Università di Pisa

Scuola di Ingegneria

Corso di Laurea Magistrale in Ingegneria Robotica e Automazione

Dipartimento di Ingegneria dell'Informazione

Tesi di laurea

A.A. 2013-2014

Cooperative Tracking using Model Predictive Control

Supervisor
Prof. Ing. **Alberto Landi**

Co-Supervisor
Ing. **Gabriele Pannocchia**

Counter-Supervisor
Ing. **Lucia Pallottino**

Candidate
Matteo Razzanelli

Abstract

Prova

Sommario

Prova

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Objectives of thesis

1.2 Organization of thesis

CHAPTER 2

Generalities on distributed system

When talk about control, we usually think to a centralized system that has to do a task, and usually optimizing certain properties. In case we have to manage large scale and dimension systems we require larger resources, computational too. These kind of resources increase exponentially respect to system dimension. Trend of lasts years has been to interconnect each other large quantity of independent subsystems to do the same task as the centralized system. This take an advantage in resources consumption. This is only a linear increasing in waste of resources respect to the number of subsystem.

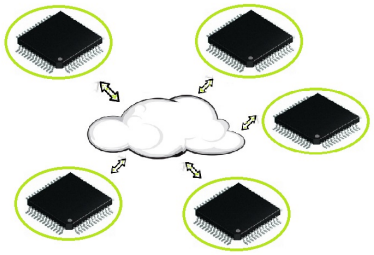

Figure 2.1: Distributed systems example

This new approach has been adopted for a wide variety of subsystems: from small devices to big industrial plant. In microelectronics environment, a lower cost of devices has lead to an integration and interaction among several sensors and actuators. On the other hand, a decentralized design has been necessary to control big processes such as petrol-chemical plants, electrical power grid, water distribution network. The same approach can be done with smaller and more general industrial plant too. Future studies could be concentrate on modelling and decentralizing subsystem in physiological environment.

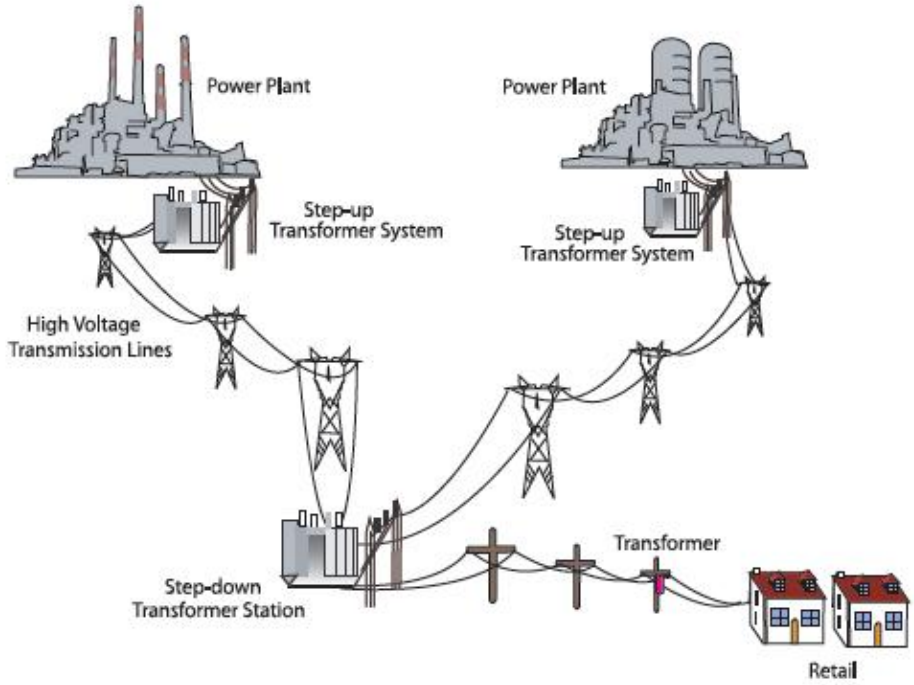

Figure 2.2: Electrical power grid example

From the controllers' point of view, this kind of (decentralized) choice should take care of several aspects. First, sensors and actuators limitations due to a big amount of information on the network. Furthermore the large number of agents suggests to use scalable algorithms. These algorithms should have a $\mathcal{O}(n)$ complexity, in which n is the number of agents. They should take account that available informations are local at single agent or limited.

Figure 2.3: Information flow

It's necessary to find a unique formalism that can be model this class of problems. We want to address a correct abstraction without loss of efficiency. For example, in the next figure, we see several kind of sensors. They are interacting as a network.

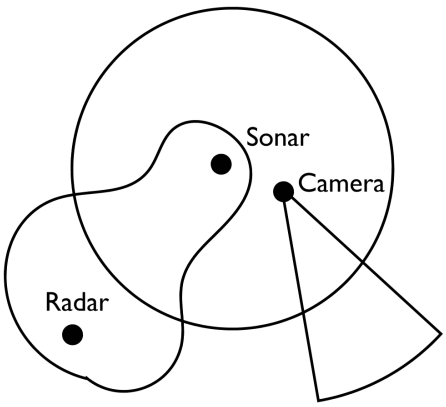

Figure 2.4: Sensors network example

It can be very complex to model sensor's geometry and/or physic. However we can model them through a graph starting from the rule "what has been seen from"

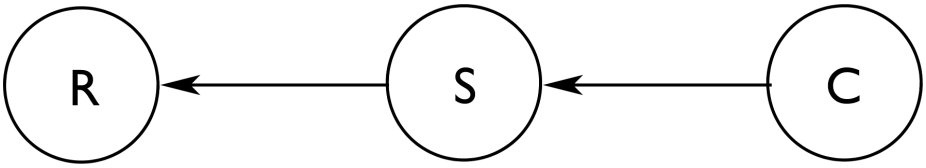

Figure 2.5: Sensors network modelling example

Graph is the most natural way to describe a network of inter agents' systems. What single node represents depends on problem we are treating. So, it's important, first of all, to take a look to graph theory.

2.1 Graph Types

A graph is a possible way to model a physic system. With reference to the Figure 2.5, single node represents a sensor and arc represents the mentioned rule. In this way we have less information about what problem geometry concerns but is simpler than before to define a control law. These laws will be based on topological properties of the network. However, decentralized control laws must take account of single sensor available informations only. Eventually, we have informations of *closer* systems. The closer concept depends on the problem we have to solve. This is due to arcs represent subsystems information flow. The way we manage the flow establishes the network topology and the graph type. We should analyse their features to understand interactions among different and several subsystems models.

2.1.1 Undirected Graphs

Definition 2.1. *A graph $G = (V, E)$ is composed by a finite set of tops (or nodes) V and a set of arcs $E \subset V \times V$ that connect couples of nodes. Two interconnected nodes are called* **adjacent**.

