclusion ............................................................................................. 177

Bibliography ............................................................................................... 179

A Matlab codes for the main methods developed ...................................... 181

A.1 Matlab code for $P^{\text{sequential}}_{\text{constant}}$ State Space Scheduling .............. 181
A.1.1 Main file ......................................................................................... 181
A.2 Matlab code for \( P_{\text{constant}}^{\text{global}} \)
State Space Scheduling .......................... 185

A.3 Matlab code for \( P_{\text{variable}}^{\text{sequential}} \), considering \( \dot{r} \leq 0 \).
State Space Scheduling .......................... 189

A.4 Matlab code for \( P_{\text{variable}}^{\text{global}} \), considering \( \dot{r} \leq 0 \).
State Space Scheduling .......................... 192

A.5 Matlab code for \( 3 - CM_3 \), considering \( \dot{r} \leq 0 \).
State Space Scheduling .......................... 196

A.6 Matlab code for \( 3 - CM_3 \), considering \( \dot{r} \leq 0 \).
Input Scheduling ................................. 205

A.7 Matlab code for \( 4 - CM_3 \), considering \( -\infty < \dot{r} \leq |d_M| \).
Input Scheduling ................................. 206

A.8 Time simulation code for \( P = \text{constant} \) ................................. 218

A.9 State Space Scheduling, for \( \dot{r} \leq 0 \) ................................. 219

A.10 Actuator Input Scheduling, for \( \dot{r} \leq 0 \) ................................. 224

A.11 Actuator Input Scheduling, for \( -\infty < \dot{r} \leq |d_M| \) ................................. 226
List of Tables

3.1 First example of solution for the \( P_{\text{constant}}^{\text{sequential}} \) scheme .......................... 36
3.2 First example of solution for the \( P_{\text{constant}}^{\text{global}} \) scheme .......................... 41

5.1 Comparing the solution for all 4 methods: \( P_{\text{constant}}^{\text{sequential}}, P_{\text{constant}}^{\text{global}}, P_{\text{variable}}^{\text{sequential}} \) and \( P_{\text{variable}}^{\text{global}} \) .................................................. 66
5.2 Comparing the solution for all 4 methods, different system: ....................... 67
5.3 Studying the unexpected result on the \( P_{\text{variable}}^{\text{sequential}} \) method. Performances on the critic part of the \( \delta \) sequence: ............................... 71
5.4 Studying the unexpected result on the \( P_{\text{variable}}^{\text{sequential}} \) method. Different sequences of \( \delta \) to check the stepsize sensibility: ...................... 75

6.1 Scheme of solution for the First Combined Method (1-CM) ................. 81
6.2 Comparing \( P_{\text{constant}}^{\text{global}}, P_{\text{variable}}^{\text{sequential}} \) and the 1-CM .......................... 83
6.3 Comparing the procedure of 1 – \( CM_3 \) with 2 – \( CM_3 \) .......................... 84
6.4 Comparing \( P_{\text{constant}}^{\text{global}}, P_{\text{variable}}^{\text{sequential}} \) and 1-CM with the 2-CM method ... 86
6.5 Comparing \( P_{\text{constant}}^{\text{global}}, P_{\text{variable}}^{\text{sequential}} \) and the 1-CM .......................... 88
6.6 Comparing the procedure of 2 – \( CM_3 \) with 3 – \( CM_3 \) ........................ 90
6.7 Comparing calculation time and numerical complexity of 3 – \( CM_3, P_{\text{variable}}^{\text{sequential}}, P_{\text{constant}}^{\text{global}}, P_{\text{variable}}^{\text{global}} \) .......................... 94
6.8 Speeding up the 3 – \( CM \) method \((R^3)\). ........................................... 96
6.9 Speeding up the 3 – \( CM \) method \((R^2)\). ........................................... 98

7.1 Summary of the Input Scheduling Criteria time evolution ..................... 118
7.2 Including the Reachable Set LMI for the inner ellipsoids \( (3 – CM_3) \) . 119

8.1 Describing the procedure for \( 4 – CM_3 \) .............................................. 136
8.2 Comparing the new method for Input Scheduling \((-\infty < \dot{r} \leq |d_M|)\) with the old methods .......................... 141
# List of Figures

2.1 Example of solution set for the first basic equations ........................................ 14
2.2 $\beta$ vs $\delta$ ........................................................................................................ 21
2.3 Sensitivity of $\beta$ vs $\alpha$ plot against different $\delta$ ........................................... 21

3.1 Graphics of $P^\text{sequential constant}$ and $P^\text{global constant}$ schemes ............... 26
3.2 Implementation example .......................................................................................... 28
3.3 $\beta$ vs $\alpha$ for different $D$ ................................................................................. 30
3.4 $\beta$ vs $D$ for $\alpha = 2$ and $\delta = 1$ ................................................................. 30
3.5 $\gamma$ vs $D$ ........................................................................................................... 33
3.6 $\gamma$ vs $\alpha$ ........................................................................................................ 34
3.7 $\gamma$ vs $\delta$ ........................................................................................................ 34
3.8 Ellipsoids and Sat. limits for $P^\text{sequential constant}$ and $\delta = 1$ ....................... 37
3.9 Ellipsoids and Sat. limits for $P^\text{sequential constant}$ and $\delta = 2$ ....................... 38
3.10 Ellipsoids and sat. limits for $P^\text{sequential constant}$ and $\delta = 4$ ....................... 38
3.11 Ellipsoids and sat. limits for $P^\text{sequential constant}$ and $\delta = 5$ ....................... 38
3.12 Graphics of $P^\text{sequential constant}$ and $P^\text{global constant}$ schemes ............... 39
3.13 $P^\text{sequential constant}$ and $P^\text{global constant}$ ................................................ 42
3.14 $P^\text{sequential constant}$ and $P^\text{global constant}$ and the hypothetical low gain method .... 44

4.1 $P^\text{sequential variable}$ and $\alpha$ I ...................................................................... 63
4.2 $P^\text{sequential variable}$ and $\alpha$ II ...................................................................... 64

5.1 Comparison of all four methods ............................................................................ 66
5.2 Second comparison for all 4 methods .................................................................... 68
5.3 Ellipsoids corresponding to the sequential/global anomaly ................................. 69
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.10</td>
<td>Actuator input, for the $\dot{r} \leq 0$ case and an Equally Distributed Disturbance</td>
</tr>
<tr>
<td>9.11</td>
<td>State Space evolution, for the $\dot{r} \leq 0$ case and an Equally Distributed Disturbance, Openloop system</td>
</tr>
<tr>
<td>9.12</td>
<td>State Space evolution, for the $\dot{r} \leq 0$ case and a Sinusoidal Signal</td>
</tr>
<tr>
<td>9.13</td>
<td>Selected controller gain, for the $\dot{r} \leq 0$ case and an Equally Distributed Disturbance</td>
</tr>
<tr>
<td>9.14</td>
<td>Actuator input, for the $\dot{r} \leq 0$ case and an Equally Distributed Disturbance</td>
</tr>
<tr>
<td>9.15</td>
<td>State Space evolution, for the $\dot{r} \leq 0$ case and a Sinusoidal Signal, Openloop system</td>
</tr>
<tr>
<td>9.16</td>
<td>Selected controller gain, for the $\dot{r} \leq 0$, Input Scheduling case, and an Equally Distributed Disturbance</td>
</tr>
<tr>
<td>9.17</td>
<td>Actuator input, for the $\dot{r} \leq 0$, Input Scheduling case, and an Equally Distributed Disturbance</td>
</tr>
<tr>
<td>9.18</td>
<td>Selected controller gain, for the $\dot{r} \leq 0$ case and Sinusoidal Disturbance Signal</td>
</tr>
<tr>
<td>9.19</td>
<td>Actuator input, for the $\dot{r} \leq 0$ case and Sinusoidal Disturbance Signal</td>
</tr>
<tr>
<td>9.20</td>
<td>Selected controller gain, for the $-\infty &lt; \dot{r} \leq</td>
</tr>
<tr>
<td>9.21</td>
<td>Actuator input, for the $-\infty &lt; \dot{r} \leq</td>
</tr>
<tr>
<td>9.22</td>
<td>Selected controller gain, for the $-\infty &lt; \dot{r} \leq</td>
</tr>
<tr>
<td>9.23</td>
<td>Actuator input, for the $-\infty &lt; \dot{r} \leq</td>
</tr>
<tr>
<td>9.24</td>
<td>Comparison of the Gain $K_2$ obtained for the 3 main approaches simulated</td>
</tr>
<tr>
<td>9.25</td>
<td>Comparison of the Scheduling parameter $r$ obtained for the 3 main approaches simulated</td>
</tr>
</tbody>
</table>
Acknowledgements

I would like to thank to my advisor, Professor Faryar Jabbari, for his support during my study, his advice and his dedication. Equally I would like to thank Mr. Pete Balsells, for offering me such a great opportunity to come to the University of California, Irvine, through the Balsells Fellowship Program.

Thanks also to Professor Roger Rangel, for his indispensable help and dedication as the Director of the Balsells Fellowship Program; to all the past and current members of the Control lab for their help and hospitality, as well as all my friends from the Mechanical and Aerospace Department. I also would like to thank the rest of my committee members, Professor Kenneth Mease and Professor James E. Bobrow.

And finally, special thanks to all my family and friends in Catalonia, the rest of Spain, Mexico and US for their unconditional support.
Abstract

Nested Lyapunov Ellipsoids
Implementation for Gain Scheduling Techniques
in Controller Design

by
José Joaquín Mohedano Robles
Master of Science in Mechanical and Aerospace Engineering
University of California, Irvine, 2005
Prof. Faryar Jabbari, Chair

State and Input scheduled controllers are designed for disturbance attenuation in linear systems with bounded actuators. The controller is scheduled according to the proximity to the origin of the state in the state-feedback case and the compensator state in the Input scheduling case. This procedure yields a linear parameter-varying controller structure that allows higher-gain, and hence higher-performance controllers as the states move closer to the origin. The results give sufficient conditions for the satisfaction of a parameter-dependent performance measure while respecting the saturation bounds. Linear splines are used to obtain solutions through the use of standard LMI software. A thorough analysis of the strategies to solve the problem is accomplished through developing new and more powerful sequential schemes of solution, particularly for the Input scheduling case, as well as through avoiding numerical problems that traditional methods have presented for some particular systems. The study finishes with several new results on time simulation of the resulting controllers.
Chapter 1

Introduction

Actuator capacity limitation has been a major topic of research in recent decades. However, due to a conservatism of the assumptions considered [5], the results produced have been less than impressive. In many cases, linear control methods are initially used to obtain desirable nominal controllers, while in the second step, anti-windup techniques are developed in order to reduce the controller gain so that saturation is avoided with only a graceful degradation of performance.

In general, most of these techniques were based on ad hoc techniques. Recently however, more precise and rigorous approaches have been developed, as robust and nonlinear control developed. These improvements have allowed researchers to focus on more specific aspects of the actuator capacity limitation and particular cases of actuator implementation, such as the disturbance attenuation and the actuator amplitude and rate saturation [7].

In the case of disturbance attenuation, the guaranteed performance levels, which are to be minimized, are functions of actuator capacity. The result is that larger actuators offer better (lower) performance, as the controller has more strength to stop the disturbances and keep the system stable.

When a LTI controller is used for a reasonably large range of disturbances, the design often is based on a worst-case scenario of disturbance. Here, we rely on concepts
of parameter-dependent Lyapunov functions and performance measures, as well as scheduling the gains in order to reduce such conservatism.

Scheduling controllers is not a new control topic. After some years as one of the research methods to consider they had led to some interesting possibilities, as the use of a family of parameterized Lyapunov matrices for the LQ type problems, often obtained from a parameterized Riccati equation. More recent work in this area includes scheduling state feedback controllers with an $H_\infty$ approach, which tries to extend the initial ideas of parameterized Lyapunov matrices to establish global stability and disturbance rejection. At this point, apart from the initial state feedback structure, the output feedback problem was also incorporated as a valid scheme.

This Thesis is focused on discussing the main causes of conservatism considered in the previously mentioned approaches, following the criteria described in references [1], [3] and [10]. Accordingly, several versions of the problem will be developed, moving from the most conservative approach to bolder and more powerful methods.

In particular, results regarding the design of high-performance scheduled state and scheduled input controllers are presented for systems with limited actuator amplitude capacity (actuator rate is not considered). The basic approach will use families of parameterized Lyapunov matrices that provide us several ellipsoids on the state space and a scheduling parameter. This parameter $(r)$ will be used to adjust the controller. In the initial state feedback case these ellipsoids will also be related to the state of the system. On the other hand, for the later input scheduling case the ellipsoids will be combined with the saturation bounds, describing scheduling areas linked to the compensator states.

The scheduling criteria will always be based on the idea of looking for the largest scheduling parameter $(r)$ such that the associated scheduling area containing the appropriate state vector is the smallest. Smaller regions will allow higher gain controllers, and this will imply a better performance. If the state vector moves further from the origin due to disturbance, smaller gains will be used. Therefore the guaranteed performance measure (mainly the $L_2$ gain) will be parameter-dependent as well, and the choice of the controller gain will be a function of the system response to that particular disturbance, and not any a-priori estimate.
Although the $L_2$ gain is the principal performance measure, other measures will be also considered in the first chapters, such as the peak to peak gain.

During this evolution from the most conservative to the boldest approaches described in the bibliography, most of the simplifications and hypotheses explained therein, as well as the details of the derivation of our control objectives expressions will be discussed.

After that, a description and comparison in detail of the traditional strategies used for solving the corresponding set of conditions and obtain the final performances, controllers and Lyapunov functions will follow. These traditional strategies will present several degrees of accuracy and complexity, and in some cases they will present numerical problems (as happens in the case of the Sequential Scheme).

Because of its reasonably low complexity and adequate results in most cases, the Sequential Scheme of solution for the the system of conditions will be considered to be the best way to obtain a controller that matches our objectives. However, the fact that in some cases its results are unexpectedly bad will justify a deep analysis of this method, and the final elaboration of a new and improved strategy (The Combined Method Scheme) which will keep the essence of the Sequential Scheme, while avoiding its defects and providing an improved performance.

Once this new scheme has been established, it will be used to solve the most powerful and least conservative approaches to the problem presented in the references. Finally, after developing the necessary Matlab codes, the simulations of the behavior of the system under the designed controllers will be obtained, and their results will be discussed and compared.

All this study will be structured through the chapters of this document as follows:

Chapter 2 will give the mathematical description of our control objectives in the form of LMI, as well as the major causes for conservatism. Chapter 3 will suggest a first approach to the concept of Gain Scheduling based on a constant Lyapunov matrix. This first approach will be followed by Chapter 4, which will introduce the spline description [6] to incorporate the linearly varying control parameter formulation.
After this initial description, Chapter 5 will analyze the basic schemes to solve the LMI system and, after comparing the different results, Chapter 6 will propose a new strategy to obtain better results at a reasonable complexity for the LMI solver [8] (The so called Combined Method Scheme).

Finally, Chapter 7 will leave the state feedback structure to introduce the Input feedback structure, which will be solved with the new Scheme. Chapter 8 will consider the most general case, in terms of the Lyapunov matrix temporal evolution, and will solve it with an adapted version of the Combined Method Scheme).

The study will be completed on Chapter 9, where the time response of the system after simulating the implementation of the most important approaches will be presented.
Chapter 2

Description of the basic equations

As stated in chapter 1, we have to begin this study by defining the main objectives that will act as constraints for the design of our controller gains. These constraints will lead us to 3 basic equations that we will have to accomplish, which are:

- The system has to remain in the vicinity of an equilibrium point given any initial conditions inside this vicinity (it has to be stable in the sense of Lyapunov [2]).

A part from this classical stability definition referred to the behavior under some given initial conditions, we will also have to include the response to a perturbation. To do so we will modify slightly the Lyapunov criteria for stability, as we will see later on.

- Controller cannot violate the saturation limit at any reachable state.

- Performance has to be as good as possible. That is to say: the effects of the disturbance have to be as small as possible all over the considered points of the space state.

To describe mathematically this constraints we will use the concept of Lyapunov function, which is based on the state of the system described by variable $x$:

$$ V = x^T P x $$  \hspace{1cm} (2.1)
And also we will use the state space description, based also in $x$, the actuator input $u$, and the disturbance input $w$.

\[
\begin{align*}
\dot{x} &= Ax + B_1w + B_2u \\
z &= Cx + D_1w + D_2u
\end{align*}
\]

And considering the state feedback, configuration turns into:

\[
\begin{align*}
\dot{x} &= (A + B_2K)x + B_1w \\
z &= (C + D_2K)x + D_1w
\end{align*}
\]

(2.2)

Where $z$ is the controlled output of the system.

### 2.0.1 First condition: Reachable set

We say that we have stability in the sense of Lyapunov when we can define a matrix $P > 0$ such that $V$ in equation (2.1) is positive definite for all states and $\dot{V} \leq 0$. This definition of stability tells us that the system is going to stay in a bounded vicinity of an equilibrium point for any initial condition on this vicinity. However, in our case we are going to modify this definition to consider the response to a perturbation. What we want is to remain on the vicinity of an equilibrium point even when we are perturbated externally.

To get this we consider that $\dot{V}$ can be positive, but under a certain bounded value given by the maximum disturbance $w_{max}^2$, the Lyapunov function value $V$, and a scale parameter $\alpha > 0$. The final expression for the condition is:

\[
\dot{V} + \alpha(V - w_{max}^2) < 0
\]

(2.3)

Which means that $\dot{V}$ can be positive but only inside the region where

\[
V < w_{max}^2
\]

(2.4)

This is a stable region (or invariance set) because, as we will see below, the state of the system cannot go out of it once it ”falls in” (and it will enter since $\dot{V} < 0$ outside it). Therefore, this equation defines the so called ”vicinity” mentioned during the description of the Lyapunov stability and the desired response to a perturbation.
To show this, we can say that on the boundary of the region \((V = w_{\text{max}}^2)\) equation (2.3) tells us that \((\dot{V}) = 0\), whereas for \((V > w_{\text{max}}^2)\) we have that \(\dot{V} < 0\). Therefore, it is impossible for the state of the system to cross the boundary of the region from inside, but not from outside.

According to the structure of expression (2.1) this limit region is going to have the shape of an ellipsoid, represented by the square matrix \(P\).

By using expression (2.3), considering \(P = \text{constant}\), and by substituting on it equations (2.2) and (2.1) we end up with the following expression for the condition, which is the one that we will use from now on:

\[
0 > x^T P x + x P \dot{x} + \alpha (x^T P x - w_{\text{max}}^2) \\
0 > [(A + B_2 K)x + B_1 w]^T P x + x P [(A + B_2 K)x + B_1 w] + \\
+ \alpha (x^T P x - w_{\text{max}}^2) \\
0 > x^T (A^T P + PA)x + x^T (K^T B_2^T P + PB_2 K)x + \\
+ w^T B_1^T P x + x^T P B_1 w + \alpha (x^T P x - w_{\text{max}}^2) \\
0 > x^T (A^T P + PA + K^T B_2^T P + PB_2 K + \alpha P)x + \\
+ w^T B_1^T P x + x^T P B_1 w - \alpha w_{\text{max}}^2
\]

By forcing some conservatism, we can change \(w_{\text{max}}^2\) for \(w^2\) in the last term and then group terms in a matrix:

\[
\begin{pmatrix}
(A^T P + PA + K^T B_2^T P + PB_2 K + \alpha P) & PB_1 \\
B_1^T P & -\alpha
\end{pmatrix} < 0 \tag{2.5}
\]

Which is the expression that we are going to work with in order to get a reachable set. As we have proved, if our parameters \(P\) and \(K\) accomplish this "Linear Matrix Inequality" (LMI) then, the Invariance Set \(V \leq w_{\text{max}}^2\) is warranted.
2.0.2 Second condition: Nonsaturation of the controller

As we have explained before, we do not want our controller to saturate. To get the mathematical expression of this fact we proceed as follows:

\[
\begin{align*}
    u_{\text{lim}}^2 & > |u|^2 \\
    w_{\text{max}}^2 u_{\text{lim}}^2 & > |u|^2 w_{\text{max}}^2 \\
    w_{\text{max}}^2 & > u^T \frac{w_{\text{max}}^2}{u_{\text{lim}}^2} u
\end{align*}
\]

On the other hand, consider that we accomplish the following LMI (Linear Matrix Inequality):

\[
\begin{pmatrix}
    P & K^T \\
    K & \frac{u_{\text{lim}}^2}{w_{\text{max}}^2} I_n
\end{pmatrix} > 0 \quad (2.6)
\]

Where \( I_n \) is an identity matrix with dimension \( n \) equal to the dimension of the actuator input vector \( u \)

After applying the Schur complement result to LMI 2.6, we can rewrite it as:

\[
\begin{align*}
    P & > K^T \frac{w_{\text{max}}^2}{u_{\text{lim}}^2} K \\
    x^T [P] x & > x^T \left[ K^T \frac{w_{\text{max}}^2}{u_{\text{lim}}^2} K \right] x \\
    V & > u^T \frac{w_{\text{max}}^2}{u_{\text{lim}}^2} u
\end{align*}
\]

Considering that we will solve LMI 2.6 together with LMI 2.5, we can be sure that the will have \( V \leq w_{\text{max}}^2 \), so we can say:

\[
\begin{align*}
    w_{\text{max}}^2 & > V > u^T \frac{w_{\text{max}}^2}{u_{\text{lim}}^2} u
\end{align*}
\]

\(^1\text{And } n = 1 \text{ in all the cases that we will consider on this report.}\)
And therefore we can say that by solving LMI 2.6, provided \( w_{\max}^2 > V \), we are enforcing that the controller will not saturate on this area of the state space.

**2.0.3 Third condition: Optimal performance of the system**

We can describe the performance of the system in 2 different ways. The first case would be based on the maximum value of the output of the system (\( |z|_{\max} = \beta \)). For this case the procedure is very similar to the derivation of the Saturation LMI:

\[
\beta^2 > |z|^2 = (Cx + Du)^T(Cx + Du)
\]

\[
\beta^2 > \left[ (Cx + Du)^T(Cx + Du) \right] \frac{w_{\max}^2}{(x^TPx)}
\]

\[
x^TPx > (Cx + Du)^T \frac{w_{\max}^2}{\beta^2} (Cx + Du)
\]

\[
\vdots
\]

\[
0 > \left( \begin{array}{cc} P & C^T + K^TD^T \\ C + DK & \frac{\beta^2}{w_{\max}^2} I_m \end{array} \right) \tag{2.7}
\]

Where \( I_m \) is an identity matrix with dimension \( m \) equal to the dimension of the output vector \( z \). On the other hand, we have the description of the performance based on the \( L_2 \) gain of the system, which is defined by the following inequality:

\[
\dot{V}(x) < \gamma^2 w^T w - z^T z
\]

\[
V(x) < \int_0^t \gamma^2 w^T w dt - \int_0^t z^T z dt
\]

\[
\int_0^\infty z^T z dt < \int_0^t z^T z dt + V < \int_0^t \gamma^2 w^T w dt
\]

\[
\int_0^{t_f} z^T z dt < \int_0^{t_f} \gamma^2 w^T w dt
\]

\[
\|z\|_{L_2} < \|\gamma^2 w\|_{L_2}
\]

\(^2m = 1\) in all the cases that we will consider on this report.
And if we consider the maximum value for $\gamma$ along the time interval $t = [0...t_f]$ we can take it out from the $L_2$ norm and end up with the expression:

$$|z|_{L_2} \leq \gamma^2_{\text{max}}\|w\|_{L_2}$$

Where the only assumption made is $V(0) = 0$.

Now, by substituting expressions (2.1) and (2.2) (and considering $P = \text{constant}$) we end up getting the corresponding matrix form:

$$0 > \dot{V}(x) - \gamma^2 w^T w + z^T z$$

$$0 > \dot{\gamma}^2 \gamma^2 w^T w + z^T z + [Ax + B_1w + B_2Kx]^T P x + x^T P [Ax + B_1w + B_2Kx] +$$

$$-\gamma^2 w^T w + (Cx + DKx)^T (Cx + DKx)$$

$$0 > x^T (AT^2 + PA + PB_2K + KTB_2K^T P)x +$$

$$+w^T (B_1B_1^T P)x + x^T (P B_1)w + w^T (-\gamma^2)w +$$

$$+x^T (C^T C + C^T DK + K^T D^T C + K^T D^T DK)x$$

We can re-write this expression as the product of a matrix with two vectors:

$$0 > \begin{pmatrix} x & w \end{pmatrix} \begin{pmatrix} AT^2 + PA + PB_2K + KTB_2K^T P + C^T C + C^T DK + K^T D^T C + (DK)^T DK \ P B_1 \ B_1^T P \end{pmatrix} \begin{pmatrix} x \\ -\gamma^2 \end{pmatrix}$$

Next we separate some of the terms that show up adding into the matrix, so we can apply a Schur complement later on.

$$0 > \begin{pmatrix} AT^2 + PA + PB_2K + KTB_2K^T P & \ P B_1 \\ B_1^T P & -\gamma^2 \end{pmatrix} + \begin{pmatrix} C^T + K^T D^T \\ 0 \end{pmatrix} \begin{pmatrix} C + DK & 0 \end{pmatrix}$$
And finally, after applying the Schur complement, we get the final expression:

$$
\begin{pmatrix}
A^T P + PA + PB_2 K + K^T B_2^T P & PB_1 & C + K^T D^T \\
B_1^T P & -\gamma^2 & 0 \\
C + DK & 0 & -I
\end{pmatrix} < 0
$$

(2.8)

When we solve this LMI, the values of $P$ and $K$ ensure a certain $L_2$ gain $\gamma$.

Therefore, the system has to be stable (Although, the Invariance Set will not be, in general, $V \leq w_{\text{max}}^2$. Unlease we solve this LMI together with LMI 2.5).
2.1 Solving the system of matrix inequalities

The expressions that we will want to solve are LMI 2.5 (Reachable set), LMI 2.6 (Nonsaturation) and LMI 2.7 or LMI 2.8 (Performance). Ideally the only unknowns that we will have are $P$, $K$ and $\beta$ or $\gamma$. Since we want to optimize $\beta$ or $\gamma$, we will solve the system through a minimization of these parameters under the corresponding LMI constraints. To solve this minimization we will use the MATLAB function \textit{mincx}, present on the \textit{LMI Control Toolbox}. This function solves the convex program of the minimization over the set of LMI conditions.

The problem comes when we have a parameter that does not show up linearly on our system, which is the case of $P$ and $K$ in (2.8). To avoid this and allow the convex search, we pre and post multiply both terms of all three inequalities by

$$
\begin{pmatrix}
P^{-1} & 0 \\
0 & 1
\end{pmatrix}
$$

Obtaining a new version for the set conditions, this time depending on some new and more convenient parameters:

$$Q = P^{-1}, \quad F = KQ \quad \text{and} \quad \beta \text{ or } \gamma$$

These new LMI are the next:

**Reachable set LMI:**

$$
\begin{pmatrix}
QA^T + AQ + F^TB_2^T + B_2F + \alpha Q & B_1 \\
B_1^T & -\alpha
\end{pmatrix}
< 0 \quad (2.9)
$$

**Saturation LMI:**

$$
\begin{pmatrix}
Q & F^T \\
F & \frac{u_2^2}{u_{\text{lim}}^2} \frac{u_2^2}{u_{\text{lim}}^2}
\end{pmatrix}
> 0 \quad (2.10)
$$
Performance LMI (based on $\beta$ or $\gamma$):

For the maximum output ($\beta$):

$$ \begin{pmatrix} Q & Q^T + F^T D^T \\ CQ + DF & \frac{\gamma^2}{w_{\text{max}}^2} \end{pmatrix} < 0 \quad (2.11) $$

For the $L_2$ gain:

$$ \begin{pmatrix} QA^T + AQ + B_2 F + F^T B_2^T P & B_1 & Q^T + F^T D \\ B_1^T & -\gamma^2 & 0 \\ CQ + DF & 0 & -I \end{pmatrix} < 0 \quad (2.12) $$

This is the first set of basic equations that we can solve and get some preliminary results. From now on we will improve this equations to achieve more powerful and accurate results for different situations.

On figure 2.1 we have an example of solution for this first set of basic LMI. The points included on that ellipsoid correspond to states of the system that accomplish:

- Reachable set condition
  $$ V \leq w_{\text{max}}^2 $$

- Non saturation condition
  $$ u_{\text{lim}}^2 > |u|^2 $$

- Present a minimized performance in front of any disturbance bounded by $w_{\text{max}}$.
  $$ \min\{\beta \text{ or } \gamma\} $$

And therefore, as a solution for the system, this ellipsoid has associated a particular value for $P$, $K$, $\gamma$ and $\beta$. 
Figure 2.1: Example of solution for the system of equations. *On the included area the system is stable in the sense of Lyapunov, Actuator does no saturate and we have a minimal performance*

Notice how, a part from the scales corresponding to the values of the two state variables, the figure includes the value of the level set of the Lyapunov function associated to the ellipsoid. For this initial example, it corresponds directly to the Reachable set condition

\[ V < w_{\text{max}}^2 = 10^6 \]
2.2 Conservatism in the model and parameter $\delta$

The Lyapunov criteria used here for stability relies on the existence of a Lyapunov function $V = x^T P x$. Other objectives such as $L_2$ gain also use a $P$ matrix, but it does not have to be the same $P$ necessarily [4]. The problem is that, as you will see below, the fact of using different $P$’s cause problems when we want to find $K$. This is a typical problem in multi-objective control.

The problem is that we may consider different $P$ (and $Q$) but $K$ has still to be the same in all the cases in order to be implementable. This fact limits our choices a lot. For example: Suppose we solve the system of LMI and we get two different $Q (= P^{-1})$. To get the same $K$ we have to enforce:

$$
\begin{align*}
F_1 &= k_1 Q_1 \\
F_2 &= k_2 Q_2 \\
F_1 &= P_1 F_1 = P_3 F_3 \\
k_1 &= k_2
\end{align*}
$$

Which destroys the convexity of the search.

Therefore, in our case we will start assuming that $P$ is the same, but on the other hand we will relax this conservatism by modifying some other parameters. Imagine that we redefine the disturbance limit of the system as follows:

$$
\hat{w}_{\text{max}} = \frac{w_{\text{max}}}{\delta} \quad \text{for} \quad \delta > 0
$$

We can substitute $\frac{w_{\text{max}}}{\delta}$ where we had $w_{\text{max}}$ before and then this would make us consider a new system of equations:

Reachable set LMI:

$$
\begin{pmatrix}
Q A^T + A Q + F^T B_2^T + B_2 F + \alpha Q & B_1 \\
B_1^T \\
\end{pmatrix} < 0
$$

(2.13)
Saturation LMI:
\[
\begin{pmatrix}
Q & F^T \\
F & \delta^2 \frac{u_{\text{lim}}}{w_{\text{max}}} \\
\end{pmatrix} > 0
\]  
(2.14)

Performance LMI (based on $\beta$ or $\gamma$):
\[
\begin{pmatrix}
Q & QC^T + F^T D^T \\
CQ + DF & \delta^2 \frac{\beta^2}{w_{\text{max}}} \\
\end{pmatrix} < 0
\]  
(2.15)

\[
\begin{pmatrix}
QA^T + AQ + B_2 F + F^T B_2^T P & B_1 & QC^T + F^T D^T \\
B_1^T & -\gamma^2 & 0 \\
CQ + DF & 0 & -I \\
\end{pmatrix} < 0
\]  
(2.16)

As a result of this change, we can say that by introducing $\delta$ we are making the system consider $\hat{w}_{\text{max}}$ smaller or bigger (depending whether $\delta > 1$ or $1 > \delta > 0$).

Throughout this thesis we will consider $\delta \geq 1$. Then we can say that we are making $\hat{w}_{\text{max}}$ smaller, while the saturation limit ($u_{\text{lim}}$) stays without any changes. This fact of dividing $w_{\text{max}}^2$ into different levels by using $\delta$ in $\frac{w_{\text{max}}^2}{\delta}$ will be the main reason for getting different controllers, and will use them for the Gain Scheduling.

Conclusions from this new set of equations:

Next we will go through the effects that $\delta$ has on our set of equations, paying special attention to the variables $P$, $K$ and $\gamma$. Which are our basic unknowns to determine.

1. Reachable set (LMI 2.13):
\[
\begin{pmatrix}
QA^T + AQ + F^T B_2^T + B_2 F + \alpha Q & B_1 \\
B_1^T & -\alpha \\
\end{pmatrix} < 0
\]

This equation stays without any changes with respect to the previous version, LMI (2.9).
That is to say: if we go in inverse order from LMI (2.13) to the initial condition (all the derivation steps in the inverse order) we end up at the same initial statement without any change:

\[ \dot{V} + \alpha (V - w_{\text{max}}^2) < 0 \]

Which means that we still are imposing \( V < w_{\text{max}}^2 \) independently of the value that \( \delta \) has. The resulting controller and Lyapunov function that check this LMI will guarantee the stability of the system, in the sense of Lyapunov, on an ellipsoid defined by the level set of the Lyapunov function defined by:

\[ V \leq w_{\text{max}}^2 \]

Regardless of the value of \( \delta \).

2. Saturation (LMI 2.14):

\[
\begin{pmatrix}
Q & F^T \\
F & \delta^2 \frac{u_{\text{lim}}^2}{w_{\text{max}}} \\
\end{pmatrix} > 0
\]

In this case we, have important changes because of the introduction of \( \delta \). First recall from the derivation of LMI 2.10 that we assumed \( V \leq w_{\text{max}}^2 \).

Right now the situation is different, so the derivation of equation (2.14) will be:

\[
\begin{align*}
\frac{u_{\text{lim}}^2}{w_{\text{max}}^2} & > |u|^2 \\
\frac{w_{\text{max}}^2}{\delta^2} \frac{u_{\text{lim}}^2}{w_{\text{max}}^2} & > |u|^2 \frac{w_{\text{max}}^2}{\delta^2} \\
\frac{w_{\text{max}}^2}{\delta^2} & > u^T \frac{w_{\text{max}}^2}{\delta^2} u
\end{align*}
\]

At this point we consider the expression equivalent to LMI 2.14, but before the change of variable, which is:

\[
\begin{pmatrix}
P & K^T \\
K & \delta^2 \frac{u_{\text{lim}}^2}{w_{\text{max}}^2} \\
\end{pmatrix} > 0
\]
After a Schur complement we get:

\[ P > K^T \frac{w_{\text{max}}^2}{\delta^2 u_{\text{lim}}^2} K \]
\[ x^T P x > u^T \frac{w_{\text{max}}^2}{\delta^2 u_{\text{lim}}^2} u \]

And we assume this time that, as long as this LMI 2.14 will be valid, we will have \( V \leq \frac{w_{\text{max}}^2}{\delta^2} \). So we can say:

\[ \frac{w_{\text{max}}^2}{\delta^2} \geq V > xK^T \frac{w_{\text{max}}^2}{\delta^2 u_{\text{lim}}^2} Kx \] (2.17)

As a result of that, we can say that by enforcing expression 2.14 we are forcing our system not to saturate, but only inside the ellipsoid defined by all the states \( x \) such that:

\[ \{ x^T P x = V \leq \frac{w_{\text{max}}^2}{\delta^2} \} \]

However, if at some point this assumption is no longer possible (actually, none of our conditions and LMI enforces it) LMI 2.14 will not hold, and we will not be guaranteed of not having saturation.

Later on we will see that this fact is what takes us to consider scheduling by different "levels" of the disturbance, indicated by \( \delta \) on \( \frac{w_{\text{max}}^2}{\delta^2} \). This way, when \( V > \frac{w_{\text{max}}^2}{\delta_1^2} \) we will switch to a lower \( \delta_2 < \delta_1 \) so we can have again \( V \leq \frac{w_{\text{max}}^2}{\delta_2^2} \). This lower \( \delta_2 \) will have different variables associated when solving LMI 2.14.

On the other hand, expression 2.17 justifies how when \( \delta > 1 \) the resulting controller \( K \) can be bigger without breaking the inequality (staying away from saturation and keeping \( V < \frac{w_{\text{max}}^2}{\delta^2} \)). According to this, we can also state that as \( \delta > 1 \) grows, decreasing the maximum disturbance considered, the resulting ellipsoid that satisfies equation (2.14) will be smaller and the controller gains \( K \) bigger.
3. **Performance** (LMI 2.15 and LMI 2.16):

\[
\begin{pmatrix}
Q & QC^T + F^T D^T \\
CQ + DF & \delta^2 \frac{\beta^2}{\mu_{max}}
\end{pmatrix} \prec 0
\]

Regarding LMI (2.15) (shown above) we have basically the same analysis as we had for the non saturation condition LMI (2.14). Basically what we have now is that since we are considering smaller values for the maximum disturbance, the corresponding maximum values for the output of the system will be smaller.

\[
\begin{pmatrix}
QA^T + AQ + B_2 F + F^T B_2^T P & B_1 & QC^T + F^T D^T \\
B_1^T & -\gamma^2 & 0 \\
CQ + DF & 0 & -I
\end{pmatrix} \prec 0
\]

On the other hand, regarding equation (2.16) (shown above) we can see that after introducing \( \delta \) on the system, on this case there are no changes with respect to the previous version of this same condition (LMI 2.12).

As a matter of fact, as we said before, the existence of a finite \( L_2 \) gain implies stability on the system response for any state. Therefore, this \( L_2 \) gain, represented by \( \gamma \), is valid not only on any resulting ellipsoid, but throughout all the state space (if we could implement the resulting controller without having saturation, which is not the case).

Therefore, to be realistic with the conclusions and results, we always have to solve this Performance LMI together with the Saturation LMI. If we do so, we will have that, as we consider \( \delta \) growing (smaller disturbances) the saturation LMI will allow bigger controller gains and as a result of that we will have smaller performances, since we will have more aggressive controllers.
Alternative interpretations

Notice that we can also interpret $\delta$ together with $u_{lim}^2$ instead of with $w_{max}^2$. If we do so, $w_{max}^2$ would stay constant and the new saturation limit would be increased, allowing our controllers to be bigger. We would be interpreting the same fact.

As a matter of fact, what we are doing by introducing the parameter $\delta$ in our system of LMI is considering different sub-levels of perturbation inside the maximum level $w_{max}^2$. As we consider smaller perturbations we see that we can increase the controller gain without having saturation, and the performances will be better.

Since the reachable set condition is referred directly to the worst disturbance possible ($w_{max}^2$) in order to define the maximum reachable set, we will just consider it for $\delta = 1$. On the other hand, the rest of conditions can be used for the lower values, and will give smaller ellipsoids from which the state of the system can go out. \(^3\)

On figures (2.3) and (2.2) -next page- we can see the effect of forcing the system to use more aggressive controllers by trying different values for this parameter $\delta$. \(^4\) As expected, as we increase $\delta$, the system allows the gain to be bigger, and as a result of that the performance decreases (which is what we want).

\(^3\) Although ultimately it will never be able to leave the ellipsoid corresponding to $\delta = 1$, since this ellipsoid also satisfies the reachable set condition.

\(^4\) Also referred as Scaling Parameter since it scales our maximum considered disturbance.
Figure 2.2: $\beta$ vs $\delta$, the performance improves as $\delta$ grows making us consider smaller disturbances.

Figure 2.3: Influence of $\delta$ on the $\beta$ vs $\alpha$ response. At the same time as $\delta$ improves performance, $\alpha$ shows a characteristic influence over the performance.
Chapter 3

Ellipsoids with $P = constant$:

First approach to Gain Scheduling

The Gain Scheduling concept is based on the fact that we divide our maximum disturbance $w^2_{max}$ into smaller "sub-levels" by considering several $\delta > 1$ on $\omega = \frac{w^2_{max}}{\delta}$. This allows us to get different controllers for each of these "sub-levels" and therefore, benefits the performance of the system in terms of lower $\beta$ (or $\gamma$) and faster response.

Consider as a particular example the initial ellipsoid $\epsilon_1$ (Fig 2.1). We know that the higher $\delta$ we use, the smaller maximum disturbance we will deal with on the saturation LMI and on the $\beta$-performance LMI. This will turn into higher gains for our controller, smaller space state area related to this value $^1$ and lower performance associated $^2$.

This fact allows us to present a new scheme of procedure which, after solving that initial system of Linear Matrix Inequalities $^3$, it will propose to solve it again considering a set sub-levels of disturbance $\frac{w^2_{max}}{\delta}$; getting different ellipsoids inside the initial alone (with different gains and performances) so we can have a specific controller designs scheduled for different areas of the space state and ranges of disturbance.

