POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale e dell'Informazione

Corso di Laurea Magistrale in Ingegneria dell'Automazione

Solutions for Model Predictive Control of Large-Scale Cascade Systems with Application to Irrigation Networks

Relatore: Prof. Marcello Farina

Correlatore: Prof. Michael Cantoni

Tesi di Laurea di:

Marco Ettore Fasani

Matr. n. 800444

Gabriele Maronati

Matr. n. 796575

Anno Accademico 2014-2015

Acknowledgments

This Thesis was developed in collaboration with the University of Melbourne. Special thanks to Prof. Marcello Farina and Prof. Michael Cantoni whose expertise and kindness made our work possible.
Finally, we would like to thank Prof. Eric C Kerrigan, Farhad Farokhi and Amir R Neshastehriz for their invaluable advice given during our stay in Melbourne.

Contents

List of Figures

Abstract

Nowadays, the modelization and control of large-scale systems of increasing size and complexity is one of the most challenging objectives in the Automation field. The aim of this Thesis is to provide and develop efficient algorithms in the context of a large-scale cascade systems for the application of Model Predictive Control (MPC) techniques. In particular, such algorithms will be applied to an irrigation network.

Two different approaches will be investigated and compared in this Thesis. First, a distributed cooperative MPC strategy will be studied where each controller exchanges information with the others, and where the computational effort required to local computing units is limited.

Secondly, two different centralized MPC formulations will be analyzed stemming from the system block triangular structure. Two solvers will be developed and implemented through the application of numerical methods. This will allow for an efficient solution to the optimization problem arising from the application of the MPC control strategy.

Eventually, simulation tests have been carried out to test and compare the two approaches.

Sommario

Oggigiorno, la modellizzazione ed il controllo dei sistemi a larga scala, con crescente complessità e dimensioni è una tematica di particolare interesse nel campo dell'Automazione. In questa Tesi ci siamo preposti di sviluppare algoritmi efficienti nel contesto del Model Predictive Control (MPC) applicato a sistemi su larga scala con struttura a cascata.

La scelta di studiare algoritmi di controllo per sistemi a larga scala con struttura a cascata è motivata, ad esempio, dal problema di controllare in modo efficace ed affidabile i canali d'irrigazione. Il caso da studiare scelto in questa Tesi è pertanto relativo a questo problema di controllo. Più nello specifico,

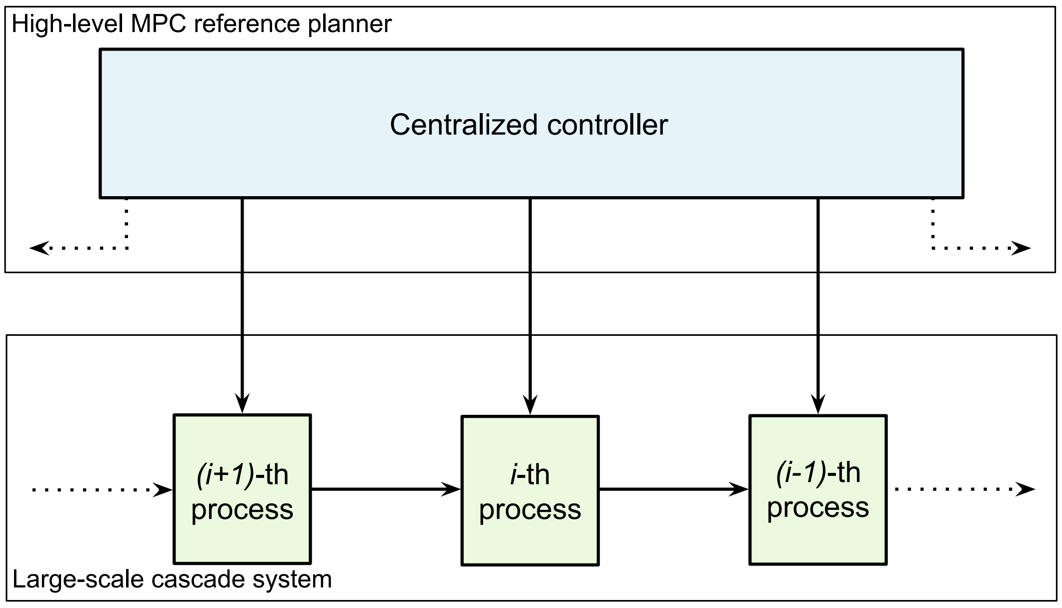
Figure 1: Canale d'irrigazione

l'esempio che sarà usato è un esempio realistico, relativo al distretto idrico di Goulburn Murray situato nello stato meridionale del Victoria, in Australia (vedi Figura 1). Questo distretto copre approssimativamente una superficie di 68000 kilometri quadrati servendo oltre 15000 aziende agricole. Le operazioni manuali sui canali di irrigazione portano ad uno spreco di acqua durante il suo trasporto lungo la rete del 30-40 %. L'automazione di tali infrastrutture porta ad una gestione più consapevole delle risorse idriche con un efficienza che può arrivare fino all' 85%. Il controllo e la gestione di tali vaste strutture è quindi giustificato, tuttavia la sua realizzazione risulta complicata.

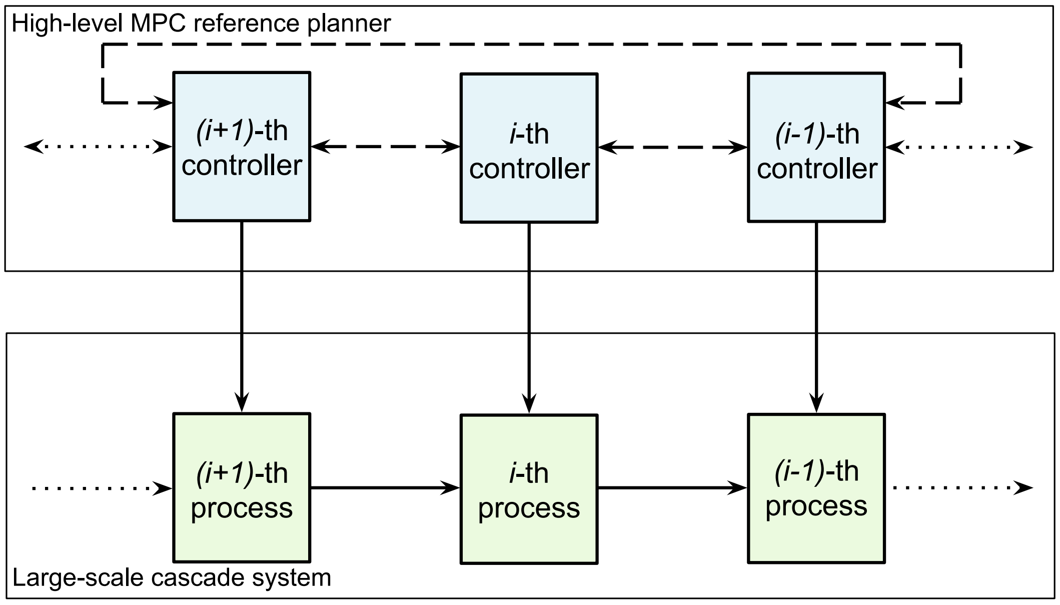
In questa Tesi verranno studiati due approcci per la soluzione del problema di controllo di sistemi a larga scala basato su MPC (vedi Figura 2). In primo luogo verrà considerato un approccio cooperativo distribuito MPC, dove tutti i controllori sono abilitati allo scambio di informazioni e alla soluzione di problemi locali di controllo. Tale tecnica viene studiata in quanto risulta essere particolarmente favorevole nella gestione dello sforzo computazionale.

Inoltre, verrà affrontato il problema della riduzione del carico computazionale legato alla soluzione del problema MPC centralizzato. In particolare verranno formalizzate due diverse modellizzazioni del sistema che mostreranno una struttura triangolare a blocchi. Alla luce di ciò, verranno sviluppati e implementati due risolutori, che fanno uso di metodi numerici, in grado di risolvere in maniera efficiente il problema di ottimizzazione derivante dal problema di controllo MPC.

Infine verranno mostrate delle simulazioni riguardanti l'applicazione degli algoritmi citati. Focalizzandoci sull'approccio distribuito, i risultati mostrano migliori prestazioni relativamente al tempo impiegato per calcolare la soluzione ottima. Ciò è dovuto all'intrinseca capacità di tale strategia nel distribuire l'onere computazionale sui diversi sottosistemi. Tuttavia si incontreranno delle limitazioni operative dovuto al fatto che tale tecnica di controllo non permette l'implementazione di vincoli stringenti sulle variabili di stato. Successivamente verranno analizzati i risultati ottenuti dai controllori centralizzati, evidenziando come le loro prestazioni dipendano fortemente dalla scelta dell'orizzonte di predizione e dal numero di sottosistemi che compongono l'intero canale.



(a) Schema centralizzato



(b) Schema distribuito

Figure 2: Architetture di controllo

Chapter 1

Introduction

The aim of this chapter is to provide an overview of the problems addressed in this Thesis. First, large-scale systems are introduced with particular focus on irrigation networks. Afterwards, advanced control strategies to control such systems are discussed, highlighting our original contributions. Eventually, the structure of the Thesis is described.

1.1 Motivations and Context

The research in the Automation and Control field has focused, classically, on the development of design and analysis methods for control and estimation of dynamical systems, with special focus on centralized procedures. More specifically, the focus has been mostly on the overall model of the system to be controlled, regardless of its size and of its internal structure. Indeed, centralized estimation and control methods suffer from the curse of dimensionality. More specifically, as the system size grows, design procedures and online implementation requirements can prevent many control and estimation algorithms from application.

These limitations have pushed research on the so-called large-scale systems (LSS). Typical examples of large-scale systems include large infrastructures, e.g. transportation, water, and power networks. They can be typically regarded as networks of interconnected systems, which exchange data, material

Figure 1.1: Irrigation Channel: focus on a gate

or energy. In view of this, they are characterized by peculiar structures, which must be specifically accounted for in an efficient way. In particular, we will focus on large-scale systems with a cascade structure, e.g. irrigation channel.

Irrigation Networks

Irrigation channel networks are large infrastructures that transfer water from sources (e.g. lakes or reservoirs) to farms under the power of gravity for agricultural activities. In particular, the Golburn-Murray irrigation district is considered in this Thesis (see Figure 1.1). It is located in Australia and it consists of 7000 Kilometers of channels, around 17000 regulators and 21000 farm outlets.

Manual operation currently used to handle the water requests of the farmers typically leads to $30 - 40\%$ waste of water along all the irrigation channels. The research on sensors, actuators and advanced control strategies has the aim to improve the efficiency of the water distribution. When fully implemented, it has been estimated that an automation scheme can improve water distribution efficiency up to 85%, [1].

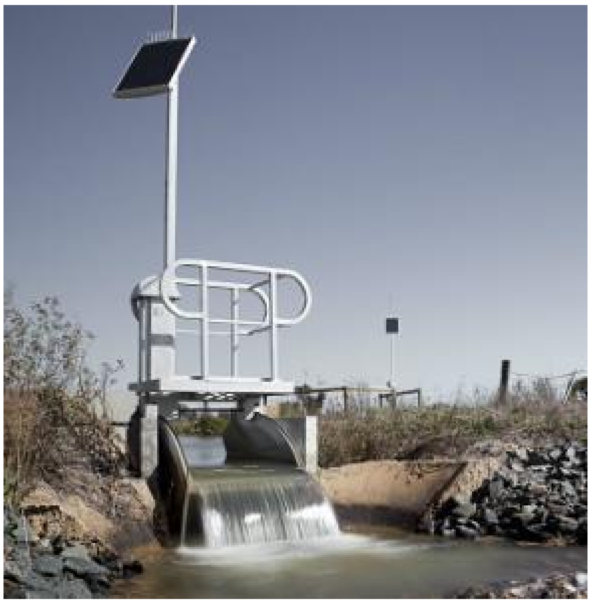


Figure 1.2: FlumeGate^{TM}

In order to regulate the water-flow from the reservoir to the end of the channel, gates are installed along the network. In Figure 1.2, a FlumeGate^{TM} is shown: basically it is a weir aiming to automatically control the flow of water by varying its position based on a desired set-point (e.g., water-level, flow-rate). The stretch of water between two consecutive gates is called pool. The open-water channel may be regarded as a series of pools interconnected by gates. Pools or channels can vary in length, from several kilometers down to hundreds of meters. Moreover, off-take points to the farms and secondary channels are placed along the pools, as shown in Figure 1.3.

At each gate, measurement and control are locally performed and data are transmitted to the operation center by means of radio telecommunications. Typically, the regulation of the water-level is performed by acting on the water-flow through the opening of the gate, with a feedback controller. Due to the slow dynamics of the water-flow and to the wide dimensions of the pools, such feedback control introduces delays in the channel dynamics, limiting the achievable dynamic performances. Moreover, the violation of operational limitations can occur as a result of the implementation of the local

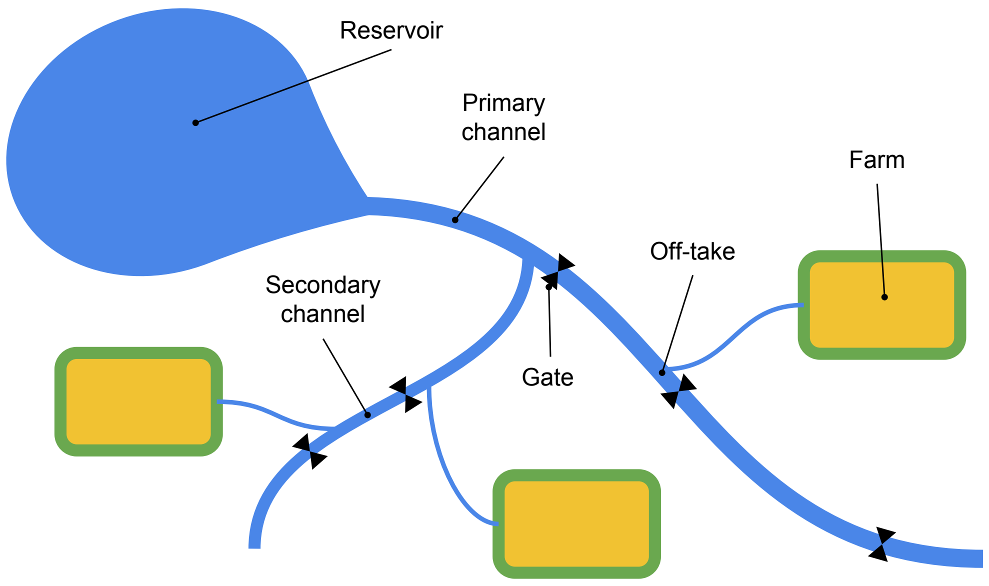


Figure 1.3: Irrigation network

feedback controllers. This can lead to actuator saturation, flooding, or water-levels falling below a low limit (i.e., provision of insufficient flow).

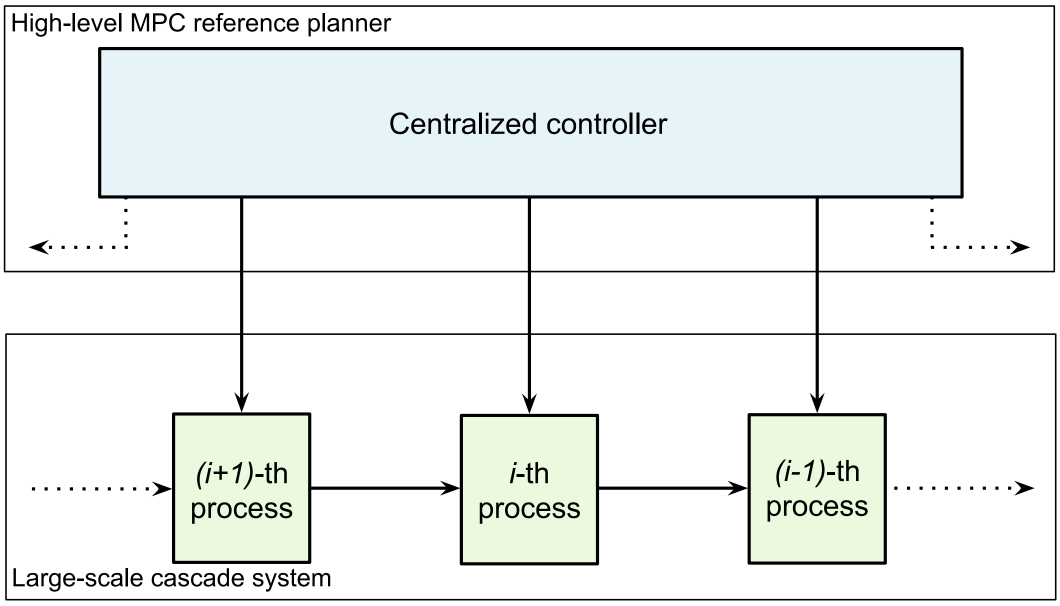
In order to limit the wastage of water and to guarantee the necessary supply for all the farmers' demand, a high level controller working on the adjustment of the water-level references for each pool is usually implemented. For this reason, in this Thesis, the inputs to each pool correspond to water-level references, as more thoroughly discussed in Chapter 3.

1.2 Control Architectures

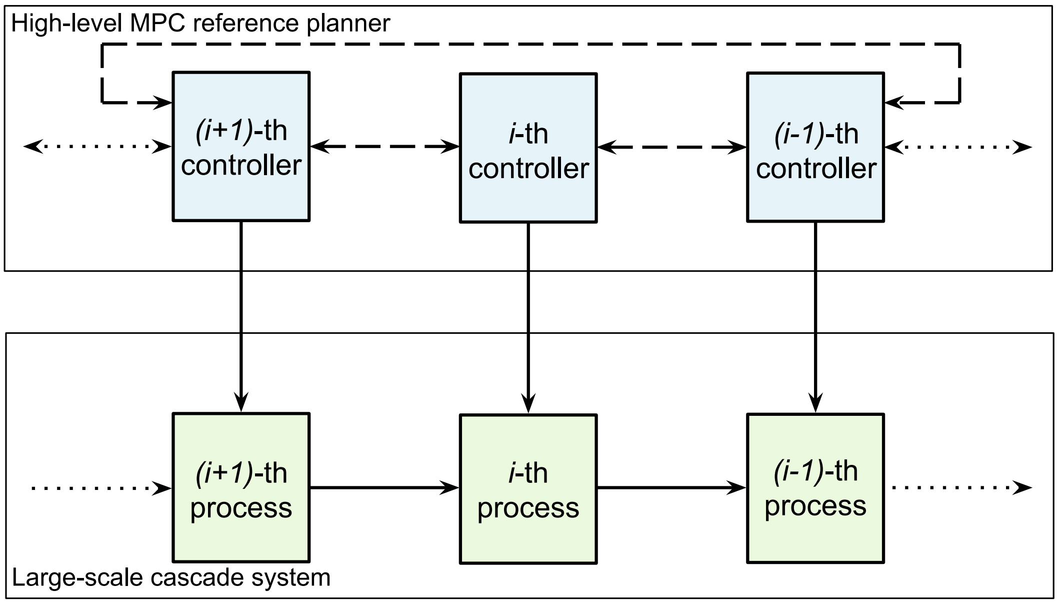
In this section we will introduce the main features of the advanced control strategies investigated in the Thesis. In particular, the computational issues arising in the implementation of a Model Predictive Control scheme to a large-scale cascade system will be highlighted and, distributed and centralized implementations will be discussed.

Centralized MPC

The main idea of a centralized control strategy is to design a regulator which can account for the plantwide interactions, performing a control action for the whole system by accounting for the complete model (see Figure 1.4a). Regarding the irrigation networks case of study, this lead to a unique reference planner for all the pools. However, due to the big amount of data to manage by the unique controller, one may address the control problem with either different solutions (e.g., decentralized, distributed) or using efficient numerical solvers. The latter will be considered in order to develop centralized algorithms. Indeed, increased computational power, faster optimization software, and algorithms designed specifically for large-scale systems, can make centralized control more practical.



(a) Centralized scheme



(b) Distributed scheme

Figure 1.4: Control architectures

Distributed cooperative MPC

A distributed approach is considered in order to divide the system-wide control problem into simpler problems with reduced computational complexity. The network interactions are explicitly modelled, and information is exchanged among local controllers (see Figure 1.4b). In particular, a cooperative algorithm will be considered. Basically it requires each subsystem to take into account the effect of local control actions on other subsystems in the network, whereas a global objective function has to be minimized by each controller.

1.3 Thesis Contributions

The problem of reducing the computational effort required by the application of MPC schemes to large-scale systems (with special focus on irrigation networks) has been already addressed in the past, showing that many improvements can be made by taking advantage of the sparse structure of the systems considered (i.e., large-scale cascade systems). The main source of inspiration for our work is [2]. In this paper a peculiar method of solving quadratic programming problems is presented. Using [2] as a starting point, in this Thesis we have modified the original optimization problem in order to improve the computational efficiency of the solution. After, we focused on an alternative and original formulation which yield to a different solution to the same optimization problem, relying on a different problem formulation. As it will be thoroughly discussed, this contribution lend itself to the application for a system very large in scale, where a classic centralized approach would be impractical.

1.4 Structure of the Thesis

This Thesis is structured as follows:

- In Chapter 2 a modelization of each subsystem composing the LSS is carried out, with a particular focus on systems with a cascade structure. Furthermore, the Model Predictive Control technique is introduced and the properties of stability and recursive feasibility are analyzed. After, the optimization problem is formulated according to two different model formulations.

- In Chapter 3 the irrigation channel case of study is introduced. More specifically, a continuous-time state-space representation of a pool controlled with its local decentralized regulator is derived and then discretized. Also, its inclusion in a system-wide control scheme is briefly discussed.

- In Chapter 4 the distributed MPC control strategy is introduced. A cooperative algorithm from the literature is studied. Algorithmic details are discussed.

- In Chapter 5 numerical methods for the solution to a centralized control algorithm are introduced. First, it is shown how an MPC optimization problem can be translated into a sequence of quadratic programs. Then, the Interior Point Methods are introduced, leading to a particular procedure which will be employed in the sequel. A brief computational analysis of such methods is eventually carried out.

- In Chapter 6 two solvers which take advantage of the particular system structure are presented and illustrated in detail. The goal of both solvers is to simplify the computational effort. Therefore, computational aspects are regarded and the two algorithms are compared together.

- In Chapter 7 simulation tests on a realistic benchmark of the algorithms presented in Chapter 4 and in Chapter 6 are shown, in order to compare computational performances. Eventually, closed loop simulations are illustrated.

- In Chapter 8 conclusions are drawn , and possible future developments are envisaged.

Chapter 2

Large-scale and structured systems. MPC control

The main difficulty that occur when dealing with large-scale system is related to the great demand of computational resources. However, the large-scale systems considered in this Thesis have a cascade structure which, regarding the application of a MPC control strategy, may be taken into account for in an efficient way. In this respect, two different model formulations can be considered, leading to two different representations of the same control problem.

2.1 Systems and Modelling

Let us consider a set of N interconnected subsystems, each characterized by the internal state and input vectors $x_i(k) \in \mathbb{R}^{n_i}$ and $u_i(k) \in \mathbb{R}^{m_i}$, respectively, with $i \in \mathcal{V} = \{1, \dots, N\}$. Focusing on linear time-invariant discrete-time systems, for each $i \in \mathcal{V}$, the dynamics of x_i is described as follows

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k) + F_i v_i(k) \tag{2.1}$$

where $A_i \in \mathbb{R}^{n_i \times n_i}$ and $B_i \in \mathbb{R}^{n_i \times m_i}$ represent the effect of local variables on the evolution of the internal state. The term $v_i(k) \in \mathbb{R}^{s_i}$ has been introduced to account for the influence of the variables of other subsystems on $x_i(k)$. More specifically, for each $i \in \mathcal{V}$, we define a set $\mathcal{W}_i$ as follows:

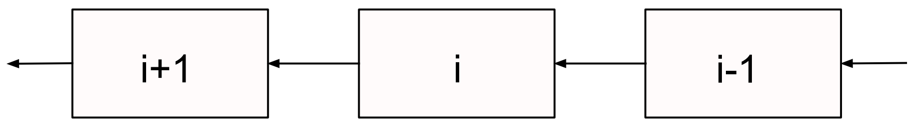
$$\mathcal{W}_i = \{j \neq i | x_j(k) \textit{ or } u_j(k) \textit{ have a direct influence on the evolution of } x_i(k+1)\}$$

In particular we can define the coupling variable $v_i(k)$ as a linear combination of internal variables of neighbour subsystems, i.e., as discussed in [3],

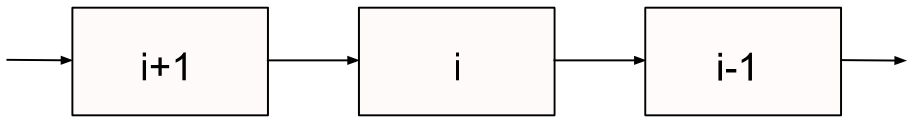
$$v_i(k) = \sum_{j \in \mathcal{W}_i} K_{ij} u_j(k) + \sum_{j \in \mathcal{W}_i} H_{ij} x_j(k) \tag{2.2}$$

where the interconnection matrices K_{ij} and H_{ij} have appropriate dimensions. In this work we focus on cascade systems, in view of their wide-spread diffusion in the context of large-scale irrigation networks. The class of cascade systems is a subclass of system modelled using (2.1)-(2.2). More specifically, in this case, it is set $\mathcal{W}_i = \{i+1\}$ for all $i = 1, \dots, N-1$ while $\mathcal{W}_N = \varnothing$ or equivalently, $\mathcal{W}_i = \{i-1\}$ for all $i = 2, \dots, N$ while $\mathcal{W}_1 = \varnothing$. Clearly this means that the dynamics of the state of the i-th subsystem is directly affected just by the internal state of either the upstream (Figure 2.1a) or the downstream one (Figure 2.1b). Note that the terms *upstream* and *downstream* have a specific reference to the definition of the set $\mathcal{V}$. This, possibly, implies a preliminary permutation of the subsystems order, highlighting the peculiar cascaded structure.

Another system assumption adopted in this Thesis is $K_{ij} = 0$ for all $i, j \in \mathcal{V}$ with $i \neq j$ meaning that interconnections are through states variables (i.e.,



(a) Coupling with the downstream subsystem: $\mathcal{W}_i = \{i-1\}$



(b) Coupling with the upstream subsystem: $\mathcal{W}_i = \{i+1\}$

Figure 2.1: Coupling in a cascade system

systems are state-coupled but input-decoupled). Remark that this may be regarded as a strong assumption. However it applies to several case of studies (among which the irrigation channel described in Chapter 3). Also, with a suitable reformulation of the state variables, all types of couplings can be given this form.

In view of the discussion above, the system form used in the sequel is

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k) + E_i x_j(k) \tag{2.3}$$

where $E_i = F_i H_{ij}$ and, up to a suitable subsystem permutation, we have the following two equivalent cases:

- $j = i-1\,,\ E_i \neq 0 \quad \forall\, i = 2, \ldots, N$ while $E_1 = 0$
- $j = i+1\,,\ E_i \neq 0 \quad \forall\, i = 1, \ldots, N-1$ while $E_N = 0$

2.2 Model Predictive Control

Model Predictive Control (MPC) is an advanced control strategy that has been widely adopted in the industrial process control community and implemented successfully in many industrial applications since 1980s.

The control problem is formulated as an optimization one, where constraints on input and state variables can be accounted for explicitly.

More specifically, the aim of the MPC technique is to find the sequence of optimal control inputs by explicitly minimizing, at each time instant, a cost function calculated through the prediction of the behaviour of the system. Such prediction is attained through the model of the system and the measurement of the current state. The optimal control law is implemented until another state measurement is available, then the updated variables of the system are used to formulate the new optimization problem for the next iteration step. This characterizes the so-called receding horizon approach.

The application of the MPC control strategy to a large-scale system (LSS) requires a demanding computational effort, limiting the use of these kinds of controllers to processes with relatively slow dynamics or small-scale models. Moreover, the presence of constraints on inputs and states further complicate the optimization problem. This pushes research on efficient numerical methods and on distributed/hierarchical architectures.

Model Predictive Control for unstructured linear systems

Let us consider a discrete-time linear time-invariant system (LTI)

$$x(k+1) = Ax(k) + Bu(k) \tag{2.4}$$

where the state vector $x \in \mathbb{R}^n$ is measurable and $u \in \mathbb{R}^m$ is the control vector.

We also consider a set of linear inequality constraints on input and state variables in order to represent physical limitations on a real system such as actuator's saturations. In particular they can be expressed by:

$$x(k) \in \mathbb{X}, \quad u(k) \in \mathbb{U} \tag{2.5}$$

where $\mathbb{X}$ and $\mathbb{U}$ are polytopic compact convex sets that include the origin in their interior.