Example 2.2. *Consider $G = (V, E)$ with $V = \{1, 2, 3, 4, 5\}$ and $E = \{(1, 2), (1, 4), (2, 3), (2, 4), (3, 4), (4, 5)\}$*

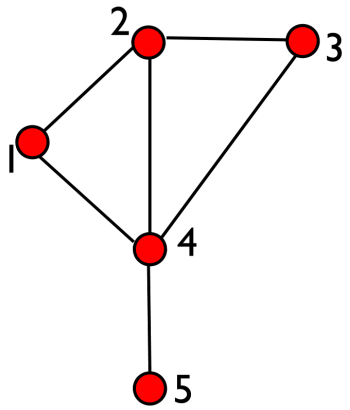

Figure 2.6: Undirected Graphs example

Definition 2.3. *The set $N_i = \{v_j | (v_i, v_j) \in E\} \subset V$ is the adjacent node of i.*

2.1.2 Directed Graphs

In the prevoius section we have analysed a bidirectional information flow. In general this is not true and data go towards graph in a unique sense. The arcs orientation establishes a node influences to an other one.

Definition 2.4. *A graph $G = (V, E)$ is an* oriented *graph if E is composed by* oriented couples nodes. *Arc (v_i, v_j) is an arc from v_i to v_j. We define the **head** of arc v_j and the **tail** of arc v_i.*

Given a directed graph $G = (V, E)$, **input-star** of node v_i is the set

$$S_i^{IN} = \{v_j \in V \,|\, (v_j, v_i) \in E\}$$

while the **output-star** of node v_i is the set

$$S_i^{OUT} = \{v_j \in V \,|\, (v_i, v_j) \in E\}.$$

Example 2.5. *We consider again the previous example but in* directed *mode.*

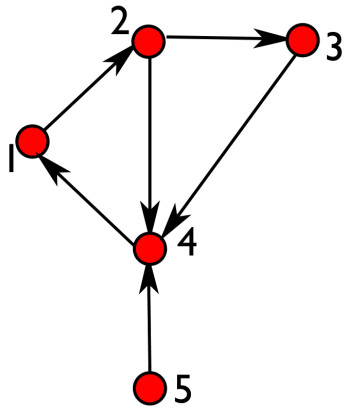

Figure 2.7: Directed Graph example

Consider node 4. The input-star si $S_4^{IN} = \{2, 3, 5\}$, while the output-star is $S_4^{OUT} = \{1\}$.

2.1.3 Directed and Weighted Graphs

Definition 2.6. *Given V and E, let $w : E \to \mathbb{R}$ a function that associate a value (or cost), for each arc. The graph $G = (V, E, w)$ is called **weighted graph**.*

By definition, this type of graph seems to be an useful way to describe interactions among subsystems. Influence of an agent to an other is can be modelled by scaled input, the weight of the arcs. See Figure (2.8) for better understanding.

2.2 Associated Graphs Matrix

How could graph node model a system? Simplify notation, indicating with i the generic node v_i and with

- j the generic node v_j beloging to i $\left(j \in S_i^{OUT}\right)$ output-star.
- k the generic nod v_k belonging to i, $\left(k \in S_i^{IN}\right)$ input-star.

Thus, if we associated to i a state vector x_i and an input vector u_i. The state evolution will be given by

$$x_i^+ = A_i\, x_i + B_i\, u_i$$

where A_i and B_i are transition state and input vector matrices. Note that on i-th agent act exogenous input from arc $(k, i)\,, \forall\, k \in S_i^{IN}$. This input can be the same as agent k but scaled from a weight. So, we will have a further contribution in the dynamic equation that will become

$$x_i^+ = A_i\, x_i + B_i\, u_i + \sum_{k \in S_i^{IN}} B_{ik}\, u_k$$

where the single matrix B_{ki} is the influence of the node k on the i one through the input u_k. Thus, $B_{k,\,i}$ is the weight of the arc.

Definition 2.7. *The set $N_i = \{v_j \mid (v_i\,,\, v_j) \in E\} \subset V$ is the set of i adjacent node.*

Definition 2.8. *For a undirected graph the **degree** $d(v_i)$ of v_i node is the number of v_i adjacent node.*

Definition 2.9. *The graph's **matrix degrees** is a diagonal matrix positive semi-definite of $n \times n$ dimension and n is the number of node in the graph. The i-th element of the diagonal is equal to the i-th node degree $d(v_i)$:*

$$\triangle\,(G) = \begin{bmatrix} d(v_1) & 0 & \cdots & 0 \\ 0 & d(v_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d(v_n) \end{bmatrix}$$

Definition 2.10. *The **adjacent matrix** of a graph G is a symmetric matrix A with $n\times n$ dimension, in which we represent the* adjacent relationship *among the network.*

$$[A(G)]_{ij} = a_{ij} = \begin{cases} 1 & se\ (v_i, v_j) \in E \\ 0 & altrimenti \end{cases}$$

Definition 2.11. *The **Laplacian** of an undirected graph G is*

$$L\left(G\right) = \triangle\left(G\right) - A(G)$$

Definition 2.12. *Let $G = (V,\ E,\ w)$ a weighted and directed graph with arc weight $(v_i, v_j) \in E$ equal to w_{ij}.*

Let A the adjacent matrix where instead of element 1 corresponding to the arc $(v_i, v_j) \in E$ there is the w_{ij} value corresponding to weight of the arc. The i input arcs' degree is given by $deg_{in}\left(v_i\right) = \sum_{j=1}^{n} a_{ji}$, while the output arcs' degree is given by $deg_{out}\left(v_i\right) = \sum_{j=1}^{n} a_{ij}$. The degrees' matrix $\triangle$ of a weighted and directed graph is the diagonal matrix with $deg_{out}\left(v_i\right)$ elements on the diagonal. The Laplacian is defined by $L = \triangle - A$ and the definition ***is consistent*** (not equal to) the that one given by undirected graph considering all weight equal to 1.

Example 2.13. *Let the graph in figure 2.6. The degrees' matrix and the adjacent one are respectively equal to*

$$\triangle = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

So, the Laplacian is

$$L = \triangle - A = \begin{bmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Consider the same graph as before but with random oriented arcs. Associate a weight to each arc. Then we obtain the following new graph. Calculate the relative

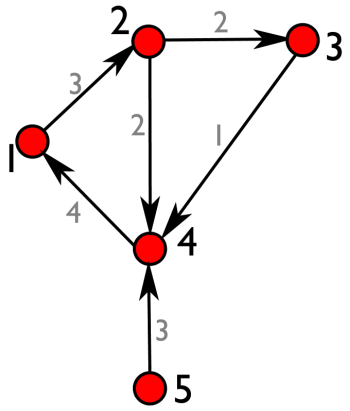

Figure 2.8: Weighted and Directed Graph example

degrees' and adjacent matrices with the new definitions for weighted and directed graphs,

$$\triangle = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \end{bmatrix}$$

Thus, the Laplacian matrix is

$$L = \triangle - A = \begin{bmatrix} 3 & -3 & 0 & 0 & 0 \\ 0 & 4 & -2 & -2 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ -4 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & -3 & 3 \end{bmatrix}$$

These notions should be useful to better understanding how a system can be really implemented by a graph. If every node has a proper state vector, the entire graph evolves in time. Can I control it?

CHAPTER 3

Model Predictive Control

If we want to control a system as a network (graph), formed by a group of independent subsystems, we have to make some annotations. Each subsystem has a proper input, output and state vector. Furthermore, if we want to model a big plant, these are a lots of interactions and (coupled) constraints among subsystems. Hence, if we want to manage the system through conventional feedback control systems like PID, we have to consider some implicit problems.