$^1$Smaller resulting ellipsoid
$^2$As we said on the previous chapter and we can see on Fig 2.2 shows.
$^3$With its maximum disturbance level $w_{max}$.
As an initial approach to this idea of considering multiple disturbance sub-levels (rather than a unique level represented by $w_{\text{max}}$, as we did on Chapter 2) on this chapter we will consider our Lyapunov function to be constant throughout these different versions of the system, each of them corresponding to a different value of $\delta$. However, on the next chapters of this thesis we will relax this conservative assumption by allowing the Lyapunov function, and more specifically, the matrix $P$ ($V = x^T P x$) to change with $\delta$.

The properties that we want the resulting set of ellipsoids to accomplish are:

- We never saturate on any of the ellipsoids, although we consider different gains on each of them.
- We improve the performance (lower $\beta$ and $\gamma$) as we consider smaller ellipsoids centered on the origin. To do this we have increasing gains as we move to the smaller ellipsoids. As a result of that we reduce the performance against disturbances as the gain grows.
- Faster convergence to the equilibrium point when the state is in the inner ellipsoids. Again because of using more aggressive gains.
- We use the actuator more efficiently, since when the state vector is smaller (we are close to the origin) we are increasing the gain.
3.1 General proceeding for gain scheduling

Next we are going to re-write the initial set of equations that we are going to use and refer throughout this chapter, incorporating the new \( \delta \) parameter:

Reachable set LMI:

\[
\left( \begin{array}{cccc}
QA^T + AQ + F^T B_2^T + B_2 F + \alpha Q & B_1 \\
B_1^T & -\alpha
\end{array} \right) < 0
\]  
(3.1)

Saturation LMI:

\[
\left( \begin{array}{cc}
Q & F^T \\
F & \delta^2 \frac{u_{\text{max}}}{u_{\text{max}}}
\end{array} \right) > 0
\]  
(3.2)

Performance LMI (based on \( \beta \) or \( \gamma \)):

\[
\left( \begin{array}{cc}
Q & QC^T + F^T D^T \\
CQ + DF & \delta^2 \frac{\beta^2}{w_{\text{max}}}
\end{array} \right) < 0
\]  
(3.3)

\[
\left( \begin{array}{cccc}
QA^T + AQ + B_2 F + F^T B_2^T P & B_1 & QC^T + F^T D^T \\
B_1^T & -\gamma^2 & 0 \\
CQ + DF & 0 & -I
\end{array} \right) < 0
\]  
(3.4)

According to what we explained at the beginning of this chapter, by taking advantage of the fact that we do not have to consider saturation, and given the coupling between the 3 LMI’s since \( Q \) and \( F \) are present in all of them; the strategy to implement the Gain Scheduling scheme would be based on solving two main points:

1. Find the initial outer ellipsoid \( \epsilon_1 \). That is to say: find all the corresponding values for \( K \), \( P \) and \( \beta \) or \( \gamma \) by solving LMI (3.1),(3.2) and (3.3) or (3.4) for \( \delta = 1 \).

This way we will get the initial solution, the ellipsoid that we will never leave, according to equation (3.1), even when the worse disturbance comes.
2. Solve for the inner ellipsoids and get their parameters ($K$ and $\gamma$ or $\beta$) from equations (3.2) and (3.3) or (3.4). Each inner ellipsoid that we solve will correspond to one different value for $\delta > 1$, and therefore, a different sub-level of disturbance.

Notice that on this second step we do not consider LMI (3.1) -reachable set-. Therefore, we are not enforcing the system to stay inside these inner ellipsoids. However, we substitute this condition by the assumption '$P = constant$' made during the derivation of LMI (3.1), and then we calculate all 3 parameters ($K$, $P$ and $\gamma$) for all the inner ellipsoids.

At the same time, we have to enforce that the performance for each of this $\delta$ improves with respect to the last $\delta$ considered $^4$. That is to say:

$$\beta|_{\delta=n} \leq \beta|_{\delta=n-1} \quad \text{or} \quad \gamma|_{\gamma=n} \leq \gamma|_{\gamma=n-1}$$

Depending on what expression for the performance are we using.

As we have explained on step 2, when we derived all this set of LMI we considered $Q$ (or $P$) to be constant. This implies that at some of the considered $\delta$ we have to calculate a value for $Q$ and make it common to all the rest. The way we deal with this is going to lead us to two different schemes of resolution for the whole system.

So far, a good option is solving $Q$ at $\delta = 1$, which is where we have 3 equations, so we can calculate 3 parameters ($K$, $P$ and $\gamma$). Then we extend its value to the rest of $\delta$ by considering

$$Q|_{\delta_1} = Q|_{\delta_2} = ... = Q|_{\delta_n}$$

This way, once we solve all values for $\delta = 1$ we have all the elements to solve $\delta_2$ and after that we can proceed with $\delta_3$ and so on. This approach is referred during this thesis as the **Sequential $P$ Constant scheme**.

$^4$Which is the main advantage of using scheduled gains on our controller
Another different possibility is solving all the equations for all the values of $\delta$ at the same time. This way we would get a value for $Q$ where all the values considered for $\delta$ have a contribution -rather than to be only a function of $\delta = 1$. This approach is referred during this thesis as the *Global P Constant scheme*.

![Diagram of Sequential and Global P Constant schemes](image)

**Figure 3.1:** Graphics of $P_{\text{sequential constant}}$ and $P_{\text{global constant}}$ schemes

Notice that the fact of considering a unique $P$ for all the values of $\delta$ (as we are doing either on the *Sequential* or the *Global* schemes) allow us to see graphically the series of values of $\delta$ as a series of level sets of a 3D parabola given by $P \left( x^T P x = \frac{w^2}{\delta} \right)$. This parabola will be the representation of the Lyapunov function over the space state hyperplane. On the next chapters we will look for better methods that avoid such an aggressive assumption by allowing $P$ to be different for each value of $\delta$, and therefore, using a different Lyapunov function at each $\delta$.

It is also important to remark that since we are not considering equation (3.1) for the inner ellipsoids, nothing stops now our system from going from inner ellipsoids to the exteriors. Nevertheless, ultimately the state of the system will never go beyond
the most exterior ellipsoid, which indeed includes equation (3.1 -Maximum Reachable Set-) among its conditions to enforce this limit.

What we are going to consider instead for the inner ellipsoids is a minimal performance, and a non saturation of the controller. This fact involves having $V \leq \frac{w_{\text{max}}^2}{\delta}$ at each inner ellipsoid as an assumption for LMI 3.2.

### 3.1.1 Practical implementation guidelines

After applying this method on any of its 2 variants, we will end up with a set of ellipsoids, one inside the other and with the same shape ($P$ is constant), and the corresponding set of increasing controller gains.

The practical implementation on a controller will be based on choosing which of the different gains corresponds to the current state of the system. To do this, we will follow these steps:

- Evaluate the state of the system
- Evaluate the Lyapunov function for that state ($V$).
- From $V$ decide in which of the ellipsoids are we included according to:
  $$V \leq \frac{w_{\text{max}}^2}{\delta_n^2} \quad \text{and} \quad V \geq \frac{w_{\text{max}}^2}{\delta_{n+1}^2}$$
- Pick up the controller gain corresponding to $\delta_n$.
- Apply the controller action to the system.
- Evaluate the new state of the system and go again through all the steps.

All this process can be implemented using Simulink and one example of possible configuration is given on figure(3.2). See section 9.2.1 for more details.
Figure 3.2: Gain scheduled controller example.
3.2 Some test for the new method

There are some interesting cross-effects to explore, some relations whose influence over the results of this proposed scheme is worth to visualize. On one hand we have the 2 proposed representations for the performance of the system ($\beta$ and $\gamma$). On the other, we have the fact that $F$ is present on equation (3.3) only when $D \neq 0$, so the total coupling of the system of LMI is not guaranteed under any circumstance. For this reason we have done a series of tests whose results are going to be shown next.

3.2.1 Importance of $D \neq 0$ over $\epsilon_1$ (considering $\beta$)

For this case the set of equations would be:

Reachable set LMI:

\[
\begin{pmatrix}
QA^T + AQ + F^T B_2^T + B_2 F + \alpha Q & B_1 \\
B_1^T & -\alpha
\end{pmatrix} < 0 \quad (3.5)
\]

Saturation LMI:

\[
\begin{pmatrix}
Q & F^T \\
F & \delta^2 \frac{\mu^2}{\omega_{max}^2}
\end{pmatrix} > 0 \quad (3.6)
\]

Performance LMI :

\[
\begin{pmatrix}
Q & QC^T + F^T D \\
CQ + DF & \delta^2 \frac{\beta^2}{\omega_{max}^2}
\end{pmatrix} < 0 \quad (3.7)
\]

From analyzing the system, we notice a very strong influence of $D$ (when combined with certain values of $\alpha$) over the performance parameter $\beta$, when solving the equations corresponding to $\epsilon_1$; we can see graphically this on figure 3.3. The most surprising fact is that the line that shows the $\beta$ vs. $\alpha$ relationship is completely flat ($\beta = 0$) for $0.12 < D < 0.35$. That means that $\beta = 0$ at any $\alpha$, which is a sign that the system has a trivial solution for some combinations of values of $D$ and $\alpha$. 

29
Figure 3.3: $\beta$ vs $\alpha$ considering different $D$.

This fact can be seen more clearly in an example for a particular value of $\alpha$, as shows figure 3.4.

Figure 3.4: $\beta$ vs $D$ for $\alpha = 2.5$. This time, the trivial solution appears for $0 < D < 1$.
To explain the reason for this trivial solution, we look at the equation that describes the system dynamics (2.2):

\[ z = Cx + Du \]
\[ \beta = 0 = z = Cx + Du \]
\[ C = -DK \]

Therefore, if D is invertible, we end up with

\[ K = -D^{-1}C \] (3.8)

So, this trivial solution shows up when we have some particular structures of C and D that allows this result. But even then it will not show at any \( \alpha \). The reason must be in LMI 3.5 (Reachable Set) which is the one that contains \( \alpha \). As we explained in chapter 1, LMI(3.5) comes from equation (2.3) which imposes stability in the sense of Lyapunov. The first component of this LMI can be explained as:

\[ (A + B_2K + \frac{\alpha}{2}) < 0 \]

According to LMI 3.8, when the trivial solution is possible K is inversely proportional to D. So for certain values of K the relationship with \( \alpha \) can be such that the system is not stable anymore, and then LMI 3.5 forces the system to be out of the trivial solution by imposing a different K. That is why in figure 3.8 we do not have the trivial solution for any value of \( \alpha \).

On the other hand, related to this fact we can say that when we solve only LMI 3.6 and LMI 3.7 (inner ellipsoids) we will get better performances than if we were solving the whole set of 3 LMI instead. The reason is that we are not considering less constraints, so we are more flexible to minimize \( \beta \) or \( \gamma \).

So, as a final summary we can say that the trivial solution is provoked by the coincidence of certain structures in D and C that allow the relation (3.8) to be true, combined with the lack of stability condition, when we do not consider LMI(3.5).
3.2.2 Importance of $D \neq 0$ over $\epsilon_1$ (considering $\gamma$)

For this case, the system of LMI would be the next:

Reachable set LMI:

$$
\begin{pmatrix}
QA^T + AQ + F^TB_2^T + B_2F + \alpha Q & B_1 \\
B_1^T & -\alpha
\end{pmatrix} < 0 
$$

(3.9)

Saturation LMI:

$$
\begin{pmatrix}
Q & F^T \\
F & \delta^2 \frac{w_{\min}}{w_{\max}}
\end{pmatrix} > 0
$$

(3.10)

Performance LMI (with $D \neq 0$):

$$
\begin{pmatrix}
QA^T + AQ + B_2F + F^TB_2^T P & B_1 & QC^T + F^TD^T \\
B_1^T & -\gamma^2 & 0 \\
CQ + DF & 0 & -I
\end{pmatrix} < 0
$$

(3.11)

Here the trivial solution is unlikely to appear given the internal structure of LMI 3.11. For this LMI the coupling of the 3 LMI is guaranteed regardless of the value of $D$. In addition, LMI 3.11 is also related to stability (although not exactly in the same way as LMI 3.9) which was not the case of LMI 3.7.

The stability-related condition on LMI 3.11 is more evident before the change of variable (when we introduce $Q$ and $F$ instead of $P$ and $K$) so we look then at LMI 2.8 (original performance LMI, with $P$ instead of $Q$) we see that the first component imposes:

$$
A^TP + PA + P B_2K + K^TB_2^TP < 0
$$

which turns into:

$$(A + B_2K) < 0$$

And this expression is very similar to inequality (2.5). Somehow less restrictive -since we do not have the positive term introduced by $\alpha$- but anyway it will impose stability
to the system. This matches with the fact that ensuring a bounded $L_2$ gain for all the states (and provided we do not have saturation) is a guarantee of stability.

For this reasons, from now on we will only consider LMI 3.11 as a representative of the performance condition, leaving aside LMI 3.7.

On the next plots (Fig 3.5, Fig 3.6, and Fig 3.7) we show the sensitivity of $\gamma$ with respect to parameters $D$, $\alpha$ and $\delta$ for $\epsilon_1$. As we can see, there is one $\alpha$ optimal and the performance improves as $\delta$ grows.

It is also remarkable that $\gamma$ decreases with $D$. This could be motivated by a progressively greater indirect influence of $\delta$ over this performance LMI (remember that this LMI does not depend explicitly on $\delta$ but through $F$).

That last result shown on Fig 3.7 is what supports our idea of dividing our maximum disturbance to design different controller gains for each level. As we can see, this would allow us to decrease the performance on the smaller levels.

![Figure 3.5: $\gamma$ vs $D$ for $\alpha = 0.21$ and $\delta = 2$](image)

As you can see, when describing the performance in terms of $L_2$ gain, there is no trivial solution.
Figure 3.6: $\gamma$ vs $\alpha$ for $D = 0$ and $\delta = 1$

Figure 3.7: $\gamma$ vs $\delta$ for $D = 0$ and $\alpha = 0.21$
3.3 First scheme of solution: $P_{\text{sequential constant}}$

As we already introduced briefly at section 3.1, $P_{\text{sequential constant}}$ scheme solves the system of LMI sequentially from $\delta_1$ to $\delta_n$ by keeping constant the value of $P$ calculated at $\delta = 1$. We also stated that from now on we will only consider $\gamma$ and not $\beta$ as a parameter representative of the performance of the system. Therefore the performance LMI corresponding to $\beta$ (LMI 3.7) will not be on the considered systems anymore.

As a result of that the succession of steps for solving the system would be:

1. Solve $\epsilon_1$ by finding the values for $P_1$, $\gamma_1$ and $K_1$ from LMI 3.9 (reachable set), LMI 3.10 (non saturation) and LMI 3.11 (performance). All this considering $\delta_1 = 1$.

2. Take $P_n = P_1$ for all $\delta_i$ in $\delta = [\delta_1 \, \delta_2 ... \, \delta_n]$.

3. Solve $\epsilon_n$ for LMI 3.10 (non saturation), LMI 3.11 (performance) and "$\gamma_n \leq \gamma_{n-1}$" by finding the corresponding values for $\gamma_n$ and $K_n$. All this considering $\delta_n > \delta_1$.

Some results:

On this section we are going to show some of the preliminary results for this scheme.

For this simulation, we will consider the described method consistent on solving the 3 LMI proposed and this way get $\epsilon_1$. After that we will solve just LMI (3.11) and (3.10) holding $Q$ constant along all the cases.

The values for the rest of the parameters will be:

$$\delta = 1, \quad \alpha = 2, \quad u^2_{\text{lim}} = 1500, \quad w^2_{\text{max}} = 6.25,$$

And also the system will be:

$$A = \begin{pmatrix} 0 & 1 \\ -5 & -1 \end{pmatrix} \quad B_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad B_2 = \begin{pmatrix} 0 & 1 \end{pmatrix} \quad D = 0.1 \quad C = \begin{pmatrix} 1 & 1 \end{pmatrix}.$$
For all these values, the results are the next

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$K$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ($\epsilon_1$)</td>
<td>(-5.4254, -4.8664)</td>
<td>0.29813</td>
</tr>
<tr>
<td>1.25 ($\epsilon_2$)</td>
<td>(-7.1337, -5.6661)</td>
<td>0.2772</td>
</tr>
<tr>
<td>1.5 ($\epsilon_3$)</td>
<td>(-7.8533, -6.2963)</td>
<td>0.2772</td>
</tr>
<tr>
<td>2.5 ($\epsilon_4$)</td>
<td>(-10.6602, -8.7545)</td>
<td>0.2772</td>
</tr>
<tr>
<td>10 ($\epsilon_5$)</td>
<td>(-21.5639, -18.3039)</td>
<td>0.2772</td>
</tr>
<tr>
<td>30 ($\epsilon_6$)</td>
<td>(-23.4934, -19.9937)</td>
<td>0.2772</td>
</tr>
</tbody>
</table>

And all corresponding to

$$P = Q^{-1} = \begin{pmatrix} 0.34685 & 0.13729 \\ 0.13729 & 0.1020 \end{pmatrix}$$

As we can see, we have made $K$ grow with $\delta$ as expected. However, the problem is that there is not an appreciable improvement in the performance. This is due to the assumptions made in the method, that we will relax later.

Basically what we are doing here is getting a $P$ for our Lyapunov function, which can be seen as a 3D paraboloid. From it we get different "slices" or "level sets" of it (the so called "ellipsoids") at different heights indicated by $\delta$ ($V = x^T P x \leq \frac{w^2}{\gamma^2}$). For each of this heights we calculate a different $K_n$.

However, it would be much better to let the system look for a different $P$ (and therefore, a different Lyapunov function and ellipsoid shape) at each $\delta$. Nevertheless, this is an important change in the assumptions made to get our current set of LMI’s and will be treated on next chapters.

On Figure (3.8) we represent the obtained ellipsoid for $\delta = 1$ as well as the corresponding limits of the actuator. On Figure (3.9), Figure (3.10) and Figure (3.11) we compare this with the results for $\delta = [2 \ 3 \ 5]$ respectively.
Notice that indeed, we never saturate. Moreover, as we get closer to the origin we are more conservative (the ellipsoid is much smaller than the saturation bounds) and we can see that the shape of the ellipsoid (defined by P) does not match that well with the shape defined by the saturation bounds. That is a first indication that we could possibly do a much better job if we consider a different P for each case.

Recall that these inner ellipsoids do not just consider non saturation, but they also consider $V < \frac{w^2}{\delta^2}$. And that is the main reason for not having the saturation limits exactly on the boundary of the ellipsoids.

Figure 3.8: Ellipsoids and sat. limits for $P_{\text{sequential}}^{\text{constant}}$ and $\delta = 1$
Figure 3.9: Ellipsoids and sat. limits for $P_{\text{sequential}}$ and $\delta = 2$, compared to $\delta = 1$

Figure 3.10: Ellipsoids and Sat. limits for $P_{\text{sequential}}$ and $\delta = 4$, compared to $\delta = 1$

Figure 3.11: Ellipsoids and Sat. limits for $P_{\text{sequential}}$ and $\delta = 5$, compared to $\delta = 1$
3.4 Second scheme of solution: $P_{\text{constant}}^{\text{global}}$

As we have seen on the previous section, the results of the implementation of the method indeed improve the performance that we would get without scheduling the gain of the controller, but they are not really spectacular.

In order to force our method to give better results, we try to relax the assumptions imposed in order to allow the minimization over $\gamma$ to get better values. One important solution to consider is allowing $P$ to change. However, before considering that, we are going to explore another possibility given by the *Global Constant P scheme*.

The *Global Constant P scheme* solves all the equations that the *Sequential Constant P scheme* solved separately for each $\delta$ at the same time. This means solving LMI 3.9 (reachable set), LMI 3.11 (performance) and LMI 3.10 (non saturation) for $\epsilon_1$ AND also LMI 3.11, LMI 3.10 and ($\gamma_n \leq \gamma_{n-1}$) for all other values of $\delta$ at the same time.

For a more graphical comparation we recall Fig 3.1 on section 3.1:

![Diagram showing Sequential and Global Constant P schemes](image-url)

**Figure 3.12:** Graphics of $P_{\text{constant}}^{\text{sequential}}$ and $P_{\text{constant}}^{\text{global}}$ schemes
This new way of solving the system implies that the calculated P will not be based only on $\epsilon_1$ and then applied to all the rest, but will be calculated considering all the situations at the same time.

The system would be a more complicated to solve, basically because if we are considering $n$ values for $\delta$ we would have to solve a system of $2n + 1$ LMI at the same time. But the P resulting would be calculated at all different situations included in each $\delta$ instead of being determined just by $\delta = 1$ as in the previous method.

That new assumption regarding how does $P$ evolve along $\delta$ allows us to speak about a $P^{global}$ scheme, and should be able to give better results than the previous method $P^{sequential}$.

We can even go one step further and weight the importance of each ellipsoid (each $\delta$ or better, its corresponding $\gamma$) so we can solve globally $P$ but focusing specially in some values. This way the performance of the method will be even better for a particular area of interest.

Therefore, instead of minimizing:

$$\min \sum_{i=1}^{n} \gamma_i$$

We will minimize a weighted version of it using a different weight ($w_i$) for each $\gamma$:

$$\min \sum_{i=1}^{n} w_i \gamma_i$$

Next we show some values obtained from different weight combinations and we compare them with the old scheme $P^{sequential}$. In all the cases we consider a set of 3 different $\delta$. The system is exactly the same as described in section 3.3.

As we can see on table 3.2, as we focus more on the first values of $\delta$ ($w_3$) or the last ($w_2$), we improve the results on this range over the ones corresponding to the same weight for all $\delta$ ($w_1$).
Table 3.2: First example of solution for the $P_{\text{constant}}^{\text{global}}$ scheme, with different sets of weights

<table>
<thead>
<tr>
<th>weights</th>
<th>$\delta$</th>
<th>$k$</th>
<th>$\gamma$</th>
<th>$P$</th>
</tr>
</thead>
</table>
| $w_1=(1 1 1)$ | 1 ($\epsilon_1$) | (-3.1392 -4.4806) | 0.3215 | \[
\begin{pmatrix}
0.2864 & 0.1402 \\
0.1402 & 0.1108
\end{pmatrix}\] |
| | 1.5 | (-8.3444 -7.0067) | 0.2294 |
| | 5 | (-17.1224 -14.8863) | 0.2294 |
| $w_2=(0.5 0.5 3)$ | 1 ($\epsilon_1$) | (-1.9924 -4.0452) | 0.3616 | \[
\begin{pmatrix}
0.2668 & 0.1403 \\
0.1403 & 0.1137
\end{pmatrix}\] |
| | 1.5 | (-8.6323 -7.3616) | 0.2131 |
| | 5 | (-17.3799 -15.3198) | 0.2131 |
| $w_3=(5 2 0.5)$ | 1 ($\epsilon_1$) | (-4.3379 -4.7651) | 0.3027 | \[
\begin{pmatrix}
0.3136 & 0.1387 \\
0.1387 & 0.1063
\end{pmatrix}\] |
| | 1.5 | (-8.0346 -6.6093) | 2.522 |
| | 5 | (-16.7547 -14.3346) | 2.522 |

Another detail to point out is the fact that for $w_3$ we do not improve $\gamma$ as $\delta$ grows. This is because for this simulation we didn’t consider the condition

$$\gamma_n \leq \gamma_{n-1}$$

This proves that this condition is not redundant (as it might look like if we are always minimizing $\gamma$ over smaller level sets of the same ellipsoid) and therefore it is important, specially with the Global scheme.

Also notice that, even though at the third case ($w_3$) we are focusing mainly on $\epsilon_1$, the corresponding $\gamma_{1}^{\text{global}} = 0.3027$ is not as good as the one that we got with $P_{\text{constant}}^{\text{sequential}}$, which was $\gamma_{1}^{\text{sequential}} = 0.29813$ (see table on section 3.3). This is due to the fact that for $P_{\text{constant}}^{\text{sequential}}$ we calculate P only thinking on $\epsilon_1$ and in the third case of $P_{\text{constant}}^{\text{global}}$, although weighted, we also consider the rest of the values of $\delta$. Therefore we can approach the results of $P_{\text{constant}}^{\text{sequential}}$ by weighting more and more $\delta = 1$ and less the rest of $\delta$.

But if we really want to compare both methods, we should look at Figure 3.13, where we can see the corresponding plot for the sequential method and three plots.
corresponding for weights $w_1 = (1 \ 1 \ 1)$, $w_2 = (1 \ 1 \ 20)$ and $w_3 = (20 \ 1 \ 1)$. In all cases the system is the same of section (3.3) and we only calculate and represent values for $\delta=(1 \ \!1.5 \ 5)$.

As we can see on Figure 3.13, the global results are much better, specially for the inner ellipsoids, where they provide a much lower performance.

![Figure 3.13: $P^{\text{sequential constant}}$ (higher plot) and different $P^{\text{global constant}}$.](image)
3.5 Limitations of both $P_{\text{constant}}$ schemes

As we have seen, these 2 methods, and the different controllers that they provide, improve the performance obtained by solving just one set of equations and getting one single controller. However, they have the important limitation of considering a single Lyapunov functions (a single $P$) for all the different $\delta$.

The next logical step, as we have already pointed out before, would be getting different $P$ for each of the values of $\delta$. To do this one possibility would be solving the initial set of 3 LMI’s (reachable set, saturation and performance) for each value of $\delta$. The resulting performance should be really better. This resolution scheme is called low gain resolution scheme, and has the inconvenience that it is not valid as a solution because of different incompatibilities with our basic specifications, as you will see below.

The main issue with this new method is that it does not have any control about the change of shape in the ellipsoids $\left(\frac{\partial P}{\partial \delta}\right)$, whereas we have assumed $P$ to be constant during the derivation of the expressions discussed in Chapter 2.

It also does not take into account the transition between one ellipsoids and the next, and we do really need to look to the whole domain as a continuous set of states with a smooth transition between this different zones if we do not want to compromise the stability of the system.

On next Figure (3.14) we can see graphically how different are the obtained values for each method (sequential, global and the hypothetical low gain resolution method). It is clear that this hypothetical method has better results, but as we said, it is not really valid as a solution. However, it will be considered as the starting point for a new scheme of resolution.
Is it really so different in essence from the previous methods? If we consider $P_{\text{constant}}^{\text{global}}$ scheme we could say that ideally, by playing and making emphasis with different weight sets, we could approach the solution if this low gain resolution method for any particular value of $\delta$. But this really does not happen because, although we can modify the results by using the weights, the global case will always have a minimal connection between the different ellipsoids. Therefore, we cannot really match the values that we get when we solve separately LMI (3.9), (3.10) and (3.4) for each value of $\delta$.

Nevertheless, we can keep this idea of getting a different $P$ for each $\delta$ (which is the main idea of this hypothetical method) to build more feasible methods. We will have to adapt our LMI to allow this, though. But the results will be much better than those given by both $P_{\text{constant}}^{\text{sequential}}$ and $P_{\text{constant}}^{\text{global}}$, as we will see on the next section.
Chapter 4

Nested Sets: \( \dot{r} \leq 0 \)

Continuing with the idea discussed on the last section of the previous chapter, we are going to go one step further on the implementation of the method by allowing \( P \) to change from one value of \( \delta \) to the next. To do so we will solve the same LMI system at each \( \delta \), but with a different approach to allow that change in \( P \). The new assumption will allow \( P \) to change, but in such a way that the resulting ellipsoids have to be nested.

Since from now on \( P \) is going to change and we will consider its time derivative, we will need to describe \( P \) as a function of some parameters whose temporal evolutions are known. In other words: we have to ensure that \( \frac{dP}{dt} \) is well defined.

On section 4.1 we will explain how are we going to allow \( P \) to become a variable, whereas in sections 4.2.1, 4.2.2 and 4.2.3 we will provide a parametrization of \( P \) so the concept of \( \dot{P} \) is defined; as well as some conclusions from this new parametrization.

As a result of this new continuous evolution of our control parameters, the control conditions themselves (LMI) are going to have a continuous evolution that also has to be parameterized (see section 4.3).

Such a situation has already been described on the bibliography [6], [9].
4.1 New assumptions regarding $\dot{P}$

We can modify some assumptions made during the derivation of the initial Performance LMI without changing it significantly. Particularly, if we go back to the derivation of LMI(2.8) on section 2.0.3:

$$0 > \dot{x}^T P x + x^T P \dot{x} + x^T \dot{P} x - \gamma^2 w^T w + y^T y$$

From here we assumed $P = constant$ on the right hand side of the inequality during the first steps of the derivation. However, once we have solved the inequality, we realize that the final result would still hold even if we had $x^T \dot{P} x < 0$.

Therefore, since the inequality still holds, we can say that when we are solving LMI(2.8) we are also considering $x^T \dot{P} x \leq 0$. The only change introduced by considering a non constant $P$ is that we are introducing some conservatism. The reason is because if now we have $x^T \dot{P} x \leq 0$, the rest of the terms of the right side did not necessarily need to be negative anymore.

In other words, by introducing $x^T \dot{P} x \leq 0$ we will relax the assumption $P = constant$, but at the same time we would be more conservative when solving LMI(2.8). That is to say: we will change our conservatism in a search for better results.

Now we have a description about what $\dot{P}$ can be, but we still have to define $P$ as a function in order to be able to work with its derivative. On the following sections we will focus on that.
4.2 Parametrization for $P$

Next we are going to describe the formal steps needed to consider $P \neq \text{constant}$. Since the time dependence of $P$ is not direct but through some intermediate parameters, we will have to define all of them for having a complete description.

Let the expression of $\dot{P}$ be:

$$\frac{dP}{dt} = \frac{\delta P \delta r}{\delta r \delta t} \quad (4.1)$$

First, we consider $\dot{P} \leq 0$ and we will divide the description of $P$ in three different sections. First we will parameterize $P$ based on a $\delta$-related parameter $r$. Then we will give a full description of the sensibility of $P$ with respect to $r$. And finally we will describe $\frac{\delta P}{\delta t}$.

4.2.1 $P$ as a Spline Function over $r$

So far, we only have considered that for different values of $\delta$, $Q$ and $P$ are going to have different expressions. For having a continuous parametrization of this variables we are going to express them as a continuous function based on $\delta$ (and not just some discrete values) or, even better, some new $\delta$-related parameter: $r$.

$$r_n = \delta_n^2 \frac{u_{2im}^2}{w_{\text{max}}^2}$$

Once we have that parametrization we still will have to define the time response for $r$, but we will take care of this point during section 4.2.2.

To characterize such a continuous function based on this set of discrete values ($r_1$, $r_2$, ..., $r_n$), we will use the concept of spline functions which is defined as follows:

For

$$r_1 < r_2 < r_3$$
And the corresponding

\[ Q_1, \ Q_2, \ Q_3 \]

We can define a spline function \( Q(r) \):

\[
Q_S(r) = \begin{cases} 
Q_1 + \frac{Q_2-Q_1}{r_2-r_1}(r-r_1) & r_1 < r < r_2 \\
Q_2 + \frac{Q_3-Q_2}{r_3-r_2}(r-r_2) & r_2 < r < r_3
\end{cases}
\]

This means that \( Q \) will be described along \( r \) space as a linear interpolation between the values obtained for the ellipsoids that divide the state space \( (Q_1, \ Q_2, \ ... \ Q_n) \) smoothed by a posterior "average" to avoid problems with the transition between one inter ellipsoidal area and the next.

In summary, we will solve for the discrete set of different values of \( \delta \) and get a set of ellipsoids with their corresponding values of \( \gamma \) and \( K \). From that, using the splines description, we will elaborate a continuous description of \( Q \) and \( K \) along all possible \( r \) (or \( \delta \)).

\subsection{4.2.2 \( P(r) \) sensitivity}

Going back to the concept of parameterizing \( P \) over \( r \); we can say that it implies having several variables also related to \( r \):

\[ \delta(r), \ P(r), \ and \ F(r) \]

Remember that all the ellipsoids \( (\epsilon) \) that we are going to work with accomplish this condition (implicit by accomplishing the Non-Saturation LMI (2.14)):

\[ \epsilon_n = \left\{ x : V(x) = x^T P x \leq \frac{u_{\max}^2}{\delta_n^2} = \frac{u_{\lim}^2}{r} \right\} \]

\[ (4.3) \]
Then, according to this we can represent shortly and completely the ellipsoids by indicating from which $P(r)$ do they come from and at what $\delta_n$ (or better $r_n$) were they evaluated (recall $P = Q^{-1}$):

$$\epsilon(Q^{-1}(r), r_n)$$

And if we want to get a nested series of ellipsoids\(^1\) we want them to be such that, for the same scale parameter $r_1$ (provided $r_2 > r_1$):

$$\epsilon(Q^{-1}(r_1), r_1) \supset \epsilon(Q^{-1}(r_2), r_1)$$

That is to say: The ellipsoid corresponding to $r_1$ includes the ellipsoid corresponding to $r_2$ when both are evaluated at the level set $\frac{\delta^2}{\delta t^2}$-disturbance level-. Equation (4.3) tells us that this is only possible when $P_2 > P_1$. Therefore, the only way to get this sensitivity to $r$ parameter is to impose:

$$\frac{\delta P}{\delta r} \geq 0 \quad (4.4)$$

And this way we have the first of the terms that we needed for completing expression (4.1).

Equation (4.4) shows a substantial change in the situation. From the $P = constant$ assumption of chapter 1, in which ellipsoids had the same shape and decreased as we considered different disturbance levels, we have gone to a new situation in which ellipsoids vary and decrease in size even when represented at the same disturbance level. That is to say: they have to be nested.

### 4.2.3 Description of $\frac{\delta r(t)}{\delta t}$

As happened before, when we implement this results into a real controller, we will have to choose the ellipsoid that best fits with the state of the system for each time "$t$". The criteria will be choosing the smallest that contains the state.

\(^1\)As we had on the previous versions of the system in order to get bigger controllers as we get closer to the origin of the space state
This means that for each time "t" we will always choose an \( r_n \) from our set of discrete \( r (r_1... r_k) \) described by this expression:

\[
  r_n = \max \left\{ r \in [r_1, r_2...r_k] : x^T(t)P(r)x(t) \leq \frac{u_{lim}^2}{r} = \frac{w_{max}^2}{\delta^2} \right\}
\]

(4.5)

Once we have picked up our \( r_n \) from our discrete set, we focus on the interval \([r_n, r_{n+1}]\) to get a more accurate value \((r(t))\).

\[
  r^*(t) = \left\{ r \in [r_n, r_{n+1}] : x^T(t)P(r)x(t) = \frac{u_{lim}^2}{r} = \frac{w_{max}^2}{\delta^2} \right\}
\]

(4.6)

And this way we have a continuous parameter description between \( r_1 \) and \( r_n \).

Now we focus on the last term of expression (4.1) which is \( \frac{\delta r}{\delta t} \). By looking at the expression and considering the previous result shown in expression (4.4); one possible result is to enforce

\[
  \frac{\delta r(t)}{\delta t} \leq 0
\]

(4.7)

But before accepting that fact, we have to express \( r(t) \) in such a way that its derivative is well defined (the same way as we are doing with \( P \)). Instead of working with \( r(t) \), which may not be differentiable (according to the way defined on equation (4.6) ) we define a new function \( \hat{r}(t) \) such that:

\[
  \hat{r}(t) = \frac{1}{\tau} \int_{t-\tau}^{t} r(h)dh
\]

Since \( r(t) \) is continuous, \( r(t) - \hat{r}(t) \) can be made as small as we want by choosing a small enough \( \tau \). So we can consider \( \hat{r}(t) \) as a smooth version of \( r(t) \) and check easily that everything that holds for \( r(t) \) holds for \( \hat{r}(t) \).

That is what we will do on the next calculations. Specially thinking on \( \frac{\delta r(t)}{\delta t} \) where, since now we have a differentiable definition for \( r(t) \), we can justify the conclusion.
shown in expression (4.7) by:

\[
\begin{align*}
\text{error} & = |r(t) - \hat{r}(t)| \\
r(t) - \hat{r}(t) & = r(t) - \frac{1}{\tau} \int_{t-\tau}^{t} r(h)dh \\
r(t) - \hat{r}(t) & = \frac{1}{\tau} \int_{t-\tau}^{t} r(t)dh - \frac{1}{\tau} \int_{t-\tau}^{t} r(h)dh \\
r(t) - \hat{r}(t) & = \frac{1}{\tau} \int_{t-\tau}^{t} (r(t) - r(h))dh \\
|r(t) - \hat{r}(t)| & \leq \frac{1}{\tau} \int_{t-\tau}^{t} |r(t) - r(h)|dh
\end{align*}
\]

Where, since \( r(t) \) is continuous, we can say by definition that:

\[
|\tau| < \lambda \implies |r(t) - r(t - \tau)| < \epsilon
\]

And therefore, we can use this fact to continue the derivation:

\[
\begin{align*}
|r(t) - \hat{r}(t)| & < \frac{1}{\tau} \int_{t-\tau}^{t} \epsilon dh \\
|r(t) - \hat{r}(t)| & < \epsilon
\end{align*}
\]

Since we are imposing a small \( \tau \) (we just said that \( \tau < \lambda \) where \( \lambda \) is a small positive number) we say that \( r(t) \simeq \hat{r}(t) \) and hence we can approach:

\[
\frac{\delta \hat{r}}{\delta t} \simeq [r(t) - r(t - \tau)] \frac{1}{\tau} \frac{\delta r}{\delta t} \leq 0
\]

So we end up considering:

\[
\frac{\delta \hat{r}}{\delta t} = \frac{\delta r}{\delta t} \leq 0
\]

Now we have mathematically justified all conclusions regarding \( r \) by finding a continuous differentiable approach that justifies them. However, we still have to find and interpretation for \( \frac{\delta r}{\delta t} \leq 0 \) so we can complete the descriptions of all the parts of \( \frac{dP}{dt} \) as presented on expression 4.1:

\[
\frac{dP}{dt} = \frac{\delta P \delta r}{\delta r \delta t} \leq 0
\]
Considering that $x(0) = 0$ \(^2\) we can say that when we implement our results in a controller, we will start with the highest possible $r$ (the smallest ellipsoid). As time goes on, disturbance may pull the system out of the origin, eventually out of the smallest ellipsoid that we were using to a bigger one. At this point all $r$ would have to decrease their values to choose that bigger ellipsoid. The important limitation brought by $\frac{\delta r}{\delta t} \leq 0$ is that even in the case where the controller brings the system back to the origin, we will not be able to recover the smallest ellipsoid.

That is to say:

*We will start with the smallest ellipsoid and as time goes on we can only go to bigger ellipsoids driven by the disturbances.*

And this is represented thanks to expression 4.6 that allows us to define finally $r(t)$ as:

$$r(t) = \max (r^*(t))$$  \hspace{1cm} (4.8)

Although that is an important limitation, it is not a critical issue when we apply this controller system to stabilize and structure exposed to an earthquake (which is one of the main particular applications that can be considered for this theory). According to the description of an earthquake the worst disturbance happens during the first instants, and that is when our system will supply the greatest actuator action.

The reaction of our system to such a disturbance would be using the more aggressive controller at the beginning \(^3\) and in case we cannot keep the state of the system inside the smallest ellipsoid we would have to go to the next bigger ellipsoid (and use a not so aggressive controller). Once the worst part of the disturbance is gone, we would stay in one of the bigger ellipsoids and therefore we would drive back the state of the system to the origin using the corresponding “soft” controller.

---

\(^2\) Which is what we assumed for LMI 2.8

\(^3\) When the maximal disturbance comes
4.3 New considerations on the set of LMI

As we said at the beginning of this chapter, from now on we are working with the same conditions but under slightly different assumptions (we are now considering non-constant values for $P$). The main issue is that we are considering the spline description around $r$, which means that we are in a continuous succession of values rather than a set of discrete values corresponding to each $\delta$.

This new description has to be reflected in our set of LMI which now is also going to become continuous. To re-write all three LMI we have to reconsider first all of their variables.

