Decentralized and Hierarchical Model
Predictive Control of Networked Systems

Davide Barcelli

Ph.D. Thesis in Information Engineering

Supervisor
Prof. Alessandro Agnetis

Co-supervisor
Prof. Alberto Bemporad

Siena
October 11, 2012
Acknowledgements

I owe my gratitude to Prof. Alberto Bemporad who supported me since my master degree thesis until the conclusion of my graduate studies with constant guidance and supervision. He introduced me to all the people I had the pleasure to work with and taught me how to “research”, for this, I will be always grateful. I want to thank Prof. Alessandro Agnetis for his presence that never lacked when needed.

For financial support I acknowledge the European Union and in particular the WIDE and HYCON2 projects for the four annual fellowships that funded my studies.

During this period I had the opportunity to work with many great European researchers who participated in the aforementioned projects. Above all, I thank Prof. Vicenç Puig for supporting me while I was visiting at Universitat Politècnica de Catalunya, working on my master thesis and, later on, during the development of WIDE project.

This Thesis summarizes the results achieved with the help of a few colleagues and friends who walked with me during these years. Because of that I feel obliged to thank Daniele Bernardini, Giulio Ripaccioli and Matteo Rubagotti for helping me with their guidance and support.

I enjoyed the time in Siena with Alessandro, Sara, Rudy and Sergio who formed with me a group of people that I will hardly forget. Moreover, I want to thank people from University of Trento and IMT Lucca, Carlo Alberto, Laura and Panagiotis.

Finally my gratitude goes to my parents, Rino and Piera, who constantly helped and supported me. I owe to them the possibility of dedicate myself full-time to my research.

A special thank to someone who make her entrance in my life in a very dark period, and turned it into light, Camilla.
## Contents

1 Introduction ........................................................................... 3
  1.1 Limitations of centralized control ................................. 3
  1.2 Model predictive control .............................................. 6
  1.3 Existing approaches to DMPC ....................................... 8
    1.3.1 DMPC approach of Jia and Krogh ......................... 9
    1.3.2 DMPC approach of Venkat, Rawlings, and Wright ...... 10
    1.3.3 DMPC approach of Dunbar and Murray ................. 11
    1.3.4 DMPC approach of Keveczy, Borrelli, and Balas ....... 12
    1.3.5 DMPC approach of Mercangöz and Doyle .............. 14
    1.3.6 DMPC approach of Magni and Scattolini .............. 14
  1.4 Other non-centralized approaches ................................. 14
  1.5 Thesis structure and contribution .................................. 17

2 Decentralized networked linear regulator .......................... 21
  2.1 Introduction .................................................................... 21
  2.2 Control over ideal networks ......................................... 23
    2.2.1 Linear controller synthesis .................................. 24
  2.3 Control over lossy networks .......................................... 28
    2.3.1 Switching controller synthesis .............................. 29
  2.4 Stochastic control under packet dropout ....................... 32
    2.4.1 Stochastic network model ................................. 33
    2.4.2 Stochastic switching controller synthesis .......... 34
  2.5 Simulation results .......................................................... 37


2.6 Conclusions .................................................. 40

3 Decentralized Networked Model Predictive Control 41
3.1 Introduction ................................................. 41
3.2 Decentralized model predictive control setup .......... 45
  3.2.1 Decentralized prediction models .................. 47
  3.2.2 Decentralized optimal control problems .......... 48
3.3 Convergence properties .................................. 50
  3.3.1 Stability tests .................................. 53
  3.3.2 Open-loop unstable subsystems .................. 53
  3.3.3 Decentralized MPC under arbitrary packet loss .. 55
  3.3.4 Extension to set-point tracking ................. 59
3.4 Decentralized temperature control in a railcar .......... 60
  3.4.1 Simulation results ................................ 63
3.5 Conclusions ............................................. 68

4 Decentralized Hierarchical Multi-rate Control 69
4.1 Introduction .............................................. 69
4.2 Centralized Hierarchical Approach ....................... 71
  4.2.1 Problem Setup .................................. 71
  4.2.2 Computation of maximum reference rates ........ 74
  4.2.3 Centralized Hierarchical Controller ............... 75
4.3 Decentralized Hierarchical Approach ..................... 77
  4.3.1 Problem setup .................................. 77
  4.3.2 Reference rate constraints ....................... 81
  4.3.3 Decentralized Hierarchical Controller ............. 83
4.4 Simulation Example ....................................... 84
  4.4.1 System description ................................ 84
  4.4.2 Controller Choice ................................ 85
  4.4.3 Simulation results ................................ 87
4.5 Conclusions ............................................. 88

5 Hierarchical DMPC: The Barcelona Drinking Water Network 89
5.1 Introduction .............................................. 89
5.2 Case-study Description ................................. 91
## Contents

5.2.1 System description ........................................ 91  
5.2.2 System management criteria ................................ 92  
5.2.3 Control-oriented modelling ................................ 94  
5.3 DWN Partitioning Approach .................................. 98  
5.3.1 Partitioning algorithm ...................................... 99  
5.3.2 Case study partitioning ..................................... 104  
5.4 DMPC Approach ................................................. 104  
5.5 Results ............................................................ 108  
5.6 Conclusions ....................................................... 112  

6 Conclusions and Outlook ........................................ 115  
6.1 Conclusions ....................................................... 115  
6.2 Outlook ........................................................... 116  

A WIDE Toolbox ..................................................... 119  
A.1 Toolbox description ........................................... 119  
A.2 Decentralized Linear Control (decLMI) ........................ 122  
A.2.1 Class description ............................................ 123  
A.2.2 Methods ....................................................... 123  
A.2.3 Example ....................................................... 125  
A.3 Decentralized MPC (dlincon) .................................. 137  
A.3.1 Class description ............................................ 137  
A.3.2 Methods ....................................................... 138  
A.3.3 Example ....................................................... 140  
A.4 Hierarchical MPC (HiMPC) ..................................... 148  
A.4.1 Class description ............................................ 148  
A.4.2 Methods ....................................................... 150  
A.4.3 Example ....................................................... 151  
A.5 NCS simulation with TrueTime (ACG) ......................... 163  
A.5.1 Class description ............................................ 163  
A.5.2 Methods ....................................................... 164  
A.5.3 Example ....................................................... 164
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Hierarchical and decentralized/distributed model predictive control of a large-scale process</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>Example of an ideal network topology with shared sensors $s_3$, $s_4$ and $s_5$</td>
<td>24</td>
</tr>
<tr>
<td>2.2</td>
<td>Example of a network topology scheme with ideal links (solid lines) and links subject to packet drop (dashed lines)</td>
<td>29</td>
</tr>
<tr>
<td>2.3</td>
<td>2-state Markov chain modeling the network status</td>
<td>33</td>
</tr>
<tr>
<td>2.4</td>
<td>Network topology used in simulations</td>
<td>38</td>
</tr>
<tr>
<td>2.5</td>
<td>Total (a) state and (b) input trajectories for robust (dashed line) and stochastic (solid line) decentralized controllers</td>
<td>39</td>
</tr>
<tr>
<td>3.1</td>
<td>Example of decomposition of a global model into four submodels. Each colored rectangle identifies the states belonging to the corresponding submodel</td>
<td>46</td>
</tr>
<tr>
<td>3.2</td>
<td>Physical structure of the railcar</td>
<td>61</td>
</tr>
<tr>
<td>3.3</td>
<td>Exogenous signals used in the reported simulations: external temperature and user defined reference temperatures on the left and right, respectively.</td>
<td>63</td>
</tr>
<tr>
<td>3.4</td>
<td>Comparison between centralized MPC (dashed lines) and decentralized MPC (continuous lines): output $h_1$ (upper plots) and input $v_1$ (lower plots). Gray areas denote packet drop intervals.</td>
<td>64</td>
</tr>
</tbody>
</table>
### List of Figures

3.5 Decentralized MPC results. Upper plots: output variables $h$ (continuous lines) and references $r$ (dashed lines). Lower plots: command inputs $v$. Gray areas denote packet drop intervals.

3.6 Markov chain modeling packet-loss probability: the network is in state $i$ if the last $i-1$ packets have been lost.

3.7 Markov chain packet-loss probability with $N = 10$ and $p = 0.7$.

3.8 Performance indices of Centralized MPC (dashed line) and Decentralized MPC (solid line).

4.1 General two layers hierarchical control scheme.

4.2 Decentralized hierarchical control scheme.

4.3 Dynamical process used in the simulation example.

4.4 Plots of $\Delta r_i(N_i)$ as a function of $N_i$ obtained by solving (4.12) and (4.28), respectively: subsystems $\Sigma_1$ (continuous blue line), $\Sigma_2$ to $\Sigma_4$ (black dashed), and $\Sigma_5$ (red dotted with circles), and centralized approach (magenta dotted with triangles).

4.5 Masses #1 (continuous) and #2 (dashed) trajectories with controllers HiMPC (blue), DHiMPC (red) and Constr (magenta). Green, yellow and red background areas depict admissible references, constraint tighten $\Delta K$ and inadmissible zone. Black bold bordered squares indicate where constraint violation occurs.

5.1 Hierarchical structure for transport network system.

5.2 Aggregate case of the Barcelona Drinking Water Network.

5.3 Conceptual scheme of the partitioned Barcelona DWN.

5.4 Hierarchy of MPC controllers $C_i$. Their solution sequence is top-down.

5.5 Cost function evaluation for CMPC and DMPC strategies.

5.6 Computed flow related to valve 54 (VBMC).

5.7 Resultant volume related to tank 10 (d130BAR).

A.1 decLMI example network topology.

A.2 Open-loop (blue stars) and closed-loop (red circles) eigenvalues of the plant with centralized ideal feedback.
List of Figures

A.3 Open-loop (blue stars) and closed-loop (red circles) eigenvalues of the plant with decentralized ideal feedback .......................... 130

A.4 Open-loop (blue stars) and closed-loop (red circles) eigenvalues of the plant with robust decentralized feedback .......................... 132

A.5 Open-loop (blue stars) and closed-loop (red circles) eigenvalues of the plant with stochastic decentralized feedback .......................... 133

A.6 State (upper plot) and input (lower plot) of closed-loop trajectories, where 1st, 2nd and 3rd components are denoted with blue, green and red color, respectively. Centralized MPC trajectories are depicted in continuous line, decentralized MPC in dashed and references in starred-continuous. Regulation problem 1 with fixed bounds .................................................. 145

A.7 State (upper plot) and input (lower plot) of closed-loop trajectories, where 1st, 2nd and 3rd components are denoted with blue, green and red color, respectively. Centralized MPC trajectories are depicted in continuous line, decentralized MPC in dashed and references in starred-continuous. Tracking problem 1 with fixed bounds .................................................. 146

A.8 State (upper plot) and input (lower plot) of closed-loop trajectories, where 1st, 2nd and 3rd components are denoted with blue, green and red color, respectively. Centralized MPC trajectories are depicted in continuous line, decentralized MPC in dashed and references in starred-continuous. Shifted-tracking problem 1 with fixed bounds .................................................. 147

A.9 State (upper plot) and input (lower plot) of closed-loop trajectories, where 1st, 2nd and 3rd components are denoted with blue, green and red color, respectively. Centralized MPC trajectories are depicted in continuous line, decentralized MPC in dashed and references in starred-continuous. Regulation problem 2 with fixed bounds .................................................. 148
List of Figures

A.10 State (upper plot) and input (lower plot) of closed-loop trajectories, where 1st, 2nd and 3rd components are denoted with blue, green and red color, respectively. Centralized MPC trajectories are depicted in continuous line, decentralized MPC in dashed and references in starred-continuous. Tracking problem 2 with fixed bounds. ........................................ 149

A.11 State (upper plot) and input (lower plot) of closed-loop trajectories, where 1st, 2nd and 3rd components are denoted with blue, green and red color, respectively. Centralized MPC trajectories are depicted in continuous line, decentralized MPC in dashed and references in starred-continuous. Shifted-tracking problem 2 with fixed bounds. ........................................ 150

A.12 $\Delta r$ as function of $N$ for HiMPC1 (blue line) and HiMPC2 (black line) ..................................................... 159

A.13 $\Delta r/N$ as function of $N$ for HiMPC1 (blue line) and HiMPC2 (black line) ..................................................... 160

A.14 Output trajectories evolution of the two masses in closed-loop, depicted in continuous (first mass) and dashed (second mass) line, respectively. Red line is user defined reference, black is HiMPC1 and blue is HiMPC2. ..................................................... 162

A.15 Input reference that HiMPCs assigned to 1st and 2nd mass controller’s, depicted in continuous and dashed line, respectively. Black and blue color indicates HiMPC1 and HiMPC2, respectively. ..................................................... 163

A.16 Simulink model generated by ACG. ........................................ 165

A.17 Closed-loop trajectories with network antenna transmit power at $-30dB$. ..................................................... 167

A.18 Closed-loop trajectories with network antenna transmit power at $-46dB$. ..................................................... 168

B.1 Comparison between current (a) and MOBY-DIC-based (b) design vision for embedded control systems. ........................................ 170
B.2 Examples of two-dimensional domain partitions: generic (a), simplicial single-resolution (b), hyperrectangular single- (c) and multi-resolution (d). The corresponding classes of PWA functions are: PWAG (a), PWAS (b), sPWAR (c), and mPWAR (d). ................................................................. 175
B.3 Structure of the controller design flow. ................................. 178
B.4 Estimate of the domain of attraction of closed-loop system with PWAG controller. ......................................................... 184
B.5 Exact PWAG (a) and approximate PWAS(b) and PWAR(c) control laws. ................................................................. 185
B.6 Simulink simulations of the closed-loop system (plant and PWAS controller). The top panel shows the evolution of the states, the bottom panel the evolution of the input computed with VHDL simulation. ......................................................... 187

C.1 Graphical representation of the sets (from the outer to the inner) $D'_N$, $D_N$, $X_f$, $S_f$, $\mathcal{R}_\infty$. ........................................... 203
C.2 Control function $u(x)$ on the simplicial partition of the set $S$. 204
C.3 Time evolution of state and control variables in the simulation example. The solid lines represent the variables $(x_1(t), x_2(t), u(t))$ that would result by applying the optimal control law $u^*(t)$, while the dashed lines represent the same variables obtained by applying the PWAS control law represented in Fig. C.2. . . 205
Chapter 1

Introduction

1.1 Limitations of centralized control

Most of the procedures for analyzing and controlling dynamical systems developed over the last decades rest on the common presupposition of centrality. Centrality means that all the information available about the system is collected at a single location, where all the calculations based on such information are executed. Information includes both \textit{a priori} information about the dynamical model of the system available off-line, and \textit{a posteriori} information about the system response gathered by different sensors on-line.

When considering large-scale systems the presupposition of centrality fails because of the lack of a centralized information-gathering system and/or of centralized computing capabilities. Typical examples of such systems are power networks, water networks, urban traffic networks, cooperating vehicles, digital cellular networks, flexible manufacturing networks, supply chains, complex structures in civil engineering, and many others. In such systems the centrality assumption often fails because of geographical separation of components (spatial distribution) which implies costs for guaranteeing reliability often unaffordable. Moreover, technological advances and reduced cost of small size microprocessors provide a new force for distributed computation. Hence the current trend for decentralized decision making, distributed computations, and hierarchical control.

Several new challenges arise when addressing a decentralized setting, where most of the existing analysis and control design methodologies cannot be directly applied. In a distributed control system which employs decentralized control techniques there are several local control stations, where each controller observes only local outputs and only controls local inputs. Besides advantages in controller implementation (namely reduced and parallel com-
putations, reduced communications), a great advantage of decentralization is maintenance: while certain parts of the overall process are interrupted, the remaining parts keep operating (possibly with reduced performance) in closed-loop with their local controllers, without the need of stopping the overall process as in case of centralized control. Moreover, a partial re-design of the process does not necessarily imply a complete re-design of the controller, as it would instead in case of centralized control. However, all the controllers are involved in controlling the same large-scale process, and is therefore of paramount importance to determine conditions under which there exists a set of appropriate local feedback control laws capable of stabilizing the entire system.

Ideas for decentralizing and hierarchically organizing the control actions in industrial automation systems date back to the 70's [25,73,74,98,114], but were mainly limited to the analysis of stability of decentralized linear control of interconnected subsystems, so the interest faded. Since the late 90's, because of the advances in computation techniques like convex optimization, the interest in decentralized control raised again [33,93], and convex formulations were developed, although limited to special classes of systems such as spatially invariant systems [3]. Decentralized control and estimation schemes based on distributed convex optimization ideas have been proposed recently in [53,97] based on Lagrangean relaxations. Here global solutions can be achieved after iterating a series of local computations and inter-agent communications.

Large-scale multi-variable control problems, such as those arising in the process industries, are often dealt with model predictive control (MPC) techniques. In MPC the control problem is formulated as an optimization one, where many different (and possibly conflicting) goals are easily formalized and state and control constraints can be included. Many results are nowadays available concerning stability and robustness of MPC, see e.g. [70]. However, centralized MPC is often unsuitable for control of large-scale networked systems, mainly due to lack of scalability and to maintenance issues of global models. In view of the above considerations, it is then natural to look for decentralized or for distributed MPC (DMPC) algorithms, in which the original large-size optimization problem is replaced by a number of smaller and easily tractable ones that work together in a possibly iterative and/or cooperative
1.1. Limitations of centralized control

manner towards achieving a common, system-wide, control objective.

Even though there is not a universal agreement on the distinction between “decentralized” and “distributed”, the main difference between the two terms depends on the type of information exchange among controllers:

- **decentralized MPC**: Control agents take control decisions independently on each other. Information exchange (such as measurements and previous control decisions) is only allowed before and after the decision making process. There is no negotiation between agents during the decision process. The time needed to decide the control action is not affected by communication issues, such as network delays and loss of packets, thus easing control computation within the time deadline;

- **distributed MPC**: An exchange of candidate control decisions may also happen during the decision making process, and iterated until an agreement is reached among the different local controllers, in accordance with a given stopping criterion, allowing for a potentially better performance.

In DMPC $M$ subproblems are solved, each one assigned to a different control agent, instead of a single centralized problem. Usually the plant model is given as an atomic block of finite difference equations, thus a set of submodel is to be obtained through a process called “decomposition”. The goal of the decomposition is twofold: first, each subproblem is much smaller than the overall problem (that is, each subproblem has far fewer decision variables and constraints than the centralized one), and second, each subproblem is coupled to only a few other subproblems (that is, it shares variables with only a limited number other subproblems). Although decentralizing the MPC problem may lead to a deterioration of the overall closed-loop performance because of the suboptimality of the resulting control actions, besides computation and communication benefits there are also important operational benefits in using DMPC solutions. For instance local maintenance can be carried out by only stopping the corresponding local MPC controller, while in a centralized MPC approach the whole process should be suspended.

A DMPC control layer is often interacting with a higher-level control layer in a hierarchical arrangement, as depicted in Figure 1.1. The goal of the higher layer is to possibly adjust set-points and constraint specifications to
the DMPC layer, based on a global (possibly less detailed) model of the entire system. Because of its general overview of the entire process, such a centralized decision layer allows one to reach levels of coordination and performance optimization otherwise very difficult (if not impossible) using a decentralized or distributed action.

In a typical DMPC framework the steps performed by the local controllers at each control instant are the following: (i) measure local variables and update state estimates, (ii) solve the local receding-horizon control problem, (iii) apply the control signal for the current instant, (iv) exchange information with other controllers. Along with the benefits of a decentralized design, there are some inherent issues that one must face in DMPC: ensuring the asymptotic stability of the overall system, ensure the feasibility of global constraints, quantify the loss of performance with respect to centralized MPC.

### 1.2 Model predictive control

In this section we review the basic setup of linear model predictive control. Consider the problem of regulating the discrete-time linear time-invariant system

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

(1.1)
to the origin while fulfilling the constraints
\[ u_{\text{min}} \leq u(t) \leq u_{\text{max}} \] (1.2)
at all time instants \( t \in \mathbb{Z}_{0^+} \) where \( \mathbb{Z}_{0^+} \) is the set of nonnegative integers, \( x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m \) and \( y(t) \in \mathbb{R}^p \) are the state, input, and output vectors, respectively, and the pair \((A, B)\) is stabilizable. In (1.2) the constraints should be interpreted component-wise and we assume \( u_{\text{min}} < 0 < u_{\text{max}} \).

The control objective is usually to steer the state to the origin or to an equilibrium state \( x_r \), for which the output \( y_r = Cx_r = r \) where \( r \) is a constant reference. A suitable change of coordinates reduces the second problem to the first which, therefore, we consider in the sequel.

MPC solves such a constrained regulation problem as described below. At each time \( t \), given the state vector \( x(t) \), the following finite-horizon optimal control problem

\[
V(x(t)) = \min_U \quad x_{t+N}'Px_{t+N} + \sum_{k=0}^{N-1} x_k'Qx_k + u_k'R u_k \\
\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, \ldots, N - 1 \\
y_k = Cx_k, \quad k = 0, \ldots, N \\
x_0 = x(t) \\
u_{\text{min}} \leq u_k \leq u_{\text{max}}, \quad k = 0, \ldots, N_u - 1 \\
u_k = Kx_k, \quad k = N_u, \ldots, N - 1
\] (1.3)
is solved, where \( U \triangleq \{u_0, \ldots, u_{N_u-1}\} \) is the sequence of future input moves, \( x_k \) denotes the predicted state vector at time \( t + k \), obtained by applying the input sequence \( u_0, \ldots, u_{k-1} \) to model (1.1), starting from \( x(t) \). In (1.3) \( N > 0 \) is the prediction horizon, \( N_u \leq N - 1 \) is the input horizon, \( Q = Q' \geq 0 \), \( R = R' > 0 \), \( P = P' \geq 0 \) are square weight matrices defining the performance index, and \( K \) is some terminal feedback gain. As we will discuss below, \( P, K \) are chosen in order to ensure closed-loop stability of the overall process.

Problem (1.3) can be recast as a quadratic programming (QP) problem (see e.g. [13, 70]), whose solution \( U^*(x(t)) \triangleq \{u_0^* \ldots u_{N_u-1}^*\} \) is a sequence of optimal control inputs. Only the first input
\[ u(t) = u_0^* \] (1.4)
is actually applied to system (1.1), as the optimization problem (1.3) is repeated at time $t + 1$, based on the new state $x(t + 1)$ (for this reason, the MPC strategy is often referred to as receding horizon control). The MPC algorithm (1.3)-(1.4) requires that all the $n$ components of the state vector $x(t)$ are collected in a (possibly remote) central unit, where a quadratic program with $mN_u$ decision variables needs to be solved and the solution broadcasted to the $m$ actuators. As mentioned in the introduction, such a centralized MPC approach may be inappropriate for control of large-scale systems, and it is therefore natural to look for decentralized or distributed MPC (DMPC) algorithms.

### 1.3 Existing approaches to DMPC

A few contributions have appeared in recent years in the context of DMPC, mainly motivated by applications of decentralized control of cooperating air vehicles [18, 60, 91]. Following the survey paper [11] in this Section we review some of the main contributions on DMPC, summarized in Table 1.1, that have appeared in the scientific literature. An application of some of the results surveyed in this Chapter in a problem of distributed control of power networks with comparisons among DMPC approaches is reported in [32].
1.3. Existing approaches to DMPC

In the followin, we denote by \( M \) be the number of local MPC controllers that we want to design, for example \( M = m \) in case each individual actuator is governed by its own local MPC controller.

1.3.1 DMPC approach of Jia and Krogh

In [27,48] the system under control is composed by a number of unconstrained linear discrete-time subsystems with decoupled input signals, described by the equations

\[
\begin{bmatrix}
x_1(k+1) \\
\vdots \\
x_M(k+1)
\end{bmatrix} =
\begin{bmatrix}
A_{11} & \cdots & A_{1M} \\
\vdots & \ddots & \vdots \\
A_{M1} & \cdots & A_{MM}
\end{bmatrix}
\begin{bmatrix}
x_1(k) \\
\vdots \\
x_M(k)
\end{bmatrix} +
\begin{bmatrix}
B_1 & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & B_M
\end{bmatrix}
\begin{bmatrix}
u_1(k) \\
\vdots \\
u_M(k)
\end{bmatrix}
\tag{1.5}
\]

The effect of dynamical coupling between neighboring states is modeled in prediction through a disturbance signal \( v \), for instance the prediction model used by controller \#j is

\[
x_j(k + i + i|k) = A_{jj}x_j(k + i|k) + B_j + u_j(k + i|k) + K_j v_j(k + i|k) \tag{1.6}
\]

where \( K_j = [A_{j1} \ \cdots \ A_{jj-1} \ A_{jj+1} \ \cdots \ A_{jM}] \). The information exchanged between control agents at the end of each sample step is the entire prediction of the local state vector. In particular, controller \#j receives the signal

\[
v_j(k + i|k) =
\begin{bmatrix}
x_1(k + i|k - 1) \\
\vdots \\
x_{j-1}(k + i|k - 1) \\
x_j(k + i|k - 1) \\
\vdots \\
x_M(k + i|k - 1)
\end{bmatrix}
\]

where \( i \) is the prediction time index, from the other MPC controllers at the end of the previous time step \( k - 1 \). The signal \( v_j(k + i|k) \) is used by controller
1. Introduction

$j$ at time $k$ to estimate the effect of the neighboring subsystem dynamics in (1.6).

Under certain assumptions of the model matrix $A$, closed-loop stability is proved by introducing a contractive constraint on the norm of $x_j(k+1|k)$ in each local MPC problem, which the authors prove to be a recursively feasible constraint.

The authors deal with state constraints in [49] by proposing a min-max approach, at the price of a possible conservativeness of the approach.

1.3.2 DMPC approach of Venkat, Rawlings, and Wright

In [109–111] the authors propose distributed MPC algorithm based on a process of negotiations among DMPC agents. The adopted prediction model is

\[
\begin{align*}
    x_{ii}(k+1) &= A_{ii}x_{ii}(k) + B_{ii}u_i(k) \quad \text{(local prediction model)} \\
    x_{ij}(k+1) &= A_{ij}x_{ij}(k) + B_{ij}u_j(k) \quad \text{(interaction model)} \\
    y_i(k) &= \sum_{j=1}^{M} C_{ij}x_{ij}(k)
\end{align*}
\]

The effect of the inputs of subsystem $j$ on subsystem $i$ is modeled by using an “interaction model”. All interaction models are assumed stable, and constraints on inputs are assumed decoupled (e.g., input saturation).

Starting from a multiobjective formulation, the authors distinguish between a “communication-based” control scheme, in which each controller $i$ is optimizing his own local performance index $\Phi_i$, and a “cooperation-based” control scheme, in which each controller $i$ is optimizing a weighted sum $\sum_{j=1}^{M} \alpha_j \Phi_j$ of all performance indices, $0 \leq \alpha_j \leq 1$. As performance indices depend on the decisions taken by the other controllers, at each time step $k$ a sequence of iterations is taken before computing and implementing the input vector $u(k)$. In particular, within each sampling time $k$, at every iteration $p$ the previous decisions $u_{j \neq i}^{p-1}$ are broadcast to controller $i$, in order to compute the new iterate $u_i^p$. With the communication-based approach, the authors show that if the sequence of iterations converges, it converges to a Nash equilibrium. With the cooperation-based approach, convergence to the optimal (centralized) control performance is established. In practical situations the process sampling interval may be insufficient for the computation
1.3. Existing approaches to DMPC

The time required for convergence of the iterative algorithm, with a consequent loss of performance. Nonetheless, closed-loop stability is not compromised: as it is achieved even though the convergence of the iterations is not reached. Moreover, all iterations are plantwide feasible, which naturally increases the applicability of the approach including a certain robustness to transmission faults.

1.3.3 DMPC approach of Dunbar and Murray

In [36] the authors consider the control of a special class of dynamically decoupled continuous-time nonlinear subsystems $\dot{x}_i(t) = f_i(x_i(t), u_i(t))$ where the local states of each model represent a position and a velocity signal

$$x_i(t) = \begin{bmatrix} q_i(t) \\ \dot{q}_i(t) \end{bmatrix}$$

State vectors are only coupled by a global performance objective

$$L(x, u) = \sum_{(i,j) \in E} \omega \|q_i - q_j + d_{ij}\|^2 + \omega \|q_\Sigma - q_d\|^2 + \nu \|\dot{q}\|^2 + \mu \|u\|^2 \quad (1.7)$$

under local input constraints $u_i(t) \in U, \forall i = 1, \ldots, M, \forall t \geq 0$. In (1.7) $E_0$ is the set of pair-wise neighbors, $d_{ij}$ is the desired distance between subsystems $i$ and $j$, $q_\Sigma = (q_1 + q_2 + q_3)/3$ is the average position of the leading subsystems 1,2,3, and $q_d = (q_c^1 + q_c^2 + q_c^3)/3$ the corresponding target.

The overall integrated cost (1.7) is decomposed in distributed integrated cost functions $L_i(x_i, x_{-i}, u_i) = L^x_i(x_i, x_{-i}) + \gamma \mu \|u_i\|^2 + L^d(i)$ where $x_{-i} = (x_{j1}, \ldots, x_{jk})$ collects the states of the neighbors of agent subsystem $\#i$, $L^x_i(x_i, x_{-i}) = \sum_{j \in N_i} \frac{\omega}{2} \|q_i - q_j + d_{ij}\|^2 + \gamma \nu \|\dot{q}_i\|^2$, and

$$L^d(i) = \begin{cases} \gamma \omega \|q_\Sigma - q_d\|^{2/3} & i \in \{1, 2, 3\} \\ 0 & \text{otherwise} \end{cases}$$

It holds that

$$L(x, u) = \frac{1}{\gamma} \sum_{i=1}^{N} L_i(x_i, x_{-i}, u_i)$$
Before computing DMPC actions, neighboring subsystems broadcast in a synchronous way their states, and each agent transmits and receives an “assumed” control trajectory $\hat{u}_i(\tau; t_k)$. Denoting by $u^p_i(\tau; t_k)$ the control trajectory predicted by controller #$i$, by $u^*_i(\tau; t_k)$ the optimal predicted control trajectory, by $T$ the prediction horizon, and by $\delta \in (0, T]$ the update interval, the following DMPC performance index is minimized

$$\min_{u^p_i} J_i(x_i(t_k), x_{-i}(t_k), u^p_i(\tau; t_k))$$

$$= \min_{u^p_i} \int_{t_k}^{t_k+T} L_i(x^p_i(s; t_k), \hat{x}_{-i}(s; t_k), u^p_i(s; t_k)) ds + \gamma \|x^p_i(t_k + T; t_k) - x^C_i\|^2_{P_i}$$

s.t.  
$$\dot{x}^p_i(\tau; t_k) = f_i(x^p_i(\tau; t_k), u^p_i(\tau; t_k))$$
$$\dot{\hat{x}}^p_i(\tau; t_k) = f_i(\hat{x}^p_i(\tau; t_k), \hat{u}^p_i(\tau; t_k))$$
$$\dot{\hat{x}}_{-i}(\tau; t_k) = f_{-i}(\hat{x}_{-i}(\tau; t_k), \hat{u}_{-i}(\tau; t_k))$$
$$u^p_i(\tau; t_k) \in U$$
$$\|x^p_i(\tau; t_k) - \hat{x}_i(\tau; t_k)\| \leq \delta^2 \kappa$$
$$x^p_i(t_k + T; t_k) \in \Omega_i(\epsilon_i)$$

The second last constraint is a “compatibility” constraint, enforcing consistency between what agent #$i$ plans to do and what its neighbors believe it plans to do. The last constraint is a terminal constraint.

Under certain technical assumptions, the authors prove that the DMPC problems are feasible at each update step $k$, and under certain bounds on the update interval $\delta$ convergence to a given set is also proved. Note that closed-loop stability is ensured by constraining the state trajectory predicted by each agent to stay close enough to the trajectory predicted at the previous time step that has been broadcasted. The main drawback of the approach is the conservativeness of the compatibility constraint.

### 1.3.4 DMPC approach of Keviczky, Borrelli, and Balas

Dynamically decoupled submodels are also considered in [55], where the special nonlinear discrete-time system structure $x^i_{k+1} = f^i(x^i_k, u^i_k)$ is assumed,
subject to local input and state constraints $x^i_k \in \mathcal{X}^i, u^i_k \in \mathcal{U}^i, i = 1, \ldots, M$. Subsystems are coupled by the cost function $l(\tilde{x}, \tilde{u}) = \sum_{i=1}^{N_e} l^i(x^i, u^i, \tilde{x}^i, \tilde{u}^i)$ and by the global constraints $g^{i,j}(x^i, x^j, u^i, u^j) \leq 0, (i, j) \in \mathcal{A}$ where $\mathcal{A}$ is a given set. Each local MPC controller is based on the optimization of the following problem

$$\min_{\tilde{U}^i_t} \sum_{k=0}^{N-1} l(\tilde{x}^i_{k+1,t}, \tilde{u}^i_{k,t}) + l_N(\tilde{x}^i_{N,t})$$  \hspace{1cm} (1.8a)

s.t. $x^i_{k+1,t} = f^i(x^i_{k,t}, u^i_{k,t})$ \hspace{1cm} (1.8b)

$x^i_{k,t} \in \mathcal{X}^i$, $u^i_{k,t} \in \mathcal{U}^i, k = 1, \ldots, N - 1$ \hspace{1cm} (1.8c)

$x^i_{N,t} \in \mathcal{X}^i_f$ \hspace{1cm} (1.8d)

$x^j_{k+1,t} = f^j(x^j_{k,t}, u^j_{k,t}), (i, j) \in \mathcal{A}$ \hspace{1cm} (1.8e)

$x^j_{k,t} \in \mathcal{X}^j, u^j_{k,t} \in \mathcal{U}^j, (i, j) \in \mathcal{A} k = 1, \ldots, N - 1$ \hspace{1cm} (1.8f)

$x^j_{N,t} \in \mathcal{X}^j_f, (i, j) \in \mathcal{A}$ \hspace{1cm} (1.8g)

$g^{i,j}(x^i_{k,t}, u^i_{k,t}, x^j_{k,t}, u^j_{k,t}) \leq 0, (i, j) \in \mathcal{A} k = 1, \ldots, N - 1$ \hspace{1cm} (1.8h)

$x^i_{0,t} = x^i_t, \tilde{x}^i_{0,t} = \tilde{x}^i_t$ \hspace{1cm} (1.8i)

where (1.8b)–(1.8d) are the local model and constraints of the agent, (1.8e)–(1.8g) are the model and constraints of the neighbors, and (1.8h) represent interaction constraints of agent #i with its own neighbors.

The information exchanged among the local MPC agents are the neighbors’ current states, terminal regions, and local models and constraints. Only the optimal input $u^i_{0,t}$ computed by controller #i is applied; the remaining inputs $u^j_{k,t}$ are completely discarded, as they are only used to enhance the prediction.

Stability is analyzed for the problem without coupling constraints (1.8h), under the assumption that the following inequality holds

$$\sum_{k=1}^{N-1} 2\|Q(x^i_{k,t} - x^j_{k,t})\|_p + \|R(u^i_{k,t} - u^j_{k,t})\|_p \leq \|Qx^i_t\|_p + \|Qx^j_t\|_p + \|Q(x^i_t - x^i_t)\|_p + \|Ru^i_{0,t}\|_p + \|Ru^j_{0,t}\|_p$$

where $\|Qx\|_2 \triangleq x'Qx$, and $\|Qx\|_1$, $\|Qx\|_\infty$ are the standard $q$ and $\infty$ norm, respectively. The value function of each agent, i.e., the optimal cost of (1.8a)
1.3.5  DMPC approach of Mercangöz and Doyle

The distributed MPC and estimation problems are considered in [72] for square plants (the number of inputs equals the number of outputs) perturbed by noise, whose local prediction models are

\[
\begin{align*}
    x_i(k+1) &= A_i x_i(k) + B_i u_i(k) + \sum_{j=1}^{M} B_j u_j(k) + w_i(k) \\
    y_i(k) &= C_i x_i(k) + v_i(k)
\end{align*}
\] (1.9)

A distributed Kalman filter based on the local submodels (1.9) is used for state estimation. The DMPC approach is similar to Venkat et al.’s “communication-based” approach, although only first moves \( u_j(k) \) are transmitted and assumed frozen in prediction, instead of the entire optimal sequences. Only constraints on local inputs are handled by the approach. Although general stability and convergence results are not proved in [72], experimental results on a four-tank system are reported to show the effectiveness of the approach.

1.3.6  DMPC approach of Magni and Scattolini

Another interesting approach to decentralized MPC for nonlinear systems has been formulated in [66]. The problem of regulating a nonlinear system affected by disturbances to the origin is considered under some technical assumptions of regularity of the dynamics and of boundedness of the disturbances. Closed-loop stability is ensured by the inclusion in the optimization problem of a contractive constraint. The considered class of functions and the absence of information exchange between controllers leads to some conservativeness of the approach.

1.4  Other non-centralized approaches

This section is based on the excellent survey [99], a comprehensive state-of-the-art of the non-centralized MPC field. This Thesis presents decentralized approaches in Chapters 2 and 3, Chapter 4 is focused on hierarchical MPC, while the two ideas are employed in a real case application in Chapter 5. Here we discuss and classify distributed MPC which is the other relevant approach to non-centralized MPC reported in literature.
We recall that the major difference between decentralized and distributed MPC consists on type and amount of information exchange occurring through the communication network. In particular in decentralized MPC no communication occurs among controllers, sensor-to-controller and controller-to-actuator are the sole type of information exchanged. Instead, the idea of distribution relies in the possibility of coordination among controllers, realized via inter-controller communication.

Clearly, the mechanism at the very basis of coordination (in literature usually referred as “consensus”) is the knowledge of other agents’ intentions, basing on which the local controller may trade its own objective in favor of others’. The use of MPC in the distributed context suggests straightforwardly the kind of information to be sent over the network to describe the local controller future behavior. In fact, future input prediction are computed together with the current control move as solution of an optimization problem, and are likely to be informative for controller future actions. Therefore, through this information a controller has an insight of others’ behavior and can act consequently. The natural drawback is the time needed to carry out communications (worse in case of unreliable medium) which unavoidably decrease sampling frequency and hence, indirectly, reactivity of control. Moreover, the knowledge of other controllers future action can be of use only if a model, possibly approximated, of their interaction with the local model is somehow available. This can be done by augmenting the state of the local model, with an increase of complexity of the optimization problem (which grows more than linearly with number of variables and constraints).

The first feature that distinguishes a distributed approach is the addressees type, which can be either the entire set of other controllers or a subset of them. First case is denoted as fully connected, while the second is partially connected. The most suitable connectivity type, given a certain application, is a choice that strongly depends on the mutual interaction among subsystem and the trade-off parameters, namely simplicity (small use of the network) and performance (relevant use of communication). Usually, only very strongly coupled subsystem may require the use of fully connected approaches.

The coordination process, which consists of solving the optimization problem and then transmit relative results (current and future control moves) to
the others, can be repeated after receiving other controllers’ predictions, with
the clear intent of improving cooperation. The number of times that such
a procedure is iterated determines the iteration type. If the communication
occurs only a single time per sampling interval, than the approach is labeled
as non-iterative, otherwise iterative.

Furthermore, the local optimization can be carried out by considering a
local or global objective function, i.e., each controller may or may not account
for other controllers’ objectives in the computation of their optimal control
move. We stress the difference between the availability of other controllers’
prediction and the use of a global control objective. In fact, it is possible
that a local controller may use predictions of other controller future moves
to “estimate” their influence on local states, and solely optimize over local
objectives. In case other controller prediction and measurements are used to
optimize a cost function that account for not-only-local states, the approach
is referred to as “cooperating”.

Approaches available in literature shows that when MPC optimization on
each local controller is carried out independently a Nash equilibrium can be
reached. In cooperative approaches a Pareto equilibrium can be achieved. We
note that Nash equilibrium is in general not sufficient for stability, thus some
ad-hoc constraints are imposed on each controller.

In order to better understand the differences between cooperative and
independent distributed MPC we informally report definitions of Nash and
Pareto equilibria.

- **Nash equilibrium**: In our context, such equilibrium is reached when
each controller cannot improve his local objective function (it is unaware
of other objectives) by varying any of his control action components;

- **Pareto equilibrium**: it occurs when an improvement on any of the
local objectives, obtained by a input variation, would necessarily lead
to a worsening of the global objective.

It is evident that independent controllers are unaware of a global objective,
thus cannot possibly achieve (with guarantees) a Pareto equilibrium. In prac-
tical situations the Nash equilibria can have a much worse performance than
the equivalent Pareto as there is a lack of a global over-viewer which may
notice that a small worsening of a single objective may lead to a convenient overall improvement. However, despite the obvious performance improvement of cooperative approaches, the price to pay in terms of quantity and spread of communications needed to realize such approach may prove it inconvenient in the majority of “simple” applications.

Overall, we shall conclude that the tuning knob of non-centralized single-layer MPC are:

- data exchange type: measurements/control (decentralized MPC) only, or also controllers info (distributed MPC);
- connectivity type: fully or partial;
- iteration type: iterative or non-iterative;
- cost function type: local (Nash equilibrium, or need of stability constraints) or global (Pareto equilibrium).

So depending on plant complexity, expressed as coupling degree among subsystem and required sampling interval, communication complexity, expressed as communication range and reliability, one may trade decentralized or distributed approaches. Among the latter, a further free degree is given by iteration of consensus, i.e. agreement, and global objectives.

### 1.5 Thesis structure and contribution

This thesis is organized as follows. After the “state-of-the-art” summary in this introduction, a decentralized linear controller synthesis approach is described in Chapter 2 which serves in some sense as the basis for the more sophisticated control techniques reported later. In Chapter 3, decentralized model predictive control is discussed along with stability properties for LTI (linear time-invariant) systems; a particular emphasis is dedicated to the trade-off arising from the decomposition degree. A hierarchical approach, composed of a two layer control structure is presented in Chapter 4, where the upper control layer is implemented in a twofold fashion, namely MPC and DMPC. Chapter 5 presents a real case study, the Barcelona water distribution
network, in which hierarchical decentralized MPC is discussed as a valuable alternative to centralized MPC.

The theoretical work reported in this thesis was the basis for the development of the WIDE Toolbox for Matlab [4] and contributed to a part of the MOBY-DIC Toolbox for Matlab [79]. For this reason two appendices are dedicated to those tools: Appendix A proposes additional examples for the theoretical approaches previously described, which are included in the toolbox release; Appendix B introduces the MOBY-DIC Toolbox for Matlab, describing its functionalities oriented toward efficient circuit implementation of explicit MPC control laws. A recent work, which can be seen as extension of the MOBY-DIC Toolbox, is finally included in Appendix C, that proposes a method to approximate an explicit MPC control law on a simplicial partition, so to exploit efficient FPGA (field programmable gate arrays) implementation while retaining closed-loop stability properties.

In the sequel a brief introduction of each chapter is given:

- Chapter 2 proposes an approach based on linear matrix inequalities (LMI) for synthesizing a set of decentralized regulators for discrete-time linear systems subject to input and state constraints. Measurements and command signals are exchanged over a sensor/actuator network, in which some links are subject to packet dropout. The resulting closed-loop system is guaranteed to asymptotically reach the origin, even if every local actuator can exploit only a (possibly time-varying) subset of state measurements. A model of packet dropout based on a finite-state Markov chain is also considered to exploit available knowledge about the stochastic nature of the network. For such model, a set of decentralized switching linear controllers is synthesized that guarantees mean-square stability of the overall controlled process under packet dropout and soft input and state constraints. The proposed control techniques are compared with standard centralized linear controllers and simulation results are reported.

- Chapter 3 proposes a decentralized model predictive control (DMPC) scheme for large-scale dynamical processes subject to input constraints. The global model of the process is approximated as the decomposition of several (possibly overlapping) smaller models used for local predic-
tions. The degree of decoupling among submodels represents a tuning knob of the approach: the less coupled are the submodels, the lighter the computational burden and the load for transmission of shared information; but the smaller is the degree of cooperativeness of the decentralized controllers and the overall performance of the control system. Sufficient criteria for analyzing asymptotic closed-loop stability are provided for input constrained open-loop asymptotically stable systems and for unconstrained open-loop unstable systems, under possible intermittent lack of communication of measurement data between controllers. The DMPC approach is also extended to asymptotic tracking of output set-points and rejection of constant measured disturbances. The effectiveness of the approach is shown on a relatively large-scale simulation example on decentralized temperature control based on wireless sensor feedback.

- Chapter 4 proposes two hierarchical multi-rate control design approaches to linear systems subject to linear constraints on input and output variables. At the lower level, a linear controller stabilizes the open-loop process without considering the constraints. The first strategy named centralized consists of a higher-level controller that commands reference signals at a lower uniform sampling frequency so as to enforce linear constraints on the process variables. The decentralized extension exploits the possible weak coupling among different process areas to push the reference feedback frequency where constraints are softer. Worst case subsystem interactions are modeled and accounted for in a robust manner. By optimally constraining the magnitude and rate of variation of the reference signals to each lower-level controller, quantitative criteria are provided for selecting the ratio between the sampling rates of the upper and lower layers of control at each location, in a way that closed-loop stability is preserved and the fulfillment of the prescribed constraints is guaranteed.

- In Chapter 5, a hierarchical and decentralized model predictive control (DMPC) strategy for drinking water networks (DWN) is proposed. The DWN is partitioned in a set of subnetworks using a partitioning algorithm that makes use of the topology of the network, historic information
about the actuator usage and heuristics. A suboptimal DMPC strategy was derived, which consists in a set of MPC controllers, whose prediction model is a plant partition, where each element solves its control problem in a hierarchical order. A comparative simulation study between centralized MPC (CMPC) and DMPC approaches is developed using a case study, which consists in an aggregate version of the Barcelona DWN. Results have shown the effectiveness of the proposed DMPC approach in terms of the scalability of computations with an acceptable admissible loss of performance in all the considered scenarios.

- Appendix A briefly describes the goals of the WIDE Toolbox for Matlab and then reports some commented examples, part as demo of the toolsox, which serves as additional demonstration of the theory developed in this thesis.

- Appendix B describes a MATLAB Toolbox for the integrated design of MPC state-feedback control laws and the digital circuits implementing them. Explicit MPC laws can be designed using optimal and sub-optimal formulations, directly taking into account the specifications of the digital circuit implementing the control law (such as latency and size), together with the usual control specifications (stability, performance, constraint satisfaction). Tools for a-posteriori stability analysis of the closed-loop system, and for the simulation of the circuit in Simulink, are also included in the toolbox.

- Appendix C is dedicated to design an approximate explicit MPC approach for regulating linear time-invariant systems subject to both state and control constraints. The proposed control law is implemented as a piecewise-affine function defined on a regular simplicial partition, and has two main positive features. First, the regularity of the simplicial partition allows a very efficient implementation of the control law on digital circuits, with computation performed in tens of nanoseconds. Second, the asymptotic stability of the closed-loop system is enforced a priori by design.
Chapter 2

Decentralized networked linear regulator

2.1 Introduction

Networked control systems (NCSs) are characterized by a topological distribution over the physical space that sometimes prevents the use of centralized control solutions. In fact, the set of measurements might not be available at each control instant, due for instance to temporarily or permanently faulty sensors connections. A natural workaround is to define a set of controllers, each one in charge of commanding only a subset of actuators. The underlying idea is that the information provided by a subset of sensor measurements might be enough to control a subset of actuators satisfactorily. In this case, a decentralized control scheme clearly reduces the communication traffic over the network, allowing for a simpler network structure.

These considerations led, since the 70’s, to look with interest to decentralized control, mainly investigating stability properties [98]. In the 90’s, the rise of convex optimization techniques allowed for convex formulations of decentralized control problems [33,93]. Decentralized/distributed estimation and control schemes based on distributed convex optimization ideas have been proposed recently by means of Lagrangian relaxations [53,97], where global solutions are achieved after a (possibly large) number of inter-agent communications. Hence, looking at a real implementation, the sample time must be set conservatively high in order to let all the agreements to conclude (or, at lead, reach an acceptable state) without having consequences on the control action. Moreover, the need of mutual exchange of information between network agents produces an overhead in the communication channel which must be taken into account when dealing with network-related issues such as delay and packet loss.

In this Chapter we present an approach for the off-line synthesis of a set
of decentralized linear regulators for discrete-time linear systems subject to input and state constraints. Measurements are provided by a distributed set of sensors to a distributed set of actuators (where the control law computation takes place) through a network connection, in which some of the links are subject to random packet dropout. We aim at enforcing stability of the closed-loop system for every possible combination of packet losses that can occur in the network at every time step. Conservativeness of the resulting control law is reduced by using a different set of local control laws for every possible network configuration, without the need of communication among different controllers. Moreover, we take into account a model of packet dropouts based on a finite-state Markov chain, in order to exploit available knowledge about the stochastic nature of the network, and improve the closed-loop performance.

In the last years, mean-square stability of networked control systems (NCSs) has been often analyzed in literature. For example, in [117] a stabilizing controller for linear systems subject to random but bounded delays in the feedback loop is designed by augmenting the state vector and modeling the overall process as a Markov jump linear system. A NCS subject to communication constraints is studied in [75], where a Markov model is used to represent the dynamics of the transmission update times, and stability is guaranteed by means of a stochastic quadratic Lyapunov function. More recently, a framework to analyze stability of stochastic linear NCSs subject to time-varying transmission intervals, delays, packet dropouts and communication constraints by means of overapproximation methods has been proposed in [35]. Most of these papers (if not all, and this work makes no exception) rely on convex optimization, and more specifically on the formulation of optimization problems constrained by a set of linear matrix inequalities (LMIs) [19,31].

This Chapter is organized as follows. In Section 2.2 we introduce the problem of synthesizing a decentralized controller for a NCS where local controllers cannot exploit the knowledge of the whole state measurements. In Section 2.3 we consider the case in which a subset of the network connections is subject to random packet dropouts, and obtain a switching decentralized controller which guarantees robust stability of the global system. In Section 2.4 we
extend the approach to consider a stochastic model of packet loss based on a Markov chain, and derive a controller which enforces closed-loop stability in mean-square sense. Simulations results are presented in Section 2.5, and conclusions are drawn in Section 2.6.