- **Interaction**, from each manipulated variable to all controlled variables;
  - Usually previous problem can be solved by **decoupler structure** but it could be create problem about robustness.
- The only admissible **constraints** are those on maximum and minimum input values or its variations;
- It's not possible to obtain a **direct optimization** of a certain variable. Only by tuning.

So, it's necessary to have a type of control that is able to handling **multi variable system** (not squared necessarily), or **complicated dynamics** with (**coupled**) **constraints** on controlled and manipulated variables. Nevertheless

it's desirable to have a certain **robustness** or a direct (**multi variable**) **optimization**.

The basic idea of *model predictive control* is to exploit the **process model** (or an estimated measure of it) to compute an equilibrium point at each decision time that satisfies constraints. At this point a **dynamic optimization** module makes a finite horizon prediction on state variables to find an optimal control sequence that minimizes an objective cost function. We then apply the first control action and the cycle restarts.

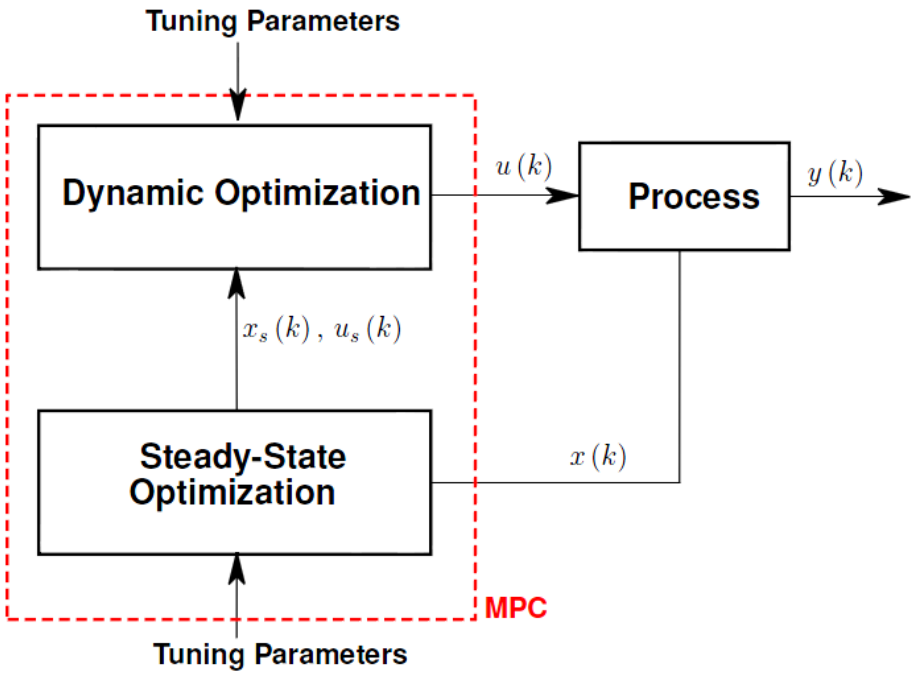

Figure 3.1: MPC architecture

Comparison of model predictive control can be made while driving your car. The classic feedback controller is based on the measured variables and the relative error. It's like when you look at the rear-view mirror and you observe if you act in the right way. You make corrections on the control actions (steer, break, acceleration). The model predictive control predicts the next state and finds an optimal sequence of control action that minimizes a cost function and satisfies certain constraints. It's like when you drive (normally) watching in front of you without going out the street and operating the best possible trajectory to consume less fuel!

Figure 3.2: MPC in real life

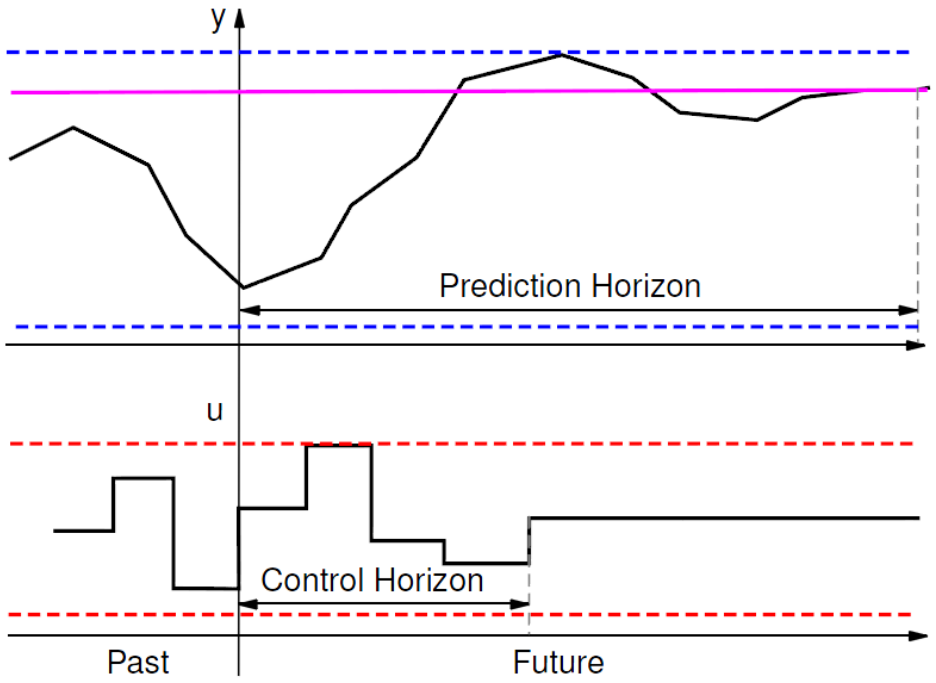

Figure 3.3: MPC Graphical Interpretation

3.1 MPC Introduction

The main goal of this section is to provide the principal tools of predictive control. We'll introduce the optimal regulation problem. State estimation is an other important properties we analyse later in Appendix B. We can say that these two particular tools together create a special form of MPC. They will be useful to understand the general form of MPC in the next section.

3.1.1 Linear Quadratic Problem

We define a discrete time invariant linear system

$$x\left(k+1\right)=A\,x\left(k\right)+B\,u\left(k\right)$$

We consider N steps and the respective action vector sequence

$$\mathbf{u}=\left\{u\left(0\right),u\left(1\right),\ldots,u\left(N-1\right)\right\}$$

We define a cost function to measure trajectory deviation of $x\left(k\right),u\left(k\right)$ from origin trough weight matrices.

$$V\left(x\left(0\right),\mathbf{u}\right)=\frac{1}{2}\sum_{k=0}^{N-1}\left(x\left(k\right)'\,Q\,x\left(k\right)+u\left(k\right)'\,R\,u\left(k\right)\right)+\frac{1}{2}x\left(N\right)'\,P_f\,x\left(N\right) \tag{3.1}$$

$$subject\ to\ x^{+}=A\,x+B\,u$$

Remember that:

- setting large values of Q in comparison to R drive the state to the origin faster at the expense of large control;
- vice-versa, large values of R respect to Q reduce control action penalizing velocity at which the state approaches the origin;
- P_f is a terminal constraint.