Let’s consider our set of values for $r$ expressed as $[r_{\text{min}}... r_{\text{max}}]$ where

$$r_{\text{min}} = \delta_{\text{min}}^2 \frac{u_{\text{lim}}^2}{u_{\text{max}}^2}$$

For the continuous values of $r$ in between $r_n$ and $r_{n+1}$ we can say that, using the spline description similarly as we had in equation (4.2), for $r \in [r_n, r_{n+1}]$:

$$\gamma^2(r) = \gamma_n^2 + \frac{\Delta \gamma_n^2}{\Delta r_n} (r - r_n)$$

$$F(r) = F_n + \frac{\Delta F_n}{\Delta r_n} (r - r_n)$$

For $Q$ we have already seen that it requires two steps, since we have to take into account its derivative, therefore we have:

$$Q(r) = \begin{cases} Q_n + \frac{Q_{n+1} - Q_n}{r_{n+1} - r_n} (r - r_n) \\ \frac{1}{r_{n+1} - \frac{r_r}{2}} \int_{r_r - \frac{r_r}{2}}^{r_r + \frac{r_r}{2}} Q_S(h) dh \end{cases}$$

(4.9)

From here, we can get $K$ as:

$$K(r) = F(r)Q^{-1}(r)$$
Now that we have presented the expressions for $F(r)$, $Q(r)$, $\gamma(r)$ and $K(r)$ we can use them for the calculations necessary to get our system of LMI. As a previous step we can try to simplify them by defining parameter $\lambda$ which is going to be:

$$\lambda = \frac{r - r_n}{r_{n+1} - r_n} = \frac{r - r_n}{\Delta r} > 0$$

And $r$ is a real value considered between $r_{n+1}$ and $r_n$ obtained as we did on expressions (4.5), (4.6) and finally (4.8).

Once we have this parameter defined, we can work out the description of a generic product between 2 spline functions. This expression will be used to re-write our LMI conditions considering the new parameterizations for $F(r)$, $Q(r)$, $\gamma(r)$ and $K(r)$.

$$T(r) = T_n + (T_{n+1} - T_n) \lambda = T_n + \Delta T_n \lambda$$

$$R(r) = R_n + (R_{n+1} - R_n) \lambda = R_n + \Delta R_n \lambda$$

$$T(r)R(r) = (T_n + \Delta T_n \lambda)(R_n + \Delta R_n \lambda)$$

$$= T_n R_n + \lambda T_n (R_{n+1} - R_n) + \lambda (T_n - T_n) R_n +$$

$$+ \lambda^2 (T_{n+1} - T_n)(R_{n+1} - R_n)$$

We add the term $\lambda^2(T_n R_n - T_n R_n)$ to the right hand side to complete the square and get:

$$T(r)R(r) = (1 - \lambda^2) T_n R_n + \lambda^2 (T_{n+1} R_{n+1}) +$$

$$+ \lambda (1 - \lambda) [(T_{n+1} - T_n) R_n + T_n (R_{n+1} - R_n)]$$

We substitute the expression $(1 - \lambda^2) = (1 - \lambda)^2 - 2\lambda^2 + 2\lambda$ and then:

$$T(r)R(r) = (1 - \lambda)^2 T_n R_n + \lambda^2 (T_{n+1} R_{n+1}) +$$

$$+ \lambda (1 - \lambda) [(T_{n+1} - T_n) R_n + T_n (R_{n+1} - R_n) + 2T_n R_n]$$
And finally, we can use this generic expression to re-write the spline function description for $Q(r)$, $F(r)$, $\gamma^2$ and $r$, by representing them as the product of themselves $(T(r))$ by 1 $(R(r))$:

$$
\begin{align*}
T(r) = Q(r) & \quad Q(r) = (1 - \lambda)^2 Q_n + \lambda^2 Q_{n+1} + \lambda (1 - \lambda)[Q_n + Q_{n+1}] \\
R(r) = 1 & \\
T(r) = F(r) & \quad F(r) = (1 - \lambda)^2 F_n + \lambda^2 F_{n+1} + \lambda (1 - \lambda)[F_n + F_{n+1}] \\
R(r) = 1 & \\
T(r) = \gamma^2(r) & \quad \gamma^2(r) = (1 - \lambda)^2 \gamma_n^2 + \lambda^2 \gamma_{n+1}^2 + \lambda (1 - \lambda)[\gamma_n^2 + \gamma_{n+1}^2] \\
R(r) = 1 & \\
T(r) = r & \quad r = (1 - \lambda)^2 r_n + \lambda^2 r_{n+1} + \lambda (1 - \lambda)[r_n + r_{n+1}] \\
R(r) = 1 &
\end{align*}
$$

Getting a new expression for the spline description of these parameters. Notice that now we have $\lambda^2$ terms whereas before introducing this new structure; $Q(r)$, $F(r)$, $\gamma^2(r)$ and $r$ depended just on $\lambda$.

On the next sections we will see how this new structure on the spline description is more convenient. Specially when describing as a spline the LMI corresponding to our main specifications.

### 4.3.1 Non-Saturation

As we explained before, now we will re-write our 3 basic LMI (Non-Saturation, Performance and Reachable set) incorporating the spline description of the main variables. As a result of that, the LMI themselves will acquire the spline description based on the generic structure represented by (4.10).
In the case of the Non-Saturation condition, the whole LMI will be re-written as:

\[
M(r) = \begin{pmatrix} Q(r) & F^T(r) \\ F(r) & r \end{pmatrix} = \begin{pmatrix} Q_n + \lambda_n \Delta Q_n & F^T_n + \lambda_n \Delta F^T_n \\ F_n + \lambda_n \Delta F_n & r_n + \lambda_n \Delta r_n \end{pmatrix} > 0
\]

\[
M(r) = M_n + \lambda \Delta M_n > 0
\]

Where:

\[
M_n = \begin{pmatrix} Q_n & F^T_n \\ F_n & r_n \end{pmatrix} > 0 
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad 1 \leq k \leq n
\]

\[
\Delta M_n = \begin{pmatrix} Q_{n+1} - Q_n & F^T_{n+1} - F^T_n \\ F_{n+1} - F_n & r_{n+1} - r_n \end{pmatrix} = M_{n+1} - M_n
\]

Since \( M(r) > 0 \) is a spline function, by applying the generic expression of the \( R(r)T(r) \) product to it we end up with:

\[
M(r) = (1 - \lambda)^2 M_n + \lambda^2 M_{n+1} + \lambda(1 - \lambda)[M_n + M_{n+1}] > 0
\]

On this expression we have three positive coefficients:

\[
(1 - \lambda)^2, \lambda^2 \text{ and } \lambda(1 - \lambda)
\]

In order to have \( M(r) > 0 \) we will need all the three variables to be positive too, that is to say:

\[
M_n > 0, \quad M_{n+1} > 0 \quad \text{and} \quad [M_n + M_{n+1}] > 0.
\]

And that is equivalent to say \( M_n > 0 \)

\[
M_n = \begin{pmatrix} Q_n & F^T_n \\ F_n & r_n \end{pmatrix} > 0 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (4.10)
\]

Therefore, the fact of considering \( M(r) > 0 \) is equivalent to consider \( M_n > 0 \). In other words: we do not have to worry about all the continuous values of \( r \) but only about the discrete set of values that we get from the proposed \( \delta \).
4.3.2 Performance

For this condition, the proceedings are conceptually the same. But the fact of having products of spline functions turns it a little bit more complicated. Particularly if we also consider \( A \) as a nonconstant matrix

\[
A(r) = A_n + \lambda \Delta A_n
\]

For this case, based on LMI (2.12) but considering now the spline function description for the variables, the Performance LMI takes this structure:

\[
J(r) = \begin{pmatrix}
A(r)Q(r) + Q(r)A^T(r) + B_2F(r) + F^T(r)B_2^T & B_1^T & Q(r)C_1^T + F^T(r)D^T \\
B_1 & -\gamma^2(r) & 0 \\
C_1Q(r) + DF(r) & 0 & -I \\
\end{pmatrix} < 0
\]

Which, after applying the generic expression for the \( R(r)T(r) \) product for \( Q(r), F(r), \gamma(r) \) and \( A(r) \), and grouping terms, turns into:

\[
J(r) = (1 - \lambda)^2 J_n + \lambda^2 J_{n+1} + \lambda(1 - \lambda)(J_n + J_{n+1}) < 0 \tag{4.11}
\]

At this point we decide to just consider \( A(r) \) as a constant in order to simplify the matrix that we are working with. As a result of that we are in the same situation as we were with the non-saturation LMI.

Continuing with expression (4.11), we will also have three positive coefficients:

\[
(1 - \lambda)^2, \lambda^2 \text{ and } \lambda(1 - \lambda)
\]

And this time, in order to have \( J(r) < 0 \) we will need all the three variables to be negative, that is to say:
\[ J_n < 0, \ J_{n+1} < 0 \quad \text{and} \quad [J_n + J_{n+1}] < 0. \]

Which is equivalent to say \( J_n < 0. \)

\[
J_n = \begin{pmatrix}
AQ_n + Q_n A^T + B_2 F_n + F_n^T B_2^T & B_1^T & Q_n C_1^T + F_n^T D^T \\
B_1 & -\gamma_n^2 & 0 \\
C_1 Q_n + D F_n & 0 & -I
\end{pmatrix} < 0 \quad (4.12)
\]

Which means that, again, when considering \( J(r) < 0 \) we just have to focus on the discrete values \( J_n < 0. \)

### 4.3.3 Reachable set

Actually, this is the only equation that does not change by the fact of introducing the spline function description. The reason is that it only holds for \( r_m \) (minimal value for \( r \), the most exterior ellipsoid) so we do not really care here about what happens when we deal with the values of \( r \) associated to the inner system states.

According to this, the LMI will be only referred to this minimum value of \( r = r_{\text{min}} \) associated to the most exterior ellipsoid:

\[
\begin{pmatrix}
AQ(r_m) + Q(r_m) A^T + \alpha Q(r_m) + B_2 F(r_m) + F^T(r_m) B_2^2 & B_1 \\
B_1^T & -\alpha
\end{pmatrix} < 0 \quad (4.13)
\]

And this LMI is related directly to what we got in LMI (2.9), also referred uniquely to this exterior ellipsoid.
4.4 Schemes of solution and practical implementation

The new description of our variables as a spline functions over parameter $r$ has brought us to a new group of LMI similar to what we were solving before, but under more relaxed assumptions. Basically, what we will consider from now on will be:

- Reachable set $\rightarrow$ LMI (4.13)
  $\begin{pmatrix}
    AQ(r_m) + Q(r_m)A^T + \alpha Q(r_m) + B_2F(r_m) + F^T(r_m)B_2^2 & B_1 \\
    B_1^T & -\alpha
  \end{pmatrix} < 0$

- Non-saturation: $M(r) < 0 \rightarrow M_n > 0$ LMI (4.10)
  $M(r_n) = \begin{pmatrix}
    Q(r_n) & F^T(r_n) \\
    F(r_n) & r_n
  \end{pmatrix} > 0$

- Optimal performance: $J(r) < 0 \rightarrow J_n < 0$ LMI (4.12)
  $J(r_n) = \begin{pmatrix}
    AQ(r_n) + Q(r_n)A^T + B_2F(r_n) + F(r_n)^T(r)B_2^T & B_1^T & Q(r_n)C_1^T + F(r_n)^T D^T \\
    B_1 & -\gamma(r_n)^2 & 0 \\
    C_1Q(r_n) + DF(r_n) & 0 & -I
  \end{pmatrix} < 0$

- Nested ellipsoids:
  $Q(r_n) < Q(r_{n-1})$

- Improved performance:
  $\gamma(r_n) \leq \gamma(r_{n-1})$
As we had before, we can engage the solution of this system of LMI by choosing a sequential scheme (solving by increasing order the system for each value of \( r \)) or we can solve globally the system considering all LMI and all the values or \( r \) at the same time.

The scheme of resolution for this two different strategies is detailed below, and the comparison of their results and the results for the two previous \( P = \text{constant} \) cases will be shown on next Chapter 5.

- \( P_{\text{sequential variable}} \)

  1. Solve \( \epsilon_1 \) (for \( r=1 \)): Reachable set, non-saturation and optimal performance.
      
      With this we get: \( P_1, \gamma_1 \) and \( K_1 \)

  2. Solve \( \epsilon_n \) (for \( r=r_n \)): Non-saturation, Optimal performance, Nested ellipsoids and Improved performance.
      
      With this we get: \( P_n, \gamma_n \) and \( K_n \)

- \( P_{\text{global variable}} \)

  1. Solve \( \epsilon_1 \) (for \( r=1 \)): Reachable set, non-saturation and optimal performance.

      Solve also, \( \epsilon_n \) (for \( r=r_n \)): Non-saturation, Optimal performance, Nested ellipsoids and Improved performance.

      With this we get: \( P_1, \gamma_1, K_1 \) AND \( P_n, \gamma_n \) and \( K_n \)

      If we choose to solve globally the system of equations, our minimization will be based on a weighted addition of \( \gamma \)'s rather than an individual minimization:

      \[
      Min \sum_{i=1}^{n} \beta_i \gamma^2
      \]

      Based on this two possibilities and the fact of allowing or not \( Q \) to vary from one value of \( r \) to the next, we can group our strategies for solving our study in four basic
schemes: \( P_{\text{constant}}^{\text{sequential}}, P_{\text{constant}}^{\text{global}}, P_{\text{variable}}^{\text{sequential}} \) and \( P_{\text{variable}}^{\text{global}} \). We have listed them from more to less conservative, and therefore we expect the results to improve as we become less conservative (as we will see on next chapter).

### 4.4.1 Practical implementation of the controller

The practical implementation of this results is similar to what we described on section 3.1.1. The only difference is that now we are considering a continuous \( r(t) \) instead of just a set of values associated to \( [\delta_1, \delta_2, \ldots, \delta_n] \).

To get \( r(t) \) first we have to use expression (4.5) to get the "surrounding calculated ellipsoids" (as before):

\[
r_n = \max \{ r \in [r_1, r_2, \ldots, r_k] : x^T(t)P(r)x(t) \leq \frac{u_{\text{lim}}^2}{r} = \frac{w_{\max}^2}{\delta^2} \}
\]

One we have bounded \( r(t) \) with \( r_n \) and \( r_{n+1} \) we perform an extra step by solving expression (4.6) and (4.8):

\[
r(t) = \max \{ r^*(t) \} \quad r^*(t) = \left\{ r \in [r_n, r_{n+1}] : x^T(t)P(r)x(t) = \frac{u_{\text{lim}}^2}{r} = \frac{w_{\max}^2}{\delta^2} \right\}
\]

And this way we get the continuous value in between them, as we described on section 4.2.3. The difference is that on the first step (expression 4.5) we were choosing among our set of discrete \( r_n \). However, on the second step (expressions 4.6) we are solving an iteration starting at \( r_n \) and with limit at \( r_{n+1} \). Before reaching \( r(t) = r_{n+1} \) we should find a value that accomplished the equality.

As always, the propose is evaluating the state of the system for each time "t" and then determine the smallest ellipsoid that includes it. The only difference is this "extra step" that allows us to get closer to this state. This is thanks to the spline description of the parameters. Once we have \( r(t), r_n \) and \( r_{n+1} \) we use this values to get \( Q(r) \) and \( K(r) \) by the corresponding spline descriptions, and implement \( K(r) \) on the controller.

On section 9.2 we have all the details regarding the practical implementation.
4.4.2 Quick facts about \( P^{\text{sequential}}_{\text{variable}} \) method

There is a special interest on this method among all four methods proposed. The reason is that it is supposed to be the best method in terms of the relationship "computational cost vs. performance achievements".

The main reason is the fact of allowing \( P \) to change, which makes the method more powerful than the constant methods. Besides, the fact of being sequential also implies a more simple set of equations (not as many equations to solve at the same time as in global methods).

According to this criteria, theory says that we can organize by decreasing performance the 4 methods as \( P^{\text{sequential}}_{\text{constant}} > P^{\text{global}}_{\text{constant}} > P^{\text{sequential}}_{\text{variable}} > P^{\text{global}}_{\text{variable}} \). However, on next chapter we will see how particular situations may change this order of performances.

We will find one example of this situation when comparing the results for all four methods (see next chapter). Regarding the preliminary tests over the \( P^{\text{sequential}}_{\text{variable}} \) method, we can say that it does improve with respect to \( P^{\text{sequential}}_{\text{constant}} \) but it is not so clear that it improves \( P^{\text{global}}_{\text{constant}} \), as expected.

There are several factors that may vary the results from this methods. On the next chapters we will describe the most important of them with detail, and we will show that they do not affect all the methods the same way. Here we introduce some of these factors:

- The presence of \( D \neq 0 \) on the state space description improves the coupling of the LMI in the system and allows us to have better results.

- The size of the controller, represented by \( |u_{\text{lim}}| \). As expected, a higher saturation limit allows us to get better performances.
• The parameter $\alpha$ is also a factor to consider. On Fig (4.1) and Fig (4.2) we see that, for $P_{\text{variable}}^{\text{sequential}}$ method we have an optimal range of values in terms of performance ($\alpha \in [1, 5]$).

• For the particular case of $P_{\text{variable}}^{\text{sequential}}$ we will see that we have a strong dependence on the sequence of $\delta$ that we consider.

This is an important issue, as we will see, since it affects only this method, which is the one that we consider potentially the best in terms of minimization results vs. computational cost.

What we can see on Figure (4.2) is the representation of a sequence of three values of $\delta$ that has been solved for a range of values of $\alpha$ in between 1 and 15. The conclusion of the plot would be that $\alpha$ has very little effect on the results until it goes beyond a certain value $\alpha = 6$. After that, we get the same results but linearly increased, as you can see on Fig (4.1).

Figure 4.1: $P_{\text{variable}}^{\text{sequential}}$ sensitivity with respect to $\alpha$
Sensitivity of the Sequential Variable Method with respect to $\alpha$

Figure 4.2: $P_{\text{sequential variable}}$ sensitivity with respect to $\alpha$
Chapter 5

Comparing the four methods for the Nested Sets

In this chapter we will compare the results obtained from the two methods that we described on Chapter 3 ($P_{\text{sequential constant}}$ and $P_{\text{global constant}}$) and the two methods described on Chapter 4 ($P_{\text{sequential variable}}$ and $P_{\text{global variable}}$). Additionally we will study the influence of new parameters. As a first comparison we present some results for this system:

$$\alpha = 1, \quad u_{\text{lim}}^2 = 1000^2, \quad w_{\text{max}}^2 = 1000^2,$$

$$A = \begin{pmatrix} 0 & 1 \\ -10 & -10 \end{pmatrix} \quad B_1 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} \quad B_2 = \begin{pmatrix} 0 & 6 \end{pmatrix} \quad C = \begin{pmatrix} 5 & 0 \end{pmatrix} \quad D = 0$$

When applied to the next set of values for $\delta$: $\delta = [1, 5, 10, 20, 30, 40, 50, 60, 75]$

For all these values, the performances obtained are shown on table 5.1 and on Figure 5.1 (see next page). As we can see in Figure 5.1, $P_{\text{global variable}}$ does improve $P_{\text{global constant}}$ results; and $P_{\text{sequential constant}}$ is clearly the worst in terms of performance. What is surprising though, is that $P_{\text{sequential variable}}$ is not better than $P_{\text{global constant}}$. This fact is unexpected given that $P_{\text{sequential variable}}$ is indeed a nested ellipsoids method. It should work better than a
conventional Q constant method since P is allowed to change, looking for the best shape for each situation, only constrained by the nesting condition.

Table 5.1: Results for all 4 methods. The weights for the global schemes are in both cases \( w = [1, 1, 1, 1, 1, 1] \).

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>( P_{\text{sequential constant}} )</th>
<th>( P_{\text{global constant}} )</th>
<th>( P_{\text{sequential variable}} )</th>
<th>( P_{\text{global variable}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0905</td>
<td>1.1044</td>
<td>1.0905</td>
<td>1.1100</td>
</tr>
<tr>
<td>5</td>
<td>0.9480</td>
<td>0.4919</td>
<td>0.7524</td>
<td>0.4903</td>
</tr>
<tr>
<td>10</td>
<td>0.9480</td>
<td>0.3491</td>
<td>0.7421</td>
<td>0.3472</td>
</tr>
<tr>
<td>20</td>
<td>0.9480</td>
<td>0.2946</td>
<td>0.7395</td>
<td>0.2672</td>
</tr>
<tr>
<td>30</td>
<td>0.9480</td>
<td>0.2946</td>
<td>0.7390</td>
<td>0.2507</td>
</tr>
<tr>
<td>40</td>
<td>0.9480</td>
<td>0.2946</td>
<td>0.7388</td>
<td>0.2446</td>
</tr>
<tr>
<td>50</td>
<td>0.9480</td>
<td>0.2946</td>
<td>0.7388</td>
<td>0.2417</td>
</tr>
<tr>
<td>60</td>
<td>0.9480</td>
<td>0.2946</td>
<td>0.7387</td>
<td>0.2401</td>
</tr>
<tr>
<td>75</td>
<td>0.9480</td>
<td>0.2946</td>
<td>0.7387</td>
<td>0.2388</td>
</tr>
</tbody>
</table>

Figure 5.1: Comparison of all 4 methods. \( P_{\text{sequential constant}}, P_{\text{sequential variable}}, P_{\text{global constant}} \) and \( P_{\text{global variable}} \).

Since this method is supposed to be the best choice in terms of performance versus computational cost (see section 4.4.2 for details), we will pay attention to this unexpected result and try to find out a way to avoid it and improve the method.
Next we consider a different system to see if these unexpected results are something common or particular to that combination of parameters.

\[ \alpha = 1, \quad w_{\text{lim}}^2 = 1000^2, \quad w_{\text{max}}^2 = 1000^2, \]

\[
A = \begin{pmatrix} -1 & 5 \\ -10 & -15 \end{pmatrix} \quad B_1 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} \quad B_2 = \begin{pmatrix} 0 & 6 \end{pmatrix} \quad C = \begin{pmatrix} 5 & 0 \end{pmatrix} \quad D = 0
\]

For all these values, the performances obtained are shown below, on table 5.2:

Table 5.2: Results for all 4 methods. The weights for the global schemes are in both cases \( w = [1, 1, 1, 1, 1] \).

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>( P_{\text{sequential constant}} )</th>
<th>( P_{\text{global constant}} )</th>
<th>( P_{\text{sequential variable}} )</th>
<th>( P_{\text{global variable}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5366</td>
<td>0.5747</td>
<td>0.5366</td>
<td>0.5424</td>
</tr>
<tr>
<td>10</td>
<td>0.2292</td>
<td>0.2016</td>
<td>0.2004</td>
<td>0.1990</td>
</tr>
<tr>
<td>20</td>
<td>0.2292</td>
<td>0.1717</td>
<td>0.1671</td>
<td>0.1527</td>
</tr>
<tr>
<td>30</td>
<td>0.2292</td>
<td>0.1717</td>
<td>0.1585</td>
<td>0.1373</td>
</tr>
<tr>
<td>40</td>
<td>0.2292</td>
<td>0.1717</td>
<td>0.1552</td>
<td>0.1309</td>
</tr>
<tr>
<td>50</td>
<td>0.2292</td>
<td>0.1717</td>
<td>0.1536</td>
<td>0.1277</td>
</tr>
<tr>
<td>60</td>
<td>0.2292</td>
<td>0.1717</td>
<td>0.1527</td>
<td>0.1259</td>
</tr>
<tr>
<td>70</td>
<td>0.2292</td>
<td>0.1717</td>
<td>0.1521</td>
<td>0.1248</td>
</tr>
<tr>
<td>80</td>
<td>0.2292</td>
<td>0.1717</td>
<td>0.1518</td>
<td>0.1241</td>
</tr>
</tbody>
</table>

As we can see on the table, for this case results correspond to what we expected (both variable schemes give better results than the two constant). On Figure 5.2 we can see the corresponding plots for this case.

By trying several more different cases (different A matrices) and different "stepsizes" (distances between consecutive values of \( \delta \)) we realize that for certain combinations of stepsizes and systems the results obtained present this unexpected result and for some others they do not. All four schemes depend on the system considered, but we
Figure 5.2: Second comparison for all 4 methods. In this case results corresponds to what was expected. For both global methods weights are \( w = [1, 1, 1, 1, 1] \)

have noticed that the sequential variable scheme is also dramatically dependent on the stepsize inside a fixed interval \([\delta_{\text{min}}, \delta_{\text{max}}]\), whereas the others are not nearly as dependent \(^1\).

So far we have exposed a comparison between our 4 methods and we have detected that, under certain circumstances, our preferred method (\( P_{\text{sequential variable}} \)) does not give the expected results.

We have also detected the dependence on the system matrix (A) considered -as expected- as well as a dependence on the particular values of \( \delta \) -the so called stepsize of the sequence- for the case of \( P_{\text{sequential variable}} \). On the next sections of this chapter we will study the reasons for this unexpected results, focusing on why is \( P_{\text{sequential variable}} \) worse than \( P_{\text{constant}} \) in some situations.

\(^1\)The Constant Sequential Scheme is not dependent at all, whereas the two Global Schemes might have small variations if we modify the stepsize inside a fixed interval \([\delta_{\text{min}}, \delta_{\text{max}}]\). Of course, if we change the length of the interval the results will change much more.
5.1 Study of the unexpected result

Roughly speaking, our problem is that at a certain value of $\delta$, the global constant scheme has an ellipsoid (determined by the Lyapunov function parameter $P$) that gives a better performance than any $P$ that the sequential variable scheme can find.

On Figure 5.3 we have the representation of the ellipsoids corresponding to both methods (sequential variable and global constant) for 2 consecutive values of $\delta$ where this anomaly occurs. Recall that each ellipsoid corresponds to a level set of the Lyapunov function considered for a value of $\delta$ such that:

$$V = x^T P(\delta_n) x = \frac{w_{\max}^2}{\delta_n^2}$$

![Ellipsoids for Delta = [1 5] Sequential variable and Global constant methods](image)

Figure 5.3: Picture of the anomaly: $E_{1\text{variable}}$ has a better performance than $E_{1\text{constant}}$ whereas $E_{2\text{variable}}$ has a worse performance than $E_{2\text{constant}}$.

Notice how the constant scheme gives us 2 ellipsoids ($\epsilon_{1\text{constant}}$ and $\epsilon_{2\text{constant}}$) and they have exactly the same shape but scaled -shape depends only on $P$ and the global constant method cannot change $P$. 

69
On the other hand, we have that for the variable case, ellipsoids are smaller (more "precise" since they focus on each value of $\delta$ individually) and different $^2$. In fact, as we can see from the variable ellipsoids of Figure 5.3, for $\delta_2$ a small counter-clockwise turn was required with respect to $\epsilon_1^{\text{variable}}$ in order to minimize the performance.

The unexpected fact is that whereas $\epsilon_1^{\text{variable}}$ has a better performance associated than $\epsilon_1^{\text{constant}}$, surprisingly $\epsilon_2^{\text{variable}}$ is not better than $\epsilon_2^{\text{constant}}$. That is to say: for $\delta_2$ the constant scheme works better than the variable. Why does not the system choose $P_2^{\text{variable}}$ to be equal to $P_2^{\text{constant}}$ if it has a better performance? The only condition that can cause this is the nesting condition.

As we will show later, the variable scheme cannot choose $\epsilon_2^{\text{constant}}$ as a valid solution for $\delta_2$ because it is not nested on $\epsilon_1^{\text{sequential}}$:

$$P(\delta_2)^{\text{global}}_{\text{constant}} < P(\delta_1)^{\text{sequential}}_{\text{variable}}$$

Whereas the nesting condition imposes that the solution for the variable sequential scheme has to be:

$$P(\delta_2)^{\text{sequential}}_{\text{variable}} > P(\delta_1)^{\text{sequential}}_{\text{variable}}$$

Therefore, $\epsilon_2^{\text{variable}}$ is the best solution that accomplishes the nesting condition, and that is why is the best option for the variable sequential method.

To see graphically whether the nesting condition is accomplished or not, we just have to compare the two ellipsoids by representing the two corresponding Lyapunov functions ($P(\delta_n)$) at the same level set (in this case we pick a random value $\delta = 5$).

$$V = x^T P(\delta_n)x = \frac{u_{\max}^2}{5^2}$$

---

$^2$To point out this fact, notice that the second variable ellipsoid ($\epsilon_2^{\text{variable}}$) does not have the same orientation as the first variable ellipsoid ($\epsilon_1^{\text{variable}}$)
We do this with the ellipsoids corresponding to Figure 5.3 and as a result of that we get next Figure 5.4

![Checking the nesting condition](image)

Figure 5.4: By representing the different Lyapunov functions at the same level set, we can check graphically the nesting condition.

As it is clear from Figure 5.4, the ellipsoid $\epsilon_{2\text{constant}}$ was not valid for the sequential scheme, since it is not nested inside $\epsilon_{1\text{variable}}$. Instead of that, $\epsilon_{2\text{variable}}$ is the ellipsoid with the lowest performance among all ellipsoids nested inside $\epsilon_{1\text{variable}}$.

Now that we know which condition is the cause of this results, we focus on the performances associated to the ellipsoids represented on Figures 5.3 to see exactly how does this conflict show up. These performances are shown on the next table:

Table 5.3: **Performances on the critic part of the $\delta$ sequence:**
*See also Figure 5.5 for graphical representation.*

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$\epsilon_{\text{sequential}}$</th>
<th>$\epsilon_{\text{constant}}$</th>
<th>$\epsilon_{\text{variable}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_1$</td>
<td>0.5366</td>
<td>&lt; 0.5370</td>
<td></td>
</tr>
<tr>
<td>$\delta_2$</td>
<td>0.3024</td>
<td>&gt; 0.3008</td>
<td></td>
</tr>
</tbody>
</table>

By looking at this performances, we can see that $\delta_1$ has a slightly better performance with the sequential method; whereas at $\delta_2$, it is the global method who has a better
result. Notice how, for \( \delta_1 \), in order to improve the performance from \( \gamma_{1_{\text{global}}} = 0.5370 \) to \( \gamma_{1_{\text{sequential}}} = 0.5366 \) the sequential method dismisses \( P_{1_{\text{constant}}} \) as a solution and gets a slightly better \( P_{1_{\text{variable}}} \).

That choice turns out to lead the sequence of solutions to a *dead end* when we have to solve the system for \( \delta_2 \) and beyond. For these values, the constant global method uses again the same \( P \); but the sequential method may look for a different \( P \).

The performance \( \gamma_{2_{\text{global}}} = 0.3008 \) seems to be fairly good solution for \( \delta_2 \). Nevertheless, the previous choice for \( \delta_1 \) makes the corresponding \( P_{2_{\text{constant}}} \) impossible to take as a solution for \( P_{2_{\text{variable}}} \) (as we can see on Fig 5.4 it cannot be nested in \( P_{1_{\text{variable}}} \)). We could say that the choice for \( P_{1_{\text{variable}}} \) was too restrictive and made very difficult to find nested ellipsoids, with a good minimal performance, for the following \( \delta \).

Instead, if we would have taken \( P_{1_{\text{variable}}} = P_{1_{\text{constant}}} \), the performance at \( \delta_1 \) would have been slightly worse. \(^3\) But it would have been easier to nest the following ellipsoids and get better performances. As a result of that, the system is forced to take a third solution, the definitive \( P_{2_{\text{variable}}} \), that has associated a performance of \( \gamma_{2_{\text{sequential}}} = 0.3024 \). This will lead the sequence of solutions progressively away from the optimal results, getting stuck on an unexpectedly bad performance because of being too aggressive on the minimization on \( \delta_1 \).

On next Figure 5.5 we have a representation of this fact. We might summarize it by saying that the sequential variable method is too aggressive at each single value of \( \delta \); and sometimes it is more convenient to accept a slightly worse minimal performance if it has associated a not so restrictive ellipsoid. This way we will have more chances to get a better minimal performance along the rest of the sequence.

\(^3\)It would have been \( \gamma_1 = 0.5370 \) instead of \( \gamma_1 = 0.5366 \)
Figure 5.5: As you can see on the zoom, this small improvement on $\delta_n$ prevents us from having a good result for $\delta_{n+1}$.

Initially the $p_{\text{sequential}}$ method has the best results, as expected.

At some point, a slight improvement over the $P_{\text{constant}}$ results forces us to take a very restrictive ellipsoid.

This restrictive ellipsoid makes very difficult to find, for the following values of the sequence, nested ellipsoids with a good performance associated.

The $p_{\text{sequential}}$ method gets stucked on this restrictive ellipsoid shape, with worse performances associated.
5.2 Stepsize influence

At this point we might think about what is the influence of the stepsize \(^4\) on this conflictive result and in all 4 methods in general.

As we have mentioned before, stepsize only influences the results of the sequential variable method. All the particular cases considered show that it has very little influence on the other three methods, apart from the logical resolution of the the results (the shorter the stepsize is, the more points we will have calculated, and the smoother the plots \(\gamma \text{ vs. } \delta\) will be.

Notice that all previous plots \(\gamma \text{ vs. } \delta\) show that most of the evolution of the performance occurs through the lower values of \(\delta\). In particular, for both systems considered on this chapter it occurs along the interval \(\delta = [1 \ 25]\).

Therefore, in order to have a good resolution and an appropriate scheduling of the the design parameters of our system, we have to focus our sequence of \(\delta\) values in that interval. As a result of that we will consider always a sequence of values of \(\delta\) with smaller stepsizes on this initial range of \(\delta\) and bigger stepsizes beyond it.

**The case of \(P^{\text{sequential}}_{\text{variable}}\)**

As it is easy to imagine, the risk of having the variable sequential scheme stuck at a particular performance is only a worry when we are on this initial range of values of \(\delta^5\). However, we have explained that we need a smaller stepsize on this range of values in order to have an appropriate "resolution".

As a result of that, the more values of \(\delta\) that we consider on the initial interval of \(\delta\), the more chances we will have of getting stuck there, and presumably at a higher performance, making more evident these unexpected results.

\(^4\)Recall that we name stepsize to the distance between consecutive values of \(\delta\).
\(^5\)Otherwise we are in a range of \(\delta\) where the performance does not change much anyway.
In order to show this, we have Fig 5.6, that shows the $\gamma$ vs. $\delta$ evolution obtained by the sequential variable case for a different set of $\delta$ with different initial stepsizes:

Table 5.4: Some $\delta$ sequences with different initial stepsizes.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>[1 1.1 1.5 2.5 5 7.5 12.5 15 20 25 50]</td>
</tr>
<tr>
<td>$S_2$</td>
<td>[1 1.5 2.5 5 7.5 12.5 15 20 25 50]</td>
</tr>
<tr>
<td>$S_3$</td>
<td>[1 2.5 5 7.5 12.5 15 20 25 50]</td>
</tr>
<tr>
<td>$S_4$</td>
<td>[1 5 7.5 12.5 15 20 25 50]</td>
</tr>
<tr>
<td>$S_5$</td>
<td>[1 7.5 12.5 15 20 25 50]</td>
</tr>
<tr>
<td>$S_6$</td>
<td>[1 12.5 15 20 25 50]</td>
</tr>
</tbody>
</table>

Figure 5.6: Stepsize influence over $P_{\text{sequential}}$.

However, we can be more precise and say that it is not the stepsize between any consecutive $\delta$ what matters. The really critical issue is on the first stepsizes, since they will condition the number of values of $\delta$ that we will consider on this so critical initial part of the sequence.
This can be shown on Figure 5.6, which shows us that the optimal initial stepsize for the variable sequential method is $\delta = [1 \ 7.5]$. We have tried to improve even more by trying different possibilities for the second stepsize, but the results does not change much. It only depends on the values of $\delta$ on the lower range, not on the stepsizes of the whole sequence.

**The case of $p_{global}$ methods**

For both global methods, the dependence on the $\delta$ sequence is very small. As we said before, mainly we only have effects in terms of the "resolution" of the plots $\gamma$ vs. $\delta$.

However, we might have different final values of the performance if we change the last $\delta$ of the sequence for or from a value on the initial rank of $\delta$ (which, as we said before, is $\delta \in [1 \ 20]$).
Chapter 6

New $P_{\text{sequential}}$ method

In this chapter, we will try to propose an alternative method, based on the $P_{\text{sequential}}$ scheme, such that we can avoid this unexpected results and therefore improve the results given by $P_{\text{global constant}}$ at a reasonable computational cost.

As we have pointed out on the previous chapters, the main problem of the $P_{\text{sequential}}$ scheme comes from its lack of global perspective. The example shown on Figure 5.5 tells us that it might be better to sacrifice a little on the minimization of the performance at a certain $\delta$, in order to not to compromise the minimization along the rest of the sequence.

As a strategy of solution, we initially attempted to incorporate scaling factors to the system [4]. This scaling factors would be designed to reduce those ellipsoids with a good performance associated but too big to be nested.

However, considering scaling factors for $P$ presented some difficulties when incorporating in the set of LMI. The problem was that we could not include these scaling parameters while respecting our initial specifications of minimal performance, non saturation and maximum reachable set. As a result of that, we dismissed this strategy.
To find our final scheme of solution we focused on the idea of the lack of global perspective on \( P_{\text{sequential}} \). The new scheme needs to have a global idea of the whole sequence of \( \delta \) that we are going to consider, but at the same time it has to keep a mainly sequential procedure in order to keep its reasonable computational cost advantage over the global methods.

The basic idea was to perform a quick global resolution of a simplified version of the whole sequence at each value of \( \delta \). This way we make sure that our solution for each \( \delta \) will consider somehow the following remaining values of the sequence. As a result of that we will not sacrifice the performance of all the following values of the sequence in order to improve slightly the performance of our current value, as happens on Figure 5.5.

Along this chapter we will discuss the ways to implement this strategy on a new version of the \( P_{\text{sequential}} \) method, as well as we will give some numerical results and parameters to evaluate how convenient this new scheme is.
6.1 New strategy: Combined methods (CM)

Let’s start thinking about this new strategies by proposing a sequence of values $\delta$:

$$\delta = [1 \ 1.1 \ 1.5 \ 2.5 \ 5 \ 7.5 \ 12.5 \ 15 \ 20 \ 25 \ 50 \ 100]$$

We also recall our set of conditions that we are going to use. Let $r_m$ represent the value of $r$ associated to the minimal $\delta$, and $r_n$ represent the value of $r$ associated to any $\delta$:

- **Reachable set:** $\rightarrow \ R < 0$
  $$R = \begin{pmatrix}
  AQ(r_m) + Q(r_m)A^T + \alpha Q(r_m) + B_2F(r_m) + F^T(r_m)B_2^2 & B_1 \\
  B_1^T & -\alpha
  \end{pmatrix} < 0$$

- **Non-saturation:** $M(r) < 0 \rightarrow M(r_n) > 0$
  $$M(r_n) = \begin{pmatrix}
  Q(r_n) & F^T(r_n) \\
  F(r_n) & r_n
  \end{pmatrix} > 0$$

- **Optimal performance:** $J(r) < 0 \rightarrow J(r_n) < 0$
  $$J(r_n) = \begin{pmatrix}
  AQ(r_n) + Q(r_n)A^T + B_2F(r_n) + F(r_n)^T(r)B_2^T & B_1^T & Q(r_n)C_1^T + F(r_n)^TDT^T \\
  B_1 & -\gamma(r_n)^2 & 0 \\
  C_1Q(r_n) + DF(r_n) & 0 & -I
  \end{pmatrix} < 0$$

- **Nested ellipsoids:**
  $$Q(r_n) < Q(r_{n-1})$$

- **Improved performance:**
  $$\gamma(r_n) \leq \gamma(r_{n-1})$$
Where, the parameter $r$ was defined as:

$$r_m = 1 \frac{u_m^2}{u_{\text{max}}^2} \quad \text{and} \quad r_n = \delta_n^2 \frac{u_n^2}{u_{\text{max}}^2}$$

As we have seen before, the variable sequential scheme, solves this sequence by applying the corresponding conditions to each of the values (sequentially). Instead of that, what we are proposing as an improvement of this scheme is to include a previous global step on each of this one by one steps.

The intention will be to combine the nice properties from both sequential and global methods in order to provide the sequential scheme with a wider perspective. However, the way we define this previous step will condition how wide -and therefore, effective- this new perspective will be.

### 6.1.1 First combined method (1-CM):

**Divide the sequence on sections solved globally**

This is our first attempt to improve the variable sequential method, and will serve as an example of what our strategy is going to be. On chapter 4 we saw how by sacrificing a little bit on $\delta_n$ we can avoid the restrictive ellipsoid shape, and therefore avoid getting stuck on the next values.