2.2 Control over ideal networks

Consider the discrete-time time-invariant linear system

\[ x(t+1) = Ax(t) + Bu(t), \]  

where \( x = [x_1, \ldots, x_n]' \in \mathbb{R}^n \) is the state, \( u = [u_1, \ldots, u_m]' \in \mathbb{R}^m \) is the input, \( t \in \mathbb{N}_0 \) is the time index, and the matrices \( A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m} \). We assume that states and inputs are subject to the constraints

\[ \|x(t)\|_2 \leq x_{\max}, \quad \|u(t)\|_2 \leq u_{\max}, \quad \forall t \in \mathbb{N}_0. \]  

The process we consider is a networked control system, where spatially distributed sensor nodes provide measurements of the system state, and spatially distributed actuator nodes implement the control action. More in detail, at every time step \( t \) every sensor \( s_1, \ldots, s_n \) measures a component \( x_i(t) \) of the state vector, \( i = 1, \ldots, n \). Then, measurements are sent to actuators \( a_1, \ldots, a_m \) through a user-defined networked connection. Note that in general a measured value can be transmitted to several different actuator nodes. The sensors that measure and broadcast these quantities are referred to as shared sensors. Given a process of the form (2.1) we define its network topology by means of an adjacency matrix \( \Lambda \in \{0,1\}^{m \times n} \) with elements

\[ \lambda_{ij} = \begin{cases} 1 & \text{if sensor } s_j \text{ is linked to actuator } a_i, \\ 0 & \text{otherwise.} \end{cases} \]  

for \( i = 1, \ldots, m \) and \( j = 1, \ldots, n \). In other words, \( \lambda_{ij} = 1 \) if and only if the measurement \( x_j(t) \) can be exploited to compute the input signal \( u_i(t) \), \( \forall t \in \mathbb{N}_0 \). We assume here that all the network links are ideal (no packet dropout,

\footnote{Other kinds of constraints, such as element-wise bounds, can be considered in a similar fashion (see, e.g., [58]).}
2. Decentralized networked linear regulator

Figure 2.1: Example of an ideal network topology with shared sensors $s_3$, $s_4$ and $s_5$ delays, etc.); the decentralized control problem is extended to consider packet dropouts in Section 2.3.

In Figure 2.1 a scheme of a networked control system with

$$
\Lambda = \begin{bmatrix}
0 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 \\
\end{bmatrix}
$$

is shown to exemplify notation, where circles and squares indicates sensors and control-actuators, respectively.

2.2.1 Linear controller synthesis

Our goal is to find a gain matrix $K \in \mathbb{R}^{m \times n}$ such that the system (2.1) in closed-loop with

$$u(t) = Kx(t)$$

(2.4)

is asymptotically stable. The desired control law must be decentralized, i.e., each actuator $a_1, \ldots, a_m$ can only exploit the measurements that are available
2.2. Control over ideal networks

in accordance with the network topology (2.3). In other words, each row $i$ of $K$ can only have non-zero elements in correspondence with the state measurements available to actuator $a_i$, $i = 1, \ldots, m$. This imposes the following structure on $K$:

$$\lambda_{ij} = 0 \Rightarrow k_{ij} = 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n,$$ (2.5)

where $k_{ij}$ is the $(i,j)$-th element of $K$.

Closed-loop stability is enforced through the condition

$$V(x(t + 1)) - V(x(t)) \leq -x(t)'Q_xx(t) - u(t)'Q_uu(t),$$ (2.6)

where $V : \mathbb{R}^n \to \mathbb{R}$ is a Lyapunov function of the state $x$ for the closed-loop system given by (2.1) and (2.4), and $Q_x \in \mathbb{R}^{n \times n}$, $Q_u \in \mathbb{R}^{m \times m}$ are weight matrices, with $Q_x = Q_x' > 0$, $Q_u = Q_u' \succeq 0$. In the following we consider quadratic Lyapunov functions, and define

$$V(x) \triangleq x'Px,$$ (2.7)

with $P \in \mathbb{R}^{n \times n}$, $P = P' > 0$. It is well known that satisfaction of (2.6), i.e. Lyapunov function decreasing rate, for all time steps $t \in \mathbb{N}_0$ implies asymptotical stability of the closed-loop system (see, e.g., [58]). If (2.6) is satisfied, then we can show that

$$V(x(t)) \geq J_\infty(t) \triangleq \sum_{i=0}^{\infty} (x(t + i)'Q_xx(t + i) + u(t + i)'Q_uu(t + i)),$$

i.e. $V(x(t))$ is an upper bound of the infinite-horizon quadratic cost-to-go $J_\infty(t)$ defined by $Q_x$, $Q_u$ [58]. Our goal is to find the smallest scalar $\gamma > 0$ such that

$$x(t)'Px(t) \leq \gamma, \quad \forall t \in \mathbb{N}_0,$$ (2.8)

or, equivalently, $x(t)'Q^{-1}x(t) \leq 1, \quad \forall t \in \mathbb{N}_0$, by substituting $Q = \gamma P^{-1}$. Clearly, the satisfiability of (2.8) depends on the initial state $x(0)$. Rather than finding the proper value of $\gamma$ for a given initial state $x(0) \in \mathbb{R}^n$, we look for a $\gamma$ which is valid for all $x(0) \in \mathcal{X}_0 \subset \mathbb{R}^n$, where $\mathcal{X}_0 \triangleq \mathcal{H}(v_1, \ldots, v_n_v)$ is a polytope with vertices $v_1, \ldots, v_n_v$, and $\mathcal{H}(\cdot)$ denotes the convex hull operator, so that the controller $K$ that we are going to synthesize is valid for any initial
condition $x(0) \in X_0$. As noted in [31], by making the standard substitution $K = YQ^{-1}$, $Y \in \mathbb{R}^{m \times n}$, we can obtain any desired structure for $K$ by imposing the same structure on $Y$ and fixing the block-diagonal structure of $Q$.

\[(\lambda_{ij} = 0) \Rightarrow y_{ij} = 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n, (2.9a)\]

\[(\lambda_{ij} = 0) \land (\lambda_{ih} = 1) \Rightarrow q_{hj} = 0, \quad q_{jh} = 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n, \quad h = 1, \ldots, n, (2.9b)\]

where $\land$ denotes logical “and”. For instance, with reference to Figure 2.1 we have

\[Y = \begin{bmatrix} 0 & 0 & * & * & * \\
* & * & 0 & 0 & * \\
0 & 0 & * & * & 0 \end{bmatrix}, \quad Q = \begin{bmatrix} * & * & 0 & 0 & 0 \\
* & * & 0 & 0 & 0 \\
0 & 0 & * & * & 0 \\
0 & 0 & 0 & 0 & * \end{bmatrix}.\]

**Theorem 1.** Consider an ideal network with topology $\Lambda \in \{0, 1\}^{m \times n}$ and let $P = \gamma Q^{-1}$, $K = YQ^{-1}$ be obtained by solving the semidefinite programming (SDP) problem

\[
\min_{\gamma, Q, Y} \gamma \quad \text{(2.10a)}
\]

s.t. \[
\begin{bmatrix}
Q & \ast & \ast & \ast \\
AQ + BY & Q & \ast & \ast \\
Q^{1/2}Q & 0 & \gamma I_n & \ast \\
Q^{1/2}Y & 0 & 0 & \gamma I_m
\end{bmatrix} \succeq 0, \quad (2.10b)
\]

\[
\begin{bmatrix}
Q & \ast \\
AQ + BY & x_{\text{max}}^2 I_n
\end{bmatrix} \succeq 0, \quad (2.10c)
\]

\[
\begin{bmatrix}
\frac{1}{v_i} & \ast \\
Y & Q
\end{bmatrix} \succeq 0, \quad i = 1, \ldots, n_v, \quad (2.10d)
\]

\[
Y \in \mathcal{Y}, \quad Q \in \mathcal{Q}, \quad (2.10e)
\]

where $I_n$ is the identity matrix in $\mathbb{R}^{n \times n}$, $0$ is a matrix of appropriate dimension with all zero entries.

\[Q \triangleq \{Q \in \mathbb{R}^{n \times n} : q_{hj} = q_{jh} = 0 \text{ if } (\lambda_{ij} = 0) \land (\lambda_{ih} = 1), \quad i = 1, \ldots, m, \quad j = 1, \ldots, n, \quad h = 1, \ldots, n\},
\]

\[\mathcal{Y} \triangleq \{Y \in \mathbb{R}^{m \times n} : y_{ij} = 0 \text{ if } \lambda_{ij} = 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n\},\]
and \( q_{ij}, y_{ij} \) are the \((i,j)\)-th elements of \( Q \) and \( Y \), respectively. If problem (2.10) is feasible, then system (2.1) with initial state \( x(0) \in X_0 \) in closed-loop with the decentralized constant feedback control law (2.4) is asymptotically stable and satisfies the constraints (2.2).

**Proof.** In the particular case where \( X_0 = \{x_0\} \) is a singleton (i.e., the initial state \( x(0) = x_0 \) is fixed), and \( \Lambda = 1_{m \times n} \) is a matrix with all one entries (i.e., the control law is centralized and we have no constraints on the structure of \( K \)), asymptotical stability is a well known result that follows by showing that \( V(x(t)) = x(t)'P x(t) \) is a Lyapunov function for the closed-loop system (see, e.g., [19, 58]). Substituting \( P = \gamma Q^{-1} \), condition (2.6) is converted by means of Schur complements to the LMI (2.10b). Using similar arguments, state and input constraints (2.2) are enforced by (2.10c) and (2.10d). It remains to prove (i) that stability is retained for every initial state \( x(0) \in X_0 \) when \( X_0 \) contains more than one point, and (ii) that the control law \( u(t) = K x(t) \), with \( K = Y Q^{-1} \), can be implemented in a decentralized way, according to the network topology \( \Lambda \). We see that (i) follows by convexity of the ellipsoid \( E_Q \triangleq \{ x \in \mathbb{R}^n : x'Q^{-1} x \leq 1 \} \). In fact, since \( E_Q \) contains the vertices \( v_i \) of \( X_0 \) due to (2.10e), then \( X_0 \subset E_Q \). Regarding (ii), as the structure of diagonal blocks is preserved by matrix inversion, \( Q \) block diagonal implies that \( Q^{-1} \) is also block-diagonal, and hence (2.9b) implies that \( q_{kj} = 0 \) and \( q_{kj} = 0 \), for all \( i,j,h \) such that \( \lambda_{ij} = 0 \) and \( \lambda_{ih} = 1 \), where \( q_{kj} \) is the \((h,j)\)-th element of \( Q^{-1} \). Since \( k_{ij} = \sum_{h=1}^n y_{ih} q_{kj} \), by (2.10f) it follows that the decentralized structure (2.5) is satisfied.

**Remark 1.** We shall note that conditions 2.9 are sufficient to ensure a decentralized structure on \( K \), but are not necessary. The conservatism degree can potentially be decreased if less stringent conditions on \( Y \) and \( Q \) are proved to suffice for guaranteeing the structure on \( K \). Moreover, \( Q \) is proportional to Lyapunov matrix \( P \), thus any constraint on \( Q \) restrict the set of admissible \( P \). A current research direction consider the work of [34] to decouple \( P \) from \( K \), hence reducing conservatism.
2.3 Control over lossy networks

In this section we consider packet dropouts occurring in some of the links of the communication network, referred to as lossy links. In the following, the control law computed at each instant can exploit the set of measurements currently available. Any missing data is ignored and no retransmission request takes place. To account for the presence of lossy links in the network, we extend the definition of the topology \( \Lambda \in \{-1, 0, 1\}^{m \times n} \) as follows

\[
\lambda_{ij} = \begin{cases} 
1 & \text{if ideal link between } s_j \text{ and } a_i, \\
-1 & \text{if lossy link between } s_j \text{ and } a_i, \\
0 & \text{if no link between } s_j \text{ and } a_i,
\end{cases}
\]  

(2.11)

for \( i = 1, \ldots, m, j = 1, \ldots, n \). In Figure 2.2 a scheme of a networked control system with

\[
\Lambda = \begin{bmatrix} 
0 & 0 & 1 & -1 & 1 \\
1 & 1 & 0 & 0 & -1 \\
0 & 0 & -1 & 1 & 0
\end{bmatrix}
\]

is shown to exemplify the notation. The network in Figure 2.2 may represent a NCS with wireless links \( s_4 \rightarrow a_1, s_5 \rightarrow a_2, s_3 \rightarrow a_3 \) and wired links \( s_5 \rightarrow a_1, s_3 \rightarrow a_1, s_1 \rightarrow a_2, s_2 \rightarrow a_2, s_4 \rightarrow a_3 \). No probabilistic model of packet loss is considered here; this will be introduced in Section 2.4.

We denote with \( l_i \) the number of lossy links connected with actuator \( a_i, i = 1, \ldots, m \) (i.e., the number of "-1" in the \( i \)-th row of \( \Lambda \)), and with \( L = \sum_{i=1}^{m} l_i \) the total number of lossy links in the network. Then, we can enumerate all the possible combinations of packet dropouts at a given time step \( t \) by replacing every "-1" in \( \Lambda \) with either a "1" or a "0". In this way we obtain a set of \( \ell = 2^L \) matrices \( \tilde{\Lambda}_h \in \{0, 1\}^{m \times n} \), \( h = 1, \ldots, \ell \), which describes all the possible network configurations. We denote by \( \tilde{\Lambda}(t) \in \{\tilde{\Lambda}_1, \ldots, \tilde{\Lambda}_\ell\} \) the network configuration at time \( t \in \mathbb{N}_0 \).
2.3 Control over lossy networks

2.3.1 Switching controller synthesis

We want to design a set of gains $K_h \in \mathbb{R}^{m \times n}$, $h = 1, \ldots, \ell$, to be used in the decentralized switching feedback control law

$$u(t) = \begin{cases} K_1 x(t) & \text{if } \tilde{\Lambda}(t) = \tilde{\Lambda}_1, \\ K_2 x(t) & \text{if } \Lambda(t) = \Lambda_2, \\ \vdots & \vdots \\ K_\ell x(t) & \text{if } \tilde{\Lambda}(t) = \tilde{\Lambda}_\ell. \end{cases} \quad (2.12)$$

Note that in general the implementation of (2.12) requires the controllers to be aware of the whole network status $\tilde{\Lambda}(t)$. This hypothesis is obviously not realistic in a decentralized framework. Hence, we impose an appropriate structure of the gains $K_1, \ldots, K_\ell$, so that every local actuator $a_i$, $i = 1, \ldots, m$, needs only to know which local measurements have been lost, regardless of the links status in the rest of the network. To accomplish this, we need to have a control law which univocally defines $u_i(t)$, $\forall i$, for all the network configurations $\tilde{\Lambda}_h$ that have identical values in their $i$-th row. Namely, $[M]_i$.
being the $i$-th row of a generic matrix $M$, we want to impose

$$[	ilde{\Lambda}_h]_i = [	ilde{\Lambda}_j]_i \Rightarrow [K_h]_i = [K_j]_i,$$  \hspace{1cm} (2.13)

for all $h, j = 1, \ldots, \ell$, $i = 1, \ldots, m$. This relation greatly reduces the number of variables to be considered in our optimization problem. In fact, we have only $2^\ell$ possible values of $[\tilde{\Lambda}(t)]_i$, $i = 1, \ldots, m$. We refer to these row vectors as $\Gamma_1^i, \ldots, \Gamma_{2^\ell}^i$, where $\Gamma_j^i \in \{0, 1\}^{1 \times n}$, $\forall i, j$. Hence, we look for $\sum_{i=1}^m 2^li$ local gains $F_1^i, \ldots, F_{2^li}^i$ which define the set of element-wise feedback control laws

$$u_i(t) = \begin{cases} F_1^i x(t) & \text{if } [\tilde{\Lambda}(t)]_i = \Gamma_1^i, \\ F_2^i x(t) & \text{if } [\tilde{\Lambda}(t)]_i = \Gamma_2^i, \\ \vdots & \vdots \\ F_{2^li}^i x(t) & \text{if } [\tilde{\Lambda}(t)]_i = \Gamma_{2^li}^i, \end{cases}$$  \hspace{1cm} (2.14)

for all $i = 1, \ldots, m$. These local gains $\{F_j^i\}$ are then combined to obtain the $\ell$ global gains $\{K_h\}$ used in (2.12). Our purpose is to guarantee the satisfaction of the stability constraint (2.6) in the presence of random packet dropouts. We are looking for a robust kind of stability, where no information on the dynamics regulating the evolution in time of $\tilde{\Lambda}(t)$ are exploited. Hence, here we take $V(x)$ in (2.7) as a common Lyapunov function for the switching closed-loop dynamics $x(t+1) = (A + BK_h)x(t)$, $h = 1, \ldots, \ell$. In order to compute $K_1, \ldots, K_\ell$ we substitute

$$K_h = Y_h Q^{-1}, \ \forall h,$$  \hspace{1cm} (2.15)

allowing a different matrix $Y_h \in \mathbb{R}^{m \times n}$ for every possible network configuration $\tilde{\Lambda}_h$. However, since we have an unique Lyapunov function we also have an unique $Q$ and it must hold

$$[Y_h Q^{-1}]_i = [Y_j Q^{-1}]_i, \ \forall i, j, h,$$  \hspace{1cm} (2.16)

in order to satisfy (2.13). In other words, the structure of $Q$ needs to preserve the structure of every $K_h$, $h = 1, \ldots, \ell$, which is the element-wise logical “and” of all the structures.
Theorem 2. Consider a network with topology $\Lambda \in \{-1, 0, 1\}^{m \times n}$, and let $K_h = Y_hQ^{-1}$, $h = 1, \ldots, \ell$, be obtained by solving the SDP problem

$$\min_{\gamma, Q(Y)} \gamma$$

s.t.

$$\begin{bmatrix} AQ + BY_h & Q \ast & \ast \\ Q^{1/2}Q & 0 & \gamma I_n \\ Q^{1/2}Y_h & 0 & 0 \end{bmatrix} \succeq 0, \ h = 1, \ldots, \ell,$$  (2.17b)

$$\begin{bmatrix} Q \ast \ast \\ AQ + BY_h x_{2\max}^+ I_n \end{bmatrix} \succeq 0, \ h = 1, \ldots, \ell,$$  (2.17c)

$$\begin{bmatrix} \gamma \ast \ast \\ u_{\max}^+ I_m \end{bmatrix} \succeq 0, \ h = 1, \ldots, \ell,$$  (2.17d)

$$\begin{bmatrix} 1 \ast \ast \\ v_i \ast \\ \hat{\Lambda}_h \end{bmatrix} \succeq 0, \ i = 1, \ldots, n_v,$$  (2.17e)

$$[\hat{\Lambda}_h]_i = [\hat{\Lambda}_j]_i \Rightarrow [Y_h]_i = [Y_j]_i,$$  (2.17f)

$$Y_h \in \tilde{Y}_h, \ h = 1, \ldots, \ell,$$  (2.17g)

$$Q \in \tilde{Q},$$  (2.17h)

where

$$\tilde{Q} \triangleq \{Q \in \mathbb{R}^{n \times n} : q_{wj} = q_{jw} = 0 \text{ if } (\hat{\Lambda}_{ij}^h = 0) \land (\hat{\Lambda}_{iw}^h = 1), \ i = 1, \ldots, m, \ j, w = 1, \ldots, n, \ h = 1, \ldots, \ell\},$$

$$\tilde{Y}_h \triangleq \{Y_h \in \mathbb{R}^{m \times n} : y_{ij}^h = 0 \text{ if } \hat{\Lambda}_{ij}^h = 0, \ i = 1, \ldots, m, \ j = 1, \ldots, n\},$$

$y_{ij}^h$ is the $(i,j)$-th element of $Y_h$, and $\hat{\Lambda}_{ij}^h$ is the $(i,j)$-th element of $\hat{\Lambda}_h$, for all $h = 1, \ldots, \ell$. If problem (2.17) is feasible, then system (2.1) with initial state $x(0) \in X_0$ in closed loop with the decentralized switching feedback control law (2.12) is asymptotically stable and satisfies the constraints (2.2) for any possible realization of packet dropout.

Proof. Constraints (2.17b), obtained by using (2.12), substituting $K_h = Y_hQ^{-1}$, $h = 1, \ldots, \ell$, and taking a Schur complement, are a sufficient condition to the satisfaction of (2.6) for every $\hat{\Lambda}(t) \in \{\hat{\Lambda}_1, \ldots, \hat{\Lambda}_\ell\}$, $\forall t$. Hence robust asymptotical stability is provided for every possible realization of packet dropout. Fulfillment of state and input constraints (2.2) and robustness with respect to
the initial state \( x(0) \in X_0 \), which are respectively enforced by (2.17c)--(2.17d) and (2.17e), follow by similar arguments of Theorem 1. It remains to prove that the resulting control law (2.12) can be implemented in a decentralized way as a combination of (2.14), \( i = 1, \ldots, m \). We fix the structure of \( Y_h \) to be equal to the structure of \( K_h \), \( h = 1, \ldots, \ell \), by means of (2.17g). Since \( K_h = Y_h Q^{-1} \), we have \( k_{ij}^h = \sum_{w=1}^n y_{iw}^h \tilde{q}_{wj} \) and we must enforce the counterpart of (2.5) for the case of a switching control law, i.e., \( \tilde{\lambda}_{ij}^h = 0 \Rightarrow k_{ij}^h = 0 \), or equivalently

\[
\tilde{\lambda}_{ij}^h = 0 \Rightarrow \sum_{w=1}^n y_{iw}^h \tilde{q}_{wj} = 0, \quad \forall i, j, h.
\]

(2.18)

Being the structure of \( Y_h \) assigned for a fixed \( h \), a sufficient condition for the satisfaction of (2.18) is given by

\[
(\tilde{\lambda}_{ij}^h = 0) \land (\tilde{\lambda}_{iw}^h = 1) \Rightarrow \tilde{q}_{wj} = 0, \quad \forall i, j, w, h,
\]

which is enforced by (2.17h) noting that \( Q \) is symmetric and block-diagonal. Finally, constraint (2.17f) together with (2.15) implies satisfaction of (2.16), which is a sufficient condition for (2.13) to hold. This ensures the uniqueness of the local control law (2.14) to be implemented given \([\tilde{\Lambda}(t)]_i, i = 1, \ldots, m\), regardless of the global value of \( \tilde{\Lambda}(t) \), and proves (2.12) to be a decentralized control law with the requested structure.

\[\square\]

### 2.4 Stochastic control under packet dropout

The robust approach undertaken in the previous section can be conservative in some cases, as it requires the existence of a common Lyapunov function which must be decreasing at every time step for every possible network configuration. In this section we pursue a relaxed stability condition by introducing a probabilistic model of the network and exploiting the possibly available stochastic information on packet dropout. We consider stability in the mean-square sense, which in this framework is equivalent to

\[
\lim_{t \to \infty} E \left[ \| x(t) \|^2 \right] = 0.
\]

(2.19)

In other words, we allow the closed-loop Lyapunov function to occasionally increase from one step to another, as long as a decreasing condition of the
2.4. Stochastic control under packet dropout

The expectation is taken with respect to the realizations of $\tilde{\Lambda}(t)$, which is now modeled as a stochastic process.

2.4.1 Stochastic network model

Following the model proposed in [118], we assume that the probability distribution of the network configurations $\{\tilde{\Lambda}_h\}$ is modeled by a finite-state Markov chain with 2 states$^2$, called $Z_1$ and $Z_2$. (see Figure 2.3).

The dynamics of the Markov chain are defined by a transition matrix

$$T = \begin{bmatrix} q_1 & 1 - q_1 \\ 1 - q_2 & q_2 \end{bmatrix} \quad (2.20)$$

such that $t_{ij} = \Pr[z(t+1) = Z_j \mid z(t) = Z_i]$, and by an emission matrix $E \in \mathbb{R}^{2 \times 2}$ such that $e_{ij} = \Pr[\tilde{\Lambda}(t) = \tilde{\Lambda}_j \mid z(t) = Z_i]$, being $t_{ij}$ and $e_{ij}$ the $(i, j)$-th element of $T$ and $E$, respectively. In order to define the values in $E$ we need to compute the probabilities of occurrence of $\tilde{\Lambda}_h$, $\forall h$. We assume that the occurrence of a packet dropout at a time step $t$ in a given network link is an i.i.d.$^3$ random variable, for every state of the Markov chain. In particular, we denote with $d_1$ and $d_2$, $0 < d_1 < d_2 < 1$, the probabilities of losing a packet at time $t$ if $z(t) = Z_1$ and $z(t) = Z_2$, respectively (for example, in $Z_1$ we have

$^2$More complex Markov chain models of packet loss could be considered here, such as the one used in [5].

$^3$Independent and identically distributed.
“few” dropouts, and in \( Z_2 \) we have “many”, according to Gilbert’s model). Moreover, let \( s_{1,h} \) and \( s_{0,h} \) be the total number of lossy links in \( \Lambda \) which are mapped as ideal links and as no links in \( \tilde{\Lambda}_h \), respectively, i.e.,

\[
\begin{align*}
    s_{1,h} &= \sum_{(i,j) \in I} \lambda_{ij}^h, \quad h = 1, \ldots, \ell, \\
    s_{0,h} &= \sum_{(i,j) \in I} (1 - \lambda_{ij}^h), \quad h = 1, \ldots, \ell,
\end{align*}
\]

(2.21a) (2.21b)

where \( I \triangleq \{(i,j) \in \{1, \ldots, m\} \times \{1, \ldots, n\} : \lambda_{ij} = -1\} \). Then, we can define the elements \( \{e_{ij}\} \) of \( E \) as

\[
e_{ij} = d_i s_{0,j} (1 - d_i)^{s_{1,j}}, \quad i = 1, 2, \quad j = 1, \ldots, \ell.
\]

(2.22)

### 2.4.2 Stochastic switching controller synthesis

Our goal is to design two sets of control gains \( K_{1,1}, \ldots, K_{1,\ell}, K_{2,1}, \ldots, K_{2,\ell} \), one for every state of the Markov chain, which define the switching control law

\[
u(t) = \begin{cases} 
K_{1,1}x(t) & \text{if } z(t) = Z_1, \quad \hat{\Lambda}(t) = \hat{\Lambda}_1, \\
\vdots & \quad \vdots \\
K_{1,\ell}x(t) & \text{if } z(t) = Z_1, \quad \hat{\Lambda}(t) = \hat{\Lambda}_\ell, \\
K_{2,1}x(t) & \text{if } z(t) = Z_2, \quad \hat{\Lambda}(t) = \hat{\Lambda}_1, \\
\vdots & \quad \vdots \\
K_{2,\ell}x(t) & \text{if } z(t) = Z_2, \quad \hat{\Lambda}(t) = \hat{\Lambda}_\ell,
\end{cases}
\]

(2.23)

so that the closed-loop system is asymptotically stable in mean-square. Consider the stochastic counterpart of the decreasing condition (2.6)

\[
E[V(x(t + 1))] - V(x(t)) \leq -x(t)'Qx(t) - E[u(t)'Q_u u(t)].
\]

(2.24)

As shown in [76], fulfillment of (2.24) for all \( t \in \mathbb{N}_0 \) implies (2.19). Here \( V(x) \) is intended to be a switching stochastic Lyapunov function for the closed-loop system, defined as

\[
V(x(t)) \triangleq \begin{cases} 
x(t)'P_1 x(t) & \text{if } z(t) = Z_1, \\
x(t)'P_2 x(t) & \text{if } z(t) = Z_2.
\end{cases}
\]

(2.25)
We assume that the state $z(t) = Z_j$ of the communication network is known at time $t$, with $j \in \{1, 2\}$. Hence, the expectations in (2.24) are

$$
E[V(x(t + 1))] = \sum_{h=1}^{\ell} \sum_{l=1}^{2} e_{jh} t_{jl} x(t)' (A + BK_{j,h})' P_l (A + BK_{j,h}) x(t) \quad (2.26a)
$$

$$
E[u(t)' Q_u u(t)] = \sum_{h=1}^{\ell} e_{jh} x(t)' K_{j,h}' Q_u K_{j,h} x(t) \quad (2.26b)
$$

In light of the above considerations, by using (2.23), (2.25) and (2.26), and substituting $P_j = \gamma Q_j^{-1}, K_{j,h} = Y_{j,h} Q_j^{-1}, \forall j, h$, we can translate (2.24) to an appropriate LMI condition with standard methods, as detailed in the following theorem.

**Theorem 3.** Consider a network with topology $\Lambda \in \{-1, 0, 1\}^{m \times n}$, where at each time step the packet dropout realizations are driven by the Markov chain defined by (2.20)–(2.22), and let $K_{j,h} = Y_{j,h} Q_j^{-1}, j = 1, 2, h = 1, \ldots, \ell$, be obtained by solving the SDP problem

$$
\min_{\gamma, \{Q_j\}, \{Y_{j,h}\}} \gamma \quad (2.27a)
$$

s.t.

$$
\begin{bmatrix}
Q_j & * & * & * \\
Q_j' & \gamma I_n & * & * \\
C_{j,1} & 0 & D_{j,1} & * \\
C_{j,2} & 0 & 0 & D_{j,2}
\end{bmatrix} \succeq 0, \quad j = 1, 2, \quad (2.27b)
$$

$$
\begin{bmatrix}
Q_j & * & * & * \\
A Q_j + B V_{j,h} x_{\max}^2 I_n & \geq 0, \quad j = 1, 2, \quad h = 1, \ldots, \ell,
\end{bmatrix} \quad (2.27c)
$$

$$
\begin{bmatrix}
u_{h,j}' & * & * & & & & & & \\
& & & & & & & &
\end{bmatrix} \geq 0, \quad j = 1, 2, \quad h = 1, \ldots, \ell, \quad (2.27d)
$$

$$
\begin{bmatrix}
1 & * & & & & & & & \\
& & & & & & & &
\end{bmatrix} \geq 0, \quad i = 1, \ldots, n_v, \quad j = 1, 2, \quad (2.27e)
$$

$$
[\tilde{A}_h]_i = [\tilde{\Lambda}_w]_i \Rightarrow [Y_{j,h}]_i = [Y_{j,w}]_i, \quad j = 1, 2, \quad h, w = 1, \ldots, \ell, \quad i = 1, \ldots, m, \quad (2.27f)
$$

$$
Y_{j,h} \in \tilde{Y}_{j,h}, \quad j = 1, 2, \quad h = 1, \ldots, \ell, \quad (2.27g)
$$

$$
Q_j \in \tilde{Q}_j, \quad j = 1, 2, \quad (2.27h)
$$

\footnote{In practice, one should estimate the state $z(t)$ of the communication network, see, e.g., [24].}
where

\[
\tilde{Q}_j \triangleq \{ Q_j \in \mathbb{R}^{n \times n} : q_{lw}^j = 0, q_{wl}^j = 0 \text{ if } (\tilde{\lambda}_{lw}^h = 0) \land (\tilde{\lambda}_{wl}^h = 1) \text{ if } (\tilde{\lambda}_{lw}^h = 1), \quad w, l = 1, \ldots, n, i = 1, \ldots, m, \ h = 1, \ldots, \ell, \ j = 1, 2, \}
\]

and

\[
\tilde{Y}_{j,h} \triangleq \{ Y_{j,h} \in \mathbb{R}^{m \times n} : y_{iw}^{j,h} = 0 \text{ if } \tilde{\lambda}_{iw}^h = 0, \quad i = 1, \ldots, m, \ w = 1, \ldots, n \}, \quad \text{for all } j = 1, 2, \ h = 1, \ldots, \ell, \]

\(q_{lw}^j\) is the \((l, w)\)-th element of \(Q_j\), \(y_{iw}^{j,h}\) is the \((i, w)\)-th element of \(Y_{j,h}\), and

\[
C_{j,1} = \begin{bmatrix}
\sqrt{\epsilon_{j1}t_{j1}}(AQ_j + BY_{j,1}) \\
\vdots \\
\sqrt{\epsilon_{j1}t_{j1}}(AQ_j + BY_{j,1}) \\
\sqrt{\epsilon_{j2}t_{j2}}(AQ_j + BY_{j,1}) \\
\vdots \\
\sqrt{\epsilon_{j2}t_{j2}}(AQ_j + BY_{j,1})
\end{bmatrix}, \quad C_{j,2} = \begin{bmatrix}
\sqrt{\epsilon_{j1}t_{j1}}(Q_{u1}/2Y_{j,1}) \\
\vdots \\
\sqrt{\epsilon_{j1}t_{j1}}(Q_{u1}/2Y_{j,1}) \\
\sqrt{\epsilon_{j2}t_{j2}}(Q_{u1}/2Y_{j,1}) \\
\vdots \\
\sqrt{\epsilon_{j2}t_{j2}}(Q_{u1}/2Y_{j,1})
\end{bmatrix},
\]

\[
D_{j,1} = \text{Blkdiag}\{Q_1, \ldots, Q_1, Q_2, \ldots, Q_2\}, \quad \ell \ \text{times} \quad \ell \ \text{times}
\]

\[
D_{j,2} = \text{Blkdiag}\{\gamma I_m, \gamma I_m, \ldots, \gamma I_m\}, \quad \ell \ \text{times}
\]

If problem \((2.27)\) is feasible, then system \((2.1)\) with initial state \(x(0) \in X_0\) in closed-loop with the decentralized switching feedback control law \((2.23)\) is asymptotically stable in mean-square.

**Proof.** Constraints \((2.27b)\), obtained by using \((2.23)\), substituting \(K_{j,h} = Y_{j,h}Q^{-1}, \ j = 1, 2, \ h = 1, \ldots, \ell, \) and taking a Schur complement, are a sufficient condition to the satisfaction of \((2.24)\) for every \(\tilde{\Lambda}(t) \in \{\tilde{\Lambda}_1, \ldots, \tilde{\Lambda}_\ell\}, \ \forall t, \) distributed as modeled by \((2.20)-(2.22)\). Hence asymptotical closed-loop stability in mean-square is provided. Robustness with respect to the initial state \(x(0) \in X_0\) and desired decentralized structure of the switching feedback control law \((2.23)\), which are respectively enforced by \((2.27e)\) and \((2.27g)-(2.27h)\), follow by similar reasonings as in Theorems 1–2. The uniqueness of the switching feedback control law \((2.23)\) is imposed by constraints \((2.27f)\) for every state \(z(t) \in \{Z_1, Z_2\}\) of the Markov chain, similarly to what proved in Theorem 2 for the case of a single \(Q\) and a single set of gains \(\{Y_h\}\). Since
by assumption the current Markov chain state $z(t)$ at time $t$ is known to every actuator $a_1, \ldots, a_m$, the feedback control law is univocally determined by choosing $u(t) = K_{j,h}x(t)$ if $z(t) = Z_j$, $\hat{\Lambda}(t) = \hat{\Lambda}_h$, and this completes the proof.

\[ \square \]

**Remark 2.** As convergence to the origin provided by Theorem 3 is intended in mean-square sense, we can no more refer to $E_{Q_1} \triangleq \{ x \in \mathbb{R}^n : x'Q_1^{-1}x \leq 1 \}$ and $E_{Q_2} \triangleq \{ x \in \mathbb{R}^n : x'Q_2^{-1}x \leq 1 \}$ as invariant ellipsoids for the closed-loop system, as we did in sections 2.2–2.3. In fact, the decreasing condition (2.24) only holds in expected value. Hence, even though mean-square stability is retained, we have that $x(t) \in E_{Q_i} \neq x(t+1) \in E_{Q_j}, \forall t \in \mathbb{N}_0, i,j = 1,2$. In other words, constraints (2.27c)–(2.27d) do not imply fulfillment of (2.2) at every time step, but only in an averaged sense.

**Remark 3.** The implementation of (2.23) requires every actuator to have the knowledge of the current state of the Markov chain $z(t)$ at every time step $t$. In the cases where this assumption may not be realistic, we can derive a suitable stabilizing controller by solving (2.27) with the additional constraints $Q_1 = Q_2, Y_{1,h} = Y_{2,h}, \forall h$, so to have a control law independent of the Markov chain state value.

### 2.5 Simulation results

In this section the proposed decentralized control schemes are tested on an open-loop unstable system (2.1) with $n = 8$ states and $m = 4$ inputs. The matrices $A, B$ in (2.1) are selected randomly and hence with a high chance that state dynamics are strongly mutually coupled. Measurements are provided from sensors to actuators according to a topology $\Lambda$ as in (2.11) with 8 states. To obtain matrices $A, B$, modified to enforce one eigenvalue of $A$ to be equal to 1.05. Numerical values of $A, B$ are omitted for space reasons.

\[^5\text{The MATLAB routine drss was used to obtain the matrices } A, B, \text{ modified to enforce one eigenvalue of } A \text{ to be equal to 1.05. Numerical values of } A, B \text{ are omitted for space reasons.}\]
ideal links and 4 lossy links, defined as

$$\Lambda = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & -1 & 0 & 0 \\
-1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & -1 \\
0 & 0 & 0 & -1 & 1 & 0 & 1 & 1
\end{bmatrix}.$$ 

The network topology is schematized in Figure 2.4. Because of the network structure, the decentralized control law can only exploit a partial knowledge of the state value at each sample time. Since there are $L = 4$ unreliable links, the number of possible network configuration is $\ell = 2^L = 16$. Packet dropouts are modeled by a 2-states Markov chain defined by (2.20) with $q_1 = 0.8$, $q_2 = 0.5$, $d_1 = 0.1$ and $d_2 = 0.5$.

We run $N_{\text{sim}} = 50$ simulations of $T_{\text{sim}} = 50$ time steps each with constraints (2.2) defined by $x_{\text{max}} = 25$, $u_{\text{max}} = 3$, weight matrices $Q_x = I_n$, $Q_u = 10^{-2}I_m$, and a random initial state $x(0) \in \mathcal{X}_0$, with $\mathcal{X}_0 = \{x_c\} + \{x \in \mathbb{R}^8 : \|x\|_\infty \leq 2\}$ and $x_c = 7 \cdot 1_{8 \times 1}$. Figure 2.5 shows the behavior of the entire state and input vectors under decentralized robust and stochastic control.

Table 2.5 shows the results obtained by the proposed decentralized techniques in comparison with a centralized controller which implements the con-
2.5. Simulation results

Figure 2.5: Total (a) state and (b) input trajectories for robust (dashed line) and stochastic (solid line) decentralized controllers.

Control law (2.4) without any restriction on the structure of $K$, or, in other words, which considers a topology $\Lambda = \mathbf{1}_{m \times n}$ where every actuator can exploit all the measurements. Performances are evaluated using the cumulated stage cost

$$J_i = \sum_{t=1}^{T_{\text{sim}}} (\|Q_x x(t)\|_2 + \|Q_u u(t)\|_2)$$

over the simulation horizon, where $J_i$ refers to the $i$-th run and $\mu(J_i), \sigma(J_i)$ are the mean and the standard deviation of $J_i$ over all the simulations. We can see that the stochastic decentralized controller achieves a good closed-loop behavior, being less conservative than the robust controller and still providing convergence to the origin. In Table 2.5 is also shown the computational time needed to solve the SDP problems off-line on a 2.8GHz Intel processor with the MATLAB modeling language YALMIP [63] and the SDP solver SeDuMi [106].

Indeed, the complexity of the stochastic SDP problem (2.27), due mainly
to the size of (2.27b), requires a CPU time of an order of magnitude larger. However, it is worthwhile to point out that this increased computational load (which represent no impediment to the online implementation) provides in turn a larger solution set, since the mean-square stability constraint (2.24) is less stringent than the robust counterpart (2.6).

2.6 Conclusions

This Chapter has proposed a method based on semidefinite programming (SDP) for synthesizing decentralized linear control laws for networked linear systems, for both ideal and lossy networks. For the latter case, packet loss was modeled as a random process driven by a two-state Markov chain. The SDP problem formulation guarantees that the resulting switching controller enforces mean-square stability of the closed-loop system. Simulation results on a numerical example have shown that the performance deteriorates with respect to an ideal centralized controller, on average, by 15% in the case of stochastic decentralized control, and by 22% in the case of robust decentralized control.

<table>
<thead>
<tr>
<th></th>
<th>$\mu(J_i)$</th>
<th>$\sigma(J_i)$</th>
<th>CPU (off-line time)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ideal network</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centralized control</td>
<td>41.0</td>
<td>0</td>
<td>2.8 s</td>
</tr>
<tr>
<td>Decentralized control</td>
<td>45.1</td>
<td>0</td>
<td>1.2 s</td>
</tr>
<tr>
<td><strong>Lossy network</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dec. robust control</td>
<td>50.0</td>
<td>1.57</td>
<td>8.1 s</td>
</tr>
<tr>
<td>Dec. stochastic control</td>
<td>47.1</td>
<td>2.38</td>
<td>59.2 s</td>
</tr>
</tbody>
</table>

Table 2.1: NCS simulation results with ideal and lossy networks, using centralized, decentralized, robust and stochastic controller, also reporting computation times.
Chapter 3

Decentralized Networked Model Predictive Control

3.1 Introduction

Large-scale systems such as power distribution grids, water networks, urban traffic networks, supply chains, formations of cooperating vehicles, mechanical and civil engineering structures, and many others, are often hard to control in a centralized way. The spatial distribution of the process impedes collecting all the measurements at a single location, where complex calculations based on all such information are executed, and redistributing the control decision to all actuators; moreover constructing and maintaining a full dynamical model of the system for control design is a time consuming task. Hence the current trend for decentralized decision making, distributed computations, and hierarchical control.

In a decentralized control scheme several local control stations only acquire local output measurements and decide local control inputs, possibly under the supervision of an upper hierarchical control layer improving their coordination. Consequently, the main advantages in controller implementation are the reduced and parallel computations, and reduced communications. However, all the controllers are involved in controlling the same large-scale process, and is therefore of paramount importance to determine conditions under which there exists a set of appropriate local feedback control laws stabilizing the entire system.

Ideas for decentralizing and hierarchically organizing the control actions in industrial automation systems date back to the 70’s [98], but were mainly limited to the analysis of stability of decentralized linear control of interconnected subsystems. The interest in decentralized control raised again since the late 90’s because of the advances in computation techniques like convex
optimization [93]. Decentralized control and estimation schemes based on
distributed convex optimization ideas have been proposed recently in [53,97]
based on Lagrangean relaxations. Here global solutions can be achieved after
iterating a series of local computations and inter-agent communications.

Large-scale multi-variable control problems, such as those arising in the
process industries, are often dealt with model predictive control (MPC) tech-
niques. In MPC the control problem is formulated as an optimization one,
where many different (and possibly conflicting) goals are easily formalized and
constraints on state and control variables can be included [26,69]. However,
centralized MPC is often unsuitable for control of large-scale and networked
systems, mainly due to lack of scalability and to maintenance issues of global
models.

In view of the above considerations, it is then natural to look for decen-
tralized or for distributed MPC (DMPC) algorithms, in which the original
large-size optimization problem is replaced by a number of smaller and easily
tractable ones that work iteratively and cooperatively towards achieving a
common, system-wide control objective. In a typical DMPC framework at
each sample instant each local controller measures local variables and update
state estimates, solves the local receding-horizon control problem, applies the
control signal for the current instant, and exchanges information with other
controllers. Besides some benefits, the decentralized design also introduces
some issues: how to ensure the asymptotic stability of the overall system,
the feasibility of global constraints, the loss of performance with respect to a
centralized MPC design. We briefly review the main contributions addressing
those issues in the following paragraphs, the reader is referred to [11] for a
more detailed survey.

In [27] the system under control is composed by a number of unconstrained
linear discrete-time subsystems with decoupled input signals. The effect of dy-
namical coupling between neighboring states is modeled in prediction through
disturbance signal, while the information exchanged between control agents
at the end of each sample step is the entire prediction of the local state vector.
Under certain assumptions of the model state matrix, closed-loop stability is
proved by introducing a contractive constraint on the state prediction norm in
each local MPC problem, which the authors prove to be a recursively feasible
3.1. Introduction

In [109–111] the authors propose a distributed MPC algorithm based on negotiations among DMPC agents. The effect of the inputs of a subsystem on another subsystem is modeled by using an “interaction model”. All interaction models are assumed stable, and constraints on inputs are assumed decoupled (e.g., input saturation). Starting from a multiobjective formulation, the authors distinguish between a “communication-based” control scheme, in which each controller is optimizing his own local performance index, and a “cooperation-based” control scheme, in which each controller is optimizing a weighted sum of all performance indices. At each time step a sequence of iterations is taken before computing and implementing the input vector. With the communication-based approach, the authors show that if the sequence of iterations converges, it converges to a Nash equilibrium. With the cooperation-based approach, convergence to the optimal (centralized) control performance is established. The stability guarantees are not compromised by stopping the iterations before convergence, as only the optimality is affected in that case.

In [36] the authors consider the control of a special class of dynamically decoupled continuous-time nonlinear subsystems where the local states of each model represent a position and a velocity signal. State vectors are only coupled by a global performance objective under local input constraints, and the overall integrated cost is decomposed in distributed integrated cost functions. Before computing DMPC actions, neighboring subsystems broadcast in a synchronous way their states, and each agent transmits and receives an “assumed” control trajectory. Closed-loop stability is ensured by constraining the state trajectory predicted by each agent to stay close enough to the trajectory predicted at the previous time step that has been broadcasted, which introduces some conservativeness.

Dynamically decoupled submodels are also considered in [55], where a special nonlinear discrete-time system structure is assumed, subject to local input and state constraints. Subsystems are coupled by the cost function and by global constraints. Stability is analyzed for the problem without coupling constraints under some technical assumptions.

Distributed MPC and estimation problems are considered in [72] for square
3. Decentralized Networked Model Predictive Control

plants perturbed by noise. A distributed Kalman filter based on the local submodels is used for state estimation. The DMPC approach is similar to Venkat et al.'s “communication-based” approach, although only first moves are transmitted and assumed frozen in prediction, instead of the entire optimal sequences. Only constraints on local inputs are handled by the approach. Experimental results on a four-tank system are reported to show the effectiveness of the approach.

Another approach to decentralized MPC for nonlinear systems has been formulated in [66]. Under some technical assumptions of regularity of the dynamics and of boundedness of the disturbances, closed-loop stability is ensured by the inclusion in the optimization problem of a contractive constraint. The absence of information exchange between controllers leads to some conservativeness of the approach. Distributed Lyapunov-based MPC of nonlinear processes was also addressed in [61], and sufficient conditions for ultimately boundedness of the closed-loop system are derived in [62] in the presence of delays and asynchronous measurements.

Finally, very recently in [108] the authors introduced a robust DMPC for multiple dynamically decoupled subsystems in which distributed control agents exchange plans to achieve satisfaction of coupling constraints. The local controllers rely on the concept of “tubes” rather than single trajectories, to achieve robust feasibility and stability despite the presence of persistent, bounded disturbances.

This Chapter proposes a decentralized MPC design approach for large-scale processes that are possibly dynamically coupled and that are subject to input constraints. A (partial) decoupling assumption only appears in the prediction models used by different MPC controllers. The chosen degree of decoupling represents a tuning knob of the approach. Sufficient criteria for analyzing the asymptotic stability of the process model in closed loop with the set of decentralized MPC controllers are provided. If such conditions are not verified, then the structure of decentralization should be modified by augmenting the level of dynamical coupling of the prediction submodels, increasing consequently the number and type of exchanged information about state measurements among the MPC controllers. To cope with the case of a non-ideal communication channel among neighboring MPC controllers, suf-
3.2. Decentralized model predictive control setup

Consider the problem of regulating the discrete-time linear time-invariant system

\[
\begin{align*}
x(t+1) &= Ax(t) + Bu(t) \\
y(t) &= Cx(t) \\
u_{\text{min}} \leq u(t) \leq u_{\text{max}}
\end{align*}
\]  

(3.1)

(3.2)

to the origin while fulfilling the constraints (3.2) at all time instants \( t \in \mathbb{Z}_{0+} \), where \( \mathbb{Z}_{0+} \) is the set of nonnegative integers, \( x(t) \in \mathbb{R}^n \), \( u(t) \in \mathbb{R}^m \) and \( y(t) \in \mathbb{R}^p \) are the state, input, and output vectors, respectively, and the pair \( (A, B) \) is stabilizable. In (3.2) the constraints should be interpreted component-wise and we assume \( u_{\text{min}} < 0 < u_{\text{max}} \). A centralized MPC scheme would approach such a constrained regulation problem by solving at each time \( t \), given the
3. Decentralized Networked Model Predictive Control

Figure 3.1: Example of decomposition of a global model into four submodels. Each colored rectangle identifies the states belonging to the corresponding submodel.

The state vector $x(t)$, the following finite-horizon optimal control problem

$$
\min_U \quad x_N'Px_N + \sum_{k=0}^{N-1} x_k'Qx_k + u_k'Ru_k \\
\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, \ldots, N - 1 \\
y_k = Cx_k, \quad k = 0, \ldots, N \\
x_0 = x(t) \\
u_{min} \leq u_k \leq u_{max}, \quad k = 0, \ldots, N_u - 1 \\
u_k = Kx_k, \quad k = N_u, \ldots, N - 1
$$

where $U \triangleq \{u_0, \ldots, u_{N_u-1}\}$ is the sequence of future input moves, $x_k$ denotes the predicted state vector at time $t + k$, obtained by applying the input sequence $u_0, \ldots, u_{k-1}$ to model (3.1), starting from $x(t)$. In (3.3) $N > 0$ is the prediction horizon, $N_u \leq N - 1$ is the control horizon, $Q = Q' \succeq 0$, $R = R' > 0$, $P = P' \succeq 0$ are square weight matrices defining the performance index, and $K$ is some terminal feedback gain. Matrices $P$ and $K$ are usually chosen to ensure closed-loop stability of the overall process [13].
3.2. Decentralized model predictive control setup

3.2.1 Decentralized prediction models

We construct a set of prediction submodels based on the observation that, typically, in large-scale systems matrices $A, B$ have a certain number of zero or negligible components, corresponding to a partial dynamical decoupling of the process. Figure 3.1 depicts a dynamically coupled system decomposed into four partially overlapping subsystems.