We then formulate the linear quadratic optimal control problem

$$\min_{\mathbf{u}}V\left(x\left(0\right),\mathbf{u}\right) \tag{3.2}$$

We assume that Q, P_f, R are *real and symmetric*; Q, P_f are *positive semi-definite*; R is *positive definite*. Such assumptions guarantee that solution os optimal control problem *exists and is unique*.

Optimizing multistage functions

A method to solve the equation 3.2 is an iterative strategy called *dynamic programming* that we briefly exposed. Suppose we have to solve the following three variables problem $(x,\, y,\, z)$

$$\min_{(x,y,z)} f\,(w,x) + g\,(x,y) + h\,(y,z), \qquad w\, fissato$$

It's possible to rewrite problem in this way:

$$\min_{x}\left[f\,(w,x) + \min_{y}\left[g\,(x,y) + \min_{z} h\,(y,z)\right]\right]$$

The iterative strategy compute first the solution of inner problem, proceeds with the intermediate one, finally explicit the outer problem.

Dynamic programming solution

To simplify notation we define stage and final cost of the global function as

$$\frac{1}{2}\left(x\,(k)^{'}\,Q\,x\,(k) + u\,(k)^{'}\,R\,u\,(k)\right) \triangleq l\;(x\,(k)\;,\,u\,(k))$$

$$\frac{1}{2} x\,(N)^{'}\,P_f\,x\,(N) \triangleq l_N\,(x\,(N))$$

It's possible to apply this technique on LQ problem. As $x(0)$ is given it's convenient to solve it with *backward* dynamic programming. If we take the problem (3.1) and optimize it on $u(N-1)$ and $x(N)$ we obtain

$$\min_{u(0),x(1),\ldots,u(N-2),x(N-1)} \sum_{k=0}^{N-2} l\;(x\,(k)\;,\,u\,(k)) +$$

$$\underbrace{\min_{u(N-1),x(N)} [l\;(x\,(N-1)\;,\,u\,(N-1)) + l_N\;(x\,(N))]}_{s.t.\; x(N)=A\,x(N-1)+B\,u(N-1)}$$

Solving first the second addendum we obtain the optimal control at step *N-1*, defined as

$$u^0(N-1) = K_N(N-1)\,x(N-1) \quad \text{with}$$

$$K_N(N-1) = -\left(B^{'}P_f\,B + R\right)^{-1} B^{'}P_f\,A$$

We repeat the process obtaining the backward *Riccati iteration*

$$u^0(N) = K_N(N)\,x(N), \quad \text{with}$$

$$K_N(N) = -\left(B^{'}\Pi(k+1)\,B + R\right)^{-1} B^{'}\Pi(k+1)\,A,$$

$$\Pi(k-1) = Q + A^{'}\Pi(k)\,A - A^{'}\Pi(k)\,B\,(B^{'}\Pi(k)\,B+R)^{-1}B^{'}\Pi(k)\,A, \quad \Pi(N) = P_f$$

Terminal condition substitute the initial one because it will treated the backward iteration. The advantage of this technique is that optimal control is computed in feedback way, thus in closed-loop.

The infinite horizon LQ problem

"In the engineering literature it is often assumed (tacitly and incorrectly) that a system with optimal control law is necessarily stable".[1]

Infact, if final time N tends to infinity and one of the state it's not controllable and/or stabilizable, the corresponding performance index tends to infinity. Let

$$K\,(0) = K \quad x\,(0) = x$$

It can be shown that in special case, if we tends k to infinity, we converge directly to the infinite horizon control law K_∞. With this motivation , we are lead to consider directly the infinite horizon case

$$V\,(x\,(0)\,,\mathbf{u}) = \frac{1}{2}\sum_{k=0}^{\infty}\left(x\,(k)^{'}\,Q\,x\,(k) + u\,(k)^{'}\,R\,u\,(k)\right)$$

In considering the infinite horizon problem, we first restrict attention to systems for which there exist input sequences that give bounded cost If $(A\,,B)$ is controllable the solution to optimal problem (3.2) exists and is unique for each x. Thus, the general control law $\kappa\,(\cdot)$ computed in infinite horizon case, $\kappa_\infty\,(\cdot)$, is defined as $u = \kappa_\infty\,(x)$. We apply only the first action of the optimal control sequence so we define $\kappa_\infty\,(x) = u^0\,(x) = \mathbf{u}^0\,(0\,;\,x)$.

Theorem 3.1. *For each couple* $(A,\,B)$ controllable *and* $(Q,\,R)$ *positive definite exist a solution, positive definite of infinite horizon algebraic Riccati e quation such that closed-loop matrix* $(A + B\,K)$ *be strictly Hurwitz and closed-loop system*

$$x^{+} = A\,x + B\,\kappa_\infty\,(x)$$

converge in the origin.

[1] R. E. Kalman. Contributions to the theory of optimal control. Bull. Soc.Math.Mex., 5:102–119, 1960.

From previous section the optimal solution is found through Riccati iteration, and control law and relative cost are

$$u^0(x) = K\,x, \quad V^0(x) = \frac{1}{2}x^{'}\Pi x$$

where

$$K = -\left(B^{'}\Pi\,B + R\right)^{-1} B^{'}\Pi\,A,$$

is defined as *Kalman gain* and

$$\Pi = Q + A^{'}\Pi\,A - A^{'}\Pi\,B\,(B^{'}\Pi\,B + R)^{-1}B^{'}\Pi\,A, \quad \Pi(\infty) = \Pi \tag{3.3}$$

Proving Lemma 3.1, has shown that for $(A,\,B)$ controllable and $Q,\,R > 0$, a positive definite solution to the discrete algebraic Riccati equation (DARE), (3.3), exists and the eigenvalues of $(A + B\,K)$ are asymptotically stable for the K corresponding to this solution.

Figure 3.4 shows us how is implemented a discrete time infinite horizon LQR architecture

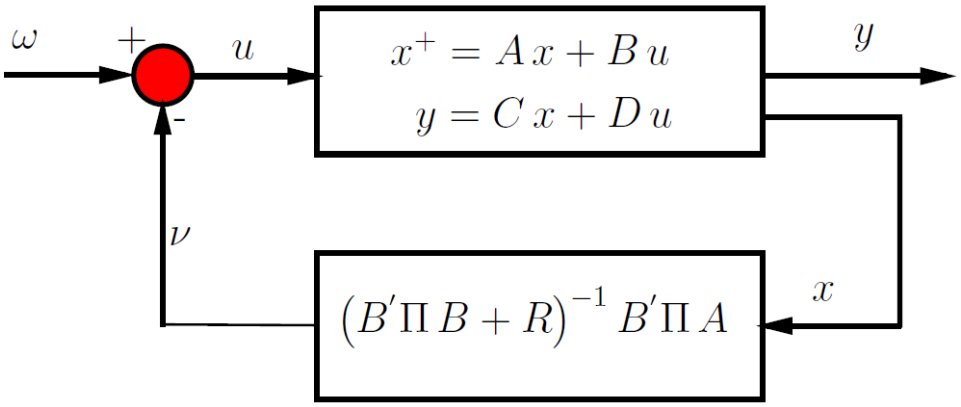

Figure 3.4: LQR architecture

We remember that in linear systems asymptotic convergence coincide with asymptotic stability. To ensure all the unstable states are present in V it's sufficient that (A,G) is detectable.