To avoid this from happening, we divide the sequence of $\delta$ in groups of 3. This segments will be solved with the variable global scheme. This way, we will minimize our performance not only for $\delta_n$, but also considering what is going to come next ($\delta_{n+1}$ and $\delta_{n+2}$). The weight that we will consider for this global resolution will be $w = [1 1 1]$

According to this, the detailed scheme of resolution for this first improved version of the variable sequential method will be the one shown on table 6.1:
Table 6.1: First proposed improved sequential variable

<table>
<thead>
<tr>
<th>Values of $\delta_i = [\delta_{i,1} \delta_{i,2} \delta_{i,3}]$</th>
<th>LMI considered</th>
</tr>
</thead>
</table>
| $\delta_1 = [1 \ 1.1 \ 1.5]$ | $\delta_{1,1}$ | $\begin{align*}
\text{ReachableSet} : & \quad R < 0 \\
\text{Nonsaturation} : & \quad M(r_{1,1}) > 0 \\
\text{Performance} : & \quad J(r_{1,1}) < 0
\end{align*}$ |
| $\delta_{1,2}$ | $\begin{align*}
\text{Nonsaturation} : & \quad M(r_{1,2}) > 0 \\
\text{Performance} : & \quad J(r_{1,2}) < 0
\end{align*}$ |
| $\delta_{1,3}$ | $\begin{align*}
\text{Nonsaturation} : & \quad M(r_{1,3}) > 0 \\
\text{Performance} : & \quad J(r_{1,3}) < 0
\end{align*}$ |

| $\delta_2 = [\delta_{2,1} \delta_{2,2} \delta_{2,3}]$ | $\delta_{2,1}$ | $\begin{align*}
\text{Nonsaturation} : & \quad M(r_{2,1}) > 0 \\
\text{Performance} : & \quad J(r_{2,1}) < 0 \\
\text{Nesting Condition} : & \quad Q(r_{2,1}) < Q(r_{(1),3}) \\
\text{Perf. Improvement} : & \quad \gamma(r_{2,1}) < \gamma(r_{(1),3})
\end{align*}$ |
| $\delta_{2,2}$ | $\begin{align*}
\text{Nonsaturation} : & \quad M(r_{2,2}) > 0 \\
\text{Performance} : & \quad J(r_{2,2}) < 0 \\
\text{Nesting Condition} : & \quad Q(r_{2,2}) < Q(r_{2,1}) \\
\text{Perf. Improvement} : & \quad \gamma(r_{2,2}) < \gamma(r_{2,1})
\end{align*}$ |
| $\delta_{2,3}$ | $\begin{align*}
\text{Nonsaturation} : & \quad M(r_{2,3}) > 0 \\
\text{Performance} : & \quad J(r_{2,3}) < 0 \\
\text{Nesting Condition} : & \quad Q(r_{2,1}) < Q(r_{2,2}) \\
\text{Perf. Improvement} : & \quad \gamma(r_{2,3}) < \gamma(r_{2,2})
\end{align*}$ |

<table>
<thead>
<tr>
<th>$\delta_i = [\delta_{i,1} \delta_{i,2} \delta_{i,3}]$</th>
<th>$\delta_{i,1}$</th>
<th>for $i \in [3, 4, \ldots n]$ same procedure as $\delta_{2,1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_{i,2}$</td>
<td>for $i \in [3, 4, \ldots n]$ same procedure as $\delta_{2,2} \delta_{2,3}$</td>
<td></td>
</tr>
</tbody>
</table>
Comparing the 1-CM with the old $P_{\text{sequential}}^{\text{variable}}$

Next we are going to compare the results of this new version of the method with the results given by the initial version of the variable sequential method. At the same time, we will compare this results with the constant global scheme, which is the reference that we are trying to improve in order to avoid the so called *unexpected result*.

The simulations correspond to the system:

$$\alpha = 1, \quad u_{\text{lim}}^2 = 1000^2, \quad w_{\text{max}}^2 = 1000^2,$$

$$A = \begin{pmatrix} 0 & 1 \\ -10 & -10 \end{pmatrix} \quad B_1 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} \quad B_2 = \begin{pmatrix} 0 & 6 \\ 0 & 6 \end{pmatrix} \quad C = \begin{pmatrix} 5 \\ 0 \end{pmatrix} \quad D = 0$$

And the following sequence of $\delta$:

$$\delta = [1 \ 2.5 \ 5 \ 10 \ 20 \ 35 \ 70 \ 150]$$

On the next table 6.2 and Figure 6.1 we show the results for 2 versions of the first combined method (1CM). The first version ($CM_3$) will be the described version, whereas the second ($CM_2$) will be basically the same but dividing the sequence of $\delta$ in groups of 2 values, instead of 3.

As we can see, it is better to do the global ”*global pre-step*” with 3 values instead of 2. Nevertheless, although the results are much better than the results of the initial variable sequential method, this first version of the Combined Methods is still worse than the constant global scheme, so the *unexpected result* has not been avoided yet.
Table 6.2: Results for all 4 methods.
The weights for the global scheme are \( w = [1, 1, 1, 1, 1] \), and
\( w = [1, 1, 1] \) and \( w = [1, 1] \) for the two versions of the CM.

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>( P_{\text{constant}} )</th>
<th>( 1 - CM_3 )</th>
<th>( 1 - CM_2 )</th>
<th>( P_{\text{sequential variable}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.1032</td>
<td>1.0936</td>
<td>1.0917</td>
<td>1.0905</td>
</tr>
<tr>
<td>2.5</td>
<td>0.6947</td>
<td>0.6982</td>
<td>0.7002</td>
<td>0.7524</td>
</tr>
<tr>
<td>5</td>
<td>0.4920</td>
<td>0.4968</td>
<td>0.6101</td>
<td>0.7421</td>
</tr>
<tr>
<td>10</td>
<td>0.3492</td>
<td>0.4397</td>
<td>0.5896</td>
<td>0.7395</td>
</tr>
<tr>
<td>20</td>
<td>0.2955</td>
<td>0.4261</td>
<td>0.5833</td>
<td>0.7390</td>
</tr>
<tr>
<td>35</td>
<td>0.2955</td>
<td>0.4231</td>
<td>0.5829</td>
<td>0.7388</td>
</tr>
<tr>
<td>70</td>
<td>0.2955</td>
<td>0.4220</td>
<td>0.5828</td>
<td>0.7388</td>
</tr>
<tr>
<td>150</td>
<td>0.2955</td>
<td>0.4217</td>
<td>0.5828</td>
<td>0.7387</td>
</tr>
</tbody>
</table>

Figure 6.1: Representation for data shown on table 6.2
6.1.2 Second Combined Method (2-CM):

Solve sequentially by small overlapped groups

Based on the idea that the 1-CM was not good enough, we go one step further following the same principles. On the 1-CM scheme we were solving sequentially a concatenation of small subsystems of LMI that we solved globally. On 2-CM we will solve the same concatenation (same LMI for each value of δ, according to table 6.1). The difference now is that in this case we will repeat it for each value of the sequence of δ, overlapping our groups of δ. The idea is shown on table 6.3:

Table 6.3: Comparing the procedure of 1−CM3 with 2−CM3. The LMI that we are considering are the same. The only difference is the way we pick the values for the "global pre-step"

<table>
<thead>
<tr>
<th>Method used</th>
<th>Sequential step</th>
<th>Values solved globally</th>
</tr>
</thead>
<tbody>
<tr>
<td>1−CM3</td>
<td>1st</td>
<td>1.0 2.5 5.0</td>
</tr>
<tr>
<td></td>
<td>2nd</td>
<td>10 20 35</td>
</tr>
<tr>
<td></td>
<td>3rd</td>
<td>70 150 230</td>
</tr>
<tr>
<td>2−CM3</td>
<td>1st</td>
<td>1.0 2.5 5.0</td>
</tr>
<tr>
<td></td>
<td>2nd</td>
<td>2.5 5.0 10</td>
</tr>
<tr>
<td></td>
<td>3rd</td>
<td>5.0 10 20</td>
</tr>
<tr>
<td></td>
<td>4th</td>
<td>10 20 35</td>
</tr>
<tr>
<td></td>
<td>5th</td>
<td>20 35 70</td>
</tr>
<tr>
<td></td>
<td>6th</td>
<td>35 70 150</td>
</tr>
</tbody>
</table>

Notice that, on 1−CM we originally did not have δ = 230 as the last value of the sequence. The developed algorithm "adds" this value to the sequence in order to have 3 values for the last step. To calculate it, the algorithm considers the same distance (stepsize) as we had between the two last values of the sequence (δ = 70 and δ = 150).

Also notice that for 2−CM the bold characters correspond to the values that we
will keep as a final solution \(^1\). On this new version of the CM scheme, we calculate each \(\delta\) 3 times \(^2\), but we only keep the results when \(\delta\) is the first value of the subsequence of 3 that we will solve globally.

This way we get a more accurate solution. Since the evolution of the relationship \(\gamma\) \textit{vs.} \(\delta\) is always a descendant line, we will keep the value that has been calculated while considering with it the next two values, whereas on the \(1 - CM\) scheme some values where considered with the two following values, and some others not.

The results from this new scheme of solutions are shown next:

**Comparing the 2-CM with 1-CM and the old methods**

As we did before for the 1-CM method, next we will compare the results from the 2-CM method with the results from both the variable sequential method and the constant global method. As a reference, we will also include the previous results from the \(1 - CM_3\) method.

The simulations correspond to the same system:

\[
\begin{align*}
\alpha &= 1, \quad u^2_{\text{lim}} = 1000^2, \quad w^2_{\text{max}} = 1000^2, \\
A &= \begin{pmatrix} 0 & 1 \\ -10 & -10 \end{pmatrix}, \quad B_1 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & 6 \end{pmatrix}, \quad C = \begin{pmatrix} 5 & 0 \end{pmatrix}, \quad D = 0
\end{align*}
\]

And the same sequence:

\[
\delta = [1 \ 2.5 \ 5 \ 10 \ 20 \ 35 \ 70 \ 150]
\]

From table 6.4 and the corresponding Fig 6.2 we can see that this new version of the method works much better. Actually, for the case where we consider 3 values for the

---

\(^1\)This values of \(\delta\) and all the corresponding values of \(P\), \(K\) and \(\gamma\), of course.  
\(^2\)With the exception of these values that are at the beginning or the end of the sequence.
Table 6.4: Results for all 5 cases.
The weights for the global scheme are \(w = [1, 1, 1, 1, 1]\), and 
\(w = [1, 1, 1]\) and \(w = [1, 1]\) for the CM.

<table>
<thead>
<tr>
<th>(\delta)</th>
<th>(P_{\text{global constant}})</th>
<th>(1 - CM_3)</th>
<th>(1 - CM_2)</th>
<th>(2 - CM_3)</th>
<th>(2 - CM_2)</th>
<th>(P_{\text{sequential}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.1032</td>
<td>1.0936</td>
<td>1.0917</td>
<td>1.0936</td>
<td>1.0917</td>
<td>1.0905</td>
</tr>
<tr>
<td>2.5</td>
<td>0.6947</td>
<td>0.6982</td>
<td>0.7002</td>
<td>0.6983</td>
<td>0.7003</td>
<td>0.7524</td>
</tr>
<tr>
<td>5</td>
<td>0.4920</td>
<td>0.4968</td>
<td>0.6101</td>
<td>0.4970</td>
<td>0.4997</td>
<td>0.7421</td>
</tr>
<tr>
<td>10</td>
<td>0.3492</td>
<td>0.4397</td>
<td>0.5896</td>
<td>0.3541</td>
<td>0.3580</td>
<td>0.7395</td>
</tr>
<tr>
<td>20</td>
<td>0.2955</td>
<td>0.4261</td>
<td>0.5833</td>
<td>0.2908</td>
<td>0.3215</td>
<td>0.7390</td>
</tr>
<tr>
<td>35</td>
<td>0.2955</td>
<td>0.4231</td>
<td>0.5829</td>
<td>0.2786</td>
<td>0.3134</td>
<td>0.7388</td>
</tr>
<tr>
<td>70</td>
<td>0.2955</td>
<td>0.4220</td>
<td>0.5828</td>
<td>0.2740</td>
<td>0.3104</td>
<td>0.7388</td>
</tr>
<tr>
<td>150</td>
<td>0.2955</td>
<td>0.4217</td>
<td>0.5828</td>
<td>0.2728</td>
<td>0.3096</td>
<td>0.7387</td>
</tr>
</tbody>
</table>

Figure 6.2: Representation for data shown on table 6.4
so called "global pre-step", it gives us better results than the global constant scheme. As a result of that, in spite of its higher computational cost \(^3\), we can consider this new \(2-CM_3\) method as our final solution. However, we have to study first the influence of the stepsize.

The influence of the stepsize on \(2-CM_3\)

As we saw on section 5.2 of last chapter, the stepsize does play a role on determining at what point of the sequence is the variable sequential method going to "get stuck". Recall that all major changes on the relationship \(\gamma vs. \delta\) occurred at the lower values of \(\delta\); and the more values we consider on this region, the more chances we have for having our method diverting from the "expected" minimization path of \(\gamma\) along \(\delta\). \(^4\)

For this reason, it is natural to think that if \(\delta_{1.1}, \delta_{1.2}\) and \(\delta_{1.3}\) are close to each other and at the lower values of the sequence (let’s consider \(\delta_{1.1} = 1, \delta_{1.2} = 1.01\) and \(\delta_{1.3} = 1.05\) as an example) the fact of considering 3 values, instead of one, will not help much, and we will get stuck the same way as happened with the traditional variable sequential method.

To see this, we will compare the variable sequential method and the \(2-CM_3\) method with 2 different sequences of \(\delta\):

\[
\text{Sequence1} \rightarrow \delta = [1, 1.1, 1.5, 2.5, 7, 15, 50, 150] \\
\text{Sequence2} \rightarrow \delta = [1, 2.5, 5, 10, 20, 35, 70, 150]
\]

For the same \(R^2\) system as usual (see previous section for details), the results are shown on the next table 6.5 and the corresponding Figure 6.3 (see next page)

\(^3\)Recall that \(2-CM_3\) calculates in 6 steps what the \(1-CM_3\) calculated on 3 (table 6.3)
\(^4\)See section 5.1 and specially Fig 5.5 for any further details
Table 6.5: Results for all 4 methods. The weights for the global scheme are \( w = [1, 1, 1, 1, 1, 1] \), and \( w = [1, 1, 1] \) and \( w = [1, 1] \) for the two versions of the CM.

<table>
<thead>
<tr>
<th>( \delta ) Seq2</th>
<th>( P_{\text{sequential variable}} )</th>
<th>( 2 - CM_3 )</th>
<th>( \delta ) Seq1</th>
<th>( P_{\text{sequential variable}} )</th>
<th>( 2 - CM_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0905</td>
<td>1.0936</td>
<td>1</td>
<td>1.0905</td>
<td>1.0910</td>
</tr>
<tr>
<td>2.5</td>
<td>0.7524</td>
<td>0.6983</td>
<td>1.1</td>
<td>1.0428</td>
<td>1.0426</td>
</tr>
<tr>
<td>5</td>
<td>0.7421</td>
<td>0.4970</td>
<td>1.5</td>
<td>0.9464</td>
<td>0.8985</td>
</tr>
<tr>
<td>10</td>
<td>0.7395</td>
<td>0.3541</td>
<td>2.5</td>
<td>0.8815</td>
<td>0.7020</td>
</tr>
<tr>
<td>20</td>
<td>0.7390</td>
<td>0.2908</td>
<td>7</td>
<td>0.8478</td>
<td>0.4263</td>
</tr>
<tr>
<td>35</td>
<td>0.7388</td>
<td>0.2786</td>
<td>15</td>
<td>0.8437</td>
<td>0.3246</td>
</tr>
<tr>
<td>70</td>
<td>0.7388</td>
<td>0.2740</td>
<td>50</td>
<td>0.8426</td>
<td>0.3034</td>
</tr>
<tr>
<td>150</td>
<td>0.7387</td>
<td>0.2728</td>
<td>150</td>
<td>0.8425</td>
<td>0.3015</td>
</tr>
</tbody>
</table>

Figure 6.3: Representation for data shown on table 6.5
Notice that, although the method $2 - CM_3$ is not as sensitive to the stepsize as the variable sequential method is, the fact of considering this two different stepsizes makes a difference. In fact, for the second sequence $2 - CM_3$ is better than our "milestone" (global constant method) that gives us $\gamma = 0.2955$ as a final result for $\delta = 150$; whereas for the first sequence it has a worse final performance.

Therefore, we have to include a last modification on the Combined Methods scheme in order to avoid this sensitivity to the stepsize, and as a result of that, make us sure that they will always be better than the global constant method. This is how we get the next version of the Combined Methods (3-CM).

6.1.3 Third Combined Method (3-CM):

Solve sequentially by small overlapped optimized groups

As we said before, the problem with the stepsize on the combined methods \(^5\) comes from the choice of values for the three $\delta$ that we use on the "global pre-step". For this third version of the Combined Method, we will respect the same strategy from $2 - CM$ but we will improve the way we choose these values.

To avoid the problem of picking three very close values, and therefore having a similar problem as the regular sequential variable scheme has (focusing only in one value of $\delta$); we will choose our three values for our "global pre-step" independently from our initial sequence of $\delta$.

As an example, let’s consider the usual sequence of $\delta$:

$$\delta = [1 \ 2.5 \ 5 \ 10 \ 20 \ 35 \ 70 \ 150]$$

\(^5\)We do not want our minimization results to depend on the sequence of $\delta$
We could say that the three first values are too close to each other, and therefore if the method picks $\delta = [1 \, 2.5 \, 5]$ as the first set for the "global pre-step”, it might get stuck at some high performance. What we do instead is to define a parameter that will force us to consider equally separated values of $\delta$ regardless of what the main sequence of $\delta$ has. This parameter is going to be called ”$\kappa$”, and as it grows it will make us consider more separated values, independently of what we have on the main sequence.

According to our explanation for the stepsize influence on the $\gamma$ vs. $\delta$ relationship, we would expect to have better performances as $\kappa$ grows\(^6\). On next Figures 6.4 and 6.5 we see the influence of this $\kappa$ parameter on the last performance of the sequence of $\delta$ obtained by $3 - CM_2$ and $3 - CM_3$.

The sequence of considered $\delta$ will be:

Table 6.6: Comparing the procedure of $2 - CM_3$ with $3 - CM_3$. Again, the LMI that we are considering are the same; and the only difference is the way we pick the values for the "global pre-step”

\(^6\)The global pre-step provides our sequential scheme with a wider perspective to avoid getting stuck at any point of the sequence of $\delta$
And the results of applying this new $\kappa$ parameter are shown on Figures 6.4 and 6.5, where we can see how it affects the final solution of the sequence ($\gamma_{final}$).

Figure 6.4: Stepsize sensibility on 3−CM$_3$ for an $R^5$ system (see next page for details). As you can see, the bigger the "global pre-step" gets, the better performance, we get.

Figure 6.5: Stepsize sensibility on 3−CM$_3$ for an $R^2$ system (see next page for details). Here the results are similar.
The systems corresponding to this two Figures are:

- For Figure 6.4

  \[ \alpha = 1, \quad u_{lim}^2 = 1000^2, \quad w_{max}^2 = 1000^2, \]

  \[
  A = \begin{pmatrix}
  -7 & 0 & 0 & 2 & 1 \\
  -10 & -10 & 0 & 0 & 4 \\
  1 & 0 & -7 & 3 & -2 \\
  0 & 0 & 0 & -8 & -1 \\
  1 & -3 & 1 & 2 & -3
  \end{pmatrix}, \quad B_1 = \begin{pmatrix} 0.5 \\ 1 \\ 0 \\ 1 \\ 2 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & 6 & 0 & 1 & 2 \end{pmatrix}
  \]

- For Figure 6.5:

  \[ \alpha = 1, \quad u_{lim}^2 = 1000^2, \quad w_{max}^2 = 1000^2, \]

  \[
  A = \begin{pmatrix}
  0 & 1 \\
  -10 & -10
  \end{pmatrix}, \quad B_1 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & 6 \end{pmatrix}, \quad C = \begin{pmatrix} 5 & 0 \end{pmatrix}, \quad D = 0
  \]

As we said, for both cases we notice the improvement as \( \kappa \) grows. Based on these Figures (specially Fig 6.4) we can establish the value of \( \kappa \) as:

\[
\kappa = \frac{\delta_{max}}{3} = \frac{150}{3} = 50
\]

Since 3 is the number of \( \delta \) that we consider on the global pre-step.

This is our last version of the Combined Methods. With it we have an essentially sequential method, with \( P \) variable, that gives us a better performance than the \( P_{\text{variable}} \) constant method. From now on we will dedicate the rest of this chapter to characterize its properties, paying special attention to its numerical complexity and calculation time.
6.2 Speeding up the new method

As we have seen on the previous sections, we have found a new version of the variable sequential method. This new version, also known as $3 - CM_3$, provides us a better performance minimization than the Constant Global method, and has no dependence upon the stepsize of the sequence of $\delta$. However, the price that we had to pay for getting these results is having a more complex and slower method than the original variable sequential method.

To make sure that, after this trade off, we still have an interesting method on $3 - CM_3$, we will study the complexity of the method, as well as the time required to go through a sequence of $\delta$ on a high order system ($R^5$). For such a system, the difference of computational cost should be evident, specially if we also consider a long sequence of $\delta$, as it is going to be the case.

However, it is important to point out that, although the speed of the method is an important factor, the main issue is its complexity in terms of number of LMI and variables to be handled at the same time. Therefore, on the next lines we will study the method in terms of speed, but without forgetting the analysis of its complexity.

Let our $R^5$ system and sequence of $\delta$ be:

$$\alpha = 1, \quad u_{\text{lim}}^2 = 1000^2, \quad w_{\text{max}}^2 = 1000^2,$$

$$A = \begin{pmatrix} -7 & 0 & 0 & 2 & 1 \\ -10 & -10 & 0 & 0 & 4 \\ 1 & 0 & -7 & 3 & -2 \\ 0 & 0 & 0 & -8 & -1 \\ 1 & -3 & 1 & 2 & -3 \end{pmatrix}, \quad B_1 = \begin{pmatrix} 0.5 \\ 1 \\ 0 \\ 1 \\ 2 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & 6 & 0 & 1 & 2 \end{pmatrix},$$

$$C = \begin{pmatrix} 5 \\ 0 \end{pmatrix}, \quad D = 0,$$

$$\delta = [ 1 \ 1.1 \ 1.2 \ 1.4 \ 1.7 \ 2.1 \ 2.5 \ 2.8 \ 3.1 \ 3.5 \ldots \ 4.3 \ 5.5 \ 6 \ 8 \ 10 \ 15 \ 40 \ 70 \ 110 \ 150 ]$$
On the following table we will show the final performance obtained (last $\gamma$ of the sequence), the total time of calculation required, and the number of LMI solved simultaneously for each of our main schemes of solution. This will help us to make an idea about their properties and complexity, in order to evaluate each of them.

Table 6.7: Comparing calculation time of $3-CM_3$, $P_{\text{sequential variable}}$, $P_{\text{global constant}}$ and $P_{\text{global variable}}$

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Time</th>
<th>Final performance</th>
<th>LMI solved together</th>
<th>Variables solved together</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{\text{sequential variable}}$</td>
<td>1.625 s</td>
<td>1.0438</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>$P_{\text{global variable}}$</td>
<td>210.234 s</td>
<td>0.1780</td>
<td>$80 \ (20\delta \times 4 \frac{LMI}{\delta})$</td>
<td>$60 \ (20\delta \times 3 \frac{\text{variables}}{\delta})$</td>
</tr>
<tr>
<td>$3-CM_3$</td>
<td>8.969 s</td>
<td>0.1957</td>
<td>12</td>
<td>9  \ ($3\delta \times 3 \frac{\text{variables}}{\delta}$)</td>
</tr>
<tr>
<td>$P_{\text{global constant}}$</td>
<td>13.105 s</td>
<td>0.2067</td>
<td>$80 \ (20\delta \times 3 \frac{LMI}{\delta})$</td>
<td>$41 \ (20\delta \times 2 \frac{\text{variables}}{\delta} + P)$</td>
</tr>
</tbody>
</table>

As we can see on table 6.7, our new method $3-CM$ is clearly slower and slightly more complex than the original variable sequential method. On the other hand, it has a much better performance, and is clearly faster than the global methods.

Regarding the rest of methods, we can see how the global methods complexity, in terms of number of LMI to be solved at the same time, grows dramatically as we consider longer sequences of $\delta$ (this case the sequence has 20 elements).

This fact results on a larger number of variables, and is the main reason for the method to be slower. At the same time, it poses a serious issue on the accuracy of the solver routine that provides the results for the system.

As a result of that, we definitively do not want to use such a complex method, and that is the main reason for not considering the global methods as a valid scheme of solution for our system of LMI. Although the $P_{\text{global variable}}$ method has proved to be the best in terms of performance achieved.
However, going back to the time of calculation \(^7\), we still can think about ways to reduce it. Regarding this issue, we may work on several aspects:

- Avoid overlapping between the different "global pre-steps":

  This possibility has been proved to have negative consequences in terms of the performance (see table 6.4 and Figure 6.2) and therefore will be dismissed.

  In addition to that, it only introduces changes on the length of the sequence considered, but not in the complexity (number of LMI). As a result of that, the time improvement will be small.

- Consider \(3 - CM_2\) instead of \(3 - CM_3\)

  This possibility will be studied next. Roughly speaking, we can say that it will simplify our scheme, going from the 12 simultaneous LMI of \(3 - CM_3\) to just 8 on \(3 - CM_2\). However, this will have consequences on the results, as we will see.

- Make the sequence shorter.

  This possibility will be studied next. Roughly speaking we can say that, again, this strategy will not vary the complexity of the method, but the length of the sequence considered. Therefore, as we will see, the time improvement will be small.

**Making the sequence shorter**

The \(\gamma\ vs. \delta\) relationship is a decreasing asymptotic plot. Once we reach the region where it is almost flat, there is no risk on switching from the method \(3 - CM\) to the regular variable sequential method. Since the variable sequential method requires less calculations per each \(\delta\), that should make the process faster on the last values of the sequence.

\(^7\)Which might not be the main issue, but a consequence of it, although in any case is a parameter interesting to be reduced.
The problem is how to know where can we switch without negative consequences on the performance. What we are going to do is to measure the improvement on the performance from the last \( \delta \) solved to the current \( \delta \). If it is below a certain level, then we will consider that we can make the switch.

On the next table we show the results for the \( R^5 \) system presented at the beginning of this section:

Table 6.8: Results of speeding up the method \( 3 - CM \) by introducing a switch on the sequence \( (R^5) \). The earlier we switch from \( 3 - CM \) to \( P_{\text{sequential}} \), the earlier we finish, but we make the minimization worse.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>% of improvement</th>
<th>( \delta ) for the switch</th>
<th>( \gamma_{\text{final}} )</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 3 - CM_3 )</td>
<td>0%</td>
<td>( \emptyset )</td>
<td>0.1957</td>
<td>8.953 s</td>
</tr>
<tr>
<td></td>
<td>0.5%</td>
<td>( \delta = 70 )</td>
<td>0.1957</td>
<td>8.938 s</td>
</tr>
<tr>
<td></td>
<td>1%</td>
<td>( \delta = 40 )</td>
<td>0.2068</td>
<td>8.844 s</td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>( \delta = 15 )</td>
<td>0.2690</td>
<td>8.500 s</td>
</tr>
<tr>
<td>( 3 - CM_2 )</td>
<td>0%</td>
<td>( \emptyset )</td>
<td>0.2009</td>
<td>4.593 s</td>
</tr>
<tr>
<td></td>
<td>0.5%</td>
<td>( \delta = 70 )</td>
<td>0.2012</td>
<td>4.531 s</td>
</tr>
<tr>
<td></td>
<td>1%</td>
<td>( \delta = 40 )</td>
<td>0.2121</td>
<td>4.375 s</td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>( \delta = 15 )</td>
<td>0.2756</td>
<td>4.234 s</td>
</tr>
<tr>
<td>( P_{\text{global constant}} )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>0.2067</td>
<td>13.469 s</td>
</tr>
</tbody>
</table>

According to table 6.8, we can reduce the total time for calculations by considering the described switch. However, this reduction is really small, and we cannot go beyond 0.5% if we do not want to make the results worse.

As we said before, this is because this modification only makes the sequence shorter, but does not vary the complexity of the systems of LMI that we are solving.

Therefore, we can conclude that this strategy is not very useful, although it might have more importance when dealing with higher order systems. But nevertheless, it will never suppose dramatic improvements in terms of computational time.
On the other hand, as we can see on the table, the fact of considering $3-CM_2$ instead of $3-CM_3$ does really improve the total time for calculations at a reasonable price in terms of performance \(^8\). On the following lines we will discuss this fact in detail.

3–$CM_2$ instead of 3–$CM_3$

According to table 6.8, we can consider 3–$CM_2$ instead of 3–$CM_3$, since both of them are better than $P_{\text{variable}}^{\text{sequential}}$. Specially if we concede preference to the complexity of the method and the total time of calculations before the performance.

However, we have to make sure that this situation is not something casual. For this purpose we show the same table for our usual $R^2$ system:

$$\alpha = 1, \quad u_{lim}^2 = 1000^2, \quad w_{\max}^2 = 1000^2,$$

$$A = \begin{pmatrix} 0 & 1 \\ -10 & -10 \end{pmatrix} \quad B_1 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} \quad B_2 = \begin{pmatrix} 0 & 6 \end{pmatrix} \quad C = \begin{pmatrix} 5 & 0 \end{pmatrix} \quad D = 0$$

$$\delta = \begin{bmatrix} 1 & 1.1 & 1.2 & 1.4 & 1.7 & 2.1 & 2.5 & 2.8 & 3.1 & 3.5 & \ldots \\ \ldots & 4.3 & 5.5 & 6 & 8 & 10 & 15 & 40 & 70 & 110 & 150 \end{bmatrix}$$

Table 6.9 confirms the idea that the lower complexity of 3–$CM_2$ (8 LMI per sequential step) instead of 3–$CM_3$ (12 LMI per sequential step) supposes a much better improvement on the computational time than the fact of considering the switch to the variable sequential scheme. Moreover, in both cases Combined Methods improve the results from the constant global scheme; avoiding the so called unexpected result.

\(^8\)3–$CM_2$ still provides better results than the constant global scheme
Table 6.9: Results of speeding up the method $3 - CM$ by introducing a switch on the sequence ($R^2$). The earlier we switch from $3 - CM$ to $P^{\text{sequential}}_{\text{variable}}$, the earlier we finish, but we make the minimization worse.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>% of improvement</th>
<th>$\delta$ for the switch</th>
<th>$\gamma_{\text{final}}$</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3 - CM_3$</td>
<td>0%</td>
<td>$\emptyset$</td>
<td>0.2524</td>
<td>1.920 s</td>
</tr>
<tr>
<td>0.5%</td>
<td>$\delta=40$</td>
<td>0.2524</td>
<td>1.922 s</td>
<td></td>
</tr>
<tr>
<td>1%</td>
<td>$\delta=40$</td>
<td>0.2524</td>
<td>1.922 s</td>
<td></td>
</tr>
<tr>
<td>2%</td>
<td>$\delta=15$</td>
<td>0.2524</td>
<td>1.859 s</td>
<td></td>
</tr>
<tr>
<td>$3 - CM_2$</td>
<td>0%</td>
<td>$\emptyset$</td>
<td>0.2629</td>
<td>1.281 s</td>
</tr>
<tr>
<td>0.5%</td>
<td>$\delta=40$</td>
<td>0.2629</td>
<td>1.265 s</td>
<td></td>
</tr>
<tr>
<td>1%</td>
<td>$\delta=40$</td>
<td>0.2629</td>
<td>1.265 s</td>
<td></td>
</tr>
<tr>
<td>2%</td>
<td>$\delta=15$</td>
<td>0.2629</td>
<td>1.265 s</td>
<td></td>
</tr>
<tr>
<td>$P^{\text{global}}_{\text{constant}}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>0.2951</td>
<td>0.672 s</td>
</tr>
</tbody>
</table>

Conclusions from the complexity/computational time study

As a conclusion, we can say that our best choices among the Combined Methods are $3CM_3$ and $3 - CM_2$. Depending on if we are more interested on speed or performance.

In both cases we can consider a switch to variable sequential method when the performance does not improve more than a 0.5% in order to get a little extra improvement on the total time of calculation.

But in any case, the really important point of this new methods $3 - CM_3$ and $3 - CM_2$ is that they can provide us with results close to those from the $P^{\text{global}}_{\text{variable}}$ method. And this is reflected on their low computational times.
6.3 Influence of the weights on the new method

There is still one important parameter to analyze on this chosen version of the Combined Methods (3 – CM): the weights for the global pre-step.

In the Global Methods, we introduced the concept of weight in order to make the resolution to be focused on certain parts of the sequence of \( \delta \), and therefore, improving the performances along them. However, the price was a worse result on the other areas. In the case of the Combined Methods, and particularly 3 – CM, we can use the weights to make our global pre-step to be focused on the first \( \delta \) (the one whose results are going to be the final answer) or to be focused on the following values (whose role is only to give a perspective of what the method will have to deal with on the next steps of the sequence).

As we have seen, the advantage of the Combined Methods is this possibility of giving a result for each \( \delta \) while taking into account the next situations that will come with the following \( \delta \). It is easy to see that the more cases we consider on the global pre-step, the better results we will have (the limit case would be doing a global pre-step with all the values of the sequence, and that would be equivalent to the Variable Global method).

Nevertheless, we have seen that if we consider more than 3 \( \delta \) for the global pre-step, the complexity grows and method becomes too slow. This fact sets up a limit on the capability of our method to "foresee" the following values of \( \delta \) and their requirements. We cannot consider a lot of \( \delta \) on each global pre-step without falling on the same problem as the \( P^\text{Global}_{\text{variable}} \) scheme. However, another way to give the method a bigger concern about the next values of \( \delta \) could be through the weights, rather than increasing the number of cases considered.

\[9\] In fact, on the last section we proposed considering only 2 \( \delta \) (the so called 3 – CM\(_2\)) in order to reduce this complexity and speed up the process.
According to this, we would expect the method $3 - CM_2$ to match the results obtained by $3 - CM_3$ through the use of appropriate sets of weights. The same way, we expect the results of $3 - CM_3$ to improve and get closer to the results of Variable Global Method.

**Description of Figure 6.6**

On Figure 6.6 we can see the representation of our main 4 methods ($P_{global\ variable}$, $P_{global\ constant}$, $3 - CM_2$ and $3 - CM_3$) with a homogeneous weight distribution (as we have been doing in most of the previous cases) and with an emphasis on the last weight.

That is to say, for the second case both global methods have used a weight distribution as the following:

$$\omega = [ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 10 ]$$

Whereas for the first case, they used a homogeneous distribution of 1.

On the other hand, the Combined methods used a homogeneous distribution of 1 for their *Global pre-steps* for the first case. And for the second they used:

$$\omega_{CM_2} = [ 1 \ 10 ] \quad \omega_{CM_3} = [ 1 \ 1 \ 10 ]$$

On the global methods, we see that the improvement is much smaller, and has worse consequences at the beginning of the sequence. The more we focus on the last values, the worse performance we will have on the first values. This effect can be accentuated if we weight more not only the last value, but some of the last values.

However, the Combined Methods improve greatly on the last values and do not get that worse at the beginning. The only area where they show a worse performance is at the middle of the sequence of $\delta$, where they both have slightly higher performances.
associated. Nevertheless, as we consider their weighted versions, they improve a lot and get really close to what the Global Methods offer.

This Figure summarizes how the Combined Methods, and specially $3-CM$, present a very good performance if we compare them with what so far has been the most powerful (and slow) method: $P_{\text{sequential variable}}$

Conclusions of this study

The use of non homogeneous weights for the $3-CM$ schemes results on a noticeable improvement of the performances. For the Global methods, the fact of focusing the weight on the last $\delta$ of the sequence also results on better performances, but in this case the improvement is not that important.

Notice that in both cases (Global Schemes and $3-CM$ schemes) the improvement at the end of the sequence of $\delta$ comes together with worse results at the beginning. However, since we are interested mainly on the last values of the sequence, this fact can be accepted.

As a final conclusion we can say that an appropriate tuning of the weight constants may allow us to get better results with $3-CM_2$ than what we got with homogeneous weights on $3-CM_3$ (see Fig. 6.6 for details)

However, it is not clear how would be do this tuning of the weights before solving the system. Nevertheless, it is a possibility that is worth to consider in the future.
Figure 6.6: Influence of the weights over the Global Methods and the Combined Methods
Chapter 7

Input Scheduling criteria with $\dot{r} \leq 0$

So far we have always considered a State Feedback controller. This implied that we choose our controller after measuring the state space variables and see which two ellipsoids bound this state (see section 4.4.1 for details).

However, the fact is that we have assumed some conservatism throughout the derivation of our LMI set. Particularly for the saturation LMI $^1$, we have made strong assumptions that allow us to say that there will be no saturation inside the ellipsoid, but in most of the cases we will still be able to use the same controller beyond the ellipsoid without risk of saturation.

To prove this statement we recall Figures Fig3.8, Fig3.9, Fig 3.10 and Fig3.11, where we show the resulting ellipsoids, as well as the saturation limits, for several values of $\delta$. As a quick reference we show again the last case on Figure 7.1 (see next page).

These plots correspond to the first $P_{\text{sequential constant}}$ method, which is the method with greater conservatism. On the other hand, our last method $3- CM$ is much less conservative, and therefore is much more unusual to see that the ellipsoids are not tangent to the lines that represent the saturation limits.

$^1$See section 2.0.2 for details
Figure 7.1: Ellipsoids and Sat. limits for $P_{\text{sequential}}$ and $\delta = 5$, compared to $\delta = 1$

However, in spite of the fact that for this $R^2$ examples the $3 - CM$ method seems to give always tangent saturation bounds to the ellipsoids, it is not that likely to do so with higher order systems ($R^5$, $R^9$, etc.). And even then, we will always have an area of the state space beyond the limits of the ellipsoid where the gain does not lead to saturation.

This subspace is related to the expression: $Kx = 0$, which for the case that we always represent ($R^2$) will be a line on the center of the "strip" defined by the two saturation lines, which can be also represented as:

$$Kx = u_{lim} \quad \text{and} \quad Kx = -u_{lim}$$

Therefore, on this chapter we will propose to extend the use of our controller gain beyond the ellipsoids based on $\delta$-scaled disturbances. Instead of this, we will develop a Gain Scheduling method based just on the saturation of the actuator. But before introducing any modification, we have to be aware of all the consequences from solving our current system of LMI.
7.1 Rethinking our previous set of LMI

Recall from the previous chapters that all our ellipsoids $\epsilon_k$, where $k = 1, 2, 3...$ enforced the following properties to the states that were inside them:

- **Saturation LMI:**
  
  The associated controller did not saturate with a scaled version of the original maximum disturbance $\frac{w_{\text{max}}}{\delta}$.

  $$M_i = \begin{pmatrix} Q_k & F_k^T \\ F_k & r_k \end{pmatrix} > 0 \text{ where } r_k = \delta^2_k \frac{u_{\text{lim}}^2}{w_{\text{max}}^2} \tag{7.1}$$

- **Performance LMI:**
  
  Maximum performance was given by the corresponding $\gamma_k$.

  $$J_k = \begin{pmatrix} A Q_k + Q_k A^T + B_2 F_k + F_k^T B_2^T & B_1^T & Q_k C_1^T + F_k^T D^T \\ B_1 & -\gamma_k^2 & 0 \\ C_1 Q_k + D F_k & 0 & -I \end{pmatrix} < 0 \tag{7.2}$$

- **Nesting condition LMI:**
  
  The ellipsoid corresponding to $\epsilon_k$ was nested inside the bigger ellipsoid $\epsilon_{k-1}$ when both were represented at the same level set $\hat{\omega}$ of the corresponding Lyapunov functions:

  $$\epsilon_k^* = \left\{ x : x^T P(r_k) x \leq \hat{\omega} \right\} \text{ and } \epsilon_{k-1}^* = \left\{ x : x^T P(r_{k-1}) x \leq \hat{\omega} \right\} \rightarrow \epsilon_k^* \subset \epsilon_{k-1}^*$$

  And therefore:

  $$P^{-1}(r_k) < P^{-1}(r_{k-1}) \tag{7.3}$$

- **Performance improvement LMI:**
  
  The associated performance was smaller than on the bigger ellipsoid.

  $$\gamma(r_k) \leq \gamma(r_{k-1}) \tag{7.4}$$
The case of the biggest ellipsoid, $\epsilon_1$ was slightly different. Here the properties were:

- **Saturation LMI:**

  In this case, the disturbance considered is the real maximum disturbance, without any scaling parameter $\delta$.

  \[
  M_1 = \begin{pmatrix}
  Q_1 & F_1^T \\
  F_1 & r_1
  \end{pmatrix} > 0 \quad \text{where} \quad r_1 = 1 \frac{u_{\text{lim}}}{w_{\text{max}}^2}
  \]  
  \[ (7.5) \]

- **Performance LMI:**

  This condition is exactly the same as before, so: \( J_1 < 0 \)

- **Reachable set LMI:**

  Given its associated gain $K_1$, and assuming that the state is inside $\epsilon_1$, it would never go beyond the limits of $\epsilon_1$ in the state space.

  \[
  \begin{pmatrix}
  AQ_1 + Q_1A^T + \alpha Q_1 + B_2F_1 + F_1^TB_2^2 & B_1 \\
  B_1^T & -\alpha
  \end{pmatrix} < 0
  \]  
  \[ (7.6) \]

  Leaving aside the obvious fact that since $\epsilon_1$ is the biggest ellipsoid, we do not have to care about nesting it or improving a previous performance; we can say that the only difference between $\epsilon_1$ and the rest of ellipsoids $\epsilon_2... \epsilon_n$ is the reachable set condition.