The main rationale of MPC is the following. At time instant t the MPC controller determines the optimal control sequence $u^o(t), u^o(t+1), \dots, u^o(t+T-1)$ along a fixed horizon of lenght T, minimizing a suitable cost function.

Then, only the first input $u^o(t)$ is implemented on the system at time t and all the other elements of the optimal control vector are discarded. After the subsequent time step $t+1$, the same procedure is carried out considering the shifted time horizon $t+1, \cdots, t+T$ and using the new measurement $x(t+1)$ of the internal state of the system. Then, only the first element of the new optimal inputs vector sequence, i.e. $u^o(t+1)$, will be applied. This technique defines the so-called *Receding Horizon* principle and leads to an implicit time invariant control law that, as shown in [4] for example, can be expressed by

$$u = \kappa_{RH}(x) \tag{2.6}$$

Typically the cost function is a quadratic function of type

$$J\big(\{x(k)\}_{k=0}^{T}, \{u(k)\}_{k=0}^{T-1}\big) = \sum_{k=0}^{T-1} \big(||x(k)||_Q^2 + ||u(k)||_R^2\big) + V_f\big(x(T)\big) \tag{2.7}$$

where the weight matrices $Q = Q^T \succeq 0$, $R = R^T \succ 0$, have appropriate dimensions and the term $V_f\big(x(k+T)\big)$ is the positive definite terminal weight. $x(0)$ is considered to be equal to the state measurement taken at the current time instant t.

The resulting optimization problem consists in the minimization of the objective function (2.7) subject to a set of equality constraints expressed by (2.4) and inequality constraints (2.5) for $k = 0, \cdots, T-1$. More specifically, the equality constraints represent the dynamic of the linear system predicted along the horizon of length T and the inequality constraints may be introduced in order to consider limitations acting on the dynamic system.

Recursive feasibility

Before proving stability, one needs to establish recursive feasibility for the MPC problem. More specifically, for all feasible initial state conditions, feasibility must be guaranteed at every future time step. In other words, let assume feasibility for the optimization problem (i.e., existence of a solution) at a generic time instant t, recursive feasibility guarantees the existence of a feasible solution to the optimization problem at time step $t+1$. This may be achieved by enforcing, in the optimization problem defined in the previous section, a suitable terminal set constraint of type

$$x(T) \in \mathbb{X}_f.$$

To properly define the set $\mathbb{X}_f$ we need the following definition.

Definition 2.1. A set Υ is said to be positively invariant for the autonomous system $x(k+1) = f\big(x(k)\big)$ if the following condition holds:

$$x(\overline{k}) \in \Upsilon \Rightarrow x(k) \in \Upsilon,\ \forall k \geq \overline{k}$$

■

Let us define the auxiliary control law

$$u(k) = \kappa_{aux}\big(x(k)\big)$$

such that the equilibrium $x = 0$ is asymptotically stable for the system

$$x(k+1) = Ax(k) + B\kappa_{aux}\big(x(k)\big), \tag{2.8}$$

the set $\mathbb{X}_f \subseteq \mathbb{X}$ a positive invariant set containing the origin, such that, considering the closed loop system (2.8) for all the states x belonging to $\mathbb{X}_f$ at time $\overline{k}$ results

$$x(k) \in \mathbb{X}_f,\ k \geq \overline{k} \tag{2.9a}$$

$$u(k) = \kappa_{aux}\big(x(k)\big) \in \mathbb{U},\ k \geq \overline{k} \tag{2.9b}$$

This means that starting from an initial state $x(\overline{k}) \in \mathbb{X}_f$ and applying the auxiliary control law, the states remain into $\mathbb{X}_f$ and the constraints (2.5) on states and inputs are satisfied.

Stability

Let us introduce the conditions needed to obtain a control law, through MPC and the RH principle, which guarantees stability for the closed loop system for linear discrete time systems in the form (2.4) subject to constraints (2.5).

The terminal weight must chosen so that $\forall x(k) \in \mathbb{X}_f$

$$V_f\Big(Ax(k) + B\kappa_{aux}\big(x(k)\big)\Big) - V_f\big(x(k)\big) + \Big(||x(k)||_Q^2 + ||\kappa_{aux}\big(x(k)\big)||_R^2\Big) \leq 0, \tag{2.10}$$

If (2.10) is verified the stability of the closed loop system is ensured with a control law expressed by (2.6). In particular, looking at (2.10), we note that, to ensure the stability, the terminal weight has to be a *Lyapunov* function for the controlled system.

Typical choices of κ_{aux} and V_f, in the linear framework, are:

- $\kappa_{aux}(x) = Kx$, where K is selected to guarantee that $A + BK$ is Schur stable, i.e. that the eigenvalues of $A + BK$ lie in the interior of the unit circle. Typical design tools are pole placement and LQ control.

- V_f is selected as follows
$$V_f(x) = ||x||_{\overline{Q}}^2 \tag{2.11}$$
where $\overline{Q}$ satisfies the Lyapunov inequality $(A+BK)^\top \overline{Q}(A+BK) - \overline{Q} \leq -(Q + K^\top RK)$

2.3 Formalization of the Optimization Problem

The computational burden that MPC control imposes mostly depends on the way the control problem is formulated as an optimization one. In fact, as seen in the previous section, the finite-horizon constrained problem consists of minimizing a certain objective function subject to constraints and this operation may be not trivial. Specifically, in a large-scale context, it is worth considering the structure of the system in order to possibly take advantage of it.

In this section we will present two different formulations for the optimization problem which will be further considered in Chapter 6. Both the representations will have a lower block-triangular structure (LBT) as shown in [5].

2.3.1 Space-LBT Formulation

Consider the expression (2.3) which represents the dynamics of the i-th subsystem. More specifically, here we consider the case where subsystem i is coupled with subsystem $i-1$, for all $i>1$, i.e.

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k) + E_i x_{i-1}(k), \quad \text{while } E_1 = 0 \tag{2.12}$$

If we stack all together the variables across the spatial coordinate it is possible to obtain such structure:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ \vdots \\ x_{\mathrm{N}}(k+1) \end{bmatrix} = \begin{bmatrix} A_1 & 0 & \cdots & \cdots & 0 \\ E_2 & A_2 & \ddots & & \vdots \\ 0 & E_3 & A_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & E_{\mathrm{N}} & A_{\mathrm{N}} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ \vdots \\ x_{\mathrm{N}}(k) \end{bmatrix} + \begin{bmatrix} B_1 & 0 & \cdots & \cdots & 0 \\ 0 & B_2 & \ddots & & \vdots \\ \vdots & \ddots & B_3 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & B_{\mathrm{N}} \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \\ u_3(k) \\ \vdots \\ u_{\mathrm{N}}(k) \end{bmatrix} \tag{2.13}$$

Rewriting this equation in a compact form we obtain:

$$X(k+1) = AX(k) + BU(k) \tag{2.14}$$

Where $X(k) \in \mathbb{R}^n$, $U(k) \in \mathbb{R}^m$, $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$, with $n = \sum_{i=1}^N n_i$ and $m = \sum_{i=1}^N m_i$. Note that in the case the coupling is with the upstream subsystem, the structure will remain the same but the matrix A would be upper-triangular.

As discussed in Section 2.2, a set of inequalities describe physical/operational limitations on the system's (both state and input) variables i.e.,

$$C_i x_i(k) \le c_i \qquad D_i u_i(k) \le d_i \tag{2.15}$$
$$i = 1, \dots, N \quad k = 0, \dots, T-1$$

where $C_i \in \mathbb{R}^{q_i \times n_i}$, $D_i \in \mathbb{R}^{h_i \times m_i}$, $c_i \in \mathbb{R}^{q_i}$ and $d_i \in \mathbb{R}^{h_i}$. q_i and h_i are the number of inequality constraints on states and inputs, respectively, for each subsystem. According to the particular arrangement of the variables, an adequate compact expression for the inequality constraints can be derived for the Space-LBT system, that is:

$$CX(k) + DU(k) \leq \delta \tag{2.16}$$

In particular, the matrices C, D and the vector δ have the following structure:

$$C = \begin{bmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & C_N \\ 0 & \cdots & \cdots & 0 \\ \vdots & 0 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & 0 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 \\ D_1 & 0 & \cdots & 0 \\ 0 & D_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & D_N \end{bmatrix} \quad \delta = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_\mathrm{N} \\ d_1 \\ d_2 \\ \vdots \\ d_\mathrm{N} \end{bmatrix} \tag{2.17}$$

where $C \in \mathbb{R}^{(q+h)\times n}$ and $D \in \mathbb{R}^{(q+h)\times m}$. Note that C and D have the same number of rows, that is $q + h$ where $q = \sum_{i=1}^{N} q_i$ and $h = \sum_{i=1}^{N} h_i$. We call $n_{in} = q + h$ the total number of inequality constraints for the generic time instant k. Note that matrices C_i, D_i and vectors c_i and d_i are assumed to be time invariant.

Similarly to (2.7),the objective function takes the following form:

$$J\big(\{X(k)\}_{k=0}^{T}, \{U(k)\}_{k=0}^{T-1}\big) = \sum_{k=0}^{T-1} \Big(||X(k)||_Q^2) + ||U(k)||_R^2 \Big) + ||X(T)||_{\overline{Q}}^2 \tag{2.18}$$

In case the weight matrices are block-diagonal

$$Q = \begin{bmatrix} Q_1 & & \\ & \ddots & \\ & & Q_N \end{bmatrix} \quad R = \begin{bmatrix} R_1 & & \\ & \ddots & \\ & & R_N \end{bmatrix} \quad \overline{Q} = \begin{bmatrix} \overline{Q}_1 & & \\ & \ddots & \\ & & \overline{Q}_N \end{bmatrix}$$

where $Q_i \in \mathbb{R}^{n_i \times n_i}$, $R_i \in \mathbb{R}^{m_i \times m_i}$ and $\overline{Q}_i \in \mathbb{R}^{n_i \times n_i}$, the cost function is formally separable with respect to the single subsystems. Therefore, it can be written as

$$J = \sum_{i=1}^{N} J_i\big(\{x_i(k)\}_{k=0}^{T}, \{u_i(k)\}_{k=0}^{T-1}\big) \tag{2.19}$$

where

$$J_i = \sum_{k=0}^{T-1} \big(||x_i(k)||_Q^2 + ||u_i(k)||_R^2\big) + V_f\big(x_i(T)\big)$$

As discussed in Section 2.2, the MPC problem can be written as a standard quadratic programming where the minimization of the cost function (2.18), has to be performed with respect to the constraints related to the dynamics of the system (2.14) and to the inequality constraints (2.16), [4].

$$\begin{aligned} \min_{X(k),U(k)} \quad & J\big(\{X(k)\}_{k=0}^{T}, \{U(k)\}_{k=0}^{T-1}\big) \\ \text{subject to} \quad & X(0) = \widehat{X} \\ & X(k+1) = AX(k) + BU(k) \quad k = 0, \cdots, T-1 \\ & CX(k) + DU(k) \leq \delta \qquad\qquad k = 0, \cdots, T-1 \\ & C_f X(T) \leq \delta_f \end{aligned} \tag{2.20}$$

where C_f and δ_f are the matrices associated to the terminal constraint.

2.3.2 Time-LBT Formulation

In this section we introduce a similar equivalent formulation, based on the reordering of the variables according to the time coordinate rather than the spatial one. For further explanations, one may refer to [3] and [6].

In particular, considering the i-th subsystem and stacking the expression (2.12) for each time instant $k = 0, \ldots, T$, the following system of equation is obtained:

$$\begin{cases} x_i(t) = \widehat{x}_i \\ x_i(t+1) = A_i x_i(t) + B_i u_i(t) + E_i x_{i-1}(t) \\ x_i(t+2) = A_i x_i(t+1) + B_i u_i(t+1) + E_i x_{i-1}(t+1) \\ \vdots \end{cases}$$

By substituting at first the expression of $x_i(t+1)$ into $x_i(t+2)$, the following equation holds:

$$x_i(t+2) = A_i^2 x_i(t) + A_i B_i u_i(t) + B_i u_i(t+1) + A_i E_i x_{i-1}(t) + E_i x_{i-1}(t+1)$$

and with the same procedure, recursively, we obtain

$$x_i(t+k) = A_i^k x_i(t) + \sum_{j=0}^{k-1} A_i^{k-1-j} B_i u_i(t+j) + \sum_{j=0}^{k-1} A_i^{k-1-j} E_i x_{i-1}(t+j)$$

Basically, the state variables are expressed as a function of the current state $x_i(t)$, input sequences $\{u_i(k)\}_{k=t}^{t+T-1}$, and the sequence $\{x_{i-1}(j)\}_{j=t}^{t+k-1}$.

Fixing for semplicity of notation the current instant $t = 0$, it is possible now to write the matrix representation for the Time-LBT formulation and the result is this expression for the dynamics of the system:

$$\begin{bmatrix} x_i(0) \\ x_i(1) \\ x_i(2) \\ \vdots \\ x_i(T) \end{bmatrix} = \begin{bmatrix} I \\ A_i \\ A_i^2 \\ \vdots \\ A_i^T \end{bmatrix} x_i(0) + \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B_i & 0 & & \vdots \\ A_i B_i & B_i & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ A_i^{T-1} B_i & \cdots & \cdots & B_i \end{bmatrix} \begin{bmatrix} u_i(0) \\ u_i(1) \\ u_i(2) \\ \vdots \\ u_i(T-1) \end{bmatrix} +$$

$$\begin{bmatrix} 0 & \cdots & \cdots & 0 & 0 \\ E_i & 0 & & \vdots & \vdots \\ A_i E_i & E_i & \ddots & \vdots & \vdots \\ \vdots & & \ddots & 0 & \vdots \\ A_i^{T-1} E_i & \cdots & \cdots & E_i & 0 \end{bmatrix} \begin{bmatrix} x_{i-1}(0) \\ x_{i-1}(1) \\ x_{i-1}(2) \\ \vdots \\ x_{i-1}(T) \end{bmatrix} \tag{2.21}$$

In a compact form, the following equation is obtained for each subsystem:

$$X_i = \Gamma_i x_i(0) + \Omega_i U_i + \Psi_i X_{i-1} \tag{2.22}$$

where

$$X_i = \begin{bmatrix} x_i(0) \\ x_i(1) \\ x_i(2) \\ \vdots \\ x_i(T) \end{bmatrix} \in \mathbb{R}^{p_i} \qquad U_i = \begin{bmatrix} u_i(0) \\ u_i(1) \\ u_i(2) \\ \vdots \\ u_i(T-1) \end{bmatrix} \in \mathbb{R}^{l_i}$$

and $\Gamma_i \in \mathbb{R}^{p_i \times n_i}$, $\Omega_i \in \mathbb{R}^{p_i \times l_i}$ and $\Psi_i \in \mathbb{R}^{p_i \times p_{i-1}}$, with $p_i = \sum_{k=0}^{T} n_i = (T+1)n_i$ and $l_i = \sum_{k=0}^{T-1} m_i = Tm_i$.

Due to the new arrangement of the variables, the inequality constraints have to be written consistently with the Time-LBT formulation. Once the stacked vectors X_i and U_i are defined, with some manipulations the expressions (2.15) becomes:

$$\mathcal{C}_i X_i + \mathcal{D}_i U_i \leq \delta_i \tag{2.23}$$

In particular, $\mathcal{C}_i$, $\mathcal{D}_i$ and δ_i take the following form:

$$\mathcal{C}_i = \begin{bmatrix} C_i & 0 & \cdots & 0 & 0 \\ 0 & C_i & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & \cdots & C_i & 0 \\ 0 & \cdots & \cdots & 0 & 0 \\ \vdots & 0 & & \vdots & \vdots \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & 0 \end{bmatrix} \quad \mathcal{D}_i = \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & 0 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 \\ D_i & 0 & \cdots & 0 \\ 0 & D_i & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & D_i \end{bmatrix} \quad \delta_i = \begin{bmatrix} c_i \\ c_i \\ \vdots \\ c_i \\ d_i \\ d_i \\ \vdots \\ d_i \end{bmatrix} \tag{2.24}$$

where $\mathcal{C}_i \in \mathbb{R}^{(\alpha_i+\beta_i)\times p_i}$, $\mathcal{D}_i \in \mathbb{R}^{(\alpha_i+\beta_i)\times l_i}$ and $\delta_i \in \mathbb{R}^{(\alpha_i+\beta_i)}$. $\alpha_i = Tq_i$ and $\beta_i = Th_i$ are the total number of inequality constraints on states and inputs respectively of the i-th subsystem along the prediction horizon. We call $\gamma_i = \alpha_i + \beta_i$ the total number of inequality constraint for the i-th subsystem. Unlike the Space-LBT formulation, where the number of inequality constraints was the same at every time instant, according to this specific formulation, it varies and thus the use of the subscript i to the dimensions α, β and γ is justified. Notice that, for each subsystem, all the states are constrained except for the states related to the last time instant, as will be discussed in the following.

Similarly as done with the Time-LBT formulation, the quadratic cost function (2.19) can be separeted into its terms defining the matrices $\mathcal{Q}_i$ and $\mathcal{R}_i$

as

$$\mathcal{Q}_i = \begin{bmatrix} Q_i & & & \\ & \ddots & & \\ & & Q_i & \\ & & & \overline{Q}_i \end{bmatrix} \quad \mathcal{R}_i = \begin{bmatrix} R_i & & \\ & \ddots & \\ & & R_i \end{bmatrix}$$

Note that is possible to formulate the cost function and it is:

$$J\big(\{X_i\}_{i=1}^N, \{U_i\}_{i=1}^N\big) = \sum_{i=1}^{N} \Big(||X_i||^2_{\mathcal{Q}_i} + ||U_i||^2_{\mathcal{R}_i} \Big) \tag{2.25}$$

The system-wide optimization problem is:

$$\begin{aligned} \min_{X_i, U_i} \quad & J\big(\{X_i\}_{i=1}^N, \{U_i\}_{i=1}^N\big) & \\ \text{subject to} \quad & x_i(0) = \widehat{x}_i & i = 1, \cdots, N \\ & X_i = \Gamma x_i(0) + \Omega U_i + \Psi X_{i-1} & i = 1, \cdots, N \\ & \mathcal{C}_i X_i + \mathcal{D}_i U_i \leq \delta_i & i = 1, \cdots, N \end{aligned} \tag{2.26}$$

Note that the terminal constraints on state variables are included in $\mathcal{C}_i X_i$, since X_i contains the term $x_i(T)$. We remind that such formalization can be obtained granted that the terminal set is rectangular i.e.,

$$X(T) \in \mathbb{X}_f = \prod_{i=1}^{N} \mathbb{X}_f^{(i)} \iff x_i(T) \in \mathbb{X}_f^{(i)} \; \forall i$$

where

$$\prod_{i=1}^{N} \mathbb{X}_f^{(i)} = \mathbb{X}_f^{(1)} \times \cdots \times \mathbb{X}_f^{(N)}$$

and $\times$ is the Cartesian product.

For LBT systems, it is always possible to compute terminal sets of this type. However, for simplicity, in the algorithms presented in the following terminal constraints on state variables will not be considered.

Chapter 3

The irrigation channel case study

The algorithms proposed in this Thesis are applied to irrigation channel networks. As introduced in Section 1.1, the pools composing the whole channel present low level controllers aiming to regulate the water-levels. However, the violation of operational limitations and bad dynamic performances can occur if local decentralized feedback controllers are applied. In order to increase the efficiency of the irrigation network and to avoid manage saturations in an optimal way, advanced control strategies are required such as Model Predictive Control. More specifically, an MPC scheme can be designed to online adjust the level references for the low level controllers. Each MPC controller solves a long horizon constrained optimal control problem. However, the design and implementation of a centralized MPC controller typically suffers from scalability issues due to the big amount of data and by the large scale of the plant. For this reason, distributed and decentralized approaches must be considered, together with ad-hoc optimization methods, which specifically account for the peculiar system structure.

3.1 Modelization

Different models are used for describing the dynamics of the single pool, spanning from complex non-linear systems, to simplified linear ones. In this chapter a simple first order modelization of the single pool controlled with a PID level regulator is presented. In particular, it will be shown how the state space representation of the closed loop system can be expressed by means of (2.3). As discussed in Chapter 2, it will be possible to reduce the complexity of the MPC optimization problem, by exploiting the structured Space-LBT and Time-LBT formulations (2.14) and (2.22), respectively.

3.1.1 First-order Model of Each Pool

Traditionally, open channel dynamics are described using the shallow water equations, or the so-called St-Venant equation. For a detailed modelization we address the reader to [7].

The first-order continuous-time model considered in this Thesis is

$$S_i \dot{y}_i(t) = \gamma_i h_i^{3/2}(t-\tau_i) - \gamma_{i-1} h_{i-1}^{3/2}(t) - d_i(t) \tag{3.1}$$

where

- S_i is the surface of the pool expressed in $[\mathrm{m}^2]$
- y_i is the water level expressed in $[\mathrm{m}]$
- γ_i is a coefficient expressed in $[\frac{\mathrm{m}^{3/2}}{\mathrm{min}}]$
- h_i is the overhead, i.e. the displacement between the top of the gate and crest of the wave, expressed in $[\mathrm{m}]$
- τ_i is the delay expressed in $[\mathrm{min}]$
- d_i is the off-take and it is a flux expressed in $[\frac{\mathrm{m}^3}{\mathrm{min}}]$

Such equation consists of a mass balance in the the pool. Note that the off-take term d_i will be neglected in the sequel for simplicity. Indeed, the variation of water-level in a pool is proportional to the difference between the upstream flow and the downstream flow.

Defining now the *scaled flow* and the *real flow* as respectively

$$f_i(t) = h_i^{3/2}(t) \quad \left[\mathrm{m}^{3/2}\right]$$
$$F_i(t) = \gamma_i h_i^{3/2}(t) \quad \left[\frac{\mathrm{m}^3}{\mathrm{min}}\right]$$

and the coefficients

$$c_{in,i} = \frac{\gamma_i}{S_i} \quad \left[\frac{\mathrm{m}^{-1/2}}{\mathrm{min}}\right]$$
$$c_{out,i} = \frac{\gamma_{i-1}}{S_i} \quad \left[\frac{\mathrm{m}^{-1/2}}{\mathrm{min}}\right]$$

the equation (3.1) becomes

$$\dot{y}_i(t) = c_{in,i} f_i(t - \tau_i) - c_{out,i} f_{i-1}(t) \tag{3.2}$$

3.1.2 Water-Level Regulation

The decentralized low-level regulation scheme discussed in this section consists in a number of PI controllers (i.e., one regulator for each pool) which aim to control the level $y_i(t)$ of each pool on the basis of a given set point $r_i(t)$. More

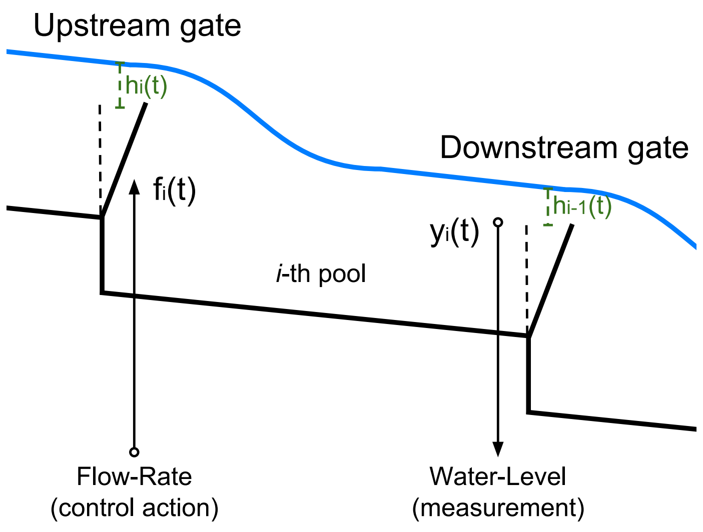


Figure 3.1: Pool: measurement and control action

specifically, each controller regulates the flow of the i-th pool acting on the upstream gate. However, the water measurement is taken at the downstream gate (see Figure 3.1) so the control action $f_i(t)$ results delayed due to the slow dynamics of the pool:

$$g_i(t) = f_i(t - \tau_i)$$

The architecture of the so called *distant-downstream decentralized control* is shown in Figure 3.2.

Let $Y_i(s) = \mathcal{L}\{y_i(t)\}$, $F_i(s) = \mathcal{L}\{f_i(t)\}$, $R_i(s) = \mathcal{L}\{r_i(t)\}$, where the operator $\mathcal{L}\{\cdot\}$ denotes the Laplace transformation. The mass balance equation of the pool becomes:

$$Y_i(s) = \frac{c_{in,i}}{s} F_i(s) e^{-\tau_i s} - \frac{c_{out,i}}{s} F_{i-1}(s)$$

Delay

The dynamics of the pools is typically very slow. For this reason the controlled system in closed loop will be designed in order to display a small bandwidth. The delay, i.e. the term $e^{-\tau_i s}$, is represented using a first order Padè approximation

$$e^{-\tau_i s} \approx \frac{1 - \frac{\tau_i}{2} s}{1 + \frac{\tau_i}{2} s} = \frac{-s + \frac{2}{\tau_i}}{s + \frac{2}{\tau_i}} \tag{3.3}$$

As well-known, such approximation reduces the phase margin of the controlled system. However, if the dynamics is sufficiently slow, this approximation does

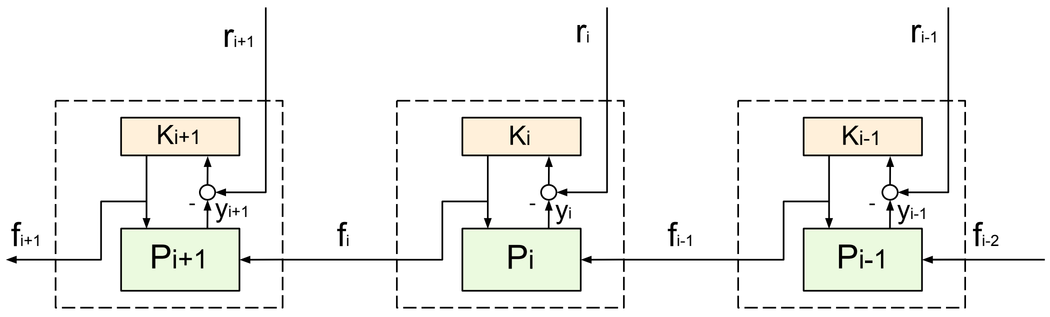


Figure 3.2: Distant-downstream decentralized control scheme

not compromise the stability properties of the closed loop system. In particular, the lower is the crossover frequency, the more accurate the Padè approximation is.

Low-level Controller

Different decentralized control architectures, aiming to control the water level of the single pool, can be found in the literature (e.g., PI, PID, PID with feed-forward action). In particular, such control strategies are mostly concerned with control of the propagation of water-level errors upstream during the transient. More specifically, any control action taken to control the water-level of a pool affect the water-level of the upstream one. Therefore, this results in a corresponding water-level error, which propagates upstream along the channel. However, the transient properties of the single pool are not a topic for this work. For this reason, in order to simplify the modelization, the low level controllers applied to each pool are supposed to be simple PI.

The transfer function of the PI regulator between the error $E_i(s) = R_i(s) - Y_i(s)$, and the control law $F_i(s)$ takes the classical form:

$$F_i(s) = \frac{K_i(1 + T_{I,i}s)}{sT_{I,i}(1 + T_{F,i}s)}\big(R_i(s) - Y_i(s)\big) \tag{3.4}$$

Where

- K_i is the proportional gain
- $T_{I,i}$ is the integral time
- $T_{F,i}$ is a time constant added in order to have the pole $1/T_{F,i}$ at high frequency. Such pole is necessary in order to obtain a strictly proper transfer function of the regulator. In particular, it will be usefull in the state representation of the regulator presented in the following.

3.1.3 State Space Representation

Controller

It is possible to rewrite the equation (3.4) as

$$F_i(s) = \frac{\frac{K_i}{T_{F,i}}s + \frac{K_i}{T_{F,i}T_{I,i}}}{s^2 + \frac{1}{T_{F,i}}s}\big(R_i(s) - Y_i(s)\big)$$

By means of the controllability canonical form [8], the state space representation of the controller becomes :

$$\begin{cases} \dot{x}_i^K(t) = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T_{F,i}} \end{bmatrix} x_i^K(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \big(r_i(t) - y_i(t)\big) \\ f_i(t) = \begin{bmatrix} \frac{K_i}{T_{F,i}} & \frac{K_i}{T_{F,i}T_{I,i}} \end{bmatrix} x_i^K(t) \end{cases} \tag{3.5}$$

Notice that using the superscript K in x_i^K we denote the state variables associated to the controller.