3.2 Main theory

In general case, Model Predictive Control, have a more general form we now consider. We will pointing out difference among standard controller and MPC.

- control action is obtained by solving *online*, at each sampling instant, a *finite horizon* optimal control problem in which the initial state is the current state of the plant (usually we compute an *offline* control law);
- If the current state is x, MPC obtains, by solving an *open-loop* optimal control problem for this initial state, a *specific* control action u to apply to the plant;
  - only the first control action is applied to the plant.
  - open-loop optimization can be solved rapidly enough; so we can manage large dimension system or hard constraints' ones;
  - the only *feedback* block is state estimator
  - MPC computes the value $\kappa(x)$ of the optimal receding horizon control law for the current state x, while DP yields the control law $\kappa(\cdot)$ that can be used for *any* state.
- difference between MPC and the others standard controller are optimization *constraints*.

3.2.1 General Formulation

We consider systems of the form

$$\frac{dx}{dt} = f\ (x,\, u)$$

For this class of systems, the control law with arguably the best closed-loop properties is the solution to the following infinite horizon, constrained optimal control problem. The cost is defined to be

$$V_{\infty}(x,\, u(\cdot)) = \int_0^{\infty} l(x(t),\, u(t))\, dt$$

where $x(t)$ and $u(t)$ satisfy $\dot{x} = f\ (x,\, u)$ and $l(\cdot)$ is the stage cost. The optimal control problem $\mathbb{P}(x)$ is defined as

$$\mathbf{min}_{u(\cdot)} V_{\infty}(x,\, u(\cdot))$$

subject to

$$\dot{x} = f\ (x,\, u) \qquad x\,(0) = x_0$$
$$u\,(t) \in \mathbb{U} \quad x\,(t) \in \mathbb{X}, \quad \text{for each},\, t \in (0,\, \infty)$$

If $l\,(\cdot)$ is positive definite, the goal of the regulator is to steer the state of the system to the origin. We denote the solution to this problem (when it exists) and the optimal value function respectively as

$$u^0_\infty\,(\cdot;\, x)\,,\ and \quad V^0_\infty\,(x)$$

(Superscripts 0 indicates optimal value). The closed-loop system under this optimal control law evolves as

$$\frac{dx\,(t)}{dt} = f\ \left(x\,(t)\,,\, u^0_\infty\,(t\,;\, x)\right)$$

If $l\,(\cdot)$ and $f\,(\cdot)$ are differentiable and certain growth assumptions are satisfied and there are no state constraints, then a solution to $V^0_\infty\,(\cdot)$ exists for all x; $V^0_\infty\,(\cdot)$ is differentiable and satisfies

$$\dot{V}^0_\infty\,(x) = -l\ \left(x,\, u^0_\infty\,(0\,;\, x)\right)$$

Using the formulation above with upper and lower bounds on $V^0_\infty\,(\cdot)$ enables global asymptotic stability of the origin to be established. Although the control law $u^0_\infty\,(0;\cdot)$ provides excellent closed-loop properties, there are several impediments to its use.

- A feedback, rather than an open-loop, control is usually necessary because of uncertainty;
- Solution of the problem yields the optimal control for the state x but does not provide a control law;
- Dynamic programming may, in principle, be employed, but is generally impractical if the state dimension and the horizon are not small.

If we use MPC approach in which we generate the on-line optimal control value for a specific state x, rather than for all x, the problem remains intractable. In this section we restrict to discrete time invariant linear system with finite horizon control problem. Thus, let

$$x^+ = f\ (x,\, u) \tag{3.4}$$

a discrete time non-linear differential equation system. Function $f(\cdot)$ is assumed to be continuous and to satisfy $f(0, 0) = 0$, i.e. 0 is an equilibrium point. If the initial state $x(0) = x_0$ each solution of 3.4 satisfies

$$x(k+1) = f\ (x(k)\,,\, u(k)) \qquad k = 0,\, 1,\, \ldots$$

Solution of 3.4 at each instant k is denoted by $\phi(k\,;\, x,\, \mathbf{u})$ that depends only by control sequence $\mathbf{u} = \{u(0)\,,\, u(1)\,, \ldots, u(N-1)\}$ and the initial state x_0.

Recalling Figure 3.1, Dynamic Optimization module has the main task to find the optimal control sequence $\mathbf{u}$. Let us zoom it.

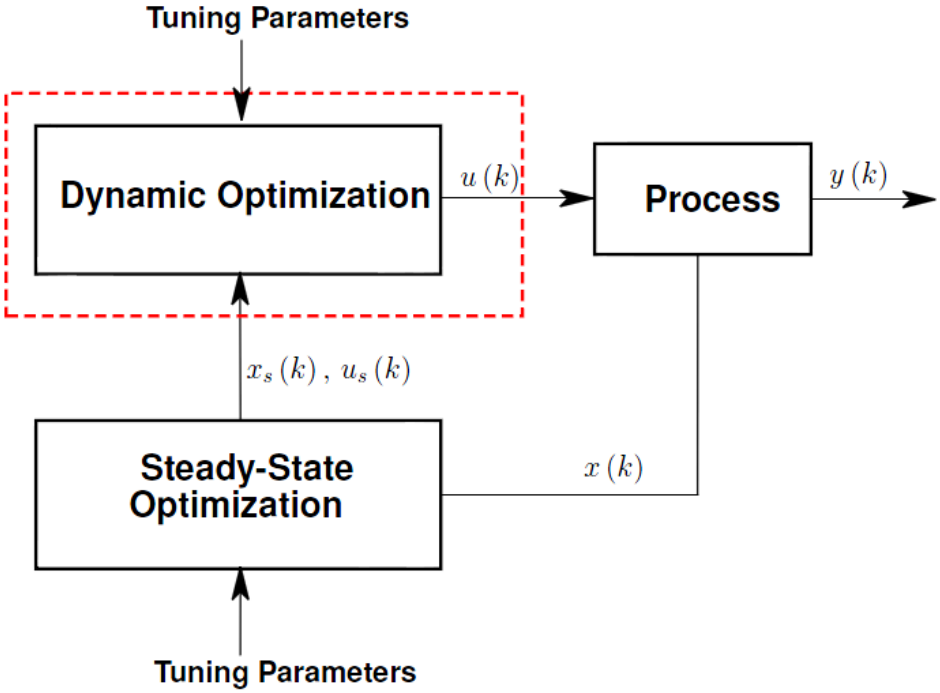

Figure 3.5: Dynamic Optimization Module

For now, $(x_s(k)\,,\, u_s(k))$ can be considered as the steady state and input such that $y_s(k) = 0$.

Proposition 3.2. ***Continuous system solution.***

Suppose function $f\ (\cdot)$ continuous. Then for each $k \in \mathbb{I}$, function $(x,\, \mathbf{u}) \to \phi(k\,;\, x,\, \mathbf{u})$ is continuous.