**Several important conclusions to recall and consider:**

As a result of this set of conditions, the controller gains for $\epsilon_2... \epsilon_n$ are calculated without requiring the system to remain inside a specific reachable set. (fact 1). That is to say: the system will still be stable in the sense of Lyapunov when using $K_2, K_3... K_n$, but we have no warrantee that it will stay on the $V \leq w_{\text{max}}^2$ region.

---

2The only ellipsoid that considers the real maximum expected disturbance $w_{\text{max}}^2$, since all the others consider scaled versions of it $\frac{w_{\text{max}}^2}{\delta}$.

3Otherwise it would not have a finite $L_2$ represented by $\gamma$. 

106
Notice also that the performance LMI is not related to the scale parameter $\delta$ that allows us to get the different ellipsoids for $\epsilon_2...\epsilon_n$. In fact, the performance LMI is derivated independently of any set of the space state, and applies for all states $^4$.

Therefore, for an ellipsoid $\epsilon_k$ in between $\epsilon_2$ and $\epsilon_n$, this means that the calculated performance $\gamma_k$ applies not only for $\epsilon_k$ itself:

$$\epsilon_k = \left\{ x : V_k(x) = x^TP_kx \leq \frac{w_{\text{max}}^2}{\delta_k^2} \right\}$$

But also to the whole state space where we do not saturate. (fact 2)

On the other hand, we already know that the saturation limits are beyond the borders of $\epsilon_k$ (fact 3) independently of how accurate the solution method was (even in the case where the actuator saturation lines were tangent to the ellipsoid).

As a result of that we can say that if we use the corresponding gain $K_k$ beyond the limits of $\epsilon_k$, provided we do not saturate, we will keep the same maximum performance. (fact 4 - final conclusion)

This possible modification of using $K_k$ beyond $\epsilon_k$ is important because it implies interesting consequences. The fact that using the gains beyond the initial ellipsoids would allow us to use higher gains for longer time, and this should speed up the time evolution of the system.

However, this modification needs to be improved before being considered. Imagine for a second that we have the situation shown on Figure 7.2.

We would start inside $\epsilon_{15}$, and according to this new conclusions we would use $K_{15}$ even when we are outside $\epsilon_{15}$ but still inside the saturation limits of $K_{15}$ (the narrow strip pointed by the to thick arrows on the Figure).

$^4$Assuming that we do not have saturation, which is considered on a different LMI.
Figure 7.2: Ellipsoids and sat. limits for $3-CM_3$ at $\delta = 1$ and $\delta = 15$.

The performance $\gamma_{15}$ would still be guaranteed inside the strip, but since there is no reachable set associated to $K_{15}$ we might leave our maximum reachable set $\epsilon_1$ if we move inside the indicated strip, without saturating the controller and with $K_{15}$.

In order to avoid this situation, before considering this new criteria for scheduling, we have to define a particular reachable set for each of the controller gains $K_k$. To do this we will incorporate our Reachable Set LMI to the LMI system solved for each $\delta_k$ (see next section 7.2 for mathematical details).

Once we will do this, since the Reachable set condition does not include the parameter $\delta$, the reachable sets that we will obtain for $K_k$ will be defined by the ellipsoid:

$$
\epsilon_k^* = \{ x : x^TQ_k^{-1}x \leq w_{\text{max}}^2 \}
$$

Instead of the $\delta$-scaled level set that we used to consider before for our nested ellipsoids:

$$
\epsilon_k = \{ x : x^TQ_k^{-1}x \leq \frac{w_{\text{max}}^2}{\delta^2} \}
$$

This new set $\epsilon_k^*$ will be much bigger than $\epsilon_k$, since its not $\delta$-scaled. But because of the nesting condition, it is still nested inside all the previous $\epsilon_1... \epsilon_i$, where $i \leq k$. 

108
Considering this modifications, the new situation would be as shown on Fig. 7.3:

![Diagram showing ellipsoids and saturation limits](image)

Figure 7.3: Ellipsoids and sat. limits for $3 - CM_3$ at $\delta = 1$ and $\delta = 15$. Notice how the non scaled version of $\epsilon_{15}$ (described as $\epsilon^*_{15}$) is nested inside $\epsilon_1$.

As you can see, here we can start on $\epsilon_{15}$ the same way we did before. However, we can guarantee that $K_{15}$ will keep the system inside $\epsilon^*_{15}$, which is a nested set.

When we reach the saturation limit, which in general is inside $\epsilon^*_{15}$, we would have to switch to the next $K$ and reachable set, which in the picture would be $\epsilon_1$ and $K_1$.

This new way of scheduling implies an important simplification for the implementation of the controller. Changing yo a spline interval based only on the actuator input and the saturation limits is really direct and simple. Specially compared with evaluating the state of the system "$x$" $^6$, and comparing the Lyapunov function "$V = x^TPx"$ with the boundaries of the ellipsoids, which is what we used to do before.

On section 7.2 we will detail the way we introduce the Reachable Set condition on the system of LMI, so we can implement this new criteria.

---

$^5$And if it is outside, as it happens with $\epsilon_1$, we are guaranteed that we will never saturate nor leave the ellipsoid.

$^6$Which in many cases may not be trivial at all.
7.2 Modifications on the system of LMI

The main conclusion from previous section 7.1 is the new idea of scheduling based on multiple reachable sets and saturation bounds, rather than the old idea of scaled ellipsoids where we did not saturate.

As we also said on section 7.1, in order to apply this new criteria we have to solve the Reachable Set LMI together with all the other LMI that we considered before, for each value $\delta_k$ of the sequence. As a result of that, we would have a reachable set LMI for each $Q_k = P_k^{-1}$ and each controller gain $K_k$, and not only for $Q_1$ and $K_1$ as we used to have before.

The expression for this LMI at the $k-th$ value of the sequence ($\delta_k$) would be the next:

$$
\begin{pmatrix}
AQ_k + Q_k A^T + \alpha Q_k + B_2 F_k + F_k^T B_2^2 & B_1 \\
B_1^T & -\alpha
\end{pmatrix} < 0
$$

And since this LMI does not have any relationship with $\delta_k$, its conclusions would apply for the non scaled version of the resulting ellipsoid $\epsilon^*_k = \{x : x^T Q_k^{-1} x \leq w_{max}^2\}$. As a result of that, we would have that the designed gains ($K_k$) would not allow the system to go beyond the limits of $\epsilon^*_k$.

To show this we recall the derivation of this LMI from section 2.0.1 where, after the change of variable $Q_k = P_k^{-1}$ and separating $F_k = K_k Q_k$, we have:

$$
\begin{pmatrix}
P_k A + A^T P_k + \alpha P_k + P_k B_2 K_k + K_k^T B_2^T P_k & P_k B_1 \\
B_1^T P_k & -\alpha
\end{pmatrix} < 0
$$

And then, after pre and post multiplying by $\begin{pmatrix} x & w \end{pmatrix}$ we get:

$$
0 > x^T (P_k A + A^T P_k + \alpha P_k + P_k B_2 K_k + K_k^T B_2^T P_k) x + w^T B_1^T P_k x + x^T P_k B_1 w - \alpha \omega^2
$$
At this point we consider that the disturbance level $w$ is below what we consider as maximum admissible value, $w_{\text{max}}$, and we get:

$$0 > x^T (P_k A + A^T P_k + \alpha P_k + P_k B_2 K_k + K_k^T B_2^T P_k) x + w^T B_1^T P_k x + x^T P_k B_1 w - \alpha w_{\text{max}}^2$$

And from here, after grouping terms and consider the state space system description (expression 2.2), we can derive the final differential equation for the condition expressed on the initial LMI:

$$0 < \dot{x}^T P_k x + x^T P_k \dot{x} + \alpha (x^T P_k x - w_{\text{max}}^2)$$

And after allowing $\dot{P}_k < 0$

$$0 < \dot{x}^T P_k x + x^T P_k \dot{x} + x^T \dot{P}_k x + \alpha (x^T P_k x - w_{\text{max}}^2)$$

Therefore, LMI 7.7 is imposing that the resulting Lyapunov function (based on $P_k$) with the resulting controller gain ($K_k$) does not go beyond the level set indicated by

$$V_k = x^T P_k x \leq w_{\text{max}}^2$$

Which is what we considered before as the non-scaled definition of the ellipsoid $\epsilon_k^*$. That is to say:

$$\epsilon_k^* = \{x : V_k(x) = x^T P_k x \leq w_{\text{max}}^2\}$$

instead of the regular scaled ellipsoid considered so far:

$$\epsilon_k = \{x : V_k(x) = x^T P_k x \leq \frac{w_{\text{max}}^2}{\delta_k^2}\}$$

And to do this, we only assume that the maximum disturbance will be smaller that $w_{\text{max}}$ and the derivative of $P$ will be negative:

$$w(t) < w_{\text{max}} \quad \text{and} \quad \dot{P} < 0$$
7.2.1 Spline description for the Reachable Set LMI

On subsection 4.3.3 we explained that the reachable set LMI did not need to be described in terms of spline function since it was only used for the very outer ellipsoid. However, since now we have the reachable set LMI for each of the inner ellipsoids, we will have to detail the spline description for this LMI. As we did for the nonsaturation LMI and the performance LMI, on sections 4.3.1 and 4.3.2.

Expressions for the spline description

Recall from section 4.3 the main expressions used to perform the spline description of our set of LMI:

- Spline function for $F(r)$ and $Q(r)$:
  \[
  F(r) = F_n + (F_{n+1} - F_n) \frac{r - r_n}{r_{n+1} - r_n} = F_n + \Delta F_n \lambda \\
  Q(r) = ... = Q_n + \Delta Q_n \lambda
  \]
  
  Where $r$ is defined by: $r_n = \delta_n \frac{w_{min}}{w_{max}}$

- Generic product of two spline functions $R(r)$ and $T(r)$:
  \[
  T(r)R(r) = (1 - \lambda)^2 T_n R_n + \lambda^2 (T_{n+1} R_{n+1}) + \\
  + \lambda(1 - \lambda)[(T_{n+1} - T_n)R_n + T_{n+1}(R_{n+1} - R_n) + 2T_n R_n]
  \]

- When $T(r)=T(r)$ and $R(r)=1$, this generic product turns into an alternative expression for a spline function:
  \[
  T(r) = (1 - \lambda)^2 T_n + \lambda^2 T_{n+1} + \lambda(1 - \lambda)[T_n + T_{n+1}] \quad (7.8)
  \]
And once we have shown again these expressions, we apply them to the Reachable Set LMI in order to get its spline description.

Let the Reachable set LMI be:

\[ N(r) = \left( \begin{array}{cc} AQ(r) + Q(r)A^T + \alpha Q(r) + B_2 F(r) + F^T(r)B_2^T & B_1 \\ B_1^T & -\alpha \end{array} \right) < 0 \]

And using expression 7.8 for \( N(r) \), we can re-write the LMI as:

\[ N(r) = (1 - \lambda)^2 N_n + \lambda^2 N_{n+1} + \lambda(1 - \lambda)(N_n + N_{n+1}) < 0 \quad (7.9) \]

c Where this new variables correspond to:

\[ N_n = \left( \begin{array}{cc} AQ_n + Q_n A^T + \alpha Q_n + B_2 F_n + F_n^T B_2^T & B_1 \\ B_1^T & -\alpha \end{array} \right) < 0 \]

And since \( \lambda < 1 \), \( N_n < 0 \) and of course \( N_{n+1} < 0 \), we can introduce some conservatism on expression 7.9 and cancel the negative terms to get:

\[ 0 > \lambda(1 - \lambda)(N_n + N_{n+1}) > N(r) \]

\[ 0 > N_n + N_{n+1} \]

Which is equivalent to say \( N_n < 0 \). And as a result of that, the whole LMI \( N(r) < 0 \) is included inside \( N_n < 0 \).

Where as always, the subindex ”n” corresponds to the solution of the system of LMI for \( \delta_n \) (or what is equivalent, \( r_n \)).
7.3 New scheduling areas on the State Space

As we have seen throughout this thesis, the idea of Gain Scheduling was based on defining different areas of the state space region (ellipsoids) where we want our system to be, and design a controller for each of the bounds of these areas. Once we had this, we positioned the state of the system on the state space, and calculated the final controller gain by splinning in between the gains of the two ellipsoids that bound the area.

After considering this new modifications on our proceeding, we have that now the controller gains are not associated only to the ellipsoids $\epsilon_k$, but to a bigger area delimited by the corresponding saturation limit and the reachable sets ($\epsilon_k^*$) \(^7\). As a result of that, the areas associated to each controller gain are quite different, as we can see on Figure 7.4, where we represent:

- On the first line we show the space state distribution from the old criteria:

  A group of nested ellipsoids according to the LMI system shown on section 7.1. For each value of $\delta$ we have a different gain an a smaller ellipsoid (as $\delta$ grows we consider lower level sets of the corresponding Lyapunov function).

  Notice how the saturation limits of the controller are always outside the ellipsoid, since we are waranteed not to saturate inside this ellipsoids.

- On the second line we show the new space state distribution:

  A group of nested reachable set ellipsoids and the saturation lines corresponding to their controllers. All ellipsoids are represented as level sets of the corresponding Lyapunov function at $V = w_{max}^2$, so they do not depend on $\delta$.

  On this second line we also represent the ellipsoids at the $V = \frac{w_{max}^2}{\delta^2}$ level set \(^8\) (with thin lines) which are the ellipsoids equivalent to the ones of old methods.

\[^7\] Also referred as nonscaled ellipsoids

\[^8\] The “scaled” version of the ellipsoid
Figure 7.4: Comparing the traditional criteria of dividing the state space on ellipsoids associated to a scale parameter $\delta$ with the new criteria based on Reachable sets and Saturation lines.
The main idea of Figure 7.4 is to see the areas corresponding to each set of values that we get after solving the system of LMI. Areas are marked with thick solid lines. For the first case they are nice ellipses whereas for the second, they are delimited by an ellipse and the saturation lines of the controller.

Notice that to get the final value of the controller, the new proceeding will be based on performing the spline from the actuator input rather than the state of the system.

Summary of the old scheduling scheme

As we said before, the old methods measured the state of the system \(x(t)\), and placed it in an area delimited by two ellipsoids \(b_I\) and \(c_I\), for instance).

After that, they solved by iteration process the value of \(r^9\) that had associated a \(Q_{b_I} > Q(r) > Q_{c_I}\) such that the resulting ellipsoid \(\epsilon(r)\) contained exactly \(x(t)\).

Finally, with this value of \(r\) they gave us the controller value from the expression:

\[
K(r) = K_{b_I} + \frac{K_{c_I} - K_{b_I}}{r_{c_I} - r_{b_I}} (r - r_{b_I}) \tag{7.10}
\]

New scheduling scheme

On the other hand, the new scheduling scheme will choose two consecutive controller gains \((K_{b_{II}}\) and \(K_{c_{II}},\) for instance) such that:

\[
K_{c_{II}}x(t) > u_{saturation} \quad \text{and} \quad K_{b_{II}}x(t) < u_{saturation}
\]

And after an iteration with expression 7.10, it will find a controller gain \(K(r)\), such that:

\[
K_{b_{II}} < K(r) < K_{c_{II}}
\]

And also:

\[
K(r)x(t) = u_{saturation}
\]

\(^9\)Remember that \(r = \delta^2 \frac{\pi^2}{w_{max}^2}\), and it was a term equivalent to \(\delta\) but with some constants multiplying
The evolution of the system with the new scheduling scheme

According to one of our assumptions, the system will start inside $c_{II}$ since we consider initial conditions to be zero $^{10}$. As the perturbation $w(t)$ appears and excites the system, the state will move inside the area marked as $c_{II}$.

This area marked as $c_{II}$ is mainly the reachable set ellipsoid for $w < w_{max}$ modified by imposing the corresponding saturation limits of $K_{c_{II}}$. If the disturbance does not go beyond this maximum admissible value, the controller $K_{c_{II}}$ is designed to keep the system inside this Reachable set (the original ellipsoid). However, even though the disturbance does not exceed the maximum value we might go out of $c_{II}$ through the saturation lines. This boundaries are not related to the reachable set, and therefore the controller $K_{c_{II}}$ is not required to consider them as a limitation.

Figure 7.5: Detail of two scheduling areas of the new criteria.

Therefore, once the system gets out of $c_{II}$ through the saturation lines, we go to the next scheduling area of the state space, $b_{II}$, and we start splitting the value of the controller between $K_{c_{II}}$ and $K_{b_{II}}$, as we described before. Ultimately, the biggest scheduling area is purely a Reachable Set ellipsoid, so there is no way that $x(t)$ can go outside it if the disturbance does not exceed $w_{max}$.

$^{10}$That is: $x(t = 0) = 0$
Table 7.1: Summary of the Input Scheduling Criteria time evolution.

<table>
<thead>
<tr>
<th>Important remarks:</th>
</tr>
</thead>
</table>
| Saturation bounds are outside the *scaled ellipsoid* : \( \epsilon_k = \left\{ x^T P_k x \leq \frac{w_{\text{max}}^2}{\delta_k^2} \right\} \)  
(See section 2.2, expression 2.17, for details) |
| Reachable set conditions refer to the *non-scaled ellipsoid* \( \epsilon_k^* = \left\{ x^T P_k x \leq w_{\text{max}}^2 \right\} \)  
(See section 2.2, expression 2.13, for details) |
| It is easy to check that \( \epsilon_k \subset \epsilon_k^* \)  
For all \( r_k \) but for \( r_1 = 1 \), where \( \epsilon_1 = \epsilon_1^* \) |

<table>
<thead>
<tr>
<th>Time evolution alternatives:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start in the smaller region</td>
</tr>
<tr>
<td>(( C_{II} ), in the case of figure 7.5)</td>
</tr>
<tr>
<td>The system moves alongside the “saturation strip” associated to ( K_{CII} ).</td>
</tr>
</tbody>
</table>
| It may go beyond \( E_{CII} \) if : \( w(t) > \frac{w_{\text{MAX}}}{\delta_{CII}^2} \)  
But it cannot go beyond \( E_{CII}^* \)  
(Reachable set associated to \( K_{CII} \) and \( w_{\text{MAX}} \)) |
| The system crosses the saturation bounds of \( K_{CII} \) and then we switch to \( K_{BII} \) |
| If we keep on switching to larger areas, at some point we will have the case corresponding to \( r_1 \).  
For this case we will not be able to cross the saturation bounds, so we will be limited exclusively by the reachable set |
7.3.1 Results of this new considerations and conclusions

After having proved that we can incorporate the reachable set condition to the LMI system that described the inner ellipsoids, we will solve this new system and compare the results. Initially, the fact of adding an extra LMI to our system might make our results to get worse. However, if we compare the two solutions we can see that after all, there is no influence when we solve the system with the $3 - CM_3$ method:

The system is our usual system for practical tests:

$$\alpha = 1, \quad u_{\text{lim}} = 1000^2, \quad w_{\text{max}}^2 = 1000^2,$$

$A = \begin{pmatrix} 0 & 1 \\ -10 & -10 \end{pmatrix}$, $B_1 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$, $B_2 = \begin{pmatrix} 0 & 6 \\ \end{pmatrix}$, $C = \begin{pmatrix} 5 & 0 \end{pmatrix}$, $D = 0$

And the results are shown on table 7.2:

Table 7.2: Comparing the effects of including the Reachable Set LMI for the inner ellipsoids ($3 - CM_3$). Notice that the results are basically the same.

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>Performance</th>
<th>$\delta$</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0973389</td>
<td>1</td>
<td>1.0973389</td>
</tr>
<tr>
<td>1.5</td>
<td>0.8971308</td>
<td>1.5</td>
<td>0.8971307</td>
</tr>
<tr>
<td>2.5</td>
<td>0.6974578</td>
<td>2.5</td>
<td>0.6974576</td>
</tr>
<tr>
<td>5</td>
<td>0.4955812</td>
<td>5</td>
<td>0.4955811</td>
</tr>
<tr>
<td>7.5</td>
<td>0.4058491</td>
<td>7.5</td>
<td>0.4058490</td>
</tr>
<tr>
<td>10</td>
<td>0.3522465</td>
<td>10</td>
<td>0.3522463</td>
</tr>
<tr>
<td>15</td>
<td>0.2936645</td>
<td>15</td>
<td>0.2936645</td>
</tr>
<tr>
<td>20</td>
<td>0.2766676</td>
<td>20</td>
<td>0.2766675</td>
</tr>
<tr>
<td>50</td>
<td>0.2566115</td>
<td>50</td>
<td>0.2566114</td>
</tr>
<tr>
<td>100</td>
<td>0.2535730</td>
<td>100</td>
<td>0.2535730</td>
</tr>
</tbody>
</table>

$^{11}$Before including the reachable set condition on $\epsilon_2 \ldots \epsilon_n$ and after

$^{12}$The only insignificant difference is probably given by the rounding of the values and the precision error of the LMI solver, since it shows at the 7th decimal digit.
Is it really worth to include this modification?

As we can deduce by comparing Figure 7.1 and Figure 7.2, the first case was more "receptive" to implement this modification than the second. The number of assumptions made and the resulting conservatism on the results made clear that the actuator was clearly underused if we base our gain switch criteria just on the limits of the ellipsoids.

However, the case shown on Figure 7.2 is quite different. Here the saturation limit is always tangent to the ellipsoid, and therefore the new region considered \(^{13}\) is minimal... but it is still there. And, as a matter of fact, we do not have any reason to think that it is going to be always minimal when considering \(R^n\), for \(n > 2\). In the case shown on Figure 7.2, the improvement given by this new modification might not be that dramatic, but nevertheless it will not have a negative influence on our results.

Therefore, we can conclude that the improvement given by this new modification may be different according to the scheme of resolution used and the particular situation, but in all the cases it will help to get a faster time evolution of the system.

Moreover, it will make our control loop easier to close, since we will not be forced to measure the state of the system \(^{14}\) to choose our controller gain. But, on the other hand, this new strategy imposes us the need to redefine the way we scheduled the controller gains on the state space, since now we do not rely on the ellipsoids \(\epsilon_k\) to choose them, but in the saturation limits and the reachable sets defined by the non scaled ellipsoids, \(^{15}\) \(\epsilon^*_k\), (see next section for details).

\(^{13}\)The new region considered for the controller to be used is, a part from the ellipsoid \(\epsilon_{15}\) that we considered before, all the non scaled ellipsoid \(\epsilon^*_{15}\) that is inside the saturation lines.

\(^{14}\)Which is something that eventually might be difficult to measure, since the state variables do not always have a physical meaning.

\(^{15}\)Now the area associated to each controller, defined by \(\epsilon^*_k\) and the saturation lines, is a stable domain in the sense of Lyapunov -it is a reachable set- except for the boundaries defined by the saturation lines. These are technically the only boundaries that the system can cross and they will be the only reason for changing to a lower controller gain -and therefore doing the scheduling and the splinning-.
Chapter 8

Input Scheduling Criteria with

\[ -\infty < \dot{r} \leq |d_M| \]

For this final and most general solution of our conditions \(^1\), we will adapt the system of LMI to the less restrictive set of assumptions introduced so far.

Notice that we began by solving our conditions considering a constant Lyapunov function for all the so called level sets, or disturbance levels (introduced by \(\delta\) through considering \(\frac{w^2}{\delta}\)). This resulted on solving all the system of LMIs for a unique matrix \(P\), as it is described briefly on the most general references as [1], or with more detail on chapter 3 of this thesis and later chapters.

As a second approach to the problem, we modified our description of the conditions in order to accept different values of \(P\) for each level set, but always keeping that \(\dot{r} \leq 0\). This took us to consider a spline description for the control parameters of the system, according to the methods described on [9] and [6]. In chapter 4 of this thesis, we detail the description of this new and more accurate situation.

\(^1\)Mainly the Reachable Set, Non saturation of the controller and Minimal performance. Although we also have as secondary conditions the improvement of the performance as \(\delta\) grows, as well as the nesting condition of the resulting ellipsoids.
That new assumption \((\dot{P} \leq 0)\) allowed us to get better controller gains at each disturbance level, and therefore better performances. At the same time, it implied that our control system could relax the controller gain in order to avoid saturation as the disturbance level grew inside the considered limits, but it would never go back to a more aggressive controller as the state gets smaller.

After refining and improving the solution strategies for this last approach (chapter 6), and modifying slightly the scheduling criteria to get better results (chapter 7), on this chapter we wonder about the possibility of allowing the Lyapunov function to change with time, but in a more general way. To do this we will use the expressions proposed on [3] so we can consider a solution for the system where

\[-\infty < \dot{r} \leq |d_M|\]

As a result of that, the new assumptions will result on both allowing our controller to relax and strengthen the gain, although for this second case only at a limited rate, if the disturbance, or the system state, increases or decreases.
8.1 Incorporating $\dot{r} > 0$ to our set of LMI

As we saw on chapter 4, the initial system of LMI from chapter 2 holds if we incorporate $x^T \dot{P} x < 0$ terms on the smaller side of the inequalities (see section 4.1). Following that idea, we can also accept $x^T \dot{P} x > 0$ terms on this smaller side of the inequality, provided they are not too big. The result will be that the solution will be harder to find and the resultant performance will be worse, but it should exist.

Recall that on chapter 4 we divided the term $\dot{P}$ in two main parts:

$$\dot{P} = \frac{\delta P \delta r}{\delta r \delta t}$$

First of all, we will keep our distribution of the space state based in nested ellipsoids, and to do so we will keep the nesting condition represented by:

$$\frac{\delta P}{\delta r} > 0 \rightarrow P(r_2) > P(r_1) \text{ if } r_2 > r_1$$

Therefore, in order to get $x^T \dot{P} x > 0$ we need to consider $\dot{r} > 0$.

As we said, we cannot accept a randomly high value for $\dot{P}$ on our LMI. We need to set up a bound. Since $P(r)$ is a spline over the solutions of the LMI (3) ($P_1, P_2, ..., P_n$), we have that the most reasonable and easiest way to bound $\dot{P}$ would be by bounding $\dot{r}$, and leave $\frac{\delta P}{\delta r} > 0$.

Therefore, we set a maximum positive value for $\dot{r}$ determined by $d_M$ such that:

$$-\infty < \dot{r} \leq |d_M|$$

Next we show the way to implement this criteria in our LMI system.

---

2Where $r$ was a parameter related directly to $\delta$ through the expression $r = \delta \frac{u_{\text{lim}}}{w_{\text{max}}}$

3These will be the reference values over which we will spline
### 8.1.1 Reachable Set LMI

The Reachable Set condition is expressed by the next inequality:

\[
\dot{V} + \alpha (V - w_{max}^2) < 0 \tag{8.1}
\]

We will now derive the final LMI as we have done in the previous chapters for the \( P(r) = \text{constant} \) case and the \( \dot{r} < 0 \) case, but this time considering \(-\infty < \dot{r} \leq |d_M|\)

If we consider the structure for the Lyapunov function, we can re-write 8.1 as:

\[
\begin{bmatrix}
\dot{x}^T P x + x^T \begin{bmatrix} \delta P & \delta r \\ \delta r & \delta t \end{bmatrix} x + x^T P \dot{x} \\
\end{bmatrix} + \alpha \left[ x^T P x - w_{max}^2 \right] < 0
\]

As we have done before, we can substitute \( w_{max}^2 \) by \( w^2 \) (considering \( w(t)^2 < w_{max}^2 \)) with the only cost of introducing some conservatism on the solution. After that, we substitute the space state description \( \dot{x} = A x + B_2 K x + B_1 w \) and group terms in order to get:

\[
\begin{bmatrix}
A^T P + PA + K^T B_2^T P + PB_2 K + \alpha P + \delta P \delta r \\
\delta P \delta t & \delta P \delta r & \delta P \delta t \\
B_2^T P & -\alpha \\
B_2^T P & -\alpha \\
\end{bmatrix} < 0
\]

And in order to make its variables linear, we pre and post multiply it by

\[
\begin{bmatrix}
P^{-1} & 0 \\
0 & 1 \\
\end{bmatrix}
\]

And we finally get:

\[
\begin{bmatrix}
QA^T + AQ + F^T B_2^T + B_2 F + \alpha Q + Q \frac{\delta P}{\delta r} Q \frac{\delta r}{\delta t} & B_1 \\
B_1^T & -\alpha \\
\end{bmatrix} < 0 \tag{8.2}
\]

This last expression 8.2 is very similar to what we were considering on the previous chapters, but with the inclusion of a new term which is somehow "annoying": \( Q \frac{\delta P}{\delta r} Q \frac{\delta r}{\delta t} \)

Recovering what we explained on the previous section of this chapter, to simplify this term we will consider \( \frac{\delta P}{\delta r} > 0 \), whereas \( \frac{\delta r}{\delta t} = \dot{r} \) will be:

\[-\infty < \dot{r} \leq |d_M|\]
Since we are going to deal with an LMI, we can say that if the LMI satisfies the two extreme values \(^4\), it will satisfy all the other possible situations. Therefore, on our system of LMI we just have to consider this two cases.

Moreover, the case corresponding to \(\dot{r} = (-\infty)\) is exactly what we considered on Chapter 4 of this thesis; where we just eliminated this term from the LMI, since it is a negative term on the smaller side of the inequality.

The other case, \(\dot{r} = |d_M|\), requires some more attention but can be also simplified considering:

\[
\begin{cases}
\frac{\delta P}{\delta r} = -(Q^{-1})\frac{\delta Q}{\delta r}Q^{-1} & \text{since} : \quad I = PQ \\
P = Q^{-1}
\end{cases}
\]  

\[0 = \frac{\delta P}{\delta r}Q + P\frac{\delta Q}{\delta r} \quad \text{(8.3)}
\]

As a result of this two situations, we can re-write LMI 8.2 in this two ways:

- For \(\dot{r} = (-\infty)\):
  \[
  \begin{pmatrix}
  QA^T + AQ + F^T B_2^T + B_2 F + \alpha Q & B_1 \\
  B_1^T & -\alpha
  \end{pmatrix}
  < 0 \quad \text{(8.4)}
  \]

  Which, again, is the same LMI that we were considering on the previous chapters.

- For \(\dot{r} = |d_M|\):
  \[
  \begin{pmatrix}
  QA^T + AQ + F^T B_2^T + B_2 F + \alpha Q - \frac{\delta Q}{\delta r}d_M & B_1 \\
  B_1^T & -\alpha
  \end{pmatrix}
  < 0 \quad \text{(8.5)}
  \]

Therefore we have that, after this new assumption regarding \(\dot{P}\), the reachable set condition will be represented by two LMI instead of one.

\(^4\)Which are \(\dot{r} = (-\infty)\) and \(\dot{r} = |d_M|\)
Spline description

As we explained on chapter 4, we need a continuous description of the evolution of the variables. To do this we will spline them in between the results from our set of LMI throughout a series of discrete values of $\delta$ (or $r$).

For that we will follow the same process as described on chapter 7.2.1 with our two LMI corresponding to the reachable set (LMI 8.4 and LMI 8.5).

Since LMI 8.4 is equivalent to what we had before, we can accept directly the solution shown on section 7.2.1 which is:

$$N^{(-\infty)}(r) = \begin{pmatrix} AQ(r) + Q(r)A^T + \alpha Q(r) + B_2 F(r) + F^T(r)B_2^2 & B_1 \\ B_1^T & -\alpha \end{pmatrix} < 0$$

And finally, it is easy to check that

$$N^{(-\infty)}(r) < 0 \quad \text{if} \quad N_n^{(-\infty)} < 0$$

On the other hand, the case corresponding to LMI 8.5 is more complicated because of the presence of $\delta Q \over \delta r$. However, we can simplify this term after considering the properties of our linear spline.
Consider the linear spline used for our variables, and particularly, the case of \( Q \), which is given by the expression:

\[
Q(r) = \begin{cases} 
Q_S(r) & = Q_n + \frac{Q_{n+1} - Q_n}{r_{n+1} - r_n}(r - r_n) \\
Q(r) & = \frac{1}{l} \int_{r-rac{l}{2}}^{r+rac{l}{2}} Q_S(h) dh
\end{cases}
\]  

(8.6)

Notice that, on the practical implementation of the scheduled controller, we always focus on the linear part of the spline and neglect the "rounding term" represented by the integral. This is equivalent to consider a very small "\( l \)", and can be assumed without any loss of generality in all the results that we are working with.

Therefore, if we can approach \( Q(r) \) by \( Q_s(r) \) the same way we have been doing so far, we have that the derivative of \( Q(r) \) in the interval \((Q_n \ Q_{n+1})\) is exactly:

\[
\frac{\delta Q(r)}{\delta r} = \frac{Q_{n+1} - Q_n}{r_{n+1} - r_n}
\]

and we can use this expression to simplify LMI 8.5. According to this we can say:

\[
N^{dM}(r) = \begin{pmatrix}
AQ(r) + Q(r)A^T + \alpha Q(r) + B_2 F(r) + F^T(r) B_2^2 - \frac{\delta Q}{\delta r} d_M & B_1 \\
B_1^T & -\alpha
\end{pmatrix} < 0
\]

\[
N^{dM}(r) = (1 - \lambda)^2 N^{dM}_n + \lambda^2 N^{dM}_{n+1} + \lambda(1 - \lambda)(N^{dM}_n + N^{dM}_{n+1}) < 0
\]

Where we have:

\[
N^{dM}_n = \begin{pmatrix}
AQ_n + Q_n A^T + \alpha Q_n + B_2 F_n + F_n^T B_2^2 - \frac{Q_{n+1} - Q_n}{r_{n+1} - r_n} d_M & B_1 \\
B_1^T & -\alpha
\end{pmatrix} < 0
\]

And finally, it is easy to check that

\[
if \quad N^{dM}_n < 0 \quad then \quad N^{dM}(r) < 0
\]
8.1.2 Performance LMI

On this condition we also have the $\frac{\delta Q}{\delta r}$ term, and therefore, we also need a special treatment. Recall the differential equation used to describe the performance of the system:

$$\dot{V} < \gamma^2 w^T w - y^T y$$

And again, substitute the definition of the Lyapunov function to get:

$$\dot{x}^T Px + x^T \left\{ \frac{\delta P}{\delta r} \frac{\delta r}{\delta t} \right\} x + x^T P \dot{x} - \gamma^2 w^T w + y^T y < 0$$

Where we can substitute the space state description ($\dot{x} = Ax + B_2 K x + B_1 w$) and group terms in order to get:

$$\begin{pmatrix} A^T P + PA + KB_2^T P + PB_2 K + \frac{\delta P}{\delta r} \frac{\delta r}{\delta t} & PB_1 & C^T + K^T D^T \\ B_1^T P & -\gamma^2 & 0 \\ C + DK & 0 & -I \end{pmatrix} < 0$$

Where we can make the variables to show up linearly after multiply it by

$$\begin{pmatrix} P^{-1} & 0 \\ 0 & 1 \end{pmatrix}$$

Finally getting:

$$\begin{pmatrix} QA^T + AQ + F^T B_2^T + B_2 F + Q \frac{\delta P}{\delta r} \frac{\delta r}{\delta t} & B_1 & QC^T + F^T D^T \\ B_1^T & -\gamma^2 & 0 \\ CQ + DF & 0 & -I \end{pmatrix} < 0$$

And again, after considering the expression of $\frac{\delta P}{\delta r}$ in terms of $Q$ (expression 8.3) we can simplify and get:

$$\begin{pmatrix} QA^T + AQ + F^T B_2^T + B_2 F - \frac{\delta Q}{\delta r} \frac{\delta r}{\delta t} & B_1 & QC^T + F^T D^T \\ B_1^T & -\gamma^2 & 0 \\ CQ + DF & 0 & -I \end{pmatrix} < 0$$

At this point, we end up having to consider our both extreme cases for $\dot{r}$. As a result of that, we will end up once more with a condition expressed through two LMI.
• For $\dot{r} = (-\infty)$ we have the same LMI as we had so far:

$$\begin{pmatrix}
QA^T + AQ + F^T B_2^T + B_2 F & B_1 & QC^T + F^T D^T \\
B_1^T & -\gamma^2 & 0 \\
CQ + DF & 0 & -I
\end{pmatrix} < 0$$

• For $\dot{r} = d_M$ we have:

$$\begin{pmatrix}
QA^T + AQ + F^T B_2^T + B_2 F - \frac{\delta Q}{\delta r} d_M & B_1 & QC^T + F^T D^T \\
B_1^T & -\gamma^2 & 0 \\
CQ + DF & 0 & -I
\end{pmatrix} < 0$$

**Spline description**

As it happened with the reachable set condition, the case for $r = (-\infty)$ gives us the same spline description based on:

$$J^{(-\infty)}(r) = \begin{pmatrix}
AQ(r) + Q(r)A^T + B_2 F(r) + F^T(r)B_2^T & B_1 & Q(r)C_1^T + F^T(r)D^T \\
B_1^T & -\gamma^2(r) & 0 \\
C_1 Q(r) + DF(r) & 0 & -I
\end{pmatrix} < 0$$

Which is also expressed as:

$$J^{(-\infty)}(r) = (1 - \lambda)^2 J_n^{(-\infty)} + \lambda^2 J_{n+1}^{(-\infty)} + \lambda (1 - \lambda) [J_n^{(-\infty)} + J_{n+1}^{(-\infty)}] < 0$$

Where:

$$J_n^{(-\infty)} = \begin{pmatrix}
AQ_n + Q_n A^T + B_2 F_n + F_n^T B_2^T & B_1^T & Q_n C_1^T + F_n^T D^T \\
B_1 & -\gamma_n^2 & 0 \\
C_1 Q_n + DF_n & 0 & -I
\end{pmatrix} < 0$$
And finally, it is easy to check that

\[ \text{if } J_{n}^{(-\infty)} < 0 \text{ then } J^{(-\infty)}(r) < 0 \]

On the other hand, for the case \( \dot{r} = d_M \) we consider again that \( l = 0 \) on the spline description given by formula 8.6. As a result of that we can substitute

\[ \frac{\delta Q}{\delta r} = \frac{Q_{n+1} - Q_n}{r_{n+1} - r_n} \]

And we finally get:

\[
J^{d_M}(r) = \begin{pmatrix}
AQ(r) + Q(r)A^T + B_2F(r) + F^T(r)B_2 - \frac{Q_{n+1} - Q_n}{r_{n+1} - r_n}d_M & B_1^T & Q(r)C_1^T + F^T(r)D^T \\
B_1 & -\gamma^2(r) & 0 \\
C_1Q(r) + DF(r) & 0 & -I
\end{pmatrix} < 0
\]

Which is also expressed as:

\[
J^{d_M}(r) = (1 - \lambda)^2J_{n}^{d_M} + \lambda^2J_{n+1}^{d_M} + \lambda(1 - \lambda)[J_{n}^{d_M} + J_{n+1}^{d_M}] < 0
\]

Where:

\[
J_{n}^{d_M} = \begin{pmatrix}
AQ_n + Q_nA^T + B_2F_n + F^T_nB_2^T - \frac{Q_{n+1} - Q_n}{r_{n+1} - r_n}d_M & B_1^T & Q_nC_1^T + F^T_nD^T \\
B_1 & -\gamma_n^2 & 0 \\
C_1Q_n + DF_n & 0 & -I
\end{pmatrix} < 0
\]

And finally, it is easy to check that

\[ \text{if } J_{n}^{d_M} < 0 \text{ then } J^{d_M}(r) < 0 \]
8.1.3 Non-Saturation LMI and the Nesting Condition LMI

Since these conditions do not include the derivative of the Lyapunov function on their LMI, they do not change with respect to what we had on the previous chapters. As a result of that, the Non-Saturation LMI which is represented by:

\[
M(r) = \begin{pmatrix}
Q(r) & F^T(r) \\
F(r) & r
\end{pmatrix}
\]

Which will hold for any splinned value if we enforce:

\[
M_n = \begin{pmatrix}
Q_n & F^T_n \\
F_n & r_n
\end{pmatrix} > 0
\]

As well as the Nesting Condition, which is also the same since we are still considering

\[
\frac{\delta P}{\delta r} > 0 \rightarrow \frac{\delta Q}{\delta r} < 0
\]

Which will still be represented by:

\[
Q_{n+1} < Q_n
\]
8.2 The LMI system to solve

As we have been doing all throughout this thesis, to solve this new LMI system we will add this last modification ($\dot{r} < |d_M|$) on the last version of our method, described on chapter 7. This way we will get the most general and powerful version of the method.