Delay

Using the Padè approximation (3.3), the delayed control action, i.e. the flow, in frequency domain is given by

$$G_i(s) = -F_i(s) + \frac{\frac{4}{\tau_i}}{s + \frac{2}{\tau_i}}F_i(s)$$

Using again the controllability canonical form, the state representation of the delayed control action may be achieved:

$$\begin{cases} \dot{x}_i^D(t) = -\frac{2}{\tau_i}x_i^D(t) + f_i(t) \\ g_i(t) = \frac{4}{\tau_i}x_i^D(t) - f_i(t) \end{cases}$$

Deriving the expression of the flow $g_i(t)$ and replacing the expression of $f_i(t)$ given by (3.5), we eventually obtain:

$$\dot{g}_i(t) = -\frac{2}{\tau_i}g_i(t) + \begin{bmatrix} \alpha & \beta \end{bmatrix} x_i^K(t) - \frac{K_i}{T_{F,i}}r_i(t) + \frac{K_i}{T_{F,i}}y_i(t) \tag{3.6}$$

$$\text{with} \quad \begin{bmatrix} \alpha & \beta \end{bmatrix} = \begin{bmatrix} \frac{2K_i}{\tau_i T_{I,i} T_{F,i}} & \frac{2K_i}{\tau_i T_{F,i}} - \frac{K_i}{T_{F,i}T_{I,i} - T_{F,i}^2} \end{bmatrix}$$

Combining the state space representations (3.2), (3.5) and (3.6) it is possible to obtain a full modelization of the controlled pool:

$$\begin{bmatrix} \dot{y}_i(t) \\ \dot{g}_i(t) \\ \dot{x}_i^{K,1}(t) \\ \dot{x}_i^{K,2}(t) \end{bmatrix} = \begin{bmatrix} 0 & C_{\text{in,i}} & 0 & 0 \\ \frac{K_i}{T_{F,i}} & -\frac{2}{\tau_i} & \alpha & \beta \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & -\frac{1}{T_{F,i}} \end{bmatrix} \begin{bmatrix} y_i(t) \\ g_i(t) \\ x_i^{K,1}(t) \\ x_i^{K,2}(t) \end{bmatrix} + \begin{bmatrix} -C_{\text{out,i}} \\ 0 \\ 0 \\ 0 \end{bmatrix} f_{i-1}(t) + \begin{bmatrix} 0 \\ -\frac{K_i}{T_{F,i}} \\ 0 \\ 0 \end{bmatrix} r_i(t)$$

All the states variables have a physical meaning. More specifically, y_i is the water level, g_i is the delayed control action over the upstream gate and $x_i^K = \begin{bmatrix} x_i^{K,1} & x_i^{K,2} \end{bmatrix}^\top$ are the variables associated to the low-level controller.

Defining

$$A_i^{\text{C}} = \begin{bmatrix} 0 & C_{\text{in,i}} & 0 & 0 \\ \frac{K_i}{T_{F,i}} & -\frac{2}{\tau_i} & \alpha & \beta \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & -\frac{1}{T_{F,i}} \end{bmatrix} \qquad B_i^{\text{C}} = \begin{bmatrix} 0 \\ -\frac{K_i}{T_{F,i}} \\ 0 \\ 0 \end{bmatrix} \qquad F_i^{\text{C}} = \begin{bmatrix} -C_{\text{out,i}} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$H_{i-1}^{\text{C}} = \begin{bmatrix} 0 & 0 & \frac{K_{i-1}}{T_{F,i-1}} & \frac{K_{i-1}}{T_{F,i-1}T_{I,i-1}} \end{bmatrix}$$

the model expressed in a compact form becomes:

$$\begin{cases} \dot{x}_i(t) = A_i^{\text{C}} x_i(t) + \begin{bmatrix} B_i^{\text{C}} & F_i^{\text{C}} \end{bmatrix} \begin{bmatrix} r_i(t) \\ f_{i-1}(t) \end{bmatrix} \\ f_{i-1}(t) = H_{i-1}^{\text{C}} x_{i-1}(t) \end{cases} \tag{3.7}$$

Discretization

By means of the matlab function *c2d* (i.e., continuous-to-discrete) the continuous-time system (3.7) has been discretized, obtaining the following representation:

$$\begin{cases} x_i(k+1) = A_i x_i(k+1) + \begin{bmatrix} B_i & F_i \end{bmatrix} \begin{bmatrix} r_i(k+1) \\ v_i(k+1) \end{bmatrix} \\ v_i(k+1) = H_{i-1} x_{i-1}(k+1) \end{cases} \tag{3.8}$$

Eventually, defining $E_i = F_i H_{i-1}$, the discrete-time model of the controlled pool becomes:

$$x_i(k+1) = A_i x_i(k) + B_i r_i(k) + E_i x_{i-1}(k) \tag{3.9}$$

Note that the coupling between subsytems is represented through state variables (i.e., the term $E_i x_{i-1}(t)$). Moreover, the expression (3.9) has the same structure of (2.3). Such modelization will allow us to use both the Space-LBT and the Time-LBT formulations presented in Section 2.3.1 and Section 2.3.2, respectively, in order to represent the dynamics of the whole irrigation network.

3.2 High-Level Controller

As shown in the previous section, the input to the low level regulators of each pool is the water-level error defined as

$$e_i(t) = r_i(t) - y_i(t)$$

where $r_i(t)$ is the water-level reference. However, the application of this decentralized control scheme to the irrigation network leads to bad transient performances and possible violation of physical constraints. Therefore, an advanced control strategy is needed in order to keep the water level around a security range. For all these reasons an MPC control strategy will be applied

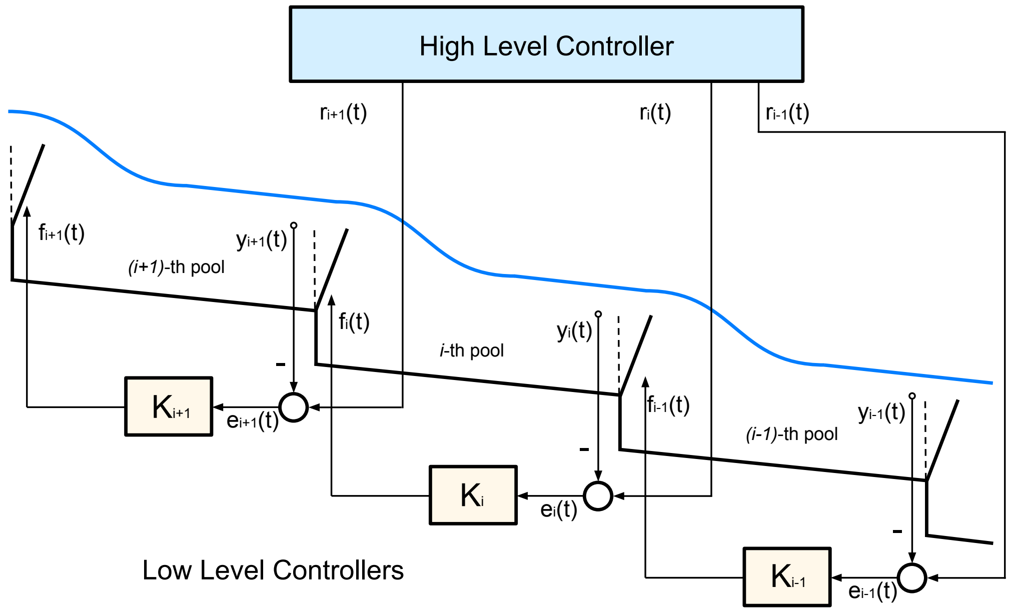


Figure 3.3: Low level and high level MPC controllers

to the whole system. More specifically, MPC aims to control the irrigation network by an on-line adjustment of the nominal water-level references $r_i(t)$ applied to the systems (3.9).

The control hierarchy is shown in Figure 3.3.

3.3 Model Parameters and Limitations

In Table 3.1 all the physical parameters of a channel composed by four pools as well as the parameters associated to the low-level controllers are reported. Such values, taken by [9], will be used for all the simulations carried out in this Thesis. All the physical data and the controller parameters for the simulations have been provided by Rubicon Water Pty. Ltd., and by the University of Melbourne, and they are referred to the East Goulburn-30 Irrigation network.

We remind that, according to the state space representation (3.9), the water-level of the i-th pool is expressed by the variable $y_i(t)$, while the scaled flow $f_i(t)$ is the linear combination of the state variables referred to the low level controller, i.e.

$$f_i(t) = \begin{bmatrix} \frac{K_i}{T_{F,i}} & \frac{K_i}{T_{F,i}T_{I,i}} \end{bmatrix} \begin{bmatrix} x_i^{K,1}(t) \\ x_i^{K,2}(t) \end{bmatrix}$$

The control strategies that will be introduced aim to constrain the water-level of each pool and the actual flow imposed by the gates, i.e. $F_i(t) = \gamma_i f_i(t)$,

Pool	C_{in}	C_{out}	τ	γ	K_D	T_I	T_F
	$[\text{m}^{-\frac{1}{2}}/\text{min}]$	$[\text{m}^{-\frac{1}{2}}/\text{min}]$	[min]	$[\text{m}^{\frac{3}{2}}/\text{min}]$			
1	0.0461	0.0461	2	107.46	1.259	50.058	5.935
2	0.0438	0.0438	3	107.46	1.057	62.716	7.436
3	0.0498	0.0305	4	175.67	0.863	67.600	8.015
4	0.0591	0.0591	3	175.67	0.913	53.830	6.382

Table 3.1: East Goulburn-30 Irrigation Channel Pools and Controller Parameters

in order to keep the water-level around a range of values, and to avoid actuator saturations. Such limitations are:

$$y_{min,i} = 0.92\,\text{m} \ \leq y_i(t) \leq 1.08\,\text{m} \ = y_{max,i} \tag{3.10a}$$

$$F_{min,i} = 0\,\frac{\text{m}^3}{\text{min}} \ \leq \gamma_i f_i(t) \leq 12\,\frac{\text{m}^3}{\text{min}} \ = F_{max,i} \tag{3.10b}$$

Constraints on input variables are eventually considered. We remind that for the high level controller the control variables are the water-level references $r_i(t)$ of the low level controllers applied at each pool. More specifically, for each subsystem, the input is a scalar. They will be expressed as

$$r_{min,i} \leq r_i(t) \leq r_{max,i}$$

where

$$r_{min,i} = y_{min,i}$$

$$r_{max,i} = y_{max,i}$$

Chapter 4

Cooperative Distributed MPC

The limitations related to the traditional implementation of MPC centralized controllers have pushed research on the so called decentralized and distributed control strategies. Especially in chemical plants, plantwide control has traditionally been implemented in a decentralized fashion, i.e., each subsystem is controlled independently and network interactions are treated at most as local subsystem disturbances. However, when the interactions between subsystems are strong, decentralized control becomes unreliable, [10].

In this chapter a Cooperative Distributed MPC control strategy [10] is applied to a large-scale system where the dynamics of each subsystem is expressed by (2.1). Through this control strategy all regulators solve in parallel, but iteratively, different optimization problems. A negotiation phase is also required for gaining a consensus of the control action to be taken. More specifically, through negotiation, a global optimal control action is obtained and the closed loop stability of the whole system is ensured. The purpose of such control strategy is to ease the computational burden required with respect to a centralized formalization of the optimal control problem. Note that, this control strategy has been applied to the same case study in [11].

4.1 Modelization

For each i-th subsystem, with $i = 1, \ldots, N$, composing the whole large-scale system, a model of the following form is used:

$$\begin{cases} z_i(k+1) = \overline{A}_i z_i(k) + \sum_{j=1}^{N} \overline{B}_{ij} u_j(k) \\ y_i(k) = \overline{C}_i z_i(k) \end{cases} \tag{4.1}$$

where $z_i \in \mathbb{R}^{n_i}$ and $y_i \in \mathbb{R}^{p_i}$ are the vectors of the state variables and outputs, respectively, of the i-th subsystem and $u_j \in \mathbb{R}^{m_j}$ is the vector of inputs of the generic j-th subsystem. Note that model (4.1) is rarely obtained through physical-based modelling of the systems. More frequently, such a formulation (denoted state-decoupled) must be obtained through a suitable model expansion, described below.

Initially (i.e., in Section 4.1.1) each subsystems composing the whole LSS is modelled as proposed by [10]. However, due to computational issues, a low order model based on the Space LBT formulation (2.14) will be introduced in Section 4.1.2.

4.1.1 Standard Model for Cooperative Distributed MPC

As shown in [10], by applying the superposition principle for each subsystem i we define a set of dynamic models, each describing the effect of each input u_j, with $j = 1, \ldots, N$, on the states and outputs of the subsystem i:

$$\begin{cases} z_{i1}(k+1) = A_{i1} z_{i1}(k) + B_{i1} u_1(k) \\ z_{i2}(k+1) = A_{i2} z_{i2}(k) + B_{i2} u_2(k) \\ \vdots \\ z_{iN}(k+1) = A_{iN} z_{iN}(k) + B_{iN} u_N(k) \\ y_i(k) = \sum_{j=1}^{N} C_{ij} z_{ij}(k) \end{cases} \tag{4.2}$$

Regarding the cascade systems considered in this Thesis, it is apparent that the i-th subsystem is influenced only by the inputs of the j-th subsystems with

$j = 1, \dots, i$ according to the numeration shown in Figure 2.1. Consistently with this remark, the system (4.2) reduces to

$$\begin{cases} z_{i1}(k+1) = A_{i1} z_{i1}(k) + B_{i1} u_1(k) \\ z_{i2}(k+1) = A_{i2} z_{i2}(k) + B_{i2} u_2(k) \\ \vdots \\ z_{ij}(k+1) = A_{ij} z_{ij}(k) + B_{ij} u_j(k) \\ y_i(k) = \sum_{j=1}^{j} C_{ij} z_{ij}(k) \end{cases}$$

with

$$A_{ij} = \begin{bmatrix} A_j & 0 & \cdots & 0 \\ E_{j+1} & A_{j+1} & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & E_i & A_i \end{bmatrix}, \qquad B_{ij} = \begin{bmatrix} B_j \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Note that all the matrices A_i, E_i and B_i are defined as in Section 2.1 and $z_{ij} \in \mathbb{R}^{n_{ij}}$, $A_{ij} \in \mathbb{R}^{n_{ij} \times n_{ij}}$, $B_{ij} \in \mathbb{R}^{n_{ij} \times m_j}$ and $C_{ij} \in \mathbb{R}^{p_i \times n_{ij}}$ with $n_{ij} = \sum_{l=j}^{i} n_l$. Collecting the state variables in a vector

$$z_i = [z_{i1}^\top, z_{i2}^\top, \dots, z_{ij}^\top]^\top$$

it is possible to obtain a model in the form (4.1), where:

$$\begin{aligned} \overline{A}_i &= \mathrm{diag}(A_{i1}, A_{i2}, \dots, A_{ij},) \\ \overline{B_{ij}} &= \begin{bmatrix} 0 \\ \vdots \\ 0 \\ B_{ij} \\ 0 \\ \vdots \end{bmatrix} \\ \overline{C}_i &= \begin{bmatrix} C_{i1} & C_{i2} & \cdots & C_{ij} \end{bmatrix} \end{aligned} \tag{4.3}$$

As the number of subsystems increases such modelization suffers from scalability issues, due to the rapid growth of the number of state variables composing the generic i-th subsystem. More specifically, supposing that each subsystem (2.1) has order $n_i = n$ for all $i = 1, \cdots, N$, with the modelization procedure described above the i-th subsystem will be described by $n \sum_{j=1}^{i} j$ states, i.e., $n\frac{i(i+1)}{2}$ variables. Therefore, to obtain a more tractable model, a lower order modelization of the LSS will be used.

4.1.2 Low Order Model

The low order dynamic model of the i-th subsystem is built in the same fashion as the Space-LBT formulation presented in Section 2.3.1. More specifically, we define

$$z_i^L(k) = \begin{bmatrix} z_{i1}^L(k) \\ z_{i2}^L(k) \\ \vdots \\ z_{ii}^L(k) \end{bmatrix}$$

and we describe the dynamics of $z_i^L(k)$ as

$$\begin{bmatrix} z_{i1}^L(k+1) \\ z_{i2}^L(k+1) \\ z_{i3}^L(k+1) \\ \vdots \\ z_{ii}^L(k+1) \end{bmatrix} = \begin{bmatrix} A_1 & 0 & \cdots & \cdots & 0 \\ E_2 & A_2 & \ddots & & \vdots \\ 0 & E_3 & A_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & E_{\mathrm{i}} & A_{\mathrm{i}} \end{bmatrix} \begin{bmatrix} z_{i1}^L(k+1) \\ z_{i2}^L(k+1) \\ z_{i3}^L(k+1) \\ \vdots \\ z_{ii}^L(k+1) \end{bmatrix} + \begin{bmatrix} B_1 & 0 & \cdots & \cdots & 0 \\ 0 & B_2 & \ddots & & \vdots \\ \vdots & \ddots & B_3 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & B_{\mathrm{i}} \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \\ u_3(k) \\ \vdots \\ u_{\mathrm{i}}(k) \end{bmatrix} \quad (4.4)$$

From this, we obtain the state decoupled formulation (4.1) by defining

$$\overline{A}_i = \begin{bmatrix} A_1 & 0 & \cdots & \cdots & 0 \\ E_2 & A_2 & \ddots & & \vdots \\ 0 & E_3 & A_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & E_{\mathrm{i}} & A_{\mathrm{i}} \end{bmatrix} \quad \text{and} \quad \overline{B}_{ij} = \begin{bmatrix} 0 \\ \vdots \\ B_j \\ 0 \\ \vdots \end{bmatrix}$$

Note that the number of state variables required to describe the dynamics of the i-th subsystem is smaller than the number of variables required by modelization described in Section 4.1.1. In particular, they are equal to $\sum_{j=1}^{i} n_j$. However, the matrix $\overline{A}_i$ is not block diagonal as proposed by [10]. In the following, we will show that the loss of this property will not compromise the features of the Cooperative Distributed MPC algorithm.

4.2 The Cooperative Distributed MPC Control Scheme

According to the modelization (4.1), for each subsystem i, with $i = 1, \dots, N$ a local cost function is defined:

$$J_i(z_i(t), u_i(t), \dots, u_i(t+T-1)) = \sum_{k=t}^{t+T-1} \left[||z_i(k)||^2_{Q_{z_i}} + ||u_i(k)||^2_{R_i} \right] + ||z_i(t+T)||^2_{\overline{Q}_{z_i}}$$

The current time instant t will be set equal to zero in order to simplify the notation. A global cost function is also defined

$$J(z_1(0), \dots, z_N(0), u_1(0), \dots, u_N(0), \dots, u_N(T-1)) = \sum_{i=1}^{N} \rho_i J_i(z_i(0), u_i(0), \dots, u_i(T-1)) \quad (4.5)$$

where $\rho_i > 0$ is the relative weight of the i-th cost function. We note that each local cost J_i is function of the local input u_i and of the j-th inputs u_j, with $j = 1, \dots, i$, since the state z_i depends on such variables.

4.2.1 Constraints on Inputs and State Variables

Constraints on inputs are considered and they are imposed on the i-th subsystem, for all $i = 1, \cdots, N$. Specifically they can be expressed by:

$$u_i(k) \in \mathbb{U}_i \quad i = 1, \ldots, N \qquad k = 0, \ldots, T-1$$

Note that constraints on the state variables are not regarded in the implementation of the cooperative distributed MPC algorithm proposed by [10]. However, for generality, we will test the capability of the control scheme to cope with the latter, and more specifically we will introduce constraints of type

$$z_i(k) \in \mathbb{Z}_i \quad i = 1, \ldots, N \qquad k = 1, \ldots, T \tag{4.6}$$

where $\mathbb{Z}_i$ is convex with the origin in its interior. For obvious reasons, such constraints are softened by means of appropriate slack variables that allow their temporary violation. The reduced efficiency of the algorithm with active constraints on state variables will be demonstrated in Chapter 7, devoted to the simulations tests. Finally, note that zero terminal constraints are in general required, but they can be neglected under the assumption that A_i is Schur stable for $i = 1, \cdots, N$, which will be the standing assumption for the remainder of the Thesis.

4.2.2 Optimization Problem

The optimal control law to be applied is obtained through a negotiation phase between all the controllers $\mathcal{C}_i$. The local solutions obtained after a number of negotiation steps are only sub-optimal compared to the solution calculated with a centralized controller applied to the whole system. Indeed, the Cooperative Distributed MPC control strategy attains the optimality only at the limit.

At each iteration p all the controllers $\mathcal{C}_i$ solve an appropriate minimization problem, minimizing the global cost function with respect to the local inputs

u_i, that is

$$\begin{aligned}
\min_{u_i(0),\ldots,u_i(T-1)} \quad & J(z_1(0),\ldots,z_N(0),u_1(0),\ldots,u_N(0),\ldots,u_N(T-1)) \\
\text{subject to} \quad & \begin{bmatrix} z_1(k+1) \\ z_2(k+1) \\ \vdots \\ z_N(k+1) \end{bmatrix} = \begin{bmatrix} \overline{A}_1 & 0 & \cdots & 0 \\ 0 & \overline{A}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \overline{A}_N \end{bmatrix} \begin{bmatrix} z_1(k) \\ z_2(k) \\ \vdots \\ z_N(k) \end{bmatrix} \\
& \qquad + \sum_{j=1}^{N} \begin{bmatrix} \overline{B}_{1j} \\ \overline{B}_{2j} \\ \vdots \\ \overline{B}_{Nj} \end{bmatrix} u_j(k) \qquad k = 0,\cdots,T-1 \\
& u_j(k) = u_j^p(k); \quad k = 0,\cdots,T-1 \ \ \forall j \neq i \\
& u_i(k) \in \mathbb{U}_i; \ \ k = 0,\cdots,T-1 \\
& z_i(k) \in \mathbb{Z}_i; \quad k = 1,\cdots,T \ \ i = 1,\cdots,N
\end{aligned} \tag{4.7}$$

For subsystem i the input sequence $u_j(k)$, with $j \neq i$, are the sequences $u_j^p(k)$, $k = 0,\cdots,T-1$, transmitted at the previous negotiation step by the other subsystems. Note that the cost function and the equality constraints referred to the dynamics of the system along the prediction horizon are global while the optimal solution is local, and consists of the optimal trajectory

$$u_i^*(k) \qquad k = 0,\cdots,T-1$$

During the cooperation phase each controller weights its control input sequence with the trajectory transmitted at the previous step, i.e.

$$u_i^{p+1}(k) = w_i u_i^*(k) + (1-w_i)u_i^p(k) \quad k = 0,\cdots,T-1 \tag{4.8}$$

where w_i is proper weight associated to the i-th subsystem. The trajectory $u_i^{p+1}(k)$ $k = 1,\cdots,T-1$ is now transmitted to the other subsystems for the subsequent negotiation step. At the last iteration $\overline{p}$ each controller $\mathcal{C}_i$ applies the control input $u_i^{\overline{p}}(k)$. The global optimality of the solution is guaranteed if for all the controllers, in two consecutive iterations the following condition is satisfied:

$$u_i^p(k) = u_i^{p+1}(k) \quad k = 0, \cdots, T-1 \quad i = 1, \cdots, N \tag{4.9}$$

However the number of iterations needed to attain (4.9) are sensitive to the initial conditions of the system and to the weights imposed on state and input variables. Indeed, such number can be very high, which could penalize the efficiency of the algorithm. For this reason a maximum number of iterations p_{max} and a tolerance for the sub-optimality of the solution ϵ are set. Therefore, introducing the parameter

$$\begin{gathered} \phi_i = \left\| \frac{u_i^p(k) - u_i^{p+1}(k)}{u_i^p(k)} \right\|^2 \\ i = 1, \cdots, N \quad k = 0, \cdots, T-1 \end{gathered} \tag{4.10}$$

the following stopping condition is enforced in the algorithm

$$p \geq p_{max} \quad \vee \quad \phi_i \leq \epsilon$$

This choice allows us to improve the performances of the algorithm in the calculation of the solution as it will be explained in the following.

4.2.3 Design Parameters

Taking into account the modelization (4.1) the cost function and constraints are expressed with respect to the outputs y_i, which have a physical meaning. For this reason the cost functions J_i and the constraints actually implemented on the algorithm have the following expressions:

$$\begin{aligned} & J_i(y_i(0), u_i(0), \ldots, u_i(T-1)) = \sum_{k=0}^{T-1} \left[||y_i(k)||^2_{Q_{y_i}} + ||u_i(k)||^2_{R_i} \right] + ||z_i(T)||^2_{Q_{z_i}} \\ & u_i(k) \in \mathbb{U}_i \\ & y_i(k) \in \mathbb{Y}_i \\ & i = 1, \ldots, N \;\; ; \; k = 1, \ldots, T \end{aligned} \tag{4.11}$$

where $\mathbb{Y}_i$ is a polytopic convex set that includes the origin in its interior. The constraints on outputs and inputs are supposed to be linear and expressed by the following inequalities:

$$\begin{aligned} &H_i y_i(k) \leq k_i \qquad i = 1, \cdots, N \quad k = 1, \cdots, T \\ &D_i u_i(k) \leq d_i \qquad i = 1, \cdots, N \quad k = 0, \cdots, T-1 \end{aligned}$$

Recalling that $y_i(k) = \overline{C}_i z_i(k)$ and defining the matrices

$$\begin{aligned} Q_{z_i} &= \overline{C}_i^\top Q_{y_i} \overline{C}_i \\ \overline{H}_i &= H_i \overline{C}_i \end{aligned} \tag{4.12}$$

it is finally possible to redefine the cost function and the constraints in the form (4.5), (4.6) as function of the states variables and inputs only.

Algorithm 1 Cooperative Distributed MPC

```
1: INPUT:
       $N$, $T$
       $A_i$, $B_i$, $E_i$, $C_i$ $H_i$, $D_i$, $k_i$, $d_i$, $Q_i$, $R_i$    $i = 1, \cdots, N$
       Initial values $z_i(\cdot)$    $i = 1, \cdots, N$
       Weights $\rho_i$, $w_i$    $i = 1, \cdots, N$
       Tollerance $\epsilon \in [10^{-6}, 10^{-10}]$, $p_{max} \in [15, 100]$
2: OUTPUT:
       Optimal control sequence $u_i^*(k)$    $i = 1, \cdots, N$    $k = 0, \cdots, T-1$
3: PROCEDURE:
4: for $i : 1 \to N$ do (Offline)
5:     Build the matrices $\overline{A}_i$, $\overline{B}_{ij}$, $\overline{C}_i$ by means of (4.1) and $Q_{z_i}$, $\overline{H}_i$ by means
       of (4.12)
6: Define $\overline{Q}_{z_i}$ such that $\overline{A}_i^\top \overline{Q}_{z_i} \overline{A}_i - \overline{Q}_{z_i} = -Q_{z_i}$ (recall that $\overline{A}_i$ is assumed
   Schur stable)
7: INIZIALIZATION OF THE GENERIC ITERATION
8: p=0
9: $\phi_i = 1 \quad i = 1, \cdots, N$
10: $u_i^o(k) = [u_i^*(0|t-1), \cdots, u_i^*(0|t+T-2), 0]^\top$ where $u_i^*(k|t-1)$ is the
    optimal input trajectory computed at the previous time instant $t-1$.
    $i = 1, \cdots, N \quad k = 0, \cdots, T-1$
11: while $\big(\phi_i > \epsilon, \forall i \,||\, p < p_{max}\big)$ do
12:     Each controller $\mathcal{C}_i$ solve the minimization problem (4.7) obtaining the
        current optimal solution $u_i^*(k)$ with $i = 1, \cdots, N \quad k = 0, \cdots, T-1$
13:     NEGOTIATION PHASE
14:     Each controller calculates the trajectory $u_i^{p+1}(k)$ through (4.8)
15:     Each controller send $u_i^{p+1}(k)$ to all the others controllers.
16:     Each controller calculates $\phi_i$ for the stopping condition with (4.10)
17:     $p = p + 1$
```

Chapter 5

Numerical methods for MPC optimization problems

In Chapter 2 the MPC control strategy has been introduced and formulated. As discussed, implementing a model predictive controller requires to solve a convex quadratic program (QP) at each time step. However, for LSS this procedure demands a great amount of online computation which scales both with the prediction horizon T and the number of subsystems N.