The system 3.4 is subject to hard constraints which may take the form

$$u(k) \in \mathbb{U} \quad x(k) \in \mathbb{X}, \quad \forall k \in \mathbb{I}_{\geq k} \tag{3.5}$$

The next ingredient of the optimal control problem is the cost function. Practical considerations require that the cost be defined over a finite horizon N to ensure the resultant optimal control problem can be solved sufficiently rapidly to permit effective control. We consider initially the regulation problem where target state is the origin. Furthermore the initial instant decision is irrelevant, hence we indicate simply $\mathbf{u}^0(x)$ and $\mathbf{x}^0(x)$ to indicate the best control action for state x and the optimal state sequence. The optimal control problem $\mathbb{P}_N(x)$ may be expressed as minimization of

$$\sum_{k=0}^{N-1} l\left(x(k), u(k)\right) + V_f\left(x(N)\right)$$

with respect to the decision variables $(\mathbf{x}, \mathbf{u})$. The optimal control and state sequence are defined as

$$\mathbf{u}^0(x) = \{u(0), u(1), \ldots, u(N-1)\}$$
$$\mathbf{x}^0(x) = \{x(0), x(1), \ldots, x(N)\}$$

For the purpose of analysis is preferable to constrain the state sequence $\mathbf{x}$ a priori to be a solution of $x^+ = f(x, u)$ enabling us to express the problem in the equivalent form of minimizing, with respect to the decision variable $\mathbf{u}$, a cost that is purely a function of the initial state $\mathbf{x}$ and the control sequence $\mathbf{u}$. This formulation is possible since the state sequence $\mathbf{x}$ may be expressed, via the difference equation $x^+ = f(x, u)$, as a function of $(\mathbf{x}, \mathbf{u})$. The cost will becomes

$$V_N(x, \mathbf{u}) = \sum_{k=0}^{N-1} l\left(x(k), u(k)\right) + V_f\left(x(N)\right) \tag{3.6}$$

where, now, $x(k) := \phi(k; x, \mathbf{u})$ for all $k \in \mathbb{I}_{0:N}$. Similarly the hard constraints on input and states, togheter with an additional terminal constraint

$$x(N) \in \mathbb{X}_f$$

where $\mathbb{X}_f \subseteq \mathbb{X}$ impose an implicit constraint on the control sequence of the form

$$\mathbf{u} \in \mathcal{U}_N(x)$$

in which the control constraint set $\mathcal{U}_N(x)$ is the set of control sequences $\mathbf{u} := \{u(0), u(), \ldots, u(N-1)\}$ satisfying the state and control constraints. It is therefore defined by

$$\mathcal{U}_N(x) := \{\mathbf{u} | (x, \mathbf{u}) \in \mathbb{Z}_N\}$$

in which the set $\mathbb{Z}_N \subset \mathbb{R}^n \times \mathbb{R}^{Nm}$ is defined by

$$\mathbb{Z}_N := \{(x, \mathbf{u}) \,|\, u(k) \in \mathbb{U},\ \phi(k; x\,\mathbf{u}) \in \mathbb{X},\ \forall k \in \mathbb{I}_{0:N-1},\ \phi(N; x\,\mathbf{u}) \in \mathbb{X}_f\}$$

The optimal control problem $\mathbb{P}_N(x)$ may be expressed as

$$\mathbb{P}_N(x): \qquad V_N^0(x) := \min_{\mathbf{u}} \{V_N(x, \boldsymbol{u}) \,|\, \mathbf{u} \in \mathcal{U}_N(x)\} \tag{3.7}$$

Problem $\mathbb{P}_N(x)$ is a parametric optimization in which the decision variable is $\mathbf{u}$, and both the cost and the constraint set depend on the paramter x. The set $\mathbb{Z}_N$ is the set of admissible $(x, \mathbf{u})$, i.e., the set of $(x, \mathbf{u})$ for which $x \in \mathbb{X}$ and the constraints of $\mathbb{P}_N(x)$ are satisfied. It can be shown that $\mathcal{X}_N$ is the set of states i $\mathbb{X}$ for which $\mathbb{P}_N(x)$ has a solution

$$\mathcal{X}_N := \{x \in \mathbb{X} \,|\, \mathcal{U}_N(x) \neq \emptyset\}$$

We assume, without further comment, that the following standing conditions are satisfied in the sequel.

Assumption 3.3. ***(Continuity of system and cost).*** *The function $f : \mathbb{X} \times \mathbb{U} \to \mathbb{R}^n$, $l : \mathbb{X} \times \mathbb{U} \to \mathbb{R}_{\geq 0}$ and $V_f : \mathbb{X}_f \to \mathbb{R}_{\geq 0}$ are continuous and $f(0, 0) = 0$, $l(0, 0) = 0$ and $V_f(0) = 0$.*

Assumption 3.4. ***(Properties of constraint sets).*** *The sets $\mathbb{X}$ and $\mathbb{X}_f$ are closed, $\mathbb{X}_f \subseteq \mathbb{X}$ and $\mathbb{U}$ are compact; each set contains the origin.*

Proposition 3.5. ***(Existence of solution to optimal control problem).*** *Suppose assumptions 3.3 and 3.4 hold. Then*

- *The function $V_N(\cdot)$ is continuous in $\mathbb{Z}_N$;*
- *For each $x \in \mathcal{X}_N$, the control constraint set $\mathcal{U}_N(x)$ is compact;*
- *For each $x \in \mathcal{X}_N$, a solution to $\mathbb{P}_N(x)$ exists.*

$\mathcal{X}_N$ is the set of admissible starting states that satisfy hard constraints and exists a control sequence such that $\phi(N;\, x,\, \mathbf{u}) \in \mathbb{X}_f$

The solution is

$$\mathbf{u}^0(x) = \arg\min_{\mathbf{u}} \{V_N(x,\, \mathbf{u}) \,|\, \mathbf{u} \in \mathcal{U}_N(x)\} \tag{3.8}$$

In MPC, the control applied to the plant is the first element of $\mathbf{u}^0(x)$, $u^0(0;\, x)$. Although MPC computes $\mathbf{u}^0(x)$ only for a specific value of x, may be used for any x in which $\mathbb{P}_N(x)$ is admissible, obtaining the implicit state feedback control law $\kappa_N(\cdot)$ defined by

$$\kappa_N(x) := u^0(0;\, x), \qquad x \in \mathcal{X}_N$$

Theorem 3.6. ***(Continuity of value function and control law).*** *Suppose that assumptions 3.3 and 3.4 hold.*

- *Suppose there is no state constraints such that $\mathbb{X} = \mathbb{X}_f = \mathbb{R}^n$. Then the objective function $V_N^0 : \mathcal{X}_N \to \mathbb{R}$ is continuous and $\mathbb{X}_N = \mathbb{R}^n$;*
- *Suppose $f(\cdot)$ is linear as $x^+ = A\,x + B\,u$ and that the state and control constraints sets are polyhedral (represented as inequality systems). Then function $V_N^0 : \mathcal{X}_N \to \mathbb{R}$ is continuous;*
- *If, in addiction, the optimal control problem solution $\mathbf{u}^0(x)$ is unique at each $x \in \mathcal{X}_N$, then the implicit MPC control law $\kappa_N(\cdot)$ is continuous.*

We take for granted, stability theory we have reported in Appendix A, to apply them to Model Predictive Control.