According to that, the resulting new version will divide the state space in areas bounded by the reachable sets \(^5\) and the saturation limits for the corresponding controller gains.

As it happened on chapter 7 and all the previous chapters, the system will have zero initial conditions. As a result of that, it will start applying the gain corresponding to $K_{max}$ that will keep it inside the ellipsoid $\epsilon_{max}$. If the system happens to cross the saturation bound, the scheduling will allow us to switch to the next values $K_{max-1}$ and $\epsilon_{max-1}$, and start performing the spline for the controller gain and the Lyapunov function on the interval $[r_{max} \ r_{max-1}]$.

On Figure 8.1 we recall a scheme about the mentioned division of the state space:

---

\(^5\)The non-scaled ellipsoids, since they do not include the disturbance scaling factor $\delta$ on their definition $\epsilon_n = x : x^TP_nx \leq w_{max}^2$
The new feature of this chapter is the fact that if the system is back again on the previous area bounded by $K_{\text{max}}$, it will go back to this value, even if it is out of the corresponding reachable set $\epsilon_{\text{max}}$. This is because the stability criteria expressed by the condition:

$$\dot{V} + \alpha(V - w_{\text{max}}^2) < 0 \quad (8.7)$$

Allows us to say that the controller will drive the system into $\epsilon_{\text{max}}$ and keep it there, even if it starts out of it, since expression 8.7 imposes:

$$\text{if } V > w_{\text{max}}^2 \text{ then } \dot{V} < 0$$

To get this response of the system, we need a set of controller gains such that:

- We can have a reachable set expressed by $\epsilon_n = x : x^TP_nx \leq w_{\text{max}}^2$ that the system will never abandon with the corresponding controller gain.

- The reachable set $\epsilon_n$ -corresponding to $\delta_n$, is nested inside the previous set $\epsilon_{n-1}$ -corresponding to $\delta_{n-1}$.

- We never saturate our controller inside the ellipsoid described by $\epsilon^*_n = x : x^TP_nx \leq \frac{w_{\text{max}}^2}{\delta_n^2}$.

- A certain maximal performance $\gamma_n$ is warranted inside $\epsilon_n$ when we use the corresponding $K_n$ (provided we do not saturate there). This performance improves as we consider higher gains.

And for this we need to solve the following system for each value of $\delta$:

- Reachable set for $\dot{r} = (-\infty)$:

$$N_n(-\infty) = \begin{pmatrix} AQ_n + Q_nA^T + \alpha Q_n + B_2F_n + F_n^TB_2^2 & B_1 \\ B_1^T & -\alpha \end{pmatrix} < 0 \quad (8.8)$$
• Reachable set for $\dot{r} = d_M$:

$$N_n^{d_M} = \begin{pmatrix}
AQ_n + Q_n A^T + \alpha Q_n + B_2 F_n + F_n^T B_2^2 - \frac{Q_n + Q_{n+1}}{r_{n+1} - r_n} d_M B_1 \\
B_1^T
\end{pmatrix} < 0$$

(8.9)

• Performance for $\dot{r} = (-\infty)$:

$$J_n^{(-\infty)} = \begin{pmatrix}
AQ_n + Q_n A^T + B_2 F_n + F_n^T B_2^T & B_1^T & Q_n C_1^T + F_n^T D^T \\
B_1 & -\gamma_n^2 & 0 \\
C_1 Q_n + DF_n & 0 & -I
\end{pmatrix} < 0$$

(8.10)

• Performance for $\dot{r} = d_M$:

$$J_n^{d_M} = \begin{pmatrix}
AQ_n + Q_n A^T + B_2 F_n + F_n^T B_2^T - \frac{Q_n + Q_{n+1}}{r_{n+1} - r_n} d_M B_1 \\
B_1 & -\gamma_n^2 & 0 \\
C_1 Q_n + DF_n & 0 & -I
\end{pmatrix} < 0$$

(8.11)

• Non saturation:

$$M_n = \begin{pmatrix}
Q_n & F_n^T \\
F_n & r_n
\end{pmatrix} > 0$$

(8.12)

• Nesting condition:

$$Q_{n-1} > Q_n$$

(8.13)

• Performance improvement:

$$\gamma_n > \gamma_{n+1}$$

(8.14)

As we explained on the previous section 8.1, the solution of this set of LMI allows us to consider the solution parameters as variables whose evolution along the state space is described by a spline. Therefore, the resulting variables of solving this set of LMI will be the reference values used for the spline description.
8.2.1 Strategy to solve the system: The 4 − CM₃ method

Before considering how are we going to use and implement the solutions of the system, we have to describe the way we are going to solve it. On one hand, we have that the incorporation of the term \( \frac{Q_{n+1} - Q_n}{r_{n+1} - r_n} d_M \) has forced us to consider two more LMI. On the other hand, this term increases the overlap in between the systems corresponding to consecutive values of \( \delta \). Up to now, this overlap was mainly due to the Nesting Condition.

As a result of all these changes, this new system of LMI is not going to be easy to solve. In order to do so we will consider a new Combined Sequential Method (4 − CM₃) which is essentially a modified version of the last method explained, the so called method 3 − CM₃, which was described on section 6.1.3.

As we saw on chapter 6, the Combined Methods try to avoid the characteristic problem of all sequential schemes. This problem is that sometimes, the method gets a really good value at a certain \( \delta \), but then it is not able to get reasonably good ellipsoids nested inside it for the rest of values of the sequence. To avoid this, the combined methods try to get a global perspective of the minimization of the sequence while keeping essentially its sequential procedure. To do this, they solve the sequence of \( \delta \) in groups of two or three values, that are solved globally, and then step sequentially for the next group.

Solving for \( \delta_n \in [\delta_1, \delta_2, \ldots, \delta_{\text{max}}] \)

As we said before, when we solve the system for \( \delta_n \) we see that LMI 8.9 and LMI 8.11 include variables related to \( \delta_{n+1} \). For this reason, it would be convenient to solve the system corresponding to \( \delta_n \) together with the system corresponding to \( \delta_{n+1} \).

---

6By "overlap" we refer to having terms related to \( \delta_{n+1} \) on the system corresponding to \( \delta_n \).
7In order to reach a trade off on the achieved performance of this particular value of \( \delta \) and the values of all the remaining \( \delta \) of the sequence (which are undoubtedly influenced by it).
8They include \( Q_{n+1} \) on their expressions.
the same time, we will include a third \( \delta_{n+2} \). This third value will not come from the original sequence, and its purpose is merely auxiliary \(^9\). Its value will be determined by the parameter \( \kappa = \frac{\delta_{\text{max}}}{3} \).

Hence, we have that considering a sequence of \( \delta \) such:

\[
[\delta_1, \delta_2, \delta_3, \delta_4, \ldots, \delta_{\text{max}}]
\]

On each sequential \( n \)-th step that the \( 4-CM_3 \) method will take, we will solve globally the system of LMI corresponding to 3 values of \( \delta \) such:

\[
[\delta_n, \delta_{n+1}, \delta_{n+1} + \kappa]
\]

Where, the two first values come directly from the sequence, and the third value will not be related to the sequence. On table 8.1 we summarize this process of selecting values for each step from the main sequence:

<table>
<thead>
<tr>
<th>Method used</th>
<th>Sequential step</th>
<th>Values solved globally ( \delta = [1 2.5 5 10 20 35 70] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 4-CM_3 )</td>
<td>1st</td>
<td>1 \hspace{1cm} 2.5 \hspace{1cm} 2.5+\kappa</td>
</tr>
<tr>
<td></td>
<td>2nd</td>
<td>2.5 \hspace{1cm} 5 \hspace{1cm} 5+\kappa</td>
</tr>
<tr>
<td></td>
<td>3rd</td>
<td>5 \hspace{1cm} 10 \hspace{1cm} 10+\kappa</td>
</tr>
<tr>
<td></td>
<td>4th</td>
<td>10 \hspace{1cm} 20 \hspace{1cm} 20+\kappa</td>
</tr>
<tr>
<td></td>
<td>5th</td>
<td>20 \hspace{1cm} 35 \hspace{1cm} 70</td>
</tr>
</tbody>
</table>

As you can see, for the last step (5th) we choose all three values directly from the sequence. This is in order to avoid an extra sequential step. Besides, on the last values of the sequence there is not much difference on the values of \( \delta \) that we pick.

Once we know the values of \( \delta \) that we are going to consider, we can explain which LMI are we going to use, and how. All the LMI to consider are described on Section 8.2 (from LMI 8.8 to LMI 8.14)

\(^9\)It will be there to give us the global perspective required for the combined method, so it has to be a value far away from \( \text{delta}_n \) and \( \delta_{n+1} \).
Assigning LMI to each $\delta$

Recall that at each sequential step we solve globally 3 values of $\delta$. Consider as an example the 4th step from table 8.1, were we have:

$$[\delta_{4,1} = 10, \delta_{4,2} = 20, \delta_{4,3} = 20 + \kappa]$$

Where, the subindex indicates the sequential step (4th), and the value considered inside this step (1, 2 or 3). From each sequential step, we will only keep as a solution the variables obtained from the second $\delta$ ($\delta_{4,2}$ in this case). The variables for $\delta_{4,1}$ will be obtained from the previous step, since $\delta_{4,1} = \delta_{3,2} = 10$ has already been calculated on the previous step. Whereas, as we said, the role of $\delta_{4,3}$ will be only auxiliary and we will not keep any of its variables. Therefore, the assignation of the LMI will be:

**For $\delta_{4,1} = 10$:**

As we said, we consider the corresponding control variables ($\gamma(\delta = 10)$, $Q(\delta = 10)$ and $K(\delta = 10)$) as already known from the previous step ($\delta_{3,2} = 10$). As a result of that, we will not need to solve many of the LMI for this value. We will be only interested on the LMI that are related to variables that remain to be determined:

- **Reachable set (LMI 8.9)**
  
  From this LMI we already know all the values with the subindex $n$:
  
  $$Q_n = Q(\delta = 10), \quad \gamma_n = \gamma(\delta = 10) \quad and \quad K_n = K(\delta = 10)$$

  But we still have to determinate:

  $$Q_{n+1} = Q(\delta = 20)$$

- **Performance (LMI 8.11)**
  
  From this LMI, again, we know all values with the subindex $n$, so we can only determinate:

  $$Q_{n+1} = Q(\delta = 20)$$
For $\delta_{4,2} = 20$:

This will be the $\delta$ whose variables will be stored as a part of the solution. Here we will consider the whole set of LMI (from LMI 8.8 to LMI 8.14). Where the subindex "$n$" refers to $\delta_{4,2} = 20$, "$n - 1$" refers to $\delta_{4,1} = 10$ and "$n + 1$" refers to $\delta_{4,3} = 20 + \kappa$.

- Variables known:
  
  $Q_{n-1}$ (corresponding to $\delta_{4,1} = 10$ and calculated at the 3rd step, on $\delta_{3,3} = 10$).

- LMI considered:
  
  Reachable set (LMI 8.8 and LMI 8.9), Performance (LMI 8.10 and LMI 8.11),
  Non saturation (LMI 8.12), Nesting condition (LMI 8.13) and Performance improvement (LMI 8.14)

- Variables determined:
  
  As we solve this system simultaneously with the systems corresponding to $\delta_{4,1}$ and $\delta_{4,3}$, from this system corresponding to $\delta_{4,2}$ we will determine:
  
  $Q_n = Q(\delta = 20)$, $Q_{n+1} = Q(\delta = 20 + \kappa)$, $\gamma_n = \gamma(\delta = 20)$,
  $\gamma_{n+1} = \gamma(\delta = 20 + \kappa)$ and $K_n = K(\delta = 20)$

For $\delta_{4,3} = 20 + \kappa$:

For this case we should consider again the whole set of LMI, but this would imply the existence of a "$n + 1$" subindex related to a fourth value that we do not have. Instead of this, we consider all the control variables to be constant after $\delta_{4,3}$, and this simplifies a lot our system of LMI for this value of $\delta_{4,3}$:

- Variables known: None

- LMI considered:
  
  Reachable set (LMI 8.8), Performance (LMI 8.10), Non saturation (LMI 8.12),
  Nesting condition (LMI 8.13) and Performance improvement (LMI 8.14)
• Variables determined:

\[ Q_n = Q(\delta = 20 + \kappa) , \gamma_n = \gamma(\delta = 20 + \kappa) \quad \text{and} \quad K_n = K(\delta = 20 + \kappa) \]

**Special case: the first step**

For this case we cannot inherit the values of the control variables of the first \( \delta \) (there is no previous sequential step) so we have to calculate them also here. As a result of that, the system of LMI corresponding to \( \delta_{1,1} = 1 \) will be:

• Variables known: None.

• LMI considered:

Reachable set (LMI 8.8 and LMI 8.9), Performance (LMI 8.10 and LMI 8.11) and Non saturation (LMI 8.12).

• Variables determined:

\[ Q_n = Q(\delta = 1) , \quad Q_{n+1} = Q(\delta = 2.5) , \quad \gamma_n = \gamma(\delta = 1) \quad \text{and} \quad K_n = K(\delta = 1) \]

As we said before, for this first sequential case, instead of just keeping the obtained control variables from the second \( \delta (\delta_{1,2} = 2.5 \text{ in this case}) \) we will also keep the values for the first \( \delta_{1,1} = 1 \).

**Special case: the last step**

As we saw before, for this last step we do not consider the parameter \( \kappa \) for the third value of \( \delta \). As a result of that, the third value comes from the main sequence, and its control variables will be kept as a solution. Therefore, for this last step, a part from the variables corresponding to \( \delta_{5,2} = 35 \), we will also keep the variables from \( \delta_{5,3} = 70 \).
8.3 Results from the new criteria

Next we are going to present the results from solving the new system, following the described method $4 - CM_3$. The model considered is our usual model given by the state equations:

$$\alpha = 1, \quad u_{\text{lim}}^2 = 1000^2, \quad w_{\text{max}}^2 = 1000^2,$$

$$A = \begin{pmatrix} 0 & 1 \\ -10 & -10 \end{pmatrix}, \quad B_1 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & 6 \end{pmatrix}, \quad C = \begin{pmatrix} 5 & 0 \end{pmatrix}, \quad D = 0$$

And the following sequence of $\delta$:

$$\delta = [1 \ 1.5 \ 2.5 \ 5 \ 7.5 \ 10 \ 15 \ 20 \ 50 \ 100]$$

Finally, we have to specify also the maximum value that we want to consider for $\dot{r} = d_M$, which in this case will be:

$$d_M = 50$$

For this values, the results correspond to what is shown on the next table 8.2, where we compare them with the results from the previous method of chapter 7 (Actuator Input Scheduling Criteria $\dot{P} \leq 0$). As we can see, the fact of considering $\dot{r} < |d_M|$ eliminates some of the conservatism. However, the fact of having more LMI on the system turns into worse results in terms of performance (it is harder to get a solution for all of them). Nevertheless, this modification is worth in terms of time response and use of the controller, as we will see on later chapters.

A very important factor on our new system of LMI is the value that we choose for $d_M$. Theoretically, the bigger that $d_M$ becomes, the more difficult it gets to find a solution for the system, and therefore the results get worse. This is clear on Figure 8.2, where we see the value of the last $\gamma$ of the sequence for a different range of values of $d_M$. 

140
Table 8.2: Comparing the new method for Input Scheduling (corresponding to $-\infty < \dot{r} \leq |d_M|$) with the old methods: Input Scheduling with $\dot{r} \leq 0$ (Chapter 7) and last method for State based Scheduling with $\dot{r} \leq 0$ (Chapter 6).

| $\delta$ | Input Sched. $-\infty < \dot{r} \leq |d_M|$ | Input Sched. $\dot{r} \leq 0$ | State Sched. $\dot{r} \leq 0$ |
|-----------|---------------------------------|-------------------|----------------------------|
| 1         | 1.0943                          | 1.0973            | 1.0976                     |
| 1.5       | 0.8994                          | 0.8971            | 0.8971                     |
| 2.5       | 0.6992                          | 0.6975            | 0.6975                     |
| 5         | 0.4970                          | 0.4956            | 0.4956                     |
| 7.5       | 0.4072                          | 0.4058            | 0.4058                     |
| 10        | 0.3535                          | 0.3522            | 0.3522                     |
| 15        | 0.3034                          | 0.2937            | 0.2937                     |
| 20        | 0.2924                          | 0.2767            | 0.2767                     |
| 50        | 0.2688                          | 0.2566            | 0.2566                     |
| 100       | 0.2688                          | 0.2536            | 0.2536                     |

Figure 8.2: Evolution of the last $\gamma$ of the sequence for different $d_M$. As you can see, the first part of the figure is an expected exponential growth of the performances. But after a certain point, it blows up.
Chapter 9

Real time simulations

Throughout this thesis, we have described several ways to solve the basic conditions for our system. Each case corresponds to a slightly different set of LMI, and brings us different results. Once we have solved our system of LMI, we have the results for several boundaries of areas that divide our state space system \( (Q, K \text{ and } \gamma) \).

The spline description of these 3 parameters all over the state space, implies that assigning controller gains along the state space is not trivial. As a result of that, we have used different techniques to simulate the time evolution of the system considering the gain scheduling.

Therefore, we have to distinguish between the process of building and solving a system of LMI \(^2\) and the process of simulating the time evolution of the system \(^3\).

Here we summarize the basic approaches to solve the system of LMI that we have described along the thesis. After that, we will describe the corresponding simulation processes.

\(^1\) Traditionally ellipsoids, but on chapters 7 and 8 we had ellipsoids combined with saturation limits. 
\(^2\) According to our basic conditions plus several simplifying assumptions.
\(^3\) Under the parameter resulting from solving the LMI system, and the associated assumptions.
LMI systems described on this thesis and their solution methods:

1. **State Space Scheduling, for \( P = constant \).**

   Described in chapter 3, it is the most basic approach to our basic conditions. We did not include the spline description for the parameters, which were constant on each of their corresponding areas of the state space. The resulting LMI system could be solved either globally \( P_{constant}^{global} \) or sequentially \( P_{constant}^{sequential} \).

2. **State Space Scheduling, for \( \dot{r} \leq 0 \).**

   Described in chapter 4, it is the first approach to the basic conditions that includes the spline description of both the parameters and the LMI, based on the spline parameter \( r \).

   The resulting LMI system could be solved either globally \( P_{variable}^{global} \) or sequentially \( P_{variable}^{sequential} \). For the sequential scheme we tried different ways to solve the LMI system avoiding numerical issues. In chapter 6 we settled on the definitive method \( 3−CM \).

3. **Input Scheduling, for \( \dot{r} \leq 0 \).**

   Described on chapter 7, it is the first approach that introduces the idea of dividing the state space on reachable sets (instead of scaled ellipsoids \(^4\)) plus actuator saturation limits.

   This approach was solved sequentially using an adapted version for \( 3−CM \).

4. **Input Scheduling, for \( -\infty < \dot{r} \leq |d_M| \).**

   Described on chapter 8, it is the most general and powerful approach. The resulting LMI system was solved using an adapted version of \( 3−CM \) named \( 4−CM_3 \), which allowed the system to go back and forth from smaller to big ellipsoids and opposite.

\(^4\)The previous methods considered ellipsoids related to scaled disturbances, according to \( w \leq \frac{w^2_{\max}}{\delta^2} \).
Simulation processes developed:

Regarding the simulations performed, notice that all of them have the same scheme on Simulink\(^5\). The only difference is the box corresponding to the *Gain Scheduling process*, where we have to implement the results from solving the LMI system, the corresponding assumptions regarding $\dot{P}$, and the distribution of the state space.

As you will see, we have developed simulations for all 4 cases of LMI system. However, we have focused our interest on the last 3 cases, where the results will be more complex, accurate, and therefore they will offer more opportunity for comment. For each of the cases considered, we will implement different kinds of disturbances. However, in some cases we have found unexpected problems that have forced us to introduced subtle modifications on the Gain Scheduling process. We will describe with detail all these situations on section 9.2 of this chapter.

\(^5\)See Figure 3.3 for details.
### 9.1 Description of the elements of the system

After recalling the four main approaches to our basic conditions and the corresponding LMI systems, on this section we will describe the generic Simulink scheme used in all the time simulations (Figure 9.1).

After we do so, on section 9.2 we will proceed individually with the main box of the simulation, the *Gain Selector box*, which is in charge or implementing the Gain Scheduling process, with the spline description.

Therefore, all the boxes described on this section are common to all cases simulated, and are basically dedicated to the description to the system plus de controller, and data acquisition.

Next we show the representation of the simulink file considered:

![Generic Simulink system developed for time simulations. The only difference from one case to another is inside the Gain Scheduler box](image)

*Figure 9.1: Generic Simulink system developed for time simulations. The only difference from one case to another is inside the Gain Scheduler box*
The System box

This box contains the system description according to matrices $A$, $B_1$, $B_2$, $C$ and $D$. It gets all the inputs and the state of the system and calculates the dynamics of the system. Its internal structure is given on the next Figure 9.2:

![Figure 9.2: Internal structure of the box "system" the values of the multiplying constants are entered on the mask, according to the corresponding values of the matrices $A$, $B$, $C$ and $D."

Notice on Figure 9.2 how the input signal corresponding to the actuator comes to the "system" bounded by saturation limits. This saturation is implicit on the design of the controller, although we make sure of it on the "controller" box, by placing a saturation box.

Notice too that this same box can be represented in a more compact way with a "State Space" Simulink box, after doing some small modifications regarding the number of states in the initial conditions of the Simulink simulation. With equivalent results.
The controller box

On this box we get the state of the system after integrating the output of the "system" box, as well as the final value of the controller gains from the "Gain Selector". Once we have this, we calculate the controller action \( u = Kx \) and we make sure that we do not go beyond the saturation limits.

This saturation box should be unnecessary, according to the specifications that we implement on all LMI systems considered. However, as we will see for the cases of the Input Scheduling, the numerical techniques for solving the LMI system might make us to have, eventually, values beyond the saturation limit.

This fact has been considered and new modifications have been introduced on the simulation code to prevent it (see section 9.2.3 for details). But nevertheless, it might be convenient to make sure that, in any case, we will never bring to the system an input signal bigger than the saturation level.

In case that the saturation box acts, we would be introducing an unexpected non-linearity on the system. This will result on a behavior of the state space response out of any assumptions and considerations made during the previous steps.

The internal structure of this box is shown next:

![Figure 9.3: Internal structure of the box "controller". We read the state of the system (x) and the corresponding controller gains (K) and calculate the input signal](image)
The rest of the Simulink boxes

Leaving aside the "gain scheduler", which will be discussed separately, the rest of the Simulink boxes are:

• **Integrators:**

  They allow us to get the updated state of the system by integrating the signal that comes from the "system" ($\dot{x}$)

• **Disturbance:**

  That is the signal that is bounded by $w_{max}$ and perturbs the initial conditions of our system. The main objective of the controller that we have designed is to minimize its effects over the state of the system.

  Mainly, the disturbance signals that we will consider for our time simulations are: equally random distributed scalar, normally distributed scalar and sinusoids.

• **Data acquisition boxes:**

  On this category we have "scope" boxes that plot directly the signal associated, a "graph" box connected at the output of the system that gives us the position on the state space plane, and several boxes to export the data of the associated signals into an "*.m" file for later manipulations.
9.2 Simulation processes considered

As we said before, we have a different simulation process for each of the four methods to solve LMI systems considered. Since all the differences among them refer to the assumptions made regarding the spline description and the gain scheduling, the only difference is going to be on the box that refers to the Gain Scheduling.

To implement all these processes on a simple box we will use S-functions. Only for the first case of State Space Scheduling and \( P = \text{constant} \) we will be able to simulate by only using standard Simulink boxes.

The S-functions allow us to customize Simulink boxes by implementing MATLAB codes inside them. That makes them the most versatile technique, and allows us to implement the whole scheduling process, plus the splines, in a single box. On the next subsections we will detail the different actions and functions called on each case.

9.2.1 State Space Scheduling, for \( P = \text{constant} \).

As we said before, this is the only case where we can simulate the system response only with regular Simulink boxes. To do this we divide the “gain scheduler” box in two sub-boxes. The first part deals with the evaluation of the Lyapunov function, whereas the second part deals with the calculation of the corresponding controller gains.

This procedure is built under the assumption that the controller gains are constant on each area of the state space. As a result of that, we basically have to measure the Lyapunov function value (we have a constant \( P \) for all state space areas) and we just check into what level set are we:

\[
\frac{w_{\max}^2}{\delta^2_n} < x^TPx \leq \frac{w_{\max}^2}{\delta^2_{n+1}}
\]

At the same time, since we have no varying parameters but \( K \), we can assume that we can go from smaller ellipsoids to bigger ellipsoids, and the other way back.
However, the main problem with this simulation model is the fact that it does not consider a limit on the switch speed. Therefore, we can change $K_n$ to $K_{n+1}$ instantaneously when we change the area of the state space, which is not a particularly realistic assumption.

The evaluation of the Lyapunov function value is done on the next simulink program:

![Diagram of the Lyapunov evaluator](image)

Figure 9.4: *Internal structure of the box ”Lyapunov evaluator”*

And the process of gain scheduling, which in this case is mainly choosing the appropriate level set and issuing the associated controller gain, is done on the simulink program shown on Figure 9.5.

On Figure 9.5 we see the scheme that assigns controller gains for a state space divided into three ellipsoids. The circuit is divided in three lines. The upper line takes the value of the Lyapunov function ($V = x^T P x$) and compares it with the boundary of the smallest ellipsoid ($V - \frac{w_{\text{max}}}{\delta^3}$). In the difference is positive, that means that we are inside this smallest ellipsoid. After this subtraction we process the result in order to get a boolean parameter to represent this classification:

<table>
<thead>
<tr>
<th>Boolean value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>State INSIDE the considered level set</td>
</tr>
<tr>
<td>0</td>
<td>State OUTSIDE the considered level set</td>
</tr>
</tbody>
</table>

150
Figure 9.5: Simulink system developed for State space based scheduling, \( P = \text{constant} \)
On the other two lines, we do exactly the same with the other two bigger level sets. Since the sets are concentric, if the system is inside the smallest set, it will be also inside the bigger sets. Therefore we have to strove the signals from the larger sets once we have determined the system to be inside one of the smaller. To do this, we use the negation boxes ("A/NOT A") which will turn a "1" corresponding to a small ellipsoid into a "0" that will multiply the signals from the larger ellipsoids.

On the other hand, if the system is neither in the smallest ellipsoid, nor the middle, it assumes automatically that it is in the biggest.

Finally, once we have made sure that there will be only a "1" corresponding to the smallest set that contains the system, we multiply the resulting 0 or 1 of each ellipsoid with the corresponding gains. If we add the results of the three multiplications, we will get only the gain corresponding to the smallest ellipsoid that contains the state of the system inside.

9.2.2 State Space Scheduling, for $\dot{r} \leq 0$.

The corresponding codes for this time simulation are presented on appendix A.9. Here, we can see that the code that we implement on the s-function box has 3 main parts:

"1-ms-function-controller-actuator.m"

Presented on appendix A.9, this is the main function that organizes the calls to all the rest of subfunctions. What it does can be summarized on this basic steps:

1. Call the function in charge of initializing the simulation ("1-initializationsimulation.m"), that will read the basic values obtained from solving the LMI system (different Q and K to do the spline and several other parameters).
2. Manage the flags that Simulink uses to coordinate the simulation.

- **Flag0**: Initialize the state variables defined for the box.

  Here Simulink reads the configuration of the box (number of inputs and outputs, number of states and their initial conditions and time sampling).

  Notice that the state variables for the box does not have to be the same as the state variables of the system.

  For this box, we define as state variables:

  \[ x_{box} = [r \ K1 \ K2] \]

- **Flag2**: Update the states of the box.

  Here Simulink calls the subfunction ("1-update-r.m"), which is in charge of the main calculations (the scheduling itself and the spline description).

- **Flag3**: Calculate outputs.

  Here Simulink calculates the value of the outputs based on the values of the state variables defined for the box. In our case, the output is directly the second and third state variables of the box (K1 and K2).

"1-initialization-simulation.m"

Presented on appendix A.9, this is the function that declares the main parameters that we are going to use. It will be called by function ("1-ms-function-controller-actuator.m") at the beginning of the simulation.

The variables declared and initialized here are:

\[ \delta = [\delta_1 \ldots \delta_n] \quad Q = [Q_1 \ldots Q_n] \quad K = [K_1 \ldots K_n] \quad r = [r_1 \ldots r_n] \]

And also an array with the values of the Lyapunov function on the boundaries of the ellipsoids will be filled (ellip − border).
Presented on appendix A.9, this is the real core of the simulation. On this subfunction we perform the Scheduling process itself, plus de spline of the variables.

The order and tasks performed are represented briefly on the next lines:

1. First we focus on finding in between which two ellipsoids is the state of the system. To do so we take $Q_1$ and compare the corresponding value of the Lyapunov function ($V_1 = x^T(Q_1)^{-1}$) with the value of the corresponding boundary ($ellip\_boder(1)$). If $V_1$ is bigger we move to $Q_2$ and repeat the calculations until we find:

$$x^T(Q_{n+1})^{-1}x < V_n < x^T(Q_n)^{-1}x$$

2. Once we have the minimal ellipsoid that contains our system ($\epsilon_n$), and the maximal ellipsoid that does not contain the system ($\epsilon_{n+1}$), we initialize the corresponding values of $K_n$, $K_{n+1}$, $r_n$ and $r_{n+1}$ to prepare the spline.

3. We spline in between $r_n$ and $r_{n+1}$ iteratively until we find and ellipsoid such that the state of the system is very close to its boundary (but inside):

$$\min \left[ \frac{u^2_{lim}}{r_{new}} - x^T(Q_{new})^{-1}x \right] > 0$$

The iteration process is performed by dividing progressively the interval in halves. The first value for $r$ ($r^*_1$) will be:

$$r^*_1 = \frac{r_n + r_{n+1}}{2}$$

If $r^*_1$ is too low, the new interval to consider will be $[r^*_1, r_{n+1}]$ (and $r^*_2$ will be $r^*_2 = r^*_1 + r_{n+1}$). On the other hand, if $r^*_1$ is too high, the new interval will be $[r_n, r^*_1]$ (and $r^*_2$ will be $r^*_2 = \frac{r_n + r^*_1}{2}$).

4. Since we must have $\dot{r} \leq 0$, if we see that the definitive $r_{new}$ is smaller than $r_{old}$, we impose $r_{new} := r_{old}$ and we also keep $K(r_{new}) = K_{old}$.
9.2.3 Input Scheduling, for $\dot{r} \leq 0$.

This case is essentially the same as "State Space Scheduling, for $\dot{r} \leq 0$" described on section 9.2.2. The only difference is on the Scheduling part carried out on function "2-update-r.m".

"2-update-r.m"

Presented on appendix A.9, in this case it is clearly different from 9.2.2. The main reason is that the space state is divided considering the reachable sets for each $\delta$ and the saturation bounds of the respective controllers, instead of our previous set of scaled ellipsoids.

The tasks performed are represented briefly on the next lines:

1. Again, we begin by focusing on the interval of state space divisions. Once we know it, we will proceed with the spline description later.

Since in this case the divisions are different, we have to use a different criteria. The divisions are formed by the reachable set and the saturation bounds, it is clear that the only way to go from one division to the next is by crossing the saturation limit (see section 7.3 for details).

Therefore, what we do is try to calculate the controller action ($u = Kx$) for each possible controller, beginning with the biggest $K_{max}$ and compare it with the saturation limit. The final interval will be:

$$xK_n \leq u_{max} < xK_{n+1}$$

2. After that, we proceed with the spline description inside the given interval. This time the criteria will be finding a $K(r_{new}) \epsilon [K_n, K_{n+1}]$ such that:

$$\min [u_{max} - K(r_{new})x] > 0$$
However, the first tests of the code show that, for certain cases of very unfavorable disturbances, the fact of choosing a controller such that $K(r_{\text{new}})x = u_{\text{max}}$ is too risky. The numerical methods to solve the simulation may lead us to values of the controller around 7% beyond the saturation limit, specially if we use a too long sampling time for our simulation. To avoid this problems, we introduce a new parameter: "security $\leq 1$", to multiply our saturation limit in order to switch before we get too close to the limit.

After some tests, we have determined that the higher value for "security" that allows us to avoid this problems is

$$\text{security} = 0.85$$

3. Since we must have $\dot{r} \leq 0$, if we see that the definitive $r_{\text{new}}$ is smaller than $r_{\text{old}}$, we impose $r_{\text{new}} := r_{\text{old}}$ and we also keep $K(r_{\text{new}}) = K_{\text{old}}$.

### 9.2.4 Input Scheduling, for $-\infty < \dot{r} \leq |d_M|$.

This case is clearly different than 9.2.2 since, a part from having a different division of the space state (the same as 9.2.3), it has a substantial difference regarding the time evolution of $\dot{P}$.

"3-ms-function-controller-actuator.m"

This subfunction is essentially the same as the one described on 9.2.2. However, two clue variables have been introduced:

1. The time corresponding of the last update of the state "t-old".

   Using this parameter will allow us to approach the derivative $\dot{r}$, which is necessary to apply our new criteria for $\dot{P}$ (see section "3-update-r.m" for details about this approach).
In order to have it ready at anytime during the simulation, we include it with the state variables of the box:

\[ x_{\text{box}} = [r \quad t_{\text{old}} \quad K1 \quad K2] \]

2. The maximum value allowed for \( \dot{r} = \dot{d}_M \).

It will be declared as a new initial parameter on "3-initialization-simulation.m".

This two modifications cause minor changes in "3-initialization-simulation.m" and major changes in "3-update-r.m"

The changes in "3-ms-function-controller-actuator.m" can be seen on section A.11.

"3-initialization-simulation.m"

Please check section A.11 to see the minor changes on this code.

"3-update-r.m"

Presented on appendix A.11, this case is also clearly different from 9.2.2. The main reason is, again, the division of the space state on reachable sets and saturation bounds of the respective controllers, instead of our previous set of scaled ellipsoids.

On the other hand, we also have the new assumption of allowing positive \( \dot{r} \), which will force us to incorporate new lines to the code.

The basic steps followed by the code are:

1. First of all, we delimit the space state region where we are. To do so we use the same criteria as is explained on section 9.2.3:

\[ K_n x \leq u_{\text{max}} < K_{n+1} x \]
2. We calculate the spline inside the given interval. Again, the criteria is the same as in section 9.2.3:

\[ \min [u_{\text{max}} - K(r_{\text{new}})x] > 0 \]

When doing so we also use the security parameter "security" in order to avoid numerical problems with the saturation bounds. The same way we explained on section 9.2.3

3. The big difference comes next. So far, at this point we used to reject the values for \( r_{\text{new}} \) that were bigger than the last \( r_{\text{old}} \). Instead of that, on this case we evaluate \( \dot{r} \) by approximation:

\[ \dot{r} = \frac{r_{\text{new}} - r_{\text{old}}}{t - t_{\text{old}}} \]

If \( \dot{r} > d_M \), then we impose that \( r_{\text{new}} \) is equal to the maximum value achievable, according to \( \dot{r}_{\text{max}} = d_M \):

\[ r_{\text{new}}^{\text{max}} = d_M(t - t_{\text{old}}) + r_{\text{old}} \]
9.3 Disturbance signals and results

Once we have described both the way to solve our LMI system and the way to implement the results on a simulation, we will show the results of these simulations.

As we said, we will perform three different kinds of simulations, related to three of the four cases that we have been considering. For each case, we will consider several kinds of disturbances in order to observe different characteristics of the feedback system.

Notice that, unless indicated otherwise, the considered system will be our usual trial system:

\[
\begin{align*}
\alpha &= 1, \quad u_{\text{lim}}^2 = 1000^2, \quad w_{\text{max}}^2 = 1000^2, \\
A &= \begin{pmatrix} 0 & 1 \\ -10 & -10 \end{pmatrix}, \quad B_1 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & 6 \end{pmatrix}, \quad C = \begin{pmatrix} 5 & 0 \end{pmatrix}, \quad D = 0
\end{align*}
\]

And the sequence of \( \delta \) will be:

\[
\delta = [1 \ 1.5 \ 2.5 \ 5 \ 7.5 \ 10 \ 15 \ 20 \ 50 \ 100]
\]

9.3.1 Disturbance signals

The main purpose of our designed control system, as we have seen, is to keep a zero state against a bounded disturbance signal. The main example that we have been considering is an earthquake, although it might be a different kind of signal according to what kind of example of application we are considering (earthquake, automatic pilot for a plane, etc...).

In order to get the most general possible idea about what the time response of the feedback system would be, we have considered two basic kinds of disturbance signals for our simulations. Next we present the signals that we will use in all the cases unless we indicate otherwise:
• Equally Distributed Random Number on \( w(t) \in [-w_{\text{max}}, w_{\text{max}}] \). (Fig. 9.6)

![Equally Distributed Random Number](image.png)

Figure 9.6: Disturbance signal1. The interval is slightly smaller than \([-w_{\text{max}}, w_{\text{max}}]\) in order to avoid problems with the saturation limit of the controller, due to the numerical resolution.

• Sinusoidal Signal with Linearly Increasing Frequency. (Fig. 9.7)

![Sinusoidal Signal with Linearly Increasing Frequency](image.png)

Figure 9.7: Disturbance signal2. In this case the variable frequency of the signal allows us to have an idea of the frequency response of the system.
9.3.2 Results for $\dot{r} \leq 0$, State Space Scheduling

This is the case corresponding to our second approach to our basic conditions using the spline description. The LMI system was introduced on Chapter 4, and the methods to solve it were described on Chapters 4, 3 and, finally, 6.

Recall the main features of this approach to our problem, which are the use of the Spline description for the variables, the fact that we are considering non constant $P$ matrices for the Lyapunov function, and finally the fact that we are considering $\dot{r} \leq 0$. According to this, we have the following results:

Equally Distributed Random Disturbance

Evolution of the state of the system:

Figure 9.8: State Space evolution, for the $\dot{r} \leq 0$ case and an Equally Distributed Disturbance. The disturbance only pushes the state of the system from the 10th ellipsoid (the smallest) to the 8th
Evolution of the chosen controller gain:

![Controller Gain](image)

Figure 9.9: Selected controller gain, for the $\dot{r} \leq 0$ case and an Equally Distributed Disturbance. According to what we saw on Fig. 9.8, we only get Gains between the reference values corresponding to the 10th ellipsoid (the smallest) to the 8th

Evolution of the actuator input signal:

![Actuator Input](image)

Figure 9.10: Actuator input, for the $\dot{r} \leq 0$ case and an Equally Distributed Disturbance. The actuator input stays within its saturation limits all the time.
Notice that if instead of having our controller, we would have had an uncontrolled system, the response on the state space would have been as shown on Figure 9.11:

Figure 9.11: State Space evolution, for the \( \dot{r} \leq 0 \) case and an Equally Distributed Disturbance, Openloop system. For this case the effects of the disturbances are much larger.

As you can see, the presence of the controller results on a much smaller disturbance of the system, given the same disturbance signal represented on Figure 9.6.

Notice also how, on Figure 9.9, as the system state goes back to the origin, the limitation \( \dot{r} \leq 0 \) forces our system to keep the controller gain constant until the system state grows again beyond the previous limit.