For all $i \in \{1, \ldots, M\}$ we define $x^i \in \mathbb{R}^{n_i}$ as the vector collecting a subset $I_{xi} \subseteq \{1, \ldots, n\}$ of the state components,

$$x^i = W^i'x = [x^i_1 \cdots x^i_{n_i}]' \in \mathbb{R}^{n_i}$$

(3.4a)

where $W^i \in \mathbb{R}^{n \times n_i}$ collects the $n_i$ columns of the identity matrix of order $n$ corresponding to the indices in $I_{xi}$, and, similarly,

$$u^i = Z^i'u = [u^i_1 \cdots u^i_{m_i}]' \in \mathbb{R}^{m_i}$$

(3.4b)

as the vector of input signals tackled by the $i$-th controller, where $Z^i \in \mathbb{R}^{m \times m_i}$ collects $m_i$ columns of the identity matrix of order $m$ corresponding to the set of indices $I_{ui} \subseteq \{1, \ldots, m\}$. Note that $W^i'W^i = I_{n_i}, \quad Z^i'Z^i = I_{m_i}, \quad \forall i \in \{1, \ldots, M\}$

(3.5)

An approximation of (3.1) is obtained by changing $W^i'A$ in (3.6) into $W^i'AW^i$ and $W^i'B$ into $W^i'BZ^iZ^i'$, therefore getting the new prediction model of reduced order

$$x^i(t + 1) = A_ix^i(t) + B_iu^i(t)$$

(3.7)

where matrices $A_i = W^i'AW^i \in \mathbb{R}^{n_i \times n_i}$ and $B_i = W^i'BZ^i \in \mathbb{R}^{n_i \times m_i}$ are submatrices of the original $A$ and $B$ matrices, respectively, describing in a possibly approximate way the evolution of the states of subsystem $\# i$.

The size $(n_i, m_i)$ of model (3.7) in general will be much smaller than the size $(n, m)$ of the overall process model (3.1). The choice of the decoupling matrices $(W^i, Z^i)$ (and, consequently, of the dimensions $n_i, m_i$ of each submodel) is a tuning knob of the DMPC procedure proposed in this Chapter.
An approach to carry out the decentralization process is described in Chapter 5.

We want to design a controller for each set of moves $u^i$ according to prediction model (3.7) and based on feedback from $x^i$, for all $i \in \{1, \ldots, M\}$. Note that in general different input vectors $u^i, u^j$ may share common components. To avoid ambiguities on the control action to be commanded to the process, we impose in the design that only a subset $\mathcal{I}_{ui}^\# \subseteq \mathcal{I}_{ui}$ of input signals computed by controller #i is actually applied to the process, with the following conditions

$$\bigcup_{i=1}^{M} \mathcal{I}_{ui}^\# = \{1, \ldots, m\}$$  \hspace{1cm} (3.8a)

$$\mathcal{I}_{ui}^\# \cap \mathcal{I}_{uj}^\# = \emptyset, \forall i, j \in \{1, \ldots, M\}, \ i \neq j$$  \hspace{1cm} (3.8b)

Condition (3.8a) ensures that all actuators are commanded, condition (3.8b) that each actuator is commanded by only one controller. For the sake of simplicity of notation, since now on we assume that $M = m$ and that $\mathcal{I}_{ui}^\# = i, i = 1, \ldots, m$, i.e., that each controller #i only controls the i-th input signal. As observed earlier, in general $\mathcal{I}_{x^i} \cap \mathcal{I}_{x^j} \neq \emptyset$, meaning that controller #i may partially share the same feedback information with controller #j, and $\mathcal{I}_{ui} \cap \mathcal{I}_{uj} \neq \emptyset$. This means that controller #i may model the effect of control actions that are actually decided by another controller #j, $i \neq j, i, j = 1, \ldots, M$, which ensures a certain degree of cooperation.

The designer has the flexibility of choosing the pairs $(W_i, Z_i)$ of decoupling matrices, $i = 1, \ldots, M$. A first guess of the decoupling matrices can be inspired by the intensity of the dynamical interactions existing in the model. The larger $(n_i, m_i)$ the smaller the model mismatch and hence the better the performance of the overall-closed loop system. On the other hand, the larger $(n_i, m_i)$ the larger is the communication and computation efforts of the controllers, and hence the larger the sampling time of the controllers. An example of applied model decomposition is given in Section 3.4.

### 3.2.2 Decentralized optimal control problems

In order to exploit submodels (3.7) for formulating local finite-horizon optimal control problems that lead to an overall closed-loop stable DMPC system, let
3.2. Decentralized model predictive control setup

the following assumptions be satisfied.

**Assumption 1.** Matrix $A$ has all its eigenvalues inside the unit circle.

Assumption 1 restricts the strategy and stability results of DMPC to processes that are open-loop asymptotically stable, leaving to the controller the mere role of optimizing the performance of the closed-loop system.

**Assumption 2.** Matrix $A_i$ has all its eigenvalues inside the unit circle, $\forall i \in \{1, \ldots, M\}$.

Assumption 2 restricts the degrees of freedom in choosing the decentralized models. Note that if $A_i = A$ for all $i \in \{1, \ldots, M\}$ is the only choice satisfying Assumption 2, then no decentralized MPC can be formulated within this framework. For all $i \in \{1, \ldots, M\}$ consider the following infinite-horizon constrained optimal control problems:

$$V_i(x(t)) = \min_{\{u_k\}_{k=0}^{\infty}} \sum_{k=0}^{\infty} (x_k^i)' Q_i x_k^i + (u_k^i)' R_i u_k^i = \min_{u_0^i} (x_1^i)' P_i x_1^i + x_1^i(t)' Q_i x_1^i(t) + (u_0^i)' R_i u_0^i \quad (3.9a)$$

subject to:

$$x_1^i = A_i x_1^i(t) + B_i u_0^i \quad (3.9c)$$
$$x_0^i = W_i' x(t) = x^i(t) \quad (3.9d)$$
$$u_{\text{min}}^i \leq u_0^i \leq u_{\text{max}}^i \quad (3.9e)$$
$$u_k^i = 0, \forall k \geq 1 \quad (3.9f)$$

where $P_i = P_i' \succeq 0$ is the solution of the Lyapunov equation

$$A_i' P_i A_i - P_i = -Q_i \quad (3.10)$$

such that $x^i P_i x = \sum_{k=0}^{\infty} (A_i^k x)' Q_i (A_i^k x)$ and that exists by virtue of Assumption 2, $Q_i = W_i' Q W_i$ and $R_i = Z_i' R Z_i$. Problem (3.9) corresponds to a finite-horizon constrained problem with control horizon $N_u = 1$ and linear stable prediction model. Note that only the local state vector $x^i(t)$ is needed to solve Problem (3.9).

At time $t$, each controller MPC $\#i$ measures the state $x^i(t)$ (usually corresponding to local and neighboring states), solves problem (3.9) and obtains the optimizer

$$u_0^i = [u_0^{i1}, \ldots, u_0^{ii}, \ldots, u_0^{imi}]' \in \mathbb{R}^{m_i} \quad (3.11)$$
3. Decentralized Networked Model Predictive Control

In the simplified case $M = m$ and $I_{\text{si}} = i$, only the $i$-th sample of $u_{0}^{\text{ii}}$

$$u_{i}(t) = u_{0}^{\text{ii}}$$

(3.12)

will determine the $i$-th component $u_{i}(t)$ of the input vector actually commanded to the process at time $t$. The inputs $u_{0}^{\text{ij}}$, $j \neq i$, $j \in I_{\text{si}}$ to the neighbors are discarded, their only role is to provide a better prediction of the state trajectories $x_{k}^{i}$, and therefore a possibly better performance of the overall closed-loop system.

The collection of the optimal inputs of all the $M$ MPC controllers,

$$u(t) = [u_{0}^{\text{11}} \ldots u_{0}^{\text{ii}} \ldots u_{0}^{\text{mm}}]^{T}$$

(3.13)

is the actual input commanded to process (3.1), whose components $u_{i}(t)$ are broadcasted at time $t+1$ to the interested local controllers $j$ such that $i \in I_{\text{uj}}$.

The optimizations (3.9) are repeated at time $t+1$ based on the new states $x^{i}(t+1) = W_{i}'x(t+1)$, $\forall i \in \{1, \ldots, M\}$, according to the usual receding horizon control paradigm in MPC. The degree of coupling between the DMPC controllers is dictated by the choice of the decoupling matrices $(W_{i}, Z_{i})$. Clearly, the larger the number of interactions captured by the submodels, the more complex the formulation (and, in general, better the solution) of the optimization problems (3.9) and hence the computations performed by each control agent.

### 3.3 Convergence properties

As mentioned in this Chapter’s introduction, one of the major issues in decentralized MPC is to ensure the stability of the overall closed-loop system. The non-triviality of this issue is due to the fact that the prediction of the state trajectory made by MPC $\#i$ about state $x^{i}(t)$ is in general not accurate, because of partial state and input information, the mismatch between $u^{\text{ij}}$ (desired by controller MPC $\#i$) and $u^{\text{jj}}$ (computed and applied to the process by controller MPC $\#j$) and the incomplete knowledge of dynamics.

The following theorem summarizes the closed-loop convergence results of the proposed DMPC scheme using the cost function $V(x(t)) \triangleq \sum_{i=1}^{M} V_{i}(W_{i}'x(t))$ as a Lyapunov function for the overall system.
Theorem 4. Let Assumptions 1, 2 hold and define $P_i$ as in (3.10), $\forall i \in \{1, \ldots, M\}$. Define

$$
\Delta u^i(t) \triangleq u(t) - Z_i u_0^i(t), \quad \Delta x^i(t) \triangleq (I - W_i W_i') x(t)
$$

$$
\Delta A^i \triangleq (I - W_i W_i') A, \quad \Delta B^i \triangleq B - W_i W_i' B Z_i
$$

and

$$
\Delta Y^i(x(t)) \triangleq W_i W_i' (A \Delta x^i(t) + B Z_i(\Delta u^i(t)) + \Delta A^i x(t) + \Delta B^i u(t)
$$

$$
\Delta S^i(x(t)) \triangleq (2(A_i W_i' x(t) + B_i u_0^i(t)))' + \Delta Y^i(x(t))' W_i') P_i W_i' \Delta Y^i(x(t))
$$

If at least one of the conditions

(i) $x' \hat{Q} x - \sum_{i=1}^{M} \Delta S^i(x) \geq 0$ \hspace{1cm} (3.15a)

(ii) $x' \hat{Q} x - \sum_{i=1}^{M} \Delta S^i(x) + \sum_{i=1}^{M} u_0^i(x)' R_i u_0^i(x) \geq \alpha x' x$ \hspace{1cm} (3.15b)

is satisfied for some scalar $\alpha > 0$ and $\forall x \in \mathbb{R}^n$, where $\hat{Q} = \left( \sum_{i=1}^{M} W_i W_i' Q W_i W_i' \right)$ and $R_i = Z_i' R Z_i$, then the decentralized MPC scheme defined in (3.9)–(3.13) in closed loop with (3.1) is globally asymptotically stable.

Proof. Since $x^i(t) = W_i' x(t)$, by exploiting (3.10) at time $t$ the optimal cost $V_i(x(t))$ of subproblem (3.9) can be rewritten as

$$
V_i(x(t)) = (W_i' x(t))'(W_i' Q W_i) W_i' x(t) + u_0^i(t)' Z_i' R Z_i u_0^i(t) + (A_i W_i' x(t) + B_i u_0^i(t))' P_i (A_i W_i' x(t) + B_i u_0^i(t))
$$

(3.16)

where $P_i$ is defined as in (3.10). As the input $u_0^i = 0$ satisfies the constraints $u_{\text{min}} \leq u_0^i \leq u_{\text{max}}$, by (3.10) the optimal cost at time $t + 1$ satisfies the following inequality

$$
V_i(x(t + 1)) \leq x(t + 1)' W_i (W_i' Q W_i) W_i' x(t + 1) + x(t + 1)' W_i A_i P_i A_i W_i' x(t + 1)
$$

$$
= (W_i' x(t + 1))'(A_i' P_i A_i + W_i' Q W_i) W_i' x(t + 1)
$$

$$
= x(t + 1)' W_i P_i W_i' x(t + 1)
$$

(3.17)
By rewriting
\[ x(t+1) = Ax(t) + Bu(t) = (A + W_i W_i'A)(x(t) \pm W_i W_i'x(t)) + (B \pm W_i W_i' B Z_i Z_i')(u(t) \pm Z_i u_0^i(t)) \]
\[ = W_i(A_i W_i'x(t) + B_i u_0^i(t)) + \Delta Y^i(x(t)) \] (3.18)
where \( \pm[\cdot] \) means that the same quantity \([\cdot] \) is added and subtracted, from (3.17) and recalling (3.5) we obtain
\[ V_i(x(t+1)) \leq V_i(x(t)) - x'(t) W_i W_i' Q W_i W_i' x(t) - u_0^i(t)' Z_i'^{-1} R Z_i u_0^i(t) + \Delta S^i(x(t)) \]
(3.19)

Consider first condition (3.15a). By positive definiteness of \( R \) and full column rank of all matrices \( Z_i \) it follows that \( Z_i' R Z_i > 0 \) and hence that the function \( V(x(t)) \triangleq \sum_{i=1}^M V_i(W_i'x(t)) \) is non-increasing. Since \( V(x(t)) \geq 0, \forall t \geq 0 \), it follows that there exists \( \lim_{t \to \infty} V(x(t)) = \lim_{t \to \infty} V(x(t+1)) \).

Hence, by (3.19) it also follows that
\[ \lim_{t \to \infty} x'(t) \bar{Q} x(t) - \sum_{i=1}^M \Delta S^i(x(t)) + \sum_{i=1}^M u_0^i(x(t))' R_i u_0^i(x(t)) = 0 \]

Because of (3.15a), it follows that \( \lim_{t \to \infty} \sum_{i=1}^M u_0^i(x(t))' R_i u_0^i(x(t)) = 0 \), and by positive definiteness of \( Z_i' R Z_i \), that \( \lim_{t \to \infty} u_0^i(x(t)) = 0 \), and hence that \( \lim_{t \to \infty} u_0^{ii}(x(t)) = 0, \forall i \in \{1, \ldots, M\} \), which in turn implies \( \lim_{t \to \infty} u(t) = 0 \). As by Assumption 1 the open-loop process (3.1) is linear and asymptotically stable, it is also input-to-state stable [50], and hence \( \lim_{t \to \infty} x(t) = 0 \).

The same follows under condition (3.15b), as \( \lim_{t \to \infty} \alpha x'(t) x(t) = 0 \) and hence \( \lim_{t \to \infty} x(t) = 0 \).

Theorem 4 provides two alternative conditions for verifying closed-loop stability. Condition (3.15a) amounts to testing that the cumulated effect of
3.3. Convergence properties

model mismatch $\sum_{i=1}^{M} \Delta S_i(x)$ is dominated by the global decreasing rate $x'Qx$, therefore advising the designer to choose a weight $Q$ large enough to dominate the influence of the prediction error due to unmodeled dynamics. Condition (3.15b) attempts to exploit also the nonnegative term $\sum_{i=1}^{M} u_{0i}^{*}(x)'R_{i}u_{0i}^{*}(x)$ to dominate model mismatch, provided that the slightly more stringent condition \( \geq \alpha x'x \) instead of \( \geq 0 \) is satisfied for some $\alpha > 0$.

3.3.1 Stability tests

By using the explicit MPC results of [13], each optimizer function $u_{0i}^{*} : \mathbb{R}^n \mapsto \mathbb{R}^{m_i}$, $i = 1, \ldots, M$, can be expressed as a piecewise affine function of $x$:

$$u_{0i}^{*}(x) = F_{ij}x + G_{ij} \text{ if } H_{ij}x \leq K_{ij}, \; j = 1, \ldots, n_{ri} \quad (3.20)$$

Hence, both condition (3.15a) and condition (3.15b) are a composition of quadratic and piecewise affine functions, so that global stability can be tested through linear matrix inequality relaxations [51] (cf. the approach of [45]).

As $u_{min} < 0 < u_{max}$, there exists a ball around the origin $x = 0$ contained in one of the regions, say $\{x \in \mathbb{R}^n : H_{i1}x \leq K_{i1}\}$, such that $G_{i1} = 0$. Therefore, around the origin both (3.15a) and (3.15b) become a quadratic form $x'(\sum_{i=1}^{M} E_{i})x$ of $x$, where matrices $E_{i}$ can be easily derived from (3.14) and (3.15b). Hence, local stability of (3.9)–(3.13) in closed loop with (3.1) can be simply tested by checking the positive semidefiniteness of the square $n \times n$ matrix $\sum_{i=1}^{M} E_{i}$. Note that, depending on the degree of decentralization, in order to satisfy the sufficient stability criterion one may need to set $Q \succ 0$ in order to dominate the unmodeled dynamics arising from the terms $\Delta S^i$.

3.3.2 Open-loop unstable subsystems

In the absence of input constraints, Assumptions 1, 2 can be removed to extend the previous DMPC scheme to the case where $(A, B)$ is not an asymptotically stable system, although stabilizable.

**Theorem 5.** Let the pairs $(A_i, B_i)$ be stabilizable, $\forall i \in \{1, \ldots, M\}$. Let
Problem (3.9) be replaced by

\[ V_i(x(t)) = \min_{\{u^i_k\}_{k=0}^{\infty}} \sum_{k=0}^{\infty} (x^i_k)^	op Q_i x^i_k + (u^i_k)^	op R_i u^i_k = \]  
\[ = \min_{u^i_0} (x^i_0)^	op P_i x^i_0 + x^i(t)^	op Q_i x^i(t) + (u^i_0)^	op R_i u^i_0 \]  
\[ \text{s.t.} \quad x^i_0 = A_i x^i(t) + B_i u^i_0 \]  
\[ x^i(t) = W_i^\top x(t) = x^i(t) \]  
\[ u^i_k = K_{LQ_i} x^i_k, \quad \forall k \geq 1 \]  
\[ \tag{3.21} \]

where \( P_i = P_i' \succeq 0 \) is the solution of the Riccati equation

\[ Q_i + K_{LQ_i}^\top R_i K_{LQ_i} + (A_i + B_i K_{LQ_i})' P_i (A_i + B_i K_{LQ_i}) = P_i \]  
\[ \tag{3.22} \]

\( K_{LQ_i} = (Z_i^\top R Z_i + B_i' P_i B_i)^{-1} B_i' P_i A_i \) is the corresponding local LQR feedback, \( Q_i = W_i^\top Q W_i \) and \( R_i = Z_i^\top R Z_i \). Let \( \Delta Y^i(x(t)) \) and let \( \Delta S^i(x(t)) \) be defined as in (3.14), in which \( P_i \) is defined as in (3.22). If (3.15a) is satisfied, or (3.15b) is satisfied for some scalar \( \alpha > 0 \), then the decentralized MPC scheme defined in (3.21), (3.13) in closed-loop with (3.1) is globally asymptotically stable.

\[ \textbf{Proof.} \] By recalling that \( x^i(t) = W^\top x(t) \) and exploiting (3.22), at time \( t \) the optimal cost \( V_i(x(t)) \) of subproblem (3.21) can be rewritten as

\[ V_i(x(t)) = (W_i^\top x(t))' (W_i^\top Q W_i) W_i^\top x(t) + (A_i W_i^\top x(t) + B_i u^i_0(t))' P_i (A_i W_i^\top x(t) + B_i u^i_0(t)) + u^i_0(t)' Z_i^\top R Z_i u^i_0(t) \]  
\[ \tag{3.23} \]

Now choose \( u^i_0(t + 1) = K_{LQ_i} (W_i^\top x(t + 1)) \), with \( P_i \), as solution of the Riccati equation (3.22), and \( K_{LQ_i} \) as the correspond local LQR feedback. By feasibility of Problem (3.21) at time \( t \), the optimal cost at time \( t + 1 \) satisfies the following equality.
By rewriting
\[
x(t + 1) = Ax(t) + Bu(t) = W_i(A_i W_i'x(t) + B_i u_0^i(t)) + \Delta Y^i(x(t))
\]
as in (3.18), from (3.24) and recalling (3.5) we obtain

\[
V_i(x(t + 1)) = (W_i(A_i W_i'x(t) + B_i u_0^i(t)) + \\
\Delta Y^i(x(t)))' P_i W_i'(W_i(A_i W_i'x(t) + B_i u_0^i(t)) + \Delta Y^i(x(t))) = \\
(W_i(A_i W_i'x(t) + B_i u_0^i(t)) + \Delta Y^i(x(t)))'
\]

where \(\Delta S^i(x(t))\) is defined as in (3.14c). By (3.23), we obtain

\[
V_i(x(t + 1)) \leq V_i(x(t)) - x'(t) W_i W_i' Q W_i x(t) - u_0^i(t) Z_i R Z_i u_0^i(t) + \Delta S^i(x(t))
\]

for which the same reasoning as in the proof of Theorem 4 can be repeated. \(\square\)

### 3.3.3 Decentralized MPC under arbitrary packet loss

So far we assumed that the communication model among neighboring MPC controllers is faultless, so that each MPC agent successfully receives the information about the states of its corresponding submodel. However, one of the main issues in networked control systems is the unreliability of communication channels, especially in wireless automation systems, which may result in data packet dropout. In this Section we prove a sufficient condition for
ensuring convergence of the DMPC closed-loop in the case packets containing measurements are lost for an arbitrary but upper-bounded number $N$ of consecutive time steps. The underlying operating assumption is that if the actual number of lost packets exceeds the given $N$, the decentralized controllers are turned off and $u = 0$ is applied persistently, so that a number of packet drops larger than $N$ is not considered. The results shown here are based on formulation (3.9) and rely on the open-loop asymptotic stability Assumptions 1 and 2.

Setting $u(t) = 0$ to an open-loop stable system is a natural backup choice when no state feedback is available because of a communication blackout. Because of (3.9f), setting $u^i(t) = 0$ also amounts to applying the prosecution of the most recent available optimal control sequence, a practice often used in MPC in case of failures of the QP solver. Different backup options may be considered, such as solving (3.9) by replacing $x^i(t)$ with an estimate obtained through model (3.7) and the available measurements, for instance by applying distributed Kalman filtering techniques [101]. Of course whether one or the other approach is better strongly depends on the amount of model mismatch introduced by the decentralized modeling.

The next theorem provides conditions for asymptotic closed-loop stability of decentralized MPC under packet loss, generalizing the result of Theorem 4.

**Theorem 6.** Let $N$ be a positive integer such that no more than $N$ consecutive steps of channel transmission blackout can occur. Assume $u(t) = 0$ is applied when no packet is received. Let Assumptions 1, 2 hold and $\forall i \in \{1, \ldots, M\}$ define $P_i$ as in (3.10), $\Delta u^i(t)$, $\Delta x^i(t)$, $\Delta A^i$, $\Delta B^i$ as in (3.14a), $\Delta Y^i(x(t))$ as in (3.14b), let $\xi_i(x) \triangleq A_i W^i x + B_i u_0^i(x)$, and for $j = 1, \ldots, N$ let

$$
\Delta S^i_j(x) \triangleq [2(A_i W^i x + B_i u_0^i(x))' W^i + \Delta Y^i(x)'] (A^{j-1})' W^i P_i W^i A^{j-1} \Delta Y^i(x) 
$$

(3.25)

If at least one of the conditions

(i) $x' Q x + \sum_{i=1}^{M} \xi_i(x)' P_{ij} \xi_i(x) - \Delta S^i_j(x) \geq 0$  \hspace{1cm} (3.26)

(ii) $x' Q x - \alpha x' x + \sum_{i=1}^{M} \xi_i(x)' P_{ij} \xi_i(x) - \Delta S^i_j(x) + u_0^i(x)' Z_i^j R Z_i u_0^i(x) \geq 0$ \hspace{1cm} (3.27)
is satisfied for some scalar \( \alpha > 0 \) and \( \forall x \in \mathbb{R}^n, \forall j \in \{1, \ldots, N\} \), where \( \bar{Q} = (\sum_{i=1}^{M} W_i W_i' Q W_i W_i') \) and \( \bar{P}_{ij} = P_i - W_i (A^{j-1})' W_i P_i W_i' A^{j-1} W_i \), then the decentralized MPC scheme defined in (3.9)–(3.13) in closed loop with (3.1) is globally asymptotically stable under packet loss.

**Proof.** Let \( \{t_k\}_{k=0}^\infty \) be the sequence of sampling steps at which packet information is received, and let \( j_k = t_{k+1} - t_k \) the corresponding number of consecutive packet drops, \( 1 \leq j_k \leq N \). We want to examine the difference \( V_i(x(t_{k+1})) - V_i(x(t_k)) \), where \( V_i(x(t)) \) is the optimal cost of subproblem (3.9) at time \( t \). As the backup input \( u(t_k + h) = 0 \) is applied from time \( t_k \) to \( t_{k+1} - 1 \) \((h = 0, \ldots, j_k - 1)\), we have

\[
x(t_{k+1}) = A^{j_k-1} (Ax(t_k) + Bu(t_k)) = A^{j_k-1} (\Delta Y'(x(t_k)) + W_i \xi(x(t_k)))
\]

Since \( x^i(t_{k+1}) = W'_i x(t_{k+1}) \), at time \( t_{k+1} \) the optimal cost \( V_i(x(t_{k+1})) \) of subproblem (3.9) can be rewritten as

\[
V_i(x(t_{k+1})) = x(t_{k+1})' W'_i W_i' Q W_i W'_i x(t_{k+1}) + [A_i W'_i x(t_{k+1}) + B_i u^*_i(t_{k+1})]' \cdot P_i (A_i W'_i x(t_{k+1}) + B_i u^*_i(t_{k+1})] + u^*_i(t_{k+1})' Z'_i R Z_i u^*_i(t_{k+1})
\]

where \( P_i \) is defined as in (3.10) and is such that \( (x_0')' P_i x_0 = \sum_{k=0}^{\infty} (x_k')' W'_i Q W_i x_k \) with \( x_{k+1} = A_i x_k \). Hence, considering that \( u^*_i(t_{k+1}) = 0 \) is a feasible suboptimal choice for problem (3.9), we obtain the following inequality

\[
V_i(x(t_{k+1})) \leq x'(t_{k+1}) W_i P_i W'_i x(t_{k+1}) \\
\leq (A^{j_k-1}[\Delta Y'(x(t_k)) + W_i \xi(x(t_k)))'] W_i P_i \cdot W'_i A^{j_k-1}[\Delta Y'(x(t_k))] + W_i \xi(x(t_k)))
\]

\[
= \Delta Y(x(t_k))'(A^{j_k-1})' W_i P_i W'_i A^{j_k-1} \Delta Y(x(t_k)) + 2\xi(x(t_k))' W'_i (A^{j_k-1})' W_i P_i W_i' A^{j_k-1} \Delta Y(x(t_k)) + \xi(x(t_k))' W'_i (A^{j_k-1})' W_i P_i W_i' A^{j_k-1} W_i \xi(x(t_k))
\]

\[
= \Delta S_{j_k}(x(t_k)) + \xi(x(t_k))' W'_i (A^{j_k-1})' W_i P_i W_i' A^{j_k-1} W_i \xi(x(t_k))
\]

Since \( V_i(x(t_k)) = x'(t_k) W'_i W_i' Q W_i W'_i x(t_k) + \xi(x(t_k))' P_i \xi(x(t_k)) + \),
\[ u^*_0(t_k)'Z'_i RZ_i u^*_0(t_k) \]

we get

\[ V_i(x(t_{k+1})) - V_i(x_i(t_k)) \]

\[ \leq \Delta S_{jk}^i(x(t_k)) + \xi(x(t_k))'W_i'(A^{jk-1})'W_i P_i W_i' A^{jk-1} W_i \xi(x(t_k)) + \]

\[ - x(t_k)'W_i(W_i'QW_i)W_i'x(t_k) + \xi(x(t_k))'P_i \xi(x(t_k)) + u^*_0(t_k)'Z'_i RZ_i u^*_0(t_k) \]

\[ \leq \Delta S_{jk}^i(x(t_k)) - u^*_0(t_k)'Z'_i RZ_i u^*_0(t_k) + x(t_k)'W_i(W_i'QW_i)W_i'x(t_k) + \]

\[ - \xi(x(t_k))'(P_i - W_i'(A^{jk-1})'W_i P_i W_i' A^{jk-1} W_i) \xi(x(t_k)) \]

Let \( V(x(t)) \equiv \sum_{i=1}^{M} V_i(W_i'x(t)) \). If (3.26) holds, then by (3.19) it follows that \( V(x(t_k)) \) is a decreasing function of \( k \) lower-bounded by zero, and therefore converges as \( k \to \infty \), which proves \( \lim_{k \to \infty} V(x(t_{k+1})) - V(x(t_k)) = 0 \). This in turn implies by (3.19) that

\[ \lim_{k \to \infty} u^*_0(t_k)'Z'_i RZ_i u^*_0(t_k) = 0 \]

As \( R \succ 0 \) and \( Z_i \) are full-column-rank matrices, it follows that \( Z'_i RZ_i \succ 0 \) and hence that \( \lim_{k \to \infty} u(t_k) = 0 \). If (3.27) holds, then in a similar way it is immediate to see that \( \lim_{k \to \infty} x(t_k) = 0 \) which again implies \( \lim_{k \to \infty} u(t_k) = 0 \), as around the origin \( u(t_k) \) is a linear function of \( x(t_k) \) (corresponding to the unconstrained solution of problem (3.9)). Since in the presence of packet drop \( u(t) = 0 \), the input sequence \( \{\ldots, 0, 0, u(t_k), 0, \ldots, 0, u(t_{k+1}), 0, \ldots, 0, u(t_{k+2}), \ldots\} \) is actually applied to the process, and clearly \( \lim_{t \to \infty} u(t) = 0 \). As asymptotically stable linear systems are also input-to-state stable [50], it immediately follows that \( \lim_{t \to \infty} x(t) = 0 \).

Note that \( V(x(t)) \) is not a common Lyapunov function for the switched system under consideration since \( V(x(t_k)) \leq V(x(t_k - 1)) \) for \( t_k - 1 \notin \{t_k\}^{\infty}_{k=0} \) does not hold for the general case. However, Assumptions 1, 2, implies \( V(x(t)) \) to be decreasing when the plant evolve unforced, i.e. \( u(t) = 0 \), condition that applies for \( t \notin \{t_k\}^{\infty}_{k=0} \). Therefore \( V(x(t)) \) is upperbounded by

\[ V(x(t)) = \begin{cases} V(x(t)) & t \in \{t_k\}^{\infty}_{k=0} \\ V(x(t_k)) & t_k \leq t < t_{k+1} \end{cases} \quad (3.28) \]

that is a common Lyapunov function for each mode of the switched system. It is trivial to verify that \( \tilde{V}(x(t)) \geq V(x(t)) \), \( \forall t \in \mathbb{Z}_{0^+} \) and that \( \tilde{V}(x(t)) \) converge to 0 by virtue of measured instants. \( \square \)
Note again that around the origin the conditions in (3.26) and (3.27) become a quadratic form, so local stability of (3.9)–(3.13) in closed loop with (3.1) under packet loss can be easily tested for any arbitrary fixed N. Note also that conditions (3.26) and (3.27) are a generalization of (3.15b), as for $j = 1$ (no drop) matrix $P_i - W_i'(A_j^{-1})'W_iW_i'A_j^{-1}W_i = P_i - P_i = 0$.

### 3.3.4 Extension to set-point tracking

Consider the following discrete-time linear global process model

\[
\begin{align*}
  z(t+1) &= A z(t) + B v(t) + F_d(t) \\
  h(t) &= C z(t)
\end{align*}
\]  

(3.29)

where $z \in \mathbb{R}^n$ is the state vector, $v \in \mathbb{R}^m$ is the input vector, $h \in \mathbb{R}^p$ is the output vector, $F_d \in \mathbb{R}^d$ is a vector of measured disturbances. Let $A$ satisfy Assumption 1 and assume $F_d$ is constant. The considered set-point tracking problem is that of $h(t)$ tracking a given reference value $r \in \mathbb{R}^p$, despite the presence of $F_d$. In order to recast the problem as a regulation problem, assume steady-state vectors $z_r \in \mathbb{R}^n$ and $v_r \in \mathbb{R}^m$ exist solving the static problem

\[
\begin{align*}
  (I - A)z_r &= B v_r + F_d \\
  r &= C z_r
\end{align*}
\]  

(3.30)

and let $x = z - z_r$, $y = h - r$, and $u = v - v_r$. Input constraints $v_{\text{min}} \leq v \leq v_{\text{max}}$ are mapped into constraints $v_{\text{min}} - v_r \leq u \leq v_{\text{max}} - v_r$. Note that in case $v_r \notin [v_{\text{min}}, v_{\text{max}}]$, perfect tracking under constraints is not possible, and an alternative is to set

\[
\begin{bmatrix}
  z_r' \\
  v_r
\end{bmatrix}
= \arg\min \quad \| \begin{bmatrix}
  I - A & -B \\
  C
\end{bmatrix} \begin{bmatrix}
  z_r' \\
  v_r
\end{bmatrix} - \begin{bmatrix}
  F_d' \\
  0
\end{bmatrix} \| \\
\text{s.t.} \quad v_{\text{min}} \leq v_r \leq v_{\text{max}}
\]

**Proposition 1.** Under the global coordinate transformation (3.30), the process (3.29) under the decentralized MPC law (3.9)–(3.13) is such that $h(t)$ converges asymptotically to the set-point $r$, either under the assumption of Theorem 4 or, in the presence of packet drops, of Theorem 6.

Note that problem (3.30) is solved in a centralized way. Defining local coordinate transformations $v_{ir}$, $z_{ir}$ based on submodels (3.7) would not lead,
in general, to offset-free tracking, due to the mismatch between global and local models. This is a general observation one needs to take into account in decentralized tracking. Note also that both \( v_r \) and \( z_r \) depend on \( F_d \) as well as \( r \), so problem (3.30) should be solved each time the value of \( F_d \) or \( r \) change and retransmitted to each controller.

3.4 Decentralized temperature control in a railcar

In this section we test the proposed DMPC approach for decentralized control of the temperature in different passenger areas in a railcar. The system is schematized in Figure 3.2. Each passenger area has its own heater and air conditioner but its thermal dynamics interacts with surrounding areas (neighboring passenger areas, external environment, antechambers) directly or through windows, walls and doors. Passenger areas are composed by a table and the associated four seats. Temperature sensors are located in each four-seat area, in each antechamber, and along the corridor. The goal of the controller is to adjust each passenger area to its own temperature set-point to maximize passenger comfort. Temperature sensors may be wired or wireless, in the latter case we assume that information packets may be dropped, because of very low power transmission, simplified transmission protocols, no acknowledgement and retransmission, and because of interferences from passengers’ electronic equipment.

Let \( 2N_a \) be the number of four-seat areas (\( N_a = 8 \) in Figure 3.2), \( N_a \) the number of corridor partitions, and \( 2 \) the number of antechambers. Under the assumption of perfectly mixed fluids in each \( j \)-th volume, \( j = 1, \ldots, n \) where \( n = 3N_a + 2 \), the heat transmission equations by conduction lead to the linear model

\[
\frac{dT_j(\tau)}{d\tau} = \sum_{i=0}^{n} Q_{ij}(\tau) + Q_{uj}, \quad Q_{ij}(\tau) = \frac{S_{ij}K_{ij}(T_i(\tau) - T_j(\tau))}{C_jL_{ij}} \tag{3.31}
\]

\( j = 1, \ldots, n \), where \( T_j(\tau) \) is the temperature of volume \( \#j \) at time \( \tau \in \mathbb{R} \), \( T_0(\tau) \) is the ambient temperature outside the railcar, \( Q_{ij}(\tau) \) is heat flow due to the temperature difference \( T_i(\tau) - T_j(\tau) \) with the neighboring volume \( \#i \), \( S_{ij} \) is the contact surface area, \( Q_{uj} \) is the heat flow of heater \( \#j \), \( K_{ij} \) is the
3.4. Decentralized temperature control in a railcar

Figure 3.2: Physical structure of the railcar

thermal coefficient that depends on the materials, \( C_j = K_j^c V_j \) is the (material dependent) heat capacity coefficient \( K_j^c \) times the fluid volume \( V_j \), and \( L_{ij} \) is the thickness of the separator between the two volumes \#i and \#j. We assume that \( Q_{ij}(\tau) = 0 \) for all volumes \( i, j \) that are not adjacent, \( \forall \tau \in \mathbb{R} \). Hence, the process can be modeled as a linear time-invariant continuous-time system with state vector \( z \in \mathbb{R}^{3N_a+2} \) and input vector \( v \in \mathbb{R}^{2N_a} \)

\[
\begin{align*}
\dot{z}(\tau) &= A_c z(\tau) + B_c v(\tau) + F T_0(\tau) \\
h(\tau) &= C z(\tau)
\end{align*}
\] (3.32)

where \( F \in \mathbb{R}^n \) is a constant matrix, \( T_0(\tau) \) is treated as a piecewise constant measured disturbance, and \( C \in \mathbb{R}^{p \times n} \) is such that \( h \in \mathbb{R}^p \) contains the components of \( z \) corresponding to the temperatures of the passenger seat areas, \( p = 2N_a \). Since we assume that the thermal dynamics are relatively slow compared to the sampling time \( T_s \) of the decentralized controller we are going to synthesize, we use first-order Euler approximation to discretize dynamics (3.32) without introducing excessive errors:

\[
\begin{align*}
z(t + 1) &= A z(t) + B v(t) + F_d T_0(t) \\
h(t) &= C z(t)
\end{align*}
\] (3.33)

where \( A = I + A_c T_s \), \( B = B_c T_s \), and \( F_d = F T_s \). We assume that \( A \) is asymptotically stable, as an inheritance of the asymptotic stability of matrix \( A_c \).
In order to track generic temperature references $r(t)$, we adopt the coordinate shift defined by (3.30). The next step is to decentralize the resulting global model. The particular topology of the railcar suggests a decomposition of model (3.1) as the cascade of four-seat areas. There are two kinds of four-seat areas, namely (i) the ones next to the antechambers, and (ii) the remaining ones. Besides interacting with the external environment, the areas of type (i) interact with another four-seat-area, an antechamber, and a section of the corridor, while the areas of type (ii) only with the four-seat areas at both sides and the corresponding section of the corridor. Note that the decentralized models overlap, as they share common states and inputs. The decoupling matrices $Z_i$ are chosen so that in each subsystem only the first component of the computed optimal input vector is actually applied to the process.

As a result, each submodel has 5 states and 2 or 3 inputs, depending whether it describes a seat area of type (i) or (ii), which is considerably simpler than the centralized model (3.1) with 26 states and 16 inputs. We apply the DMPC approach (3.9) with

$$Q = \begin{bmatrix} 200I_{16} & 0 \\ 0 & 2I_{10} \end{bmatrix}, \quad R = 10^5 I_{16}, \quad v_{\text{max}} = -v_{\text{min}} = 0.03 \text{ W}, \quad T_s = 540 \text{ s}$$

(3.34)

where $v_{\text{min}}$ is the lower bound on the heat flow subtracted by the air-conditioners, and $v_{\text{max}}$ is the maximum heating power of the heaters (with a slight abuse of notation we denoted by $v_{\text{min}}, v_{\text{max}}$ the entries of the corresponding lower and upper bound vectors of $R^{16}$). Note that the first sixteen diagonal elements of matrix $Q$ correspond to the temperatures of the four-seat areas. It is easy to check that with the parameters in (3.34) condition (3.15a) for local stability is satisfied. For comparison, a centralized MPC approach (3.3) with the same weights, horizon, and sampling time as in (3.34) is also designed. The associated QP problem has 16 optimization variables and 32 constraints, while the complexity of each DMPC controller is either 2 (or 3) variables and 4 (or 6) constraints. The DMPC approach is in fact largely scalable: for longer railcars the complexity of the DMPC controllers remains the same, while the complexity of the centralized MPC problem grows (more than linearly) with the increased model size. Note also that, even if a centralized computation is taken, the DMPC approach can be immediately parallelized.
3.4. Decentralized temperature control in a railcar

3.4.1 Simulation results

We investigate different simulation outcomes depending on four ingredients: i) type of controller (centralized / decentralized), ii) packet-loss probability, iii) change in reference values, iv) changes of external temperature (acting as a measured disturbance). Figure 3.3 shows the external temperature and reference scenarios, respectively, used in all simulations.

In order to compare closed-loop performances in different simulation scenarios, define the following performance index

\[
J = \sum_{i=1}^{N_{\text{sim}}} e_z'(t)Qe_z(t) + e_v'(t)Re_v(t)
\]  

(3.35)

where \( e_z(t) = h(t) - r(t) \), \( e_v(t) = v(t) - v_r \) and \( N_{\text{sim}} = 160 \) (one day) is the total number of simulation steps.

The initial condition is 17°C for all seat-area temperatures, except for the antechamber, which is 15°C. Note that the steady-state value of antechamber temperatures is not relevant for the posed control goals. The closed-loop trajectories of centralized MPC feedback vs. decentralized MPC with no packet-loss are shown in Figure 3.4 (we only show the first state and input for clarity). In both cases the temperatures of the four-seat areas converge...
Figure 3.4: Comparison between centralized MPC (dashed lines) and decentralized MPC (continuous lines): output $h_1$ (upper plots) and input $v_1$ (lower plots). Gray areas denote packet drop intervals.

to the set-point asymptotically. Figure 3.5 shows the temperature vector $h(t)$ tracking the time-varying reference $r(t)$ in the absence of packet-loss, where the coordinate transformation (3.30) is repeated after each set-point and external temperature change.

To simulate packet loss, we assume that the probability of losing a packet depends on the state of a Markov chain with $N$ states (see Figure 3.6). We parameterize with the probability parameter $p$, $0 \leq p \leq 1$ the probabilities associated with the Markov chain: the Markov chain is in the $j$-th state if $j - 1$ consecutive packets have been lost. The probability of losing a further packet is $1 - p$ in every state of the chain, except for the $(N + 1)$-th state.
3.4. Decentralized temperature control in a railcar

Figure 3.5: Decentralized MPC results. Upper plots: output variables $h$ (continuous lines) and references $r$ (dashed lines). Lower plots: command inputs $v$. Gray areas denote packet drop intervals

where no packet can be lost any more.

Let $\pi$ be the stationary probability vector of the Markov chain of Figure 3.6, obtained through the one-step probability matrix $P$, obtained by solving

\[
\begin{cases}
\pi' = \pi' P \\
\sum_{i=1}^{N} \pi_i = 1
\end{cases}
\Rightarrow P = \begin{bmatrix}
p & 1-p & 0 & \cdots & 0 \\
p & 0 & 1-p & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
p & 0 & 0 & \cdots & 1-p \\
1 & 0 & 0 & \cdots & 0
\end{bmatrix}
\]

Recalling the meaning of the nodes of the Markov chain, the steady-state probability $\pi_i$ of the $i$-th state is the probability of losing consecutively exactly $i-1$ packets. The packet-loss discrete probability is shown in Figure 3.7 when $N = 10$ maximum consecutive packet-losses are possible and $p = 0.7$. 
Figure 3.7 highlights the exponential decrease of the stationary probabilities as a function of consecutive packets lost. Such a probability model is confirmed by the experimental results on relative frequencies of packet failure burst length observed in [116]. Note that our model assumes that the probability of losing a packet is null after $N$ packets, hence satisfying the assumption of an upper-bound on the number of consecutive drops (as mentioned earlier, we can assume for instance that if $k > N$ consecutive packets are lost, the control loops are shut down). The simulation results obtained with $p = 0.5$ are shown in Figure 3.4.1 and Figure 3.4.1. In case of packet loss, we also compare the performance of centralized vs. decentralized MPC. Note that in case packet loss occurs also on the communication channel between the point computing the coordinate shift and the decentralized controllers, the last received coordinate shift is kept. The stability condition (3.26) of Theorem 6 was tested and proved satisfied for values of $j$ up to 160.

Figure 3.8 shows that the performance index $J$ defined in Eq. (3.35) increases as the packet-loss probability grows, implying performance to deteriorate due to the conservativeness of the backup control action $u = 0$ (that is, $v = v_r$). The results of Figure 3.8 are averaged over 10 simulations per probability sample. As a general consideration, centralized MPC dominates over the decentralized, although for certain values of $p$ the average performance of decentralized MPC is slightly better, probably due to the particular packet

Figure 3.6: Markov chain modeling packet-loss probability: the network is in state $i$ if the last $i - 1$ packets have been lost
3.4. Decentralized temperature control in a railcar

Figure 3.7: Markov chain packet-loss probability with $N = 10$ and $p = 0.7$.

Figure 3.8: Performance indices of Centralized MPC (dashed line) and Decentralized MPC (solid line)

loss sequences that have realized. However, the loss of performance due to decentralization, with regard to the present example, is largely negligible.

The simulations were run on a MacBook Air 1.86 GHz running Matlab R2008a under OS X 10.5.6 and the Hybrid Toolbox for Matlab [10]. The average CPU time for solving the centralized QP problem associated with (3.3) is 6.0 ms (11.9 ms in the worst case). For the decentralized case, the average CPU time for solving the QP problem associated with (3.9) is 3.3 ms (7.4 ms in the worst case). Although the decrease of CPU time is only a few milliseconds, we remark that for increasing $N_a$ the complexity of DMPC remains constant, while the complexity of centralized MPC would grow with $N_a$. To
quantify this aspect consider that, if one thinks to the explicit form of the
MPC controllers [13], the number of regions of the centralized MPC is upper
bounded by $3^{16}$, while in decentralized case by $3^2$ for submodels with two
inputs and by $3^3$ for submodels with three inputs, respectively.

Note that in all simulations the reference vectors $v_r$, $z_r$ are computed
globally by a higher-level control layer in the hierarchical setting. In this
example the complexity of such a static calculation is negligible with respect
to solving the QP problems. Moreover, the communication burden is also
negligible, as new reference vectors are transmitted individually to each MPC
agent only when set-point and disturbances change.

3.5 Conclusions

In this Chapter we proposed an approach for controlling large-scale processes
subject to input constraints using the cooperation of multiple decentralized
model predictive controllers. Each controller is based on a submodel of the
overall process, and different submodels may share common states and inputs,
to possibly decrease modeling errors in case of dynamical coupling, and
to increase the level of cooperativeness of the controllers. The possible loss
of global optimal performance is compensated by the gain in controller scalability, reconfigurability, and maintenance. Open research issues remain to be
further explored to extend the proposed DMPC scheme, such as: systematic
ways to decompose the model into local submodels, when this is not obvious
from the physics of the process, determining the optimal model decomposition
(i.e., the best achievable closed-loop performance) for a given channel
capacity and computer power available to the control agents; hierarchical
MPC schemes, in which the DMPC controllers are supervised by a central-
ized (possibly hybrid) MPC controller running at a slower sampling frequency
to enforce global constraints.

The two latter points are addressed in Chapters 5 and 4, respectively.
Chapter 4
Decentralized Hierarchical Multi-rate Control

4.1 Introduction

Modern technological innovations increased the inner complexity of human implantations but the most impressive advancement relies in their size. The challenge of control of such giant infrastructures, usually referred as large-scale systems (LSS), rose since the last decades and its importance is becoming more and more crucial as the massive innovations in the field make this phenomena more evident. Classical control techniques, which lies in the common presupposition of centrality, rapidly fail with the growth in size of even linear system subject to constraints on input and output variables. Moreover the need of scalability for maintenance, fault robustness and quick fixing make control engineer to inevitably look for different time scale models among the sample plant, following the top-down analysis approach. The result is a hierarchical control structure, with decentralized lower level, which proves much more flexible and scalable. Furthermore, it is capable, in opposition with the centralized schemes, to exploit the computation potentials of modern control systems (parallel computing).

Following this trend, first ideas date back to the seventies (see the book of [112]) and was recently developed as integration of the framework of model predictive control (MPC) (see e.g. [1, 27, 36, 49, 55, 61, 66, 109] and the survey [11]), giving rise to decentralized MPC, DMPC. While pure DMPC dispenses form solving a large-scale optimization problem at each sampling time, performance is typically degraded with respect to corresponding MPC (increasingly with coupling degree of local dynamics and objectives) and global constraints are often hard to impose without time-consuming iterative decision processes. Hierarchical control is a good compromise: lower-level local controllers take care of stabilization tasks based on simplified local dynamical
models, and are orchestrated by a upper-level control layer that maximizes global performance and enforces global constraints [99]. A similar concept was also adopted in the reference governor (RG) literature, where the complexity of MPC was mitigated by separating the stabilization problem from the constraint fulfillment problem [12, 42, 43, 46, 56]. Moreover, [6, 7] also imposed constraints on the variations of the references, giving a RG control strategy which in the case was implemented by means of MPC.

Extending this latter results, in this Chapter we propose two hierarchical multi-rate control approaches that exploit the idea of manipulating reference signals to enforce constraints. We assume that the open-loop process is stabilized by a linear (possibly decentralized) controller with sampling time $T_L$ without taking care of the constraints. Its reference signals are generated by a higher control, i.e. upper control layer, level running at slower pace. Such control layer is based on a plant model which is resampled at the controller frequency, so as to neglect excessively fast dynamics and grasp the major behavior. Secondly, this Chapter proposes to investigate the opportunity of use a decentralized controller even at the high control layer to obtain the immediate benefits in terms of scalability and complexity but also to decouple the low level references update sample times. To this end, we define a single centralized controller running at frequency $T^H = NT^L$, which is based on the full resampled model and that feeds each lower layer controller with the same update frequency. Then, a decentralization of that controller is obtained and, consequently, the lower level controllers are divided in set of influence, each controlled by an upper level. Exploiting the mutual independence, the upper layer controllers act at different frequencies adapting the feedback pace to the real needing of the plant portion they are in charge of. The price to pay to guarantee independence is a certain degree of conservatism, introduced by modeling interactions among subsystems in a worst-case manner.

The Chapter is organized as follows. Section 4.2 defines the hierarchical control problem and the computation of maximum reference rates to guarantee tracking while enforcing constraints. Section 4.3 extends the control architecture to have a decentralized supervisor while retaining constraint handling and stability. Section 4.4 demonstrates the effectiveness of both approaches showing a comparison. Conclusions are drawn in Section 4.5.
4.2 Centralized Hierarchical Approach

4.2.1 Problem Setup

Consider the two-level hierarchical control architecture depicted in Figure 4.1 where the open-loop plant is stabilized by the lower control layer (LCL) which run at a sampling frequency \( \frac{1}{T_L} \) and does not account for constraints.

![Figure 4.1: General two layers hierarchical control scheme.](image)

The LCL reference signals are commanded by the supervising upper level controller (ULC), running at sampling frequency \( \frac{1}{T_H} \), so as to enforce some linear constraints on the plant inputs and outputs while optimizing a performance criterion.

Let \( N = \frac{T_H}{T_L} \) be the positive integer ratio between ULC and LCL sample times, which makes the reference for the LCL be piecewise-constant with period \( N \) steps.