In this chapter the *Interior point methods* (and in particular the *Mehrotra's Predictor-Corrector Algorithm*) will be presented and used to solve the optimization problem with a naive approach: this corresponds to solving the quadratic program without exploiting the inherent structure of the system. As we will more thoroughly discuss, an algorithm stemming from this naive approach has a complexity that scales as $\mathcal{O}(N^3T^3)$.

The search for efficient numerical methods for solving such optimization problem is justified. With this aim, the formulations denoted Space-LBT and Time-LBT will be efficiently employed. In Chapter 6, we will exploit the two state space representations (2.14) and (2.22), leading to algorithmic solutions where the computational effort is reduced.

5.1 MPC Problem and Quadratic Programming

In this section we want to show how an MPC control problem of the type presented in Section 2.3 can be written as a sequence of convex quadratic programs.

Let us consider the formulation introduced in Section 2.3.1. The constraints of the optimization problem (2.20) are defined for each time instant $k = 0, \dots, T$ where T is the prediction horizon. Recalling the particular arrangement of the variables

$$X(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_N(k) \end{bmatrix} \qquad U(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \\ \vdots \\ u_N(k) \end{bmatrix}$$

it is possible to define a vector which contains all the states and inputs for the whole system along the prediction horizon T:

$$Z = \left[X(0)^\top, U(0)^\top, X(1)^\top, U(1)^\top, \dots, U(T-1)^\top, X(T)^\top\right]^\top \tag{5.1}$$

The quadratic problem can be defined with its standard formulation, where the cost function is quadratic and the equality and inequality constraints are linear in Z

$$A_{eq} Z = b_{eq}$$
$$A_{ineq} Z \leq b_{ineq}$$

More specifically, the equality constraints are derived from the dynamic model of the centralized system (2.14), i.e.

$$A_{eq} = \begin{bmatrix} I & & & & & \\ A & B & -I & & & \\ & & & \ddots & & \\ & & & A & B & -I \end{bmatrix} \qquad b_{eq} = \begin{bmatrix} \widehat{X} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Likewise, the inequality constraints, which have to be consistent with (2.16), are

$$A_{ineq} = \begin{bmatrix} C & D & & & \\ & & \ddots & & \\ & & & C & D \\ & & & & 0 \end{bmatrix} \quad b_{ineq} = \begin{bmatrix} \delta \\ \vdots \\ \delta \\ 0 \end{bmatrix}$$

Finally the standard formulation for the optimization problem is:

$$\begin{aligned} \min_{Z} \quad & J(Z) = \frac{1}{2} Z^\top H Z \\ \text{subject to} \quad & A_{eq} Z = b_{eq} \\ & A_{ineq} Z \leq b_{ineq} \end{aligned} \tag{5.2}$$

where matrix H is defined by rearranging properly the weight matrices Q and R used in (2.18) as it is shown in the following:

$$H = \begin{bmatrix} 0 & & & & & \\ & R & & & & \\ & & Q & & & \\ & & & R & & \\ & & & & \ddots & \\ & & & & & \overline{Q} \end{bmatrix}$$

It is also possible to take into account in the cost function the so-called cross term matrix S, which penalize, in the cost function, also the hybrid term $x^\top S u$. In particular, in the latter case, the matrix H takes the form

$$H = \begin{bmatrix} 0 & & & & & & & \\ & R & & & & & & \\ & & Q & S & & & & \\ & & S^\top & R & & & & \\ & & & & \ddots & & & \\ & & & & & Q & S & \\ & & & & & S^\top & R & \\ & & & & & & & \overline{Q} \end{bmatrix}$$

The MPC problem has been formulated as a quadratic problem with optimization variable Z. The same procedure can be applied to the Time-LBT formulation (Section 2.3.2) in order to obtain a similar quadratic problem as (5.2), which only differs in how the equality constraints are expressed (i.e., the variables are stacked differently in this state space representation).

5.2 Interior Point Methods (IPM)

The problem (5.2) is a convex quadratic problem and, due to the presence of the inequality constraints, an explicit solution does not exist. In this section we introduce a family of numerical methods which aim to solve a convex optimization problem with linear equality and inequality constraints which reduces it to a sequence of linear equality constrained problems, [12]. In particular the *Mehrotra's Predictor Corrector algorithm* will be described.

The Lagrangian function can be defined for a generic optimization problem as the sum of different contributions, i.e.

$$\mathcal{L} = (\textit{cost function}) + p^\top (\textit{eq. constr.}) + \lambda^\top (\textit{ineq. constr.})$$

where the vectors p and λ are the Lagrangian multipliers of the equality and inequality constraints, respectively. Specifically, considering the optimization problem (5.2), the Lagrangian function is

$$\mathcal{L} = \frac{1}{2} Z^\top H Z + p^\top \left(A_{eq} Z - b_{eq}\right) + \lambda^\top \left(A_{ineq} Z - b_{ineq}\right)$$

The Karush-Kuhn-Tucker (KKT) conditions for the optimization problem are obtained by setting to zero the first derivative of the Lagrangian function, with respect to Z, p, and λ [12]. We obtain in this way

$$\begin{aligned}
&HZ + A_{eq}^\top p + A_{ineq}^\top \lambda = 0 \\
&-A_{eq} Z + b_{eq} = 0 \\
&-A_{ineq} Z + b_{ineq} \geq 0 \\
&\lambda \geq 0 \\
&\lambda_j (-A_{ineq} Z + b_{ineq})_j = 0 \qquad j = 1, \ldots, n_{ineq}
\end{aligned}$$

where $(\cdot)_j$ denotes the j-th element of a vector, and n_{ineq} is the number of rows of A_{ineq}, i.e. the total number of inequality constraints.

In order to convert the inequality constraints into equality constraints, a *slack* variable t is introduced and the KKT conditions can be rewritten as:

$$F(Z,p,\lambda,t) = \begin{bmatrix} HZ + A_{eq}^\top p + A_{ineq}^\top \lambda \\ -A_{eq}Z + b_{eq} \\ -A_{ineq}Z - t + b_{ineq} \\ T\Lambda e \end{bmatrix} = 0 \tag{5.3a}$$

$$(\lambda, t) \geq 0 \tag{5.3b}$$

where T and Λ are diagonal matrices defined by

$$T = \mathrm{diag}(t_1, t_2, \dots, t_{n_{ineq}}), \qquad \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \dots, \lambda_{n_{ineq}})$$

and e is a vector of elements equal to 1, used for dimensional consistency

$$e = (1, 1, \dots, 1)^\top.$$

In view of the fact that the KKT conditions are satisfied only in the optimal point $(Z^*, p^*, \lambda^*, t^*)$, the Interior Point Method is used to find the optimal solution for the non linear system of equations (5.3a) with condition (5.3b). More specifically, it makes use of a sequence of Newton's steps, each involving a linearization of (5.3a).

At iteration τ, computing the Jacobian of (5.3a), the linearized system around the point $(Z^{\tau-1}, p^{\tau-1}, \lambda^{\tau-1}, t^{\tau-1})$ is given by:

$$\begin{bmatrix} H & A_{eq}^\top & A_{ineq}^\top & 0 \\ -A_{eq} & 0 & 0 & 0 \\ -A_{ineq} & 0 & 0 & -I \\ 0 & 0 & T & \Lambda \end{bmatrix} \begin{bmatrix} \Delta Z \\ \Delta p \\ \Delta \lambda \\ \Delta t \end{bmatrix} = \begin{bmatrix} r^H \\ r^{A_{eq}} \\ r^{A_{ineq}} \\ r^T \end{bmatrix} \tag{5.4}$$

The right hand terms of the system (5.4) will be defined according to the *Merhotra's predictor corrector algortihm.* This will be explained in the following sections. Given the generic Newton's iteration, denoted by the superscript

τ, the so-called search directions $(\Delta Z^{\tau}, \Delta p^{\tau}, \Delta\lambda^{\tau}, \Delta t^{\tau})$ are calculated solving (5.4). The vector of the decision variables is updated according to

$$(Z^{\tau}, p^{\tau}, \lambda^{\tau}, t^{\tau}) = (Z^{\tau-1}, p^{\tau-1}, \lambda^{\tau-1}, t^{\tau-1}) + (\Delta Z^{\tau}, \Delta p^{\tau}, \Delta\lambda^{\tau}, \Delta t^{\tau})$$

If the problem is feasible, the interior point method allows to approach the optimal solution $(Z^{*}, p^{*}, \lambda^{*}, t^{*})$ in a certain number of iterations and with a certain accuracy.

However, a full step along the Newton's direction is usually not permitted, since it may violate the bound $(\lambda, t) \geq 0$. Due to this feasibility issue, the steplength applied to the vector $(Z^{\tau}, p^{\tau}, \lambda^{\tau}, t^{\tau})$ is scaled using a parameter $\alpha \in (0, 1]$:

$$(Z^{\tau}, p^{\tau}, \lambda^{\tau}, t^{\tau}) = (Z^{\tau-1}, p^{\tau-1}, \lambda^{\tau-1}, t^{\tau-1}) + \alpha(\Delta Z^{\tau}, \Delta p^{\tau}, \Delta\lambda^{\tau}, \Delta t^{\tau})$$

Unfortunately, using the pure Newton's directions, the allowed steplength ensuring the satisfaciton of the condition $(\lambda, t) \geq 0$, may be very small $(\alpha \ll 1)$ which slows down dramatically the convergence speed of the algorithm.

There are different methods allowing to modify the basic Newton procedure to obtain higher performances. Typically these methods constrain the search direction to be towards the interior of the nonnegative orthant $(\lambda, t) \geq 0$, so that it would be possible to move further along the growth direction before one of the component of (λ, t) becomes negative. They also keep (λ, t) far enough from the boundary of the nonnegative orthant when necessary, because the directions computed from the starting point too close to the boundary usually result distorted. The method used in this Thesis to improve the performances of the basic Newton method is the so-called *Mehrotra's predictor corrector algorithm* [13].

5.2.1 Mehrotra's Predictor Corrector Algorithm

The *Mehrotra's Predictor-Corrector algorithm* splits the pure Newton's step defined by (5.4) in two parts: the *affine-scaling step* and the *centering-corrector* *step.* This algorithm has proved to be one of the most effective approaches for general linear and convex quadratic programs [2].

Affine-scaling Step

In this first step of the Mehrotra's algorithm, the right hand side terms of the system (5.4) are defined as follows, considering the solutions of the previous Newton's iteration:

$$\begin{bmatrix} r^H \\ r^{A_{eq}} \\ r^{A_{ineq}} \\ r^T \end{bmatrix} = -F(Z^{\tau-1}, p^{\tau-1}, \lambda^{\tau-1}, t^{\tau-1}) = \\ - \begin{bmatrix} HZ^{\tau-1} + A_{eq}^\top p^{\tau-1} + A_{ineq}^\top \lambda^{\tau-1} \\ -A_{eq}Z^{\tau-1} + b_{eq} \\ -A_{ineq}Z^{\tau-1} - t^{\tau-1} + b_{ineq} \\ T^{\tau-1}\Lambda^{\tau-1}e \end{bmatrix} \tag{5.5}$$

The corresponding search directions obtained by solving the system (5.4), are denoted by:

$$(\Delta Z_{\text{aff}}, \Delta p_{\text{aff}}, \Delta \lambda_{\text{aff}}, \Delta t_{\text{aff}},)$$

note that we have dropped the superscript τ, meaning that from now all the variables are intended to be related to the current Newton's iteration.

Centering-corrector Step

The corrector step is needed to move the point closer to the central path. In this part of the algorithm the right hand terms of (5.4) are defined as:

$$\begin{bmatrix} r^H \\ r^{A_{eq}} \\ r^{A_{ineq}} \\ r^T \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\Delta T_{\text{aff}} \Delta \Lambda_{\text{aff}} e + \sigma \mu e \end{bmatrix} \tag{5.6}$$

where

$$\Delta T_{\text{aff}} = \text{diag}(\Delta t_{\text{aff},1}, \Delta t_{\text{aff},2}, \dots, \Delta t_{\text{aff},n_{ineq}})$$

$$\Delta \Lambda_{\text{aff}} = \text{diag}(\Delta \lambda_{\text{aff}1}, \Delta \lambda_{\text{aff}2}, \dots, \Delta \lambda_{\text{aff},n_{ineq}})$$

σ is the *centering parameter* and μ is the *Duality gap.* The Duality gap is by definition the difference between the primal and dual solution: it is a parameter used to check the optimality of the current solution $(Z^\tau, p^\tau, \lambda^\tau, t^\tau)$. In our case it is given by

$$\mu = \lambda^\top t / n_{ineq}$$

Defining now the maximum steplength α_{aff} that can be taken along the affine-scaling direction as

$$\alpha_{\text{aff}} = \text{arg max}\big(\alpha \in [0,1] | (\lambda, t) + \alpha(\Delta\lambda_{\text{aff}}, \Delta t_{\text{aff}}) \geq 0\big)$$

and the duality gap μ_{aff} obtained from the affine-scaling step as

$$\mu_{\text{aff}} = (\lambda + \alpha_{\text{aff}} \Delta\lambda_{\text{aff}})^\top (t + \alpha_{\text{aff}} \Delta t_{\text{aff}})/m$$

it is shown by [2] and [14] that a good way to define the *centering parameter* is given by the following heuristic:

$$\sigma = \big(\mu_{\text{aff}}/\mu\big)^3$$

The solutions of the *Centering-corrector step* are defined as:

$$(\Delta Z_{\text{cent}}, \Delta p_{\text{cent}}, \Delta\lambda_{\text{cent}}, \Delta t_{\text{cent}})$$

Overall procedure

The search directions are obtained summing the solutions of the *Affine-scaling step* with the solutions of the *Centering-corrector step*:

$$(\Delta Z, \Delta p, \Delta\lambda, \Delta t) = (\Delta Z_{\text{aff}}, \Delta p_{\text{aff}}, \Delta\lambda_{\text{aff}}, \Delta t_{\text{aff}}) + (\Delta Z_{\text{cent}}, \Delta p_{\text{cent}}, \Delta\lambda_{\text{cent}}, \Delta t_{\text{cent}},)$$

As done for the *Affine-scaling step* we select, at first, the coefficient α_{max} which represents the maximum step that can be taken without violating the condition (5.3b):

$$\alpha_{\text{max}} = \text{arg max}\big(\alpha \in [0,1] | (\lambda, s) + \alpha(\Delta\lambda, \Delta t) \geq 0\big)$$

Then, selecting a parameter $\gamma \in (0,1)$, $\alpha_{\max}$ is further reduced to enforce a smooth convergence to the optimal solution:

$$\alpha = \alpha_{\max}\gamma$$

See [14] for the details on a heuristic choice of γ.

Eventually, we can update the variables for the next IPM's iteration as follows:

$$\begin{aligned}
Z^{\tau} &= Z^{\tau-1} + \alpha\Delta Z^{\tau} \\
p^{\tau} &= p^{\tau-1} + \alpha\Delta p^{\tau} \\
\lambda^{\tau} &= \lambda^{\tau-1} + \alpha\Delta\lambda^{\tau} \\
t^{\tau} &= t^{\tau-1} + \alpha\Delta t^{\tau}
\end{aligned}$$

5.2.2 Convergence Analysis

Let us consider ξ as the generic decision variable for simplicity, defined as

$$\xi = \begin{bmatrix} Z \\ p \\ \lambda \\ t \end{bmatrix}$$

In $\tau = 1$, the so-called *initial guess* ξ^0 has to be properly defined. If the problem is feasible, as the number of iterations τ increases, the vector of the search directions $\Delta\xi$ will approach zero and ξ converges to the optimal solution ξ^*. However, from a numerical point of view, $\Delta\xi$ will never be exactly zero, due to the approximation introduced by the Newton's method. The main idea of the *inexact Newton's methods* is to terminate, with a suitable stopping condition, the iterative linear solver early with less accuracy, reducing the computational effort, [15].

Moreover, it is worth remarking that the algorithm does not require to specify a feasible starting point. Indeed, it may generate infeasible iterates, attaining feasibility only in the limit, [2], as shown in Figure 5.1. A trajectory in the state space is shown, where the variables, starting from a generic point, converge to a solution being very close to the optimal one as mentioned. The solution lies in the feasible region, bounded by the box, which represents the collection of points into the surface that satisfy the constraints.

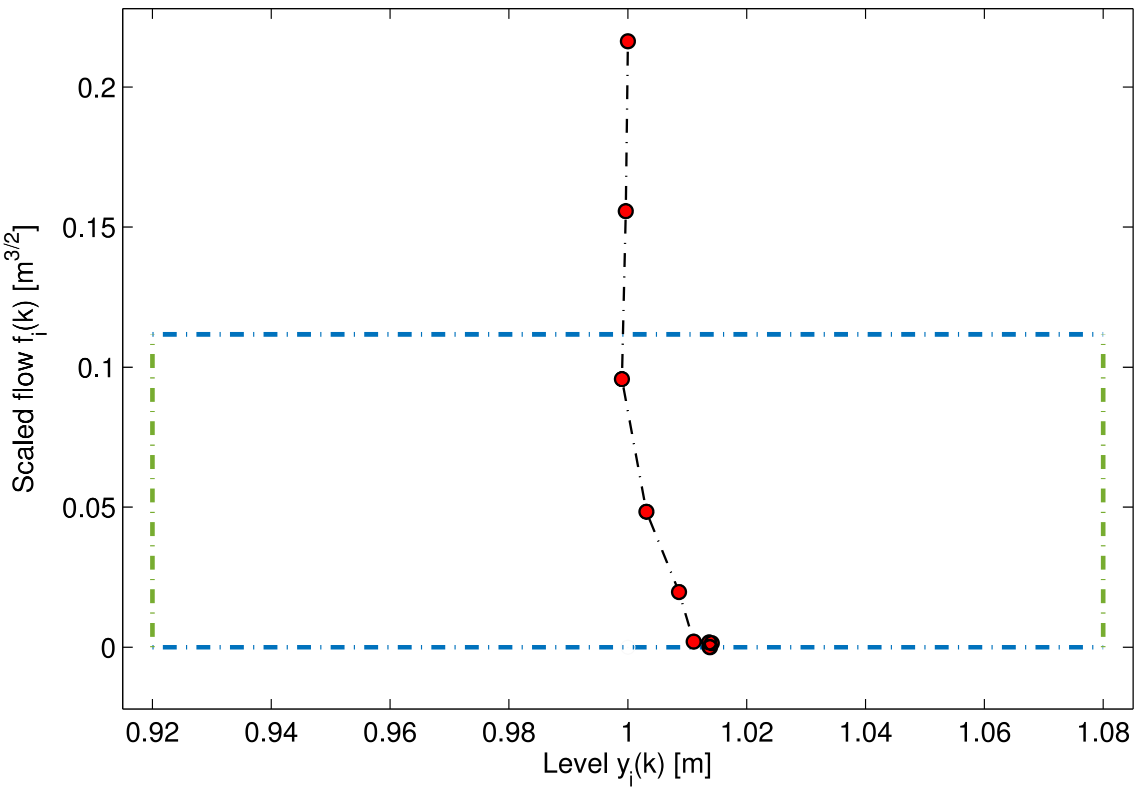


Figure 5.1: Convergence of the solution

5.2.3 Complexity and Separability

The *Interior Point Method* shown in the previous section can be used to solve MPC problems in presence of constraints on inputs and states of type (5.2).

Note that the main idea of the IPM is to solve, at each iteration, linear equations of type (5.4), involving a matrix inversion. However, matrix inversions may be inefficient from the computational point of view, especially when the size of the large-scale system and/or the length of the prediction horizon grow. More specifically, the submatrices H, A_{eq} and A_{ineq} grow in dimension with the prediction horizon T and the number of subsystems N. The algorithms implemented on standard solvers typically lead to a complexity which scales as $\mathcal{O}(T^3N^3)$ and their application tends to be significantly burdensome or even impractical for large-scale systems.

Significant improvements can be made by exploiting the multistaged structure of such optimization problem (see (2.20)). In particular a discrete-time Riccati recursion can be used to solve the linear system of equations efficiently at each iteration of the Interior Point Method. This so-called *block factorization approach* consists of reducing the optimization problem to a collection of simpler problem to be solved with a complexity that scales as $\mathcal{O}(TN^3)$ (as discussed in [2]).

On the other hand, note that the minimization problem (2.26) takes the same form as (2.20). The only difference lies in the fact that the equality constraints are defined differently. However, instead of the temporal partition of the interconnected system, one can apply the dynamic programming principle on a subsystem basis, [3]. This is possible in view of the fact that the plant belongs to the particular class of cascade systems, which has a specific LBT structure. Note that the approach that will be proposed relies on more than just the cascade structure, since the separability across both time and space of the cost function and the constraints of the original problem is needed for the Time-LBT formulation. These properties allow us to formulate differently-structured IPM problems for the efficient design and implementation of the optimal controller. More specifically, given the Time-LBT formulation introduced in Section 2.3.2, the associated quadratic program (2.26) is multistaged, with as many stages as the number of subsystems N. It will be possible to formulate a problem with a complexity that scales as $\mathcal{O}(T^3N)$.

The former solutions are described in the next Chapter 6.

Chapter 6

Proposed solvers for LBT systems

In this chapter, two algorithms that solve efficiently a class of quadratic programming problems are presented. More specifically, we address an MPC optimization problem applied to a large-scale cascade system. The two algorithms discussed in this chapter exploit the spatial and temporal interconnections of such systems. In particular, they take advantage of either the Space-LBT formulation or the Time-LBT one, introduced in Section 2.3.1 and in Section 2.3.2, respectively.

Considering the linear system of equations (5.4) obtained applying the IPM to a QP problem, such system, for both problems (2.20) and (2.26), is structured. In particular, the structure is block-diagonal and it has a strong dependence on the prediction horizon T and the number of subsystems N.

6.1 Space-LBT Solver

The first algorithm proposed evolves along the line of paper [2] and it relies on the Space-LBT formulation. More specifically, the dynamic equation (2.14) is recalled:

$$X(k+1) = AX(k) + BU(k) \tag{6.1}$$

A set of inequality constraints on states and inputs for each subsystem has been introduced. It is based on the inequalities (2.15) which can be rewritten in the more compact form (2.16), i.e.,

$$CX(k) + DU(k) \leq \delta \tag{6.2}$$

The weight matrices for the optimization problem have already been introduced in Section 2.3.1, but one might consider the presence of cross-penalty terms in order to relate state and input variables. In particular, the latter terms are defined by means of:

$$S = \begin{bmatrix} S_1 & & & \\ & S_2 & & \\ & & \ddots & \\ & & & S_N \end{bmatrix}$$

with $S_i \in \mathbb{R}^{n_i \times m_i}$.

Therefore, the cost function (2.18) becomes:

$$\begin{aligned} J\big(\{X(k)\}_{k=1}^{T}, \{U(k)\}_{k=0}^{T-1}\big) = \frac{1}{2}U(0)^\top RU(0) + \sum_{k=1}^{T-1} \frac{1}{2}\Big(X(k)^\top QX(k) \\ + U(k)^\top RU(k) + 2X(k)^\top SU(k)\Big) + \frac{1}{2}X(T)^\top \overline{Q}X(T) \end{aligned} \tag{6.3}$$

where Q, $\overline{Q} \in \mathbb{R}^{n\times n}$, $R \in \mathbb{R}^{m\times m}$ and $S \in \mathbb{R}^{n\times m}$. In particular, they are considered to be time invariant along the prediction horizon T, and they are built according to the stacking of the variables.

The minimization of the cost function (6.3) has to be performed with respect to the constraints related to the dynamics of the system (6.1) and to the inequality constraints (6.2). Consequently, the minimization problem can be written as:

$$\begin{aligned} \min_{X(k),U(k)} \quad & J\big(X(k),U(k)\big) && (6.4)\\ \text{subject to} \quad & X(0)=\widehat{X} \\ & X(k+1)=AX(k)+BU(k) \quad k=0\dots T-1 \\ & CX(k)+DU(k)\le\delta \qquad\qquad k=0\dots T-1 \end{aligned}$$

Note that we have explicitly formulate the constraint on the initial state, because it is accounted as a known term.

It is very important to acknowledge that this optimization problem, as it is formulated, relies inherently to separability of cost function and constraint across time. Moreover, the block diagonal structure of weight matrices and inequality constraint matrices is intentional, since it allow us to have separability across space as well.

6.1.1 KKT Conditions

In order to solve the optimization problem (6.4) one need at first to derive the KKT conditions. The Lagrangian function of the minimization problem is given by:

$$\mathcal{L}=\mathcal{L}_0+\sum_{k=1}^{T-1}\mathcal{L}_{\mathrm{k}}+\mathcal{L}_{\mathrm{T}}$$

The Lagrangian associated to the first and the last time instants ($\mathcal{L}_0$ and $\mathcal{L}_{\mathrm{T}}$) are slightly different from the one referred to $k=1,\cdots,T-1$, since cost function and constraints are different at $k=0$ and $k=T$. Let $p(k)$ and $\lambda(k)$ be the vectors of the Lagrangian multiplier referred to the equality and inequality constraints respectively, the elements composing the Lagrangian function take the following form.

First time instant:

$$\begin{aligned}\mathcal{L}_0=\frac{1}{2}U(0)^\top RU(0)+p(0)^\top\big(AX(0)+BU(0)-X(1)\big)\\ +\lambda(0)^\top\big(CX(0)+DU(0)-\delta\big)\end{aligned}$$

For $k = 1, \cdots, T-1$:

$$\mathcal{L}_{\mathrm{k}} = \frac{1}{2}\Big(X(k)^\top Q X(k) + U(k)^\top R U(k) + 2X(k)^\top S U(k)\Big) \\ + p(k)^\top\big(AX(k) + BU(k) - X(k+1)\big) \\ + \lambda(k)^\top\big(CX(k) + DU(k) - \delta\big)$$

Last time instant:

$$\mathcal{L}_{\mathrm{T}} = \frac{1}{2}X(T)^\top \overline{Q} X(T)$$

The following KKT conditions can be obtained by setting to zero the first derivative of the Lagrangian function $\mathcal{L}$ with respect to the decision variables $X(k)$, $U(k)$, $p(k)$ and $\lambda(k)$:

$$\text{First Stage} \quad \begin{cases} \frac{\partial \mathcal{L}}{\partial X(0)} = 0 & \big(\textit{X(0) is given}\big) \\ \frac{\partial \mathcal{L}}{\partial U(0)} = RU(0) + B^\top p(0) + D^\top \lambda(0) = 0 \\ \frac{\partial \mathcal{L}}{\partial p(0)} = AX(0) + BU(0) - X(1) = 0 \\ \frac{\partial \mathcal{L}}{\partial \lambda(0)} = CX(0) + DU(0) - \delta + t(0) = 0 \\ \Lambda(0)\Theta(0)e = 0 \end{cases} \\ (\lambda(0), t(0)) \geq 0 \tag{6.5}$$

$$\text{Generic Stage} \quad \begin{cases} \frac{\partial \mathcal{L}}{\partial X(k)} = QX(k) + SU(k) + A^\top p(k) + C^\top \lambda(k) - p(k-1) = 0 \\ \frac{\partial \mathcal{L}}{\partial U(k)} = RU(k) + S^\top X(k) + B^\top p(k) + D^\top \lambda(k) = 0 \\ \frac{\partial \mathcal{L}}{\partial p(k)} = AX(k) + BU(k) - X(k+1) = 0 \\ \frac{\partial \mathcal{L}}{\partial \lambda(k)} = CX(k) + DU(k) - \delta + t(k) = 0 \\ \Lambda(k)\Theta(k)e = 0 \end{cases} \\ (\lambda(k), t(k)) \geq 0 \tag{6.6}$$

$$\text{Last Stage} \quad \begin{cases} \frac{\partial \mathcal{L}}{\partial X(T)} = \overline{Q}X(T) - p(T-1) = 0 \end{cases}$$

A slack variable $t(k)$ has been introduced in order to convert the inequality constraints into equality constraints, and the following relationship must hold:

$$\begin{aligned} & t(k) = -CX(k) - DU(k) + \delta && k = 0, \ldots, T-1 \\ & \lambda_j(k) t_j(k) = 0 && k = 0, \ldots, T-1 \qquad j = 1, \ldots, n_{in} \end{aligned}$$

The matrices $\Theta(k)$ and $\Lambda(k)$ are diagonal and they contain the Lagrangian multipliers referred to the inequality constraints and the corresponding slack variables:

$$\Lambda(k) = \begin{bmatrix} \lambda_1(k) & & & \\ & \lambda_2(k) & & \\ & & \ddots & \\ & & & \lambda_{n_{in}}(k) \end{bmatrix} \quad \Theta(k) = \begin{bmatrix} t_1(k) & & & \\ & t_2(k) & & \\ & & \ddots & \\ & & & t_{n_{in}}(k) \end{bmatrix}$$

while e is a vector containing only elements equal to one, in order to keep the dimensions consistent.