As a result of the gain scheduling application, the actuator input stays at a reasonable percentage of the actuator capabilities on all the range of states that the system has during the simulation.
Sinusoidal Signal with Linearly Increasing Frequency

Evolution of the state of the system:

![State Space Evolution of the System](image)

Figure 9.12: State Space evolution, for the $\dot{r} \leq 0$ case and a Sinusoidal Signal with Linearly Increasing Frequency. The given disturbance pushes the state from the 10th ellipsoid (the smallest) to the 6th

Evolution of the chosen controller gain:

See Figure 9.13

Evolution of the actuator input signal:

See Figure 9.14

Notice that if instead of having our controller, we would have had an uncontrolled system, the response on the state space for the same disturbance would have been as shown on Fig. 9.15. Here we can see also how the presence of the controller results on a much smaller disturbance of the system, given the same disturbance signal represented on Figure 9.7.
Figure 9.13: *Selected controller gain, for the $\dot{r} \leq 0$ case and an Equally Distributed Disturbance. We only get Gains between the reference values corresponding to the 10th ellipsoid (the smallest) to the 6th.*

Figure 9.14: *Actuator input, for the $\dot{r} \leq 0$ and an Equally Distributed Disturbance. The actuator input stays within its saturation limits all the time.*
Figure 9.15: State Space evolution, for the $\dot{r} \leq 0$ case and a Sinusoidal Disturbance Signal, Openloop system. For this case the effects of the disturbances are much larger.
9.3.3 Results for $\dot{r} \leq 0$, Input Scheduling

This is the case corresponding to our third approach to our basic conditions using the spline description. The LMI system was introduced on Chapter 7, as well as the methods to solve it. Recall the main features of this approach to our problem which are: The use of the Spline description for the variables, the fact that we are considering non constant $P$ matrices for the Lyapunov function, the fact that we are considering $\dot{r} \leq 0$ and finally, the division of the state space in areas marked by the Reachable Sets plus the actuator bounds. According to this, we have the following results:

Equally Distributed Random Disturbance

For this case, we will consider $security = 0.87$ (see section 9.2.3 for details)

Evolution of the state of the system:

On this case, the big number of lines required to represent the space state (see Figure 7.4 for more details) makes us think that a representation of the state space evolution with the borders of the different areas would be hard to see. Therefore we will skip this representation. However, the controller gain plots give us an idea of the situation of our system with respect of the state space areas considered (Fig. 9.16), as well as final Figures 9.24 and 9.25.

Evolution of the chosen controller gain:

If we compare current Fig 9.16 with its equivalent from section 9.3.2 (Fig 9.9) we see that on this case, the controller Gain $K$ increases slowly. For instance, if we take as a reference the time when the system reaches $K_{2,9}$ on both cases, we can see that for Figure 9.16 it is close to $t=1s$, whereas for Figure 9.9 it was before $t=0.25s$.

So this new method for scheduling indeed makes our controller to work more time with higher controller gains. As a result of that we use better our actuator (see Fig. 9.17).
Figure 9.16: Selected controller gain, for the $\dot{r} \leq 0$, Input Sched. case, and an Equally Distributed Disturb. The Figure is very similar to the previous 9.9 but the evolution is slower, staying more time with the higher controller gains.

Evolution of the actuator input signal:

Figure 9.17: Actuator input, for the $\dot{r} \leq 0$, Input Scheduling case, and Equally Distrib. Disturb. Notice that here we use better the actuator, compared with Fig. 9.10.
Sinusoidal Signal with Linearly Increasing Frequency

For this case, we will consider security = 0.85

Evolution of the chosen controller gain:

![ControllerGainGraph](image)

Figure 9.18: Selected controller gain, for the \( \dot{r} \leq 0 \) case and Sinusoidal Disturbance Signal. Notice that this time we only get Gains between the reference values corresponding to the 10th ellipsoid (the smallest) to the 7th

As we can see on Figure 9.18, here we have again the same situation. The new structure of the space state introduced on this model allows us to use the higher gains for more time.

This turns into a better use of our controller, as we will see on next Figure 9.19.

Evolution of the actuator input signal:

On this case, again, we have an idea of the frequency response of the system. At the same time, we can see how the use of higher controller gains for longer periods
results on a better use of our controller (we stay closer to the saturation bounds than in previous case on Fig 9.14).

![Figure 9.19: Actuator input, for the $i \leq 0$ case and Sinusoidal Disturbance Signal. The actuator input stays within its saturation limits all the time.](image)

Figure 9.19: *Actuator input, for the $i \leq 0$ case and Sinusoidal Disturbance Signal. The actuator input stays within its saturation limits all the time.*
9.3.4 Results for $-\infty < \dot{r} \leq |d_M|$ simulations

This is the case corresponding to our fourth and definitive approach to our basic conditions using the spline description. The LMI system was introduced on Chapter 8, as well as the methods to solve it.

Recall the main features of this approach to our problem which are:

- The use of the Spline description for the variables.
- The fact that we are considering non constant P matrices for the Lyapunov function.
- The fact that we are considering $\dot{r} \leq 0$.
- The division of the state space in areas marked by the Reachable Sets plus the actuator bounds.
- The fact that we allow positive time derivatives for $\dot{r}$.

Notice also that for this case, we will consider security = 0.85 (see section 9.2.3 for details about this parameter) and a maximum positive derivative of $\dot{r} = d_M = 1000$.

According to all this, we have the following results:

**Equally Distributed Random Disturbance**

Evolution of the chosen controller gain:

Following the same approach used in section 9.3.3, in this case we will stay on the higher values of the gain even longer, and furthermore, we will be back to them when this is possible. If we compare current Fig 9.20 with its equivalent Figures from previous sections (Fig 9.9 and 9.16), we can see that for this current case gains are much higher for the same disturbance in all three cases.
Figure 9.20: Selected controller gain, for the $-\infty < \dot{r} \leq |d_M|$ case, Input Scheduling case, and Equally Distrib. Disturb. On this case the evolution is the slowest, staying more time with the higher $K_2$ and coming back to them when allowed.

Evolution of the actuator input signal:

Figure 9.21: Actuator input, for the $-\infty < \dot{r} \leq |d_M|$ case, Input Scheduling, and Equally Distributed Disturbance. On this case, we stay closer to the saturation bounds, using much better the actuator capabilities.
Sinusoidal Signal with Linearly Increasing Frequency

Evolution of the chosen controller gain:

![Controller Gain K_2(t) compared with the reference values for K_2 at δ_10, δ_9, and δ_8](image)

Figure 9.22: *Selected controller gain, for the \(-\infty < \dot{r} \leq |d_M|\) case and Sinusoidal Disturbance Signal. On this case, we also have higher gains for more time*

As we can see on Figure 9.22, the new situation reached with this new solution allows us to get higher gains, and this turns into a better use of our controller.

Evolution of the actuator input signal:

See Figure 9.23 on next page.

On this case, we can see how the use of higher controller gains for longer periods results on a better use of our controller (we stay really close to the saturation bounds during all the simulation time). At the same time, this fact should turn into a faster time evolution of the system. We will discuss that fact on section 9.3.5.
Figure 9.23: Input, for the $-\infty < \dot{r} \leq |d_M|$ case and Sinusoidal Disturbance Signal. The actuator input stays within its saturation limits all the time.
9.3.5 Final comparisons

As a final comparison between this three main methods simulated, we present Figures 9.24 and 9.25. This figures are also based in an Equally Distributed Random Disturbance, although it is different from the one shown on Figure 9.6:

![Comparison of the evolution of $K_2$ for the three simulated methods](image)

Figure 9.24: Comparison of the Gain $K_2$ obtained for the 3 main approaches simulated. Notice how the fact of considering Input Scheduling implies bigger gains than the State Scheduling, and the allowance of a positive $\dot{r}$ increases even more this effect.

Figure 9.24 shows again the controller gains for the three methods, but makes easier the comparison. We can see how the initial gain is the same for both State Scheduling $\dot{r} \leq 0$ and Input Scheduling $\dot{r} \leq 0$, although the evolution is quite different.

On the other hand, Input Scheduling $\infty < \dot{r} \leq d_M$, presents a higher initial controller gain and the big advantage of being able to recover higher gains when the disturbance allows it.

As we have seen on previous sections, this different evolution of the gain allows us to make a better use of the actuator capacities. The higher the controller gains stay,
the better we use the actuator. A little bit surprising, though, is the fact that the evolution of the state of the system is almost identical in all the three cases. Initially, we would expect that higher gains will keep the system closer to the origin and/or drive it faster. However, the representation of both state variables ($x_1$ and $x_2$) shows very similar plots. A possible continuation of this work could be checking if this fact is something peculiar of the considered system or it is general.

Finally, Figure 9.25 shows the evolution of the scheduling parameter $r$ on the three cases. As expected, this plot is very similar to Figure 9.24, since the controller gain is parameterized by $r$.

![Figure 9.25: Comparison of the Scheduling parameter $r$ obtained for the 3 main approaches simulated.](image)
Chapter 10

Conclusion

Based on recent progress in linear parameter varying control design, control techniques for dealing with actuator saturation have been discussed. The saturation non-linearity has been avoided by imposing the actuator to work below saturation limit as a control objective. This simplified the computation and implementation of the controller. The other main control objectives considered included obtaining desirable performance guarantees that were functions of the actuator capacity.

The study has followed the main ideas and techniques exposed on the bibliography, which describe the selection and description of the control objectives, and the use of linear varying control parameters. Once these concepts were presented and analyzed the attention was focused on the strategies to solve the resulting system of LMI. During this analysis several numerical issues were detected, specially the "unexpected result" described on Chapter 5. The analysis of this issues brought the development of an original technique (the so called "Combined Method" described on Chapter 6) which tries to combine the best features from the two basic techniques of solution to get a desirable performance at a reasonable computational complexity.

This new technique of solution, as well as the time-simulation based on its results
(Chapters 6 and 9), together with the resolution and simulation of the system under the $-\infty < \dot{r} \leq d_M$ and Actuator Input Scheduling assumptions (Chapters 8 and 9) are the most innovative issues of this Thesis.

As a summary, it can be said that a complete perspective of the formulation of the problem of the Gain Scheduling using linear varying control parameters has been given. Moreover, a new and competitive technique to solve the system -in terms of the low performances disturbance/output obtained, reduced complexity, and full use of our actuator capabilities- has been proposed (see Chapter 8).

Besides, the use of this new technique of resolution has allowed the solution and the simulation of the response for the most powerful approaches described on the bibliography.

Finally, the sequential presentation of the different methods studied, from the simplest and most conservative to the most complex and powerful, has contributed to have a good idea about the implications of the successive simplifying assumptions and their influence on the results.

As a possible continuations of this work, a more complete analysis of the role of the weights on the combined method techniques can be performed. The fact that the different techniques, although having improved greatly the usage of the actuator, have not been able to make the time response of the closed-loop system is also an important topic to explore -although it might be due exclusively to a particularity of the considered system-.

Finally, a more detailed study of the role and sensibility towards the parameter $d_M$ on the combined method $4 - CM$ would be interesting.
Bibliography


Appendix A

Matlab codes for the main methods developed

A.1 Matlab code for $P_{constant}^{sequential}$ State Space Scheduling

A.1.1 Main file

```matlab
clear clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% First the coefficients are defined %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

C=[5 0];
w_2=1000^2;
u_lim_2=1000^2;
A=[0 1; -10 -10];
A=[-1 5; -10 -15];
B1=[0.5 1]';
B2=[0 6]';
D=0;
alpha=1;
```
\begin{verbatim}
delta=[1 2.5 5 10 20 30 40 50 60 70 80];

gamma_list=[ ]; k_list=[ ];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NOW SOLVE THE MOST EXTERNAL RING %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SYSTEM FOR THE MINIMIZATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

setlmis([ ]); % kQ=F Q=inv(P)
Q=lmivar(1,[2 1]); % (symmetric, [block size=2 1=full symmetric blk])
F=lmivar(2,[1 2]); % (rectangular, [1 row, 2 columns])
gamma_2=lmivar(1,[1 1]);

lmiterm([3 1 1 Q],A,1,'s'); % Performance LMI
lmiterm([3 1 1 F],B2,1,'s');
lmiterm([3 2 0],B1');
lmiterm([3 2 2 gamma_2],-1,1);
lmiterm([3 3 1 Q],C,1);
lmiterm([3 3 1 F],D,1);
lmiterm([3 3 2 0],0);
lmiterm([3 3 3 0],-1);

lmiterm([1 1 1 Q],A,1,'s'); % Reachable set LMI
lmiterm([1 1 1 F],B2,1,'s');
lmiterm([1 1 1 Q],alpha,1);
lmiterm([1 2 0],B1');
lmiterm([1 2 2 0],-alpha);

lmiterm([-2 1 1 Q],1,1); % Saturation LMI
lmiterm([-2 1 1 F],1,1);
lmiterm([-2 2 2 0],(delta(1)^2)*u_lim_2/w_2);

case2=getlmis;

n=decnbr(case2);
c=zeros(1,n-1);
c=[c 1]; % we cancel all the internal variables but the last,
   % which corresponds to gamma_2
   % since it was the last that we defined.
options=[1e-7 1e3 1e9 10 0]; % [accuracy on optimal value /

\end{verbatim}
% max n iterations / feasibility radius / tolerance on iterations
% for optimization / turns off the trace of execution

[copt,xopt]=mincx(case2,c,options);
% copt= global min for the objective c'x
% xopt= minimizing value of the vector x
% IN OUR CASE, LAST TERM OF xopt IS EQUAL TO copt
% BY DEFINITION OF gamma

gamma_square=dec2mat(case2,xopt,\gamma_2);
Q_var=dec2mat(case2,xopt,Q);
F_var=dec2mat(case2,xopt,F);
P=inv(Q_var);
k=F_var*P;

\gamma_old_2=gamma_square;
gamma_list=[gamma_list sqrt(gamma_square)];
k_list=[k_list k'];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NOW SOLVE SEQUENTIALLY THE REST OF THE RINGS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=2:1:length(delta)
    [kk,gammak]=sequence_gamma(gamma_old_2,delta(i),P,A,B1,B2,C,D,...
    ...alpha,w_2,u_lim_2);
    gamma_list=[gamma_list gammak];
    k_list=[k_list kk];
gamma_old_2 = gammak^2;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FINAL RESULTS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delta
gamma_list
k_list

plot(delta,gamma_list,'r')
Function for the inner ellipsoids

function [kk,gammak]=sequence_gamma(gammak_1_2,delta,P,A,B1,B2,C,...
    D,alpha,w_2,u_lim_2)

    % First define the coefficients
    Q=inv(P);
    Q=inv(P) k is a scalar
    %SYSTEM FOR THE MINIMIZATION gamma is a variable
    setlmis([ ]); % kQ=FQ=inv(P) k is a scalar
    k=lmivar(2,[1 2]); % (rectangular, [1 row, 2 columns])
    gamma_2=lmivar(1,[1 1]);

    lmiterm([2 1 1 0],A*Q+Q*A');
    lmiterm([2 1 1 k],B2,Q,'s');
    lmiterm([2 2 1 0],B1');
    lmiterm([2 2 2 gamma_2],-1,1);
    lmiterm([2 2 1 0],C*Q);
    lmiterm([2 2 1 k],D,Q);
    lmiterm([2 2 2 0],0);
    lmiterm([2 2 3 0],-1);

    lmiterm([-1 1 1 0],Q);
    lmiterm([-1 2 1 k],1,Q);
    lmiterm([-1 2 2 0],(delta^2)*u_lim_2/w_2);

    lmiterm([3 1 1 gamma_2],1,1);
    lmiterm([3 1 1 0],-gammak_1_2);

    case2=getlmis;
    n=decnbr(case2);
    c=zeros(1,n-1);
    c=[c 1];

    options=[1e-5 1e3 1e9 10 0];
    [copt,xopt]=mincx(case2,c,options);

    gamma_square=dec2mat(case2,xopt,gamma_2);
    kk=dec2mat(case2,xopt,k);
    kk=kk';
    gammak=sqrt(gamma_square);
A.2 Matlab code for \( P_{\text{global}} \) constant

State Space Scheduling

clear
clc
tic

%%% first we define the coefficients
C=[5 0];
w_2=1000^2; % o 100
u_lim_2=1000^2;%
A=[0 1; -10 -10];% A=[-1 5; -10 -15];
B1=[0.5 1]’;
B2=[0 6]’;
D=0;%
alpha=1;%
delta=[1 2.5 5 10 15];
delta2=delta(2); delta3=delta(3);
delta4=delta(4); delta5=delta(5);
cte1=1; cte2=1; cte3=1; cte4=1; cte5=1;
delta_list=[delta(1) delta2 delta3 delta4 delta5];

% SYSTEM FOR THE MINIMIZATION betha_is a variable
setlmis([ ]); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Q=lmivar(1,[2 1]);
F1=lmivar(2,[1 2]);
F2=lmivar(2,[1 2]);
F3=lmivar(2,[1 2]);
F4=lmivar(2,[1 2]);
F5=lmivar(2,[1 2]);

gamma1_2=lmivar(1,[1 1]);
gamma2_2=lmivar(1,[1 1]);
gamma3_2=lmivar(1,[1 1]);
gamma4_2=lmivar(1,[1 1]);
gamma5_2=lmivar(1,[1 1]);
lmiterm([1 1 1 Q], A, 1, 's');
lmiterm([1 1 1 F1], B2, 1, 's');
lmiterm([1 1 1 Q], alpha, 1);
lmiterm([1 2 1 0], B1');
lmiterm([1 2 2 0], -alpha);

lmiterm([-2 1 1 Q], 1, 1);
lmiterm([-2 2 1 F1], 1, 1);
lmiterm([-2 2 2 0], u_lim_2/w_2);

lmiterm([3 1 1 Q], A, 1, 's');
lmiterm([3 1 1 F1], B2, 1, 's');
lmiterm([3 2 1 0], B1');
lmiterm([3 2 2 gamma1_2], -1, 1);
lmiterm([3 3 1 Q], C, 1);
lmiterm([3 3 1 F1], D, 1);
lmiterm([3 3 2 0], 0);
lmiterm([3 3 3 0], -1);

%%%}
lmiterm([-4 1 1 Q], 1, 1);
lmiterm([-4 2 1 F2], 1, 1);
lmiterm([-4 2 2 0], (delta2^2)*u_lim_2/w_2);

lmiterm([5 1 1 Q], A, 1, 's');
lmiterm([5 1 1 F2], B2, 1, 's');
lmiterm([5 2 1 0], B1');
lmiterm([5 2 2 gamma2_2], -1, 1);
lmiterm([5 3 1 Q], C, 1);
lmiterm([5 3 1 F2], D, 1);
lmiterm([5 3 2 0], 0);
lmiterm([5 3 3 0], -1);

%%%}
lmiterm([-6 1 1 Q], 1, 1);
lmiterm([-6 2 1 F3], 1, 1);
lmiterm([-6 2 2 0], (delta3^2)*u_lim_2/w_2);

lmiterm([7 1 1 Q], A, 1, 's');
lmiterm([7 1 1 F3], B2, 1, 's');
lmiterm([7 2 1 0], B1');

186
lmiterm([7 2 2 gamma3_2],-1,1);
lmiterm([7 3 1 Q],C,1);
lmiterm([7 3 1 F3],D,1);
lmiterm([7 3 2 0],0);
lmiterm([7 3 3 0],-1);

%%% 

lmiterm([-8 1 1 Q],1,1);
lmiterm([-8 2 1 F4],1,1);
lmiterm([-8 2 2 0],(delta4^2)*u_lim_2/w_2);

lmiterm([9 1 1 Q],A,1,'s');
lmiterm([9 1 1 F4],B2,1,'s');
lmiterm([9 2 1 0],B1');
lmiterm([9 2 2 gamma4_2],-1,1);
lmiterm([9 3 1 Q],C,1);
lmiterm([9 3 1 F4],D,1);
lmiterm([9 3 2 0],0);
lmiterm([9 3 3 0],-1);

%%% 

lmiterm([-10 1 1 Q],1,1);
lmiterm([-10 2 1 F5],1,1);
lmiterm([-10 2 2 0],(delta5^2)*u_lim_2/w_2);

lmiterm([11 1 1 Q],A,1,'s');
lmiterm([11 1 1 F5],B2,1,'s');
lmiterm([11 2 1 0],B1');
lmiterm([11 2 2 gamma5_2],-1,1);
lmiterm([11 3 1 Q],C,1);
lmiterm([11 3 1 F5],D,1);
lmiterm([11 3 2 0],0);
lmiterm([11 3 3 0],-1);

%%% 

lmiterm([-12 1 1 gamma1_2],1,1);
lmiterm([-12 1 1 gamma2_2],-1,1);

lmiterm([-13 1 1 gamma2_2],1,1);
lmiterm([-13 1 1 gamma3_2],-1,1);
lmiterm([-14 1 1 gamma3_2],1,1);
lmiterm([-14 1 1 gamma4_2],-1,1);
lmiterm([-15 1 1 gamma4_2],1,1);
lmiterm([-15 1 1 gamma5_2],-1,1);

case2=getlmis;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 n=decnbr(case2);
c=zeros(1,n-5);
c=[c cte1 cte2 cte3 cte4 cte5];
options=[1e-6 1e3 1e9 10 0];
[copt,xopt]=mincx(case2,c,options);

gamma1_square=dec2mat(case2,xopt,gamma1_2);
gamma2_square=dec2mat(case2,xopt,gamma2_2);
gamma3_square=dec2mat(case2,xopt,gamma3_2);
gamma4_square=dec2mat(case2,xopt,gamma4_2);
gamma5_square=dec2mat(case2,xopt,gamma5_2);

gamma1=sqrt(gamma1_square);
gamma2=sqrt(gamma2_square);
gamma3=sqrt(gamma3_square);
gamma4=sqrt(gamma4_square);
gamma5=sqrt(gamma5_square);

gammas=[gamma1 gamma2 gamma3 gamma4 gamma5];

Q_var=dec2mat(case2,xopt,Q);
F1_var=dec2mat(case2,xopt,F1);
F2_var=dec2mat(case2,xopt,F2);
F3_var=dec2mat(case2,xopt,F3);
F4_var=dec2mat(case2,xopt,F4);
F5_var=dec2mat(case2,xopt,F5);

P=inv(Q_var)
k_list=[(F1_var*P)' (F2_var*P)' (F3_var*P)' (F4_var*P)'(F5_var*P)'];
A.3 Matlab code for $P_{\text{sequential}}^{\text{variable}}$, considering $\dot{r} \leq 0$.

State Space Scheduling

Main file

clear clc tic

% first we define the coefficients
C=[5 0];
w_2=1000^2;
u_lim_2=1000^2;
A=[0 1; -10 -10];
B1=[0.5 1]';
B2=[0 6]';
D=0;
alpha=1;
Q=[ ]; P=[ ]; k=[ ]; R=[ ]; gamma_list=[ ];
r=(u_lim_2)/(w_2);
gamma_alpha=[ ];

% First part OUTER RING

setlmis([ ]); Q1=lmivar(1,[2 1]); F1=lmivar(2,[1 2]); gamma_1_2=lmivar(1,[1 1]);

lmiterm([1 1 1 Q1],A,1,'s');
lmiterm([1 1 1 F1],B2,1,'s');
lmiterm([1 2 1 0],B1');
lmiterm([1 2 2 gamma_1_2],-1,1);
lmiterm([1 3 1 Q1],C,1);
lmiterm([1 3 1 F1],D,1);
lmiterm([1 3 2 0],0);
lmiterm([1 3 3 0],-1);

lmiterm([3 1 1 Q1],A,1,'s');
lmiterm([3 1 1 F1],B2,1,'s');
lmiterm([3 1 1 Q1],alpha,1);
lmiterm([3 2 1 0],B1');
lmiterm([3 2 2 0],-alpha);
```matlab
lmiterm([-2 1 1 Q1],1,1);
lmiterm([-2 2 1 F1],1,1);
lmiterm([-2 2 0],r*delta(1)^2);
case2=getlmis;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=decnbr(case2);
c=zeros(1,n-1);
c=[c 1];  %we cancel all the internal variables but
            %the last, which corresponds to beta_2
            %since it was the last one that we defined.
options=[1e-7 1e3 1e9 10 0];  
            %[accuracy on optimal value
            % max n iterations
            % feasibility radius
            % tolerance on iterations for optimization
            % turns off the trace of execution]
[copt,xopt]=mincx(case2,c,options);
            %copt=global minimum for the objective c’x
            %xopt= minimizing value of the vector x
            %IN OUR CASE, LAST TERM OF xopt IS EQUAL
            %TO copt BY DEFINITION OF bet
gamma_list=[gamma_list sqrt(dec2mat(case2,xopt,gamma1_2))];
gamma=[sqrt(dec2mat(case2,xopt,gamma1_2))];
Q1=dec2mat(case2,xopt,Q1);
F1=dec2mat(case2,xopt,F1);
Q=Q1;
P=inv(Q1);
k=[(F1*P)'];
R=[r*delta(1)^2];
Q_old=Q1;
gamma_old_2=gamma^2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=2:1:length(delta)
    [Qi,Pi,Fi,ki,gammai]=inner_ellipsoids(delta(i-1),delta(i),r,Q_old,...
        ...gamma_old_2,A,B1,B2,C,D,alpha);

    gamma=[gamma gammai];
    Q=[Q Qi];
P=[P Pi];
k=[k ki'];
R=[R r*delta(i)^2];
Q_old=Qi;

```

190
gamma_old_2 = gamma_i^2;
end;
toc

Function for the inner ellipsoids

function [Qk,Pk,Fk,kk,gammak]=inner_ellipsoids(deltak_1,deltak,r,Qk_1,...
...gammak_1_2,A,B1,B2,C,D,alpha)

setlmis([ ]);  
Qk=lmivar(1,[2 1]);
Fk=lmivar(2,[1 2]);
gammak_2=lmivar(1,[1 1]);

lmiterm([1 1 1 Qk],A,1,'s');
lmiterm([1 1 1 Fk],B2,1,'s');
lmiterm([1 2 1 0],B1');
lmiterm([1 2 2 gammak_2],-1,1);
lmiterm([1 3 1 Qk],C,1);
lmiterm([1 3 1 Fk],D,1);
lmiterm([1 3 2 0],0);
lmiterm([1 3 3 0],-1);

lmiterm([-2 1 1 Qk],1,1);
lmiterm([-2 1 1 Fk],1,1);
lmiterm([-2 2 2 0],r*deltak^2);

lmiterm([4 1 1 Qk],1,1);
lmiterm([4 1 1 0],-Qk_1);

lmiterm([3 1 1 gammak_2],1,1);
lmiterm([3 1 1 0],-gammak_1_2);

% case2=getlmis;
% n=decnbr(case2);
c=zeros(1,n-1);
c=[c 1]; options=[1e-7 1e3 1e9 10 0];
[copt,xopt]=mincx(case2,c,options);
gammak=sqrt(dec2mat(case2,xopt,gammak_2));
Qk=dec2mat(case2,xopt,Qk);
Fk=dec2mat(case2,xopt,Fk);
Pk=inv(Qk);
kk=Fk*Pk;
A.4 Matlab code for $P_{\text{global}}^{\text{variable}}$, considering $\dot{r} \leq 0$.  

State Space Scheduling

clear clc tic

% first we define the coefficients

C=[5 0];
w_2=1000^2;
u_lim_2=1000^2;
A=[0 1;-10 -10];
B1=[0.5 1]';
B2=[0 6]';
D=0;
alpha=1;

delta1=1.0; delta2=5; delta3=10; delta4=15; delta5=20;
cte1=1; cte2=1; cte3=1; cte4=1; cte5=1;
delta=[delta1 delta2 delta3 delta4 delta5];
r=u_lim_2/w_2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%SYSTEM FOR THE MINIMIZATION betha_is a variable
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

setlmis([],);
Q1=lmivar(1,[2 1]);
Q2=lmivar(1,[2 1]);
Q3=lmivar(1,[2 1]);
Q4=lmivar(1,[2 1]);
Q5=lmivar(1,[2 1]);
F1=lmivar(2,[1 2]);
F2=lmivar(2,[1 2]);
F3=lmivar(2,[1 2]);
F4=lmivar(2,[1 2]);
F5=lmivar(2,[1 2]);
gamma1_2=lmivar(1,[1 1]);
gamma2_2=lmivar(1,[1 1]);
gamma3_2=lmivar(1,[1 1]);
gamma4_2=lmivar(1,[1 1]);
gamma5_2=lmivar(1,[1 1]);
lmiterm([1 1 1 Q1],A,1,'s');
lmiterm([1 1 1 F1],B2,1,'s');
lmiterm([1 1 1 Q1],alpha,1);
lmiterm([1 2 1 O],B1');
lmiterm([1 2 2 O],-alpha);

lmiterm([-2 1 1 Q1],1,1);
lmiterm([-2 2 1 F1],1,1);
lmiterm([-2 2 2 O],r*delta(1)^2);

lmiterm([3 1 1 Q1],A,1,'s');
lmiterm([3 1 1 F1],B2,1,'s');
lmiterm([3 2 1 O],B1');
lmiterm([3 2 2 gamma1_2],-1,1);
lmiterm([3 3 1 Q1],C,1);
lmiterm([3 3 1 F1],D,1);
lmiterm([3 3 2 O],0);
lmiterm([3 3 3 O],-1);

%%%

lmiterm([-5 1 1 Q2],1,1);
lmiterm([-5 2 1 F2],1,1);
lmiterm([-5 2 2 O],r*delta(2)^2);

lmiterm([4 1 1 Q2],A,1,'s');
lmiterm([4 1 1 F2],B2,1,'s');
lmiterm([4 2 1 O],B1');
lmiterm([4 2 2 gamma2_2],-1,1);
lmiterm([4 3 1 Q2],C,1);
lmiterm([4 3 1 F2],D,1);
lmiterm([4 3 2 O],0);
lmiterm([4 3 3 O],-1);

lmiterm([6 1 1 Q2],1,1);
lmiterm([6 1 1 Q1],-1,1);

%%%

lmiterm([7 1 1 Q3],A,1,'s');
lmiterm([7 1 1 F3],B2,1,'s');
lmiterm([7 2 1 O],B1');
lmiterm([7 2 2 gamma3_2],-1,1);
lmiterm([7 3 1 Q3],C,1);
lmiterm([7 3 1 F3],D,1);
lmiterm([7 3 2 O],0);
lmiterm([7 3 3 O],-1);
lmiterm([-8 1 1 Q3],1,1);
lmiterm([-8 2 1 F3],1,1);
lmiterm([-8 2 2 0],r*delta(3)^2);

lmiterm([9 1 1 Q3],1,1);
lmiterm([9 1 1 Q2],-1,1);

\%
\%
lmiterm([10 1 1 Q4],A,1,'s');
lmiterm([10 1 1 F4],B2,1,'s');
lmiterm([10 2 1 0],B1');
lmiterm([10 2 2 gamma4_2],-1,1);
lmiterm([10 3 1 Q4],C,1);
lmiterm([10 3 1 F4],D,1);
lmiterm([10 3 2 0],0);
lmiterm([10 3 3 0],-1);

lmiterm([-11 1 1 Q4],1,1);
lmiterm([-11 2 1 F4],1,1);
lmiterm([-11 2 2 0],r*delta(4)^2);

lmiterm([12 1 1 Q4],1,1);
lmiterm([12 1 1 Q3],-1,1);

\%
\%
lmiterm([13 1 1 Q5],A,1,'s');
lmiterm([13 1 1 F5],B2,1,'s');
lmiterm([13 2 1 0],B1');
lmiterm([13 2 2 gamma5_2],-1,1);
lmiterm([13 3 1 Q5],C,1);
lmiterm([13 3 1 F5],D,1);
lmiterm([13 3 2 0],0);
lmiterm([13 3 3 0],-1);

lmiterm([-14 1 1 Q5],1,1);
lmiterm([-14 2 1 F5],1,1);
lmiterm([-14 2 2 0],r*delta(5)^2);

lmiterm([15 1 1 Q5],1,1);
lmiterm([15 1 1 Q4],-1,1);

\%
\%
lmiterm([-28 1 1 gamma1_2],1,1);
lmiterm([-28 1 1 gamma2_2],-1,1);

lmiterm([-29 1 1 gamma2_2],1,1);
lmiterm([-29 1 1 gamma3_2],-1,1);
lmiterm([-30 1 1 gamma3_2],1,1);
lmiterm([-30 1 1 gamma4_2],-1,1);
lmiterm([-31 1 1 gamma4_2],1,1);
lmiterm([-31 1 1 gamma5_2],-1,1);

case2=getlmis;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

n=decnbr(case2);
c=zeros(1,n-5);
c=[c cte1 cte2 cte3 cte4 cte5 ];
options=[1e-6 1e3 1e9 10 0];
[copt,xopt]=mincx(case2,c,options);

gamma1_square=dec2mat(case2,xopt,gamma1_2);
gamma2_square=dec2mat(case2,xopt,gamma2_2);
gamma3_square=dec2mat(case2,xopt,gamma3_2);
gamma4_square=dec2mat(case2,xopt,gamma4_2);
gamma5_square=dec2mat(case2,xopt,gamma5_2);
gamma1=sqrt(gamma1_square);
gamma2=sqrt(gamma2_square);
gamma3=sqrt(gamma3_square);
gamma4=sqrt(gamma4_square);
gamma5=sqrt(gamma5_square);
gamma_sec=[gamma1 gamma2 gamma3 gamma4 gamma5]

F1_var=dec2mat(case2,xopt,F1);
F2_var=dec2mat(case2,xopt,F2);
F3_var=dec2mat(case2,xopt,F3);
F4_var=dec2mat(case2,xopt,F4);
F5_var=dec2mat(case2,xopt,F5);
Q1=dec2mat(case2,xopt,Q1);
Q2=dec2mat(case2,xopt,Q2);
Q3=dec2mat(case2,xopt,Q3);
Q4=dec2mat(case2,xopt,Q4);
Q5=dec2mat(case2,xopt,Q5);

k_list=[(F1_var*inv(Q1))' (F2_var*inv(Q2))' (F3_var*inv(Q3))'...
     ...(F4_var*inv(Q4))' (F5_var*inv(Q5))']

toc
plot(delta,gamma_sec,'r')
A.5 Matlab code for $3 - CM_3$, considering $\dot{r} \leq 0$.

State Space Scheduling

Main file

clear clc tic

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%first we define the coefficients
w_2=1000^2;
u_lim_2=1000^2;
A=[0 1; -10 -10];
B1=[0.5 1]';
B2=[0 6]';
C=[5 0];
D=0;
alpha=1;%
delta=[1 1.5 2.5 5 7.5 10 15 20 50 100];
kappa=delta(length(delta))/3
gamma=[ ]; Q=[ ]; P=[ ]; k=[ ]; R=[ ];
r=(u_lim_2)/(w_2);
E_counter=1;
final=0;
go_to_pure_sequential=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% % First part OUTER RING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% i=[E_counter E_counter+1 E_counter+2];

[Qi,Pi,ki,gamma]=global_step_initial(kappa,i,delta,r,A,B1,B2,C,...
...D,alpha);

gamma=[gamma gammai(1)];
Q=[Q Qi(:,:1:2)];
P=[P Pi(:,:1:2)];
k=[k ki(:,1)];
R=[R r*delta(E_counter)^2];
Q_old=Qi(:,1:2);
gamma_old_2=gammai(1)^2;
E_counter=2;
while final==0 && E_counter<=length(delta)

    if E_counter==(length(delta)-2) && go_to_pure_sequential==0
        final=1;
    end;
    i=[E_counter E_counter+1 E_counter+2];

    if go_to_pure_sequential==1
        [Qi,Pi,ki,gammai]=inner_ellipsoids(delta(E_counter),r,Q_old,...
            ...gamma_old_2,A,B1,B2,C,D, alpha);
    else
        [Qi,Pi,ki,gammai]=global_step(kappa,final,i,delta,r,Q_old,...
            ...gamma_old_2,A,B1,B2,C,D);
        if i(3)==31
            tre=4
        end
    end;

    improvement=(sqrt(gamma_old_2)-gammai(1))/(delta(E_counter)...
        ...-delta(E_counter-1));

    if improvement<0.01 && go_to_pure_sequential==0;
        go_to_pure_sequential=1;
    end;

    if final==1
        gamma=[gamma gammai];
        Q=[Q Qi];
        P=[P Pi];
        k=[k ki];
        R=[R r*delta(E_counter)^2 r*delta(E_counter+1)^2 ...
            ...r*delta(E_counter+2)^2];
    else
        gamma=[gamma gammai(1)];
        Q=[Q Qi(:,1:2)];
        P=[P Pi(:,1:2)];
        k=[k ki(:,1)];

    end;
R=[R r*delta(E_counter)^2];
Q_old=Qi(:,1:2);
gamma_old=gamma(1)^2;
end;
E_counter=E_counter+1;
end;
toc

Function for the outer ellipsoid

function
[Q,P,k,gamma]=global_step_initial(kappa,i,delta,r,A,B1,B2,C,D,alpha);
delta1=delta(i(1));
delta2=1+kappa;
delta3=1+2*kappa;
cte1=1; cte2=1; cte3=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%SYSTEM FOR THE MINIMIZATION betha_is a variable
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
setlmis([ ]); Q1=lmivar(1,[2 1]);
Q2=lmivar(1,[2 1]);
Q3=lmivar(1,[2 1]);

F1=lmivar(2,[1 2]);
F2=lmivar(2,[1 2]);
F3=lmivar(2,[1 2]);

gamma_1_2=lmivar(1,[1 1]);
gamma_2_2=lmivar(1,[1 1]);
gamma_3_2=lmivar(1,[1 1]);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
limiter([-1 1 1 Q1],1,1);
limiter([-1 2 1 F1],1,1);
limiter([-1 2 2 0],r*delta1^2);
\texttt{lmiterm([2 1 1 Q1],A,1,'s');}
\texttt{lmiterm([2 1 1 F1],B2,1,'s');}
\texttt{lmiterm([2 2 1 0],B1');}
\texttt{lmiterm([2 2 2 gamma1_2],-1,1);}
\texttt{lmiterm([2 3 1 Q1],C,1);}
\texttt{lmiterm([2 3 1 F1],D,1);}
\texttt{lmiterm([2 3 2 0],0);}
\texttt{lmiterm([2 3 3 0],-1);}
\texttt{lmiterm([3 1 1 Q1],A,1,'s');}
\texttt{lmiterm([3 1 1 F1],B2,1,'s');}
\texttt{lmiterm([3 1 1 Q1],alpha,1);}
\texttt{lmiterm([3 2 1 0],B1');}
\texttt{lmiterm([3 2 2 0],-alpha);}

%%%
\texttt{lmiterm([-4 1 1 Q2],1,1);}
\texttt{lmiterm([-4 2 1 F2],1,1);}
\texttt{lmiterm([-4 2 2 0],r*delta2^2);}
\texttt{lmiterm([5 1 1 Q2],A,1,'s');}
\texttt{lmiterm([5 1 1 F2],B2,1,'s');}
\texttt{lmiterm([5 2 1 0],B1');}
\texttt{lmiterm([5 2 2 gamma2_2],-1,1);}
\texttt{lmiterm([5 3 1 Q2],C,1);}
\texttt{lmiterm([5 3 1 F2],D,1);}
\texttt{lmiterm([5 3 2 0],0);}
\texttt{lmiterm([5 3 3 0],-1);}
\texttt{lmiterm([6 1 1 Q2],1,1);}
\texttt{lmiterm([6 1 1 Q1],-1,1);}

%%%
\texttt{lmiterm([7 1 1 Q3],A,1,'s');}
\texttt{lmiterm([7 1 1 F3],B2,1,'s');}
\texttt{lmiterm([7 2 1 0],B1');}
\texttt{lmiterm([7 2 2 gamma3_2],-1,1);}
\texttt{lmiterm([7 3 1 Q3],C,1);}
\texttt{lmiterm([7 3 1 F3],D,1);}
\texttt{lmiterm([7 3 2 0],0);}
\texttt{lmiterm([7 3 3 0],-1);}
lmiterm([-8 1 1 Q3],1,1);
lmiterm([-8 2 1 F3],1,1);
lmiterm([-8 2 2 0],r*delta3^2);

lmiterm([9 1 1 Q3],1,1);
lmiterm([9 1 1 Q2],-1,1);