Let the lower level closed-loop system be described by the following linear time-invariant discrete-time model

\[
\begin{align*}
    x(t+1) &= \bar{A}x(t) + \bar{B}u(t) \\
    y(t) &= \bar{C}x(t) + \bar{D}u(t) \\
    u(t) &= Fx(t) + Er(t)
\end{align*}
\] (4.1)

where \( x(t) \in \mathbb{R}^n \), \( y(t) \in \mathbb{R}^{nr} \), \( u(t) \in \mathbb{R}^{nu} \), and \( r(t) \in \mathbb{R}^{nr} \) is the reference signal. We assume that \( F \) is an asymptotically stabilizing gain, which does not account for constraint satisfaction. If in \( F \) some structure holds, i.e. \( F_{i,j} = 0 \) for some \( i, j \), then the \( j \)-th state value is not required for computation of \( i \)-th
input, which result in a decentralized feedback (see e.g. [31] for LMI-based synthesis of decentralized linear controllers). We also assume that a gain $E \in \mathbb{R}^{n_u \times n_r}$ exists such that the DC-gain from $r$ to $y$ is the identity,

$$E = ((\bar{C} + \bar{D}F)(I - \bar{A} - \bar{B}F)^{-1}\bar{B} + \bar{D})^{-1}$$

The closed-loop system (4.1) can be rewritten as

$$\begin{align*}
    x(t+1) &= Ax(t) + Br(t) \\
    y(t) &= Cx(t) + Dr(t)
\end{align*}$$

where $A = \bar{A} + \bar{B}F$, $B = \bar{B}E$, $C = \bar{C} + \bar{D}F$, $D = \bar{D}E$.

The higher-level controller processes the desired reference vector $p(t) \in \mathbb{R}^{n_r}$ and commands the piecewise constant vector of references $r(t)$

$$r(t) = r^k, \ t = kN, \ldots, (k + 1)N - 1, \ k = 0, 1 \ldots$$

to the lower-level controller $u(t) = Fx(t) + Er(t)$ in a way that the state vector $x_i(t)$ and the reference $r_i(t)$ are kept within the admissible polytope

$$\mathcal{X} = \{ [\bar{r}] \in \mathbb{R}^{n_x + n_r} : H_x x + H_r r \leq K \}$$

where $H_x \in \mathbb{R}^{q \times n_x}$, $H_r \in \mathbb{R}^{q \times n_r}$, $K \in \mathbb{R}^q$. Note that (4.5) covers the case of input, state, and output constraints, and constraints on the local tracking error $y - r$.

The main goal of the centralized approach is to determine simultaneously the ratio $N$ and restrictions on both $r^k$ and $\Delta r^k = r^k - r^{k-1}$ to enforce $[\bar{r}] \in \mathcal{X}$. To this end, let the reference vector $r(t)$ be constrained within the tightened set

$$\mathcal{R} = \{ r \in \mathbb{R}^{n_r} : (H_x G + H_r)r \leq K - \Delta K \}$$

where $G = (I - A)^{-1}$ is the reference-to-state DC gain of (4.1), $\Delta K \in \mathbb{R}^q$, $\Delta K > 0$ component-wise. Moreover, assume that updates of set points $r(t)$ always let the tracking error $\Delta x(t) \triangleq x(t) - Gr(t)$ is kept within the set

$$\mathcal{E} = \{ \Delta x \in \mathbb{R}^{n_x} : H_x \Delta x \leq \Delta K \}$$

When a reference variation occurs, the system dynamic induces output oscillations with consequent potential constraint violation. To prevent this,
4.2. Centralized Hierarchical Approach

reference range is restricted so as to establish a margin for the tracking error in which oscillations may safely occur. Thus, vector $\Delta K$ is a tuning knob of the proposed approach: the smaller the components of $\Delta K$, the larger is the set $\mathcal{R}$ of admissible set points $r^k$, but the smaller will be the admissible reference increments $\Delta r^k$ to maintain tracking errors within $\mathcal{E}$.

Let $x_r \in \mathbb{R}^n$ be the steady-state state corresponding to a reference signal $r \in \mathbb{R}^n$, $x_r = Ax_r + Br$, $x_r = Gr$, and define the shift of coordinates $\Delta x = x - x_r$. Then, (4.3) can be rewritten as

$$
\begin{align*}
\Delta x(t+1) &= A\Delta x(t) \\
e(t) &= C\Delta x(t)
\end{align*}
$$

(4.8)

Let $\Omega(0) \subseteq \mathbb{R}^n$ be the maximum admissible output set (MOAS) [44] for the closed-loop system (4.8) under the constraint $e(t) \in \mathcal{E}$

$$
\begin{align*}
\Omega(0) &= \{ \Delta x \in \mathbb{R}^n : (H x G + H r) A^k \Delta x \leq \Delta K, \forall k \geq 0 \} \\
&= \{ x \in \mathbb{R}^n : H_0 \Delta x \leq K_0 \}
\end{align*}
$$

(4.9)

where $H_0 \in \mathbb{R}^{n_0 \times n}$ and $K_0 \in \mathbb{R}^{n_0}$, and define the reference-dependent invariant set

$$
\Omega(r) = \{ x \in \mathbb{R}^n : H_0(x - Gr) \leq K_0 \}
$$

(4.10)

Lemma 1. Let $x(0) \in \Omega(r)$ and $r(t) \equiv r \in \mathcal{R}, \forall t \geq 0$. Then $x(t) \in \mathcal{X}$, $\forall t \geq 0$.

Proof. By (4.9), $x(0) \in \Omega(r)$ implies that $H x_0 \Delta x(t) = H x_0 x(t) - H x G r \leq \Delta K$, $\forall t \geq 0$. By (4.6) it follows that $H x_0 x(t) - H x G r \leq \Delta K \leq K - H x G r - H r r$ which in turns implies $H x_0 x(t) + H r r \leq K$, $\forall t \geq 0$.

In the following, we derive sufficient conditions to let the system state to always lie in invariant sets at times $t = kT^H$, $\forall k \in \mathbb{N}^+$ while enforcing (4.5). To this end we bound the instantaneous element-wise maximum reference variation.

1As $\Delta K_0 > 0$ and $A$ is asymptotically stable, $\Omega(0)$ is generated by a finite number of inequalities, as proved in [44]. We assume, without loss of generality, that $(H_0, K_0)$ are a minimal hyperplane representation of $\Omega(0)$. 
4. Decentralized Hierarchical Multi-rate Control

4.2.2 Computation of maximum reference rates

For given $N$, consider the problem of determining the initial state $x(0) \in \Omega(r^1)$ and the minimum reference variation $\Delta r(N) = ||r^2 - r^1||_\infty$ between two reference values $r^1, r^2 \in \mathcal{R}$ such that the state $x(N)$ is outside the invariant set $\Omega(r^2)$:

$$\Delta r(N) = \inf_{r^1, r^2, x(0)} ||r^2 - r^1||_\infty$$  \hspace{2cm} (4.11a)

s.t. $r^1, r^2 \in \mathcal{R}$  \hspace{2cm} (4.11b)

$$x(0) \in \Omega(r^1)$$  \hspace{2cm} (4.11c)

$$x(t + 1) = Ax(t) + Br^2 \quad t = 0, \ldots, N - 1$$  \hspace{2cm} (4.11d)

$$x(N) \notin \Omega(r^2)$$  \hspace{2cm} (4.11e)

Constraint (4.11e) makes optimization problem (4.11) nonconvex. By means of convenient manipulations (see [6] for details) the problem can be recast in the following Mixed Integer Linear Program (MILP)

$$\Delta r(N) = \min_{[x'(r')](r'), \delta', \epsilon'} \epsilon$$  \hspace{2cm} (4.12a)

s.t. $\epsilon \geq \pm((r^2)^j - (r^1)^j)$, $j = 1, \ldots, n_r$  \hspace{2cm} (4.12b)

$$(H_xG + H_r)r^1 \leq K - \Delta K$$  \hspace{2cm} (4.12c)

$$(H_xG + H_r)r^2 \leq K - \Delta K$$  \hspace{2cm} (4.12d)

$$H_0(x - Gr^1) \leq K_0$$  \hspace{2cm} (4.12e)

$$H_0(A^N x + R_G r^2) - K_0^i \leq M_+^i(1 - \delta^i)$$  \hspace{2cm} (4.12f)

$$H_0(A^N x + R_G r^2) - K_0^i \geq (M_-^i - \epsilon)\delta^i + \sigma$$  \hspace{2cm} (4.12g)

$$\sum_{i=0}^{n_0} \delta^i \leq n_0 - 1$$  \hspace{2cm} (4.12h)

$$\delta^i = \{0, 1\}, \quad i = 1, \ldots, n_0$$  \hspace{2cm} (4.12i)

where $R_G = \left(\sum_{h=0}^{N-1} A^h B\right) - G$. The quantity $\Delta r(N)$ in (4.12) is the smallest element-wise change of reference vector such that, even if the closed-loop system (4.3) starts from within an invariant set $\Omega(r^k)$, the state vector may lands outside a new invariant set $\Omega(r^{k+1})$ after $N$ steps. Or, in other words, for all reference changes $||r^k - r^{k-1}||_\infty \leq \Delta r(N) - \sigma$, $\forall \sigma > 0$, the closed-loop
system (4.3) is such that, starting from an invariant set \( \Omega(r^k) \), the state vector always arrives into a new invariant set \( \Omega(r^{k+1}) \) after \( N \) steps.

Note that, because of the constraint \( r^1, r^2 \in \mathcal{R} \), problem (4.12) is likely to become infeasible for large \( N \), that is, any feasible perturbation of the set-point keeps the state within the invariant set \( \Omega(r^2) \) after \( N \) steps. However, such \( N \) not necessarily exist for small \( \Delta K \) constraints (4.12c) and (4.12d) are very restrictive. The following lemma shows a monotonicity property of \( \Delta r(N) \) with respect to \( N \), for those values \( N \in \mathbb{N} \) for which \( \Delta r(N) \) is defined.

**Lemma 2.** Let \( \Delta r(N) \) be defined by the optimization problem (4.12). Then for any \( N_1, N_2 \in \mathbb{N} \), \( N_1 < N_2 \), such that \( \Delta r(N_1) \), \( \Delta r(N_2) \) are defined it holds that

\[
\Delta r(N_1) \leq \Delta r(N_2) \tag{4.13}
\]

**Proof.** We first prove by contradiction that \( \Delta r(N) \leq \Delta r(N + 1) \), \( \forall N \in \mathbb{N} \) such that \( \Delta r(N + 1) \) is defined. Assume that \( N \in \mathbb{N} \) exists such that \( \Delta r(N + 1) < \Delta r(N) \). This implies that there exists a state \( x \) and two references \( r^1, r^2 \in \mathcal{R} \) such that \( \Delta r(N + 1) \leq \|r^1 - r^2\|_{\infty} < \Delta r(N) \), \( x \in \Omega(r^1) \), \( A^{N+1}x + \sum_{i=1}^{N} A^iBr^2 \not\in \Omega(r^2) \). Then, also \( A^Nx + \sum_{i=1}^{N-1} A^iBr^2 \not\in \Omega(r^2) \), otherwise, by invariance of \( \Omega(r^2) \), also \( A^{N+1}x + \sum_{i=1}^{N} A^iBr^2 \) would belong to \( \Omega(r^2) \).

Hence, the optimality of \( \Delta r(N) \) is violated, a contradiction. The monotonicity condition (4.13) for generic \( N_1, N_2 \) easily follows.

### 4.2.3 Centralized Hierarchical Controller

Assume that \( N \) has been fixed and that the upper control layer commands set-points \( r^k \) under the constraints

\[
\|r^k - r^{k-1}\|_{\infty} \leq \Delta r(N) - \sigma, \forall k = 0, 1, \ldots \tag{4.14a}
\]
\[
r^k \in \mathcal{R}, \forall k = -1, 0, 1, \ldots \tag{4.14b}
\]

feeding the lower control layer as in (4.4).

**Theorem 7.** Let \( F \) be a lower-level feedback gain such that \( \bar{A} + \bar{B}F \) is a strictly Schur matrix\(^2\), and assume that matrix \( E \) in (4.2) exists. Assume a

---

\(^2\)All matrix eigenvalues are within the unit circle
vector $r^{-1} \in \mathcal{R}$ exists such that the initial state \( x(0) \in \Omega(r^{-1}) \). Let the upper-level controller change the set-points \( r^k \) according to the constraints (4.14), in which \( \Delta r(N) \) is the solution of (4.12) and \( \sigma > 0 \) is arbitrary small. Then the linear system \( (\bar{A}, \bar{B}, \bar{C}, \bar{D}) \) satisfies the constraints \( \tilde{x} \in \mathcal{X} \) for all \( t \geq 0 \). If in addition \( \lim_{t \to \infty} r(t) = r \in \mathcal{R} \) then \( \lim_{t \to \infty} y(t) = r \).

**Proof.** Because of (4.14), \( x(kN) \in \Omega(r^k), \forall k \geq 0 \). By Lemma 1, it follows that \( \tilde{x} \in \mathcal{X}, \forall t = kN, \ldots, k(N+1) - 1, \forall k = 0, 1, \ldots \), that is \( \begin{bmatrix} x(t) \\ r(t) \end{bmatrix} \in \mathcal{X}, \forall t \geq 0 \). To prove convergence of \( y(t) \) to \( r \) when \( \lim_{t \to \infty} r(t) = r \), similarly to (4.8) define \( \Delta x(t) = x(t) - G_x r \) and rewrite (4.3) as

\[
\begin{cases}
\Delta x(t+1) = A\Delta x(t) + B(r(t) - r) \\
e(t) = C\Delta x(t) + D(r(t) - r)
\end{cases}
\] (4.15)

As (4.15) is an asymptotically stable linear system it is also input-to-state stable [50], and hence it immediately follows that \( \lim_{t \to \infty} \Delta x(t) = 0 \), which in turn implies that \( \lim_{t \to \infty} y(t) - r = 0 \).

Theorem 7 shows that any upper-level reference generation strategy satisfying constraints (4.14) guarantees the fulfillment of output constraints and asymptotic convergence to constant set-points. The MILP (4.12) provides the supremum \( \Delta r(N) \) of the reference variations \( \|r^k - r^{k-1}\|_{\infty} \) that the higher-level controller can apply for a given ratio \( N = T_H/T_L \) between sampling times. It is worth to investigate the relation between \( \Delta r(N) \) and \( N \) further. In fact, the design of the higher control layer could be addressed from a different point of view: given a desired \( \Delta r \), determine the minimum \( N \) such that \( \Delta r < \Delta r(N) \). In practical applications \( N \) is restricted to a range \([N_{\min}, N_{\max}]\) of values: the upper layer is executed at a slower pace than the lower layer (\( N_{\min} \) not too small), but at the same time the upper layer should be reactive enough to adjust set-points (\( N_{\max} \) not too large). Hence, it is worth to solve the MILP (4.12) only within the restricted range \( N \in [N_{\min}, N_{\max}] \) to characterize \( \Delta r(N) \) that, by Lemma 2, we know increases with \( N \). In particular, it is of interest the ratio \( R(N) = \frac{\Delta r(N)}{N} \) which characterizes the maximum speed of change of the reference signal. In fact, the larger \( N \), the larger is the supremum of the variations \( \Delta r \) that the supervisor can issue per sample time, and the more frequently such variation is allowed,
the greater the variation over time. Another issue related to tuning of the upper control layer is the choice of $\Delta K$: from one hand a larger $\Delta K$ tightens the range of admissible references $\mathcal{R}$, but on the other hand it enlarges the size of the invariant set $\Omega(r)$, and therefore augments the achievable $\Delta r(N)$. There is therefore a tradeoff: the designer must choose between constraints on admissible reference signals ($\mathcal{R}$) and constraints on reference speed of variation ($R(N)$).

4.3 Decentralized Hierarchical Approach

4.3.1 Problem setup

In this Section we extend the approach presented in Section 4.2 to handle a fully decentralized hierarchical control structure. To that end, both upper and lower control layers are decentralized, i.e. based on subsystems derived from plant model (4.1), and retain the same tasks as in the previous Section. In fact the LCL is in charge of stabilization without constraint handling while the UCL should manipulate the desired reference vector $p(t)$ to feed the LCL so that constraints (4.5) hold. The resulting control structure is depicted in Figure 4.2, where the LCL runs at the same frequency of the plant. Contrarily to the centralized approach, where the UCL is an atomic block, here it is composed of a set of $m$ independent controllers, running at lower sampling frequencies $f^H_i = \frac{1}{T^H_i}$, $i = 1, \ldots, m$ which feed their respective underlying controllers in LCL. Denote by $\mathcal{I}_i^x$, $\mathcal{I}_i^u$, $\mathcal{I}_i^r$ the sets of state, input, and output indices, respectively, corresponding to the $i$-th subsystem, and by $n_i^x$, $n_i^u$, $n_i^r$ their cardinalities, $i = 1, \ldots, m$. Note that in general such subsets of indices may overlap, in case different subsystems share common states, inputs, or outputs. In this section, we assume that in $F$ and $E$ the following decentralized structure

\begin{align}
F_{h,j} &= 0, \ \forall h \in \mathcal{I}_i^u \text{ and } j \notin \mathcal{I}_i^x, \ \forall i = 1, \ldots, m \quad \text{(4.16a)} \\
E_{h,j} &= 0, \ \forall h \in \mathcal{I}_i^u \text{ and } j \notin \mathcal{I}_i^r, \ \forall i = 1, \ldots, m \quad \text{(4.16b)}
\end{align}

hold, which can be imposed computing gains $F$ and $E$ by several methods, see e.g. [8, 31, 112, 113]. We point out that constraints (4.16) ensure the sole
need of local feedback, i.e. in order to compute $u_i$ only elements in the $i$-th subsystem are required.

Despite the fact that only local state feedback is used, the global model (4.3) may not be block diagonal, because of possible dynamical coupling through matrices $\bar{A}, \bar{B}$. Hence, system (4.3) can be written as the collection of $m$ dynamical systems

$$\Sigma_i : x_i(t+1) = A_i x_i(t) + B_i r_i(t) + d_i(t) \quad (4.17)$$

where $d_i(t)$ captures the unmodeled dynamics due to the neglected dynamical couplings.

**Assumption 3.** Matrix $A_i$ has spectral radius within the unit circle for $i = 1, \ldots, m$.

Assumption 3 states that the nominal local closed-loop system ($d_i(t) = 0$) is asymptotically stable, which is a condition that have to be imposed in the synthesis of $F$.

Given a matrix $M$, let $M_{I,J}$ be the submatrix of matrix $M$ obtained by collecting the row indices in $I$ and the column indices in $J$, and by $M_I$ the submatrix collecting all the rows indexed by $I$. A similar notation is used
4.3. Decentralized Hierarchical Approach

for subvectors \( v_l \) of a given vector \( v \). Then we set \( A_i = A_{T_x^i \cap J} \), \( B_i = B_{T_x^i \cap J} \) in (4.17).

The neglected dynamics are modeled as follows. Let \( J_x^i = \{1, \ldots, n_x\} \setminus T_x^i \), \( J_r^i = \{1, \ldots, n_r\} \setminus T_r^i \), and set \( \tilde{x}_i = x_{J_x^i} \), \( \tilde{r}_i = r_{J_r^i} \), \( \tilde{A}_i = A_{T_x^i \cap J} \), \( \tilde{B}_i = B_{T_x^i \cap J} \), where \((\tilde{A}_i, \tilde{B}_i)\) capture the influence of unmodeled states \( \tilde{x}_i \) and reference signals \( \tilde{r}_i \) on \( \Sigma_i \).

The upper-level controllers are assumed to act independently, so their sampling intervals may differ. Define the following ratios \( N_i = T_i^U / T_L \), where \( 1/T_L \) is the sampling frequency of the lower-level decentralized controller, \( N_i \in \mathbb{N} \), \( i = 1 \ldots m \).

The idea of this extension is the following. Assuming that each controller \( i \) is capable of enforcing constraints (4.5) concerning state variables and references in \( T_x^i \) and \( T_r^i \), respectively, then the whole set of constraints holds. Moreover, this allows to exactly compute worst case influence that the rest of the system has on \( \Sigma_i \) since bounds on unknown state variables and references are imposed (and enforced) by (4.5).

The goal of the \( i \)-th upper-level controller is to generate a piecewise-constant reference \( r_i(t) \)

\[
\begin{align*}
\tilde{r}_i(t) &= r_i^k, \quad t = kN_i, \ldots, (k+1)N_i - 1, \quad k = 0, 1 \ldots
\end{align*}
\]

(4.18)

in order to keep the state vector \( x_i(t) \) and the reference \( r_i(t) \) within the admissible polytope

\[
\mathcal{X}_i = \{ [\tilde{x}_i^T, \tilde{r}_i^T] : H_{x}^i x_i + H_{r}^i r_i \leq K^i \}
\]

(4.19)

where \( H_{x}^i \in \mathbb{R}^{q_i \times n_x^i} \), \( H_{r}^i \in \mathbb{R}^{q_i \times n_r^i} \), \( K^i \in \mathbb{R}^q \). Note that (4.18) and (4.19) correspond to (4.4) and (4.5), respectively but concern the sole entries in \( \Sigma_i \).

Let \( A_i^0 \in \mathbb{R}^{n_x^i \times n_x^i} \) be the matrix obtained by collecting from \( A \) the rows indexed by \( T_x^i \) and then zeroing the columns indexed by \( T_r^i \), and similarly \( B_i^0 \in \mathbb{R}^{n_r^i \times n_r} \) is the matrix obtained by collecting from \( B \) the rows indexed by \( T_x^i \) and then zeroing the columns indexed by \( T_r^i \). Hence, \( A_{T_x^i} x = A_i x_i + B_i \tilde{r}_i + A_i^0 x + B_i^0 r \).

Moreover, let \( \mathcal{X} = \{ [\tilde{x}, \tilde{r}] : H_x x + H_r r \leq K \} \) where \( H_x, H_r, K \) define the set of states and references such that their subvectors \( x_j, r_j \) belong to \( \mathcal{X}_j \), for \( j = 1, \ldots, m \), which is a bounded set as \( \mathcal{X}_j \) are polytopes. Under the assumption that \( [x_i(t), r_i(t)] \in \mathcal{X}_i \) holds for all \( t \geq 0 \), \( i = 1, \ldots, m \), and therefore
that \( \begin{bmatrix} x(t) \\ r(t) \end{bmatrix} \in \mathcal{X}, \forall t \geq 0 \), the local “disturbance” \( d_i(t) \) modeling the effect of the other subsystems on \( \Sigma_i \), belongs to polytopic set \( \mathcal{D}_i \) to be determined.

We now follow the same steps as in Section 4.2 to determine of the ratio \( N_i \) and restrictions on the reference values \( r_i(t) \) and their variations \( \Delta r_i = r_i - r_{i-1} \) generated by the upper-level controllers simultaneously so that \( \begin{bmatrix} x_i(t) \\ r_i(t) \end{bmatrix} \in \mathcal{X}_i \), for each \( i = 1, \ldots, m \), for any \( d_i \in \mathcal{D}_i \).

Let the reference vector \( r_i(t) \) be constrained within the tightened set
\[
\mathcal{R}_i = \{ r_i \in \mathbb{R}^{n_i} : (H_i^I G_i + H_i^I r_i) r_i \leq K_i - \Delta K_i \} \tag{4.20}
\]
where \( G_i = (I - A_i)^{-1} B_i \) is the reference-to-state DC gain of (4.17) for \( \Sigma_i \). Vector \( \Delta K_i \in \mathbb{R}^{q_i} \) is selected to have positive components. We assume that set-points \( r_i(t) \) are changed in a way that the tracking error \( \Delta x_i(t) \triangleq x_i(t) - G_i r_i(t) \) is kept within the set
\[
\mathcal{E}_i = \{ \Delta x_i \in \mathbb{R}^{n_i} : H_i^I \Delta x_i \leq \Delta K_i \} \tag{4.21}
\]
that, expressed in the new coordinates, is
\[
\mathcal{E}^x_i = \{ \begin{bmatrix} x_i \\ r_i \end{bmatrix} \in \mathbb{R}^{n_i + n_i} : H_i^I x_i - H_i^I G_i r_i \leq \Delta K_i \} \tag{4.22}
\]
As \( x_i^r = G_i r_i = A_i x_i^r + B_i r_i \), in the new coordinates \( \Delta x_i = x_i - x_i^r \) equation (4.17) becomes
\[
\Delta x_i(t + 1) = A_i \Delta x_i(t) + d_i(t) \tag{4.23}
\]
where \( d_i \) belong to the polytope
\[
\mathcal{D}_i = \{ d_i \in \mathbb{R}^{n_i} : \exists x \in \mathbb{R}^{n_x}, r \in \mathbb{R}^{n_r} \mid d_i = A_i^0 x + B_i^0 r, [x] \in \mathcal{X} \cap \mathcal{E}_i \} \tag{4.24}
\]
Note that by construction \( \mathcal{D}_i \) comprehends any influence which may perturb \( \Sigma_i \) by means of dynamical couplings from feasible state configuration and admissible references. Recalling discussion of equation (4.7), the set of vectors \( \Delta K_i \), \( i = 1, \ldots, m \) is a tuning knob of the decentralized approach. In the present decentralized setting, \( \Delta K_i \) has also a strong influence on the conservativeness of the approach, due to its role played in the definition of the disturbance set \( D_i \) (the smaller \( \Delta K_i \), the smaller the set \( D_i \)).
4.3. Decentralized Hierarchical Approach

The presence of couplings, modeled as unknown but bounded disturbances, implies an extension of (4.9). Let \( \Omega_i(0) \subseteq \mathbb{R}^{n_i} \) be the maximum output admissible robustly invariant set (MOARS, [41]) for system (4.23) under the constraint \( \Delta x_i(t) \in \mathcal{E}_i \)

\[
\Omega_i(0) = \{ \Delta x_i \in \mathbb{R}^{n_i} : H^i_x(A^i_k \Delta x_i + \sum_{j=0}^{k-1} (A^i_k)^j d_i(k-1-j)) \leq \Delta K^i, \]
\[
d_i(k-1-j) \in D_i, \ \forall k \geq 0 \}
\] (4.25)

Let \( (H^i_0, K^i_0) \) be a minimal hyperplane representation of \( \Omega_i(0) \),

\[
\Omega_i(0) = \{ \Delta x_i \in \mathbb{R}^{n_i} : H^i_0 \Delta x_i \leq K^i_0 \}
\]

and let \( n^i_0 \) be the number of such hyperplanes, that under Assumption 3 exists and is finite for each \( i = 1, \ldots, m \), see [41]. Then

\[
\Omega_i(r_i) = \{ x_i \in \mathbb{R}^{n_i} : x_i - G_i r_i \in \Omega_i(0) \}
\] (4.26)

The following lemma extends Lemma 1 to cover the case of polytopic uncertainty.

**Lemma 3.** For all subsystems \( i = 1, \ldots, m \), let \( x_i(0) \in \Omega_i(r_i) \) and \( r_i(t) \equiv r_i \in \mathcal{R}_i, \ \forall t \geq 0 \). Then \( x_i(t) \in \mathcal{X}_i, \ \forall t \geq 0 \).

**Proof.** By (4.25), \( x_i(0) \in \Omega_i(r_i) \) implies that \( H^i_x \Delta x(t) = H^i_x x(t) - H^i_x G_i r_i \leq \Delta K^i, \ \forall d_i(j) \in D_i, \ j < t, \ \text{and} \ \forall t \geq 0 \). By (4.6) it follows that \( H^i_x x(t) - H^i_x G_i r_i \leq \Delta K^i \leq K_i - H^i_x G_i r_i - H^i_x r_i \), which in turn implies \( H^i_x x(t) + H^i_x r_i \leq K_i, \ \forall t \geq 0 \).

Lemma 3 allows to derive the relation between the reference variations and the sample times ratio for each of the \( m \) control loops of Figure 4.2 while accounting for mutual influence.

### 4.3.2 Reference rate constraints

Assume that the integer \( N_i = T^H_i / T^L_i \) of the \( i \)-th control hierarchy is given. Consider the following problem: determine the initial state \( x_i(0) \in \Omega_i(r^1_i) \) and the minimum reference variation \( \Delta r_i(N_i) = r^2_i - r^1_i \) between two reference
values $r^1_i, r^2_i \in \mathcal{R}_i$ such that the state $x_i(N_i)$ is outside the invariant set $\Omega_i(r^2_i)$ for some disturbance sequence $D_{N_i} = \{d_i(t)\}_{t=1}^{N_i-1}$, with $d_i(t) \in \mathcal{D}_i$, $\forall t \in \{1, \ldots, N_i - 1\}$:

\[
\Delta r_i(N_i) = \inf_{r^1_i, r^2_i, x_i(0), D_{N_i}} ||r^2_i - r^1_i||_{\infty}
\]

\hspace{1cm} (4.27a)

\[
\text{s.t.} \quad r^1_i, r^2_i \in \mathcal{R}_i 
\]

\hspace{1cm} (4.27b)

\[
x_i(0) \in \Omega_i(r^1_i)
\]

\hspace{1cm} (4.27c)

\[
x_i(t + 1) = A_i x_i(t) + B_i r^2_i + d_i(t)
\]

\hspace{1cm} (4.27d)

\[
d_i(t) \in \mathcal{D}_i, \ t = 0, 1, \ldots, N_i - 1
\]

\hspace{1cm} (4.27e)

\[
x(N_i) \not\in \Omega_i(r^2_i)
\]

\hspace{1cm} (4.27f)

Similarly to (4.11), each problem (4.27) is non convex and can be conveniently recast as MILP (see [7] for details):

\[
\Delta r_i(N_i) = \min_{[x, r^1_i, r^2_i, D_{N_i}, \delta, \epsilon]} \epsilon
\]

\hspace{1cm} (4.28a)

\[
\text{s.t.} \quad \epsilon \geq \pm((r^1_i)_j - (r^1_i)_j), \ j = 1, \ldots, n^i_r
\]

\hspace{1cm} (4.28b)

\[
(H_i G_i + H_i^2)r^1_i \leq K^i - \Delta K^i
\]

\hspace{1cm} (4.28c)

\[
(H_i G_i + H_i^2)r^2_i \leq K^i - \Delta K^i
\]

\hspace{1cm} (4.28d)

\[
H_0^i(x_i - G_i r^1_i) \leq K_0^i
\]

\hspace{1cm} (4.28e)

\[
(H_0^i)_h \left(A_i^{N_i} x_i + \sum_{j=0}^{N_i-1} A^j d_i(N_i - 1 - j) + R_G r^2_i - G_i r^2_i\right) +
\]

\[-(K_0^i)_h \leq (M_i^h)_h (1 - \delta^h)
\]

\hspace{1cm} (4.28f)

\[
(H_0^i)_h \left(A_i^{N_i} x_i + \sum_{j=0}^{N_i-1} A^j d_i(N_i - 1 - j) + R_G r^2_i - G_i r^2_i\right) +
\]

\[+ (K_0^i)_h \leq -((M_i^h)_h - \sigma)\delta^h - \sigma
\]

\hspace{1cm} (4.28g)

\[
d_i(t) \in \mathcal{D}_i, \ t = 0, 1, \ldots, N_i - 1
\]

\hspace{1cm} (4.28h)

\[
\sum_{w=0}^{n_i^h} \delta^w \leq n_i^h - 1
\]

\hspace{1cm} (4.28i)

\[
\delta^h = \{0, 1\}, \ h = 1, \ldots, n_0^i
\]

\hspace{1cm} (4.28j)
where $R_G = \left( \sum_{h=0}^{N_i-1} A_h B_i \right) - G_i$. The quantity $\Delta r_i(N_i)$ in (4.28) is the smallest change of reference components in $T_i^*$ (expressed in infinity norm) that can be applied to the closed-loop system (4.3) such that, if $x_i$ starts from an invariant set $\Omega_i(r_i^k)$, the $i$-th state vector lands outside the new invariant set $\Omega_i(r_i^{k+1})$ after $N_i$ steps. Similar considerations as for problem (4.12) applies. In fact the extension of Lemma 2 for the case of decentralized UCL follows.

**Lemma 4.** Let $\Delta r_i(N_i)$ be defined by the optimization problem (4.12). Then for any $N^1_i, N^2_i \in \mathbb{N}$, $N^1_i < N^2_i$, such that $\Delta r_i(N^1_i)$, $\Delta r_i(N^2_i)$ are defined it holds that

$$\Delta r_i(N^1_i) \leq \Delta r_i(N^2_i)$$  \hfill (4.29)

**Proof.** We first prove by contradiction that $\Delta r_i(N_i) \leq \Delta r_i(N_i + 1)$, $\forall N_i \in \mathbb{N}$ such that $\Delta r_i(N_i + 1)$ is defined. Assume that $N_i \in \mathbb{N}$ exists such that $\Delta r_i(N_i + 1) < \Delta r_i(N_i)$. This implies that there exists a state $x_i$, a disturbance sequence $D_{N_i}$, and two references $r_i^1, r_i^2 \in \mathcal{R}_i$ such that $\Delta r_i(N_i + 1) \leq ||r_i^1 - r_i^2||_{\infty} < \Delta r_i(N_i)$, $x_i \in \Omega_i(r_i^1)$, $A_i^{N_i+1} x_i + \sum_{h=0}^{N_i} A_h^k (d_i(N_i - h) + B_i r_i^2) \not\in \Omega_i(r_i^2)$. Then, also $A_i^{N_i} x_i + \sum_{h=0}^{N_i-1} A_h^k (d_i(N_i - 1 - h) + B_i r_i^2) \not\in \Omega_i(r_i^2)$, otherwise, by invariance of $\Omega_i(r_i^2)$, also $A_i^{N_i+1} x_i + \sum_{h=0}^{N_i} A_h^k (d_i(N_i - h) + B_i r_i^2)$ would belong to $\Omega_i(r_i^2)$. Hence, the optimality of $\Delta r_i(N_i)$ is violated, a contradiction. The monotonicity condition (4.29) for generic $N^1_i, N^2_i$ easily follows. \hfill $\Box$

### 4.3.3 Decentralized Hierarchical Controller

Assume that, for each subsystem $i$, $N_i$ has been fixed and that the upper control layer commands set-points $r_i^k$ under the constraints

$$\|r_i^k - r_i^{k-1}\|_{\infty} \leq \Delta r_i(N_i) - \sigma, \; \forall k = 0, 1, \ldots \; \hfill (4.30a)$$
$$r_i^k \in \mathcal{R}_i, \; \forall k = -1, 0, 1, \ldots \; \hfill (4.30b)$$

for some small $\sigma > 0$, to the lower control layer as in (4.4).

**Theorem 8.** Assume that $L$ is a decentralized asymptotically stabilizing linear gain, that Assumption 3 holds, and that a set of vectors $r_i^{-1} \in \mathcal{R}_i$ exists such that the initial states $x_i(0) \in \Omega_i(r_i^{-1})$, for $i = 1, \ldots, m$. Let all the upper-level controllers change the set-points $r_i^k$ according to the constraints (4.14), in which $\Delta r_i(N_i)$ is the solution of (4.12) and $\sigma > 0$ is an arbitrary small
number. Then the linear system (4.1) satisfies the constraints \[
\begin{bmatrix}
x(t) \\
r(t)
\end{bmatrix} \in \mathcal{X},
\]
for all \( t \geq 0 \). If in addition \( \lim_{t \to \infty} r(t) = r \in \mathcal{R} \) then \( \lim_{t \to \infty} x(t) = Gr \), \( G = (I - A)^{-1}B \).

Proof. Because of (4.14), \( x_i(kN_i) \in \Omega_i(r^k), \forall k \geq 0 \). By Lemma 3, it follows that \[
\begin{bmatrix}
x_i(t) \\
r_i(t)
\end{bmatrix} \in \mathcal{X}, \forall t = kN_i, \ldots, k(N_i + 1) - 1, \forall k = 0, 1, \ldots, i = 1, \ldots, m,
\]
that is \( \begin{bmatrix}
x(t) \\
r(t)
\end{bmatrix} \in \mathcal{X}, \forall t \geq 0 \). As \( \lim_{t \to \infty} r(t) = r \), similarly to (4.8) define \( \Delta x(t) = x(t) - Gr \) and rewrite (4.3) as
\[
\Delta x(t + 1) = A\Delta x(t) + B(r(t) - r) \tag{4.31}
\]
As (4.31) is an asymptotically stable linear system it is also input-to-state stable [50], and hence it immediately follows that \( \lim_{t \to \infty} \Delta x(t) = 0 \), which in turn implies that \( \lim_{t \to \infty} x(t) = Gr \).

As for the centralized case, one can solve problem (4.28) for several values of the sample times ratios, so as to obtain some design margin. Note that each hierarchical loop may run at proper frequency ratio since Theorem 8 states complete independence. By similar considerations as for Theorem 7 the ratio \( R_i(N_i) = \frac{\Delta r_i(N_i)}{N_i} \) defines the maximum speed of change for the generic controller \( i \), for each \( i = 1, \ldots, m \), while the trade-off involving the choice of \( \Delta K^i \) is evident.

4.4 Simulation Example

4.4.1 System description

The effectiveness of the two proposed approaches is tested in comparison on a multi-mass-spring system derived from the one proposed in [7]. As depicted in Figure 4.3, the number of hanged masses is increased up to 6 to better approximate a LSS; for details on the plant description we refer to [7] and [6] with the only variation that inter-mass friction coefficient is doubled, increasing the coupling degree. The 12-order LTI model is sampled at \( T_L = 0.25 \) [8] and is subject to state constraints \( x(t) \in \mathcal{X} \), where \( \mathcal{X} = \{ x \in \mathbb{R}^8 : -0.3 \leq z_i \leq 1, i = 1, \ldots, 6, z_2 \leq z_1 + 0.3 \} \), corresponding to constraining
mass positions between $-0.3$ and $1$ m, and by preventing mass #1 to go below mass #2 by more than $0.3$ m.

The equivalent discrete-time model (4.1) of the process is decentralized according to the following index selection:

$$I^1_u = I^1_r = \{1, 2\}, \ I^i_x = \{2i + 1, 2i + 2\}, \ (4.32a)$$
$$I^1_x = \{1, \ldots, 4\}, \ I^i_u = I^i_r = \{i + 1\}, \ i = 2, 3, 4, 5 \ (4.32b)$$

which clearly makes the constraint involving masses #1 and #2 to be local on subsystem $\Sigma^1$, in turns all constraints are local.

The tracking error threshold $\Delta_0$ is used to fairly make a comparison over the proposed approaches. In fact, the tuning knob parameters are chosen as $\Delta K = [\Delta_0, \ldots, \Delta_0, 0.4\Delta_0]'$ for the centralized and as $\Delta K_i = [\Delta_0, \Delta_0, 0.4\Delta_0]'$, $\Delta K_i = [\Delta_0, \Delta_0]'$, $i = 2, 3, 4, 5$ for the decentralized upper control layer. The resulting allowed reference variations are depicted in Figure 4.4 for both the proposed approaches as function of the sample times ratio.

### 4.4.2 Controller Choice

The two proposed approach, resumed in Theorems 7 and 8, respectively, are based on any control action which enforces constraints (4.14) and (4.30),
Figure 4.4: Plots of $\Delta r_i(N_i)$ as a function of $N_i$ obtained by solving (4.12) and (4.28), respectively: subsystems $\Sigma_1$ (continuous blue line), $\Sigma_2$ to $\Sigma_4$ (black dashed), and $\Sigma_5$ (red dotted with circles), and centralized approach (magenta dotted with triangles) respectively. A widely used technique capable of handling constraint is MPC, which we propose in a twofold fashion.

The UCL of the centralized approach is implemented by a standard MPC controller based on a resampled model of the plant in closed loop with the LCL. The sample time is chosen as the maximizer of the ratio $R(N) = \frac{\Delta r(N)}{N}$ so as to guarantee to restrict as less as possible the user commanded reference speed of change. In the following this controller will be referred as $HiMPC$, which stands for Hierarchical MPC. The decentralized approach concerns a set of supervisors at the upper control layer, each in charge of enforce constraints on its respective variables. The control strategy used is MPC, leading to decentralized MPC feedback. Each $DH\text{iMPC}^i$ is based on relative model $\Sigma_i$, resampled at the frequency $\frac{1}{T_i^{\text{H}}}$, where $T_i^{\text{H}} = T_L \arg \max_{N_i} \frac{\Delta r_i(N_i)}{N_i}$. Finally, the necessity of reference rate constraint is tested by means of the controller Constr, which is a centralized MPC supervisor similar to $HiMPC$, where no rate constraint is enforced and no account for underlying dynamics id made.

Both the aforementioned controllers were implemented using the Hybrid
4.4. Simulation Example

Toolbox [10] and the WIDE Toolbox [4].

4.4.3 Simulation results

In order to demonstrate the sufficiency of reference restrictions $\Delta r_i$ we show in Figure 4.5 the first two masses trajectories with feedback provided by three different controllers, i.e. HiMPC, DHiMPC and Constr. HiMPC and DHiMPC implements the centralized and decentralized hierarchical approaches presented in Section 4.2 and 4.3, respectively. The Constr controller reduces the user defined reference so that absolute constraints hold in the applied reference, thus it ignores both system dynamics and variation constraints. HiMPC (blue color lines) and DHiMPC (black color lines) behaves similarly with the first being more reactive due to faster sampling rate ($41T_L$ and $42T_L$ respectively) and, as showed in Figure 4.4, both of those enforce constraints in whole simulation but are able to track references (red lines) in the green area only, accordingly to the admissible references tightening described in

Figure 4.5: Masses #1 (continuous) and #2(dashed) trajectories with controllers HiMPC (blue), DHiMPC (red) and Constr (magenta). Green, yellow and red background areas depict admissible references, constraint tighten $\Delta K$ and inadmissible zone. Black bold bordered squares indicate where constraint violation occurs.
previous Sections. Constr controller feedback trajectories (magenta lines) are most responsive to reference updates and track everywhere, but violates the constraints in multiple occasions, as showed in Figure 4.5. Because of space limitations we do not report all masses trajectories. Instead we define a performance index as $P = 100 \frac{I_{HMPC} - I_{DHMPC}}{I_{DHMPC}}$, where $I_x$ is the MPC cost for controller $x$ in the whole simulation horizon, which permits to compare the two presented approaches. For the previous simulation the index value is 0.001%.

In order to examine the two approaches with less dependence on the user defined reference $p(t)$, we ran 50 simulation with $p(t)$ set as described in the following. Using the full admissible range for each mass, i.e. $[0, 0.7]$, $p(t)$ is piecewise constant with a random number of variations between 0 and 8, occurring at random instants. The result is that the faster sampling frequency that DHiMPC exploits for masses #3 – #6 let it overperform HiMPC, as indicated by the average performance index, $P = 1.48\%$.

4.5 Conclusions

This Chapter proposed a centralized and a decentralized hierarchical control approaches to handle state-dependent constraints in large-scale linear control systems. The control design is carried out in two steps: first, a lower-level set of decentralized linear controllers is designed to stabilize the process without accounting for the constraints; second, each regulator is fed by an upper-level controller, running at a slower pace, that manipulates the desired references so as to guarantee the fulfillment of the constraints. Such upper layer controller can be either centralized or decentralized. Although in this latter case some conservatism is introduced by treating the dynamic couplings as bounded disturbances, the approach is totally scalable and therefore suitable for constrained linear systems of large size.
Hierarchical DMPC: The Barcelona Drinking Water Network

5.1 Introduction

Drinking water management in urban areas is a subject of increasing concern as cities grow. Limited water supplies, conservation and sustainability policies, as well as the infrastructure complexity for meeting consumer demands with appropriate flow pressure and quality levels, make water management a challenging problem. Many modern water systems are operated through centralized tele-control systems. In most cases, network operation is carried out using heuristic rules and “historic” strategies, which were the product of years of operational experience and empirical results. While these strategies may generally be adequate, the best operational policies may be very complex to determine in large-scale interconnected networks. Thus, decision-support systems for operational control, which are based on mathematical models of network operation and optimal control techniques, provide useful guidance for efficient management of water networks.

Model-based Predictive Control (MPC) has been proved to be one of the most widely accepted advanced control technique for the operational control of water systems [22,68,82]. The main reason is that, once the network dynamical model is available, the MPC design just consists in expressing the desired performance specifications through different control objectives and constraints on system variables (e.g., minima/maxima of selected process variables and/or their rates of change), which are necessary to ensure process safety and asset health. The rest of the MPC design is automatic: the given model, constraints and weights define an optimal control problem over a finite time horizon in the future (for this reason the approach is said predictive). This is translated into an equivalent optimization problem and solved on line to obtain an optimal
sequence of future control moves. Only the first of these moves is applied to
the process, as at the next time step a new optimal control problem is solved,
to exploit the information coming from fresh new measurements using in a
receding horizon strategy. In this way, an open-loop design methodology (i.e.,
optimal control) is transformed into a feedback one.

Nevertheless, the main hurdle for MPC control, as any other model based
control technique, when applied to large-scale networks in a centralized way, is
the non-scalability. The reason is that a huge control model is required along
with the need of being rebuilt on every change in the system configuration
as, for example, when some part of the network should be stopped because
of maintenance actions or malfunctions. Subsequently, a model change would
require re-tuning the centralized controller. It is obvious that the cost of
setting up and maintaining the monolithic solution of the control problem is
prohibitive. A way of circumventing these issues might be by looking into de-
centralized MPC (DMPC) or distributed MPC techniques, where networked
local MPC controllers are in charge of controlling each one a part of the en-
tire system. The main difference between distributed and decentralized MPC
is that the former uses negotiations and recomputations of local control ac-
tions within the sampling period to increase the level of cooperation, whereas
the latter does not (at the benefit of computation time, but at the cost of
optimality), for further details see [11].

The success of centralized MPC (CMPC) drives now a new interest in this
old area of distributed control, becoming DMPC one of the hottest topics
in control during the early 21st century. Thus, two research projects (HD-
MPC and WIDE) are recently concluded in Europe, both focused on the
development of decentralized and distributed MPC techniques. Few works
have been recently published in this area; see, e.g., [6, 55, 77, 89, 99, 109],
among others. However, there is a prior problem to be solved: the system
decomposition into subsystems. The importance of this issue has already
been noticed in classic-control books addressing the decentralized control of
large-scale systems (LSS); see, e.g., [102] or [65]. These references propose
some approaches for dealing with the decomposition of dynamical networked
systems under certain assumptions, which are related to the level of coupling
of the constitutive elements belonging to the considered LSS.
The main contribution of this Chapter consists in presenting the application in simulation of a hierarchical-like DMPC approach to the Barcelona drinking water network (DWN). The aim of DMPC is to reduce the computational burden and increase scalability and modularity with respect to the centralized counterpart, but still maintaining a convenient level of suboptimality with respect to the desired control objectives. Moreover, the advantage of the hierarchical-like DMPC approach is the simplicity of its implementation given the absence of negotiations among controllers which allows for a simple implementation from the networking viewpoint. This fact involves much shorter computational times since only one optimization problem should be solved for each subsystem. Furthermore, each local MPC controller could be converted into its \textit{explicit} form \cite{13} leading to a low online complexity. To apply the proposed DMPC approach, the DWN is partitioned in a set of subnetworks using a partitioning algorithm that makes use of the topology of the network, the information about the actuator usage and heuristics. The partition approach finds a set of non-overlapping subsystems weakly interconnected.

The Chapter is structured as follows: Section 5.2 describes the considered case study. Section 5.3 briefly introduces the partitioning approach for dynamical systems used with the case study of this Chapter. Section 5.4 presents and discusses the hierarchical-like DMPC strategy applied to the case study. Section 5.5 discusses the main simulation results derived from the application of the proposed control approach over the considered case study. Finally, conclusions and directions for further development are reported in Section 5.6.

5.2 Case-study Description

5.2.1 System description

The Barcelona DWN, managed by Aguas de Barcelona, S.A. (AGBAR), not only supplies drinking water to Barcelona city but also to the metropolitan area. The sources of water are the Ter and Llobregat rivers, which are regulated at their head by some dams with an overall capacity of 600 cubic hectometres. Currently, there are four drinking water treatment plants (WTP): the Abrera and Sant Joan Despí plants, which extract water from the Llobregat river, the Cardedeu plant, which extracts water from Ter river,
and the Besòs plant, which treats the underground flows from the aquifer of the Besòs river. There are also several underground sources (wells) that can provide water through pumping stations. Those different water sources currently provide a flow of around 7 m$^3$/s. The water flow from each source is limited and with different water prices depending on water treatments and legal extraction canons.

The Barcelona DWN is structurally organized in two layers. The upper layer, named as transport network, links the water treatment plants with the reservoirs distributed all over the city. The lower layer, named distribution network is sectorized in subnetworks. Each subnetwork links a reservoir with each consumer. This Chapter is focused on the transport network. Thus, each subnetwork of the distribution network is modelled as a demand sector. The demand of each sector is characterized by a demand pattern, which can be predicted by using a time-series model [87]. The control system of the transport network is also organized in two layers (see Figure 5.1). The upper layer is in charge of the global control of the network, establishing the set-points of the regulatory controllers at the lower layer. Regulatory controllers are of PID$^1$ type, while the supervisory layer controller is of MPC type. Regulatory controllers hide the network non-linear behavior to the supervisory controller. This fact allows the MPC supervisory controller to use a control-oriented linear model.

5.2.2 System management criteria

AGBAR has provided the management policies for the Barcelona DWN, given their knowledge of the system, which are described below.

Minimizing water production and transport costs

The main economic expenses associated with drinking water production (treatment) are due to chemicals, legal canons, and electricity costs. The corresponding performance index to be minimized is expressed as

\[ f_1(t) = W_e \left( \alpha_1 + \alpha_2(t) \right) u(t), \]  

(5.1)

$^1$Linear controller where the control action is computed as a weighted sum of proportional, integral and derivative terms of the state
5.2. Case-study Description

Set-points
determination
(MPC, set of rules)

Control trajectories
realisation
(PID controllers)

real
disturbances
(demands)

GLOBAL CONTROL LEVEL
Supervision

LOCAL CONTROL LEVEL
Regulation

NETWORKED SYSTEM

Figure 5.1: Hierarchical structure for transport network system

where \( u \in \mathbb{R}^m \) denotes the manipulated flows through the system actuators, \( \alpha_1 \) corresponds to a known vector of dimension \( 1 \times m \) related to the economic costs of the water according to the source (treatment plant, dwell, etc.), and \( \alpha_2(t) \) is a vector of dimension \( 1 \times m \) associated with the economic cost of the flow through certain actuators (pumps only) and their control cost (pumping). Note the time variance of \( \alpha_2 \) due to the fact that pumping electricity costs have different values according to the time of the day. The weight \( W_e \) is the penalty associated with economic costs with respect to the other objectives that will be included in the MPC optimization problem.