3.2.2 Stability in MPC

Our task in this chapter is to find a function $V(\cdot)$ with properties (A.2) and (A.3) of theorem (A.11). The origin is asymptotically stable with a region of attraction X for the system $x^+ = f(x)$ if there exists a Lyapunov function V, a positive invariant set X, two $\mathcal{K}_\infty$ functions $\alpha_1(\cdot)$, $\alpha_2(\cdot)$, and a positive definite function $\alpha_3(\cdot)$ satisfying:

- $V(x) \geq \alpha_1(|x|)$
- $V(x) \leq \alpha_2(|x|)$

- $V\left(f\left(x\right)\right) \leq V\left(x\right) - \alpha_3\left(|x|\right)$

for all $x \in X$. Our task is to find a function $V\left(\cdot\right)$ with these properties for MPC system $x^+ = f\ \left(x,\, \kappa_N\left(x\right)\right)$. This suggests the use of V_N^0, the value function for the finite horizon optimal control problem whose solution yields the model predictive controller, as a Lyapunov function. Since, as is often pointed out, optimality does not imply stability, this property does not usually hold when the horizon is finite. One of the main task of this section is show that if $l\left(\cdot\right)$, $V_f\left(\cdot\right)$ and $\mathbb{X}_f$ of the finite horizon optimal control problem are chosen appropriately, then A.11 conditions are satisfied. In the sequel we shall denote as $\mathbf{u}^0\left(x\right)$ any optimal control sequence and $\kappa_N\left(x\right)$ the first element of this sequence, $u^0\left(0;\, x\right)$.

Stabilizing Condition

In case there are no state constraint ($\mathbb{X} = \mathbb{X}_f = \mathbb{R}^n$) we can be shown that $V_f\left(\cdot\right)$ satisfy

$$V_f\left(f\left(x,\, u\right)\right) - V_f\left(x\right) + l\left(x,\, u\right) \leq 0$$

so, if function $l\left(\cdot\right)$ is positive definite, is a global control Lyapunov function. Thus V_N^0 has the desired descent property and global asymptotic stability of the origin for the system $x^+ = f\ \left(x,\, \kappa_N\left(x\right)\right)$ under MPC may be established. We consider the case when state and control constraints are present. MPC is stabilizing if a global Control Lyapunov Function (CLF) is employed as the terminal cost. A global CLF is seldom available, however, either because the system is non-linear or because constraints are present. Hence, we must set our sights lower and employ as our terminal cost function V_f a local CLF, one that is defined only on a neighbourhood $\mathbb{X}_f$ of the origin where $\mathbb{X}_f \subseteq \mathbb{X}$. A consequent requirement is that the terminal state must be constrained, explicitly or implicitly, to lie in $\mathbb{X}_f$. Our stabilizing condition now takes the form:

Assumption 3.7. ***(Basic stability assumption).***

$$\min_{\mathbf{u}\in\mathbb{U}} \left\{V_f\left(f\left(x,\, u\right)\right) + l\left(x,\, u\right) \,|\, f\left(x,\, u\right) \in \mathbb{X}_f\right\} \leq V_f\left(x\right), \quad \forall\, x \in \mathbb{X}_f$$

This assumption implies the following other one.

Assumption 3.8. ***(Implied invariance assumption).*** *The set $\mathbb{X}_f$ is control invariant for the system $x^+ = f\ \left(x,\, u\right)$, i.e., there exists $u \in \mathbb{U}$ such that $x^+ = A\, x + B\, u \in \mathbb{X}_f$*

Assumptions (3.7) and (3.8), specify properties which, if possessed by the terminal cost function and terminal constraint set, enable us to employ the value function $V_N^0(x) = V_N\left(x, \mathbf{u}^0(x)\right)$ for the optimal control problem $\mathbb{P}_N$ as a Lyapunov function. therefore if (3.3), (3.4), (3.7) and (3.8) are verified the important descent and monotonicity properties of V_N^0 are established:

Lemma 3.9. ***(Optimal cost decrease).***

$$V_N^0\left(f\left(x, \kappa_N(x)\right)\right) \leq V_N^0(x) - l\left(x, \kappa_N(x)\right) \quad \forall x \in \mathcal{X}_N \tag{3.9}$$

Lemma 3.10. ***(Monotonicity of the value function).***

$$\begin{aligned} V_{j+1}^0(x) &\leq V_j^0(x) \qquad \forall x \in \mathcal{X}_N, \quad \forall j \in \mathbb{I}_{0:N-1} \\ V_N^0(x) &\leq V_f(x) \qquad \forall x \in \mathbb{X}_f \end{aligned} \tag{3.10}$$

We report here a route of stability to simplify

Theory Step	Description
3.3	Continuity of system and cost
3.4	Properties of constraint sets
3.5	Existence of a solution
3.7	Basic stability assumption
3.8	Implied invariance assumption
3.9	Optimal cost decrease
3.10	Monotonicity of the value function

Table 3.1: Route of Stability

Lemma 3.9 shows that the value function $V_N^0(\cdot)$ has a descent property that makes it a suitable candidate for a Lyapunov function that may be used to establish stability of the origin for a wide variety of MPC systems. To proceed, we postulate two alternative conditions on the stage cost $l(\cdot)$ and terminal cost $V_f(\cdot)$ required to show that $V_N^0(\cdot)$ has the properties given in Appendix A, which are sufficient to establish stability of the origin. Our additional assumption is:

Assumption 3.11. ***(Bounds on stage and terminal costs).*** *The stage cost $l(\cdot)$ and terminal cost $V_f(\cdot)$ satisfy*

$$\begin{gathered} l(x, u) \geq \alpha_1(|x|) \quad \forall x \in \mathcal{X}_N, \quad \forall u \in \mathbb{U} \\ V_f(x) \leq \alpha_2(|x|) \quad \forall x \in \mathbb{X}_f \end{gathered}$$

in which $\alpha_1(\cdot)$ *e* $\alpha_2(\cdot)$ *are functions* $\mathcal{K}_\infty$ *or*

$$l(x, u) \geq c_1 |x|^a \quad \forall x \in \mathcal{X}_N, \quad \forall u \in \mathbb{U}$$
$$V_f(x) \leq c_2 |x|^a \quad \forall x \in \mathbb{X}_f$$

for some $c_1, c_2, a, > 0$

In the situations in which $\mathbb{X}_f$ has no origin in its interior, we don't establish an upper bound for $V_N^0(\cdot)$. From the assumption 3.7 and 3.8 and resort to the following assumption.

Assumption 3.12. ***(Weak controllability).*** *There exists a* $\mathcal{K}_\infty$ *function,* $\alpha(\cdot)$ *such that* $V_N^0(x) \leq \alpha(|x|) \quad \forall x \in \mathcal{X}_N$.

3.2.3 Summary, MPC Stability Theorems

In the sequel we apply the previous results to establish asymptotic or exponential stability of a wide range of MPC systems. To facilitate application, we summarize these results and some of their consequences in the following theorem.

Theorem 3.13.