6.1.2 Solver

The optimal values of the decision variables $X(k)$, $U(k)$, $p(k)$, $\lambda(k)$ and $t(k)$ have to satisfy the KKT conditions above. In particular, such values will be calculated by means of the numerical iterative method described in Section 5.2. This method consists of a sequence of Newton's steps which solve efficiently a linear system of equations. More specifically, according to the *Mehrotra's predictor corrector algorithm*, each Newton's iteration will be defined as the sequence of an *Affine* and a *Centering* step. However, the KKT conditions are not linear due to the presence of the equations $\Lambda(k)\Theta(k)e = 0$. Therefore, a linearization has to be carried out, and the resulting linear system is the following:

$$
\begin{bmatrix}
R & B^\top & D & 0 & & & & & & & \\
B & 0 & 0 & 0 & -I & & & & & & \\
D & 0 & 0 & I & & & & & & & \\
0 & 0 & \Theta(0) & \Lambda(0) & & & & & & & \\
 & -I & & & Q & S & A^\top & C^\top & 0 & & \\
 & & & & S^\top & R & B^\top & D^\top & 0 & & \\
 & & & & A & B & 0 & 0 & 0 & -I & \\
 & & & & C & D & 0 & 0 & I & & \\
 & & & & 0 & 0 & 0 & \Theta(1) & \Lambda(1) & & \\
 & & & & & & -I & & & \ddots & \\
 & & & & & & & & & & \overline{Q}
\end{bmatrix}
\begin{bmatrix}
\Delta U(0) \\ \Delta p(0) \\ \Delta\lambda(0) \\ \Delta t(0) \\ \Delta X(1) \\ \Delta U(1) \\ \Delta p(1) \\ \Delta\lambda(1) \\ \Delta t(1) \\ \vdots \\ \Delta X(T)
\end{bmatrix}
=
\begin{bmatrix}
r^R(0) \\ r^A(0) \\ r^C(0) \\ r^S(0) \\ r^Q(1) \\ r^R(1) \\ r^A(1) \\ r^C(1) \\ r^S(1) \\ \vdots \\ r^Q(T)
\end{bmatrix}
\tag{6.7}
$$

The Δ terms are the so-called *search directions* for the interior point method, and they represent the increment to apply to the decision variables which lead iteratively to the optimal solution, while the definition of the right-hand terms will be explained in the following.

The stages are arranged blockdiagonal-wise in order to be consistent with the vector of the decision variables, which are interleaved according to the stage index (i.e., the time instant). Indeed, (6.7) is structured in T stages and each stage is connected with both the previous and the next one by means of coupling terms. The strength of this algorithm is in how the system is solved: in fact, as discussed in Section 5.2.3, a recursive law will be applied in order to take advantage of the great sparsity. This will result in a significant increase on efficiency, especially when the number of stages is much higher than the dimension of the generic stage (i.e., $T \gg N$).

Affine-scaling step

As discussed in Section 5.2.1, the right-hand terms of the system (6.7) have a suitable definition for the Affine-scaling step of the Mehrotra's algorithm.

More specifically, for the generic stage they are:

$$\begin{bmatrix} r^R(k) \\ r^A(k) \\ r^C(k) \\ r^S(k) \\ r^Q(k+1) \end{bmatrix} = \begin{bmatrix} -\big(S^\top X(k) + RU(k) + B^\top p(k) + D^\top \lambda(k)\big) \\ -\big(AX(k) + BU(k) - X(k+1)\big) \\ -\big(CX(k) + DU(k) - \delta + t(k)\big) \\ -\big(\Lambda(k)\Theta(k)e(k)\big) \\ -\big(QX(k+1) + SU(k+1) + A^\top p(k+1) + C^\top \lambda(k+1) - Ip(k)\big) \end{bmatrix} \tag{6.8}$$

The algorithm involves the block-elimination of every stage, starting from the last, into the previous one. In order to obtain a more compact structure of (6.7), it is convenient to eliminate certain search directions. At first, the slackness variables Δt are eliminated. Considering the generic stage k, the last two equations state:

$$\Theta(k)\Delta\lambda(k) + \Lambda(k)\Delta t(k) = r^S(k) \tag{6.9}$$

$$C\Delta X(k) + D\Delta U(k) + \Delta t(k) = r^C(k) \tag{6.10}$$

solving (6.9) for the slackness variable and substituting its expression into (6.10), it is easy to obtain

$$C\Delta X(k) + D\Delta U(k) + \Lambda^{-1}(k)\big(r^S(k) - \Theta(k)\Delta\lambda(k)\big) = r^C(k) \tag{6.11}$$

The new terms $Z(k) = -\Lambda^{-1}(k)\Theta(k)$ and $r^Z(k) = r^C(k) - \Lambda^{-1}(k)r^S(k)$ are introduced and therefore, equation (6.11) becomes

$$C\Delta X(k) + D\Delta U(k) + Z(k)\lambda(k) = r^Z(k) \tag{6.12}$$

Due to the conditions (6.5) and (6.6), the matrices $\Lambda(k)$ and $\Theta(k)$ are positive semidefinite, and therefore invertible. On the other hand, it is possible to invert those matrices only far enough from the optimal solution because, in that point, the KKT conditions have to be satisfied and thus $\Lambda(k)\Theta(k) = 0$ has to hold. This is the reason why in Section 5.2.2 the concept of *inexact Newton's method* was introduced, meaning that matrices $\Lambda(k)$ and $\Theta(k)$ should never be null (or values too close to zero) otherwise it would be impossible to invert them. We will see that the algorithm has to stop to iterate before reaching the theoretical optimal solution, resulting in a less accurate (but still acceptable) one.

Regarding the generic k-th stage, the term $\Delta t(k)$ has been eliminated. In particular, the system (6.7) takes the more compact form:

$$\begin{bmatrix} \ddots & & & & & & \\ -I & \cdots & Q & S & A^\top & C^\top & \\ & & S^\top & R & B^\top & D^\top & \\ & & A & B & 0 & 0 & -I \\ & & C & D & 0 & Z(k) & \\ & & & & & & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ \Delta p(k-1) \\ \vdots \\ \Delta X(k) \\ \Delta U(k) \\ \Delta p(k) \\ \Delta\lambda(k) \\ \Delta X(k+1) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ r^Q(k) \\ r^R(k) \\ r^A(k) \\ r^Z(k) \\ \vdots \end{bmatrix} \tag{6.13}$$

With the same procedure, the terms $\Delta\lambda$ may be eliminated as well. Specifically, solving (6.12) for $\Delta\lambda(k)$, follows that:

$$\Delta\lambda(k) = Z^{-1}(k)\big(r^Z(k) - C\Delta X(k) - D\Delta U(k)\big)$$

where $Z(k)^{-1} = -\Lambda(k)\Theta^{-1}(k)$. The term $\Delta\lambda(k)$ can be substituted only into the first two equations of (6.13), and they become:

$$\begin{aligned} &Q\Delta X(k) + S\Delta U(k) + A^\top \Delta p(k) \\ &\qquad + C^\top Z^{-1}(k)\big(r^Z(k) - C\Delta X(k) - D\Delta U(k)\big) = r(k)^Q \end{aligned} \tag{6.14}$$

$$\begin{aligned} &S^\top\Delta X(k) + R\Delta U(k) + B^\top \Delta p(k) \\ &\qquad + D^\top Z^{-1}(k)\big(r^Z(k) - C\Delta X(k) - D\Delta U(k)\big) = r(k)^R \end{aligned} \tag{6.15}$$

After few calculations, defining the matrices

$$\begin{aligned} \widetilde{Q}(k) &= Q - C^\top Z^{-1}(k)C \\ \widetilde{S}(k) &= S - C^\top Z^{-1}(k)D \\ \widetilde{S}(k)^\top &= S^\top - D^\top Z^{-1}(k)C \\ \widetilde{R}(k) &= R - D^\top Z^{-1}(k)D \\ \widetilde{r^Q}(k) &= r^Q(k) - C^\top Z^{-1}(k)r^Z(k) \\ \widetilde{r^R}(k) &= r^R(k) - D^\top Z^{-1}(k)r^Z(k) \end{aligned}$$

the equations (6.14) and (6.15) take the form

$$\widetilde{Q}\Delta X(k) + \widetilde{S}\Delta U(k) + A^{\top}\Delta p(k) = \widetilde{r}^{Q}(k) \tag{6.16a}$$

$$\widetilde{S}^{\top}\Delta X(k) + \widetilde{R}\Delta U(k) + B^{\top}\Delta p(k) = \widetilde{r}^{R}(k) \tag{6.16b}$$

Eventually, after the elimination of the Lagrangian multipliers of the inequality constraints and their slack variables, the linear system (6.7) takes the more compact form:

$$\begin{bmatrix} \widetilde{R}(0) & B^{\top} & & & & & & \\ B & 0 & -I & & & & & \\ & -I & \widetilde{Q}(1) & \widetilde{S}(1) & A^{\top} & & & \\ & & \widetilde{S}^{\top}(1) & \widetilde{R}(1) & B^{\top} & & & \\ & & A & B & 0 & -I & & \\ & & & & -I & \ddots & & \\ & & & & & & & \overline{Q} \end{bmatrix} \begin{bmatrix} \Delta U(0) \\ \Delta p(0) \\ \Delta X(1) \\ \Delta U(1) \\ \Delta p(1) \\ \Delta X(2) \\ \vdots \\ \Delta X(T) \end{bmatrix} = \begin{bmatrix} \widetilde{r}^{R}(0) \\ r^{A}(0) \\ \widetilde{r}^{Q}(1) \\ \widetilde{r}^{R}(1) \\ r^{A}(1) \\ \widetilde{r}^{Q}(2) \\ \vdots \\ r^{Q}(T) \end{bmatrix} \tag{6.17}$$

Note that, the weight matrices Q, $\overline{Q}$, R and S has been assumed to be time-invariant, but, due to the elimination of the variables Δt and $\Delta\lambda$, the matrices $\widetilde{Q}(k)$, $\widetilde{R}(k)$ and $\widetilde{S}(k)$ are time-varying.

Only the last stage remained unchanged, since it is only composed by the equation $-I\Delta p(T-1) + \overline{Q}\Delta X(T) = r^{Q}(T)$. This is the starting point for the recursive algorithm. Basically, the aim is to block-eliminate every stages into the previous one with a procedure similar to the elimination of the variables Δt and $\Delta\lambda$. Such procedure is called *backward recursion*, and it ends when all the stages are block-reduced into the first one. Afterwards, starting from the first stage, all the search directions Δ will be found by means of the so-called *forward recursion*.

In the following, the backward recursion is explained in detail. Let consider the T-th and the $(T-1)$-th stage:

$$\begin{bmatrix} \ddots & & & & \\ -I & \widetilde{Q}(T-1) & \widetilde{S}(T-1) & A^\top & \\ & \widetilde{S}^\top(T-1) & \widetilde{R}(T-1) & B^\top & \\ & A & B & 0 & -I \\ & & & -I & \Pi(T) \end{bmatrix} \begin{bmatrix} \vdots \\ \Delta p(T-2) \\ \Delta X(T-1) \\ \Delta U(T-1) \\ \Delta p(T-1) \\ \Delta X(T) \end{bmatrix} = \begin{bmatrix} \vdots \\ \widetilde{r}^Q(T-1) \\ \widetilde{r}^R(T-1) \\ r^A(T-1) \\ \pi(T) \end{bmatrix} \tag{6.18}$$

Regarding the equation of the last stage, the weight matrix and the corresponding right-hand term have been renamed as $\overline{Q} = \Pi(T)$ and $r^Q(T) = \pi(T)$ respectively. As a result, the following expression for $\Delta X(T)$ holds:

$$\Delta X(T) = \Pi^{-1}(T)\big(\pi(T) + \Delta p(T-1)\big) \tag{6.19}$$

The block-elimination of the last stage into the previous one can be simply done by substituting $\Delta X(T)$ into the last equation of the $(T-1)$-th stage, and it results in this:

$$A\Delta X(T-1) + B\Delta U(T-1) - \Pi^{-1}(T)\Delta p(T-1) = r^\pi(T-1) \tag{6.20}$$

where

$$r^\pi(T-1) = r^A(T-1) + \Pi^{-1}(T)\pi(T)$$

The linear system (6.18) becomes:

$$\begin{bmatrix} \ddots & & & \\ -I & \widetilde{Q}(T-1) & \widetilde{S}(T-1) & A^\top \\ & \widetilde{S}^\top(T-1) & \widetilde{R}(T-1) & B^\top \\ & A & B & -\Pi^{-1}(T) \end{bmatrix} \begin{bmatrix} \vdots \\ \Delta p(T-2) \\ \Delta X(T-1) \\ \Delta U(T-1) \\ \Delta p(T-1) \end{bmatrix} = \begin{bmatrix} \vdots \\ \widetilde{r}^Q(T-1) \\ \widetilde{r}^R(T-1) \\ r^\pi(T-1) \end{bmatrix} \tag{6.21}$$

The last block has been eliminated, but in order to find the recursive law it is necessary to eliminate also the $(T-1)$-th stage into the $(T-2)$-th one.

The first variable to cancel out is $\Delta p(T-1)$, that is expressed as:

$$\Delta p(T-1) = \Pi(T)\big[A\Delta X(T-1) + B\Delta U(T-1) - r^{\Pi}(T-1)\big]$$

Taking this expression and putting it into the first and the second equation of (6.21), it follows:

$$\begin{cases} -\Delta p(T-2) + \big[\widetilde{Q}(T-1) + A^{\top}\Pi(T)A\big]\Delta X(T-1) + \\ \qquad\qquad \big[\widetilde{S}(T-1) + A^{\top}\Pi(T)B\big]\Delta U(T-1) = \overline{r}^{Q}(T-1) \\ \big[\widetilde{R}(T-1) + B^{\top}\Pi(T)B\big]\Delta U(T-1) + \\ \qquad\qquad \big[\widetilde{S}(T-1)^{\top} + B^{\top}\Pi(T)A\big]\Delta X(T-1) = \overline{r}^{R}(T-1) \end{cases}$$

Defining the matrices

$$\begin{aligned} \overline{Q}(T-1) &= \big[\widetilde{Q}(T-1) + A^{\top}\Pi(T)A\big] \\ \overline{R}(T-1) &= \big[\widetilde{R}(T-1) + B^{\top}\Pi(T)B\big] \\ \overline{S}(T-1) &= \big[\widetilde{S}(T-1) + A^{\top}\Pi(T)B\big] \\ \overline{S}(T-1)^{\top} &= \big[\widetilde{S}(T-1)^{\top} + B^{\top}\Pi(T)A\big] \\ \overline{r}^{R}(T-1) &= \widetilde{r}^{R}(T-1) + B^{\top}\Pi(T)r^{\pi}(T-1) \\ \overline{r}^{Q}(T-1) &= \widetilde{r}^{Q}(T-1) + A^{\top}\Pi(T)r^{\pi}(T-1) \end{aligned}$$

the $(T-1)$-th stage (6.21) takes the following form:

$$\begin{bmatrix} \ddots & & & \\ -I & \overline{Q}(T-1) & \overline{S}(T-1) \\ & \overline{S}^{\top}(T-1) & \overline{R}(T-1) \end{bmatrix} \begin{bmatrix} \vdots \\ \Delta p(T-2) \\ \Delta X(T-1) \\ \Delta U(T-1) \end{bmatrix} = \begin{bmatrix} \vdots \\ \overline{r}^{Q}(T-1) \\ \overline{r}^{R}(T-1) \end{bmatrix} \tag{6.22}$$

Eventually, solving the second equation of (6.22) for $\Delta U(T-1)$, we obtain:

$$\Delta U(T-1) = \overline{R}^{-1}(T-1)\Big(\overline{r}^{R}(T-1) - \overline{S}(T-1)^{\top}\Delta X(T-1)\Big)$$

Inserting $\Delta U(T-1)$ into the first one and defining

$$\Pi(T-1) = \overline{Q}(T-1) - \overline{S}(T-1)\overline{R}^{-1}(T-1)\overline{S}(T-1)^{\top} \tag{6.23a}$$

$$\pi(T-1) = \overline{r}^Q(T-1) - \overline{S}(T-1)\overline{R}^{-1}(T-1)\overline{r}^R(T-1) \tag{6.23b}$$

it follows that

$$-\Delta p(T-2) + \Pi(T-1)\Delta X(T-1) = \pi(T-1) \tag{6.24}$$

The block-elimination of the $(T-1)$-th stage has been performed. More specifically, the system (6.18), can be written as:

$$\begin{bmatrix} \ddots & & & & \\ -I & \widetilde{Q}(T-2) & \widetilde{S}(T-2) & A^\top & \\ & \widetilde{S}^\top(T-2) & \widetilde{R}(T-2) & B^\top & \\ & A & B & 0 & -I \\ & & & -I & \Pi(T-1) \end{bmatrix} \begin{bmatrix} \vdots \\ \Delta p(T-3) \\ \Delta X(T-2) \\ \Delta U(T-2) \\ \Delta p(T-2) \\ \Delta X(T-1) \end{bmatrix} = \begin{bmatrix} \vdots \\ \widetilde{r}^Q(T-2) \\ \widetilde{r}^R(T-2) \\ r^A(T-2) \\ \pi(T-1) \end{bmatrix}$$

Highlighting the fact that expanding (6.23a), it becomes:

$$\Pi(k-1) = \left(\widetilde{Q}(k-1) + A^\top\Pi(k)A\right) - \left(\widetilde{S}(k-1) + A^\top\Pi(k)B\right) \left(\widetilde{R}(k-1) + B^\top\Pi(k)B\right)^{-1}\left(\widetilde{S}(k-1)^\top + B^\top\Pi(k)A\right) \tag{6.25}$$

Note that equation (6.25) is the well-known discrete-time Riccati equation for time-varying weighting matrices. Indeed, this block-elimination procedure can be applied to all the others upstream stages, until the first one is reached and the system is completely reduced, i.e.

$$\begin{bmatrix} \widetilde{R}(0) & B^\top & 0 \\ B & 0 & -I \\ 0 & -I & \Pi(1) \end{bmatrix} \begin{bmatrix} \Delta U(0) \\ \Delta p(0) \\ \Delta X(1) \end{bmatrix} = \begin{bmatrix} \widetilde{r}^R(0) \\ r^A(0) \\ \pi(1) \end{bmatrix}$$

After the backward recursive procedure it is possible to find the search directions $\Delta U(k)$, $\Delta p(k)$, $\Delta X(k)$, $\Delta\lambda(k)$ and $\Delta t(k)$, through the forward algorithm. Basically, by means of the relationships used before to apply the block-eliminations, one can restore every eliminated block finding the search directions. At first the equations of the first stage are solved and then all the others, with the equations resumed below:

$$\text{First stage (0)} \begin{cases} \Delta U(0) = \big(\widetilde{R}(0) + B^\top \Pi(1) B\big)^{-1} \\ \qquad\qquad \big(\widetilde{r}^R(0) + B^\top \Pi(1) r^A(0) + B^\top \pi(1)\big) \\ \Delta p(0) = \Pi(1) B \Delta U(0) - \Pi(1) r^A(0) - \pi(1) \\ \Delta X(1) = \Pi^{-1}(1)\Delta p(0) + \Pi^{-1}(1)\pi(1) \\ \Delta\lambda(0) = Z^{-1}(0)\big(r^Z(0) - D\Delta U(0)\big) \\ \Delta t(0) = \Lambda^{-1}(0)\big(r^S(0) - \Theta(0)\Delta\lambda(0)\big) \end{cases} \tag{6.26}$$

$$\text{Generic stage (k)} \begin{cases} \Delta U(k) = \big(\widetilde{R}(k) + B^\top \Pi(k+1) B\big)^{-1} \\ \qquad\qquad \big[\overline{r}^R(k) - \big(\widetilde{S}^\top(k) + B^\top \Pi(k) A\big)\Delta X(k)\big] \\ \Delta X(k+1) = A\Delta X(k) + B\Delta U(k) - r^A(k) \\ \Delta p(k) = \Pi(k+1)\Delta X(k+1) - \pi(k+1) \\ \Delta\lambda(k) = Z^{-1}(k)\big(r^Z(k) - C\Delta X(k) - D\Delta U(k)\big) \\ \Delta t(k) = \Lambda^{-1}(k)\big(r^S(k) - \Theta(k)\Delta\lambda(k)\big) \end{cases} \tag{6.27}$$

$$\text{Last stage (T)} \qquad \Delta X(T) = \Pi^{-1}(T)\big(\pi(T) + \Delta p(T-1)\big) \tag{6.28}$$

Regarding the forward procedure, one might notice that actually, solutions on a time instant k relies on $\Pi(k+1)$. For this reason, a proper initialization of the matrix $\Pi(T)$ is necessary.

Centering-corrector step

To complete a full iteration, a centering step is needed. The algorithm used to find the search directions in the centering phase is exactly the same used for the affine one. The only difference lies in the definition of the right-hand terms of the linear system (6.7). More specifically, they are defined as:

$$\begin{bmatrix} r^R(k) \\ r^A(k) \\ r^C(k) \\ r^S(k) \\ r^Q(k+1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\Delta\Theta_{\text{aff}}(k)\Delta\Lambda_{\text{aff}}(k)e + \sigma\mu e \\ 0 \end{bmatrix} \tag{6.29}$$

The terms $\Delta\Theta_{\text{aff}}$ and $\Delta\Lambda_{\text{aff}}$ are diagonal matrices containing all the $\Delta t_i(k)$ and $\Delta\lambda_i(k)$ respectively, which are obtained from the previous affine step (as detailed explained in Section 5.2). In order to compute the *duality gap*, all the terms $\lambda(k)$ and $t(k)$, obtained from the previous IPM's iteration $(\tau - 1)$, have to be stacked into two different vectors:

$$\boldsymbol{\lambda} = [\lambda(0)^\top, \lambda(1)^\top, \ldots, \lambda(T-1)^\top]^\top \tag{6.30}$$

$$\mathbf{t} = [t(0)^\top, t(1)^\top, \ldots, t(T-1)^\top]^\top \tag{6.31}$$

the *duality gap* is thus defined by:

$$\mu = \big(\boldsymbol{\lambda}^\top \mathbf{t}\big)/\nu \tag{6.32}$$

where $\nu = Tn_{in}$ is the total number of inequality constraints along the prediction horizon.

Once the *duality gap* parameter is defined, the *centering parameter* σ has to be calculated. Likewise to (6.30), all the affine directions related to the Lagrangian multipliers of the inequality constraints and the respective slack variables have to be stacked in vectors:

$$\Delta\boldsymbol{\lambda}_{\text{aff}} = [\Delta\lambda_{\text{aff}}(0)^\top, \Delta\lambda_{\text{aff}}(1)^\top, \ldots, \Delta\lambda_{\text{aff}}(T-1)^\top]^\top$$

$$\Delta\mathbf{t}_{\text{aff}} = [\Delta t_{\text{aff}}(0)^\top, \Delta t_{\text{aff}}(1)^\top, \ldots, \Delta t_{\text{aff}}(T-1)^\top]^\top$$

As explained in Section 5.2, the maximum affine steplength α_{aff} has to be found. In particular, it represents the maximum steplength that may be achieved along the affine-scaling direction. It is used to compute the affine duality gap μ_{aff} attained from this full step to the boundary:

$$\alpha_{\text{aff}} = \textit{argmax}\big[\alpha \in [0,1] \mid (\boldsymbol{\lambda}, \mathbf{t}) + \alpha(\Delta\boldsymbol{\lambda}_{\text{aff}}, \Delta\mathbf{t}_{\text{aff}}) \geq 0\big] \tag{6.33}$$

$$\mu_{\text{aff}} = \big[(\boldsymbol{\lambda} + \alpha_{\text{aff}}\Delta\boldsymbol{\lambda}_{\text{aff}})^\top(\mathbf{t} + \alpha_{\text{aff}}\Delta\mathbf{t}_{\text{aff}})\big]/\nu \tag{6.34}$$

Finally the centering parameter σ is selected as:

$$\sigma = (\mu_{\text{aff}}/\mu)^3 \tag{6.35}$$

At this point, both the duality gap and the centering parameters have been calculated so that it is possible to continue with the centering-corrector step. Henceforward, the procedure is identical to the affine-scaling step. Indeed, given the generic IPM iteration τ, the same algorithm has to be applied two times: one for the affine step and another one for the centering step, varying only the right-hand terms. Moreover, for each of these two steps, the solver has to perform both a backward recursion and a forward recursion procedure. The whole process is illustrated by Figure 6.1.

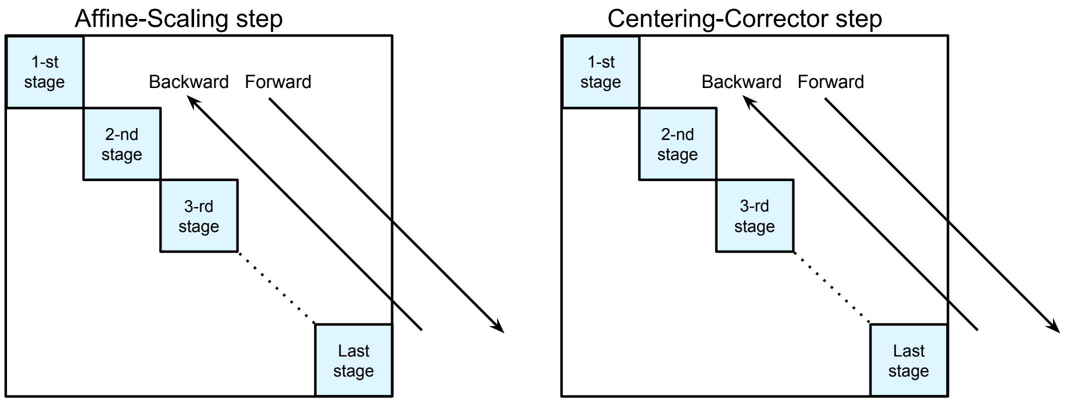


Figure 6.1: Affine and Centering steps

Maximum Steplenght

After the centering step, the generic IPM's iteration can be considered concluded. The total steplength is thus defined:

$$\begin{aligned}\Big(\Delta X(k), \Delta U(k), \Delta p(k), \Delta\lambda(k), \Delta t(k)\Big) &= \\ &\Big(\Delta X(k)_{\text{aff}}, \Delta U(k)_{\text{aff}}, \Delta p(k)_{\text{aff}}, \Delta\lambda(k)_{\text{aff}}, \Delta t(k)_{\text{aff}}\Big) + \\ &\Big(\Delta X(k)_{\text{cent}}, \Delta U(k)_{\text{cent}}, \Delta p(k)_{\text{cent}}, \Delta\lambda(k)_{\text{cent}}, \Delta t(k)_{\text{cent}}\Big) \qquad (6.36)\end{aligned}$$

However this steplength is further reduced of a factor α in order to avoid the violation of the conditions (6.5) and (6.6). More specifically, both the vectors containing the new $\Delta\lambda$ and Δt are created:

$$\begin{aligned}\Delta\boldsymbol{\lambda} &= [\Delta\lambda(0)^\top, \Delta\lambda(1)^\top, \dots, \Delta\lambda(T-1)^\top]^\top \\ \Delta\mathbf{t} &= [\Delta t(0)^\top, \Delta t(1)^\top, \dots, \Delta t(T-1)^\top]^\top\end{aligned}$$

A reducing coefficient $\alpha_{\max}$ is selected as:

$$\alpha_{\max} = \mathit{argmax}\big[\alpha \in [0,1] \mid (\boldsymbol{\lambda}, \mathbf{t}) + \alpha(\Delta\boldsymbol{\lambda}, \Delta\mathbf{t}) \geq 0\big] \qquad (6.37)$$

Eventually, setting the parameter $\gamma \in (0,1]$ (usually very close to 1) which scales $\alpha_{\max}$, the effective steplength reduction is defined by:

$$\alpha = \alpha_{\max}\gamma \qquad (6.38)$$

Let τ be the current IPM's step, we can update the vectors of the search directions for the next IPM's iteration through these equations:

$$\begin{aligned}X^{\tau+1}(k) &= X^{\tau}(k) + \alpha\Delta X(k) \\ U^{\tau+1}(k) &= U^{\tau}(k) + \alpha\Delta U(k) \\ p^{\tau+1}(k) &= p^{\tau}(k) + \alpha\Delta p(k) \qquad (6.39) \\ \lambda^{\tau+1}(k) &= \lambda^{\tau}(k) + \alpha\Delta\lambda(k) \\ t^{\tau+1}(k) &= t^{\tau}(k) + \alpha\Delta t(k)\end{aligned}$$

Stopping Condition

Since the optimization problem considered is convex, then the KKT conditions are both necessary and sufficient for optimality, so if the solution exits it is unique. With a feasible problem, we expect that the search directions become smaller at each iteration because the algorithm is converging to the optimal solution. In the optimal point this condition holds:

$$\lambda_j(k)t_j(k) = 0 \quad \forall j = 1, \dots, n_{in}\, k = 1, \dots, T$$

From a numerical point of view, since the *Mehrotra's algorithm* is an inexact IPM method, this condition is not reached in practice. Moreover, getting closer to the optimal solution, the matrices $\Lambda^{-1}(k)$ and $\Theta^{-1}(k)$ become ill conditioned. For these reasons a stopping condition has been enforced for the algorithm, which selects a solution that is close enough to the optimal one. One can use different types of stopping conditions. In our simulations we set a threshold on the terms λ and t

$$\boldsymbol{\lambda}^\top \mathbf{t} \leq \epsilon \tag{6.40}$$

such that while their product is bigger than $\epsilon = 10^{-5}$, the algorithm will continue to iterate.