Safety storage term

The satisfaction of water demands should be fulfilled at every time instant. However, some risk prevention mechanisms should be introduced in the tank management so that, the stored volume is preferably maintained around a given safety value in case of emergency, and to guarantee future water availability in case of demand forecast estimation errors. A quadratic expression for this concept is used and written as follows:

\[
f_2(t) = (x(t) - \beta x^{\max})^T W_x (x(t) - \beta x^{\max}), \tag{5.2}
\]

where \( x \in \mathbb{R}^n \) denotes the water volumes at network tanks, \( \beta \) is a term which determines the safety volume to be considered for the control law computation and matrix \( W_x \) defines the weight of the objective in the cost function. This
term prevents the controller from keeping the lowest possible water volumes in the tanks, which would reduce robustness to demand forecast errors.

**Smoothness of the control actions**

To smooth out the control action of MPC in order to avoid overpressures which can cause structural damage and leaks in the network, the following third term is included in the objective function to penalize variations $\Delta u(t) = u(t) - u(t - 1)$ of the control signal between consecutive sampling intervals

$$f_3(t) = \Delta u(t)^T W_{\Delta u} \Delta u(t),$$

where $W_{\Delta u}$ is a $m \times m$ weight matrix.

5.2.3 Control-oriented modelling

Control-oriented modelling principles for DWNs have been widely presented in the literature, see [22, 78]. In order to obtain a control-oriented model of the DWN, the constitutive network elements as well as their basic relationships should be discussed. The reader is referred to the aforementioned references and to [23, 37] for further details of DWN modelling and specific insights related to the case study of this Chapter.

**Tanks**

A water tank provides the entire system with the storage capability of drinking water. The mass balance expression relating the stored volume $x_i$ and the manipulated inflows and outflows (including the demand flows as outflows) for the $i$-th tank can be written as the difference equation

$$x_i(t + 1) = x_i(t) + \Delta t \left( \sum_i q_{in,i}(t) - \sum_j q_{out,j}(t) \right),$$

for all discrete-time instant $t$, where $q_{in,i}(t)$ and $q_{out,j}(t)$ correspond to the $i$-th inflow and the $j$-th outflow, respectively, given in m$^3$/s. The physical constraint related to the range of volume capacities for the $i$-th tank is expressed as

$$x_i^{\text{min}} \leq x_i(t) \leq x_i^{\text{max}}, \quad \forall t,$$
where $x_i^{\text{min}}$ and $x_i^{\text{max}}$ denote the minimum and the maximum volume capacity, respectively, given in m$^3$. Since this constraint is physical, it is impossible to send more water to a tank than it can store, or draw more water than the stored amount.

**Actuators**

There are two types of control actuators: pumps and valves. The manipulated flows through the actuators represent the control input variables of the model, denoted in the sequel as $u$. Both pumps and valves have lower and upper physical limits that are also model constraints. As in (5.5), they are expressed as

$$ u_i^{\text{min}} \leq u_i(t) \leq u_i^{\text{max}}, \quad \forall t, $$

where $u_i^{\text{min}}$ and $u_i^{\text{max}}$ denote the minimum and the maximum flow capacity, respectively, given in cubic meters per second. It is assumed that there is a local controller, which ensures that the required flow through the actuator is satisfied, following the discussion done for Figure 5.1.

**Nodes**

These elements correspond to the network points where water flows are merged or split. Thus, the nodes represent mass balance relations, being modelled as equality constraints related to inflows (from other tanks through valves or pumps) and outflows, these latter being represented not only by manipulated flows but also by demand flows. The expression of the mass conservation in these elements can be written as

$$ \sum_i q_{\text{in},i}(t) = \sum_j q_{\text{out},j}(t). $$

**Sectors of Consume**

A sector of consume represents the water demand made by the network users of a certain physical area. It is considered as a system disturbance. Since

---

3With a slight abuse of notation, the node inflows and outflows are still denoted by $q_{\text{in}}$ and $q_{\text{out}}$, respectively, despite they can be manipulated flows and hence denoted by $u$, if correspond.
the demand shows a periodic behavior with daily and weekly seasonalties, it can be forecasted by using methods based on time series. In this Chapter, the demand forecasting algorithm used by the MPC consists in a two-level scheme [87] composed by

- a time-series model to represent the daily aggregate flow values, and
- a set of different daily flow demand patterns according to the day type to cater for different consumption during the weekends and holidays periods. Every pattern consists of 24 hourly values for each daily pattern.

This algorithm runs in parallel with the MPC algorithm. The daily series of hourly-flow predictions are computed as a product of the daily aggregate flow value and the appropriate hourly demand pattern.

**Aggregate daily-flow model** The aggregate daily-flow model is built on the basis of a time series modelling approach using an ARIMA strategy. A time series analysis was carried out on several daily aggregate series, which consistently showed a weekly seasonality, as well as the presence of deterministic periodic components. A general expression for the aggregate daily flow model, to be used for a number of demands in different locations, was derived using three main components:

- A *weekly-period oscillating signal*, with zero-average value to cater for cyclic deterministic behavior, implemented using a second-order (two-parameter) model with two oscillating modes \( p_{1,2} = \cos(\frac{2\pi}{7}) \pm j\sin(\frac{2\pi}{7}) \).

- An *integrator* takes into account possible trends and the non-zero mean value of the flow data.

- An autoregressive component to consider the influence of previous flow values within a week. For the general case, the influence of seven previous days is considered. However, after parameter estimation and significance analysis, the models are usually reduced implementing a smaller number of parameters \( a_i \) such as the model output \( y \) is expressed as

\[
y(t) = -a_1y(t-1) - a_2y(t-2) - a_3y(t-3) - a_4y(t-4).
\]
Combining the previous components in the following way:

\[
\begin{align*}
\Delta y_{\text{int}}(t) &= y(t) - y(t - 1), \\
\Delta y_{\text{osc}}(t) &= \Delta y_{\text{int}}(t) - 2 \cos(2\pi/7) \Delta y_{\text{int}}(t - 1) + \Delta y_{\text{int}}(t - 2), \quad (5.9) \\
y_p(t) &= -\left( \sum_{i=1}^{4} a_i \Delta y_{\text{osc}}(t - i) \right), \\
\end{align*}
\]

the structure of aggregate daily flow model for each demand sensor is therefore

\[
y_p(t) = -b_1 y(t - 1) - b_2 y(t - 2) - ... - b_7 y(t - 7). \quad (5.11)
\]

The parameters \( b_1, \ldots, b_7 \) should be adjusted using least-squares-based parameter estimation methods and historical data (after pre-processing to obtain fault-free set). In parallel with the forecasting and control module, a data validation module should be considered, which validates the used information.

**1-hour flow model** The 1-hour flow model is based on distributing the daily flow prediction provided by the time-series model described in previous section using a one-hour-flow pattern that takes into account the daily/monthly variation in the following way:

\[
y_{p1h}(t+i) = \frac{y_{\text{pat}}(t,i)}{\sum_{j=1}^{24} y_{\text{pat}}(t,i)} y_p(j), \quad i = 1, \ldots, 24,
\]

where \( y_{p1h}(t) \) is the predicted flow for the current day \( j \) using (5.11) and \( y_{\text{pat}}(t) \) is the prediction provided by the one-hour-flow pattern with the flow pattern class day/month of the actual day. Demand patterns are obtained from statistical analysis. See [87] for further details.

**Network Model**

Considering the expressions presented above, the control-oriented model of a DWN in discrete-time state space can be written as

\[
x(t + 1) = A x(t) + B u(t) + B_p d(t), \quad (5.12)
\]
where \( x \in \mathbb{R}^n \) is the state vector corresponding to the water volumes of the \( n \) tanks, \( u \in \mathbb{R}^m \) represents the vector of manipulated flows through the \( m \) actuators, and \( d \in \mathbb{R}^p \) corresponds to the vector of the \( p \) demands. \( A, B, \) and \( B_p \) are the system matrices of suitable dimensions. Since the demands can be forecasted, we can assume they are known, hence \( d \) is a known vector containing the measured disturbances affecting the system. Additionally, (5.12) can be rewritten as

\[
\begin{align*}
x(t+1) &= A x(t) + \Gamma \nu(t), \\
\begin{bmatrix}
E_u & E_d
\end{bmatrix} \nu(t) &= 0,
\end{align*}
\]

where \( \Gamma = [B \ B_p] \), \( \nu(t) = [u(t)^T \ d(t)^T]^T \), and \( E_u, E_d \) are matrices of suitable dimensions. Notice that (5.13a) comes from the mass balance in tanks while (5.13b) at the network nodes (see (5.7)). Also notice that when all the network flows are manipulated, then \( A \) is an identity matrix of suitable dimensions.

This modelling methodology has been applied to the Barcelona DWN aggregate network in Figure 5.2. From this figure, it can be seen that the network is comprised of 17 tanks (state variables), 61 actuators (26 pumping stations and 35 valves), 11 nodes and 25 main sectors of water demand (model disturbances). The model has been simulated and compared against real behavior assessing its validity. The detailed information about physical parameters and other system values are reported in [37].

### 5.3 DWN Partitioning Approach

The application of DMPC to DWN depends crucially on how the network is decomposed into subsystems. Identifying subsystems is not an easy task in a large-scale network as it involves to find “sufficiently small” sections of the networked plant that are not “too coupled” among them. The partitioning algorithm, aims at obtain this decomposition automatically by identifying clusters of elements that are strongly connected with each other but weakly interconnected with the other clusters, in order to represent the whole network as a set of loosely coupled subsystems [100]. The algorithm is intended to be used off-line, that is, the partitioning of the system is static and is unable
5.3. DWN Partitioning Approach

Figure 5.2: Aggregate case of the Barcelona Drinking Water Network

to account for on-line plant modifications. A possible improvement is the adaptation of the proposed algorithm such that the partitioning could be done on-line when, for instance, some structural change of the network occurs.

5.3.1 Partitioning algorithm

As a starting point, the partitioning algorithm requires the following information of the DWN:

1. The interconnection structure characterized by the matrix

\[ M = \begin{bmatrix} A_{sp} & B_{sp} \end{bmatrix}, \quad (5.14a) \]

where

\[ A_{sp} = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}, \quad B_{sp} = \begin{bmatrix} B \\ E \end{bmatrix}. \quad (5.14b) \]
where $A$ and $B$ are the system matrices in (5.12), the subscript $sp$ identifies the matrices used for system decomposition, and $E \triangleq [E_1 \ E_2]$ is the matrix related to the equality constraints (5.13b). In order to take into account input bounds, new normalized inputs are introduced $\bar{u} \triangleq u/u_{\max}$ so that $\bar{u} \in [0, 1]$. Thus, new matrices $\bar{B}$ and $\bar{E}$ are introduced in (5.14b) to take into account the rescaling. From matrix $M$, the adjacency matrix $\Psi$ of the network graph can be obtained by replacing the non-zero elements by ones, leaving the null elements unchanged.

2. A threshold value $\varepsilon$ is used for determining whether a term, which takes into account the actuator capacity (maximum allowable flow) and its usage frequency, has a negligible effect on the entire plant. In this way the less important actuators are filtered out, in order to reduce the coupling degree of the system and identify independent subnetworks.

The partitioning algorithm proceeds by decomposing the matrix $M$ into a set of submatrices, named as partitions and denoted by $P_{\varepsilon} = \{M_1, \ldots, M_{np}\}$. Then, $P_{\varepsilon}$ correspond to a set of subgraphs (subsystems) obtaining by deleting the edges corresponding to elements of $M$ with magnitude no larger than $\varepsilon$. That is, the idea behind the partitioning approach is to neglect less important elements (i.e., links) in matrix $M$ such that the resulting $\hat{M}$ is less coupled. Ideally, $\hat{M}$ should lead to a permutation matrix $P$ such that $P^T \hat{M} P$ is block-diagonal. This procedure is repeated iteratively by reducing $\varepsilon$ until a satisfactory number of partitions is obtained. Algorithm 1 summarizes the steps of the proposed partitioning approach.

Partitions can be tuned by means of parameter $\varepsilon$, which gives to the user a tuning knob toward matching the desired number and size of subsystems.

Typically, in the first iteration, Algorithm 1 neglects a high number of elements of $M$, highly reducing the matrix connectivity degree and obtaining a subsystem decomposition. Then, once the sets of states/inputs relative to each partition are computed, the task of finding a suitable $P$ that block-diagonalizes the matrix $P^T \hat{M} P$ is a matter of linear algebra implementation. Every subsystem is composed by sets of state and input variables that are linked, meaning that are in the same block in the $P^T \hat{M} P$ diagonal. Let $X^i$ and $U^i$ be respectively the sets of state and input variables assigned to subsystem $i$, while $L(X^i)$ and $L(U^i)$ determine the number of variables for each
5.3. DWN Partitioning Approach

set. A subsystem is created if both numbers are different than zero. All state and input variables that are not assigned to any of the currently created subsystems, that is, that does not belong to $X^i$ or $U^i$, respectively, are available for the next iteration. Otherwise, variables already assigned to a subsystem in the current or in a previous iteration, are masked\(^3\) to prevent their reassignment to other subsystem.

Then, a new iteration of the algorithm starts by decreasing $\varepsilon$ (e.g., halving $\varepsilon$). Algorithm 1 iterates until all state variables are assigned to a subsystem. Note that the algorithm may terminate even if some inputs are not be assigned to any subsystem, which is due to automatic threshold based neglecting process. Such issue can be managed by manually include unassigned inputs to proper subsystem following engineering insight.

The importance of the mask arises from the structure of the algorithm. In fact, if not excluded, all previously assigned states and inputs would be part of the next iteration partition, introducing couplings and hence increasing the size of the resulting submodels. The aforementioned inclusion easily follows from the decreasing of $\varepsilon$ among sequential iterations.

Few remarks on the above algorithm:

1. At any iteration of Algorithm 1, the numerical value of $\varepsilon$ is a crucial tuning knob of the approach. A guideline is that the larger is the decreasing step, the larger is the size of the obtained subsystems. Ways for automatically determining the step size are a subject of current research.

2. Matrix $E$ in (5.14b) defines a constraint among actuators that can be easily taken into account if all the actuators belong to the same subsystem. Otherwise, since each controller manipulates every partition independently from the others, negotiations between controllers would be required to guarantee the fulfillment of node constraints.

3. The use of masks to prevent state reassignment avoids that submodels have overlapping states and inputs: if a state variable is used in a model by a controller, no other controller can use it. The main benefit of this\(^3\)Let us consider a variable to be masked when it does not belong to any set since it has already been classified in a previous iteration.
Algorithm 1 Automatic partitioning algorithm

1: Initialise masks to a neutral value
2: Initialise the sets of unassigned variables $X$ and $U$ with all state and input variables, respectively
3: Determine the number of unassigned states: $N_x = L(X)$;
4: Init $\varepsilon$
5: while $N_x > 1$ do
6: Apply masks to $A_{sp}$ and $B_{sp}$
7: $M = [A_{sp} B_{sp} \bar{u}]$
8: For all elements of $M$
9: if $M_{i,j} < \varepsilon$ then
10: $\tilde{M}_{i,j} = 0$
11: else
12: $\tilde{M}_{i,j} = 1$
13: end if
14: Find $P$ such that $P'\tilde{M}P$ is block diagonal
15: Identify parts satisfying $N_{xi} = L(X^i) > 0$ and $N_{ui} = L(U^i) > 0$ and add to previous ones
16: Update $N_x$
17: Update masks with updated states and inputs
18: Update $\varepsilon$
19: end while

choice is the very low level of coupling between partitions, but the price to pay is a potential decrease of closed-loop performance.

4. The current structure of the algorithm is unsuitable to handle state overlaps because it relies on links between elements that present different degree of coupling. Hence, once the stronger couplings are eliminated (using masking), the weaker ones gain relative importance. State overlaps may be introduced a posteriori based on engineering insight, in order to increase the adherence with respect the original centralized model. Handling overlapping in an automatic way is also a current research topic.
5. In some cases even relatively small connections, i.e., capable of carrying a minor amount of water, are very important for demand satisfaction. A way of accounting for such an issue is to perform a simulation using, for instance, a CMPC controller, and compute the average percentage of use for each actuator. Thus, this information could be used to weight $\bar{u}$ component-wise. The main drawback of this approach is the need of (and dependence on) simulation.

6. Note that the proposed algorithm can be customized by setting different importance levels of states vs. inputs, by weighting the related components in $M$. By defining

$$M = [W_A' A_{sp} W_A \quad W_B' B_{sp} W_B \quad W_u' \bar{u} W_u],$$

where $W_A$, $W_B$ and $W_u$ are weights respectively of $A$, $B$ and $u$, it is possible to affect the resulting partitioning outcome.

7. The structure of the proposed algorithm suggests that termination is achieved if the $\varepsilon$ value is decreased at each iteration. However, at the current status of development, the algorithm cannot guarantee any property for the resulting partitioning but the assignment of all system-state variables to a subsystem.

The decomposition process of matrix $M$ reported here is similar to the one proposed by the $\varepsilon$-decomposition method in [100]. The underlying idea in both cases is to disconnect those actuators corresponding to interconnections with strength smaller than the prescribed $\varepsilon$, identifying the disconnected subsystems. According to [100], there are $s$ different $\varepsilon$-decompositions $P_\varepsilon$ that can be obtained for different values of $\varepsilon$ satisfying

$$\max_{i \neq j} |m_{ij}| = \varepsilon_1 < \varepsilon_2 < \cdots < \varepsilon_K = 0,$$

with $K \leq s$, where $s = \dim(M)$. Moreover, such decompositions are nested, that is, the partitions obtained satisfy: $P_{\varepsilon_1} \subset P_{\varepsilon_2} \cdots P_{\varepsilon_K}$ with $P_{\varepsilon_1}$ being the finest and $P_{\varepsilon_K}$ the coarsest. The main novelty of the algorithm is the matrix normalization taking into account actuator physical/operative limits, and the iterative threshold updating that allows one to take into account weaker coupling without being influenced by the stronger ones.
5.3.2 Case study partitioning

Using the partitioning algorithm presented in this section, the aggregate model of the Barcelona DWN is decomposed in three subsystems, as depicted in Figure 5.2 in different colors. The resultant decomposition follows the scheme shown in Figure 5.3. The subsystems are defined by the following elements:

- **Subsystem 1**: Composed by tanks $x_i$, $i \in \{1, 2\}$, inputs $u_j$, $j \in \{1 : 5\}$, demands $d_l$, $l \in \{1, 2, 3\}$, and nodes $n_q$, $q \in \{1, 2\}$. It is represented in Figure 5.2 with red color and corresponds to Subsystem $S_1$ in Figure 5.3.

- **Subsystem 2**: Composed by tanks $x_i$, $i \in \{3, 4, 5, 12, 17\}$, inputs $u_j$, $j \in \{7 : 16, 18, 19, 25, 26, 32, 34, 40, 41, 47, 48, 56, 60\}$, demands $d_l$, $l \in \{4 : 7, 15, 18, 22\}$, and nodes $n_q$, $q \in \{3, 4, 7\}$. It is represented in Figure 5.2 with green color and corresponds to Subsystem $S_2$ in Figure 5.3.

- **Subsystem 3**: Composed by tanks $x_i$, $i \in \{6 : 11, 13 : 16\}$, the inputs $u_j$, $j \in \{6, 17, 20 : 24, 27 : 31, 33, 35 : 39, 42 : 46, 49 : 55, 57, 58, 59, 61\}$, demands $d_l$, $l \in \{8 : 14, 16, 17, 19, 20, 21, 23, 24, 25\}$, and nodes $n_q$, $q \in \{5, 6, 8 : 11\}$. It is represented in Figure 5.2 with blue color and corresponds to Subsystem $S_3$ in Figure 5.3.

Table 5.3.2 collects the resultant dimensions for each subsystem and the corresponding comparison with the dimensions of the vectors of variables for the entire aggregate network.

5.4 DMPC Approach

Using the Barcelona DWN decomposition which is obtained from the partitioning algorithm in Section 5.3, a DMPC strategy is implemented in order to manage the networked system. This DMPC strategy considers

- the dynamical system model in (5.13) split in 3 subsystems;

- the physical constraints (5.5) and (5.6) for each subsystem;
5.4. DMPC Approach

<table>
<thead>
<tr>
<th>Elements</th>
<th>Subsist 1</th>
<th>Subsist 2</th>
<th>Subsist 3</th>
<th>Whole Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tanks</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>Actuators</td>
<td>5</td>
<td>22</td>
<td>34</td>
<td>61</td>
</tr>
<tr>
<td>Demands</td>
<td>3</td>
<td>7</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>Nodes</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 5.1: Dimension comparison between the subsystems and the whole network.

![Diagram](image)

Figure 5.3: Conceptual scheme of the partitioned Barcelona DWN

- a demand forecasting algorithm (presented in Section 5.2.3);
- a multi-objective cost function including the objectives (5.1), (5.2), and (5.3) and scalarized using weights. The function can be written as

\[
J(t) = \sum_{i=0}^{H_u-1} f_1(t + i|t) + \sum_{i=1}^{H_p} f_2(t + i|t) + \sum_{i=0}^{H_u-1} f_3(t + i|t),
\]

(5.15)

where \(H_p\) and \(H_u\) correspond to the prediction and control horizons, respectively, index \(t\) represents the current time instant while index \(i\) represents the predicted time along \(H_p\). In this Chapter, the prediction horizon is related to the 24-hours demand seasonality. Regarding the value of \(H_u\), it has been set to be equal to \(H_p\), following the criterion of the DWN management company.

In order to explain and discuss the implementation of the solution sequence for the considered hierarchical-like DMPC strategy, denote \(C_i\) as the MPC
controller related to the subsystem $S_i$ (for $i \in \{1, \ldots, 3\}$), and $\mu_{ij}$ as the set of control actions $u$ (manipulated flows, see (5.13)) going from $S_i$ to $S_j$ (for $j \in \{1, \ldots, 3\}$, $i \neq j$). Notice that $\mu_{ij}$ not only contains values of each component at time $t$ but also all values over $H_u$, i.e., if $\mu_{ij} = \{u_a, u_b, \ldots\}$, then

$$u_a \triangleq [u_a(t|t) \, u_a(t+1|t) \ldots u_a(t + H_u - 1|t)]^T,$$

$$u_b \triangleq [u_b(t|t) \, u_b(t+1|t) \ldots u_b(t + H_u - 1|t)]^T,$$

$$\vdots$$

with $u_a(t+i|t)$ denoting the value of $u_a$ at time $t+i$ (over the control horizon) given $t$. Once introduced this notation and according to the scheme in Figure 5.3, the particular sets $\mu_{ij}$ for the case study are defined as

$$\mu_{13} = u_6,$$

$$\mu_{23} = [u_{20}, u_{21}]^T,$$

$$\mu_{32} = [u_{18}, u_{32}, u_{34}, u_{40}, u_{47}, u_{56}, u_{60}]^T.$$

According to [102], the pure hierarchical control scheme determines a sequence of information distribution among the subsystems, where top-down communication is available from upper to lower levels of the hierarchy. Note that, despite the subsystems coupling (given by the shared links), the main feature of the pure hierarchical control approach relies on the unidirectionality of the information flow between controllers.

Looking at Figure 5.4, where the directions of sets $\mu_{ij}$ are graphically shown, it is possible to realize that the set $\mu_{32}$ (red-dashed line in the figure) breaks the mentioned unidirectional flow between MPC controllers. This fact implies that the standard hierarchical control scheme for partitioned LSS cannot be straight applied. To solve this situation and design a DMPC strategy, a hierarchical-like DMPC approach is proposed and conveniently implemented. This strategy follows the hierarchical control philosophy and the sequential way of solving the optimization subproblems of the corresponding MPC controllers but also considering the appearance of bidirectional information flows.

---

4With some abuse of the notation, the elements of vector $u$ are denoted with the corresponding discrete-time dependence in order to differentiate the vector from its components.
5.4. DMPC Approach

For this purpose, additional constraints and heuristics are taken into account in order to cope with the feature of having the double direction in the flow of information between some of the controllers. In particular, Figure 5.4 presents the considered hierarchy for the case study, where the controller at the upper level determines the values of variables shared with controllers at lower level. Notice that Figure 5.4 also shows why the pure hierarchical control approach cannot be employed since the MPC controller $C_2$ shares bidirectional information with $C_3$.

Therefore, the proposed solution sequence of the described hierarchical-like control problem for the aggregate model of the Barcelona DWN at each time step $t \in \mathbb{Z}_{\geq 1}$ is the following:

- $C_3$ computes the control actions of $S_3$ and sets $\mu_{13}$ and $\mu_{23}$. Set $\mu_{32}$ is considered as a set of virtual demands\(^5\) within the controller $C_3$.

- $C_1$ computes the control actions of $S_1$ considering $\mu_{13}$ as a set of virtual demands.

- In parallel, $C_2$ computes the control actions of $S_2$ considering $\mu_{23}$ as a set of virtual demands. Additionally, $C_2$ also computes $\mu_{32}$ to be used as a set of virtual demands for $C_3$ at time step $t + 1$.

**Remark 4.** Notice that in the proposed DMPC scheme, at the first time step ($t = 1$), the initial values of the control actions belonging to set $\mu_{32}$ are not available. Those values can be obtained by solving a constraint satisfaction problem (CSP) defined by the models and constraints of subsystems $S_2$ and $S_3$ through the algorithm proposed in [47]. The solution of this CSP provides feasible control actions for the set $\mu_{32}$, which allows starting the solution sequence described above. For subsequent time steps, values of $\mu_{32}$ take values computed by $C_2$ in the previous time step, i.e., the elements belonging to those

---

\(^5\)Consider two subsystems $S_1$ and $S_2$, which share a set of manipulated flows $\mu_{12}$. According to the notation employed in the Chapter, those flows come from $S_1$ to $S_2$. If the solution sequence of optimization subproblems — defined by the pre-established hierarchical order — determines that $\mu_{12}$ is computed by the MPC controller of $S_1$, then flows in $\mu_{12}$ are considered as virtual demands in the controller related to $S_2$ since their value are now imposed in the same way as the water demands.
5. Hierarchical DMPC: The Barcelona Drinking Water Network

Figure 5.4: Hierarchy of MPC controllers $C_i$. Their solution sequence is top-down

Sets at time $t$ are now assigned as (see (5.16))

$$u = \begin{bmatrix} u(t+1|t-1) \\ \vdots \\ u(t+H_u-1|t-1) \\ u(t+H_u-1|t-1) \end{bmatrix}.$$ \hspace{1cm} (5.17)

5.5 Results

The results obtained by using the proposed DMPC strategy, presented in Section 5.4, are compared with those obtained when a CMPC strategy is used. The model parameters and measured disturbances (demands) have been supplied by AGBAR. Demands data correspond to the consume of drinking water of the city of Barcelona during the year 2007. Different scenarios are considered by modifying some controller parameters corresponding to different prioritisations of the control objectives. These parameters are the safety volume, denoted as $\beta$, and the weight matrices in the cost function (5.15). Regarding $\beta$, it has been set to the following values:

1. 80% of $x^{max}$, that is denoted as $\mu = 0.8 \, x^{max}$. This value is purely illustrative to show the effectiveness of the MPC controller;
2. 20% over the minimum tank volumes requested to satisfy the demands\(^6\) (except for tanks \(x_5, x_6\) and \(x_8\) in Figure 5.2, since they are considered as sources due to their strategic management requirements and network location). This second vector of safety volumes, denoted as \(\eta\), is more convenient since it keeps the volumes of the tanks as low as possible, satisfying the demands at each time instant. These minimum volumes are taken from previous studies reported in [23].

In particular, consider \(\Omega = (\omega_\alpha, \omega_x, \omega_\Delta u)\) as the 3-tuple of weights associated to the matrices \(W_\alpha = \omega_\alpha I, W_x = \omega_x I\) and \(W_u = \omega_u I\) used in (5.1), (5.2) and in (5.3), respectively. Thus, the following scenarios have been stated:

- **Scenario 1**: \(\beta = \mu\) and \(\Omega = (1, 0.1, 10^{-3})\);
- **Scenario 2**: \(\beta = \mu\) and \(\Omega = (1, 1, 10^{-3})\);
- **Scenario 3**: \(\beta = \mu\) and \(\Omega = (1, 0.1, 0.1)\);
- **Scenario 4**: \(\beta = \eta\) and \(\Omega = (1, 0.1, 10^{-3})\);
- **Scenario 5**: \(\beta = \eta\) and \(\Omega = (1, 1, 10^{-3})\);
- **Scenario 6**: \(\beta = \eta\) and \(\Omega = (1, 0.1, 0.1)\).

All results have been obtained considering four days real-demand scenarios (with 1 hour of sampling time), and \(H_p = H_u = 24\). The network has been simulated using the same model than the MPC controller but fed with real water demands. The network model has been calibrated and validated using real data coming from AGBAR databases. All simulations were performed in MATLAB\(^\text{®}\) 7.2 implementations running on an Intel\(^\text{®}\) Core\(^\text{TM}\) 2, 2.4 GHz machine with 4Gb RAM.

The hierarchical-like DMPC controller is compared with a CMPC in the considered scenarios. The computational burden of each controller was determined as the time required for the QP solver in obtaining the solution when

\(^6\)The minimum volume to satisfy the water demand is based on considering the worst-case scenario, i.e., the minimum volume within tanks that is required to satisfy the maximum network demand with no inflow. Some extra percentage of this volume can be added in order to take into account demand forecast inaccuracy (safety amount).
Table 5.2: Computation time and performance comparisons

<table>
<thead>
<tr>
<th>Scenario</th>
<th>CMPC</th>
<th>DMPC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\sum f_1$</td>
<td>Total Time</td>
</tr>
<tr>
<td>Scenario 1</td>
<td>49.86</td>
<td>360.02</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>52.95</td>
<td>373.25</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>49.94</td>
<td>395.18</td>
</tr>
<tr>
<td>Scenario 4</td>
<td>49.84</td>
<td>394.24</td>
</tr>
<tr>
<td>Scenario 5</td>
<td>52.93</td>
<td>396.12</td>
</tr>
<tr>
<td>Scenario 6</td>
<td>49.97</td>
<td>488.79</td>
</tr>
</tbody>
</table>

the CMPC is implemented while as the sum of the time used by all the MPC controllers in the DMPC scheme. Table 5.5 gathers the simulation time (in s) for all the mentioned scenarios. This time regards to the time used for solving the optimization problem only. As the way of showing the minimum loss of performance, Table 5.5 also presents the total economic cost related to the fourth simulation day. Notice that the simulations do not consider a warm starting so the first simulation day corresponds to a transient of the system behavior. The behavior of the rest of the days corresponds with a steady state (taking into account the cyclic pattern of the demands). In order to clearly illustrate this feature, Table 5.5 collects the complete information of the economic costs for Scenario 1, considering the discrimination by water and electrical costs (pumping) for each control scheme. These costs have been obtained by replacing the computed optimal actions (flows) and the expression of the economic costs in (5.1). It is also important to highlight that the economic costs collected in Tables 5.5 and 5.5 are given in economic units rather than in real values (Euro) due to confidentiality reasons. Additionally, Figure 5.5 shows a comparative of the evaluation of the cost function $J(t)$ in (5.15) for both control strategies (CMPC and DMPC), without considering the first simulation day.

Notice from Table 5.5 that the loss of performance given when using the DMPC strategy is never greater than 2%, that is a remarkable result given the reduction of the computation time, which can achieve up to 35% improvement. Thus, despite the DMPC approach inevitably leads to a small loss of performance, the benefits in terms of time and computational load are signifi-
5.5. Results

Figure 5.5: Cost function evaluation for CMPC and DMPC strategies

<table>
<thead>
<tr>
<th>Day</th>
<th>CMPC</th>
<th>DMPC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Water</td>
<td>Electric</td>
</tr>
<tr>
<td>1</td>
<td>29.601</td>
<td>22.107</td>
</tr>
<tr>
<td>2</td>
<td>27.733</td>
<td>22.141</td>
</tr>
<tr>
<td>3</td>
<td>27.722</td>
<td>22.138</td>
</tr>
<tr>
<td>4</td>
<td>27.722</td>
<td>22.138</td>
</tr>
</tbody>
</table>

Table 5.3: Complete discrimination of economic costs for the Scenario 1

cant. In this particular application, the CMPC could also satisfy the real-time constraint since the sampling time is one hour. Thus, the main motivation for using DMPC in this application would not be the improvement in computation but the scalability and the potential change adaptability easiness that the control strategy could offer. In fact, according to discussions with the AGBAR company, the main reason for using a DMPC approach in the case study, additionally to the easier maintenance of the (sub)system models, is that it allows replacing the current legacy control in multiple steps,
where the DMPC is implemented on a selected network part only at each step. This ability is important for practical application and maintenance, which allows moving some part of the network to the current legacy control when some malfunction/fault is detected without stopping the supervisory MPC controller.

Regarding Table 5.5, notice that economic costs reach their steady state after the second day. Moreover, it is worth to highlight the inverse behavior of the electrical costs between the CMPC and DMPC cases. In global terms, the performance given by the DMPC is slightly worse, but this consideration should be done with the total cost. When the system is controlled by employing DMPC, the subsystems have no information of the water costs related to external sources. This fact explains that the optimization over-emphasizes the reduction of pumping costs inside the subsystem. By contrast, CMPC has the information of all water costs so it optimizes this factor but moves the water inside the network, incrementing the electric costs. Thus, CMPC strategy selects the optimal source regarding water costs while the proposed DMPC approach emphasizes local objectives (pumping costs) but is not able to minimize the water costs since it requires a global vision of the system.

In particular, the results show that with a suitable tuning of the MPC controller, no matter the topology considered (centralized or decentralized), the behavior for some volumes and manipulated flows remain almost the same. This fact can be seen in Figures 5.6 and 5.7, where the flow through the valve 54 and the volume in tank 10, respectively, are depicted when they are computed for a DMPC and CMPC schemes. Notice for instance that volume behaviors are quite similar and, given the weight tuning of Scenario 3, they smoothly oscillate around the desired reference, thanks to a convenient compromise of the safety and smoothness terms in the cost function.

5.6 Conclusions

In this Chapter, a DMPC strategy for DWN has been proposed. The DWN is decomposed in a set of subnetworks using a partitioning algorithm that makes use of the topology of the network, the information about the actuator usage and heuristics. A hierarchical structure related to the order of execu-
5.6. Conclusions

The use of the DMPC controllers allows one to take into account global network constraints. A comparative study between the CMPC and DMPC approaches has been developed using as case study the aggregate model of the Barcelona DWN. Results have shown that the partition algorithm, helped by an analysis of the system topology and heuristics, yields a proper segmentation of the whole network without overlapping models. The performances of CMPC and DMPC schemes were compared in terms of economical benefits and computational burden. Results have shown the effectiveness of the DMPC strategy in the important reduction of such computational burden despite the loss of performance of the control scheme, which in turn, has resulted to be quite small.

In future research, issues related to the the possibility of allowing the subsystems to overlap will be investigated, in addition to topics of stability of the proposed control approach. From the point of view of the control problem, feasibility issues are not discussed, being this a topic of ongoing research. However, the feasibility of the solution is guaranteed given the topological design of the network, since it has been properly dimensioned to supply the
Figure 5.7: Resultant volume related to tank 10 (d130BAR)

water to each demand sector from the available sources. On the other hand, the proposed partitioning approach provides a first automatic subsystem decomposition but since the partitioning result depends on the value of $\varepsilon$, which should be selected by trial and error. A further improvement would consists in selecting the value of $\varepsilon$ through optimization by establishing criteria to decide whether or not the obtained system decomposition is adequate. Moreover, some restrictions could be added regarding the complete assignment of inputs and states to the different subsystems. Finally, different schemes of hierarchical control and in general, strategies that deal with partitioned systems should be designed and further investigated for being tested over this type of networked systems.
Chapter 6

Conclusions and Outlook

6.1 Conclusions

This thesis focused on the problem of controlling a networked large scale system, employing decentralized and hierarchical approaches, mainly focusing on MPC theory.

Chapter 1 surveyed existing approaches available in the literature about non-centralized control.

Chapter 2 introduced a synthesis approach for decentralized linear control of linear system which can account for network topology. Then the approach was extended to cope with unreliable network connections, dealing with both robust and stochastic uncertainties. The first aimed at guarantee convergence and constraint holding in a worst-case scenario fashion, while the second relaxed the hard constraints and considered mean-square stability in favor of better closed-loop performances.

Chapter 3 focused on decentralized MPC exploring the trade-off which is inherited with the decentralization procedure and proposed sufficient criteria to test closed-loop stability \textit{a-posteriori}. Two frameworks were considered, for open-loop stable systems, where the controller’s task is performance optimization, and for unconstrained stabilizable plants. Finally, an extension considering network imperfections such as packet dropouts was presented, allowing the stability test to be applicable to networked systems.

In Chapter 4 two approaches were developed to implement hierarchical multi-rate control with \textit{a-priori} stability guarantees. The supervisor exploits a resampled model of the closed-loop of plant and lower layer controller to derive optimal restriction on admissible range and speed of change of the reference signal. To exploit potential improvements that derive from differentiating the sample time of various parts of the system, a decentralized supervisor design
extension was finally proposed.

The case study of the Barcelona WDN was described in Chapter 5 where a hierarchical-like decentralized control approach was tested in real-data base scenarios against a CMPC controller. The plant decentralization was carried out according to a proposed algorithm. The proposed decentralized approach performances were compared with a centralized one using real data based scenarios.

Most of the theoretical work described in these chapters was translated into Matlab tools, included in the publicly available toolboxes described in the Appendices A and B.

6.2 Outlook

The field of non-centralized control is not in its infancy, but there still is plenty of place for extensions and industrial applications.

Probably, the most crucial open problem concerns the plant decomposition into subsystems. In fact, not only the current literature lacks of automated algorithms for solving this problem, but also no formal definition of “optimal decomposition” is agreed among researchers. Intuitively, the decomposition should allow the best compromise between the competing objectives of “small” submodels capable of guaranteeing performances “as close as possible” to centralized approaches. Unfortunately these two objectives are incompatible as to obtain a good performance a model-based control technique should exploit a fairly complete model.

Chapter 5 proposed the idea of using simulation to evaluate closed-loop performances, and then use trial and error to trade it with subsystem size. In principle one can think of building an optimization problem over the decomposition where the cost function is a weighted sum of subsystem size and performance, the latter being computed by means of a simulation. However, the complexity of such optimization problem would be quite large. This makes the problem still open, and engineering insight the most applicable solution.

Furthermore, the ideal decomposition algorithm should also account for network model, as it appears obvious that such a parameter influences the “optimal” subdivision. One can also consider that many systems, such as un-
manned aerial vehicles, may have their network topology and dynamics varied over time. In such a case the decentralization is to be updated, accounting for the new configuration. Therefore, auto-reconfiguration of decentralization and, in general, of networked control is a relevant open issue.

Considering the recent spread of wireless nodes in control, another promising research direction is the explicit inclusion of communication efforts in the control design. The majority of wireless devices are employed to save wiring costs/efforts, hence are powered by a limited capacity battery. It follows that control action which heavily use the radio are unsuitable and battery life aware approaches are particularly welcome in the field.

Stochastic control, as presented in Chapter 2 is a noticeable example of new research lines, as it permits to overcome conservativeness typical of robust approaches. In real life applications, plant constraints are often set using engineering insight and are not necessarily strict. Therefore a “small” violation which rarely occurs is likely to be largely tolerable. In view of this consideration, stochastic control, which can look at the average case rather than the worst one with the price of unfrequent constraint violations, is a relevant alternative. However, most stochastic approaches (and Chapter 2 is no exception) are unable to properly describe constraint enforcement properties, for instance giving the probability of violation as a function of violation magnitude.

Few non-centralized approaches can guarantee stabilization if state feedback is not available. Output feedback is largely common in industry as many state variables are either unmeasurable or relative sensors are too expensive, thus, a common practice is to employ a state estimator in the loop. Giving formal proof of stability in such a case is surely useful in a wide variety of contexts, especially in the field of distributed, decentralized or hierarchical control, where also the estimator cannot exploit full output measurements.

Occasionally, the use of non-centralized MPC is motivated by the need of speeding up the computation, rather than communication issues. The explicit implementation can be in some cases the solution, implementable e.g. using tools such as the one of Appendix B, but the complexity is turned from time into space. In fact, a large MPC leads to a large number number of explicit regions and the memory used to store is therefore to be considered. Moreover,
the problem of determining, given the current state measure, the region this
belongs to (the so-called “point location problem”) is not trivial and hence
time consuming.

There exist regular region shapes for which the point location problem is
trivial and very few memory is required for storage, examples are simplices and
rectangles. This consideration suggests to consider approximations of explicit
MPC such that the region vertices lies on those efficiently memorizable shapes,
as presented in Appendix B. However, the use of such approximation, besides
affecting performances, may also compromise stability, thus the approach of
Appendix C is relevant.

The above consideration makes the problem of using approximation of
decentralized MPC a topic of promising future research. In fact, the curse of
dimensionality is counter-acted in a twofold manner: i) smaller system model
as result of decentralization and ii) less point location effort following regu-
larity of regions as consequence of approximation. Moreover, such a control
scheme may be used as supervisor layer for underlying super-fast dynamics,
employing MPC as described in Chapter 4, also in a decentralized manner.
Appendix A

WIDE Toolbox

A.1 Toolbox description

The WIDE Toolbox for Matlab is named after the project “WIDE - Decentralized and Wireless Control of Large-Scale Systems”, contract number FP7-IST-224168 of the European Commission which founded its development (http://ist-wide.dii.unisi.it/). It is freely available and hosted in the Project web-site at the URL http://ist-wide.dii.unisi.it/index.php?p=toolboxsp.

The WIDE Toolbox for Matlab is set a of functionalities oriented to centralized and decentralized/distributed/hierarchical Large-Scale control that takes explicitly into account network effects. The use of Object-Oriented programming ensures the usability of the tool in any context.

Due to both wiring costs and improved reliability of wireless communication, nowadays many control systems implement sensor-to-controller and controller-to-sensor connections without a physical wire. In some cases the use of a wire is impeded by the physics of the system, making wireless connections the only implementable solution. The price inherited from this technology is the plant exposition to inconstant, unpredictable but “describable”, negative effects introduced by the wireless network between the control system components.

Moreover, the layout of the plant may cover a considerably large physical space. As it is well known from physic electromagnetic theory, in a wireless transmission the signal attenuation increases approximately with the square of the distance between transmitter and receiver. That consists in a serious challenge on the field, which give rise to a number of undesired phenomena, such as unsuccessful communication or presence of random delays, that must be taken into proper account at controller design time.
An additional difficulty that follows the from large scale of system\(^1\) is the direct proportionality with plant complexity. In fact, the big size of the control system give rise to the well known “curse of dimensionality” for the control problem, especially when the control law computation is based on model-oriented approaches which solve an optimization problem on-line.

Furthermore, in almost all real plants, actuator saturations or safety policies impose some sort of constraint on plant variables, therefore making it crucial to have a constraint-handling controller. For this reason Model Predictive Control (MPC) use on industry is increasing over the last decades, thanks to its capability of handling constraints and guarantee an improved performance with respect to classical approaches such as PIDs.

The general control orientation of the WIDE Toolbox is MPC, which however suffer the aforementioned “curse of dimensionality”. Moreover, it is hard to collect measurements reliably in a single point (the controller physical location) due to the aforementioned network effects. Even if such information gathering succeeds the control action computation must take place “in time”, i.e. within the controller sample period, which is difficult because of the optimization problem complexity inherited from the plant size. To cope with this and successfully apply MPC to large-scale systems, we propose decentralized/distributed versions of MPC in which the main idea is to decompose the plant model in a number of (as much as possible) independent subsystem.

Advantages of this approach lie in the smaller size of each control problem, i.e. optimization problem to be solved online, consequently reducing computation times. Moreover, the independency of each controller from the others allows control computation in parallel, hence exploiting the current trend of computer manufacturer\(^2\) furthering improving performances. Since each subsystem needs only a subset of measurements in order to compute its control

---

\(^1\) The “scale” is generally intended as the number input and output variables, which, consequently, is related to the number of state variables. A plant is usually said to be large if there are tens or more of variables.

\(^2\) Accounting for the fall of the Moore’s law, which predicts an increase of CPU frequency over the years, hardware solution to keep improving performances rest in the increase of core and processors. Naturally, the total computation power increases, but the challenge of parallel computation is not trivial to solve as many known algorithms are intrinsically serial. Current implementations of algorithms for solving optimization problems cannot efficiently exploit multiple computing units.
action (typically from “near-by” sensors) also the communication problem is softened.

However, challenges arise from lack of centrality. The real coupling between logical subsystem that is not taken into account into the subsystem model (neglected to perform the decentralization), practically gives to the controller some not precise information about the underlying dynamics. This can lead to serious consequences, such as loss of stability properties and/or inability of handle inter-subsystem constraints.

In order to cope with this latter issues, two main ideas have been considered in literature: i) consensus iterative algorithms, ii) supervisor approaches.

The first is used in distributed control and consists in each controller to compute its control action and then share with the others some information (information type characterize the various approaches) iteratively, until some agreement is reached. Naturally, performances are improved with respect to decentralized in which each controller acts fully independently. However, the whole set of consensus iteration must take place within the sample time, so as to have the control action ready to be sent to actuators. This increases the sample time, making the controller to be less reactive.

In the second approach a supervisor coordinates the underlying controllers, typically with a lower frequency, so that the only communication that have to take place is between the supervisor and each controller (controllers does not communicate with each other). The supervisor exploits a model that allows a general overview of the whole plant but with slower dynamics (the fast ones are handles by the lower layer controllers), that is consequently smaller than the full model (model reduction techniques may remove unnecessary states which previously modeled the now neglected fast dynamics). In fact, such model is obtained by the closed-loop of plant with lower layer controllers. Also in this case the improvement with respect to decentralized approach is the capability of viewing the system as a whole, but reactiveness is the issue as the supervisors runs at a lower pace then the plant.

The toolbox is organized in three main areas, dealing with previously listed issues:
A. WIDE Toolbox

• DHMPC (developed by UNISI$^3$, UNITN$^4$ and IMTL$^5$) offer to the user decentralized control strategies that account for some major network effects (dlincon, HiMPC, decLMI), facilities for wireless network simulation using TrueTime (ACG) and energy aware control strategies (eampc);

• LSMM (developed by HPL$^6$) is mostly oriented to model and analyze a large-scale system by providing standard analysis functionalities (LSmodel), e.g. Bode and Nyquist diagrams, and decentralization procedure ($\epsilon$-decomposition), with particular focus to water networks (WNmodel) and network aware Kalman filtering (NKF);

• NCS (developed by TU/e$^7$) is concerned with network effects which are dealt with using discretized and hybrid models (ncs) and network linear controller synthesis available also with a Graphical User Interface.

This Appendix is focused on implementations of control approaches described in Chapters 2, 3 and 4, that are reported in Sections A.2, A.3 and A.4, respectively. Class descriptions and examples are reported with source code and Matlab output, with the aim of improve the approaches applicability presentation. The final Section is devoted to Networked Control System quick simulation setup that uses the automatic code generation for TrueTime functionality.

Verbatim black font text is used for Matlab code, while grey color indicates Matlab response to the code execution.

A.2 Decentralized Linear Control (decLMI)

The class decLMI provides a tool to synthesize a stabilizing decentralized linear controller for discrete-time LTI systems with time-varying connection topology. Functionalities are given to achieve either robust or stochastic convergence to the origin, as described in Chapter 2.

---

$^3$University of Siena  
$^4$University of Trento  
$^5$Institution Market Technologies - Institute for Advanced Studies Lucca  
$^6$Honeywell Prague  
$^7$Eindhoven University of Technology
A.2. Decentralized Linear Control (decLMI)

A.2.1 Class description

A decLMI object is composed of: network topology, \textit{i.e.} the adjacency matrix of the graph describing sensor-to-node links; matrices A and B of the LTI discrete-time state-space representation of system dynamics, \textit{i.e.} \( x(t + 1) = Ax(t) + Bu(t) \); state and input weights used in the Lyapunov function decrease rate condition (see Chapter 2 for details); state and input constraints; initial state uncertainty, set of vertices of the uncertain polytope; Markov chain modeling packet dropouts (needed for stochastic stability only).

The goal of the class is to synthesize a stabilizing matrix control gain of the form \( u(t) = Kx(t) \) as the solution of a SDP problem. Four types of stability are considered, based on the amount of available information, \textit{i.e.} centralized or decentralized, and based on the treatment of network uncertainties, \textit{i.e.} robust or stochastic. The resulting type of stability for the closed-loop are: Centralized, the network is assumed fully connected, all links are reliable and faultless (\textit{i.e.}, no packet dropouts are considered); Decentralized Ideal, the network is only partially connected, all links are reliable and faultless; Decentralized Robust, the network is only partially connected and some links are faulty, \textit{i.e.}, packets transmitted in those links can be lost; Decentralized Stochastic, the network is only partially connected, and some links are faulty and are modeled by a Markov chain, guaranteeing stability in the mean square sense.

The objects are created using the class constructor, but no controller is immediately available. Appropriate methods should be invoked to compute relative gains, which are then stored in the object property \( K \), a structure with fields: \( ci \), centralized ideal; \( di \), decentralized ideal; \( dl \), decentralized lossy, \textit{i.e.} robust; \( ds \), decentralized stochastic.

In order to compute the stochastic gain, a Markov chain must be provided in the constructor invocation. Relative informations are stored in the \( Mc \) structure which has fields \( T \) and \( E \), which are Markov chain transition and emission matrices, respectively.

A.2.2 Methods

Class constructor creates the decLMI object with the syntax

\[
\text{obj} = \text{decLMI(Net}, A, B, Qx, Qu, X0, xmax, umax, Mc)\]

where the inputs are:
1. **Net**: matrix of row size equal to number of actuators and column size equal to number of states. Net\(i, j\) is either:
   - 1: if the state \(j\) is connected to actuator \(i\) by an ideal link;
   - 0: if there is no link between state \(j\) and actuator \(i\);
   - -1: if the state \(j\) is connected to actuator \(i\) by a lossy link;
   
   If left empty determine a full connected graph.