%%%%

lmiterm([-10 1 1 gamma1_2],1,1);
lmiterm([-10 1 1 gamma2_2],-1,1);
lmiterm([-11 1 1 gamma2_2],1,1);
lmiterm([-11 1 1 gamma3_2],-1,1);

% case 2 = getlmis;

n=decnbr(case2);
c=zeros(1,n-3);
c=[c cte1 cte2 cte3];
options=[1e-6 1e3 1e9 10 0];
[copt,xopt]=mincx(case2,c,options);
gamma1=sqrt(dec2mat(case2,xopt,gamma1_2));
gamma2=sqrt(dec2mat(case2,xopt,gamma2_2));
gamma3=sqrt(dec2mat(case2,xopt,gamma3_2));
gamma=[gamma1 gamma2 gamma3];
F1_var=dec2mat(case2,xopt,F1);
F2_var=dec2mat(case2,xopt,F2);
F3_var=dec2mat(case2,xopt,F3);
F=[F1' F2' F3'];

Q1=dec2mat(case2,xopt,Q1);
Q2=dec2mat(case2,xopt,Q2);
Q3=dec2mat(case2,xopt,Q3);
Q=[Q1 Q2 Q3];

P=[inv(Q1) inv(Q2) inv(Q3)];

k1=(F1_var*inv(Q1));
k2=(F2_var*inv(Q2));
k3=(F3_var*inv(Q3));
k=[k1' k2' k3'];
Function for the inner ellipsoids

function

\[ [Q,P,k,\gamma] = \text{global-step}(kappa, final, i, delta, r, Q\_old, \gamma\_old\_2, A, \ldots \ B1, B2, C, D); \]

delta1 = delta(i(1));
delta2 = delta(i(1)) + kappa;
delta3 = delta(i(1)) + 2 * kappa;

if final ~= 0
    delta2 = delta(i(2));
    delta3 = delta(i(3));
end;

cte1 = 1; cte2 = 1; cte3 = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SYSTEM FOR THE MINIMIZATION \( \beta_{\text{h}} \) is a variable
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

setlmis([ ]);
Q1 = lmivar(1, [2 1]);
Q2 = lmivar(1, [2 1]);
Q3 = lmivar(1, [2 1]);

F1 = lmivar(2, [1 2]);
F2 = lmivar(2, [1 2]);
F3 = lmivar(2, [1 2]);

gamma1_2 = lmivar(1, [1 1]);
gamma2_2 = lmivar(1, [1 1]);
gamma3_2 = lmivar(1, [1 1]);

lmiterm([-1 1 1 Q1], 1, 1);
lmiterm([-1 2 1 F1], 1, 1);
lmiterm([-1 2 2 0], r * delta1 - 2);

lmiterm([2 1 1 Q1], A, 1, 's');
lmiterm([2 1 1 F1], B2, 1, 's');
lmiterm([2 2 0], B1');
lmiterm([2 2 2 gamma1_2], -1, 1);
lmiterm([2 3 1 Q1], C, 1);
lmiterm([2 3 1 F1], D, 1);
lmiterm([2 3 2 0], 0);
lmiterm([2 3 3 0], -1);

201
lmiterm([3 1 1 Q1],1,1);
lmiterm([3 1 1 0],-Q_old);

%%%
lmiterm([-4 1 1 Q2],1,1);
lmiterm([-4 2 1 F2],1,1);
lmiterm([-4 2 2 0],r*delta2^2);

lmiterm([5 1 1 Q2],A,1,'s');
lmiterm([5 1 1 F2],B2,1,'s');
lmiterm([5 2 1 0],B1');
lmiterm([5 2 2 gamma2_2],-1,1);
lmiterm([5 3 1 Q2],C,1);
lmiterm([5 3 1 F2],D,1);
lmiterm([5 3 2 0],0);
lmiterm([5 3 3 0],-1);

lmiterm([6 1 1 Q2],1,1);
lmiterm([6 1 1 Q1],-1,1);

%%%
lmiterm([7 1 1 Q3],A,1,'s');
lmiterm([7 1 1 F3],B2,1,'s');
lmiterm([7 2 1 0],B1');
lmiterm([7 2 2 gamma3_2],-1,1);
lmiterm([7 3 1 Q3],C,1);
lmiterm([7 3 1 F3],D,1);
lmiterm([7 3 2 0],0);
lmiterm([7 3 3 0],-1);

lmiterm([-8 1 1 Q3],1,1);
lmiterm([-8 2 1 F3],1,1);
lmiterm([-8 2 2 0],r*delta3^2);

lmiterm([9 1 1 Q3],1,1);
lmiterm([9 1 1 Q2],-1,1);

%%%
lmiterm([-10 1 1 gamma1_2],-1,1);
lmiterm([-10 1 1 0],gamma_old_2);

lmiterm([-11 1 1 gamma1_2],1,1);
lmiterm([-11 1 1 gamma2_2],-1,1);

lmiterm([-12 1 1 gamma2_2],1,1);
lmiterm([-12 1 1 gamma3_2],-1,1);
case2=getlmis;

function [Qk,Pk,kk,gammak]=inner_ellipsoids(deltak,r,Qk_1,gammak_1_2,A,B1,B2,...
...C,D,alpha)

    setlmis([ ]);
    Qk=lmivar(1,[2 1]);
    Fk=lmivar(2,[1 2]);
    gammak_2=lmivar(1,[1 1]);
lmitem([1 1 1 Qk],A,1,'s');
lmitem([1 1 1 Fk],B2,1,'s');
lmitem([1 2 1 0],B1');
lmitem([1 2 2 gammak_2],-1,1);
lmitem([1 3 1 Qk],C,1);
lmitem([1 3 1 Fk],D,1);
lmitem([1 3 2 0],0);
lmitem([1 3 3 0],-1);

lmitem([-2 1 1 Qk],1,1);
lmitem([-2 2 1 Fk],1,1);
lmitem([-2 2 2 0],r*deltak^2);

lmitem([4 1 1 Qk],1,1);
lmitem([4 1 1 0],-Qk_1);

lmitem([3 1 1 gammak_2],1,1);
lmitem([3 1 1 0],-gammak_1_2);

case2=getlmis;
%we cancel all the internal variables but the last, %which corresponds to beta_2 %since it was the last one that we defined.
options=[1e-7 1e3 1e9 10 0];
%[accuracy on optimal value / max n iterations / ... %feasibility radius / tolerance on iterations / ... %turns off the trace of execution]
[copt,xopt]=mincx(case2,c,options);
%copt= global minimum for the objective c’x
%xopt= minimizing value of the vector x
%IN OUR CASE, LAST TERM OF xopt IS EQUAL TO copt
%BY DEFINITION OF bet

gammak=sqrt(dec2mat(case2,xopt,gammak_2));
Qk=dec2mat(case2,xopt,Qk); Fk=dec2mat(case2,xopt,Fk); Pk=inv(Qk);
kk=(Fk*Pk)';
A.6 Matlab code for $3 - CM_3$, considering $\dot{r} \leq 0$.

Input Scheduling

This code is essentially the same as in appendix A.5, but adding the Reachable Set Condition for each value of $\delta$, as explained on Chapter 7.
A.7 Matlab code for $4 - CM_3$, considering $-\infty < \dot{r} \leq |d_M|$. 

Input Scheduling

Main file

clear clc tic
%
/* we define the system

w_2=1000^2;
u_lim_2=1000^2;
A=[0 1; -10 -10];
B1=[0.5 1]’;
B2=[0 6]’;
C=[5 0];
D=0;
alpha=1;
delta=[1 1.5 2.5 5 7.5 10 15 20 50 100];
%
% kappa=delta(length(delta))/3;
gamma=[]; Q=[]; P=[]; k=[]; R=[];
r=(u_lim_2)/(w_2);
E_counter=1;
final=0;
go_to_pure_sequential=0;
%
% First part OUTER RING
%
i=[E_counter E_counter+1 E_counter+2];
[Qi,Pi,ki,gammai]=global_step_initial(kappa,i,delta,r,A,B1,B2,C,D,...
...alpha,dM);

gamma=[gamma gamma(1)];
Q=[Q Qi(:,1:2)];
P=[P Pi(:,1:2)];
k=[k ki(:,1)];
R=[R r*delta(i(1))^2 ];
Q_old=Qi(:,3:4);
 gamma_old_2=gamma(2)^2;
K_old=ki(:,2);
E_counter=2;

while final==0 && E_counter<=length(delta)

    if E_counter==(length(delta)-2) && go_to_pure_sequential==0
        final=1;
    elseif E_counter==(length(delta)-1) && go_to_pure_sequential==1
        final=1;
    end;
    i=[E_counter E_counter+1 E_counter+2];
    if go_to_pure_sequential==1
        [Qi,Pi,ki,gammai]=inner_ellipsoids(delta,i,r,Q_old,...
                                                ...gamma_old_2,K_old,A,B1,B2,C,D,alpha,dM);
    else
        [Qi,Pi,ki,gammai]=global_step(kappa,final,i,delta,r,Q_old,...
                                                ...gamma_old_2,K_old,A,B1,B2,C,D,alpha,dM);
    end;
    gamma=[gamma gammai(1)];
    Q=[Q Qi(:,1:2)];
    P=[P Pi(:,1:2)];
    k=[k ki(:,1)];
    R=[R r*delta(i(1))^2];
    Q_old=Qi(:,3:4);
    gamma_old_2=gammai(2)^2;
    K_old=ki(:,2);
    improvement=abs((gammai(1)-gammai(2))/(delta(i(1))-delta(i(2))));
    if improvement<0.01 && go_to_pure_sequential==0;
        go_to_pure_sequential=1;
    end;
    if final==1 && go_to_pure_sequential==0
        gamma=[gamma gammai(2:3)];
        Q=[Q Qi(:,3:6)];
        P=[P Pi(:,3:6)];
        k=[k ki(:,2:3)];
        R=[R r*delta(E_counter+1)^2 r*delta(E_counter+2)^2];
    elseif final==1 && go_to_pure_sequential==1
        gamma=[gamma gammai(2)];
        Q=[Q Qi(:,3:4)];
        P=[P Pi(:,3:4)];
        k=[k ki(:,2)];
        R=[R r*delta(E_counter+1)^2 ];
    end;

end;
E_counter=E_counter+1;
end;

Function for the inner ellipsoids

function [Q,P,k,gamma]=global_step_initial(kappa,i,delta,r,A,B1,...
...B2,C,D,alpha,dM);

delta1=delta(i(1));
delta2=delta(i(2));
delta3=delta(i(2))+kappa;
cte1=1;
cte2=1;
cte3=1;
Dr12=r*(delta2^2-1);
Dr23=r*(delta3^2-delta2^2);

%SYSTEM FOR THE MINIMIZATION betha_is a variable
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

setlmis([ ]);
Q1=lmivar(1,[2 1]);
Q2=lmivar(1,[2 1]);
Q3=lmivar(1,[2 1]);
F1=lmivar(2,[1 2]);
F2=lmivar(2,[1 2]);
F3=lmivar(2,[1 2]);
gamma1_2=lmivar(1,[1 1]);
gamma2_2=lmivar(1,[1 1]);
gamma3_2=lmivar(1,[1 1]);

% Q1 system % % % % % % % % % % % % % % % % % % % % % % % % % % %

%Saturation LMI
lmiterm([-1 1 1 Q1],1,1);
lmiterm([-1 2 1 F1],1,1);
lmiterm([-1 2 2 0],r*delta1^2);

%Performance LMI with negative $\dot{P}$
lmiterm([2 1 1 Q1],A,1,'s');
lmiterm([2 1 1 F1],B2,1,'s');
lmiterm([2 2 0],B1');
lmiterm([2 2 2 gamma1_2],-1,1);
\texttt{lmiterm([2 3 1 Q1],C,1);}
\texttt{lmiterm([2 3 1 F1],D,1);}
\texttt{lmiterm([2 3 2 0],0);}
\texttt{lmiterm([2 3 3 0],-1);}

\texttt{lmiterm([3 1 1 Q1],A,1,'s');}
\texttt{lmiterm([3 1 1 F1],B2,1,'s');}
\texttt{lmiterm([3 1 1 Q1],dM/Dr12,1); \%}
\texttt{lmiterm([3 1 1 Q2],-dM/Dr12,1); \%}
\texttt{lmiterm([3 2 1 0],B1');}
\texttt{lmiterm([3 2 2 gamma1_2],-1,1);}
\texttt{lmiterm([3 3 1 Q1],C,1);}
\texttt{lmiterm([3 3 1 F1],D,1);}
\texttt{lmiterm([3 3 2 0],0);}
\texttt{lmiterm([3 3 3 0],-1);}

\texttt{lmiterm([4 1 1 Q1],A,1,'s');}
\texttt{lmiterm([4 1 1 F1],B2,1,'s');}
\texttt{lmiterm([4 1 1 Q1],alpha,1);}
\texttt{lmiterm([4 2 1 0],B1');}
\texttt{lmiterm([4 2 2 0],-alpha);}

\texttt{lmiterm([5 1 1 Q1],A,1,'s');}
\texttt{lmiterm([5 1 1 F1],B2,1,'s');}
\texttt{lmiterm([5 1 1 Q1],alpha,1);}
\texttt{lmiterm([5 1 1 Q1],dM/Dr12,1); \%}
\texttt{lmiterm([5 1 1 Q2],-dM/Dr12,1); \%}
\texttt{lmiterm([5 2 1 0],B1');}
\texttt{lmiterm([5 2 2 0],-alpha);}

\texttt{\% Q2 system \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \%}

\texttt{\% Saturation LMI}
\texttt{lmiterm([-6 1 1 Q2],1,1);}
\texttt{lmiterm([-6 2 1 F2],1,1);}
\texttt{lmiterm([-6 2 2 0],r*delta2^2);}

\texttt{\% Performance LMI with negative \;\textit{\texttt{dot}}\{P\}}
\texttt{lmiterm([7 1 1 Q2],A,1,'s');}
\texttt{lmiterm([7 1 1 F2],B2,1,'s');}
\texttt{lmiterm([7 2 1 0],B1');}
\texttt{lmiterm([7 2 2 gamma2_2],-1,1);
lmiterm([7 3 1 Q2],C,1);
lmiterm([7 3 1 F2],D,1);
lmiterm([7 3 2 0],0);
lmiterm([7 3 3 0],-1);

%Performance LMI with positive $\dot{P}$
lmiterm([8 1 1 Q2],A,1,'s');
lmiterm([8 1 1 F2],B2,1,'s');
lmiterm([8 1 1 Q2],dM/Dr23,1); %
lmiterm([8 1 1 Q3],-dM/Dr23,1); %
lmiterm([8 2 1 0],B1');
lmiterm([8 2 2 gamma2_2],-1,1);
lmiterm([8 3 1 Q2],C,1);
lmiterm([8 3 1 F2],D,1);
lmiterm([8 3 2 0],0);
lmiterm([8 3 3 0],-1);

%Reachable set LMI with negative $\dot{P}$
lmiterm([9 1 1 Q2],A,1,'s');
lmiterm([9 1 1 F2],B2,1,'s');
lmiterm([9 1 1 Q2],alpha,1);
lmiterm([9 2 1 0],B1');
lmiterm([9 2 2 0],-alpha);

%Reachable set LMI with positive $\dot{P}$
lmiterm([10 1 1 Q2],A,1,'s');
lmiterm([10 1 1 F2],B2,1,'s');
lmiterm([10 1 1 Q2],alpha,1);
lmiterm([10 1 1 Q2],dM/Dr23,1); %
lmiterm([10 1 1 Q3],-dM/Dr23,1); %
lmiterm([10 2 1 0],B1');
lmiterm([10 2 2 0],-alpha);

lmiterm([11 1 1 Q2],1,1);
lmiterm([11 1 1 Q1],-1,1);

% Q3 system
% For this case we don't have "extra terms" since we assume $Q4=Q3$

%Satisfaction LMI
lmiterm([-12 1 1 Q3],1,1);
lmiterm([-12 2 1 F3],1,1);
lmiterm([-12 2 2 0],r*delta3^2);

%Performance LMI
lmitus([13 1 1 Q3],A,1,’s’);
lmiterm([13 1 1 F3],B2,1,’s’);
lmiterm([13 2 1 0],B1’);
lmiterm([13 2 2 gamma3_2],-1,1);
lmiterm([13 3 1 Q3],C,1);
lmiterm([13 3 1 F3],D,1);
lmiterm([13 3 2 0],0);
lmiterm([13 3 3 0],-1);

%Reachable set LMI
lmiterm([14 1 1 Q3],A,1,’s’);
lmiterm([14 1 1 F3],B2,1,’s’);
lmiterm([14 1 1 Q3],alpha,1);
lmiterm([14 2 1 0],B1’);
lmiterm([14 2 2 0],-alpha);
lmiterm([15 1 1 Q3],1,1);
lmiterm([15 1 1 Q2],-1,1);

lmiterm([-16 1 1 gamma1_2],1,1);
lmiterm([-16 1 1 gamma2_2],-1,1);
lmiterm([-17 1 1 gamma2_2],1,1);
lmiterm([-17 1 1 gamma3_2],-1,1);
case2=getlmis;

n=decnbr(case2);
c=zeros(1,n-3);
c=[c cte1 cte2 cte3];
options=[1e-6 1e3 1e9 10 0];
[copt,xopt]=mincx(case2,c,options);
gamma1=sqrt(dec2mat(case2,xopt,gamma1_2));
gamma2=sqrt(dec2mat(case2,xopt,gamma2_2));
gamma3=sqrt(dec2mat(case2,xopt,gamma3_2));
gamma=[gamma1 gamma2 gamma3];
F1_var=dec2mat(case2,xopt,F1);
F2_var=dec2mat(case2,xopt,F2);
F3_var=dec2mat(case2,xopt,F3);
F=[F1’ F2’ F3’];
Q1=dec2mat(case2,xopt,Q1);
Q2=dec2mat(case2,xopt,Q2);
Q3=dec2mat(case2,xopt,Q3);
Q=[Q1 Q2 Q3];
P=[inv(Q1) inv(Q2) inv(Q3)];
k1=(F1_var*inv(Q1));

211
\[ k_2 = (F_2 \text{var} \times \text{inv}(Q_2)) \]
\[ k_3 = (F_3 \text{var} \times \text{inv}(Q_3)) \]
\[ k = [k_1', k_2', k_3'] \]

**Function for the inner ellipsoids**

```matlab
function [Q,P,k,gamma]=global_step(kappa,final,i,delta,r,Q_old,...
    ...gamma_old_2,K_old,A,B1,B2,C,D,alpha,dM);
```

```matlab
delta1=delta(i(1));
delta2=delta(i(2));
delta3=delta(i(2))+kappa;
if final~=0
    delta3=delta(i(3));
end;
cte1=1; cte2=1; cte3=1;
Dr12=r*(delta2^2-delta1^2);
Dr23=r*(delta3^2-delta2^2);
F_old=K_old'*Q_old;
```

```matlab

%SYSTEM FOR THE MINIMIZATION betha_is a variable
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

setlmis([]);
Q2=lmivar(1,[2 1]);
Q3=lmivar(1,[2 1]);
F2=lmivar(2,[1 2]);
F3=lmivar(2,[1 2]);
gamma2_2=lmivar(1,[1 1]);
gamma3_2=lmivar(1,[1 1]);

% Q1 system % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Saturation LMI
% NOT NECESSARY

% Performance LMI with negative \dot{P}
% NOT NECESSARY

% Performance LMI with positive \dot{P}
lmiterm([1 1 1 0],A*Q_old+Q_old*A');
lmiterm([1 1 1 0],B2*F_old+F_old'*B2');
lmiterm([1 1 1 0],(dM/Dr12)*Q_old);
lmiterm([1 1 1 Q2],-dM/Dr12,1);
lmiterm([1 2 1 0],B1');
```

212
lmiterm([1 2 2 0],-gamma_old_2);
lmiterm([1 3 1 0],C*Q_old);
lmiterm([1 3 1 0],D*F_old);
lmiterm([1 3 2 0],0);
lmiterm([1 3 3 0],-1);

%Reachable set LMI with negative $\dot{P}$
%NOT NECESSARY

%Reachable set LMI with positive $\dot{P}$
lmiterm([2 1 1 0],A*Q_old+Q_old*A');
lmiterm([2 1 1 0],B2*F_old+F_old'*B2');
lmiterm([2 1 1 0],alpha*Q_old);
lmiterm([2 1 1 0],(dM/Dr12)*Q_old); %
lmiterm([2 1 1 Q2],-dM/Dr12,1); %
lmiterm([2 2 1 0],B1');
lmiterm([2 2 2 0],-alpha);

% Q2 system % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Saturation LMI
lmiterm([-3 1 1 Q2],1,1);
lmiterm([-3 2 1 F2],1,1);
lmiterm([-3 2 2 0],r*delta2^2);

% Performance LMI with negative $\dot{P}$
lmiterm([4 1 1 Q2],A,1,'s');
lmiterm([4 1 1 F2],B2,1,'s');
lmiterm([4 2 1 0],B1');
lmiterm([4 2 2 gamma2_2],-1,1);
lmiterm([4 3 1 Q2],C,1);
lmiterm([4 3 1 F2],D,1);
lmiterm([4 3 2 0],0);
lmiterm([4 3 3 0],-1);

% Performance LMI with positive $\dot{P}$
lmiterm([5 1 1 Q2],A,1,'s');
lmiterm([5 1 1 F2],B2,1,'s');
lmiterm([5 1 1 Q2],dM/Dr23,1);
lmiterm([5 1 1 Q3],-dM/Dr23,1);
lmiterm([5 2 1 0],B1');
lmiterm([5 2 2 gamma2_2],-1,1);
lmiterm([5 3 1 Q2],C,1);
lmiterm([5 3 1 F2],D,1);
lmiterm([5 3 2 0],0);
lmiterm([5 3 3 0],-1);

%Reachable set LMI with negative $\dot{P}$

lmiterm([6 1 1 Q2],A,1,'s');
lmiterm([6 1 1 F2],B2,1,'s');
lmiterm([6 1 1 Q2],alpha,1);
lmiterm([6 2 1 0],B1');

lmiterm([6 2 2 0],-alpha);

%Reachable set LMI with positive $\dot{P}$

lmiterm([7 1 1 Q2],A,1,'s');

% Q3 system

% For this case we don't have "extra terms" since we assume Q4=Q3

% Saturation LMI

lmiterm([-9 1 1 Q3],1,1);

% Performance LMI

lmiterm([10 1 1 Q3],A,1,'s');

% Reachable set LMI
\begin{verbatim}
lmiterm([12 1 1 Q3],1,1);
lmiterm([12 1 1 Q2],-1,1);

%%%
lmiterm([-13 1 1 0],gamma_old_2);
lmiterm([-13 1 1 gamma2_2],-1,1);
lmiterm([-14 1 1 gamma2_2],1,1);
lmiterm([-14 1 1 gamma3_2],-1,1);
case2=getlmis;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=decnbr(case2);
c=zeros(1,n-3);
c=[c 0 cte2 cte3];
options=[1e-6 1e3 1e9 10 0];
[copt,xopt]=mincx(case2,c,options);
gamma2=sqrt(dec2mat(case2,xopt,gamma2_2));
gamma3=sqrt(dec2mat(case2,xopt,gamma3_2));
gamma=[sqrt(gamma_old_2) gamma2 gamma3];
F2_var=dec2mat(case2,xopt,F2);
F3_var=dec2mat(case2,xopt,F3);
F=[F_old F2' F3'];
Q2=dec2mat(case2,xopt,Q2);
Q3=dec2mat(case2,xopt,Q3);
Q=[Q_old Q2 Q3];
P=[inv(Q_old) inv(Q2) inv(Q3)];
k2=(F2_var*inv(Q2));
k3=(F3_var*inv(Q3));
k=[K_old k2' k3'];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Function for the switch to the pure sequential scheme

function [Qi,Pi,ki,gamma]=inner_ellipsoids(delta,i,r,Q_old,...
        ...gamma_old_2,K_old,A,B1,B2,C,D,alpha,dM)

delta1=delta(i(1));
delta2=delta(i(2));
Dr12=r*(delta2^2-delta1^2);
F_old=K_old'*Q_old;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

setlmis([ ]); Q2=lmivar(1,[2 1]); F2=lmivar(2,[1 2]); gamma2_2=lmivar(1,[1 1]);
\end{verbatim}
% Q1 system

% Saturation LMI
% NON NECESSARY

% Performance LMI with negative $\dot{P}$
% NON NECESSARY

% Performance LMI with positive $\dot{P}$

lmiterm([1 1 1 0], A*Q_old + Q_old*A');
lmiterm([1 1 1 0], B2*F_old + F_old'*B2');
lmiterm([1 1 1 0], (dM/Dr12)*Q_old);
lmiterm([1 1 1 Q2], -dM/Dr12, 1);
lmiterm([1 2 1 0], B1');
lmiterm([1 2 2 0], -gamma_old_2);
lmiterm([1 3 1 0], C*Q_old);
lmiterm([1 3 1 0], D*F_old);
lmiterm([1 3 2 0], 0);
lmiterm([1 3 3 0], -1);

% Reachable set LMI with negative $\dot{P}$
% NON NECESSARY

% Reachable set LMI with positive $\dot{P}$

lmiterm([2 1 1 0], A*Q_old + Q_old*A');
lmiterm([2 1 1 0], B2*F_old + F_old'*B2');
lmiterm([2 1 1 0], alpha*Q_old);
lmiterm([2 1 1 0], (dM/Dr12)*Q_old);  %
lmiterm([2 1 1 Q2], -dM/Dr12, 1);  %
lmiterm([2 2 1 0], B1');
lmiterm([2 2 2 0], -alpha);

% Q3 system

% For this case we don't have "extra terms" since we assume Q4=Q3

% Saturation LMI

lmiterm([-3 1 1 Q2], 1, 1);
lmiterm([-3 2 1 F2], 1, 1);
lmiterm([-3 2 2 0], r*delta2^2);

% Performance LMI

lmiterm([4 1 1 Q2], A, 1, 's');
lmiterm([4 1 1 F2], B2, 1, 's');
lmiterm([4 2 1 0], B1');
lmiterm([4 2 2 gamma2_2],-1,1);
lmiterm([4 3 1 Q2],C,1);
lmiterm([4 3 1 F2],D,1);
lmiterm([4 3 2 0],0);
lmiterm([4 3 3 0],-1);

lmiterm([5 1 1 Q2],A,1,'s');
lmiterm([5 1 1 F2],B2,1,'s');
lmiterm([5 1 1 Q2],alpha,1);
lmiterm([5 2 1 0],B1');
lmiterm([5 2 2 0],-alpha);

lmiterm([6 1 1 Q2],1,1);
lmiterm([6 1 1 0],-Q_old);

lmiterm([-7 1 1 gamma2_2],1,1);
lmiterm([-7 1 1 0],-gamma_old_2);
case2=getlmis;

n=decnbr(case2);
c=zeros(1,n-1);
c=[c 1];
options=[1e-7 1e3 1e9 10 0];
[copt,xopt]=mincx(case2,c,options);
gammad=dec2mat(case2,xopt,gamma2_2);
Qk=dec2mat(case2,xopt,Q2);
Fk=dec2mat(case2,xopt,F2);
Pk=inv(Qk);
kk=(Fk*Pk)';
Qi=[Q_old Qk];
ki=[K_old kk];
Pi=[inv(Q_old) Pk];
gammai=[sqrt(gamma_old_2) gammak];
A.8 Time simulation code for $P = constant$

This time simulation is the only that does not require the implementation of Matlab codes through s-function Simulink boxes. Therefore, all the programming is done with standard Simulink boxes, which are described on Chapter 9.1 and 9.2.1.
A.9 State Space Scheduling, for $\dot{r} \leq 0$

$m$ function $c$ controller $- a$ actuator.m

function [sys,x0,str,ts] = ms_function_controller(t,x,u,flag)

% previous step: initialization of our variables

[K,r,ellip_border,Q,ul]=initializacion_simulacion;

switch flag,

%%%%%%%%%%%%%%%%%%%
% Initialization %
%%%%%%%%%%%%%%%%%%%
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(r,K);
    sys;

%%%%%%%%%%%%%%%%%%%
% Updating states %
%%%%%%%%%%%%%%%%%%%
case 2,
    sys=mdlUpdate(t,x,u,r,K,Q,ellip_border,ul);
    sys;

%%%%%%%%%%%%%%%%%
% Outputs %
%%%%%%%%%%%%%%%%%
case 3,
    sys=mdlOutputs(t,x,u);

%%%%%%%%%%%%%%%%%
% Others %
%%%%%%%%%%%%%%%%%
case {1, 4, 9}
    sys=[ ];
    %Unused flags

otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end;

219
%mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the
% S-function
function [sys,x0,str,ts]=mdlInitializeSizes(r,K)
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 3; %"r" and "K"
sizes.NumOutputs = 2; %K=[k1 k2]
sizes.NumInputs = 2; %state x=[x1 x2]
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1; % at least one sample time is needed
sys = simsizes(sizes);

% initialize the initial conditions
%
x0 = [r(length(r)) (K(:,length(r)))'; %value for r at the inner ellipsoid
str = [ ]; %No state ordering

% initialize the array of sample times
%
nts = [-1 0]; %Inherited sample time
% end mdlInitializeSizes

%mdlUpdate
% Calculate next value for the state.
function sys=mdlUpdate(t,x,u,r,K,Q,ellip_border,ul)
sys=update_r(t,x,u,r,K,Q,ellip_border,ul); sys;
% end mdlUpdate

%mdlOutputs
% Return the block outputs.
function sys=mdlOutputs(t,x,u)
sys=x(2:3)'; sys;
% end mdlOutputs
function [K,r,ellip_border,Q,ul]=inicializacion_simulacion

% On the Workspace of Matlab, the initial state vector has to be:
% xInitial=[0 0 10000 -2007.82212092985 -181.956068804005]
% xInitial=[x1 x2 r K1 K2]

delta=[1 1.5 2.5 7.5 10 15 20 50 100];
ul=1000; wm=1000;


r=[ ]; ellip_border=[ ];

for i=1:1:length(delta)
    r(i)=delta(i)^2*(ul^2)/(wm^2);
ellip_border(i)=(ul^2)/r(i);
end;
j=1; for i=1:1:length(delta)
    Q(:,i)=Qsol(:,j:j+1);
j=j+2;
end;
function sys=update_r(t,state_old,x,r,K,Q,ellip_border,ul)

jump=0;
state_old(1);
K_old=state_old(2:3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% first focus on the interval of r_n
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
found=0;
interval=[1,2]; %We know it is inside ellipsoid 1
i=2;

while found==0 && i<=length(r)
    p=x'*inv(Q(:,:,i))*x;
    if p<ellip_border(i) %We look if it is also inside ellipsoid 2
        interval=[i,i+1];
        i=i+1;
    else
        found=1; %We're not in the next ellipsoid, finish the search
    end;
end;
if interval(1)==length(r) %if we are inside the smallest ellipsoid
    interval(2)=length(r); %we have to change some results
end;
nn=interval(2); n=interval(1); rnn=r(nn); rn=r(n);

%Now we have the interval where we are right now. Each interval has a
%value for "r" associated, but we can't accept a value for that point
%smaller than what we got on the previous step. Therefore, we check
%whether this interval has smaller "r" associated than our minimum
%acceptable. If that is the case, we impose the minimum value and
%skip any further calculation regarding r.
if rn>r_old
    jump=1;
    r_new=r_old;
    K_new=K_old;
end;
if jump==0
error=0.01; finish=0;
Qn=Q(:,:,n); Qnn=Q(:,:,nn);
Kn=K(:,n); Knn=K(:,nn);
i=1;
if rnn ~= rn %if we are not at the inner ellipsoid
    r_petit=rn;
    r_gran=rnn;
    while i<100 && rnn~=rn
        r_new=(r_petit+r_gran)/2;
        lambda=(r_new-rn)/(rnn-rn);
        Q_new=Qn+(Qnn-Qn)*lambda; %Do we also have to "round" this?
        dif=((ul^2)/(r_new))-(x'*inv(Q_new)*x);
        if dif>0 %If the point is in the proposed spline ellipsoid
            r_petit=r_new;
        else %If the point is outside the proposed ellipsoid
            r_gran=r_new;
        end;
        i=i+1;
    end;
    r_new=r_petit; %we take the biggest ellipsoid, since it’s safer
    if r_new>r_old
        r_new=r_old;
        K_new=K_old;
    else
        lambda=(r_new-rn)/(rnn-rn);
        K_new=Kn+(Knn-Kn)*lambda; %Do we also have to "round" this?
    end;
else %When we are on the inner ellipsoid
    lambda=0;
    r_new=rn; K_new=Kn;
end;
K_new=K_new';
sys=[r_new K_new];
A.10 Actuator Input Scheduling, for $\dot{r} \leq 0$.

This case is essentially the same as "State Space Scheduling, for $\dot{P} \leq 0$" described on section A.9. The only difference is on the Scheduling part carried out on function "update-r.m".

_The same code as in A.9_

_initialization – simulation.m_

Here the only difference with A.9 are the values for Q and K, which are different since the LMI system is different, and the rest of values calculated from them.

_update – r.m_

```matlab
function sys=update_r(t,state_old,x,r,K,Q,ellip_border,ul)
jump=0; r_old=state_old(1); K_old=state_old(2:3);

% First we focus on the interval of r_n
found=0; i=length(r); security=0.86;
while found==0 && i>=1
    u_t=K(:,i)’*x;
    if u_t<=ul*security
        interval=[i,i+1]; found=1;
    else
        i=i-1;
    end;
end;
if interval(2)>length(r) %if we are inside the smallest ellipsoid
    interval(2)=length(r); %we have to change some results
end;
nn=interval(2); n=interval(1); rnn=r(nn); rn=r(n);
```

224
% Now we have the current interval. Each interval has a value for "r"
% associated, but we can’t accept a value for that point smaller than what
% we got on the previous step. Therefore, we check whether this interval
% has smaller "r" associated than our minimum acceptable. If that is the case,
% we impose the minimum value and skip any further calculation regarding r.

if rn>r_old
    jump=1;  r_new=r_old;  K_new=K_old;
end;

% Now, if nothing weird happens, we can calculate accurately the value of r

if jump==0
    error=0.01;  finish=0;  Qn=Q(:,:,n);  Qnn=Q(:,:,nn);  Kn=K(:,n);  Knn=K(:,nn);
i=1;
if rnn ~= rn  %if we are not at the inner ellipsoid
    r_petit=rn;
    r_gran=rnn;
    while i<100 & & rnn ~= rn
        r_new=(r_petit+r_gran)/2;
        lambda=(r_new-rn)/(rnn-rn);
        K_new=Kn+(Knn-Kn)*lambda;
        dif=ul*security-K_new'*x;
        if dif>0  %If the point is in the proposed spline ellipsoid
            r_petit=r_new;
        else  %If the point is outside the proposed ellipsoid
            r_gran=r_new;
        end;
        i=i+1;
    end;
    r_new=r_petit;  %we take the smallest controller, since it’s safer
    if r_new>r_old
        r_new=r_old;  K_new=K_old;
    else
        lambda=(r_new-rn)/(rnn-rn);
        K_new=Kn+(Knn-Kn)*lambda;
        Q_new=Qn+(Qnn-Qn)*lambda;
    end;
else  %When we are on the inner ellipsoid
    lambda=0;
    r_new=rn;  K_new=Kn;
end;
end;
K_new=K_new’;  sys=[r_new  K_new];
A.11 Actuator Input Scheduling, for $-\infty < \dot{r} \leq |d_M|$. 

`ms - function - controller - actuator.m`

```matlab
function [sys,x0,str,ts] = ms_function_controller(t,x,u,flag)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% previous step: initialization of our variables

[K,r,ellip_border,Q,ul,t_old,dM]=inicializacion_simulacion;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch flag,
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Initialization %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes(r,t_old,K);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Updating states %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 2,
        sys=mdlUpdate(t,x,u,r,K,Q,ellip_border,ul,dM);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Outputs %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 3,
        sys=mdlOutputs(t,x,u);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % The rest %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case {1, 4, 9}
        sys=[]; %Unused flags
    otherwise
        error(['Unhandled flag = ',num2str(flag)]);
end;
```

226
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the
% S-function
%===================================================================

function [sys,x0,str,ts]=mdlInitializeSizes(r,t_old,K)

sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 4; % "r","t_old" and "K"
sizes.NumOutputs = 2;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 1; % the output depends directly on the input
sizes.NumSampleTimes = 1; % at least one sample time is needed
sys = simsizes(sizes);

% initialize the initial conditions
%
x0 = [r(length(r)) t_old (K(:,length(r)))
]%value for r at the inner ellipsoid
str = [ ]; %No state ordering

% initialize the array of sample times
%
ts = [-1 0]; %Inherited sample time
% end mdlInitializeSizes

% mdlUpdate
% Calculate next value for the state.
%===================================================================

function sys=mdlUpdate(t,x,u,r,K,Q,ellip_border,ul,dM)
sys=update_r(t,x,u,r,K,Q,ellip_border,ul,dM);
%end mdlUpdate

% mdlOutputs
% Return the block outputs.
%===================================================================

function sys=mdlOutputs(t,x,u) sys=x(3:4)
%sys=x(2:3)

% end mdlOutputs
This subfunction is basically the same as we had on A.9. The only difference again, are the values for the different $Q_n$ and $K_n$, as well as all the values calculated from them (since they come from different LMI systems).

In addition to that, we have the addition of the 2 new parameters described on 9.2.4

t_old=-10000; %Initial value to avoid problems. Later on it will be updated with the last time considered for the box
dM=2500;

function sys=update_r(t,state_old,x,r,K,Q,ellip_border,ul,dM)

%notice that we get the previous state of the system, represented by r_old because of our constraints, our new state (r_new) has to be smaller or equal
r_old=state_old(1) t_old=state_old(2) K_old=state_old(3:4) x

% first focus on the interval of r_n
found=0;
i=length(r);
security=0.85; %security against oversaturation when we switch controller
while found==0 && i>=1
    u_t=K(:,i)'*x;
    if u_t<=ul*security
        interval=[i,i+1];
        found=1;
    else
        i=i-1;
    end;
end;
if interval(2)>length(r) %if we are inside the smallest ellipsoid
    interval(2)=length(r); %we have to change some results
end;
nn=interval(2) 
n=interval(1)
rnn=r(nn);
rn=r(n);
if n ~= 10
   caca=0;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, if anything weird happens, we can calculate accurately the
% value of r
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
error=0.01;
finish=0;
Qn=Q(:,:,n); Qnn=Q(:,:,nn);
Kn=K(:,n);Knn=K(:,nn);

% Starting point
i=1;
if rnn ~= rn %if we are not at the inner ellipsoid
   r_petit=rn;
   r_gran=rnn;
   while i<100 && rnn ~= rn
      r_new=(r_petit+r_gran)/2;
      lambda=(r_new-rn)/(rnn-rn);
      K_new=Kn+(Knn-Kn)*lambda;
      dif=ul*security-K_new'*x;
      if dif>0 %If the point is in the proposed spline ellipsoid
         r_petit=r_new;
      else %If the point is outside the proposed ellipsoid
         r_gran=r_new;
      end;
      i=i+1;
   end;
   r_new=r_petit; %At the end, we take the smallest controller, %since it’s safer
else %When we are on the inner ellipsoid
   r_new=rn;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%So far we have "r_new" and rn rnn Kn Knn Qn Qnn
%Now, we check the current value of dr/dt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

r_new dr=(r_new-r_old)/(t-t_old)
if dr>dM
    
    r_new=dM*(t-t_old)+r_old

    %now we look to what interval corresponds this "forced" r_new
    check=0;
    counter=1;
    if r_new>r(length(r))
        r_new=r(length(r));
        lambda=0;
        K_new=Kn;
        check=1;
    end;
end;
while check==0 && counter<length(r)
    
    if r(counter+1)>r_new
        check=1;
        rn=r(counter);
        rnn=r(counter+1);
        Qn=Q(:,:,counter);
        Qnn=Q(:,:,counter+1);
        Kn=K(:,counter);
        Knn=K(:,counter+1);
    end;
    counter=counter+1;
end;

r_new if rnn == rn
    lambda=0;
else
    lambda=(r_new-rn)/(rnn-rn);
end;

K_new=Kn+(Knn-Kn)*lambda; Q_new=Qn+(Qnn-Qn)*lambda;

%Since we have finished, we store this time for
%the next iteration

t_old=t;
sys=[r_new t_old K_new'];

230