6.1.3 Complexity

The aim of this section is to show how the aforementioned solver for the MPC control problem, reduces significantly its computational effort compared with a standard solver. The complexity in the resolution of optimization problems like (6.4) with classical methods scales cubically with the number of subsystems N and the prediction horizon T characterizing the system.

As discussed in [2], the presented algorithm exploits the structure of the linearized system (6.17) which is banded and its bandwidth is independent on T. However, it grows linearly with N. In fact the larger N is, the bigger the dimensions of the matrices of the generic stage are. Instead, the dimension of the system (i.e., number of stages) rises with the prediction horizon T.

In the backward recursion the computational effort grows cubically with N, owing to the inversion of the term $\Pi(k)$, as shown in (6.19). At the same time it grows linearly with T due to the $2T$ recursions (backward and forward) made at each IPM iteration. For these reasons the algorithm complexity is $\mathcal{O}(TN^3)$.

Algorithm 2 Space-LBT Solver

```
 1: INPUT:
        $A_i$, $B_i$, $E_i$, $C_i$, $D_i$, $c_i$, $d_i$, $Q_i$, $R_i$, $S_i$, N, T
        Initial guess $(X^0(k)$, $U^0(k)$, $p^0(k)$, $\lambda^0(k)$, $t^0(k))$
        Tollerance $\epsilon \in [10^{-5}, 10^{-3}]$
 2: OUTPUT:
        Optimal solution $(X^*(k)$, $U^*(k)$, $p^*(k)$, $\lambda^*(k)$, $t^*(k))$
 3: PROCEDURE:
 4: Stacking the variables according to the Space-LBT formulation in order to
    obtain the matrices A, B, C, D, $\delta$, Q, R, S used in (6.1), (6.2) and(6.3)
 5: while $\boldsymbol{\lambda}^\top \mathbf{t} > \epsilon$ do    (stopping condition, $\tau$ = current iteration)
 6:     AFFINE-SCALING STEP:
 7:     Set the right-hand side as in (6.8)
 8:     Set $\Pi(T)$ and $\pi(T)$
 9:     for $k : (T-1) \to 1$ do    (Backward recursions)
10:         Block elimination of $\Delta t(k)$, $\Delta\lambda(k)$ and $\Delta X(k+1)$
11:         Calculation of $\Pi(k)$ and $\pi(k)$ through (6.23a) and (6.23b)
            respectively
12:     for $k : 0 \to (T-1)$ do    (Forward recursion)
13:         Calculation of $\Delta U_{\text{aff}}(k)$, $\Delta p_{\text{aff}}(k)$, $\Delta\lambda_{\text{aff}}(k)$, $\Delta t_{\text{aff}}(k)$, $\Delta X_{\text{aff}}(k+1)$
            using (6.26) if $k = 0$ or (6.27) and (6.28) else
14:     CENTERING-CORRECTOR STEP:
15:     Set the *duality gap* $\mu$ (6.32)
16:     Set the *affine duality gap* $\mu_{\text{aff}}$ (6.34) using $\alpha_{\text{aff}}$ (6.33)
17:     Set the *centering parameter* $\sigma$ (6.35)
18:     Set the right-hand side as in (6.29)
19:     Set $\Pi(T)$ and $\pi(T)$
20:     for $k : (T-1) \to 1$ do    (Backward recursions)
21:         Block elimination of $\Delta t(k)$, $\Delta\lambda(k)$ and $\Delta X(k+1)$
22:         Calculation of $\Pi(k)$ and $\pi(k)$ through (6.23a) and (6.23b)
            respectively
23:     for $k : 0 \to (T-1)$ do    (Forward recursion)
24:         Calculation of $\Delta U_{\text{cent}}(k)$, $\Delta p_{\text{cent}}(k)$, $\Delta\lambda_{\text{cent}}(k)$, $\Delta t_{\text{cent}}(k)$,
            $\Delta X_{\text{cent}}(k+1)$ using (6.26) if $k = 0$ or (6.27) and (6.28) else
25:     MAXIMUM STEPLENGTH:
26:     Obtain $(\Delta X(k), \Delta U(k), \Delta p(k), \Delta\lambda(k), \Delta t(k))$ summing the affine and
        the centering contributes as done in (6.36)
27:     Set $\alpha_{\max}$ (6.37)
28:     Choose $\gamma \in (0, 1]$
29:     Set $\alpha$ (6.38)
30:     Update the variables $(X^{\tau+1}(k), U^{\tau+1}(k), p^{\tau+1}(k), \lambda^{\tau+1}(k), t^{\tau+1}(k))$
        through (6.39)
```

6.2 Time-LBT Solver

The solver introduced in Section 6.1 refers to the Time-LBT formulation discussed in Section 2.3.2. In this case the variables of each subsystem are stacked all across the prediction horizon, resulting in a different formulation of the optimization problem. The main contribution of this work is to show that, if certain properties are guaranteed (i.e., cascade structure of the system, separability of cost function and constraints across space), it will be possible to solve a quadratic programming problem spatially, with the same procedure of the solver presented in Section 6.1, which applies the dynamic programming principle with its classical formulation (i.e., temporal-wise).

More specifically, we consider systems of type (2.22)

$$X_i = \Gamma_i \widehat{x}_i + \Omega_i U_i + \Psi_i X_{i-1} \tag{6.41}$$

Constraints on states and inputs (2.15) have been rewritten in a compact form (2.23) and the result is the following:

$$\mathcal{C}_i X_i + \mathcal{D}_i U_i \leq \delta_i \tag{6.42}$$

Weight matrices for the optimization problem have already been introduced in Section 2.3.2. However, it is again appropriate to consider the possible introduction of cross-terms in the cost function design for increasing generality. In particular, the matrix which takes into account the cross-terms is structured as:

$$\mathcal{S}_i = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & S_i & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & S_i \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$$

Note that the structure of matrix $\mathcal{S}_i$ highlights the fact that the vector X_i contains $T+1$ elements, while the vector U_i contains instead T of them. This remarks that the matrix of cross-terms is not supposed to be diagonal.

The cost function (2.25) becomes as follows:

$$J(\{X_i\}_{i=1}^N, \{U_i\}_{i=1}^N) = \frac{1}{2}\sum_{i=1}^{N}\Big(X_i^\top \mathcal{Q}_i X_i + U_i^\top \mathcal{R}_i U_i + 2X_i^\top \mathcal{S}_i U_i\Big) \tag{6.43}$$

where $\mathcal{Q}_i \in \mathbb{R}^{p_i \times p_i}$, $\mathcal{R}_i \in \mathbb{R}^{l_i \times l_i}$ and $\mathcal{S}_i \in \mathbb{R}^{p_i \times l_i}$.

As a result, the minimization problem takes the form

$$\begin{aligned} \min_{X_i, U_i} \quad & J(\{X_i\}, \{U_i\}) \qquad\qquad i = 1 \dots N \\ \text{subject to} \quad & x_i(0) = \widehat{x}_i \\ & X_i = \Gamma_i x_i(0) + \Omega_i U_i + \Psi_i X_{i-1} \\ & \mathcal{C}_i X_i + \mathcal{D}_i U_i \le \delta_i \end{aligned} \tag{6.44}$$

The equality constraint $x_i(0) = \widehat{x}_i$, enforces the decision variable $x_i(0)$ to be equal to the state measurement taken at the current time instant.

6.2.1 KKT Conditions

The Lagrangian function $\mathcal{L}$ has to be calculated as the sum of contributions from each subsystem:

$$\mathcal{L} = \sum_{i=1}^{N} \mathcal{L}_i$$

where the single term $\mathcal{L}_i$ is

$$\begin{aligned} \mathcal{L}_i = \frac{1}{2}\Big(X_i^\top \mathcal{Q}_i X_i + U_i^\top \mathcal{R}_i U_i + 2X_i^\top \mathcal{S}_i U_i\Big) \\ + p_i^\top(-X_i + \Gamma_i \widehat{x}_i + \Omega_i U_i + \Psi_i X_{i-1}) + \lambda_i^\top(\mathcal{C}_i X_i + \mathcal{D}_i U_i - \delta_i) \end{aligned}$$

In particular, $p_i \in \mathbb{R}^{p_i}$ and $\lambda_i \in \mathbb{R}^{\gamma_i}$ are the Lagrange multipliers of the equality and inequality constraints, respectively. Setting the first derivative of the Lagrangian function with respect to the decision variables to zero, the non linear KKT equations are obtained:

$$\text{generic stage}\quad\begin{cases}\frac{\partial\mathcal{L}}{\partial X_i} = \mathcal{Q}_i X_i + \mathcal{S}_i U_i - p_i + \mathcal{C}_i^\top \lambda_i + \Psi_{i+1}^\top p_{i+1} = 0\\ \frac{\partial\mathcal{L}}{\partial U_i} = \mathcal{R}_i U_i + \mathcal{S}_i^\top X_i + \Omega_i^\top p_i + \mathcal{D}_i^\top \lambda_i = 0\\ \frac{\partial\mathcal{L}}{\partial p_i} = -X_i + \Gamma_i \widehat{x}_i + \Omega_i U_i + \Psi_i X_{i-1} = 0\\ \frac{\partial\mathcal{L}}{\partial \lambda_i} = \mathcal{C}_i X_i + \mathcal{D}_i U_i - \delta_i + t_i = 0\\ \Lambda_i \Theta_i e = 0\end{cases}$$
$$(\lambda_i, t_i) \geq 0 \tag{6.45}$$

A slack term t_i has been introduced such that the following relationships hold:

$$\begin{aligned} &t_i = -\mathcal{C}_i X_i - \mathcal{D}_i U_i + \delta_i && i = 1, \dots, N\\ &\lambda_{i,j} t_{i,j} = 0 && i = 1, \dots, N;\ j = 1, \dots, \gamma_i\end{aligned}$$

Moreover, Λ_i and Θ_i matrices are defined such that they are positive semidefinite.

$$\Lambda_i = \begin{bmatrix} \lambda_{i,1} & & & \\ & \lambda_{i,2} & & \\ & & \ddots & \\ & & & \lambda_{i,\gamma_i}\end{bmatrix} \qquad \Theta_i = \begin{bmatrix} t_{i,1} & & & \\ & t_{i,2} & & \\ & & \ddots & \\ & & & t_{i,\gamma_i}\end{bmatrix} \qquad e = \begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix}$$

Note that in this case, the KKT conditions have the same structure for each subsystem, and they are non linear, due to the presence of $\Lambda_i \Theta_i e = 0$.

6.2.2 Solver

Once the non linear KKT conditions have been defined, the IPM requires the solution of a linear system of equations. Therefore, such conditions are linearized, and the following system is obtained:

$$\begin{bmatrix}
\mathcal{Q}_1 & \mathcal{S}_1 & -I & \mathcal{C}_1^\top & 0 & & & \Psi_2^\top & & & \\
\mathcal{S}_1^\top & \mathcal{R}_1 & \Omega_1^\top & \mathcal{D}_1^\top & 0 & & & & & & \\
-I & \Omega_1 & 0 & 0 & 0 & & & & & & \\
\mathcal{C}_1 & \mathcal{D}_1 & 0 & 0 & I & & & & & & \\
0 & 0 & 0 & \Theta_1 & \Lambda_1 & & & & & & \\
 & & & & & \mathcal{Q}_2 & \mathcal{S}_2 & -I & \mathcal{C}_2^\top & 0 & \\
 & & & & & \mathcal{S}_2^\top & \mathcal{R}_2 & \Omega_2^\top & \mathcal{D}_2^\top & 0 & \\
\Psi_2 & & & & & -I & \Omega_2 & 0 & 0 & 0 & \\
 & & & & & \mathcal{C}_2 & \mathcal{D}_2 & 0 & 0 & I & \\
 & & & & & 0 & 0 & 0 & \Theta_2 & \Lambda_2 & \\
 & & & & & & & & & & \ddots
\end{bmatrix}
\begin{bmatrix}
\Delta X_1 \\ \Delta U_1 \\ \Delta p_1 \\ \Delta \lambda_1 \\ \Delta t_1 \\ \Delta X_2 \\ \Delta U_2 \\ \Delta p_2 \\ \Delta \lambda_2 \\ \Delta t_2 \\ \vdots
\end{bmatrix}
=
\begin{bmatrix}
r_1^Q \\ r_1^R \\ r_1^A \\ r_1^C \\ r_1^S \\ r_2^Q \\ r_2^R \\ r_2^A \\ r_2^C \\ r_2^S \\ \vdots
\end{bmatrix}
\tag{6.46}$$

The multistaged system (6.46) is now composed by N stages. In a similar way, the system (6.7) regarding the Space-LBT solver, was composed by T stages. However, also in this case, each stage is coupled with the previous and the next one through the terms Ψ_i and $\Psi_{i+1}^\top$ respectively. Moreover, the first stage and the last one have only one interconnection term (i.e., Ψ_2 and Ψ_N respectively) due to the cascade structure of the systems taken into account. This will allow us to apply recursive backward and forward procedures in a similar fashion of the previous algorithm.

In the following, the *Mehrotra's predictor corrector algorithm* is applied to solve the optimization problem.

Affine-scaling step

The choice of the right-hand side is such that the non-linear KKT conditions are satisfied, for the current IPM iteration. It is widely described in Section

5.2 and the result is the following:

$$\begin{bmatrix} r_i^Q \\ r_i^R \\ r_i^A \\ r_i^C \\ r_i^S \end{bmatrix} = \begin{bmatrix} -(\mathcal{Q}_i X_i + \mathcal{S}_i U_i - p_i + \mathcal{C}_i^\top \lambda_i + \Psi_{i+1}^\top p_{i+1}) \\ -(\mathcal{R}_i U_i + \mathcal{S}_i^\top X_i + \Omega_i^\top p_i + \mathcal{D}_i^\top \lambda_i) \\ -(-X_i + \Gamma_i \widehat{x}_i + \Omega_i U_i + \Psi_i X_{i-1}) \\ -(\mathcal{C}_i X_i + \mathcal{D}_i U_i - \delta_i + t_i) \\ -(\Lambda_i \Theta_i e) \end{bmatrix} \tag{6.47}$$

In order to write the system in a more convenient form, some variable are eliminated. The generic i-th stage of (6.46) is:

$$\begin{bmatrix} & & \mathcal{Q}_i & \mathcal{S}_i & -I & \mathcal{C}_i^\top & 0 & \cdots & \Psi_{i+1}^\top \\ & & \mathcal{S}_i^\top & \mathcal{R}_i & \Omega_i^\top & \mathcal{D}_i^\top & 0 & & \\ \Psi_i & \cdots & -I & \Omega_i & 0 & 0 & 0 & & \\ & & \mathcal{C}_i & \mathcal{D}_i & 0 & 0 & I & & \\ & & 0 & 0 & 0 & \Theta_i & \Lambda_i & & \end{bmatrix} \begin{bmatrix} \Delta X_{i-1} \\ \vdots \\ \Delta X_i \\ \Delta U_i \\ \Delta p_i \\ \Delta \lambda_i \\ \Delta t_i \\ \vdots \\ \Delta p_{i+1} \end{bmatrix} = \begin{bmatrix} r_i^Q \\ r_i^R \\ r_i^A \\ r_i^C \\ r_i^S \end{bmatrix} \tag{6.48}$$

The first elimination concerns the terms Δt_i . Since Λ_i is a positive definite diagonal matrix according to a suitable initialization of the Lagrangian multipliers, it is possible to invert it obtaining:

$$\Delta t_i = \Lambda_i^{-1}(r_i^S - \Theta_i \Delta \lambda_i) \tag{6.49}$$

Substituting (6.49) into the equation

$$\mathcal{C}_i \Delta X_i + \mathcal{D}_i \Delta U_i + \Delta t_i = r_i^C$$

we obtain

$$\mathcal{C}_i \Delta X_i + \mathcal{D}_i \Delta U_i + Z_i \Delta \lambda_i = r_i^Z \tag{6.50}$$

where $Z_i = -\Lambda_i^{-1}\Theta_i$ and $r_i^Z = r_i^C - \Lambda_i^{-1} r_i^S$.

The stage (6.48) takes the following form:

$$\begin{bmatrix} & & \mathcal{Q}_i & \mathcal{S}_i & -I & \mathcal{C}_i^\top & \cdots & \Psi_{i+1}^\top \\ & & \mathcal{S}_i^\top & \mathcal{R}_i & \Omega_i^\top & \mathcal{D}_i^\top & & \\ \Psi_i & \cdots & -I & \Omega_i & 0 & 0 & & \\ & & \mathcal{C}_i & \mathcal{D}_i & 0 & Z_i & & \end{bmatrix} \begin{bmatrix} \Delta X_{i-1} \\ \vdots \\ \Delta X_i \\ \Delta U_i \\ \Delta p_i \\ \Delta \lambda_i \\ \vdots \\ \Delta p_{i+1} \end{bmatrix} = \begin{bmatrix} r_i^Q \\ r_i^R \\ r_i^A \\ r_i^Z \end{bmatrix} \tag{6.51}$$

The second variable that has to be eliminated is the Lagrange multiplier $\Delta\lambda_i$ of the inequality constraints. From equation (6.50) it can be expressed by:

$$\Delta\lambda_i = Z_i^{-1}(r_i^Z - \mathcal{C}_i\Delta X_i - \mathcal{D}_i\Delta U_i) \tag{6.52}$$

Substituting such expression into the first two equations of (6.51), they become as follows:

$$(\mathcal{Q}_i - \mathcal{C}_i^\top Z_i^{-1}\mathcal{C}_i)\Delta X_i + (\mathcal{S}_i - \mathcal{C}_i^\top Z_i^{-1}\mathcal{D}_i)\Delta U_i - \Delta p_i \\ + \Psi_{i+1}\Delta p_{i+1} = r_i^Q - \mathcal{C}_i^\top Z_i^{-1} r_i^Z$$

$$(\mathcal{S}_i^\top - \mathcal{D}_i^\top Z_i^{-1}\mathcal{C}_i)\Delta X_i + (\mathcal{R}_i - \mathcal{D}_i^\top Z_i^{-1}\mathcal{D}_i)\Delta U_i + \Omega_i^\top \Delta p_i \\ = r_i^R - \mathcal{D}_i^\top Z_i^{-1} r_i^Z$$

Let us define the terms

$$\widetilde{\mathcal{Q}}_i = (\mathcal{Q}_i - \mathcal{C}_i^\top Z_i^{-1}\mathcal{C}_i) \qquad \widetilde{\mathcal{R}}_i = (\mathcal{R}_i - \mathcal{D}_i^\top Z_i^{-1}\mathcal{D}_i) \qquad \widetilde{\mathcal{S}}_i = (\mathcal{S}_i - \mathcal{C}_i^\top Z_i^{-1}\mathcal{D}_i)$$
$$\widetilde{r}_i^Q = r_i^Q - \mathcal{C}_i^\top Z_i^{-1} r_i^Z \qquad \widetilde{r}_i^R = r_i^R - \mathcal{D}_i^\top Z_i^{-1} r_i^Z$$

Since Z is invertible, we obtain:

$$\widetilde{\mathcal{Q}}_i\Delta X_i + \widetilde{\mathcal{S}}_i\Delta U_i - \Delta p_i + \Psi_{i+1}\Delta p_{i+1} = \widetilde{r}_i^Q$$
$$\widetilde{\mathcal{S}}_i^\top \Delta X_i + \widetilde{\mathcal{R}}_i\Delta U_i + \Omega_i^\top \Delta p_i = \widetilde{r}_i^R$$

Therefore, the generic stage takes its final form, which is shown below:

$$\begin{bmatrix} & & \widetilde{\mathcal{Q}}_i & \widetilde{\mathcal{S}}_i & -I & \cdots & \Psi_{i+1}^\top \\ & & \widetilde{\mathcal{S}}_i^\top & \widetilde{\mathcal{R}}_i & \Omega_i^\top & & \\ \Psi_i & \cdots & -I & \Omega_i & 0 & & \end{bmatrix} \begin{bmatrix} \Delta X_{i-1} \\ \vdots \\ \Delta X_i \\ \Delta U_i \\ \Delta p_i \\ \vdots \\ \Delta p_{i+1} \end{bmatrix} = \begin{bmatrix} \widetilde{r}_i^Q \\ \widetilde{r}_i^R \\ r_i^A \end{bmatrix} \tag{6.53}$$

Using the more convenient structure (6.53), finally it is possible to rewrite the whole linear system (6.46) as:

$$\begin{bmatrix} \widetilde{\mathcal{Q}}_1 & \widetilde{\mathcal{S}}_1 & -I & & & \Psi_2^\top & & & & \\ \widetilde{\mathcal{S}}_1^\top & \widetilde{\mathcal{R}}_1 & \Omega_1^\top & & & & & & & \\ -I & \Omega_1 & 0 & & & & & & & \\ & & & \widetilde{\mathcal{Q}}_2 & \widetilde{\mathcal{S}}_2 & -I & & & \Psi_3^\top & \\ & & & \widetilde{\mathcal{S}}_2^\top & \widetilde{\mathcal{R}}_2 & \Omega_2^\top & & & & \\ \Psi_2 & & & -I & \Omega_2 & 0 & & & & \\ & & & & & & \widetilde{\mathcal{Q}}_3 & \widetilde{\mathcal{S}}_3 & -I & \\ & & & & & & \widetilde{\mathcal{S}}_3^\top & \widetilde{\mathcal{R}}_3 & \Omega_3^\top & \\ & & & \Psi_3 & & & -I & \Omega_3 & 0 & \\ & & & & & & & & & \ddots \end{bmatrix} \begin{bmatrix} \Delta X_1 \\ \Delta U_1 \\ \Delta p_1 \\ \Delta X_2 \\ \Delta U_2 \\ \Delta p_2 \\ \Delta X_3 \\ \Delta U_3 \\ \Delta p_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} \widetilde{r}_1^Q \\ \widetilde{r}_1^R \\ r_1^A \\ \widetilde{r}_2^Q \\ \widetilde{r}_2^R \\ r_2^A \\ \widetilde{r}_3^Q \\ \widetilde{r}_3^R \\ r_3^A \\ \vdots \end{bmatrix} \tag{6.54}$$

Once the linear system of equation (6.54) has been obtained, the aim is to find a relationship between consecutive stages with an algorithm similar to the one used in [2]. Starting from the final stage (N-th), once a relationship holds between that subsystem and the previous one, by induction arguments it is possible to conclude that the same relationship holds for all the other stages. This is possible because in view of the fact that all the N stages have the same structure. Specifically, the last stage takes the following form:

$$\begin{bmatrix} & \ddots & & & \\ & & \widetilde{\mathcal{Q}}_N & \widetilde{\mathcal{S}}_N & -I \\ & & \widetilde{\mathcal{S}}_N^\top & \widetilde{\mathcal{R}}_N & \Omega_N \\ \Psi_N & \cdots & -I & \Omega_N & 0 \end{bmatrix} \begin{bmatrix} \Delta X_{N-1} \\ \vdots \\ \Delta X_N \\ \Delta U_N \\ \Delta p_N \end{bmatrix} = \begin{bmatrix} \vdots \\ \widetilde{r}_N^Q \\ \widetilde{r}_N^R \\ r_N^A \end{bmatrix} \tag{6.55}$$

At first, the last stage (6.55) has to be reduced to a single equation, in order to perform the block-eliminations. More specifically, the two equations without the coupling term Ψ_N are resolved for ΔX_N and ΔU_N:

$$\begin{cases} \widetilde{\mathcal{Q}}_N \Delta X_N + \widetilde{\mathcal{S}}_N \Delta U_N - \Delta p_N = \widetilde{r}_N^Q \\ \qquad \Rightarrow \Delta X_N = \widetilde{\mathcal{Q}}_N^{-1}\Big(\widetilde{r}_N^Q - \widetilde{\mathcal{S}}_N \Delta U_N + \Delta p_N\Big) \\ \\ \widetilde{\mathcal{S}}_N^\top \Delta X_N + \widetilde{\mathcal{R}}_N \Delta U_N + \Omega_N^\top \Delta p_N = \widetilde{r}_N^R \\ \qquad \Rightarrow \Delta U_N = \widetilde{\mathcal{R}}_N^{-1}\Big(\widetilde{r}_N^R - \widetilde{\mathcal{S}}_N^\top \Delta X_N - \Omega_N^\top \Delta p_N\Big) \end{cases}$$

Solving the linear system of two equations in two unknown variables, the expressions of ΔX_N and ΔU_N are calculated as

$$\begin{cases} \Delta X_N = \Big(\widetilde{\mathcal{Q}}_N - \widetilde{\mathcal{S}}_N \widetilde{\mathcal{R}}_N^{-1} \widetilde{\mathcal{S}}_N^\top\Big)^{-1} \Big[\widetilde{r}_N^Q - \widetilde{\mathcal{S}}_N \widetilde{R}_N^{-1} \widetilde{r}_N^R + \Big(I + \widetilde{\mathcal{S}}_N \widetilde{\mathcal{R}}_N^{-1} \Omega_N^\top\Big)\Delta p_N\Big] \\ \\ \Delta U_N = \Big(\widetilde{\mathcal{R}}_N - \widetilde{\mathcal{S}}_N^\top \widetilde{\mathcal{Q}}_N^{-1} \widetilde{\mathcal{S}}_N\Big)^{-1} \Big[\widetilde{r}_N^R - \widetilde{\mathcal{S}}_N^\top \widetilde{\mathcal{Q}}_N^{-1} \widetilde{r}_N^Q - \Big(\Omega_N^\top + \widetilde{\mathcal{S}}_N^\top \widetilde{\mathcal{Q}}_N^{-1}\Big)\Delta p_N\Big] \end{cases}$$

and they are ready to be substituted into the third equation of (6.55). The final result is the following simple equation:

$$\Psi_N \Delta X_{N-1} - \Pi_N \Delta p_N = \pi_N \tag{6.56}$$

where

$$\begin{aligned} \Pi_N = \Big(\widetilde{\mathcal{Q}}_N - \widetilde{\mathcal{S}}_N \widetilde{\mathcal{R}}_N^{-1} \widetilde{\mathcal{S}}_N^\top\Big)^{-1} \Big(I + \widetilde{\mathcal{S}}_N \widetilde{\mathcal{R}}_N^{-1} \Omega_N^\top\Big) + \\ \Omega_N \Big(\widetilde{\mathcal{R}}_N - \widetilde{\mathcal{S}}_N^\top \widetilde{\mathcal{Q}}_N^{-1} \widetilde{\mathcal{S}}_N\Big)^{-1} \Big(\Omega_N^\top + \widetilde{\mathcal{S}}_N^\top \widetilde{\mathcal{Q}}_N^{-1}\Big) \end{aligned} \tag{6.57a}$$

$$\pi_N = r_N^A + \left(\widetilde{\mathcal{Q}}_N - \widetilde{\mathcal{S}}_N \widetilde{\mathcal{R}}_N^{-1} \widetilde{\mathcal{S}}_N^\top\right)^{-1} \left(\widetilde{r}_N^Q - \widetilde{\mathcal{S}}_N \widetilde{\mathcal{R}}_N^{-1} \widetilde{r}_N^R\right) - \\ \Omega_N \left(\widetilde{\mathcal{R}}_N - \widetilde{\mathcal{S}}_N^\top \widetilde{\mathcal{Q}}_N^{-1} \widetilde{\mathcal{S}}_N\right)^{-1} \widetilde{r}_N^R - \widetilde{\mathcal{S}}_N^\top \widetilde{\mathcal{Q}}_N^{-1} \widetilde{r}_N^Q \quad (6.57b)$$

The similarity in the notation of (6.56) compared with (6.24) is remarkable. In fact, they both represent the recursive equation which make the elimination of a block into the previous one possible. In particular, in the solver discussed in Section 6.1, the equation (6.25) highlights the fact that such backward recursion was performed by using a Riccati recursive law. On the other hand, (6.57a) shows that a backward recursion is still possible in (6.54), even though it takes a slightly different form (i.e., not Riccati equation anymore).