2. **A**: state matrix of the LTI system modeling the plant;

3. **B**: input matrix of the LTI system modeling the plant;

4. **Qx**: [optional] state weight matrix of the Lyapunov function decay condition, default value is identity;

5. **Qu**: [optional] input weight matrix of the Lyapunov function decay condition, default value is identity;

6. **X0**: [optional] set of vertices of the polytope that defines the uncertainty of the initial state condition;

7. **xmax**: [optional] Euclidean norm constraint on state if scalar or vector (dimension of state) of element-wise norm state constraints;

8. **umax**: [optional] Euclidean norm constraint on input if scalar or vector (dimension of input) of element-wise norm inout constraints;

9. **Mc**: [optional] two-states Markov chain that models the probability of losing a packet. Must be a structure with fields:
   - \(d\): array, where \(d(i)\) is the probability of losing a packet being in the \(i\)-th state of the Markov chain;
   - \(q\): array, where \(q(i)\) is the probability of remaining in the \(i\)-th state of the Markov chain.

Available methods for objects of class decLMI are:
A.2. Decentralized Linear Control (decLMI)

- **solve_centralized_lmi**: Computes the solution of the SDP problem assuming the network to be fully connected and each link completely reliable. The goal is to give a reference for the performances that can be achieved via the decentralized methods;

- **solve_dec_ideal_lmi**: Computes the solution considering only present links, but assuming that all of them are reliable. Basically does not account for miss-reception of packets containing measurements;

- **solve_dec_lossy_lmi**: Computes a decentralized controller which provides stability and constraints satisfaction for any possible occurrence of the packet dropouts;

- **solve_dec_stoch_lmi**: Computes a decentralized controller that exploits the available knowledge on the dropouts probability distributions. A two-states Markov chain is used to model the packet losses. Mean-square stability is guaranteed.

A.2.3 Example

In the following, the example code demonstrating the use of decLMI class is presented. The network topology is fixed and depicted in Figure A.1.

```matlab
% The variable Net describes how the network is connected.
% e f g h j k l m
Net = [ 1 1 1 0 0 -1 0 0 % a
       -1 1 0 1 1 0 0 0 % b
       0 0 1 0 1 1 -1 % c
       0 0 0 -1 1 0 1 1 ]; % d
[m,n]=size(Net);

% State space matrices of the LTI discrete-time model representation
A = [0.2863 -0.2895 0.1268 -0.0624 -0.2098 0.1882 -0.0061 0.2227;
     -0.2729 0.5052 0.0129 -0.1057 -0.1361 0.2154 0.2799 -0.0183;
     0.1610 -0.1229 0.1829 -0.4267 0.0589 -0.0905 -0.0417 -0.1709;
     -0.1084 -0.0138 -0.4611 -0.0735 -0.2013 0.1463 0.0854 -0.1567;
     -0.0300 -0.1167 -0.0012 -0.1847 0.3547 0.2081 0.0567 0.1566;
     0.2852 0.3036 -0.0994 0.1240 0.1099 0.1262 0.0229 -0.1599;
     -0.0032 0.1558 -0.1285 0.1898 -0.0040 0.0017 0.7065 -0.1697;
     0.1961 -0.0173 -0.1023 -0.1888 0.0327 -0.2866 -0.1202 0.3196];
B = [ 0 -2.2374 -0.4531 0;
     -0.1794 1.0976 1.3996 0.4287;
     -1.4671 0 -0.4620 -0.7370;
     1.3953 -1.6146 0.0327 0.5649];
```
Figure A.1: decLMI example network topology

\[
\begin{bmatrix}
0.4408 & -1.2287 & 0.7888 & -1.3842 \\
0.5654 & 0.2074 & 0.8968 & 0.4603 \\
0 & 0 & 0.1379 & 0 \\
0 & -1.0061 & 0 & 0.3798
\end{bmatrix};
\]

% Sample time
Ts = 1;

Constraints on state $\sqrt{x'x} \leq x_{\text{max}}$ and input $\sqrt{u'u} \leq u_{\text{max}}$ and relative weights in the LMI are

\[
\begin{align*}
x_{\text{max}} &= 150; \\
u_{\text{max}} &= 150; \\
Q_x &= \text{eye}(n); \\
Q_u &= 1e^{-2}\text{eye}(m);
\end{align*}
\]

Vertices of the initial state uncertainty polytope are

\[
X_0 = [
\begin{bmatrix}
7.7618 & 7.7618 & 8.7618 & 8.7618 \\
6.0485 & 7.0485 & 6.0485 & 7.0485 \\
6.2866 & 6.2866 & 6.2866 & 6.2866 \\
6.6018 & 6.6018 & 6.6018 & 6.6018 \\
5.1547 & 5.1547 & 5.1547 & 5.1547 \\
7.8859 & 7.8859 & 7.8859 & 7.8859 \\
5.1507 & 5.1507 & 5.1507 & 5.1507 \\
5.0470 & 5.0470 & 5.0470 & 5.0470
\end{bmatrix};
\]
A.2. Decentralized Linear Control (decLMI)

The two-state Markov chain is described by fields \( d \), array where \( d(i) \) is the probability of losing a packet being in the \( i \)-th state of the Markov chain and field \( q \), array where \( q(i) \) is the probability of of remaining in the \( i \)-th state of the Markov chain.

\[
\begin{align*}
\text{M.c.} \cdot d & = [.1 .5]; \\
\text{M.c.} \cdot q & = [.8 .5];
\end{align*}
\]

The decLMI object is created by

\[
\text{obj} = \text{decLMI} (\text{Net}, A, B, Qx, X0, xmax, umax, \text{M.c});
\]

In the sequel all the controller synthesis problems are formulated and solved. The time needed for each computation is stored for comparison purposes.

Centralized ideal

tic
\[
\text{obj}=\text{obj.solve\_centralized\_lmi}();
\]

CPUtime.centralized = toc;

Solving centralized SDP problem...

Optimization started


Alg = 2: xz-corrector, theta = 0.250, beta = 0.500

eps \( m = 89 \), order \( n = 101 \), dim = 1575, blocks = 9

\[
\text{max(A)} = 1.972 \times 10^4, \text{max(ADA)} = 4761, \text{max(L)} = 2415
\]

\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
\text{it} & \text{con} & \text{delta} & \text{rate} & \text{t/P} & \text{t/D} & \text{feas} & \text{cg} & \text{prec} \\
\hline
0 & 7.53E+04 & 0.000 & & & & & & \\
1 & -1.8E+04 & 2.49E+04 & 0.000 & 0.3311 & 0.9000 & 0.9000 & 1.07 & 1 \ 8.5E+00 \\
2 & -1.8E+04 & 6.62E+03 & 0.000 & 0.2654 & 0.9000 & 0.9000 & 1.41 & 1 \ 2.8E+00 \\
3 & -5.1E+03 & 2.68E+03 & 0.000 & 0.4016 & 0.9000 & 0.9000 & 2.59 & 1 \ 7.4E-01 \\
4 & -4.4E+02 & 6.18E+02 & 0.000 & 0.2299 & 0.9000 & 0.9000 & 2.61 & 1 \ 3.3E-01 \\
5 & -1.2E+02 & 1.50E+02 & 0.000 & 0.4499 & 0.9000 & 0.9000 & 1.20 & 1 \ 2.8E+00 \\
6 & -7.9E+01 & 8.63E+01 & 0.000 & 0.5769 & 0.9000 & 0.9000 & 0.98 & 1 \ 2.2E+00 \\
7 & -1.3E+02 & 4.53E+01 & 0.000 & 0.5247 & 0.9000 & 0.9000 & 0.39 & 1 \ 1.4E-01 \\
8 & -1.77E+02 & 1.70E+01 & 0.000 & 0.3762 & 0.9000 & 0.9000 & 0.43 & 1 \ 6.9E-02 \\
9 & -1.92E+02 & 1.18E+01 & 0.008 & 0.6939 & 0.9000 & 0.9000 & 0.47 & 1 \ 5.6E-02 \\
10 & -2.25E+02 & 6.08E+00 & 0.000 & 0.5095 & 0.9000 & 0.9000 & 0.17 & 1 \ 5.1E-02 \\
11 & -2.47E+02 & 3.89E+00 & 0.000 & 0.6483 & 0.9000 & 0.9000 & 0.04 & 1 \ 5.1E-02 \\
12 & -3.03E+02 & 1.59E+00 & 0.000 & 0.4079 & 0.9000 & 0.9000 & 0.17 & 1 \ 3.1E-02 \\
13 & -3.32E+02 & 8.80E-01 & 0.000 & 0.5545 & 0.9000 & 0.9000 & 0.31 & 1 \ 2.5E-02 \\
14 & -3.60E+02 & 4.68E-01 & 0.000 & 0.5296 & 0.9000 & 0.9000 & 0.42 & 1 \ 1.8E-02 \\
15 & -3.82E+02 & 2.68E-01 & 0.000 & 0.5668 & 0.9000 & 0.9000 & 0.55 & 1 \ 1.2E-02 \\
16 & -3.98E+02 & 1.47E-01 & 0.000 & 0.5652 & 0.9000 & 0.9000 & 0.66 & 1 \ 8.2E-03 \\
17 & -4.10E+02 & 7.85E-02 & 0.000 & 0.5347 & 0.9000 & 0.9000 & 0.76 & 1 \ 5.0E-03 \\
18 & -4.19E+02 & 3.92E-02 & 0.000 & 0.5001 & 0.9000 & 0.9000 & 0.84 & 1 \ 2.7E-03 \\
19 & -4.24E+02 & 1.71E-02 & 0.000 & 0.4346 & 0.9000 & 0.9000 & 0.91 & 1 \ 1.3E-03 \\
20 & -4.27E+02 & 5.79E-03 & 0.000 & 0.3394 & 0.9000 & 0.9000 & 0.95 & 1 \ 4.4E-04 \\
21 & -4.28E+02 & 1.32E-03 & 0.000 & 0.2271 & 0.9000 & 0.9000 & 0.98 & 1 \ 1.0E-04 \\
22 & -4.29E+02 & 8.26E-05 & 0.192 & 0.0628 & 0.9000 & 0.9000 & 0.99 & 1 \ 6.5E-06 \\
23 & -4.29E+02 & 4.75E-06 & 0.000 & 0.0575 & 0.9000 & 0.9000 & 1.00 & 1 \ 3.7E-07 \\
24 & -4.29E+02 & 1.56E-08 & 0.398 & 0.0033 & 0.9990 & 0.9990 & 1.00 & 4 \ 1.2E-09 \\
\hline
\end{array}
\]

Run into numerical problems.
Figure A.2: Open-loop (blue stars) and closed-loop (red circles) eigenvalues of the plant with centralized ideal feedback.

<table>
<thead>
<tr>
<th>iter</th>
<th>seconds</th>
<th>digits</th>
<th>c*</th>
<th>b*</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>1.1</td>
<td>-9.3</td>
<td>-4.286543504e+02</td>
<td>-4.286543527e+02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.6e+11</td>
<td>1.1E-08,</td>
</tr>
<tr>
<td>Detailed timing (sec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre</td>
<td>IPM</td>
<td>Post</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.7E-02</td>
<td>6.54E-01</td>
<td>3.76E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max-norms:</td>
<td></td>
<td>b</td>
<td></td>
<td>=1,</td>
</tr>
<tr>
<td>Cholesky</td>
<td>add = 1,</td>
<td>skip = 10,</td>
<td></td>
<td>L.L</td>
</tr>
<tr>
<td>Centralized SDP problem solved!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Decentralized ideal

tic
obj = obj.solve_dec_ideal_lmi();
CPUtime.dec_ideal = toc;
### A.2. Decentralized Linear Control (decLMI)

Solving decentralized SDP problem...


Alg = 2: xz-corrector, theta = 0.250, beta = 0.500

eqs m = 29, order n = 101, dim = 1573, blocks = 9

\[ \text{nnz}(A) = 616 + 0, \text{nnz}(ADA) = 841, \text{nnz}(L) = 435 \]

<table>
<thead>
<tr>
<th>it</th>
<th>b*y</th>
<th>gap</th>
<th>delta</th>
<th>rate</th>
<th>t/tP*</th>
<th>t/tD*</th>
<th>feas</th>
<th>cg</th>
<th>prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7.5E+04</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-1.76E+04</td>
<td>2.43E+04</td>
<td>0.000</td>
<td>0.3227</td>
<td>0.9000</td>
<td>0.9000</td>
<td>1.07</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>-1.02E+04</td>
<td>6.63E+03</td>
<td>0.000</td>
<td>0.2727</td>
<td>0.9000</td>
<td>0.9000</td>
<td>1.64</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>-5.30E+03</td>
<td>2.85E+03</td>
<td>0.000</td>
<td>0.2639</td>
<td>0.9000</td>
<td>0.9000</td>
<td>2.68</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>-4.06E+02</td>
<td>5.81E+02</td>
<td>0.000</td>
<td>0.2038</td>
<td>0.9000</td>
<td>0.9000</td>
<td>2.43</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>-1.02E+02</td>
<td>1.51E+02</td>
<td>0.000</td>
<td>0.2589</td>
<td>0.9000</td>
<td>0.9000</td>
<td>1.14</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>-8.58E+01</td>
<td>8.72E+01</td>
<td>0.000</td>
<td>0.2593</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.96</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>-3.34E+02</td>
<td>5.06E+01</td>
<td>0.000</td>
<td>0.5807</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.35</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>-1.94E+02</td>
<td>1.77E+01</td>
<td>0.226</td>
<td>0.3498</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.38</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>-2.12E+02</td>
<td>1.25E+01</td>
<td>0.280</td>
<td>0.7077</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.47</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>-2.26E+02</td>
<td>9.78E+00</td>
<td>0.000</td>
<td>0.7492</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.23</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>-2.34E+02</td>
<td>7.31E+00</td>
<td>0.000</td>
<td>0.7692</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.31</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>-2.95E+02</td>
<td>3.23E+00</td>
<td>0.000</td>
<td>0.4419</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.15</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>-3.83E+02</td>
<td>1.02E+00</td>
<td>0.000</td>
<td>0.3148</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.18</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>-4.30E+02</td>
<td>4.25E-01</td>
<td>0.000</td>
<td>0.4176</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.39</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>-4.54E+02</td>
<td>2.26E-01</td>
<td>0.000</td>
<td>0.5300</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.47</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>-4.87E+02</td>
<td>8.14E-02</td>
<td>0.000</td>
<td>0.3637</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.64</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>-5.08E+02</td>
<td>1.18E-02</td>
<td>0.000</td>
<td>0.3053</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.90</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>-5.12E+02</td>
<td>1.11E-02</td>
<td>0.000</td>
<td>0.3078</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>-5.12E+02</td>
<td>3.43E-03</td>
<td>0.000</td>
<td>0.3083</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.97</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>-5.12E+02</td>
<td>3.22E-04</td>
<td>0.000</td>
<td>0.3800</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.98</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>-5.12E+02</td>
<td>3.68E-05</td>
<td>0.000</td>
<td>0.2042</td>
<td>0.9000</td>
<td>0.9000</td>
<td>1.00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>-5.12E+02</td>
<td>2.16E-07</td>
<td>0.000</td>
<td>0.0035</td>
<td>0.9999</td>
<td>0.9999</td>
<td>1.00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>-5.12E+02</td>
<td>5.16E-08</td>
<td>0.000</td>
<td>0.0000</td>
<td>0.9999</td>
<td>0.9999</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>-5.12E+02</td>
<td>1.39E-09</td>
<td>0.000</td>
<td>0.0000</td>
<td>0.9999</td>
<td>0.9999</td>
<td>0.00</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

iter seconds digits c*b y

```
24 0.7 10.3 -5.1298252214e+02 -5.1298252217e+02
```

\[ |\text{Ax-b}| = 2.9e-12, |\text{Ay-c}|_+ = 1.1E-09, |x| = 5.1e+02, |y| = 9.8e+02 \]

Decentralized SDP problem solved!

Comparing Figures A.2 and A.3 one may notice that closed loop eigenvalues in case of centralized control are all real and generally close to the origin. This indicates a good degree of intuitive stability, i.e. the greater the distance between each eigenvalue and the unitary circumference, the lower the likelihood of instability due to modeling error.

### Decentralized lossy (robust)

```
tic
obj = obj.solve_dec_lossy_lmi();
CPUtime.dec_lossy = toc;
```

Solving dec. lossy SDP problem...
Figure A.3: Open-loop (blue stars) and closed-loop (red circles) eigenvalues of the plant with decentralized ideal feedback

Alg = 2: xz-corrector, theta = 0.250, beta = 0.500
eqs m = 37, order n = 941, dim = 19333, blocks = 54
nnz(A) = 606e + 0, nnz(ADA) = 1273, nnz(L) = 691

<table>
<thead>
<tr>
<th>it</th>
<th>b*y</th>
<th>gap</th>
<th>delta</th>
<th>rate</th>
<th>t/tP*</th>
<th>t/tD*</th>
<th>feas</th>
<th>cg</th>
<th>prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.78E+06</td>
<td>0.000</td>
<td></td>
<td></td>
<td>1.01</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>1</td>
<td>-1.93E+04</td>
<td>4.80E+05</td>
<td>0.000</td>
<td>0.2692</td>
<td>0.9000</td>
<td>0.9000</td>
<td>1.01</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>-3.01E+04</td>
<td>4.34E+04</td>
<td>0.000</td>
<td>0.0906</td>
<td>0.9900</td>
<td>0.9900</td>
<td>1.14</td>
<td>1.00</td>
<td>9.3E+00</td>
</tr>
<tr>
<td>3</td>
<td>-2.40E+04</td>
<td>1.34E+04</td>
<td>0.000</td>
<td>0.3093</td>
<td>0.9000</td>
<td>0.9000</td>
<td>1.36</td>
<td>1.00</td>
<td>3.3E+00</td>
</tr>
<tr>
<td>4</td>
<td>-6.26E+03</td>
<td>4.85E+03</td>
<td>0.000</td>
<td>0.3608</td>
<td>0.9000</td>
<td>0.9000</td>
<td>2.43</td>
<td>1.00</td>
<td>7.9E-01</td>
</tr>
<tr>
<td>5</td>
<td>-7.24E+02</td>
<td>1.03E+03</td>
<td>0.000</td>
<td>0.2117</td>
<td>0.9000</td>
<td>0.9000</td>
<td>1.99</td>
<td>1.00</td>
<td>4.1E-01</td>
</tr>
<tr>
<td>6</td>
<td>-3.94E+02</td>
<td>5.58E+02</td>
<td>0.000</td>
<td>0.5441</td>
<td>0.9000</td>
<td>0.9000</td>
<td>1.13</td>
<td>1.00</td>
<td>3.7E-01</td>
</tr>
<tr>
<td>7</td>
<td>-3.69E+02</td>
<td>4.89E+02</td>
<td>0.000</td>
<td>0.8763</td>
<td>0.9000</td>
<td>0.9000</td>
<td>1.01</td>
<td>1.00</td>
<td>3.5E-01</td>
</tr>
<tr>
<td>8</td>
<td>-4.46E+02</td>
<td>5.38E+02</td>
<td>0.000</td>
<td>0.7310</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.55</td>
<td>1.00</td>
<td>2.7E-01</td>
</tr>
<tr>
<td>9</td>
<td>-5.19E+02</td>
<td>2.28E+02</td>
<td>0.000</td>
<td>0.6362</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.56</td>
<td>1.00</td>
<td>1.9E-01</td>
</tr>
<tr>
<td>10</td>
<td>-5.63E+02</td>
<td>4.68E+02</td>
<td>0.000</td>
<td>0.6404</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.64</td>
<td>1.00</td>
<td>1.3E-01</td>
</tr>
<tr>
<td>11</td>
<td>-5.74E+02</td>
<td>1.30E+02</td>
<td>0.133</td>
<td>0.8925</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.60</td>
<td>1.00</td>
<td>1.2E-01</td>
</tr>
<tr>
<td>12</td>
<td>-6.23E+02</td>
<td>8.48E+01</td>
<td>0.000</td>
<td>0.6501</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.35</td>
<td>1.00</td>
<td>1.1E-01</td>
</tr>
<tr>
<td>13</td>
<td>-6.74E+02</td>
<td>5.40E+01</td>
<td>0.000</td>
<td>0.6397</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.29</td>
<td>1.00</td>
<td>1.8E-02</td>
</tr>
<tr>
<td>14</td>
<td>-7.84E+02</td>
<td>1.85E+01</td>
<td>0.000</td>
<td>0.3432</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.54</td>
<td>1.00</td>
<td>3.5E-02</td>
</tr>
<tr>
<td>15</td>
<td>-8.24E+02</td>
<td>8.69E+00</td>
<td>0.000</td>
<td>0.4686</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.67</td>
<td>1.00</td>
<td>2.0E-02</td>
</tr>
<tr>
<td>16</td>
<td>-8.49E+02</td>
<td>4.24E+00</td>
<td>0.000</td>
<td>0.4686</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.78</td>
<td>1.00</td>
<td>1.1E-02</td>
</tr>
<tr>
<td>17</td>
<td>-8.69E+02</td>
<td>9.78E-01</td>
<td>0.000</td>
<td>0.2305</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.89</td>
<td>1.00</td>
<td>2.7E-03</td>
</tr>
<tr>
<td>18</td>
<td>-8.76E+02</td>
<td>2.06E-01</td>
<td>0.000</td>
<td>0.2101</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.96</td>
<td>1.00</td>
<td>5.8E-04</td>
</tr>
<tr>
<td>19</td>
<td>-8.76E+02</td>
<td>5.66E-02</td>
<td>0.000</td>
<td>0.2762</td>
<td>0.9000</td>
<td>0.9000</td>
<td>0.99</td>
<td>1.00</td>
<td>1.6E-04</td>
</tr>
</tbody>
</table>
A.2. Decentralized Linear Control (decLMI)

20 : -8.76E+02 1.91E-02 0.000 0.3359 0.9000 0.9000 1.00 1 1 5.5E-05
21 : -8.76E+02 4.35E-03 0.000 0.2283 0.9000 0.9000 1.00 1 1 2.5E-05
22 : -8.76E+02 8.86E-04 0.000 0.2035 0.9000 0.9000 1.00 1 1 2.5E-05
23 : -8.76E+02 6.95E-05 0.000 0.2035 0.9000 0.9000 1.00 1 1 2.5E-05
24 : -8.76E+02 1.73E-05 0.000 0.2035 0.9000 0.9000 1.00 7 7 4.8E-09
25 : -8.76E+02 1.67E-06 0.433 0.0967 0.9900 0.9900 1.00 15 16 4.8E-09
26 : -8.76E+02 4.25E-07 0.000 0.2541 0.9000 0.9000 1.00 26 26 1.2E-09

Run into numerical problems.

iter seconds digits c*x b*y
7.9 8.8 -8.7629587322e+02 -8.7629587466e+02

|Ax-b| = 9.7e-11, [Ay-c]_+ = 3.8E-09, |x|= 8.8e+02, |y|= 1.4e+03

Detailed timing (sec)

Pre Post
7.343e-03 4.935e+00 6.478e-03

Max-norms: ||b||=1, ||c|| = 22500,
Cholesky |add|=5, |skip| = 5, ||L.L|| = 96443.6.

Closed loop max eig: 0.76383
Closed loop max eig: 0.76245
Closed loop max eig: 0.76016
Closed loop max eig: 0.75831
Closed loop max eig: 0.76156
Closed loop max eig: 0.76035
Closed loop max eig: 0.76016
Closed loop max eig: 0.76035
Closed loop max eig: 0.76016
Closed loop max eig: 0.76035
Closed loop max eig: 0.76016
Closed loop max eig: 0.76035
Closed loop max eig: 0.76016
Closed loop max eig: 0.76035
Closed loop max eig: 0.76016
Closed loop max eig: 0.76035
Closed loop max eig: 0.76016
Closed loop max eig: 0.76035
Closed loop max eig: 0.76016
Closed loop max eig: 0.76035
Closed loop max eig: 0.76016
Closed loop max eig: 0.76035

Figure A.4 depicts in red circles the closed-loop eigenvalues of all possible realization of the plant, i.e. all the combination of network topology which may occur. It also shows that all of them are within the unitary circle, demonstrating closed-loop stability in any possible occurrence of successful receptions over time.

Decentralized stochastic

tic
obj = obj.solve_dec_stoch_lmi();
CPUtime.dec_stoch = toc;

Solving decentralized stochastic SDP problem...

Alg = 2: xz-corrector, theta = 0.250, beta = 0.500
Put 2 free variables in a quadratic cone

eqs = 47, order n = 814, dim = 227311, blocks = 17
max(l) = 11569 + 0, max(ADA) = 2029, max(L) = 1038
it : b*y gap delta rate t/tP* t/tD* feas cg cg prec
0 : 1.73E+07 9.000
1 : 1.89E+06 4.82E+06 0.000 0.2788 0.9000 0.9000 1.00 1 1 7.6E+01
Figure A.4: Open-loop (blue stars) and closed-loop (red circles) eigenvalues of the plant with robust decentralized feedback.

Decentralized lossy feedback: Eigenvalues

2 : 7.96E+05 2.47E+06 0.000 0.5121 0.9000 0.900 1.09 1 1 3.8E+01
3 : 4.80E+05 1.70E+06 0.000 0.6886 0.9000 0.900 1.03 1 1 2.7E+01
4 : 2.04E+05 8.57E+05 0.000 0.5042 0.9000 0.900 1.01 1 1 1.4E+01
5 : 4.11E+04 2.99E+05 0.000 0.3686 0.9000 0.900 1.01 1 1 5.4E+00
6 : 1.19E+04 7.23E+04 0.000 0.2418 0.9000 0.900 1.01 1 1 2.0E+00
7 : 3.64E+04 2.76E+03 0.000 0.0383 0.9000 0.900 1.01 1 1 1.7E+00
8 : 1.40E+04 1.68E+03 0.000 0.1287 0.9000 0.900 2.38 1 1 4.3E-01
9 : 3.23E+03 1.12E+03 0.000 0.7645 0.9000 0.900 8.01 1 1 1.3E-01
10 : 9.29E+02 6.92E+02 0.000 0.6197 0.9000 0.900 3.93 1 1 7.3E-02
11 : 8.25E+02 5.66E+02 0.000 0.8172 0.9000 0.900 1.86 1 1 6.1E-02
12 : 8.76E+02 4.14E+02 0.000 0.7310 0.9000 0.900 1.15 1 1 4.4E-02
13 : 8.92E+02 1.88E+02 0.000 0.4505 0.9000 0.900 1.11 1 1 1.9E-02
14 : 8.81E+02 8.04E+01 0.000 0.4317 0.9000 0.900 1.04 1 1 8.4E-03
15 : 8.77E+02 2.08E+01 0.000 0.2491 0.9000 0.900 1.01 1 1 2.1E-03
16 : 8.76E+02 5.72E+00 0.000 0.2855 0.9000 0.900 1.00 1 1 6.0E-04
17 : 8.76E+02 2.77E+00 0.000 0.4846 0.9000 0.900 1.00 1 1 2.9E-04
18 : 8.76E+02 1.50E+00 0.000 0.5409 0.9000 0.900 1.00 1 1 1.6E-04
19 : 8.76E+02 4.18E+00 0.000 0.5607 0.9000 0.900 1.00 1 1 8.3E-05
20 : 8.76E+02 4.52E+00 0.000 0.5377 0.9000 0.900 1.00 1 1 4.7E-05
21 : 8.76E+02 2.41E+00 0.000 0.5339 0.9000 0.900 1.00 1 1 2.5E-05
22 : 8.76E+02 1.08E+00 0.000 0.4397 0.9000 0.900 1.00 1 1 1.1E-05
23 : 8.76E+02 3.76E+00 0.000 0.3544 0.9000 0.900 1.00 1 1 3.9E-06
24 : 8.76E+02 8.62E+00 0.000 0.2293 0.9000 0.900 1.00 2 2 9.1E-07
25 : 8.76E+02 1.88E+00 0.000 0.2176 0.9000 0.900 1.00 2 2 2.0E-07
26 : 8.76E+02 4.06E+00 0.000 0.2184 0.9000 0.900 1.00 3 3 4.3E-08
Figure A.5: Open-loop (blue stars) and closed-loop (red circles) eigenvalues of the plant with stochastic decentralized feedback.

Figure A.5 depicts, similarly to Figure A.4, the closed-loop eigenvalues of the whole set of possibilities of network topology realization, but in both the Markov chain states. However, the majority of eigenvalues are overlapping (graphically indicated by thick red circles) and less spreaded with respect to the robust case. This is due to the averaged formulation which does not
consider the worst case but the mean instead. As we shall see, this will lead to improved performances.

**Simulation**

In this subsection the code for computing 50 simulations of the aforementioned plant is reported. Each simulation uses the same networked topologies, but effectively received packets varies from a simulation to the other, according the the Markov chain model. The goal is to demonstrate the four controllers behaviors in a wide spread of situations.

```matlab
% Initial state
x0 = [8.3272, 5.7953, 7.2282, 8.6640, 5.8317, 6.6488, 8.1784, 6.5376];
% Simulation length
Tsim = 50;
% Number of simulations to be carried out
Nsim = 50;
% A sequence of states and emissions of the Markov chain model is stored.
load Statechange_paper
load Randvals_paper
Sim = [];
for iter=1:Nsim
    % Generate realizations of Markov chain
    statechange = Statechange(1,:,iter);
    randvals = Randvals(1,:,iter);
    % Initial Markov chain state
    initialstate = 1;
    % Compute sequence of Markov chain states
    trc = cumsum(obj.Mc.T,2);
    ec = cumsum(obj.Mc.E,2);
    currentstate = initialstate;
    numStates = size(obj.Mc.T,1);
    numEmissions = size(obj.Mc.E,2);
    for i = 1:Tsim
        % Calculate state transition
        stateVal = statechange(i);
        state = 1;
        for innerState = numStates-1:-1:1
            if stateVal > trc(currentstate,innerState)
                state = innerState + 1;
                break
            end
        end
        % Calculate emission
        val = randvals(i);
        emit = 1;
    end
```

for inner = numEmissions-1:-1:1
    if val > ec(state,inner)
        emit = inner + 1;
        break
    end
end
% Add values and states to output
seq(i) = emit;
states(i) = state;
currentstate = state;
end

Get controllers gains from decLMI object
Kc = obj.K.ci;
Kd = obj.K.di;
Kl = obj.K.dl;
Ks = obj.K.ds;

Init states, inputs, state norms and input norms
Xc = x0; Xd = x0; Xn = x0; Xl = x0; Xs = x0;
Uc = []; Ud = []; Un = []; Ul = []; Us = [];
Xcnorm = []; Xdnorm = []; Xnnorm = []; Xlnorm = []; Xsnorm = [];
Ucnorm = []; Udnorm = []; Unnorm = []; Ulnorm = []; Usnorm = [];

Perform current simulation
for i=1:Tsim
    % Centralized
    Uc(:,i) = Kc*Xc(:,i);
    Xc(:,i+1) = A*Xc(:,i) + B*Uc(:,i);
    Xcnorm(i) = norm(Xc(:,i));
    Ucnorm(i) = norm(Uc(:,i));
    % Decentralized with ideal network
    Ud(:,i) = Kd*Xd(:,i);
    Xd(:,i+1) = A*Xd(:,i) + B*Ud(:,i);
    Xdnorm(i) = norm(Xd(:,i));
    Udnorm(i) = norm(Ud(:,i));
    % Decentralized with lossy network and robust stability
    Ul(:,i) = Kl{seq(i)}*Xl(:,i);
    Xl(:,i+1) = A*Xl(:,i) + B*Ul(:,i);
    Xlnorm(i) = norm(Xl(:,i));
    Ulnorm(i) = norm(Ul(:,i));
    % Decentralized with lossy network and stochastic stability
    Us(:,i) = Ks{seq(i),states(i)}*Xs(:,i);
    Xs(:,i+1) = A*Xs(:,i) + B*Us(:,i);
    Xsnorm(i) = norm(Xs(:,i));
    Usnorm(i) = norm(Us(:,i));
end
Performance indices initialization

\[ J_c = []; \quad J_d = []; \quad J_l = []; \quad J_s = []; \]

The used performance index is a sum over the entire simulation horizon of the state and input norms weighted as of Lyapunov decay condition.

\[
\begin{align*}
\text{for } i=1:T_{\text{sim}} \\
J_c(i) &= \sqrt{X_c(:,i)\cdot Q_x \cdot X_c(:,i) + U_c(:,i)\cdot Q_u \cdot U_c(:,i)}; \\
J_d(i) &= \sqrt{X_d(:,i)\cdot Q_x \cdot X_d(:,i) + U_d(:,i)\cdot Q_u \cdot U_d(:,i)}; \\
J_l(i) &= \sqrt{X_l(:,i)\cdot Q_x \cdot X_l(:,i) + U_l(:,i)\cdot Q_u \cdot U_l(:,i)}; \\
J_s(i) &= \sqrt{X_s(:,i)\cdot Q_x \cdot X_s(:,i) + U_s(:,i)\cdot Q_u \cdot U_s(:,i)};
\end{align*}
\]

Store the current simulation results

\[
\begin{align*}
\text{Sim}.J_c(\text{iter}) &= \text{sum}(J_c); \\
\text{Sim}.J_d(\text{iter}) &= \text{sum}(J_d); \\
\text{Sim}.J_l(\text{iter}) &= \text{sum}(J_l); \\
\text{Sim}.J_s(\text{iter}) &= \text{sum}(J_s);
\end{align*}
\]

Global results over 50 simulations:

- Central. ideal performance: 2048.6692
- Decentr. ideal performance: 2251.9299
- Decentr. lossy performance: 2487.4872
- Decentr. stoch performance: 2349.5637
- Central. ideal performance AVG: 40.9734
- Decentr. ideal performance AVG: 45.0386
- Decentr. lossy performance AVG: 49.7497
- Decentr. stoch performance AVG: 46.9913
- Central. ideal performance STD: 3.5888e-14
- Decentr. ideal performance STD: 7.1776e-15
- Decentr. lossy performance STD: 1.5545
- Decentr. stoch performance STD: 1.4656

CPU time:

- centralized: 2.7659
- dec_ideal: 0.8478
- dec_lossy: 8.3288
- dec_stoch: 58.5784

Conclusions

As expected, the best performance is achieved by the centralized ideal controller, as it can use all the state measurements for all actuators at all time steps.
However, the decentralized ideal controller achieves a performance which is only 10% worse than the ideal centralized one, while using much less communications between sensors and actuators.

A robustly stabilizing controller has been found also for the case where some of the links are subject to possible packet dropouts. In this case constraints are fulfilled at every time step, regardless of the occurrence of the dropouts in the network.

Exploiting a stochastic model of the packet dropouts, a stochastic controller has been computed which obtains an improvement on the performance index of around 6% with respect to the robust control scheme.

### A.3 Decentralized MPC (dlincon)

The class Decentralized LINear CONstrained (DLINCON) implements an extension of the class LINCON (Hybrid toolbox), for handling decentralized model predictive control (DMPC) of linear systems. Such extension is based on the approach described in Chapter 3. Moreover, *a-posteriori* the stability test is included as class method.

#### A.3.1 Class description

Starting with an user defined decentralization, dlincon automatically creates the corresponding set of decentralized lincon controllers of appropriate dimension. Class constructor is fed with plant model and controller type and with optional parameters such as the decentralization structure, QP cost function, model constraints, QP solver. Moreover each subcontroller can be customized as it is a lincon object.

Control action computation is implemented by method the Deval according to the selected mode:

- **global**: centralized controller;
- **Dglobal**: run all DMPCs and assemble the components to return the complete set of inputs;
- **i**: run the i-th controller and return its result;
Both regulator and tracking modes are supported. It is also possible to implement tracking as a regulator plus coordinates shift, as described in Chapter 3. However, this choice requires shift recalculation for any reference update. Time varying bounds are supported, thus they can be updated online.

A dlincon class internally stores decentralization matrices accordingly to Chapter 3. Such read-only properties are: $W$, selection matrix for states; $Z$, selection matrix for inputs; $G$, selection matrix for outputs; $APP$, selection matrix for applied inputs. Moreover, if the selected mode uses the coordinate shift the quantities $Zr$, state coordinate shift, and $Vr$, input coordinate shift, take a not null value. A cell array of ss objects is stored in the property $sub_sys$ with the subcontrollers models. Finally, a centralized controller is stored in $ccon$, while the cell array of lincon objects of sub controllers is readable as $dcon$.

A.3.2 Methods

The dlincon class constructor can be invoked using either the syntax: 
\[ dl = \text{dlincon} (\text{model}, \text{type}) \]
or the syntax:
\[ dl = \text{dlincon} (\text{model}, \text{type}, \text{decent}, \text{cost}, \text{interval}, \text{limits}, \text{qpsolver}, \ldots, \text{yzerocon}, \text{varbounds}, \text{ctrlTs}, \text{defaultMode}) \]

If the first mode is invoked no decentralization is used (i.e. only the global mode is supported for the method $\text{Deval}$).

Constructor parameters are:

1. model : centralized LTI model of the plant;
2. type : 'reg' for regulator, 'track' for tracking or 'sTrack' for regulator with coordinate shift;
3. decent : Decentralization structure, must be an array of structures containing fields:
   - x (indices of states of the subsystem)
   - y (indices of outputs of the subsystem)
   - u (indices of inputs of the subsystem)
A.3. Decentralized MPC (dlincon)

- applied (indices of inputs effectively applied, no overlap is allowed).
  each containing the array of indices to be included in the subsystem.

4. cost : structure with state and input matrices weights (see lincon for more details);

5. interval : structure with fields N and Nu, prediction and simulation horizons, respectively (see lincon for more details);

6. limits : structure with bounds on states and inputs (see lincon for more details);

7. qpsolver : QP solver (see lincon for details);

8. yzerocon : if 1 constraints are enforced also at the starting instant (see lincon for more details);

9. varbounds : if 1 additional states are added to the plant model to handle variant bounds specified by the user online.

10. ctrlTs : controller sample time (optional)

11. defaultMode : default mode when calling Deval.

The decentralization matrices are computed directly with the objects creation, along with the subsystems and relative controllers. In order to use the controller the method Deval is available and described in the following.

Deval computes the optimal control move basing on the mode, and state/output measurements and previous input, using the following syntax: `[u dl] = Deval(dl,xk,r,uold,qpsolver,verbose,mode)`. Input parameters are: dl, valid dlincon object; xk, r, hold, qpsolver and verbose, equivalent to correspondent of lincon (see lincon for details); mode, decentralization mode. Note that, because of nature of the approach, controller type 'sTrack' is not available in mode i, as the whole state and reference information are needed to compute reference shift.

Basing on the theory developed in Chapter 3 a sufficient only stability test of the closed-loop around the origin is available. The invocation uses the syn-
tax: \( \text{res} = \text{stability\_test}(\text{dl}, \text{range}, \text{cost}) \), where the input parameters are:

1. \( \text{dl} \): a valid dlincon object;

2. \( \text{range} \): the range in which compute the exploit controller (see \text{expcon} for details)

3. \( \text{cost} \): cost for the Lyapunov function

Additional methods are:

- \( \text{dl}=\text{setDefaultMode}(\text{dl}, \text{str}) \): Set the default mode to be used in the method Deval of this object;

- \( \text{res} = \text{decent\_verify}(\text{decent}, \text{sys}) \): Given a decentralization structure \( \text{decent} \) and a ss LTI model \( \text{sys} \), checks if the decentralization is consistent.

### A.3.3 Example

The following example aims at showing the usage of the class dlincon in a simple but complete control problem in which the plant has some a relevant degree of decoupling between its states.

The plant we are about to control is the LTI discrete-time system below, while \( \text{Nx} \), \( \text{Nu} \) and \( \text{Ny} \) are number of states, input and outputs, respectively

\[ Ts = \text{.1}; \]
\[
\text{sys} = \text{ss}([1.1 \ 0; \ 2.3 \ 0; \ 2.1.3], [1 \ 0; \ 0 \ 0.1; \ 0 \ 1], \text{eye}(3), \text{zeros}(3,2), Ts);
\]
\[
[\text{Nx}, \text{Nu}] = \text{size(sys.B)};
\]
\[
\text{Ny} = \text{size(sys.c,1)};
\]

Engineering insight suggests the following decentralization

```python
dec(1).u=[1 2];
dec(1).x=[1 2];
dec(1).y=[1 2];
dec(1).applied=1;
dec(2).u=[1 2];
dec(2).x=[2 3];
dec(2).y=[2 3];
dec(2).applied=2;
```
Below, 3 simulations are performed for each of two control problems, so as to allow comparison. The three approaches are regulation to the origin problem, direct reference tracking problem and reference tracking problem implemented as regulation of shifted model. This latter is referred as shifted-tracking. Problems 1 and 2 share plant dynamics but, problem 2 has more restrictive constraints.

Each simulation shows the behavior of the set of DMPC against the centralized controller that is used as reference for comparison. Both states and inputs are plotted.

```matlab
% Simulation parameters
x_c0=[-0.4286;0.2182;-0.3596];
u_c0=zeros(Nu);
Tsim=1;
for j=1:6
    % Setup controllers parameters
    switch mod(j,3)
    case 1
        type='reg';    % even values of j
    case 2
        type='track';
    case 0
        type='sTrack';
    end
    if ~strcmp(type, 'track')
        cost.Q=1e1*eye(Nx);
        cost.R=eye(Nu);
    else
        cost.S=1e1*eye(Nx);
        cost.T=eye(Nu);
    end
    interval.N=10;
    interval.Nu=5;
    var_bounds=(j>3);
    if var_bounds
        % Impose static constraint in case of variant bounds
        k=inf;
    else
        k=3;
    end
    limits.umin=-k*ones(Nu,1);
    limits.umax=k*ones(Nu,1);
    limits.dumin=-inf*ones(Nu,1);
    limits.dumax=inf*ones(Nu,1);
    limits.ymin=-k*ones(Ny,1);
    limits.ymax=k*ones(Ny,1);
```

cost.rho=inf;  
model=sys;  
if strcmp(type,'sTrack')

The method is unable to cope with open loop unstable plant, thus we stabilize it linearly

\[ K = -dlqr(sys.A,sys.B,cost.Q,cost.R); \]
\[ model.A = sys.A + sys.B*K; \]
end

Invocation of the dlincon constructor

\[ Dcon = dlincon(model,type,dec,cost,interval,limits,[],0,var_bounds); \]

range.xmin=-xx*ones(Nx,1);  
range.xmax=-range.xmin;  
if ~Dcon.var_bounds & ~strcmp(type,'track')

Test stability

if Dcon.stability_test(range,cost)  
disp('Stability test succeeded');  
else  
disp('Stability test failed');  
end
end

x_c=x_c0;  x_d=x_c;  u_c=u_c0;  u_d=u_c;  
disp('Start simulation')

Hybrid Toolbox ...  
Hybrid Toolbox ...  
Hybrid Toolbox ...
Open neighbors / regions found:
Analyzing region size ...
-->>Number of regions in the control law: 1
Open neighbors / regions found:
Analyzing region size ...
-->>Number of regions in the control law: 1
Closed Loop is stable with this decentralization
at least around the origin.(feedback method)
The closed loop with this decentralization is stable at
least around the origin even with a maximum of 0 losses (stability test)!!
Stability test success
Start simulation
Hybrid Toolbox ...
A.3. Decentralized MPC (dlinccon)

Hybrid Toolbox ...
Hybrid Toolbox ...
Start simulation
Hybrid Toolbox ...
Hybrid Toolbox ...
Hybrid Toolbox ...
Open neighbors / regions found:
Analyzing region size ...
-->Number of regions in the control law: 1
Open neighbors / regions found:
Analyzing region size ...
-->Number of regions in the control law: 1
Closed Loop is stable with this decentralization
at least around the origin. (feedback method)
The closed loop with this decentralization is stable at
least around the origin even with a maximum of 0 losses (stability test)!!
Stability test success
Start simulation
Hybrid Toolbox ...
Hybrid Toolbox ...
Hybrid Toolbox ...
Start simulation
Hybrid Toolbox ...
Hybrid Toolbox ...
Hybrid Toolbox ...
Start simulation

Setup variant bounds

    bounds=[-.2;.2];
    for i=2:Dcon.M
        bounds=[bounds; [-.2;.2]];
    end
    [A,B,C,D]=ssdata(sys);
    uoldc=u_c0(:,1);
    uoldd=uoldc(:,1);
    for i=1:Tsim/Ts
        x_c(:,i+1) = A*x_c(:,i) + B*u_c(:,i);
        x_d(:,i+1) = A*x_d(:,i) + B*u_d(:,i);
        if var_bounds
            if i*Ts>.2
                bounds(1:2:end)=-.3;
                bounds(2:2:end)=.3;
            end
            if mod(j,3)==1
                r=zeros(Nx+2*Nu,1);
            else
                r=zeros(Nx+2*Nu,1);
            end
        end
    end

There are only two inputs, hence only 1st and 3rd state components can
achieve zero-error convergence. $2Nu$ zeros are added because of the variant bound that are mapped as fake outputs.

```matlab
r=[-.1;0;0.05;zeros(2*Nu,1)];
end
if strcmp(type,'reg')
    u_c(:,i+1)=Dcon.Deval([x_c(:,i+1) ; bounds] , r, [], [], [], 'global');
    u_d(:,i+1)=Dcon.Deval([x_d(:,i+1) ; bounds] , r, [], [], [], 'Dglobal');
else if strcmp(type,'track')
    u_c(:,i+1)=uoldc+Dcon.Deval([x_c(:,i+1) ; bounds] , r,uoldc, [], [], 'global');
    u_d(:,i+1)=uoldd+Dcon.Deval([x_d(:,i+1) ; bounds] , r,uoldd, [], [], 'Dglobal');
    uoldc=u_c(:,i+1);
    uoldd=u_d(:,i+1);
else
    bb=bounds;
    bb(1:2:end)=bb(1:2:end)-K*x_c(:,i+1) ;
    bb(2:2:end)=bb(2:2:end)-K*x_c(:,i+1) ;
    u_c(:,i+1)=Dcon.Deval([x_c(:,i+1) ; bb] , r, [], [], [], 'global')... + K*x_c(:,i+1);
    u_d(:,i+1)=Dcon.Deval([x_d(:,i+1) ; bb] , r, [], [], [], 'Dglobal')... + K*x_d(:,i+1);
end
else
    if mod(j,3)==1
        r=zeros(Nx,1);
    else
        r=[-.1;0;0.05];
    end
    if strcmp(type,'reg')
        u_c(:,i+1)=Dcon.Deval(x_c(:,i+1), r, [], [], [], 'global');
        u_d(:,i+1)=Dcon.Deval(x_d(:,i+1), r, [], [], [], 'Dglobal');
    else if strcmp(type,'track')
        u_c(:,i+1)=uoldc+Dcon.Deval(x_c(:,i+1), r,uoldc, [], [], 'global');
        u_d(:,i+1)=uoldd+Dcon.Deval(x_d(:,i+1), r,uoldd, [], [], 'Dglobal');
        uoldc=u_c(:,i+1);
        uoldd=u_d(:,i+1);
    else
        u_c(:,i+1)=Dcon.Deval(x_c(:,i+1), r, [], [], [], 'global') + K*x_c(:,i+1);
        u_d(:,i+1)=Dcon.Deval(x_d(:,i+1), r, [], [], [], 'Dglobal') + K*x_d(:,i+1);
    end
end
end
figure;
t=0:Ts:Tsim;
subplot(2,1,1)
Figure A.6: State (upper plot) and input (lower plot) of closed-loop trajectories, where 1\textsuperscript{st}, 2\textsuperscript{nd} and 3\textsuperscript{rd} components are denoted with blue, green and red color, respectively. Centralized MPC trajectories are depicted in continuous line, decentralized MPC in dashed and references in starred-continuous. Regulation problem 1 with fixed bounds.