Regarding the last two stages, the resulting structure is the following:

$$\begin{bmatrix} \ddots & & & & \\ & \widetilde{\mathcal{Q}}_{N-1} & \widetilde{\mathcal{S}}_{N-1} & -I & \Psi_N^\top \\ & \widetilde{\mathcal{S}}_{N-1}^\top & \widetilde{\mathcal{R}}_{N-1} & \Omega_{N-1}^\top & \\ & -I & \Omega_{N-1} & 0 & \\ & \Psi_N & & & -\Pi_N \end{bmatrix} \begin{bmatrix} \vdots \\ \Delta X_{N-1} \\ \Delta U_{N-1} \\ \Delta p_{N-1} \\ \Delta p_N \end{bmatrix} = \begin{bmatrix} \vdots \\ \widetilde{r}_{N-1}^Q \\ \widetilde{r}_{N-1}^R \\ r_{N-1}^A \\ \pi_N \end{bmatrix}$$

In order to check if the recursive law holds for each backward iteration, one last elimination of the variable Δp_N is needed:

$$\Delta p_N = \Pi_N^{-1} \left(\Psi_N \Delta X_{N-1} - \pi_N\right) \quad (6.58)$$

Equation (6.58) is used in the first equation of the $(N-1)$-th stage, and it becomes:

$$\left(\widetilde{\mathcal{Q}}_{N-1} + \Psi_N^\top \Pi_N^{-1} \Psi_N\right) \Delta X_{N-1} + \widetilde{\mathcal{S}}_{N-1} \Delta U_{N-1} - \Delta p_{N-1} = \widetilde{r}_{N-1}^Q + \Psi_N^\top \Pi_N^{-1} \pi_N$$

Defining the new terms

$$\mathcal{Q}_{N-1}^* = \widetilde{\mathcal{Q}}_{N-1} + \Psi_N^\top \Pi_N^{-1} \Psi_N \qquad r_{N-1}^{*Q} = \widetilde{r}_{N-1}^Q + \Psi_N^\top \Pi_N^{-1} \pi_N$$

the $(N-1)$-th stage takes the final form:

$$\begin{bmatrix} \ddots & & & & \\ & & \mathcal{Q}^*_{N-1} & \widetilde{\mathcal{S}}_{N-1} & -I \\ & & \widetilde{\mathcal{S}}^\top_{N-1} & \widetilde{\mathcal{R}}_{N-1} & \Omega^\top_{N-1} \\ \Psi_{N-1} & \cdots & -I & \Omega_{N-1} & 0 \end{bmatrix} \begin{bmatrix} \vdots \\ \Delta X_{N-2} \\ \vdots \\ \Delta X_{N-1} \\ \Delta U_{N-1} \\ \Delta p_{N-1} \end{bmatrix} = \begin{bmatrix} \vdots \\ r^{*Q}_{N-1} \\ \widetilde{r}^R_{N-1} \\ r^A_{N-1} \end{bmatrix} \tag{6.59}$$

It is possible to see that the structure in (6.59) is identical to the structure in (6.55): they only differ in the matrix $\widetilde{\mathcal{Q}}_N$ which becomes $\mathcal{Q}^*_{N-1}$, and the right-hand term $\widetilde{r}^Q_N$ which becomes r^{*Q}_{N-1} . Therefore, the conclusion is that the same recursive law is applicable for each stage from $i = N$ to $i = 1$. The first stage, after all the block-eliminations, is

$$\begin{bmatrix} \mathcal{Q}^*_1 & \widetilde{\mathcal{S}}_1 & -I \\ \widetilde{\mathcal{S}}^\top_1 & \widetilde{\mathcal{R}}_1 & \Omega^\top_1 \\ -I & \Omega_1 & 0 \end{bmatrix} \begin{bmatrix} \Delta X_1 \\ \Delta U_1 \\ \Delta p_1 \end{bmatrix} = \begin{bmatrix} r^{*Q}_1 \\ \widetilde{r}^R_1 \\ r^A_1 \end{bmatrix}$$

Eventually, it is further reduced to the single equation $-\Pi_1 \Delta p_1 = \pi_1$, which is used to calculate Δp_1.

At the end of all the backward recursions, the forward procedure is applied to find the search directions, starting from the first stage toward the last one. More specifically, the solution of the first stage is given by:

$$\begin{cases} \Delta p_1 = -\Pi_1^{-1}\pi_1 \\ \Delta U_1 = \big(\widetilde{\mathcal{R}}_1 - \widetilde{\mathcal{S}}^\top_1 \mathcal{Q}^{*-1}_1 \widetilde{\mathcal{S}}_1\big)^{-1}\big[\widetilde{r}^R_1 - \widetilde{\mathcal{S}}^\top_1 \mathcal{Q}^{*-1}_1 r^{*Q}_1 - \big(\Omega^\top_1 + \widetilde{\mathcal{S}}^\top_1 Q^{*-1}_1\big)\Delta p_1\big] \\ \Delta X_1 = \mathcal{Q}^{*-1}_1\big(r^{*Q}_1 + \Delta p_1 - \widetilde{\mathcal{S}}_1 \Delta U_1\big) \\ \Delta\lambda_1 = Z_1^{-1}\big(r^Z_1 - C_1\Delta X_1 - D_1 \Delta U_1\big) \\ \Delta t_1 = \Lambda_1^{-1}\big(r^S_1 - \Theta_1 \Delta\lambda_1\big) \end{cases} \tag{6.60}$$

The expressions in (6.60) can be generalized for all the other stages considering that the only thing that changes is how we calculate the term Δp_i since it is present the coupling term with the previous stage Ψ_i.

$$\begin{cases} \Delta p_i = -\Pi_i^{-1}\big(\Psi_i \Delta X_{i-1} - \pi_i\big) \\ \Delta U_i = \big(\widetilde{\mathcal{R}}_i - \widetilde{\mathcal{S}}_i^\top \mathcal{Q}_i^{*-1}\widetilde{\mathcal{S}}_i\big)^{-1}\big[\widetilde{r}_i^R - \widetilde{\mathcal{S}}_i^\top \mathcal{Q}_i^{*-1} r_i^{*Q} - \big(\Omega_i^\top + \widetilde{\mathcal{S}}_i^\top \mathcal{Q}_i^{*-1}\big)\Delta p_i\big] \\ \Delta X_i = \mathcal{Q}_i^{*-1}\big(r_i^{*Q} + \Delta p_i - \widetilde{\mathcal{S}}_i \Delta U_i\big) \\ \Delta \lambda_i = Z_i^{-1}\big(r_i^Z - C_i \Delta X_i - D_i \Delta U_i\big) \\ \Delta t_i = \Lambda_i^{-1}\big(r_i^S - \Theta_i \Delta \lambda_i\big) \end{cases} \tag{6.61}$$

Centering-Corrector step

The *Mehrotra's predictor corrector algorithm* requires a centering step where the right-hand side is:

$$\begin{bmatrix} r_i^Q \\ r_i^R \\ r_i^A \\ r_i^C \\ r_i^S \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -\Delta\Theta_{\text{i,aff}}\Delta\Lambda_{\text{i,aff}} e + \sigma\mu e \end{bmatrix} \tag{6.62}$$

The *duality gap* μ and the *centering parameter* σ are calculated in the same way as done in Section 6.1, but in this case the vectors $\boldsymbol{\lambda}$ and $\mathbf{t}$ are stacked according to the spatial index, instead of the temporal one.

$$\begin{aligned} \boldsymbol{\lambda} &= [\lambda_1^\top, \lambda_2^\top, \dots, \lambda_N^\top]^\top \\ \mathbf{t} &= [t_1^\top, t_2^\top, \dots, t_N^\top]^\top \end{aligned} \tag{6.63}$$

The search directions for the centering step are then calculated using the same algorithm of the affine one (i.e., backward and forward recursions). Eventually, the two contributions are summed and the solution of the τ-th IPM iteration is obtained:

$$\begin{aligned} \Big(\Delta X_i, \Delta U_i, \Delta p_i, \Delta \lambda_i, \Delta t_i\Big) &= \Big(\Delta X_{\text{i,aff}}, \Delta U_{\text{i,aff}}, \Delta p_{\text{i,aff}}, \Delta \lambda_{\text{i,aff}}, \Delta t_{\text{i,aff}}\Big) \\ &\quad + \Big(\Delta X_{\text{i,cent}}, \Delta U_{\text{i,cent}}, \Delta p_{\text{i,cent}}, \Delta \lambda_{\text{i,cent}}, \Delta t_{\text{i,cent}}\Big) \end{aligned} \tag{6.64}$$

Maximum steplength

The solution (6.64) represents the total steplength achievable but, as explained in (5.2.1), it is necessary to reduce the steplength with a scaling factor α. Eventually the updated variables are:

$$\begin{aligned}
X_i^{\tau+1} &= X_i^\tau + \alpha \Delta X_i \\
U_i^{\tau+1} &= U_i^\tau + \alpha \Delta U_i \\
p_i^{\tau+1} &= p_i^\tau + \alpha \Delta p_i \\
\lambda_i^{\tau+1} &= \lambda_i^\tau + \alpha \Delta \lambda_i \\
t_i^{\tau+1} &= t_i^\tau + \alpha \Delta t_i
\end{aligned} \tag{6.65}$$

They will be used for the next IPM iteration $(\tau + 1)$.

6.2.3 Complexity

The complexity of the algorithm presented in Section 6.2 is analized as done in Section 6.1.3, highlighting how the number of subsystems and the prediction horizon affect differently the computational burden.

The system of linearized KKT conditions (6.54) has a block diagonal structure similar to (6.17). However, in this case, due to the spatial causality of the interconnected system, the dimension of the single block is proportional to the prediction horizon T. On the other hand, the dimension of the system (i.e., the number of stages) corresponds to the number of subsystems N.

In the backward recursion the most burdensome computational effort is required by the inversion of the term Π_i, with a complexity which scales cubically to its dimension (i.e., proportional to T). On the other hand, the complexity also grows linearly with the number of subsystems, since, at each iteration of the Interior Point Method, $2N$ recursions (backward and forward) are performed.

For all the aforementioned reasons, we can expect that the complexity of such algorithm is $\mathcal{O}(NT^3)$, as it will be empirically demonstrated in the next chapter.

Algorithm 3 Time-LBT Solver

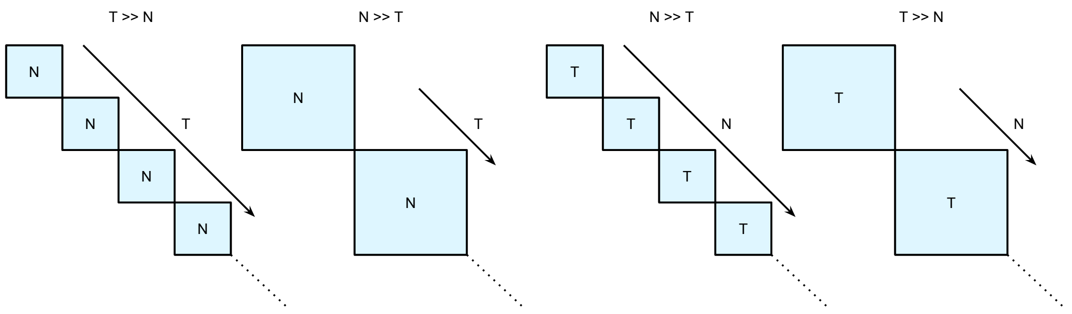
```
1: INPUT:
       $A_i$, $B_i$, $E_i$, $C_i$, $D_i$, $c_i$, $d_i$, $Q_i$, $R_i$, $S_i$, N, T
       Initial guess ($X_i^0$, $U_i^0$, $p_i^0$, $\lambda_i^0$, $t_i^0$)
       Tollerance $\epsilon \in [10^{-5}, 10^{-3}]$
2: OUTPUT:
       Optimal solution ($X_i^*$, $U_i^*$, $p_i^*$, $\lambda_i^*$, $t_i^*$)
3: PROCEDURE:
4: Stacking the variables according to the Time-LBT formulation in order to
   obtain the matrices $\Gamma_i$, $\Omega_i$, $\Psi_i$, $\mathcal{C}_i$, $\mathcal{D}_i$, $\delta_i$, $\mathcal{Q}_i$, $\mathcal{R}_i$, $\mathcal{S}_i$ used in (6.41), (6.42)
   and (6.43)
5: while $\boldsymbol{\lambda}^\top \mathbf{t} > \epsilon$ do    (stopping condition, $\tau$ = current iteration)
6:     AFFINE-SCALING STEP:
7:     Set thre right-hand side as in (6.47)
8:     Set $\Pi_i$ and $\pi_i$
9:     for $i : N \to 2$ do    (Backward recursions)
10:        Block elimination of $\Delta t_i$, $\Delta\lambda_i$ and $\Delta p_i$
11:        Calculation of $\Pi_i$ and $\pi_i$ through (6.57a) and (6.57b) respectively
12:    for $i : 1 \to N$ do    (Forward recursion)
13:        Calculation of $\Delta U_{\text{i,aff}}$, $\Delta p_{\text{i,aff}}$, $\Delta\lambda_{\text{i,aff}}$, $\Delta t_{\text{i,aff}}$, $\Delta X_{\text{i,aff}}$ using (6.60)
           if $i = 1$ or (6.61) else
14:    CENTERING-CORRECTOR STEP:
15:    Set the *duality gap* $\mu$ (6.32)
16:    Set the *affine duality gap* $\mu_{\text{aff}}$ (6.34) using $\alpha_{\text{aff}}$ (6.33)
17:    Set the *centering parameter* $\sigma$ (6.35)
18:    Set the right-hand side as in (6.62)
19:    Set $\Pi_i$ and $\pi_i$
20:    for $i : N \to 2$ do    (Backward recursions)
21:        Block elimination of $\Delta t_i$, $\Delta\lambda_i$ and $\Delta p_i$
22:        Calculation of $\Pi_i$ and $\pi_i$ through (6.57a) and (6.57b) respectively
23:    for $i : 1 \to N$ do    (Forward recursion)
24:        Calculation of $\Delta U_{\text{i,cent}}$, $\Delta p_{\text{i,cent}}$, $\Delta\lambda_{\text{i,cent}}$, $\Delta t_{\text{i,cent}}$,
           $\Delta X_{\text{i,cent}}$ using (6.60) if $i = 1$ or (6.61) else
25:    MAXIMUM STEPLENGTH:
26:    Obtain $(\Delta X_i, \Delta U_i, \Delta p_i, \Delta\lambda_i, \Delta t_i)$ summing the affine and the centering
       contributes as done in (6.64)
27:    Set $\alpha_{\max}$ (6.37)
28:    Choose $\gamma \in (0, 1]$
29:    Set $\alpha$ (6.38)
30:    Update the variables $(X_i^{\tau+1}, U_i^{\tau+1}, p_i^{\tau+1}, \lambda_i^{\tau+1}, t_i^{\tau+1})$ through (6.65)
```

6.3 Concluding Remarks

The proposed algorithms solve a multistaged system of linear equations (i.e., (6.17) and (6.54)) through a recursive method whose efficiency depends on the number of blocks composing such system and their dimension. In particular, focusing on the Space-LBT algorithm, it has a complexity which scales as $\mathcal{O}(TN^3)$. On the other hand, the Time-LBT algorithm is supposed to have a complexity which scales as $\mathcal{O}(T^3N)$.

In view of this, it is easy to conclude that for systems with a greater number of subsystems than the prediction horizon ($N \gg T$) Time-LBT Solver is recommended; in the opposite case ($T \gg N$) the use of Space-LBT Solver is suggested. This situation is represented in Figure 6.2.

Therefore, the computational benefits of Time-LBT Solver are remarkable only for the application on systems which are significantly large in scale.



(a) Space-LBT formulation (b) Time-LBT formulation

Figure 6.2: Multistaged structures

Chapter 7

Simulations

In this chapter the algorithms presented in Chapter 4 and 6 are applied to the irrigation network introduced in Chapter 3. All the physical data and the low level controller parameters are referred to the East Goulburn-30 irrigation channel in Australia, see Section 3.3. Since this network, being composed by just four pools, may not have a large scale such that the advantages of the proposed algorithms are appreciable, where needed we will refer to a channel with $N \geq 4$ pools, modelled as in Chapter 3, with parameters derived from the real ones.

In Section 7.1 we present the results attained through the application of the Cooperative Distributed MPC. In Section 7.2 a computational analysis of all the different algorithms implemented is presented. Eventually, a closed-loop simulation is carried out.

All the simulations have been performed in the Matlab environment, making use, when necessary, of the quadratic optimization function *quadprog*. The main index of performance used for comparing the different solutions is the computational time needed to solve the specific control problem, and it is related to the computational capacity of the CPU used for the simulations. In order to obtain comparable results, all the simulations have been carried out with the same calculator.

7.1 Application of Cooperative Distributed MPC

In this section, simulations regarding the application of Cooperative Distributed MPC to the irrigation network are presented. In particular, such technique is applied making use of both the standard modelization (4.2) and the low order one (4.4) presented in Chapter 4. More specifically, it will be shown how the application of the low order model will lead to better computational performances. For this reason, the model (4.1) will be discarded for all the subsequent simulation tests.

7.1.1 Design Parameters

Weight Matrices

The cooperative algorithms result to be very sensitive to the choice of the weight matrices of the optimization problem (4.7). At first, the weight matrices for the cooperative algorithm have been set. As discussed in Section 4.2.3, we will actually minimize the cost function (4.11), and therefore the actual weights are applied to the outputs $y_i(k) = \overline{C}_i z_i(k)$. An empirical tuning of such parameters has been conducted. As a result we notice that

$$Q_{y_i} = \begin{bmatrix} 10000 & & & \\ & 10 & & \\ & & 10 & \\ & & & 10 \end{bmatrix} \qquad R_i = \begin{bmatrix} 1 \end{bmatrix}$$

are suitable to obtain good performances. This is due to the fact that the level (i.e., the first variable) is related to the process, while all the others are referred to the output of the low-level controllers. For this reason, the classical normalization of contributes can not be performed due to the non-homogeneous nature of the system variables. Moreover, the weight matrices related to the low-level control input variables (i.e., flow-rates) are tuned in order to enhance the response speed of the system.

Stopping Conditions

In Section 4.2.2 the cooperative algorithm needs a suitable stopping condition in order to exit from the negotiation phase and to provide a sub-optimal, but acceptable, solution. More specifically, the optimal solution is theoretically reached when, in two consecutive iterations p, holds:

$$u^p(k) = u^{p+1}(k) \qquad k = 0, \cdots, T-1$$

However, such condition is attained in practice with several iterations. In order to limit their number, a sub-optimality threshold has to be enforced

$$\left\| \frac{u_i^p(k) - u_i^{p+1}(k)}{u_i^p(k)} \right\|^2 \leq \epsilon$$

with $\epsilon = 10^{-10}$. A further stopping condition on the maximum number of iterations has been introduced

$$p \leq p_{max} \tag{7.1}$$

with $p_{max} = 100$. This value has been chosen as a secondary stopping condition. In particular, it is used for security reasons.

7.1.2 Performances Achieved with the Two Models

A simulation is presented in order to show the main differences between the models introduced for the Cooperative Distributed MPC. In particular, computational efficiency will be analysed and the best performance obtained through the low order model will be highlighted. The simulation has been carried out regarding an irrigation network composed by four pools and with a prediction horizon $T = 10$. The receding horizon principle has been applied to the control variables for a short simulation time equal to 10 min with a sampling time $T_s = 1$ min.

In Figure 7.1 it is possible to see that both modelizations lead to the same optimal control sequence. In these simulations the stopping condition (7.1) has been neglected, since the algorithm which makes use of (4.1) exceeds the

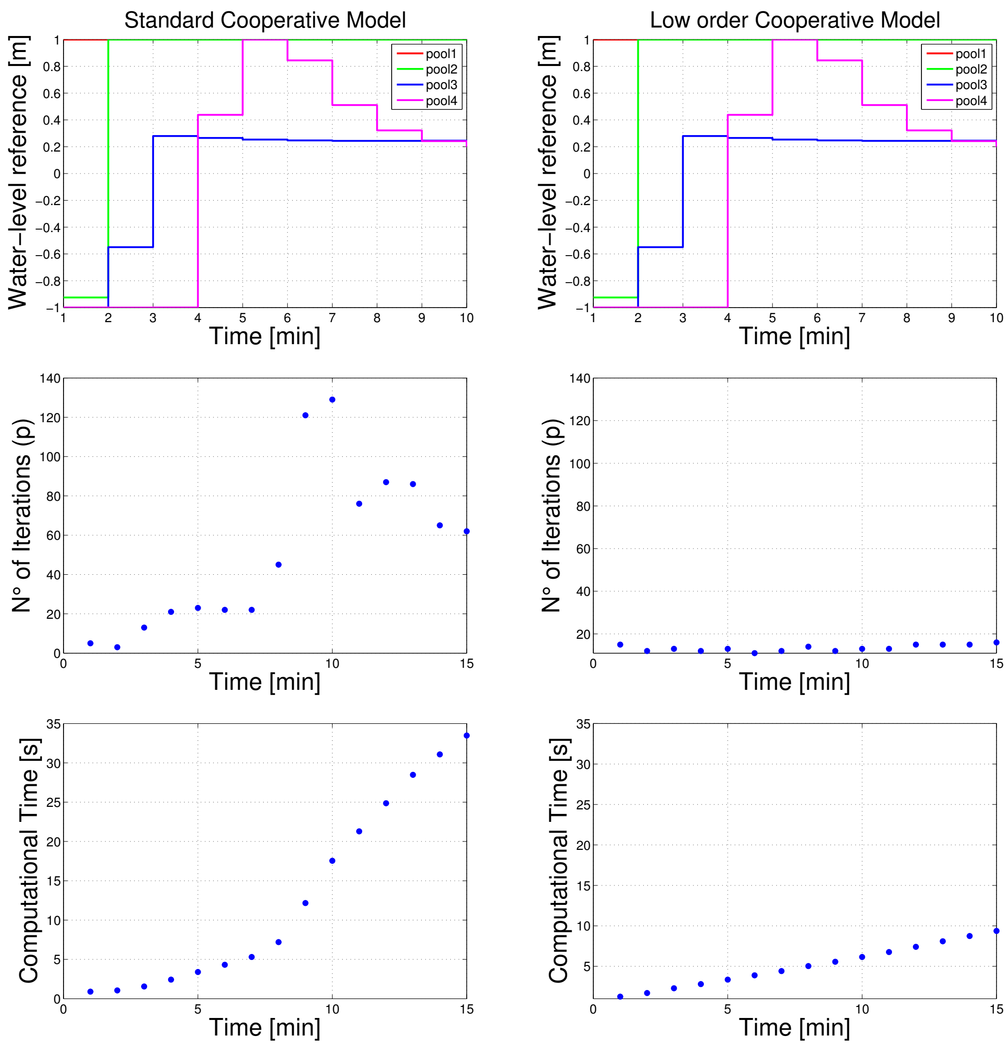


Figure 7.1: Comparison between standard and low order cooperative models

maximum number of allowed iterations. Note that, in case the standard modelization is used, the number of iterations p reaches an order of magnitude higher than the case when the low-order model is employed. Considering that, at each time instant, at every iteration p we solve an optimization problem, as a result, the computational time increases dramatically, making its implementation particularly inefficient. Indeed, the use of the low order model (4.4) provides better performances, and for this reason, it is the one which will be taken into account in the sequel.

7.1.3 State constraints

Finally, the implementation of state constraints through slack variables has been explored. Experimental tests have shown that, as the weight associated to the slack variable increases (i.e., which has the effect of enforcing the satisfaction of hard constraints if possible), an excessive number of iterations were needed to attain the optimal solution. On the other hand, by decreasing such weight (i.e., allowing the violation of the constraints) the security level bounds resulted to be strongly violated. For these reasons we did not take into account anymore constraints on state variables.

7.2 Computational Analysis

In this section an analysis of the computational efficiency of the discussed algorithms is carried out. In particular, the two solvers proposed in Chapter 6 will be analysed, highlighting how their computational effort scales favourably with respect to the values of T and N. Eventually, a computational comparison between the solvers and the cooperative algorithm proposed in Chapter 4 will be presented.

7.2.1 Proposed Algorithm Analysis

Convergence to the optimal solution

The purpose of this part is to analyse the convergence properties of the two IPM solvers introduced in Chapter 6. Indeed, as stated in Chapter 6, the optimization problems (2.20) and (2.26) are the same, even though the two formulations (i.e., Space-LBT and Time-LBT) make use of a different arrangement of the variables.

A test has been performed considering an irrigation channel composed by four pools and a prediction horizon $T = 7$. In Figures 7.2 and 7.3 it is shown how the solvers attain the optimal solution within a single sampling interval.