```matlab
plot(t,x_c);
hold on
plot(t,x_d,'--');
plot(t,r(1:Nx)*ones(1,length(t)),'*--');
hold off
str='States: cent(-), dec(--)';
str=[type ' ' str];
if var_bounds
str=[str ' variant bounds'];
end
title(str);
subplot(2,1,2)
plot(t,u_c);
hold on
plot(t,u_d,'--');
hold off
title('Inputs: cent(-), dec(--)');
```
Figure A.7: State (upper plot) and input (lower plot) of closed-loop trajectories, where 1\textsuperscript{st}, 2\textsuperscript{nd} and 3\textsuperscript{rd} components are denoted with blue, green and red color, respectively. Centralized MPC trajectories are depicted in continuous line, decentralized MPC in dashed and references in starred-continuous. Tracking problem 1 with fixed bounds.

\begin{verbatim}
if j<4
    disp('Press any key to test next configuration');
    pause(1);
end
end
\end{verbatim}

\textbf{Comment to the results}

Figures A.6 and A.9 depict the regulator configuration with fixed and time varying bounds, respectively. Convergence in the second case is slower due to the more restrictive bound imposed. However in both cases convergence is achieved by both the centralized and set of decentralized controllers.

Figure A.7 and A.10 depict the centralized and decentralized reference
tracking, respectively. This tracking is realized using standard lincon tracking. Figure A.10 exhibits more restrictive bounds and thus a slower convergence to the reference.

Figure A.8 and A.11 depict the reference tracking configuration with fixed and time varying bounds, respectively. Note that in figure A.8 the input bounds are not enforced. That is because the actual input is not the MPC but the sum of MPC and stabilizing gain. That changes at every time instant thus cannot be handled using fixed bounds. The more restrictive variant bounds make the latter case to need more time to achieve convergence with respect to the fixed bounds case. In particular Figure A.8 shows the main difference between the centralized and decentralized controllers, that is the longer transient in the decentralized case. However, Figure A.11 shows that
in the centralized case the bounds on input are enforced.

**A.4 Hierarchical MPC (HiMPC)**

The class HiMPC, Hierarchical MPC, implements the hierarchical and decentralized hierarchal MPC control techniques described in Chapter 4.

**A.4.1 Class description**

The HiMPC class has been developed to automatically generate the multi-layer control architectures proposed in Chapter 4. Both centralized and decentralized hierarchical schemes can be supported, depending on the decentral-
A.4. Hierarchical MPC (HiMPC)

The HiMPC object aims at computing the maximum element-wise reference variation $\Delta r$ that the upper layer(s) may give to the lower layer controller while ensuring global constraint satisfaction. Such value, which depends on the sample time ratio among the two control layer is computed for all integer ratios, until unfeasibility is reach, i.e. no variation of the global constraints can be determined within the admissible references set, such that instability may occur.

The approach is sensible to the tuning knob parameter $\Delta K$. The purpose of this Section is to deepen the study of the influence of such feature.

Figure A.10: State (upper plot) and input (lower plot) of closed-loop trajectories, where 1$^{st}$, 2$^{nd}$ and 3$^{rd}$ components are denoted with blue, green and red color, respectively. Centralized MPC trajectories are depicted in continuous line, decentralized MPC in dashed and references in starred-continuous. Tracking problem 2 with fixed bounds.
Figure A.11: State (upper plot) and input (lower plot) of closed-loop trajectories, where 1st, 2nd and 3rd components are denoted with blue, green and red color, respectively. Centralized MPC trajectories are depicted in continuous line, decentralized MPC in dashed and references in starred-continuous. Shifted-tracking problem 2 with fixed bounds.

A.4.2 Methods

The HIMPC class constructor has the syntax: \( \text{obj} = \text{HiMPC}(\text{model}, \text{dec}, \text{Xcon}, \Delta \text{K}, \text{coupledCons}) \) where the input parameters are:

1. \text{model} : \text{ss} object with the plant LTI model;
2. \text{Xcon} : state bounds structure with fields \text{min} and \text{max};
3. \Delta \text{K} : Array of the same size of the constraints that determines their tightening;
4. \text{coupledCons} : Cell array of non element-wise state constraints with fields \text{H} and \text{K} to be appended to the constraint polytope of the corresponding subsystem;
5. **dec**: Decentralization structure as described in Section A.3;

Once the object is created, one is ready to perform computation of invariant set of the subsystem(s). Such task is performed using the class method `obj=computeMOARS(obj,ray)`, where `ray` is the ray of the unit ball used to avoid empty polytopes (default 1e-6). The necessity of `ray` follows from the inability of the Multi-parametric toolbox to handle not-full-dimensional polytopes.

After the MOARS computation took place a plot is available using `plotMOARS(obj)`. If the dimension exceeds 3 a projection is plotted.

Once the MOARS is computed the computation of the $\Delta r$ can take place by invoking `obj = computeDeltaR(obj)`. After, a plot is available using either `[h1 h2]=plotDeltaR(obj)` or `[h1 h2]=plotDeltaR(obj,h1,h2)`, where Matlab figure handles `h1` and `h2` are used to set the figure in which to draw; otherwise a new figure is used.

Finally, a method for computing the length of the longest segment inscribed in a polytope is available, with the syntax: 

`l = maximumLengthSegmentInPolytope(obj,P)`, where the polytope `P` is a MPT-toolbox polytope object.

### A.4.3 Example

This demo explores the tuning knob parameter $\Delta K$ of Chapter 4. We focus on centralized hierarchical implementation carrying out calculation with two different choices of $\Delta K$ in a simplified version of the problem described in the theory Chapter. In fact, a two mass system is considered, where the coupling constraint between the two restricts the first mass to not be below the second more than 0.3 meters. The singular mass constraints as well as the plant parameters are equivalent to those used in Chapter 4, thus each mass position is restricted within $[-0.3; 1]$ meters and each velocity in $[-10; 10]$ meters per second. The code is however capable of dealing with an arbitrary number of masses.

The first controller, which is referred as HiMPC1 uses a shrinking parameter of the admissible references of 0.2 and 2 for positions and velocities, respectively, while the coupling constraint shrinking is 0.08. The second con-
A. WIDE Toolbox

troller, namely HiMPC2, varies only the coupled constraint parameter, that is 0.04.

% Sample time and number os hanged masses
TL=.25;
N=2;

Create model of hanged masses with friction coupling (no DC coupling, so as to satisfy Chapter 4 assumptions)

[sys N]=buildModelSpring(TL,N);
[A B C D] = ssdata(sys);
[Nx Ny]=size(B);
csys=ss(A,B,C,D,TL);
cdcg=dcgain(csys);
csys.B=csys.B*inv(cdcg);
% Constraints definition
Xcon.min=[];
for i=1:N,Xcon.min=[Xcon.min;-.3;-10];end
Xcon.max=[];
for i=1:N,Xcon.max=[Xcon.max;1;10];end

Define a decentralization with a single subsystem identical to the whole plant for generate a centralized controller

cdec(1).x=1:N*2;
cdec(1).y=1:N;
cdec(1).u=1:N;
cdec(1).applied=1:N;

Define the coupling constraint involving first and second mass and the shrinking parameters, i.e. \( \Delta K \)

ccoupledCons(1).H=[-1 0 1 0 zeros(1,N*2-4)];
ccoupledCons(1).K=[.3];
cDeltaK{1}=[];
for i=1:N*2
    cDeltaK{1}=[cDeltaK{1};0.2;2];
end

% Define the two controller shrinking parameters and create the HiMPC objects.
for jj=1:2
    if jj==1
        cDeltaK{1}=[cDeltaK{1};0.08];
    else
        cDeltaK{1}=[cDeltaK{1}(1:end-1);0.04];
    end

Compute the Maximum Output Admissible Robust Sets and the maximum element-wise reference variations such that at the next execution the supervisor the controller state is within the new reference invariant set, for each sample time ratio N and each controller
A.4. Hierarchical MPC (HiMPC)

cobj(jj) = HiMPC(csys, cdec, Xcon, cDeltaK, ccoupledCons);
cobj(jj) = cobj(jj).computeMOARS();
cobj(jj) = cobj(jj).computeDeltaR();

Determine the 'best' tradeoff, fastest velocity slope, by maximizing the allowed reference variation over the corresponding sample time

$$[\text{sup ind}] = \max(cobj(jj).DrN{1})$$
cDrOpt = cobj(jj).Dr{1}(ind);

$$\text{cTH(jj)} = \text{ind} \times \text{TL};$$
ncu = length(cobj(jj).Kr{1});
cc = [csys.c; zeros(ncu, N*2)];
dd = [csys.d; cobj(jj).Hr{1} * cdcg];
cmodel = ss(A, B, cc, dd, TL);

% Define the MPC controller using lincon class of the Hybrid Toolbox

% clear limits cost interval
% cmodel = d2d(cmodel, cTH(jj));

type = 'track';
cost.S = blkdiag(1e1*eye(N), zeros(ncu));
cost.T = 1e0*eye(N);
cost.rho = inf;
interval.N = 10;
interval.Nu = 5;
limits.ymin = [csys.c*Xcon.min; -inf*ones(ncu, 1)];
limits.ymax = [csys.c*Xcon.max; cobj(jj).Kr{1}];
limits.dumin = -cdrOpt * ones(N, 1);
limits.dumax = cdrOpt * ones(N, 1);
if jj == 1
    cL1 = lincon(cmodel, type, cost, interval, limits, [], 1);
else
    cL2 = lincon(cmodel, type, cost, interval, limits, [], 1);
end

looking for available solvers...

MPT toolbox 2.6.3 initialized...

Copyright (C) 2003-2006 by M. Kvasnica, P. Grieder and M. Baotic
Send bug reports, questions or comments to mpt@control.ee.ethz.ch
For news, visit the MPT web page at http://control.ee.ethz.ch/~mpt/

LP solver: CDD Criss-Cross
QP solver: SeDuMi
MILP solver: YALMIP
MIQP solver: YALMIP

Vertex enumeration: CDD

Run 'mpt_studio' to start the GUI. Run 'mpt_setup' to set global parameters.

Computing the MOARS for subsystem 1

Oinf=
Normalized, minimal representation polytope in R^4

H: [9x4 double]
K: [9x1 double]
normal: 1
mixrep: 1
xCheb: [4x1 double]
NCheb: 0.0992
Done

1-th problem of the 1 subsystem \( \Delta R = 7.0711 \times 10^{-5} \)
2-th problem of the 1 subsystem \( \Delta R = 7.1471 \times 10^{-5} \)
3-th problem of the 1 subsystem \( \Delta R = 7.3158 \times 10^{-5} \)
4-th problem of the 1 subsystem \( \Delta R = 7.6228 \times 10^{-5} \)
5-th problem of the 1 subsystem \( \Delta R = 8.0832 \times 10^{-5} \)
6-th problem of the 1 subsystem \( \Delta R = 8.6042 \times 10^{-5} \)
7-th problem of the 1 subsystem \( \Delta R = 9.2885 \times 10^{-5} \)
8-th problem of the 1 subsystem \( \Delta R = 0.00010219 \)
9-th problem of the 1 subsystem \( \Delta R = 0.00011469 \)
10-th problem of the 1 subsystem \( \Delta R = 0.00012882 \)
11-th problem of the 1 subsystem \( \Delta R = 0.00014819 \)
12-th problem of the 1 subsystem \( \Delta R = 0.00017167 \)
13-th problem of the 1 subsystem \( \Delta R = 0.00020089 \)
14-th problem of the 1 subsystem \( \Delta R = 0.00023417 \)
15-th problem of the 1 subsystem \( \Delta R = 0.00026641 \)
16-th problem of the 1 subsystem \( \Delta R = 0.00028952 \)
17-th problem of the 1 subsystem \( \Delta R = 0.00030659 \)
18-th problem of the 1 subsystem \( \Delta R = 0.00028931 \)
19-th problem of the 1 subsystem \( \Delta R = 0.00027438 \)
20-th problem of the 1 subsystem \( \Delta R = 0.00026849 \)
21-th problem of the 1 subsystem \( \Delta R = 0.00024472 \)
22-th problem of the 1 subsystem \( \Delta R = 0.0002353 \)
23-th problem of the 1 subsystem \( \Delta R = 0.00023028 \)
24-th problem of the 1 subsystem \( \Delta R = 0.00022468 \)
25-th problem of the 1 subsystem \( \Delta R = 0.00022571 \)
26-th problem of the 1 subsystem \( \Delta R = 0.00023104 \)
27-th problem of the 1 subsystem \( \Delta R = 0.00024074 \)
28-th problem of the 1 subsystem \( \Delta R = 0.00025528 \)
29-th problem of the 1 subsystem \( \Delta R = 0.00027558 \)
30-th problem of the 1 subsystem \( \Delta R = 0.00030317 \)
31-th problem of the 1 subsystem \( \Delta R = 0.00034053 \)
32-th problem of the 1 subsystem \( \Delta R = 0.00039172 \)
33-th problem of the 1 subsystem \( \Delta R = 0.00046371 \)
34-th problem of the 1 subsystem \( \Delta R = 0.00056938 \)
35-th problem of the 1 subsystem \( \Delta R = 0.0012181 \)
36-th problem of the 1 subsystem \( \Delta R = 0.0016815 \)
37-th problem of the 1 subsystem \( \Delta R = 0.0061118 \)
38-th problem of the 1 subsystem \( \Delta R = 0.00348 \)
39-th problem of the 1 subsystem \( \Delta R = 0.003976 \)
40-th problem of the 1 subsystem \( \Delta R = 0.002897 \)
41-th problem of the 1 subsystem \( \Delta R = 0.002863 \)
42-th problem of the 1 subsystem \( \Delta R = 0.0028957 \)
43-th problem of the 1 subsystem \( \Delta R = 0.0028954 \)
44-th problem of the 1 subsystem \( \Delta R = 0.0029663 \)
45-th problem of the 1 subsystem \( \Delta R = 0.0031022 \)
46-th problem of the 1 subsystem \( \Delta R = 0.0032879 \)
47-th problem of the 1 subsystem \( \Delta R = 0.0035185 \)
48-th problem of the 1 subsystem \( \Delta R = 0.0036501 \)
49-th problem of the 1 subsystem \( \Delta R = 0.0036834 \)
50-th problem of the 1 subsystem \( \Delta R = 0.0037662 \)
51-th problem of the 1 subsystem \( \Delta R = 0.0039613 \)
52-th problem of the 1 subsystem \( \Delta R = 0.0041779 \)
53-th problem of the 1 subsystem \( \Delta R = 0.004279 \)
54-th problem of the 1 subsystem \( \Delta R = 0.0050419 \)
55-th problem of the 1 subsystem \( \Delta R = 0.005163 \)

Hybrid Toolbox ...
Computing the MGAB for subsystem 1
Omx=
Normalized, minimal representation polytope in R^4
K: [9x4 double]
K: [9x1 double]
normal: 1
minrep: 1
xCheb: [4x1 double]
RCheb: 0.0446
Since MOARS of both controllers are in $\mathbb{R}^4$, rather than plotting a projec-
tion, we report the maximal volume inner ellipsoid\(^8\) and the minimal volume bounding box\(^9\) so as to give an idea of the polytope volumes.

\[
\text{moars1} = \text{cobj(1).Oinf{1}}; \\
\text{moars2} = \text{cobj(2).Oinf{1}}; \\
[B1 \ d1] = \text{maximumVolumeInscribedEllipsoid(moars1)};
\]

Calling sedumi: 616 variables, 123 equality constraints

For improved efficiency, sedumi is solving the dual problem.


\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
\text{it} & \text{b*y} & \text{gap} & \text{delta rate} & t/IP* & t/D* & \text{feas cg cg prec} \\
0 & 5.63E+00 & 0.000 \\
1 & 4.22E-01 & 4.63E+00 & 1.2921 & 0.9000 & 0.9000 & 2.98E+00 \\
2 & 1.66E-01 & 2.57E+00 & 4.5000 & 1.1E+00 \\
3 & 6.26E-02 & 7.69E-01 & 1.8800 & 2.92E-01 \\
4 & 9.24E-02 & 3.18E-01 & 1.1600 & 1.42E-01 \\
5 & 1.20E-01 & 5.20E-01 & 0.9400 & 7.22E-02 \\
6 & 1.20E-01 & 7.48E-02 & 0.9800 & 3.72E-02 \\
7 & 1.33E-01 & 1.07E-02 & 0.9900 & 9.82E-03 \\
8 & 1.33E-01 & 4.63E-03 & 1.0000 & 2.92E-03 \\
9 & 1.33E-01 & 1.48E-03 & 1.0000 & 1.1E-03 \\
10 & 1.33E-01 & 1.51E-04 & 1.0000 & 1.62E-04 \\
11 & 1.33E-01 & 3.40E-05 & 1.0000 & 4.32E-05 \\
12 & 1.33E-01 & 6.85E-06 & 1.0000 & 1.26E-05 \\
13 & 1.33E-01 & 9.52E-07 & 1.0000 & 2.11E-06 \\
14 & 1.33E-01 & 2.07E-07 & 1.0000 & 5.86E-07 \\
15 & 1.33E-01 & 6.35E-08 & 1.0000 & 1.90E-07 \\
16 & 1.33E-01 & 1.96E-08 & 1.0000 & 5.62E-08 \\
17 & 1.33E-01 & 5.25E-09 & 1.0000 & 2.15E-08 \\
18 & 1.33E-01 & 1.66E-09 & 1.0000 & 4.84E-09 \\
\end{array}
\]

iter    seconds    digits    c*x    b*y
18     0.4       Inf     1.3310037955e-01 1.3310038003e-01

|Ax-b| = 1.2e-09, [Ay-c]_+ = 1.7e-09, |x| = 2.0e+00, |y| = 5.2e-01

Detailed timing (sec)

Pre IPM Post

1.000E-01 4.100E-01 1.000E-02

Max-norms: ||b||=1, ||c|| = 2.000000e-01, Cholesky |add|=0, |skip| = 0, ||L.L|| = 98.0224.

Status: Solved
Optimal value (cvx_optval): +0.1331

\[
\text{bb1} = \text{bounding_box(moars1)}; \\
[B2 \ d2] = \text{maximumVolumeInscribedEllipsoid(moars2)};
\]

Calling sedumi: 616 variables, 123 equality constraints

For improved efficiency, sedumi is solving the dual problem.

\[^8\text{Based on Section 8.4.1 of [20]}\]

\[^9\text{Make use of MPT function bounding_box}\]
A.4. Hierarchical MPC (HiMPC)

Alg = 2: xz-corrector, Adaptive Step-Differentiation, theta = 0.250, beta = 0.500
m = 123, order n = 301, dim = 648, blocks = 100

nnz(A) = 1995 + 0, nnz(ADA) = 3137, nnz(L) = 1632

<table>
<thead>
<tr>
<th>iter</th>
<th>b*ygap</th>
<th>delta</th>
<th>rate</th>
<th>t/tP*</th>
<th>t/tD*</th>
<th>feas</th>
<th>cg</th>
<th>prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.26E-01</td>
<td>0.000</td>
<td>12.08</td>
<td>1</td>
<td>1</td>
<td>4.8E+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4.26E-01</td>
<td>3.96E+00</td>
<td>0.000</td>
<td>0.7936</td>
<td>0.9000</td>
<td>1.36E-01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.72E-01</td>
<td>2.29E+00</td>
<td>0.000</td>
<td>0.5780</td>
<td>0.9000</td>
<td>4.59E+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6.66E-02</td>
<td>8.41E-01</td>
<td>0.000</td>
<td>0.3671</td>
<td>0.9000</td>
<td>4.3E+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6.76E-02</td>
<td>3.47E-01</td>
<td>0.000</td>
<td>0.1428</td>
<td>0.9000</td>
<td>1.36E-01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9.2E-02</td>
<td>1.35E-01</td>
<td>0.000</td>
<td>0.2392</td>
<td>0.9000</td>
<td>1.36E-01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.01E-01</td>
<td>2.26E-02</td>
<td>0.000</td>
<td>0.6435</td>
<td>0.9033</td>
<td>1.36E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.06E-01</td>
<td>1.88E-02</td>
<td>0.000</td>
<td>0.2604</td>
<td>0.9010</td>
<td>1.36E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1.06E-01</td>
<td>9.4E-03</td>
<td>0.000</td>
<td>0.2626</td>
<td>0.9072</td>
<td>1.36E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1.06E-01</td>
<td>3.5E-02</td>
<td>0.000</td>
<td>0.3196</td>
<td>0.9000</td>
<td>1.36E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.06E-01</td>
<td>1.48E-04</td>
<td>0.000</td>
<td>0.1331</td>
<td>0.9156</td>
<td>1.36E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1.05E-01</td>
<td>1.25E-05</td>
<td>0.000</td>
<td>0.0594</td>
<td>0.9901</td>
<td>1.36E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1.05E-01</td>
<td>9.16E-07</td>
<td>0.000</td>
<td>0.0734</td>
<td>0.9086</td>
<td>1.36E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1.05E-01</td>
<td>1.63E-07</td>
<td>0.000</td>
<td>0.1780</td>
<td>0.9105</td>
<td>1.36E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1.05E-01</td>
<td>9.5E-08</td>
<td>0.000</td>
<td>0.3038</td>
<td>0.7304</td>
<td>1.36E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1.05E-01</td>
<td>1.44E-08</td>
<td>0.000</td>
<td>0.2911</td>
<td>0.6399</td>
<td>1.36E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1.05E-01</td>
<td>3.46E-09</td>
<td>0.000</td>
<td>0.2400</td>
<td>0.8234</td>
<td>1.36E-02</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

iter seconds digits c*x b*y
16 0.4 Inf 1.0538955245e-01 1.0538955323e-01

|Ax-b| = 4.8e-09, [Ay-c]_+ = 2.4E-09, |x|= 2.3e+00, |y|= 4.7e-01

Detailed timing (sec)
Pre IPM Post
2.000E-02 3.800E-01 1.000E-02
Max-norms: ||b||=1, ||c|| = 2.000000e-01,
Cholesky [add]=0, [skip] = 0, ||L.L|| = 43.0684.

Status: Solved
Optimal value (cvx_optval): +0.10539

bb2=bounding_box(moars2);
disp(MDARS1 is a polytope included between the inner ellipsoid')
disp('E = \{ Bu + d | || u || <= 1 \} (ellipsoid)')
disp('with B:') disp(B1) disp('d:') disp(d1)
disp('and the bounding box') display(bb1);
disp(MDARS2 is a polytope included between the inner ellipsoid')
disp(E = \{ Bu + d | || u || <= 1 \} (ellipsoid)')
disp('with B:') disp(B2) disp('u:') disp(d2)
disp('and the bounding box')
display(bb2);

MDARS1 is a polytope included between the inner ellipsoid
E = \{ Bu + d | || u || <= 1 \} (ellipsoid)
with B:

0.1537  -0.0367  0.0532  -0.0234
-0.0367  0.1373  -0.0234  0.0425
0.0532  -0.0234  0.1537  -0.0367
-0.0234  0.0425  -0.0367  0.1373
d:

0.0316
0.0227
-0.0316
and the bounding box

\text{bb1}=
Normalized, minimal representation polytope in $\mathbb{R}^4$

\text{RCheb: 0.2000}

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
-1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 \\
\end{bmatrix}
\begin{bmatrix}
0.2 \\
0.22888 \\
0.22325 \\
0.22325 \\
0.2 \\
0.22481 \\
0.2 \\
0.22481 \\
\end{bmatrix}
\]

MOARS2 is a polytope included between the inner ellipsoid

$E = \{ Bu + d \mid \|u\| \leq 1 \}$ (ellipsoid)

with $B$:

\[
\begin{bmatrix}
0.1483 & -0.0351 & 0.0932 & -0.0268 \\
-0.0351 & 0.1344 & -0.0268 & 0.0882 \\
0.0932 & -0.0268 & 0.1483 & -0.0351 \\
-0.0268 & 0.0882 & -0.0351 & 0.1344 \\
\end{bmatrix}
\]

and $u$:

\[
\begin{bmatrix}
0.0194 \\
0.0102 \\
-0.0194 \\
-0.0102 \\
\end{bmatrix}
\]

and the bounding box

\text{bb2}=
Normalized, minimal representation polytope in $\mathbb{R}^4$

\text{RCheb: 0.2000}

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
-1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 \\
\end{bmatrix}
\begin{bmatrix}
0.2 \\
0.22481 \\
0.2 \\
0.22481 \\
0.2 \\
0.22481 \\
0.2 \\
0.22481 \\
\end{bmatrix}
\]

The size of MOARS1 and MOARS2 is estimable by geometric considerations. Comparing diagonal entries of matrix $B$ of the two cases, which are proportionally related to the ellipsoids diagonals, is possible to see that the inner ellipsoid of HiMPC1 is bigger in volume than the one of HiMPC2. The same consideration can be done considering the bounding boxes and in particular the entries on the right-hand-side of the polytope inequality, for which the same property holds.
A.4. Hierarchical MPC (HiMPC)

We can therefore state that MOARS1 is “bigger in volume” than MOARS2, even if we have not given a formal proof. Recalling the definition of $\Delta K$, HiMPC1 applies a greater shrinking than HiMPC2, thus it restrict the set of admissible references and this results in an enlarged MOARS. However, the set of admissible error, \( i.e. \mathcal{E} \), is enlarged as well, thus the consideration is not general and should be studied case dependently.

Figure A.12 shows that HiMPC1 (blue line) grows more rapidly than HiMPC2 (black line), even if they both reach a maximum variation of 0.9 meters. That is because the coupling constraint is active only in the transient, as expected from theoretical assumptions. However, Figure A.13 shows that the maximum speed of variation differ from HiMPC1 and HiMPC2, as
the latter allows the full reference variation with a sample time ratio of 58, while the first at 54. This makes HiMPC1 more responsive, which is coherent with aforementioned consideration about invariant set hyper-volume.

\[
T_{\text{sim}} = 3 \times 60; \quad t = 0:TL:T_{\text{sim}}-TL; \quad TT = T_{\text{sim}}/TL;
\]

References for 2 states and initial condition

\[
r = [.65 \times \text{ones}(1,TT/2-15) \quad .2 \times \text{ones}(1,TT/2+15); \quad .8 \times \text{ones}(1,TT/4+10) \quad .7 \times \text{ones}(1,TT+2/4+10) \quad .4 \times \text{ones}(1,TT*1/4-20)];
\]

\[
x0 = \text{zeros}(N*2); \quad u0 = \text{zeros}(N,1);
\]

Simulation of plant closed-loop evolution

for \( k=2:TT-1 \)
  if mod(k,cTH(1)/TL)==2
    \( u(:,k)=u(:,k-1)+\text{eval}(cL1,x(:,k),[r(:,k);zeros(ncu,1)],u(:,k-1)); \)
  end
  if mod(k,cTH(2)/TL)==2
    \( cu(:,k)=cu(:,k-1)+\text{eval}(cL2,cx(:,k),[r(:,k);zeros(ncu,1)],cu(:,k-1)); \)
  end
A.4. Hierarchical MPC (HiMPC)

\[
x(:,k+1)=Ax(:,k)+Bu(:,k);
u(:,k+1)=u(:,k);
cx(:,k+1)=A*cx(:,k)+B*cu(:,k);
cu(:,k+1)=cu(:,k);
\]

end

figure(4);
plot(t,x(1,:)','b',t,x(3,:)','--b',t,r(1,:)','r',t,r(2,:)','--r',t,cx(1,:)','k',t,cx(3,:)','--k');
title('First and second mass positions (blue=1st ; black=2nd)')
figure(5);
plot(t,u(1,:)','b',t,u(2,:)','--b',t,cu(1,:)','k',t,cu(2,:)','--k')
title('First and second inputs (blue=1st ; black=2nd)')
figure(6);

Computation of performance indices

index1=0;
index2=0; for i=2:TT
  index2=index2+(r(:,i)-C*cx(:,i))'*cost.S(1:N,1:N)*(r(:,i)-C*cx(:,i))...+(cu(:,i)-cu(:,i-1))'*cost.T*(cu(:,i)-cu(:,i-1));
  index1=index1+(r(:,i)-C*x(:,i))'*cost.S(1:N,1:N)*(r(:,i)-C*x(:,i))...+(u(:,i)-u(:,i-1))'*cost.T*(u(:,i)-u(:,i-1));
end
disp(['Cumulated cost with centralized feedback HiMPC1 is ' num2str(index1)]);
disp(['Cumulated cost with centralized feedback HiMPC2 is ' num2str(index2)]);

Cumulated cost with centralized feedback HiMPC1 is 294.9379
Cumulated cost with centralized feedback HiMPC2 is 220.5983

Cumulated cost, i.e. MPC objective function computed for the whole simulation horizon, shows that HiMPC2 outperform HiMPC1. This is in contrast with considerations made about the reference speed of change and the MOAS size in volume. We stress that the used performance index is strongly dependent on the simulation as the two controller exhibit better performance in difference user defined reference situations. In fact, Figure A.14 shows that the blu line, namely HiMPC1, is in general more reactive as it can track the reference variation more quickly than HiMPC2. However, the bigger value of $\Delta K$ implies a greater shrinking of the admissible references, thus when an infeasible state is requested by the user (time instants from 86.25 to 140 seconds) the black line can get closer to the reference than the blue one.

Moreover, the user defined reference influences controllers comparison also from another viewpoint. The time instant at which the user defined reference
update occurs is independent on the two controllers sample time ratio. Therefore, even if HiMPC1 is usually more reactive than HiMPC2, there can be situations like at instant 86.25 seconds, where HiMPC1 updated his output just before the user reference, while HiMPC2 will do so immediately after. Therefore, HiMPC2 update can take into account the new reference value, while HiMPC1 needs to wait until its next execution. That explains the better performance of HiMPC2 with respect to HiMPC1 in this simulation.

Figure A.15 shows that the output of the HiMPCs, i.e. the lower layer controllers input, is remarkably different than the user defined reference (depicted in red in Figure A.14) even in steady state, namely when the transient of previous reference update is expired. That is motivated by the DC gain of the plant that is not unitary.
A.5 NCS simulation with TrueTime (ACG)

The class Automatic Code Generation (ACG) creates a ready-to-use setup networked control system simulation by generating a Simulink model based on TrueTime blocks and corresponding configuration m-files. The Network model is specified by the number of actuators and sensors, while controller and plant should be customized by the user fulfilling the respective subsystem.

A.5.1 Class description

TrueTime is a very powerful tool which can handle simulation of a variety of tasks, with a particular focus on real-time implementation of controller and sensors/actuators codes. As a result of this plenty of possibilities offered, the user needs to set-up a lot of features, some of which might be out of scope in some contexts.

The class ACG (Automatic Code Generator) allows to generate automati-
cally all the configuration m-files needed to run a standard networked-control simulation. The user is asked to simply set the number of sensors and actuators that are desired to be in the network and a name for the Simulink model to be generated.

Then the code generation creates a Simulink model that contain two sub-models, Plant and Controller, that can be customized, saving to the designed the wireless network setup.

A.5.2 Methods

The ACG class objects are created using the class constructor with following syntax: \texttt{acg\_obj=ACG(Ns,Na,name,period)}, where input parameters are:

1. \texttt{Ns}: number of sensors in the network (Optional: default value=1);
2. \texttt{Na}: number of actuators in the network (Optional: default value=1);
3. \texttt{name}: name of the Simulink model (Optional: default value=’acg\_default’);
4. \texttt{period}: period of sample time.

The class dispose of a method to generate the code basing on the objected properties, \textit{i.e.} \texttt{GenerateCode(acg)}. Moreover, there exist the possibility of removing previously generated code, by invoking \texttt{RemoveOldCode(acg)}.

A.5.3 Example

This example shows how to use ACG class for simulating a networked control system. First the TrueTime code and relative Simulink model are generated, then a plant and a controller are defined to perform a simulation. Finally the TrueTime antennas transmit power is reduced so as to show the effect on the control system performance.

```matlab
% Define number and positions of sensors and actuators
Ns = 2;
Na = 2;
Xpos = rand((Ns+Na)*2,1);
Ypos = rand((Ns+Na)*2,1);
% Set the name to give to the generated .mld file
name=’Example’;
```
A.5. NCS simulation with TrueTime (ACG)

Create a new instance of ACG class

```matlab
acg_obj=ACG(Ns,Na,name,0.1);
acg_obj.RemoveOldCode;
acg_obj.GenerateCode;
```

The LTI discrete-time system we consider is defined by the following state space matrix:

\[
A = \text{diag}([0.1 \ 0.2]) + 0.01 \times \text{rand}(2);
B = \text{diag}([1 \ 0.3]);
C = \text{diag}([2 \ 4]);
D = \text{zeros}(2);
x_0 = [10 \ 20];
\]

Create and configure the Plant and Controller submodels of the generated Simulink model. Here the generation is performed in a script fashion for automatization proposes, however the user may use the graphical interface.

```matlab
add_block('simulink/Signal Routing/Mux',[name '/Plant/Mux']);
set_param([name '/Plant/Mux'],'inputs','2');
add_block('simulink/Signal Routing/Demux',[name '/Plant/Demux']);
set_param([name '/Plant/Demux'],'outputs','2');
add_block('simulink/Continuous/State-Space',[name '/Plant/State-Space']);
add_line([name '/Plant'],'Mux/1','State-Space/1');
add_line([name '/Plant'],'State-Space/1','Demux/1');
for i=1:Na
    add_line([name '/Plant'],[‘Actuator’ num2str(i*2+Ns+1) ’/1’],[’Mux’/’Demux’/’1’]);
    add_line([name '/Plant'],[‘Demux’/’State-Space’/’1’],[’Sensor’ num2str(2*i-1) ’/1’]);
end
set_param([name '/Plant/State-Space’],’A’,’A’,’A’,’B’,’B’,’C’,’C’,’D’,’y0’,’x0’);
% Configure the controller
add_block('simulink/Signal Routing/Demux',[name '/Controller/Demux']);
set_param([name '/Controller/Demux’],’outputs’,’2’);
for i=1:Ns
    ii=num2str(i);
    add_block('simulink/Math Operations/Sum',[name '/Controller/Sum’ ii]);
```

Figure A.16: Simulink model generated by ACG.
set_param([name '/Controller/Sum' ii],'ListOfSigns','+-|');
add_block('simulink/Math Operations/Gain',[name '/Controller/Gain' ii]);
add_line([name '/Controller'],[['Demux' ii],[['Sum' ii '/1']]);
add_line([name '/Controller'],[['Sensor' num2str(2*i-1) ,'/1'],['Sum' ii '/2']]);
add_line([name '/Controller'],[['Sum' ii '/1'],['Gain' ii '/1']]);
add_line([name '/Controller'],[['Gain' ii '/1'],['Actuator' num2str(2*i+Ns+1) '/1']]);
end
add_line([name '/Controller'],[['Reference/1'],['Demux/1']]);
set_param([name '/Controller/Gain1'],[['gain','30']]);
set_param([name '/Controller/Gain2'],[['gain','6']]);

Now we need to setup the parameters in order to perform the first simulation.

% Reference signal
ref = [8 -5]';
% Antennas transmit power
set_param([name '/Wireless Network/TrueTime Wireless Network' ],['TransPower','=-30']);
save_system(name); sim(name);
plot(simout.time,simout.signals.values,'b',simout.time,ref*ones(1,length(simout.time)),'r');
disp('Few packets have gone lost using the current transmission power.
Press any key for use a lower transmission power and observe the effects.

In order to demonstrate the effectiveness of TrueTime wireless network simulator, we decrease antenna transmit power, expecting a performance degradation in the simulation.

set_param([name '/Wireless Network/TrueTime Wireless Network' ],['TransPower','=-46']);
save_system(name); sim(name); hold on
plot(simout.time,simout.signals.values,'--b',simout.time, ref*ones(1,length(simout.time)),'r');
Figure A.17: Closed-loop trajectories with network antenna transmit power at $-30dB$.

Conclusions

Figures A.17 and A.18 demonstrate how the transmit power determine the performance of the control actions. In Figure A.17 the case with $-30dB$ transmit power shows that convergence is achieved, while for the lower transmit power case $-46dB$, depicted in Figure A.18, is possible to observe that one state diverges.
Figure A.18: Closed-loop trajectories with network antenna transmit power at $-46dB$. 
Appendix B

MOBYDIC Toolbox

B.1 Introduction

Despite the importance of embedded control in many applications, there are only few integrated methods to design and deploy embedded control systems in a systematic and highly efficient manner. There is an abundance of methods for the separate design of the control algorithms eventually embedded into a hardware platform, or to create a suitable hardware platform for the implementation of a given control algorithm. These methods typically follow a sequential design procedure as depicted in Fig. B.1. On the other hand, to the best of our knowledge, no tools are available for automated and integrated design flows from the mathematical model of the process to the synthesis of the electronic circuits. Ideally, such tools should incorporate both control specifications (stability, constraint satisfaction, performance) and circuit requirements (speed, size, power usage, cost), as depicted in Fig. B.1.

This Appendix is concerned with a software toolbox (called MOBY-DIC) implementing an integrated design flow that is based on explicit MPC [13]. The resulting control law is a piecewise-affine (PWA) function of the system state, which allows one to cope with constraints on input and state variables without the need to solve an optimization problem on-line, and is therefore very effective to control small-sized systems for which small sampling times are needed [2]. MATLAB toolboxes are available that allow to design explicit MPC control laws, see, e.g., [10,59]. Interestingly, the hardware implementation of PWA functions has been deeply studied in circuit theory (see, e.g., [83,104] and the references therein), and the related results can be exploited to setup an integrated design methodology for MPC-based embedded controllers.

The MOBY-DIC Toolbox for MATLAB provides an automatic tool chain
for the circuit design of embedded control systems, given a linear time-invariant model of the physical process, the constraints on states and input variables, and the specifications on the digital circuit actually employed to implement the control law, according to the design flow shown in Fig. B.1. This means that the toolbox allows the designer to have information in advance on the resulting circuit implementation of the controller, in order to ease trading off between control performance, restrictions in area occupancy and speed of implementation. Circuit implementations (with automatic generation of VHDL code) are provided for both exact (optimal) control laws (obtained through the direct interaction of the MOBY-DIC Toolbox with the Hybrid Toolbox [10] or the Multiparametric Toolbox [59]), and for sub-optimal ones based on recent results by [14,40,64]. The sub-optimal methods permit to obtain faster circuit implementations, since they exploit PWA functions defined over regular partitions of the domain, at the cost of loss of both optimality and, in some cases, a-priori closed-loop stability. Methods for the a-posteriori stability analysis of the closed-loop system, and for the evaluation of the domain of attraction (mainly based on [95]), are also provided with the toolbox. Moreover, the toolbox offers several methods for the simulation of the closed-loop system, which allows one to test the designed circuit and, if necessary,
to intervene both on the circuit settings and on the control parameters. The toolbox can also interact with Xilinx System Generator, to obtain a Simulink block representing the controller. In this way, one can simulate the effect of the circuit latency and the quantization error introduced by the fixed point representation.

Finally, the toolbox allows also the automated circuit implementation of direct virtual sensors ([85]), but this functionality will not be described in the present Appendix for brevity.

The toolbox can be freely downloaded.\footnote{http://ncas.dibe.unige.it/software/MOBY-DIC_Toolbox} A complete HTML documentation describing all objects, methods, functions as well as the circuit architectures is also available.

\section*{B.2 Model Predictive Control}

Consider a linear discrete-time system

\begin{equation}
    x(t + 1) = Ax(t) + Bu(t)
\end{equation}

where $x \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$ denote the system states and inputs, respectively. The control input $u(t)$ can be computed using MPC, by solving the following quadratic program

\begin{align}
    \min_{U} & \quad x_N'Px_N + \left( \sum_{k=0}^{N-1} x_k'Qx_k + u_k'Ru_k \right) + \rho \epsilon^2 \\
    \text{s.t.} & \quad x_0 = x(t), \\
    & \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, \ldots, N - 1, \\
    & \quad u_k = Kx_k, \quad k = N_u, \ldots, N - 1 \\
    & \quad E_u u_k \leq G_u, \quad k = 0, \ldots, N_u - 1 \\
    & \quad E_u u_k \leq G_u + V_u \epsilon, \quad k = N_u, \ldots, N - 1 \\
    & \quad E_x u_k + F_x x_k \leq G_x + V_x \epsilon, \quad k = 0, \ldots, N - 1 \\
    & \quad \epsilon \geq 0
\end{align}
where $N$ and $N_u$ are the prediction and control horizons, respectively, $U = \begin{bmatrix} u_0' \cdots u_{N_u-1}' \end{bmatrix}' \in \mathbb{R}^{n_u N_u+1}$ is the vector of optimization variables, $\epsilon$ is a slack variable relaxing the constraints, and $\rho > 0$ is a (large) weight penalizing constraint violations. $E_u, G_u$ and $V_u$ are matrices of appropriate dimensions defining input constraints, while $E_x, F_x, G_x$ and $V_x$ represent mixed input-state constraints. The above problem can be solved off-line as a multi-parametric quadratic program (mpQP) which leads to obtaining $u_0$ (the other elements of $U$ are discarded, according to the receding horizon principle) as an explicit function of the current state (parameter) $x$ [2]. More specifically, the control law $u_0(x)$ is a PWA function of the current state $x_0$ defined over a generic polytopic partition. Other cost functions based on 1-norm and $\infty$-norm can also be employed instead of (B.2), leading to a multi-parametric Linear Program (mpLP). Since also in this case the resulting control law $u_0(x)$ is a PWA function of the state, the approximation and circuit tools presented here can be applied analogously.\footnote{In general, the solution of a mpQP or mpLP can be a PWA function defined over overlapping polytopes; this happens for example in control applications of PWA or mixed logic dynamical systems. This situation cannot be managed by our approximation techniques. Nevertheless, for the applications we are interested in (control of linear systems), the obtained polytopes never overlap.}

### B.3 PWA functions

Given a hyper-rectangular compact domain $D$ partitioned into $n_r$ convex polytopes $P_i$, $i = 1, \ldots, n_r$ a PWA function $f_{\text{pwa}} : \mathbb{R}^n_d \to \mathbb{R}^n_f$ can be expressed as:

$$f_{\text{pwa}}(x) = F_i x + G_i, \quad \text{if } x \in P_i \quad (B.3)$$

where $F_i \in \mathbb{R}^{n_f \times n_d}$ and $G_i \in \mathbb{R}^{n_f \times 1}$, $i = 1, \ldots, n_r$ are the coefficients of the affine expressions over each region. A generic polytope $P_i$ is represented by a set of inequalities $H_i x \leq K_i$, with $H_i \in \mathbb{R}^{n_{e,i} \times n_f}$ and $K_i \in \mathbb{R}^{n_{e,i} \times 1}$; $n_{e,i}$ denotes the number of edges of polytope $P_i$. Therefore, a PWA function is completely defined by all coefficients $F_i, G_i, H_i$ and $K_i$. PWA functions defined over any irregular polytopic domain partition are referred to as PWAG functions.
B.3. PWA functions

Particular cases of domain partitions (see Fig. B.2) are the simplicial partition, leading to PWAS functions, and the hyper-rectangular partition, leading to PWAR functions.

- **PWAG functions**: the evaluation of a PWAG function at a point $x$ is performed in two steps: (i) locate the polytope $P_i$ containing $x$ (point location problem) and (ii) compute the affine expression $F_i x + G_i$. The second step is trivial; a higher computational effort is required for the point location problem. Some efficient algorithms, besides the inefficient direct search, have been proposed in the literature (see for example [9, 107]). We implemented in the toolbox the algorithm by [107], in which a binary search tree is constructed off-line, based on the domain partition. Each non-leaf node of the tree corresponds to a partition edge and each leaf node to a polytope. The tree is then explored on-line to locate the region containing a given point. The binary search tree can be easily mapped into a Finite State Machine, for a digital circuit implementation, as described in [81]. An example of PWAG function is shown in Fig. B.5(a).

- **PWAS functions**: a simplicial PWA (PWAS) function is a PWA function defined over a domain partitioned into regular simplices. A simplex is a segment in one dimension, a triangle in two dimensions, a tetrahedron in three dimensions and so on. A simplicial partition is built by subdividing each domain component into $p_j$ segments, $j = 1, \ldots, n_d$ with arbitrary length (if all segments have the same length we obtain uniform partitions); the domain is therefore partitioned into $\prod_{j=1}^{n_d} p_j$ hyper-rectangles which are further partitioned into $n_d$ simplices each. Any continuous PWAS function can be represented as a weighted sum of PWAS basis functions in this way:

$$f_{\text{pwas}}(x) = \sum_{i=1}^{n_b} w_i \alpha_i(x)$$  \hfill (B.4)

where $w_i$ are the weights and $\alpha_i$ are the PWAS basis functions defined as $\alpha_i(v_j) = \delta_{ij}$, being $v_j, j = 1, \ldots, n_b$ the vertices of the simplicial partition and $\delta_{ij}$ the Kronecker delta. Therefore, any PWAS function is univocally characterized by its domain, the simplicial partition and the set of weights $w_i, i = 1, \ldots, n_b$. In case of $\alpha$ basis functions, the weights $w$ correspond to the values of the PWAS function in the vertices of the simplicial partition. The value of the function in any point $x$ can be computed by linear interpolation
of the value of the function in the vertices of the simplex containing the point. The location of the simplex (point location problem) can be carried on in a very simple way, due to the regularity of the partition [83]. Digital architectures for the computation of PWAS functions have been proposed in [104] and they are employed in the toolbox. An example of a PWAS function is shown in Fig. B.5(b).

- **PWAR functions**: a hyper-rectangular PWA (PWAR) function is a PWA function defined over a domain partitioned into hyper-rectangles. We consider two kinds of PWAR functions: the single- (sPWAR) and the multi-resolution (mPWAR) type. The single-resolution type is characterized by the fact that the axes $x_j$, $j = 1, \ldots, n_d$ of domain $\mathcal{D}$ are divided into $m_j$ intervals, possibly of different lengths, in which case the partition is non-uniform; the regions are stacked like the components of a matrix, as shown in Fig. B.2(c). In contrast, the multi-resolution type of partition is built by selecting hyper-rectangular regions from multiple single-resolution partitions of $\mathcal{D}$; these $r$ partitions have $m_j = 2^l$, $l = 1, \ldots, r$ divisions along each axis (for circuit implementation reasons), as shown in Fig. B.2(d). Because of its hierarchical structure, this type of partition has a corresponding search tree (a generalized oct-tree), which is very beneficial for solving the point location problem. Digital architectures for the evaluation of PWAR functions have been proposed by [30] and are implemented in the toolbox. An example of a PWAR function is shown in Fig. B.5(c).

### B.4 Description of the toolbox

The MOBY-DIC Toolbox is a MATLAB/Simulink toolbox which makes extensive use of Object Oriented Programming. In the following, the main objects for the representation of a linear system, for the definition of constraints and for the description of PWA functions are described.

- **linearSystem**: this object represents a discrete time affine system in the form

$$x(t + 1) = Ax(t) + Wp + Bu(t) + f$$
$$y(t) = x(t)$$
B.4. Description of the toolbox

Figure B.2: Examples of two-dimensional domain partitions: generic (a), simplicial single-resolution (b), hyperrectangular single- (c) and multi-resolution (d). The corresponding classes of PWA functions are: PWAG (a), PWAS (b), sPWAR (c), and mPWAR (d).

where $p \in \mathbb{R}^{n_p}$ is a vector of fixed parameters and $y_k \in \mathbb{R}^{n_x}$ represents the system outputs (always identical to the system states); $A$, $W$, $B$ and $f$ are matrices of appropriate dimensions. The object is created by specifying the number of state variables ($n_x$), parameters ($n_p$) and inputs ($n_u$); the value
of the matrices and the sampling time are then set by means of methods 
\textit{setMatrices} and \textit{setSamplingTime}, respectively. Mnemonic names can also be 
assigned to states, parameters and inputs. Methods \textit{simplot} and \textit{sim} allow to 
simulate the linear system in closed-loop with any PWA controller, with or 
without time evolutions plotting.

- **constraints**: the \textit{constraints} object allows to set constraints on system 
states, inputs and parameters for any time instant within a time horizon \( N_c \).
The general form of the constraints is:

\[
H_c \begin{bmatrix} x_k \\ u_k \\ p \end{bmatrix} \leq K_c
\]

with \( k = 0, \ldots, N_c \). This formulation includes typical saturation constraints 
in the form \( x_{\text{MIN}} \leq x_k \leq x_{\text{MAX}}, p_{\text{MIN}} \leq p \leq p_{\text{MAX}} \) or \( u_{\text{MIN}} \leq u_k \leq u_{\text{MAX}} \).
The object is created by specifying \( n_x, n_u, n_p \) as well as the horizon \( N_c \). Each 
constraint is added to the object via method \textit{setConstraints}, which allows a 
very simple and flexible usage.

- **pwaFunction**: the \textit{pwaFunction} object is an abstract class (not instan-
tiable directly), which represents any PWA function \( f_{\text{pwa}} : \mathbb{R}^{n_d} \rightarrow \mathbb{R}^{n_f} \). Inher-
ited objects are \textit{pwag}, \textit{pwas} and \textit{pwar}. In this object the common properties 
to all derived classes are declared: the function domain (\( D \)) and its dimension 
(\( n_d \)), the number of dimensions of the codomain (\( n_f \)), and a structure (\textit{syn-
thesisInfo}) with details on the circuit implementation of the function. The 
most important common methods to all derived classes are \textit{eval}, for the eval-
uation of the function in an array of points, \textit{getRegions}, which returns all the 
regions partitioning the domain, \textit{synthesize}, which generates the VHDL files 
describing a circuit implementing the function, and \textit{generateSimulinkModel}, 
for the automatic generation of a Simulink model of the closed-loop system.

- **pwag**: the \textit{pwag} object (derived from the \textit{pwaFunction} class) repre-
sents a PWAG function. In this object, three structures are used to store 
all data, \textit{edges}, \textit{functions} and \textit{regions}. Structure \textit{edges} stores all edges 
of the polytopes without repetitions and without considering the sign of the

---

\(^3\)The constraints involving only \( x \) can be imposed from time \( k+1 \) to \( k+N_c \); the constraints 
involving also \( u \) can be imposed from time \( k \) to \( k + N_c - 1 \).
edge. The coefficients of the edges are stored in fields H and K of structure edges. Since the polytopes have, in general, some common edges, many repetitions in the coefficients \( H_i \) and \( K_i \), also with opposite sign, can occur; the approach we used allows, therefore, to save memory occupation sensibly. Structure functions stores, in fields F and G, the coefficients \( F_i \) and \( G_i \) of all affine functions. As for previous structure, coefficients related to the same affine function are saved only once. regions is actually an array of \( n_\alpha \) structures, one for each polytope partitioning the domain. These structures have two fields: Iedges and Ifunctions. The Iedges field of the i-th element of regions is a matrix with \( n_{e,i} \) rows and 2 columns. The first column contains the indices of the coefficients \( H_i \) and \( K_i \) in matrices edges.H and edges.K. The second column contains the sign of the edges. In the same way, the field Ifunctions contains the indices of \( F_i \) and \( G_i \) in matrices functions.F and functions.G. The pwag object contains also a structure tree which stores the binary search tree necessary for a circuit computation of the function ( [81]).

The structure contains a field nodes which is an array of structures whose elements correspond to the nodes of the tree. Each node is characterized by a univocal name and by the names of his children nodes; it contains moreover information about the set of edges and polytopes lying at the two sides of the edge corresponding to the node itself.

- **pwas**: the pwas object (derived from the pwaFunction class) represents a PWAS function. Any PWAS function is univocally defined by the domain partition and by the set of weights \( w_\alpha \); in this toolbox we only refer to PWAS functions described through \( \alpha \) basis functions since they can be handled in a computationally efficient way (with sparse matrices). The simplicial partition is represented, in the MATLAB object, with a cell array \( P \) with as many elements as the function domain dimensions. The i-th entry of \( P \) contains the projection of the vertices of the partition along the i-th domain component. The weights are stored in an ordered one-dimensional array \( w \).\(^5\)

- **pwar**: the pwar object (derived from the pwaFunction class) represents a PWAR function. This object has seven properties: H, K, F, G, tree and details. The main information is stored in cell arrays H, K, F, and G, with

\(^4\) \( Hx \leq K \) and \( -Hx \leq -K \) denote the same edge, with opposite sign.

\(^5\) If the codomain is multi-dimensional, \( w \) is a matrix with as many columns as the number of dimensions of the codomain.
$n_r$ elements, which contain the corresponding coefficients $F_i$, $G_i$, $H_i$ and $K_i$, $i = 1, \ldots, n_r$. The matrix `tree` stores the information about the search tree necessary for circuit implementation of the function. This information is also used to speed up point location in some functions of the toolbox. The structure `details` is used to store various types of information, such as the original options used for the approximation, the type of partition (single- or multi-resolution), etc.

### B.5 Functionalities

In this subsection the main functionalities of the toolbox are illustrated. Figure B.3 shows how objects and functions are used to automatically generate a digital circuit starting from the definition of the system to control and of the constraints to impose. The functionalities are described in the following subsections.
B.5. Functionalities

B.5.1 Design of MPC PWAG controller

The MOBY-DIC Toolbox provides a simple and flexible interface both to Hybrid Toolbox ([10]) and Multi Parametric Toolbox ([59]), for the design of explicit MPC controllers. Given a linearSystem object, a constraints object and some additional options, the generateMPC function returns a pwag object representing the explicit MPC control law, by resorting to one of the aforesaid toolboxes, which solve problem (B.2). The additional options, provided by means of a structure, are the prediction and control horizons, the cost function matrices and the type of norm for the MPC optimization problem, and some other settings specific to each toolbox. Default values are automatically set in case an option is not provided.

B.5.2 Design of approximate control laws

An approximate MPC controller can also be designed starting from the exact MPC control law. Two methods are available to accomplish to this task.

- **PWAS controller**: the pwasApproximation function allows to obtain a pwas object that approximates the pwag one by using the method described in [14]. The same constraints satisfied by the exact MPC law are also enforced in the approximate one, though possibly softened with slack variables. The routine needs the pwag, linearSystem and constraints objects and the desired simplicial partition and returns a pwas object. Some options can be provided to this function, such as the norm to use in the optimization problem and the solver to be employed. Once the approximation has been found, the maximum approximation error can be computed with function suboptimalityAnalysis, as explained in [14].

- **PWAR controller**: the pwarApproximation function allows to approximate the pwag object by a pwar one by using the methods described in [40] and [64]. The same constraints satisfied by the exact MPC law are also enforced in the approximate one, and constraints on stability and performance can be enforced depending on the method used. The routine needs the pwag, linearSystem and constraints objects and returns a pwar object. Some options can be provided to this function, such as the norm to use in the optimization problem and the solver to be employed. To facilitate setting these options, two functions are included. The pwarApproxSettings function can be used to
create an options structure in which the non specified entries are filled with default values, whereas \texttt{pwarApproxSettingsCheck} can be used to check for inconsistencies and commonly made errors in the options structure.

\subsection*{B.5.3 Stability analysis of the closed-loop}

Most of the described design methods for approximate MPC control law are aimed at practical implementation on digital circuits, and therefore do not incorporate conditions that guarantee the a-priori stability of the closed-loop system. For this reason, the function \texttt{stabilityTest} is provided. The method employed by this function is based on the use of PWA Lyapunov functions, which permits to certify the stability of the origin and to find an evaluation of the domain of attraction as a polytope. Also the presence of additive disturbances (not taken into account during the MPC design) can be considered. Since, in case of persistent disturbances, the convergence to the origin cannot be assured, \texttt{stabilityTest} will evaluate the set including the origin, where the state will be ultimately bounded. If the stability test fails, a new approximation can be obtained with a finer domain partition and the test can be repeated. The stability test can also be performed for the exact controller (not only for the approximate one) in case it is designed without stability guarantees. For more details, the reader is referred to [95,96].

\subsection*{B.5.4 Circuit implementation of digital controllers}

One of the main features of MOBY-DIC Toolbox is the possibility of automatically generating VHDL code for the circuit implementation (on FPGA) of exact and approximate MPC controllers. Serial and parallel architectures are available for PWAG, PWAS and PWAR controllers, which allow to trade-off between circuit complexity and speed. A description of the architectures can be found in [30,81,104] and in the toolbox documentation. The VHDL files are generated by resorting to method \texttt{synthesize} of a \texttt{pwag}, \texttt{pwas}, or \texttt{pwar} object once the circuit specifications (number of bits for inputs and outputs, type of input acquisition, circuit sampling time) have been provided. The synthesis process also generates a log file in which information about the circuit performances is shown: input/output representation, latency, number of multipliers, memory occupation. The generated files include also a testbench.
for the circuit so that it can be simulated by any VHDL simulator (e.g., Modelsim). The results of the simulation can be read by MOBY-DIC Toolbox which automatically performs the suitable data conversions and scalings.

**B.5.5 Closed-loop simulation of controllers**

Simulations of the closed-loop system with exact or approximate MPC controller can be performed in several ways with this toolbox. The discrete-time controlled system can be simulated in MATLAB by means of methods `sim` and `simplot` of `linearSystem` object. If the discrete-time system was obtained as a discretization of a continuous-time plant, for the purpose of generating the MPC control law, it could be useful to simulate directly the continuous-time plant in closed-loop with the exact or approximate controller. This task is accomplishable with method `generateSimulinkModel`, which automatically generates a Simulink model ready to be simulated. If the VHDL files have been generated for the controller, it is also possible (always through method `generateSimulinkModel`) to directly simulate them for the computation of the controller in the closed-loop system; this allows to take into account the effects of circuit latency (delays) and of the fixed point representation (quantization error). Xilinx System Generator is required for this functionality.

**B.6 A simple case study**

In this subsection we illustrate the functioning of the toolbox in a very simple example, in order to focus on the capabilities of the software rather than on the control problem itself. MOBY-DIC Toolbox has been also used, however, in more complex applications such as Adaptive Cruise Control systems and Buck-Boost DC-DC converters ([30,80,103]). Let us consider the continuous-time double integrator system defined by the following state equations:

\[ \dot{x}(t) = A_c x(t) + B_c u(t) \]  

(B.5)
with $A_c = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ and $B_c = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, subject to soft constraints

$$-1 \leq u \leq 1$$

$$\begin{bmatrix} -8 \\ -4 \end{bmatrix} \leq x \leq \begin{bmatrix} 8 \\ 4 \end{bmatrix}$$

(B.6)

In order to compute the MPC control function we discretized system (B.5) with sampling time $T_s = 1$, thus obtaining:

$$x(t + 1) = A_dx(t) + B_du(t)$$

(B.7)

with $A_d = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ and $B_d = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. The system has 2 state variables ($n_x = 2$), one input variable ($n_u = 1$) and zero parameters $n_p = 0$. We computed the MPC control law with Hybrid Toolbox by setting $N = N_u = 5$, $Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and $R = 0.1$, thus obtaining a $pwag$ object. The following snippet of code shows how these steps are performed:

```matlab
% Create linearSystem object
linSys = linearSystem(nx,nu,np);
% Set discrete-time system matrices
linSys = linSys.setMatrices('A',Ad);
linSys = linSys.setMatrices('B',Bd);
% Set sampling time
linSys = linSys.setSamplingTime(Ts);
% Create constraints object
constr = constraints(nx,nu,np,N);
% Set input constraints from k to k+Nu-1
constr = constr.setConstraints('u1',-1,1,0:Nu-1);
% Set state constraints from k+1 to k+N
constr = constr.setConstraints('x1',-8,8,1:N);
constr = constr.setConstraints('x2',-4,4,1:N);

% MPC options
MPCoptions = struct( ...
    'toolbox','hybtbx', ... % Use Hybrid Toolbox
    'N',N, ... % Prediction horizon
)```
B.6. A simple case study

% Control horizon
'Nu',Nu, ...
% Cost function matrix Q
'Q',Q, ...
% Cost function matrix R
'R',R, ...