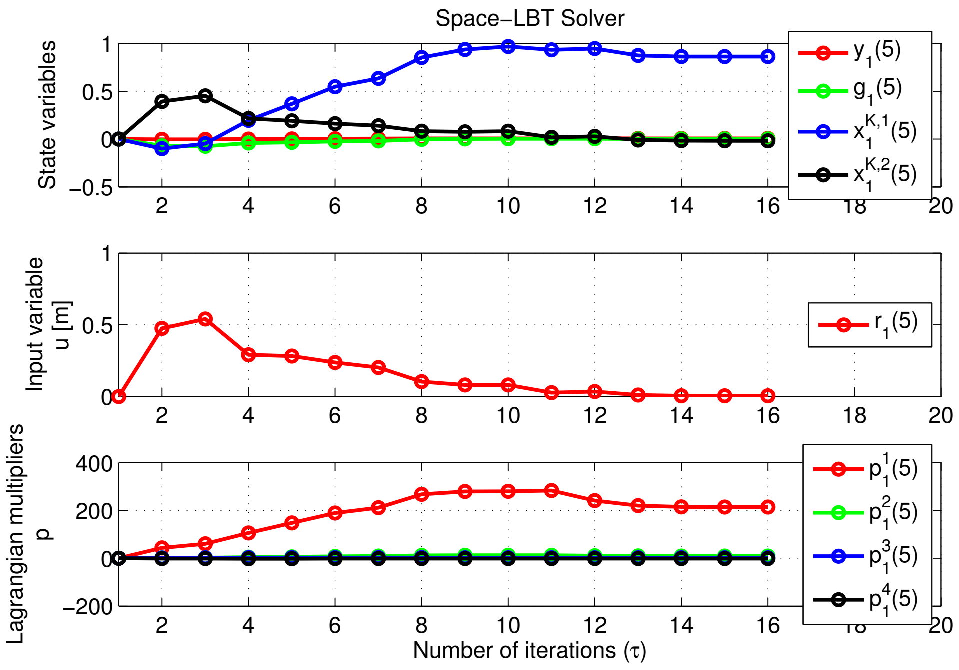


Figure 7.2: Convergence of the solution: Space-LBT

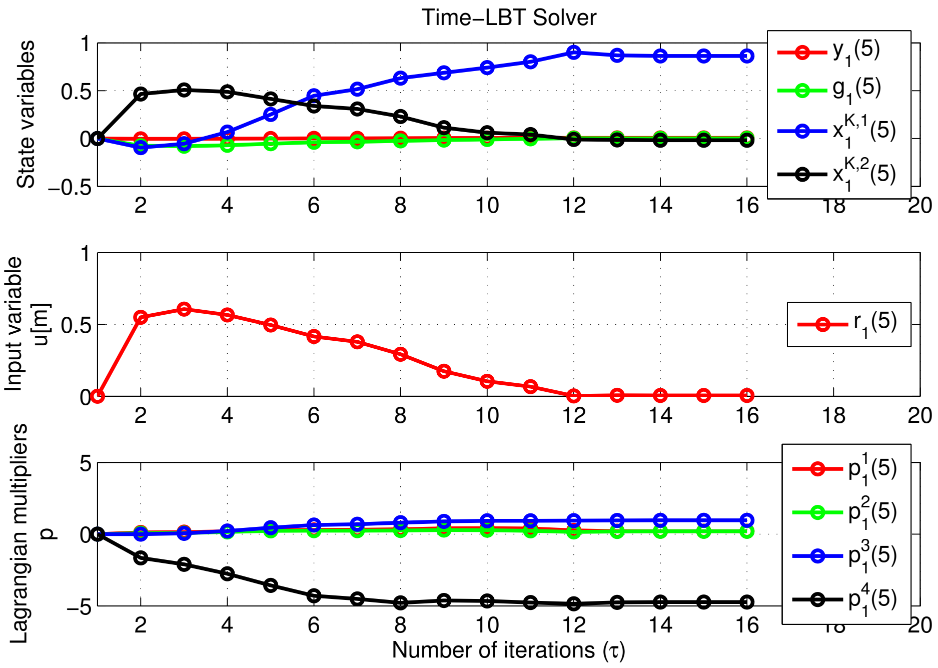


Figure 7.3: Convergence of the solution: Time-LBT

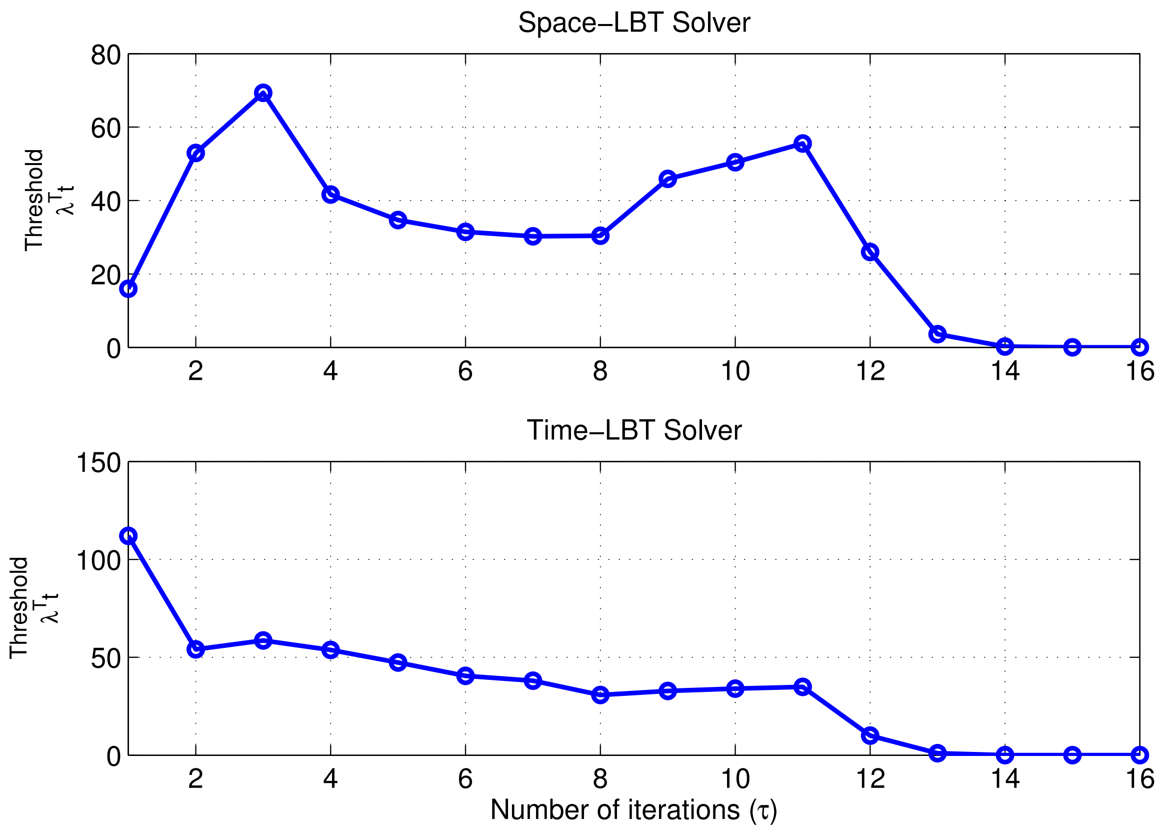
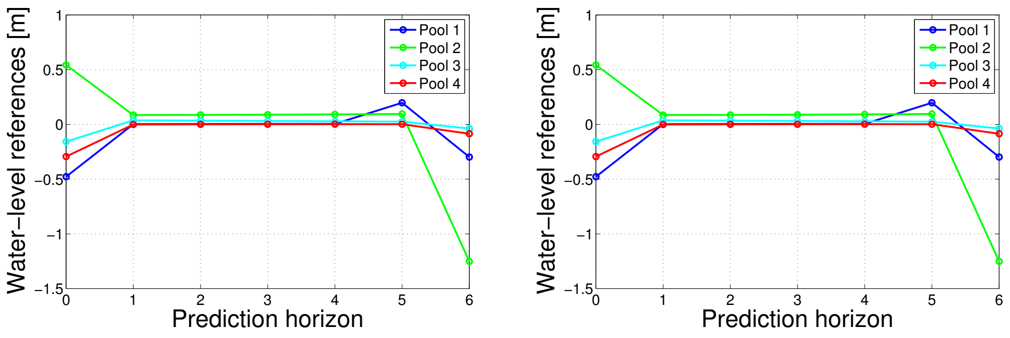


Figure 7.4: Stopping condition

The plots are referred to states, input and Lagrange multipliers of the equality constraints for pool $i = 1$ and instant $k = 5$. Note that both solvers reach the same optimal values after a finite and comparable number of iterations. We remark that the variables of actual interest for the control are the level references, i.e. $r_i(k)$, $k = 0, \cdots, T-1$.

A further evidence of the convergence to the optimal solution, is remarked by the analysis of the stopping condition. In fact, as already discussed, the solution reached is only sub-optimal in order to avoid that the matrices related to the Lagrange multipliers of inequality constraints and to the slack variables become ill conditioned. More specifically, the variable subject to the stopping criterion is the scalar product between the vectors $\boldsymbol{\lambda}$ and $\mathbf{t}$, defined by (6.30) for *Space-LBT Solver* and (6.63) for *Time-LBT Solver*. These values are used as a measure of proximity to the optimal solution, and they approach zero



(a) Space-LBT Solver (b) Time-LBT Solver

Figure 7.5: Optimal trajectories of the water-level references

as the number of iterations grows (see Figure 7.4). Indeed, such number of iterations is directly related to the tolerance of sub-optimality of the solution. In this case it has been set to $\epsilon = 10^{-5}$.

Eventually, the optimal trajectories of the water-level references for the four pools are shown in Figure 7.5. Note that the trends are identical along the prediction horizon for both solvers.

Complexity

The focus of this part is to analyse the computational times required to obtain a solution using the two IMP solvers described in Chapter 6. As a good indicator for the computational effort of the solvers we use the computational time required from the CPU to solve the optimization problem. Such value results to be less accurate than more suitable indicators (e.g., the flop counter), but it is easier to implement and anyway it lends itself to our purpose.

Two cases of study are then introduced, and the aim is to show how the computational time changes as the parameters N or T grow.

Case 1: fixed N, varying T

In this case, the number of subsystems is fixed equal to $N = 10$ and the optimization problems have been solved for both the algorithms varying the prediction horizon from $T = 2$ to $T = 50$. In Figure 7.6, the trend of the

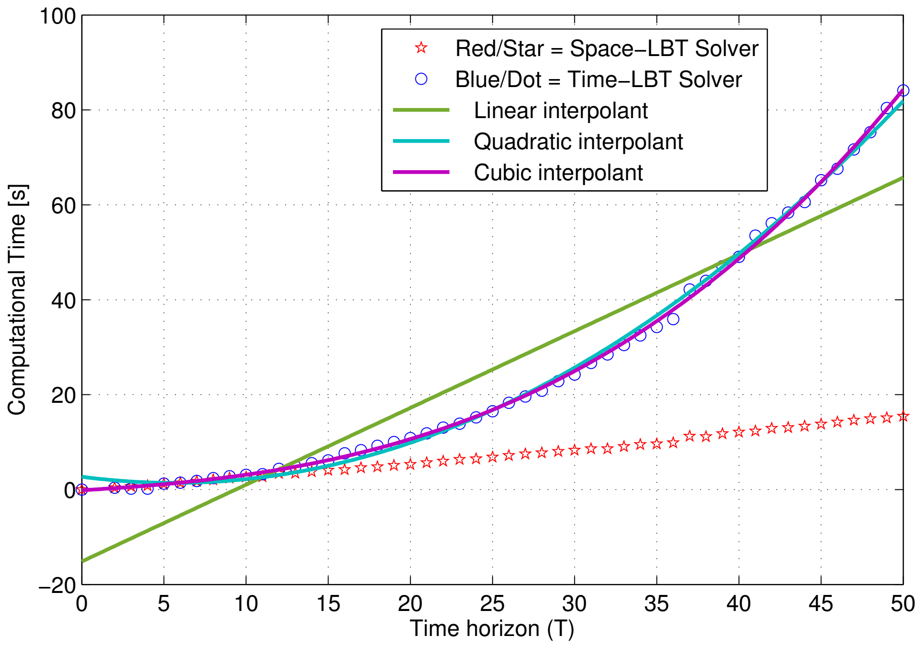


Figure 7.6: Complexity varying T

computational time for both solvers is shown. In particular, it is remarkable that for *Space-LBT Solver* (i.e, red star line) the complexity is linear with T, as asserted in Section 6.1.3. Regarding *Time-LBT Solver* (i.e., blue dot line) the computational time has a polynomial growth, and in order to identify the order of such polynomial, different types of interpolants are used. More specifically, we note that the trend is not linear (green line) but it results to be fitted quite well from the quadratic and the cubic interpolants. The parameters of the cubic interpolant are the following:

$$y = p_1 x^3 + p_2 x^2 + p_3 x + p_4$$

$$p_1 = 0.00042702$$

$$p_2 = 0.0084276$$

$$p_3 = 0.19986$$

$$p_4 = -0.13909$$

The coefficient of the cubic term p_1, even if small, is comparable to the quadratic one p_2. Therefore, we can state that the growth of the computa-

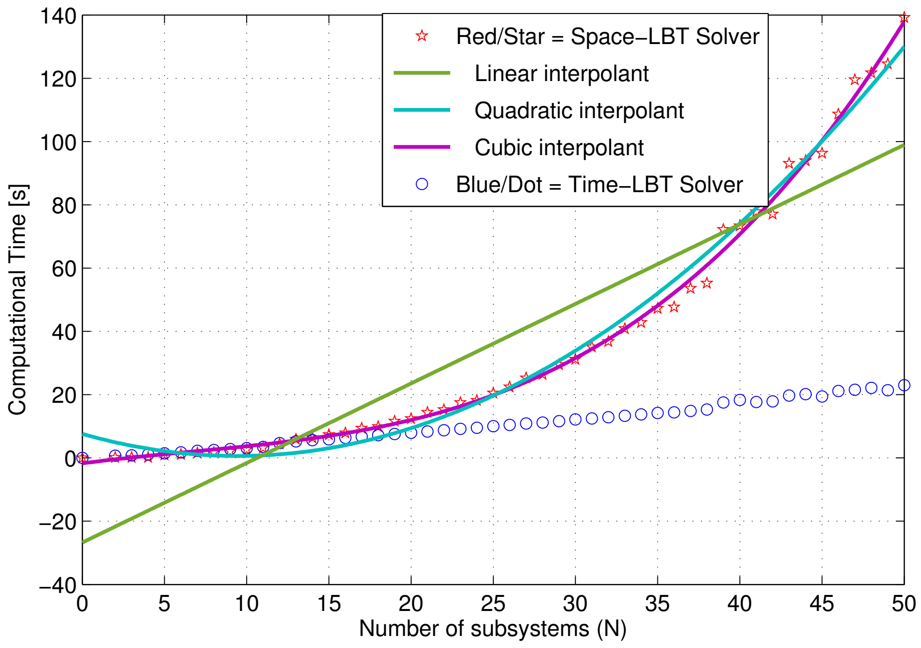


Figure 7.7: Complexity varying N

tional time is cubic.

Case 2: fixed T, varying N

In this simulation, the fixed prediction horizon $T = 10$ has been considered and the optimization problems have been solved varying the number of subsystems from $N = 2$ to $N = 50$. As expected, the solvers display a dual behaviour with respect to the previous case. More specifically, the trend of the complexity of *Time-LBT Solver* is linear in N, while the one of *Space-LBT Solver* is polynomial (see Figure 7.7).

Also in this case different fitting functions have been used (i.e., linear, quadratic and cubic) and, in the same fashion as in Case 1, the coefficients of the cubic interpolant are presented below:

$$y = p_1 x^3 + p_2 x^2 + p_3 x + p_4$$

$$p_1 = 0.0013971$$

$$p_2 = -0.027585$$

$$p_3 = 0.67867$$

$$p_4 = -1.7109$$

Analyzing the coefficients p_1 and p_2 we can say that the growth of the computational time, and so the complexity of the algorithm with respect to N, are cubic.

In view of these results, if $T \gg N$, the use of *Space-LBT Solver* is recommended, since the computational effort is linear in T. Otherwise, if $N \gg T$, *Time-LBT Solver*, whose complexity grows linearly with N, will offer better performances.

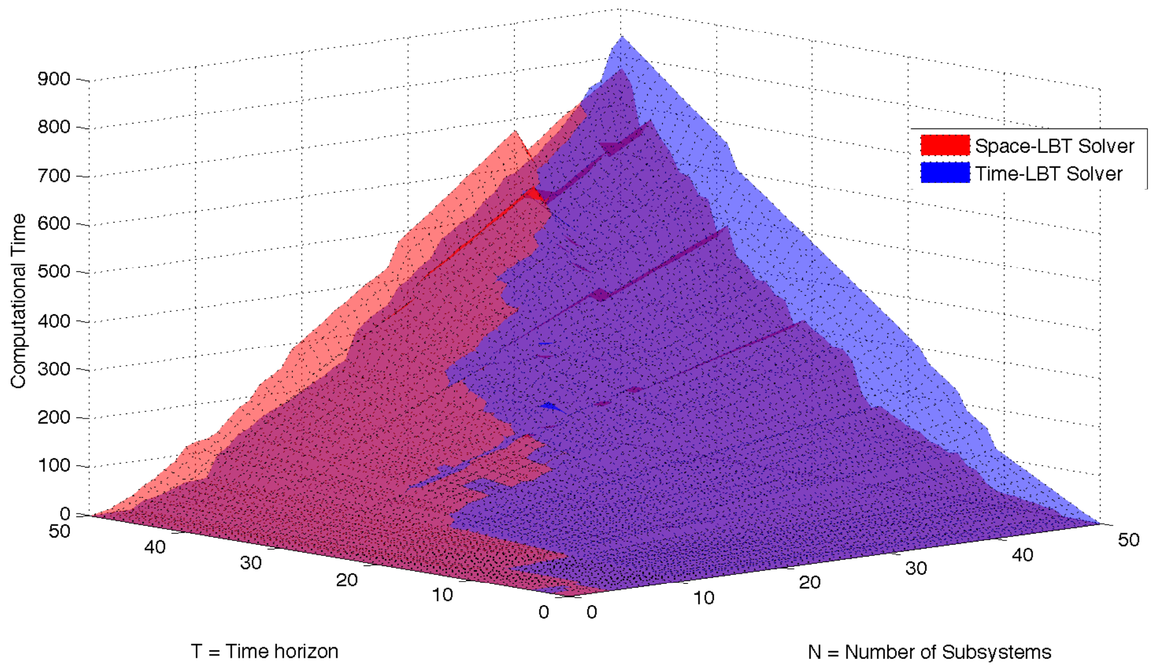


Figure 7.8: 3D representation of the complexity varying N and T simultaneously

Regarding the case where T and N are comparable, it is basically equivalent to employ one solver rather than the other one, since it does not provide any considerable advantage. In Figure 7.8, the computational time for each solver, where both T and N vary, is shown. Basically, one can see that Figure 7.6 and Figure 7.7 are particular sections of this 3-dimensional plot, where either N or T are fixed respectively. The region in the middle (i.e., $T \approx N$), where the two surfaces intersect each other, represent the case in which the performances are similar.

7.2.2 Comparison: Cooperative and LBT Solvers

A comparison between the computational times required for the solution, to be obtained at each time instant, to the distributed MPC optimization problem and for the application of the two LBT Solvers is now proposed. In particular, the algorithms performances are analysed by means of a simulation where the number of subsystems have been fixed to $N = 12$, varying the prediction horizon from $T = 2$ to $T = 50$.

Computational aspects

In Figure 7.9 the computational times needed from the solvers to compute the optimal trajectories are represented. In particular, regarding the Cooperative algorithm, a practical implementation was to solve in series (at each simulation time) the N optimization problems and considering only the computational time of the slowest controller.

The Cooperative algorithm results to be faster due to its distributed nature. In particular, in this strategy, each controller minimize a global cost function with respect to its local control variable, resulting in a smaller number of decision variables. Instead, the LBT Solvers have been applied in a centralized fashion which inherently rely on a bigger amount of decision variables. However, it is worth to remark that in the Cooperative strategy, the times related to the negotiation phase between the controllers have not been taken

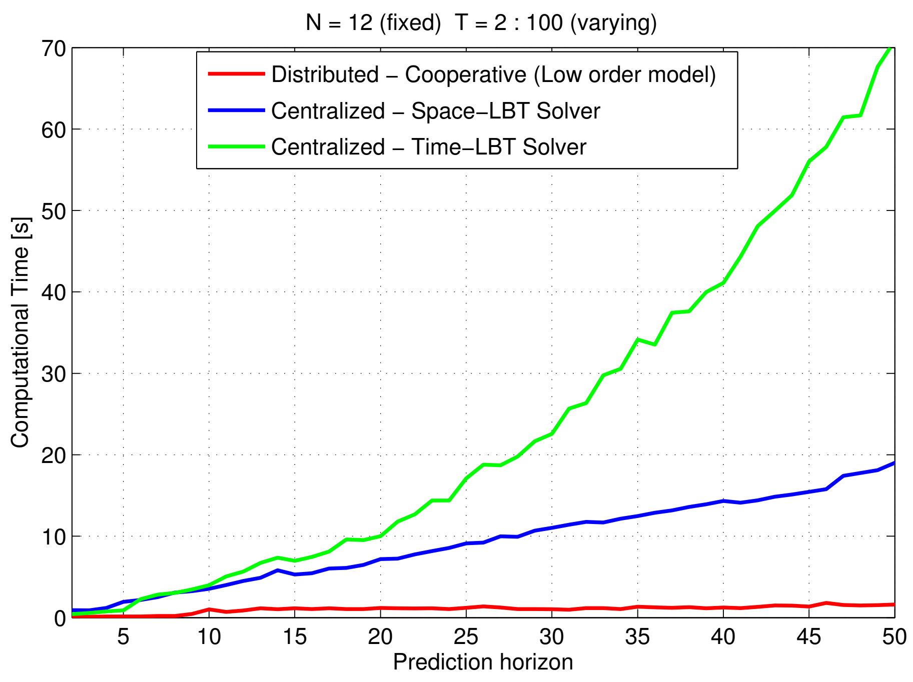


Figure 7.9: Computational time: Cooperative and LBT Solvers

into account. Also, the cooperative MPC scheme has been implemented as if N computers were available, while in the centralized implementation only one computing unit has been employed. For these reasons, a fair simulation which takes into account these considerations should lead to more comparable results where the centralized solvers result to be competitive with the distributed one.

A plot showing similar results in the case when the prediction horizon is kept fixed and the number of subsystem varies is not shown for brevity. This is due to the fact that the performances of the cooperative algorithm is not significantly affected by the variation of N, in view of its distributedness. On the other hand, we experienced a dual behaviour of the LBT Solvers, as in Figure 7.9.

Coding issues

Focusing on implementation aspects, we note that the Cooperative algorithm solve the optimization problem by means of standard solvers (e.g., *Quadprog*, *C-plex*), which are optimized from a coding point of view. On the other hand, the LBT Solvers have not been optimized yet, affecting their performances. Moreover, it would be possible to code such algorithms e.g., the C++ environment, in order to ease the computational load.

Considerations on Constraints

The simulation shown in Figure 7.9 has been carried out considering only constraints on the input variables. This choice is due to the fact that, as discussed in Chapter 4, the Cooperative algorithm does not lend itself to the application of hard constraints on state variables. Indeed, they may be implemented only as soft constraints, allowing the control strategy to temporary violate them. Considering the particular case of study, this involves the violation of water-level security bounds and actuator saturations (note, in fact, that the flow-rates are inputs to the low-level controlled systems). For these reasons, the Cooperative algorithm shows some limitations in real plant implementation.

7.3 Closed-loop Simulations

In this last section we illustrate the closed loop simulations of an irrigation channel obtained by applying the receding horizon principle, to all the three algorithms presented in this Thesis. First, in Figure 7.10 the trajectories of the water-level references, the water-levels and the flow-rates are shown for a simulation time of 30 min. In particular, in this first simulation only constraints on input variables $r_i(t)$ have been considered. However, without imposing constraints on the state variables, it may occur the violation of the security bounds on water-level and flow expressed by the physical limitations (3.10a) and (3.10b). In fact, the variables referred to the flows, violate such conditions during the transient. Secondly, in Figure 7.11 a simulation similar to the previous one is carried out. In such simulation, only hard constraints on state variables has been considered. We remark that, regarding the cooperative distributed MPC, even the application of soft constraints, by means of slack variables, resulted to be impractical. Therefore, only the LBT solvers have been used. Note that the constraints on the water-levels and the flows are satisfied for all the transient.

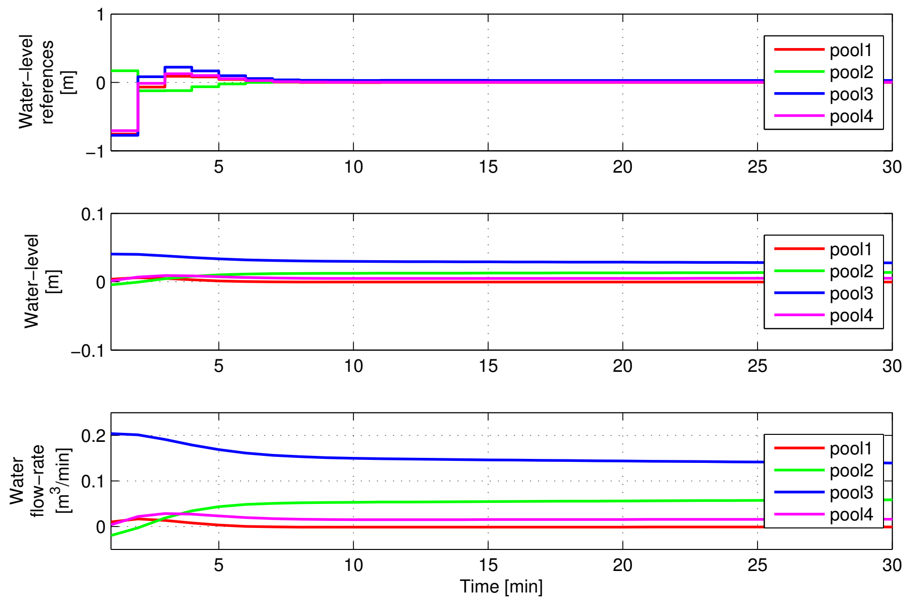


Figure 7.10: Closed loop trajectories: constraints on input variables

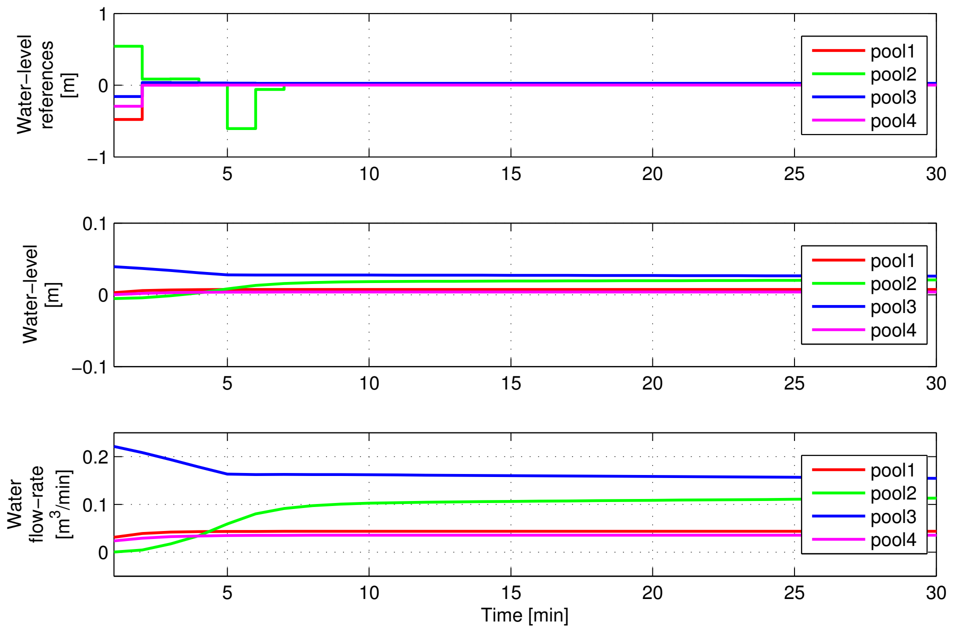


Figure 7.11: Closed loop trajectories: constraints on state variables

Chapter 8

Conclusions and Future Work

In this Thesis the application of an MPC control strategy to a large-scale cascade systems have been explored. In particular, such control technique has been applied in a distributed and centralized fashion regarding the irrigation networks, highlighting the computational performances.

At first, a Cooperative Distributed MPC has been analyzed and implemented. Such control strategy leads to satisfactory performances. However, the impossibility in the imposition of hard constraints on the state variables yields to the possible violation of the security levels related to the irrigation channels.

Then, the centralized approach has been taken into account. In particular, we have focused our attention on the implementation of numerical methods exploiting the sparsity of the optimization problems obtained with two LBT formulations. The two proposed algorithms resulted to be suitable for increasing the efficiency in the centralized control of the whole LSS.

As shown in Chapter 7, regarding the LBT solvers, an optimization of the codes is recommended in order to attain competitive performances. In particular, more efficient methods for the inversion of matrices are recommended. We refer to the Cholesky factorization (see [15]) and to the use of the nilpotent property of a matrix dicussed in [6]. Furthermore, we envisage the implementation of a pre-compiled code in a C++ environment.

A further future work includes the research in the application of the LBT solvers in a distributed fashion, to further increase their applicability to large-scale systems.

Eventually, it would be interesting the implementation of the proposed algorithms on a real channel, in order to analyse their performances in a realistic benchmark.

Bibliography

[1] Department of environment and primary industries, the state government of Victoria, Australia and Goulburn-Murray water, “improving irrigation efficiency,” Tech. Rep. Available online. `http://www.depi.vic.gov.au/water/rural-water-and-irrigation/murray-darling-basin`.

[2] Christopher V Rao, Stephen J Wright, and James B Rawlings. Application of interior-point methods to model predictive control. *Journal of optimization theory and applications*, 99(3):723–757, 1998.

[3] Iman Shames and Michael Cantoni. On computing quadratic controls for acyclic networks of heterogeneous systems. In *European Control Conference (ECC)*, pages 3306–3311, 2013.

[4] Riccardo Scattolini and Lalo Magni. Complementi di controlli automatici. *Pitagora Editrice Bologna*, 2006.

[5] Dragoslav D Siljak. *Decentralized control of complex systems.* Academic Press, 1991.

[6] Juan Luis Jerez, Eric C Kerrigan, and George A Constantinides. A condensed and sparse QP formulation for predictive control. In *Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 5217–5222. IEEE, 2011.

[7] Michael Cantoni, Erik Weyer, Yuping Li, Su Ki Ooi, Iven Mareels, and Matthew Ryan. Control of large-scale irrigation networks. *Proceedings of the IEEE*, 95(1):75–91, 2007.

[8] Paolo Bolzern, Riccardo Scattolini, and Nicola Schiavoni. *Fondamenti di controlli automatici.* McGraw-Hill Libri Italia, 1998.

[9] Amir R Neshastehriz, Michael Cantoni, and Iman Shames. Water-level reference planning for automated irrigation channels via robust MPC. In *Control Conference (ECC), 2014 European*, pages 1331–1336. IEEE, 2014.

[10] Brett T Stewart, Aswin N Venkat, James B Rawlings, Stephen J Wright, and Gabriele Pannocchia. Cooperative distributed model predictive control. *Systems & Control Letters*, 59(8):460–469, 2010.

[11] Alireza Farhadi, Michael Cantoni, and Peter M Dower. Computation time analysis of a distributed optimization algorithm applied to automated irrigation networks. In *Conference on Decision and Control (CDC)*, pages 2193–2199, 2013.

[12] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge University Press, 2004.

[13] Stephen J Wright. *Primal-dual interior-point methods.* Society for Industrial and Applied Mathematics, 1997.

[14] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4):575–601, 1992.

[15] Amir Shahzad, Eric C Kerrigan, and George A Constantinides. A stable and efficient method for solving a convex quadratic program with application to optimal control. *SIAM Journal on Optimization*, 22(4):1369–1393, 2012.