% Generate pwag object representing the MPC control law
Cpwag = generateMPC(linSys,constr,MPCoptions);

Since closed-loop stability and constraint satisfaction are not guaranteed a priori, stabilityTest is employed to get a stability certificate for the origin, together with an estimate of its domain of attraction, shown in Fig. B.4.

% Generate the closed-loop system with PWAG controller
CpwagCL = Cpwag.getClosedLoop(linSys);
% Check stability of the closed-loop system
[XPg XFg] = stabilityTest(CpwagCL);

Then we approximate the exact MPC control law with a PWAS function defined over a non-uniform simplicial partition and we compute the maximum approximation error $M = 0.385$ at point $x = [4.83 - 2.26]$.

% Choose simplicial partition (non-uniform)
P = cell(1,2);
P1 = [linspace(-8,-0.5,8) linspace(0.5,8,8)];
P2 = [-4 linspace(-2.275,-0.2,7) linspace(0.3,3.619,7) 4];
% Choose domain for the pwas function
D = Cpwag.getDomain(); % Use the same as the pwag function

% Approximation options
appr_opts = struct( ...
    'norm',2, ... % Use 2 norm for approximation
    'solver','cvx', ... % Use CVX solver
    'equality',2, ... % Impose equality constraints
    'multithreading',1 ... % Enable multithreading
);
% Perform the approximation
Cpwas = pwasApproximation(Cpwag,linSys,constr,D,P,appr_opts);

% Find maximum approximation error
[M x] = suboptimalityAnalysis(Cpwas,Cpwag);
Figure B.4: Estimate of the domain of attraction of closed-loop system with PWAG controller.

Next, a multi-resolution PWAR approximation is obtained by imposing a maximum error of 0.24.

```
% Approximation options
appr_opts = pwarApproxSettings( ... 
    'norm',inf,... % Use inf norm for approx.
    'rhomax',0.24,... % Maximum approx. error
    'con_input', 1,... % Input constraints enforced
    'resolutiontype', 'multi',... % Type of resolution used
    'minlevel', 1,... % Minimal level of refinement
    'maxlevel', 5,... % Maximal level of refinement
    'solver', 'cdd',... % Use CDD solver
    'multithreading', 1... % Enable multithreading
```
B.6. A simple case study

Figure B.5: Exact PWAG (a) and approximate PWAS(b) and PWAR(c) control laws.

\begin{verbatim}
); % Perform the approximation
Cpwar = pwarApproximation(Cpwas, linSys, constr, D, options);
\end{verbatim}

The plots (obtained with method \textit{plot}) of the exact and approximate control laws are shown in Fig. B.5.

The following piece of code shows how to generate VHDL files for the FPGA implementations of exact and approximate controllers.

\begin{verbatim}
% Circuit specifications
cir_specs = struct( ...
'\text{\textit{nbit}}',12, ... % N. of bits for inputs
'\text{\textit{type}}','\text{\textit{serial}}', ... % Type of architecture
'\text{\textit{inputAcquisition}}','\text{\textit{parallel}}', ... % Input acquisition
'\text{\textit{frequency}}',20e6, ... % Working frequency
'\text{\textit{samplingTime}}',Ts, ... % System sampling time ...
);

% Generate VHDL files for exact and approx. controllers
Cpwas = Cpwas.synthesize(cir_specs);
Cpwar = Cpwar.synthesize(cir_specs);
\end{verbatim}

The circuit performance of both serial and parallel architectures, as provided by MOBY-DIC Toolbox, are shown in Table B.1.
<table>
<thead>
<tr>
<th>Architecture</th>
<th>Latency @20MHz</th>
<th>Mem. occup.</th>
<th>N. mult.</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwag_ser</td>
<td>1850 ns</td>
<td>175.5 bytes</td>
<td>1</td>
</tr>
<tr>
<td>pwag_par</td>
<td>950 ns</td>
<td>175.5 bytes</td>
<td>2</td>
</tr>
<tr>
<td>pwas_ser</td>
<td>400 ns</td>
<td>384 bytes</td>
<td>3</td>
</tr>
<tr>
<td>pwas_par</td>
<td>250 ns</td>
<td>1152 bytes</td>
<td>5</td>
</tr>
<tr>
<td>pwar_ser</td>
<td>450 ns</td>
<td>585 bytes</td>
<td>1</td>
</tr>
<tr>
<td>pwar_par</td>
<td>350 ns</td>
<td>585 bytes</td>
<td>2</td>
</tr>
</tbody>
</table>

Table B.1: Performances of the circuit architectures implementing PWAG, PWAS and PWAR controllers

Finally, a Simulink model for the simulation of the closed-loop system considering circuit delays and fixed point representation has been generated.

```matlab
% Model options
model_opts = struct( ...
    'simulateVHDL',1, ... % Simulate VHDL files
    'samplingTime',Ts, ... % Sampling time of the system
);
% Continuous time plant
plant = ss(Ac,Bc,eye(2),0);
% Initial condition for simulation
x0 = [-4 -2];
% Generate model for exact and approx. controllers
Cpwag.generateSimulinkModel(plant,x0,model_opts)
Cpwas.generateSimulinkModel(plant,x0,model_opts)
Cpvar.generateSimulinkModel(plant,x0,model_opts)
```

Figure B.6 shows the results of the Simulink simulation and evidences the correct regulation of the system to the origin. In case of approximate controllers, the design flow listed above corresponds to the one shown in figure B.1(b). Indeed, the control law is derived taking into account both control specifications and circuit complexity either by setting the simplicial partition (i.e., the cell array P) for the PWAS approximation, or by setting the maximum level of refinement for the PWAR approximation. It is important to note that only the results obtained with the PWAS controller are shown for brevity. Almost identical plots have been obtained with PWAG and PWAR controllers.
B.7. Conclusions and future work

stress that the circuit performances are known before actually implementing the controller on FPGA and they can be known even before designing the controller itself. The user can therefore redesign or change the approximation parameters in order to meet circuit specifications. Further tools to ease this task are under development and will be available in a future release with a Graphical User Interface.

Figure B.6: Simulink simulations of the closed-loop system (plant and PWAS controller). The top panel shows the evolution of the states, the bottom panel the evolution of the input computed with VHDL simulation.

B.7 Conclusions and future work

The Appendix introduced a complete software toolbox in MATLAB for the automatic generation of VHDL code describing embedded control systems, and illustrated its capabilities through a simple case study. The toolbox is still in phase of development and some capabilities will be added, such as the synthesis of a-priori stabilizing PWAS controllers, see Appendix C for an overview. A complete Graphical User Interface (GUI) will be designed to facilitate the use of the toolbox and to graphically monitor the circuit performance as a function of control and circuit parameters.
Appendix C

PWAS stable approximation of explicit MPC

C.1 Introduction

Model predictive control (MPC) is becoming increasingly popular both in academia and in industry due to its ability to solve control problems optimally while satisfying constraints on state and control variables (see, e.g., [21] and the references therein). The main drawback of MPC is the computation time required for solving on-line an optimization problem, which has historically prevented its application to fast processes [86].

To circumvent this problem, two main research directions were pursued in the last decade (we limit our overview to the control of linear time-invariant (LTI) systems, that are the subject of this Appendix). The first relates to fast algorithms for on-line optimization (see, e.g., [15, 38, 92, 115]). The second regards computing the control law off-line as an explicit piecewise-affine (PWA) function of the state vector (see, e.g., [13], [2]): the off-line computation employs a multiparametric programming solver, thus achieving the same solution obtained by solving an optimization problem on-line. The on-line computation in explicit MPC relies on determining the region of the PWA partition where the current state value is located (usually referred to as the point location problem, which takes usually a high percentage of the overall on-line computation time), and then on evaluating an affine function from a pre-stored lookup table. The number of regions of the PWA partition which defines the explicit MPC control law depends typically exponentially on the number of constraints included in the multiparametric program. For this reason, explicit MPC is usually applied to control processes of relatively small size (approximatively, up to 2 manipulated inputs and 10 state variables). However, it permits to use short sampling periods, such as 1-50 ms. To simplify the complexity of explicit MPC controllers, approximate explicit MPC
techniques have been considered ([2,29,45,52]). In these approaches, optimality is sacrificed for a control law defined over a smaller number of regions. In order to obtain even faster controllers, explicit MPC has been recently implemented on hardware devices, such as field programmable gate arrays (FPGAs) in [67], pushing the time needed to compute the control law (latency) down to hundreds of nanoseconds.

In a recent paper [14], an approximate MPC controller for LTI systems is proposed, based on a special class of functions, hereafter referred to as piecewise-affine simplicial (PWAS) functions, proposed by [54]. The choice of PWAS functions leads to a regular partition, where the point-location problem is solved with a negligible effort if compared to explicit MPC defined on generic PWA partitions. For this reason, PWAS functions can be efficiently implemented on digital circuits [104], which allows an extremely fast computation of the control law. More precisely, for an example with two state variables and two control inputs [14], the latency for an FPGA implementation could be reduced by one order of magnitude with respect to the implementation of the exact controller on the same FPGA (from hundreds to tens of nanoseconds). The control law proposed in [14] presents feasibility and local optimality properties, but the asymptotic stability of the origin of the closed-loop system and the evaluation of its domain of attraction can be determined only a-posteriori, using Lyapunov-based techniques (see also [96]). In order to define stabilizing approximate MPC controllers on regular partitions, two approaches based on the use of PWA hyper-rectangular partitions have been recently proposed in [39, 64], based on ISS Lyapunov functions and control Lyapunov functions, respectively.

In this Appendix, we propose an approximate explicit MPC control approach for LTI systems based on PWAS functions, which can be implemented on a digital circuit exactly as in [14]. However, we guarantee the asymptotic stability of the resulting closed-loop system a-priori, also obtaining the domain of attraction in which hard constraints on state and input variables are satisfied.

More specifically, a suitable bound is imposed a-priori on the approximation error. Then, a robust MPC control law $u^*(x)$ based on tightened constraints is defined, based on ideas of [28,57]. The obtained optimal con-
Notation can be implemented as an explicit PWA control law, similarly to the nominal case in [13]. After obtaining the optimal control law, an approximation procedure is carried out in order to find an approximate PWAS control law $u(x)$, such that the approximation error $u(x) - u^*(x)$ satisfies the previously-defined bound, which implies the asymptotic stability of the closed-loop system.

The Appendix is organized as follows: the main notation used throughout the work and the formulation of the control problem are introduced in Subsections C.2 and C.3, respectively. Subsection C.4 describes the structure of the PWAS control law, and gives a brief overview of its implementation on digital circuits. In Subsection C.5, the synthesis of the robustly stabilizing MPC control law is described, while Subsection C.6 deals with the approximation procedure leading to the stabilizing PWAS control law. In Subsection C.7 the proposed control law is synthesized for a simple system and tested in simulation. Finally, conclusions are drawn in Subsection C.8.

C.2 Notation

The sets $\mathbb{Z}_{>0}$, $\mathbb{Z}_{\geq0}$, $\mathbb{R}$, $\mathbb{R}_{>0}$ are the sets of positive integers, non-negative integers, real, and positive real numbers, respectively. Given a set $A \subset \mathbb{R}^n$, its interior is referred to as $\text{int}(A)$. Given two sets $A$ and $B$, let $A \oplus B \triangleq \{a+b : a \in A, b \in B\}$ and $A \sim B = \{a : a + b \in A, \forall b \in B\}$ be their Minkowski addition and Pontryagin difference, respectively. Given two vectors $u, v \in \mathbb{R}^n$, the notation $u \leq v$ refers to component-wise inequalities. Given , its Positive definiteness of a square matrix $M \in \mathbb{R}^{n \times n}$ is indicated as $M \succ 0$. Given a vector $v \in \mathbb{R}^n$ and a matrix $M \in \mathbb{R}^{n \times n}$, $\|v\|^2_M \triangleq v' M v$. Given a matrix $M \in \mathbb{R}^{n \times m}$ and a compact set $W \subset \mathbb{R}^m$, the product $MW$ denotes the image of $W$ under the mapping defined by $M$, $MW \triangleq \{v \in \mathbb{R}^n : v = Mw, w \in W\}$. In case $W$ is a polytope, $MW$ can be computed as the convex hull of the images of the vertices of $W$.

When convenient, the explicit dependence on time of the dynamic variables will be omitted for the sake of readability of the work.
C.3 Problem statement

The controlled plant is described by the following LTI state space model

\[ x(t + 1) = Ax(t) + Bu(t) \]  \hspace{1cm} (C.1)

where \( t \in \mathbb{Z}_{\geq 0} \), \( x \in \mathbb{R}^n \), \( u \in \mathbb{R}^m \), \( A \in \mathbb{R}^{n \times n} \), \( B \in \mathbb{R}^{n \times m} \). It is assumed that the whole state vector \( x \) is available for feedback. The state and input values are required to satisfy

\[ x \in X, \quad X \triangleq \{ x \in \mathbb{R}^n : C_x x \leq g_x \} \]
\[ u \in U, \quad U \triangleq \{ u \in \mathbb{R}^m : C_u u \leq g_u \} \]

with \( C_x \in \mathbb{R}^{s_x \times n} \), \( C_u \in \mathbb{R}^{s_u \times m} \), \( g_x \in \mathbb{R}^{s_x} \), \( g_u \in \mathbb{R}^{s_u} \).

Assumption 4. The following holds for system (C.1):

1. the pair \((A, B)\) is stabilizable;
2. \( X \) and \( U \) are nonempty, compact, and contain the origin in their interior. \(\square\)

The objective is to regulate the state \( x \) to the origin without violating the constraints (C.2)-(C.2), using a control profile \( u(x) \) defined on a PWAS partition, whose structure is described in the next section.

C.4 Control law on a simplicial partition

This section is devoted to the description of the control law \( u(x) \) as a PWAS function, and on its practical implementation on hardware devices.

C.4.1 Description of the control law

The function \( u(x) \) is defined on a closed hyper-rectangle \( S \subset \mathbb{R}^n \), \( S = \{ x \in \mathbb{R}^n : \min x \leq x \leq \max x \} \), which is partitioned as \( S = \bigcup_{i=0}^{L_S-1} S_i \), where \( \{ S_i \}_{i=0}^{L_S-1} \) are simplices. A simplex \( S_i \) in the Euclidean space \( \mathbb{R}^n \) is a polytope given by the convex hull of its \( n + 1 \) vertices \( x_i^0, x_i^1, \ldots, x_i^n \in \mathbb{R}^n \). The partitioning of \( S \) is performed as follows:
1. Divide every dimensional component $x_j$ of $S$ into $p_j$ subintervals of length $(x_{\text{max},j} - x_{\text{min},j})/p_j$. These intervals define a number $\prod_{j=1}^n p_j$ of hyper-rectangles, and $S$ contains $N_v \triangleq \prod_{j=1}^n (p_j + 1)$ vertices $v_k$, collected into a set named $V_S$.

2. Partition every rectangle into $n!$ simplices with non-overlapping interiors. The set $S$ contains $L_S \triangleq n! \prod_{j=1}^n p_j$ simplices $S_i$, such that $S = \bigcup_{i=0}^{L_S - 1} S_i$ and $\text{int}(S_i) \cap \text{int}(S_j) = \emptyset$, $\forall i, j = 0, \ldots, L_S - 1$.

Since the partitioning of the hyper-rectangles into simplices is univocally determined, the resulting number of simplices is determined by $p_1, \ldots, p_n$. After defining the sets $S_i$, it is possible to introduce the related PWAS function. We choose to define each component of $u(x)$, namely $u_j(x)$, $j = 1, \ldots, n$, as the weighted sum of $N_v$ linearly independent basis functions. Even though different basis functions can be used (see, e.g., [90,105]), in this work we refer to the so-called $\alpha$-basis ([54]). Every element of the $j$-th basis is affine over each simplex and satisfies

$$\alpha_{j,k}(v_h) = \begin{cases} 1 & \text{if } h = k \\ 0 & \text{if } h \neq k. \end{cases}$$

After ordering the functions of the $\alpha$-basis, we can consider them as an $N_v$-length vector $\phi(x)$. Then, each component of $u(x)$, namely $u_j(x)$, is a scalar PWAS function defined as

$$u_j(x) \triangleq \sum_{k=1}^{N_v} \theta_{j,k} \phi_k(x) = \phi(x)' \theta_j$$

where $\theta_j = [\theta_{j,1} \ldots \theta_{j,N_v}]' \in \mathbb{R}^{N_v}$ is the weight vector. Note that the coefficients $\theta_{j,k}$ coincide with the values of the PWAS function $u_j(x)$ at the vertices of the simplicial partition. The PWAS vector function $u : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is defined by the weight vector $\theta = [\theta_1' \theta_2' \ldots \theta_m']' \in \mathbb{R}^{mN_v}$, as

$$u(x) = \begin{bmatrix} u_1(x) \\ \vdots \\ u_m(x) \end{bmatrix} \triangleq \begin{bmatrix} \phi(x)' \theta_1 \\ \vdots \\ \phi(x)' \theta_m \end{bmatrix} = \begin{bmatrix} \phi'(x) & 0 & \cdots & 0 \\ 0 & \phi'(x) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \phi'(x) \end{bmatrix} \theta = \Phi(x) \theta.$$ 

(C.3)
C. PWAS stable approximation of explicit MPC

C.4.2 Implementation on digital circuits

The main reason for defining $u(x)$ as in (C.3) is that PWAS functions can be implemented in digital circuits using linear interpolators. In fact, by exploiting the regularity of the partition, the point location problem becomes much easier than for the case of generic PWA partitions. The value of $u(x)$ can be obtained, for any $x \in \mathcal{S}$, as a linear interpolation of the values $u(v_{i,j})$ at the $n + 1$ vertices $v_{i,0}, ..., v_{i,n}$ of the simplex $S_i$ that contains $x$,

$$u(x) = \sum_{j=0}^{n} \mu_j u(v_{i,j}). \quad (C.4)$$

With reference to FPGA implementations [104], the circuit realization is given by three functional blocks:

(a) an internal memory that stores the $N_v$ weights $\theta_k$;

(b) a block that finds the simplex $S_i$ to which $x$ belongs (using the algorithm proposed in [94]) and computes the coefficients $\mu_j$ (this step requires a sorting algorithm designed for digital implementation, see, e.g., [84]);

(c) a block that computes the weighted sum in (C.4), using high-speed multipliers and accumulators in the FPGA.

The latency of the described circuit (i.e., the time needed to obtain $u(x)$ once the measurement of $x$ is available) does not depend on the number of intervals $p_j$ (which influences the area occupancy). However, the latency grows linearly with $n$ and the number of bits used to encode $x$.

C.5 Robustly stabilizing optimal MPC

The next step is to obtain a PWAS function $u(x)$ as in (C.3) using a procedure that leads to asymptotic convergence to the origin for the closed-loop system. The proposed approach consists in expressing the control variable $u(x)$ as

$$u(x) = u^*(x) + w(x)$$

where $u^*(x)$ is an optimal control which satisfies

$$u^* \in \mathcal{U}, \quad (C.5)$$
while $w(x)$ represents an approximation error (a priori unknown), and is considered as a bounded disturbance. System (C.1) can therefore be expressed as

$$x(t + 1) = Ax(t) + Bu^*(t) + Bw(t) \quad (C.6)$$

### C.5.1 Definition of a linear auxiliary control law

In order to formulate the MPC control law $u^*(x)$, we define an auxiliary control law $\kappa(x) = Kx$, where $K$ is such that the closed-loop system

$$x(t + 1) = (A + BK)x(t) \quad (C.7)$$

is asymptotically stable. The value of $K$ can be determined by pole-placement techniques, or it can be the solution for system (C.7) of the infinite-horizon linear quadratic regulator, given weight matrices $Q \in \mathbb{R}^{n \times n}$ on the state and $R = R' \in \mathbb{R}^{m \times m}$ on the input, with $Q, R \succ 0$. The resulting closed-loop system obtained by applying $u^*(x) = \kappa(x)$ in (C.6) is

$$x(t + 1) = A_\kappa x(t) + Bw(t), \quad (C.8)$$

where $A_\kappa \triangleq A + BK$. This control law will be used in the remainder of the work as baseline to design the MPC controller.

### C.5.2 Definition of invariant sets

We want to define a robust MPC control law $u^*(x)$, mainly based on the approach described in [28], which leads to robust convergence of the state to the origin without violating constraints (C.2) and (C.5). Let $\bar{w} \in \mathbb{R}_{\geq 0}$ be a fixed scalar such that

$$w(x) \in W \triangleq \{w \in \mathbb{R}^m : \|w\|_\infty \leq \bar{w}, \forall x \in \mathcal{X} \}. \quad (C.9)$$

We now find the minimal robust positively invariant (RPI) set (see, e.g., [17]) for the closed-loop system (C.8). We denote by

$$\mathcal{R}_j \triangleq \bigoplus_{i=0}^{j-1} A_\kappa^i BW \quad (C.10)$$
the set of states of system (C.8) reachable in \( j \) steps from the origin. Define

\[ R_\infty \triangleq \lim_{j \to \infty} R_j \]

as the minimal RPI set for system (C.8) with \( w \in W \). Relying for instance on [17, Proposition 6.9], one can prove that \( R_\infty \) is bounded and convex in our case, as \((A,B)\) is a reachable pair, \( A_\kappa \) is asymptotically stable, \( W \) is convex, compact, and includes the origin in its interior. Nonetheless, an explicit computation of \( R_\infty \) is in general impossible (apart from the very specific case of \( A_\kappa \) nilpotent, as stated by [71]). Therefore, it is useful to compute a convex polytopic over-approximation (not necessarily RPI) \( \hat{R}_\infty \) such that \( R_\infty \subseteq \hat{R}_\infty \). There exist different methods to compute \( \hat{R}_\infty \), and the reader is referred to [16, Secs. 6.4-6.5], and [88] for an overview.

Let the MPC control law acting on system (C.1) be

\[ u^*(x) \triangleq Kx + \zeta^*(x), \quad (C.11) \]

Note that \( \zeta^*(x) \in \mathbb{R}^m \) represents the difference between the MPC control move and the baseline linear control law \( \kappa(x) \). In the following, we will make use of tightened constraints on the nominal evolution of (C.8) to ensure the fulfillment of the actual constraints for the perturbed system. Given \( x(t) = x \), the nominal evolution of (C.8) is denoted by \( \hat{x}(t+k|t) \), while the actual system with the same initial condition is \( x(t+k|t) \). Both evolutions are obtained by applying the corresponding control sequence denoted by \( \zeta^*(t|t), \ldots, \zeta^*(t+k-1|t) \). It is well known from the set-theoretical analysis in [28] and [57], that, given \( X_k \triangleq X \sim R_k \) and \( U_k \triangleq U \sim K\mathcal{R}_k \), one has that, for all \( k \in \mathbb{Z}_{\geq 0} \),

\[ \hat{x}(t+k|t) \in \mathcal{X}_k \Leftrightarrow x(t+k|t) \in \mathcal{X}, \quad \forall w \in \mathcal{W}, \]

\[ K\hat{x}(t+k|t) \in U_k \Leftrightarrow Kx(t+k|t) \in U, \quad \forall w \in \mathcal{W}. \]

The next step is to find the maximal output admissible robust set (MOARS) for system (C.8), defined as

\[ \mathcal{X}_f \triangleq \{ x(0) \in \mathbb{R}^n : x(k|0) \in \mathcal{X}, \ Kx(k|0) \in \mathcal{U}, \forall k \in \mathbb{Z}_{\geq 0}, \forall w \in \mathcal{W} \}. \quad (C.12) \]

The same set can be conveniently expressed, using tightened sets, as

\[ \mathcal{X}_f = \{ x : A^k_\kappa x \in \mathcal{X}_k, \ K A^k_\kappa x \in \mathcal{U}_k, \forall k \in \mathbb{Z}_{\geq 0} \}. \quad (C.13) \]
C.5. Robustly stabilizing optimal MPC

The set $\mathcal{X}_f$ can be computed by Algorithm 6.1 in [57] using linear programming. In particular, exploiting the results in Theorems 6.2 and 6.3 in [57], $\mathcal{X}_f$ is finitely generated if $0 \in \text{int}(\mathcal{X} \sim \mathcal{R}_{\infty})$ and $0 \in \text{int}(\mathcal{U} \sim K \mathcal{R}_{\infty})$. If $\mathcal{R}_{\infty}$ is not computable, one can use the above-mentioned over-approximation $\tilde{\mathcal{R}}_{\infty}$ instead.

**Assumption 5.** It is supposed that $0 \in \text{int}(\mathcal{X} \sim \mathcal{R}_{\infty})$ and $0 \in \text{int}(\mathcal{U} \sim K \mathcal{R}_{\infty})$ (which ensures the computability of $\mathcal{X}_f$). Moreover, we assume that $\tilde{\mathcal{R}}_{\infty} \subset \text{int}(\mathcal{X}_f)$. $\Box$

**Remark 5.** The condition $\tilde{\mathcal{R}}_{\infty} \subset \text{int}(\mathcal{X}_f)$ represents a slightly stronger requirement with respect to condition $\mathcal{R}_{\infty} \subset \mathcal{X}_f$, which always holds. It is important to note that, if $\tilde{\mathcal{R}}_{\infty} \subset \text{int}(\mathcal{X}_f)$, being $\tilde{\mathcal{R}}_{\infty}$ a closed set, any state trajectory that converges to $\tilde{\mathcal{R}}_{\infty}$ asymptotically, converges to $\mathcal{X}_f$ in finite time. $\Box$

Recalling the sets $S_i$ defined in Section C.4, we introduce the set

$$S_f \triangleq \bigcup S_i : S_i \subseteq \mathcal{X}_f, \ i = 0, \ldots, L - 1 \quad (C.14)$$

which will be useful to formulate the subsequent results.

### C.5.3 MPC with tightened constraints

For the proposed robust MPC strategies, the prediction of the system trajectory on the finite prediction horizon $N \in \mathbb{Z}_{>0}$ will make use of the nominal trajectory of the system, but the fulfillment of the constraints will be required for all realizations of the disturbance $w \in \mathcal{W}$. The vector of optimization variables (inputs) to be determined at time $t$ is $Z \triangleq [\zeta'(t|t) \cdots \zeta'(t|t+N-1)]' \in \mathbb{R}^{mN}$. The definition of the optimal sequence $\zeta^*(x)$ is based on the solution of the following finite-horizon optimal control problem (FHOCP) at each time $t$, with $x(t) = x$:

$$Z^*(x) = \arg \min_Z \sum_{k=0}^{N-1} \|\zeta(k)\|_\Psi^2, \ \Psi = \Psi' > 0 \quad (C.15a)$$

s.t. $\dot{x}(k) \in \mathcal{X}_k, \ k = 0, \ldots, N - 1 \quad (C.15b)$

$K \dot{x}(k) + \zeta(k) \in \mathcal{U}_k, \ k = 0, \ldots, N - 1 \quad (C.15c)$

$\dot{x}(N) \in \mathcal{X}_f \sim \mathcal{R}_N \quad (C.15d)$
For ease of notation, implying that the solution of the FHOCP is computed at time $t$, we set $\zeta(k) \triangleq \zeta(t + k|t)$ and $\hat{x}(k) \triangleq \hat{x}(t + k|t)$. Note that (C.15b) and (C.15c) lead to the fulfillment of (C.2) and (C.5), respectively, along the prediction horizon. Finally, (C.15d) guarantees that $x(t + k|t) \in \mathcal{X}_f$ for all possible disturbance sequences.

The FHOCP (C.15) is quadratic with respect to the decision variable $Z$, and is subject to linear constraints. Also, the current state $x$ can be considered as a parameter. Therefore, (C.15) can be recast as a multi-parametric quadratic program (mpQP), where the set of parameters $x$ for which a feasible solution exists is called $\mathcal{D}_N$. Since $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{W}$ are polytopes, $\mathcal{D}_N$ is also a polytope and can be easily computed using linear programming and projections. Also, an increase of the prediction horizon leads to a larger set $\mathcal{D}_N$, i.e. $\mathcal{D}_N \supseteq \mathcal{D}_{N-1} \supseteq \ldots \supseteq \mathcal{D}_1 \supseteq \mathcal{X}_f$. Note that solving the given robust problem is not computationally more involved than solving the nominal problem obtainable with $\mathcal{W} = 0$, since the same number of decision variables and constraints are obtained. The nominal case (i.e., $\mathcal{W} = 0$) can be seen as a limit case of the robust case, and $\mathcal{D}_N$ is always included in the corresponding set obtained for the nominal case, namely $\mathcal{D}_N'$.

The application of the receding horizon principle leads to defining the MPC control law $\zeta^*(x)$ as $\zeta^*(x) \triangleq [I \ 0 \ldots \ 0]Z^*(x)$. Following the development in [13], explicit expressions for the optimal value of the cost function in (B.2a), namely $J^*(x)$, and for $Z^*(x)$, can be obtained solving an mpQP. In particular, both $J^*(x)$ and $Z^*(x)$ are Lipschitz continuous, and more precisely $J^*(x)$ is piecewise-quadratic, while $Z^*(x)$ is piecewise-affine. This implies that also $\zeta^*(x)$ and $u^*(x)$ are piecewise-affine function defined in $\mathcal{D}_N$. The set $\mathcal{D}_N$ is then partitioned as $\mathcal{D}_N = \bigcup_{i=0}^{L_D-1} D_i$, where $\{D_i\}_{i=0}^{L_D-1}$ are polytopes (not necessarily simplices) with non-overlapping interiors.

**Theorem 9.** Let Assumptions 4-5 hold for system (C.6) with $w \in \mathcal{W}$, and let $u^*(x)$ be defined in (C.11). Assume also that $0 \in \text{int}(\mathcal{S}_f)$ (this latter being defined in (C.14)), $\hat{\mathcal{R}}_\infty \subset \text{int}(\mathcal{S}_f)$, and

$$w(x) = 0, \ \forall \ x \in \mathcal{S}_f.$$  \hfill (C.16)

\footnote{Compute the set of optimal inputs for the whole prediction horizon, apply only the first move and then redo the computation at the next sample time.}
If \( x(0) \in D_N \), the origin is an asymptotically stable equilibrium for system (C.6), with domain of attraction equal to \( D_N \). Moreover, \( x(t) \in \mathcal{X} \) and \( u^*(t) \in \mathcal{U} \) for all \( t \geq 0 \).

**Proof.** We recall that Assumptions (A1)-(A5) in [28] are automatically satisfied if Assumptions 4-5 and (C.9) hold. Therefore, according to Lemma 7 and Theorem 8 in [28], recursive feasibility is ensured if \( x(0) \in D_N \). As a consequence, \( x(t) \in \mathcal{X} \) and \( u^*(t) \in \mathcal{U} \) for all \( t \geq 0 \). Also, \( x(t) \to R_{\infty} \) as \( t \to \infty \), since \( K \) is stabilizing for the nominal system (C.7). On the other hand, according to the expression of \( \mathcal{X}_f \) in (C.13), the evolution of the nominal system given by \( \hat{x}(k) \) with initial condition \( x \in \mathcal{X}_f \) and \( \zeta(t+k|t) = 0 \), \( \forall k = 1, ..., N-1 \), fulfills the constraints (C.15b)-(C.15c). Also, as noticed in [28], the constraints \( \hat{x}(k) \in \mathcal{X}_k \) and \( K\hat{x}(k) \in \mathcal{U}_k \) for \( k \geq N \) are equivalent to the terminal constraint (C.15d). Then, we conclude that \( Z = [0 \cdots 0]' \) is a feasible solution for (C.15) whenever \( x \in \mathcal{X}_f \), and is the minimizer of (C.15), since it is the global minimum of the objective function, i.e., \( x \in \mathcal{X}_f \implies Z^*(x) = [0 \cdots 0]' \).

Since \( R_{\infty} \subset \text{int}(S_f) \), then there exists \( \epsilon \in \mathbb{R}_{>0} \) arbitrary small, such that \( (1 + \epsilon)R_{\infty} \subset \text{int}(S_f) \). Considering that \( R_{\infty} \) is a RPI set for system (C.8), it is a RPI set for system (C.7) as well. Therefore, by linearity of the system, \( (1 + \epsilon)R_{\infty} \) is also a RPI set for (C.7). Considering now the actual dynamics (C.6), from the trivial relation \( R_{\infty} \subset (1 + \epsilon)R_{\infty} \) it follows that (for all initial conditions \( x(0) \in D_N \)) there exists \( t_1 \in \mathbb{Z}_{\geq 0} \) such that \( x(t_1) \in (1 + \epsilon)R_{\infty} \). Since it is assumed that \( w(x) = 0 \) for all \( x \in S_f \), and \( (1 + \epsilon)R_{\infty} \) is positively invariant for the system (C.7), one has that the system dynamics is equal to (C.7) for all \( t \geq t_1 \), which leads to the asymptotic convergence of the state of system (C.6) to the origin for all \( x(0) \in D_N \). 

### C.6 Approximation procedure

First of all, we assume that a control law \( u^*(x) \) has been computed for system (C.1), therefore obtaining the domain of attraction \( D_N \). We define \( S \) as described in Section C.4 as the smallest hyper-rectangle such that \( D_N \subseteq S \). Then, we partition the (not necessarily convex) set \( S \setminus D_N \) as \( S \setminus D_N = \bigcup_{i=0}^{L-1} D_i \), where \( \{D_i\}_{i=0}^{L-1} \) are polytopes with non-overlapping interiors. In
this way, a generic partition of $S$ as $S = \bigcup_{i=0}^{L_D-1} D_i$ is obtained, where $\hat{L}_D \triangleq L_D + \hat{L}_D$, while we denote its set of vertices as $\hat{V}_D$. In order to introduce the approximation procedure, we use the concept of mixed partition (see, e.g., [14]), as the partition of $S$ induced by the facets of both simplicial ($S_i$) and generic ($D_i$) partitions. As a result, $S$ is further partitioned into convex polytopes, and the partition is completely defined by the sets of vertices $V_S$, $\hat{V}_D$ and $V_M$, the latter representing the set of vertices given by the intersection of the two partitions and belonging neither to $V_S$ nor to $\hat{V}_D$. Finally, let

$$V_I \triangleq \left\{ v \in (V_S \cup \hat{V}_D \cup V_M) : v \in D_N \right\}$$

and note that $D_N$ is the convex hull of all $v \in V_I$.

In this work, we choose to find $u(x)$ that minimizes the maximum discrepancy with respect to $u^*(x)$ for all $x \in D_N$ (note that $u^*(x)$ is not defined on $S \setminus D_N$), that is

$$F_\infty \triangleq \max_{j=1,...,m} \sup_{x \in D_N} \{ |u_j(x) - u^*_j(x)| \}$$  \hspace{1cm} (C.17)

When minimizing $F_\infty$ in (C.17), some constraints have to be imposed for all $x \in D_N$. Since the minima and maxima of the PWA function $w(x) = u(x) - u^*(x)$ on any of the regions of the mixed partition are on vertices, it is sufficient to impose constraints only on the vertices of $V_I$. In particular:

1. The control law $u(x)$ must satisfy the constraint (C.2), which is already satisfied by $u^*(x)$. This can be done imposing $C_u v \leq g_u$ for all $v \in V_I$, which implies $C_u u(x) \leq g_u$ for all $x \in D_N$.

2. The value of $u(x)$ must be computed such that $\|u(x) - u^*(x)\|_\infty \leq \bar{w}$, in order for system (C.1) to satisfy (C.9). This can be obtained simply imposing $\|u(v) - u^*(v)\|_\infty \leq \bar{w}$ for all $v \in V_I$;

3. In order to obtain (C.16), we impose that $u(v) = u^*(v)$ for all $v \in V_I \cap D_f$.

Therefore, after recalling the relationship between the vector $\theta$ and the control variable $u(x)$ in (C.2)-(C.3), our proposal is to obtain $u(x)$ by solving
C.6. Approximation procedure

the following linear program:

\[
\begin{align*}
\min_{\theta, \eta} & \quad \eta \\
\text{s.t.} & \quad \eta \geq \pm \left( \phi(v)^T \theta_j - u_j^*(v) \right), \ v \in V_I, \ j = 1, \ldots, m \\
& \quad C_u \Phi(v) \theta \leq g_u, \ v \in V_I \\
& \quad \Phi(v) \theta = u^*(v), \ v \in V_I \cap S_f \\
& \quad \eta \leq \bar{w}
\end{align*}
\]

(C.18a) (C.18b) (C.18c) (C.18d) (C.18e)

The formulation of the cost function (C.18a) together with the constraint (C.18b) leads to finding the vector \( \theta \) that minimizes the maximum difference between \( u_j(x) \) and \( u_j^*(x) \) for all \( x \in D_N \) and all components \( j \). Conditions (C.18c) and (C.18d) lead to the fulfillment of (C.2) and (C.16), respectively. Condition (C.18e) ensures the fulfillment of (C.9). Once a feasible solution to (C.18) has been found, vector \( \theta \) determines the control law \( u(x) \) for all \( x \in S \).

C.6.1 Properties of the PWAS control law

The following result holds when the approximate control law \( u(x) \) is applied to system (C.1).

**Theorem 10.** Assume that system (C.6) fulfills all the assumptions required in Theorem 9, and that a feasible solution for the FHOCP (C.15) has been determined as \( u^*(x) \), together with the set \( D_N \). Also, assume that there exists a realization of \( u(x) \) obtained through a feasible solution of (C.18). Applying the obtained control law \( u(x) \) to system (C.1), if \( x(0) \in D_N \), one has \( x(t) \in X \) and \( u(t) \in U \) for all \( t \geq 0 \). Moreover, the origin of system (C.1) is an asymptotically stable equilibrium point, with domain of attraction equal to \( D_N \).

**Proof.** Conditions (C.18d)-(C.18e) allow to consider \( w(x) = u(x) - u^*(x) \) as a disturbance term that satisfies all the requirements to synthesize \( u^*(x) \) in (C.11). Therefore, by application of Theorem 9, all the mentioned properties are proved.

Considering that the feasibility of both (C.15) and (C.18) is not guaranteed, we give some guidelines on choosing the design parameters. We assume
that the number of vertices \( N_v \) of the simplicial partition is fixed a priori, taking into account the available hardware device. Given the sets \( X \) and \( U \), the tuning parameter on which the designer can act to design \( u^\ast(x) \) is \( \bar{w} \). It is easy to see that, if (C.15) is feasible for a given \( \bar{w} = \bar{w}_1 \), then the same problem will be feasible for all \( \bar{w} \leq \bar{w}_1 \). Then, one can find by bisection the maximum feasible value of \( \bar{w} \), namely \( \bar{w}_{\text{max}} \), and then (C.15) will be feasible for all \( \bar{w} \) such that \( 0 \leq \bar{w} \leq \bar{w}_{\text{max}} \). We also know that a smaller value of \( \bar{w} \) leads to a larger set \( D_N \). On the other hand, a small value of \( \bar{w} \) could impose a too tight approximation in problem (C.18), making it infeasible.

The designer can start obtaining a feasible realization of the PWAS control for a value of \( \bar{w} \) close to \( \bar{w}_{\text{max}} \). Then, this value can be decreased in order to enlarge the set \( D_N \). If a feasible solution cannot be obtained, then one can increase the number of vertices of the simplicial partition and restart the procedure.

### C.7 Simulation example

Consider the problem of regulating to the origin the unstable plant defined by

\[
A = \begin{bmatrix} 1.1 & -1.4 \\ 0.9 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}
\]

subject to the constraints

\[
X = \{ x \in \mathbb{R}^2 : \| x \|_\infty \leq 1 \}
\]

\[
U = \{ u \in \mathbb{R} : | u | \leq 1 \}.
\]

Assumption 4 is satisfied. We impose \( \bar{w} = 7 \cdot 10^{-2} \). A fixed gain for the auxiliary control law is defined as \( K = \begin{bmatrix} -1.4299 & -0.3588 \end{bmatrix} \), as the infinite-horizon linear quadratic regulator using the weight matrices

\[
Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 0.1,
\]

The auxiliary control law leads to the satisfaction of Assumption 5: the sets \( \hat{R}_\infty, S_f \), and \( X_f \) are shown in Fig. C.1.
C.7. Simulation example

The MPC control law $u^*(x)$ in (C.11) is computed with $\Psi = 1$ and $N = 4$, and its domain of attraction $D_N$ is also shown in Fig. C.1. The approximate control law $u(x)$ is computed with $p_1 = p_2 = 13$ (defined in Section C.4.1), obtaining $N_v = 196$ vertices and $L_s = 338$ simplices, with a maximum approximation error $\eta = 6.75 \cdot 10^{-2}$. The PWAS control law so obtained is shown in Fig. C.2. For comparison purposes, we obtained the optimal MPC control law described in (C.11) in case $W = \{0\}$ (i.e., no approximation error was considered), as previously mentioned, maximizes the domain of attraction. This latter, namely $D'_N$, is also shown in Fig. C.1. Using the recently developed MOBY-DIC toolbox ([79] and Appendix B), the VHDL code defining the PWAS control law has been generated. The latency on a Xilinx Spartan 3 FPGA (xc3s200) board (using the architecture B in [104] and encoding the state variables (circuit inputs) with 12 bits words) has been
C. PWAS stable approximation of explicit MPC

Figure C.2: Control function $u(x)$ on the simplicial partition of the set $S$.

estimated to be approximately 65 ns. Finally, in Fig. C.3, the time evolution of the state and control variables are shown starting from the initial condition $x(0) = \begin{bmatrix} 0.88 & 0.2 \end{bmatrix}$. Note that the approximation does not significantly affect the state and control trajectories, and keeps constraints satisfied.

C.8 Conclusions

In this Appendix an approximate MPC control law for LTI systems based on PWAS functions was proposed, that can guarantee both $a$-priori stability for the closed-loop system and efficient implementation on digital hardware. The applicability of the proposed control strategy is limited to the case of small-sized systems, similarly to standard explicit MPC. The theoretical properties of the control law have been proved based on robust MPC synthesis, and the simulation results on a second-order unstable plant have confirmed the expected results, both for the theoretical properties of the PWAS controller
Figure C.3: Time evolution of state and control variables in the simulation example. The solid lines represent the variables \((x_1(t), x_2(t), u(t))\) that would result by applying the optimal control law \(u^*(t)\), while the dashed lines represent the same variables obtained by applying the PWAS control law represented in Fig. C.2.

and for the performance of the related FPGA implementation.
Bibliography


Index

AGBAR, Aguas de Barcelona, S.A., 93
Approximation procedure, 203
Centralized HiMPC, 72
Consensus, 15
CSP, Constraint satisfaction problem, 109
Decentralization matrices, 48
Decentralization scheme, 48
Decentralized HiMPC, 79
decentralized MPC, 5
Decentralized optimal control problems, 50
Decentralized prediction models, 48
dclmi, decentralized LMI, 124
distributed MPC, 5
dlinc, decentralized lincon, 139
DMPC convergence properties, 52
DWN, Drinking water network, 92
Explicit MPC PWAS Approximation with stability certificate, 191
FHOCP, finite-horizon optimal control problem, 201
HiMPC, Hierarchical MPC, 70
ISS, Input-to-state stability, 194
LCL, Lower Control Layer, 73
Linear controller synthesis, 26
Linear switching controller synthesis, 30
LMI, Linear Matrix Inequalities, 24
Markov chain, 35
Maximum reference rates, 75
MILP, Mixed Integer Linear Program, 76
MOARS, Maximum output admissible robustly invariant set, 82
MOAS, Maximum Admissible Output Set, 75
MOBYDIC Toolbox for Matlab, 170
MPC, Model Predictive Control, 6
mpQP, multi-parametric quadratic program, 202
Nash equilibria, 16
NCS, Networked Control Systems, 23
Network topology, 25
Pareto equilibria, 16
PID, proportional integral derivative controller, 94
Point location problem, 176
PWA, Piece-wise Affine, 173
PWAG, generic PWA, 176
PWAR, hyper-rectangular PWA, 178
PWAS, simplicial PWA, 177

Receding horizon, 202
Reference governor, 71

Schur matrix, 77
SDP, Semi-Definite Programming, 28
Set-point tracking, 61
Stability in mean-square sense, 34
Stochastic network model, 35
Stochastic switching controller synthesis, 36

Temperature control in a railcar, 62

UCL, Upper Control Layer, 73

VHDL, VHSIC hardware description language, 173

WIDE Toolbox, 120
WTP, Water treatment plants, 93