Suppose that Assumptions 3.3, 3.4,3.7, 3.8, and 3.11(a), are satisfied and that $\mathcal{X}_N = \mathbb{X}_f = \mathbb{R}^n$ *so that* $V_f(\cdot)$ *is a global CLF. Then the origin is globally asymptotically stable for* $x^+ = f(x, \kappa_N(x))$. *If, in addition, Assumption 3.11(b) is satisfied, then the origin is globally exponentially stable.*

Suppose that Assumptions 3.3, 3.4,3.7, 3.8, and 3.11(a), are satisfied and that $\mathbb{X}_f$ *contains the origin in its interior. Then the origin is asymptotically stable with a region of attraction* $\mathcal{X}_N$ *for the system* $x^+ = f(x, \kappa_N(x))$. *If, in addition, Assumption 3.11(b) is satisfied, and* $\mathcal{X}_N$ *is bounded, then the origin is exponentially stable with a region of attraction* $\mathcal{X}_N$ *for the system* $x^+ = f(x, \kappa_N(x))$*; if* $\mathcal{X}_N$ *is unbounded, the the origin is exponentially stable with a region of attraction that is any sublevel set of* $V_N^0(\cdot)$.

Suppose that Assumptions 3.3, 3.4,3.7, 3.8, and 3.12 are satisfied and that $l(\cdot)$ *satisfies* $l(x, u) \geq \alpha_1(|x|)$ *for all* $x \in \mathcal{X}_N$, *all* $u \in \mathbb{U}$, *where* α_1 *is a* $\mathcal{K}_\infty$ *function. Then the origin is asymptotically stable with a region of attraction* $\mathcal{X}_N$ *for the system* $x^+ = f(x, \kappa_N(x))$. *If* $l(\cdot)$ *satisfies* $l(x, u) \geq c_1 |x|^a$ *for all for all* $x \in \mathcal{X}_N$, *all* $u \in \mathbb{U}$ *and Assumption 3.12 is satisfied with* $\alpha(r) = c_2 r^a$ *for some* $c_1 > 0$, $c_2 > 0$ *and* $a > 0$, *then the origin is exponentially stable with a region of attraction* $\mathcal{X}_N$ *for the system* $x^+ = f(x, \kappa_N(x))$.

Suppose that Assumptions 3.3, 3.4,3.7 and 3.8 are satisfied, that $l(\cdot)$ *satisfies* $l(x, u) \geq c_1 |x|^a + c_1 |u|^a$ *and Assumption 3.12 is satisfied with* $\alpha(r) = c_2 r^a$ *for some* $c_1 > 0$, $c_2 > 0$ *and* $a > 0$. *Then* $\kappa_N(x) \leq c|x|$ *for all* $x \in \mathcal{X}_N$ *where* $c = \left(\frac{c_2}{c_1}\right)^{\frac{1}{a}}$.

We have not yet made any assumptions on controllability (stabilizability) or observability (detectability) of the system 3.4. being controlled, which may be puzzling since such assumptions are commonly required in optimal control to, for example, establish existence of a solution to the optimal control problem. The reasons for this omission are that such assumptions are implicitly required, at least locally, for the basic stability Assumption 3.7 and that we restrict attention to $\mathcal{X}_N$, the set of states that can be steered to $\mathbb{X}_f$ in N steps satisfying all constraints. For example, one version of MPC uses a target set $\mathbb{X}_f = \{0\}$ so that the optimal control problem requires determination of an optimal trajectory terminating at the origin; clearly some assumption on controllability to the origin such as Assumption 3.12 is required. Similarly, f the system being controlled is linear, and the constraints polytopic or polyhedral, a common choice for $\mathbb{X}_f$ is the maximal invariant constraint admissible set for a controlled system where the controller is linear and stabilizing. The terminal constraints set $\mathbb{X}_f$ is then the set $\{x \mid x(i) \in \mathbb{X}, Kx(i) \in \mathbb{U}\}$ where $x(i)$ is the solution at time i of $x^+ = (A + BK)x$ and $u = \kappa_f(x) = Kx$ is a stabilizing control law. Stabilizability of the system being controlled is then required. Detectability assumptions also are required, mainly in proofs of asymptotic or exponential stability.

We report a simple table with stability Assumptions path for better understanding

Assumption	Description
3.3	Continuity of system and cost
3.4	Properties of constraint sets
3.7	Basic stability assumption
3.8	Implied invariance assumption
3.11	Bounds on stage and terminal costs
3.12	Weak controllability

Table 3.2: Stability Assumptions

3.2.4 Example of MPC: LTI systems

We want to formulate an MPC example system to show the basic route to prove stability. We describe a simple MPC prototype. Consider a Linear Quadratic system with Control and State Constraints. Given the current state $x(0) = x$, solve for input sequence $\mathbf{u} = \{u(0; x), u(1; x), \ldots, u(N-1; x)\}$ the following optimal control problem

$$\mathbb{P}_N(x): \qquad \min_{\mathbf{u}} V_N(x, \mathbf{u})$$

subject to

$$\begin{gathered} x^+ = A x + B u \\ x(k) \in \mathbb{X} \quad k = 0, \ldots, N-1 \\ u(k) \in \mathbb{U} \quad k = 0, \ldots, N-1 \\ x(N) \in \mathbb{X}_f \end{gathered}$$

Cost function is

$$V_N(x, \mathbf{u}) = \sum_{k=0}^{N-1} l(x(k), u(k)) + V_f(x(N)) \qquad l(x, u) = x' Q x + u' R u$$

and chose as terminal cost function, rather than terminal constraint set $\mathbb{X}_f$

$$V_f(x) = x' P x$$

as the solution of DARE. Given the optimal solution sequence $\mathbf{u}^0(x)$, function of *current* state, denote as the **implicit MPC control** law

$$\kappa_N(x) = u^0(0; x)$$

So the closed-loop system is

$$x^+ = A x + B \kappa_N(x)$$

Notice that $\kappa_N : \mathcal{X}_N \to \mathbb{U}$ is not linear. The basic route to prove stability is:

- Show that $V_N^0(\cdot)$ is a Lyapunov function for $x^+ = f(x) = A x + \kappa_N(x)$
- Show that the feasibility set, $\mathcal{X}_N$, is positively invariant
- Control invariance of $\mathbb{X}_f$. For every $x \in \mathbb{X}_f$, there exists $u \in \mathbb{U}: x^+ = A x + B u \in \mathbb{X}_f$. From here $V_f(x^+) - V_f(x) \leq -l(x, u)$.

So stability proof derive from optimal cost decrease lemma.

Lemma 3.14. *For all $x \in \mathcal{X}_N$, there holds:*

$$V_N^0\left(A\,x + B\kappa_N\left(x\right)\right) - V_N^0\left(x\right) \leq -l\left(x,\, \kappa_N\left(x\right)\right)$$

3.3 Tracking for MPC

Recalling node state vector, can we move measured outputs of an entire graph or of a single agent to a target? Can we have different target for each agent? It is a standard objective in applications to move the measured outputs of a system to a specified setpoints (y_{target}). We consider here a linear invariant discrete time system

$$\begin{aligned} x^+ &= A\,x + B\,u \\ y &= C\,x + D\,u \end{aligned}$$

where matrix $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times m}$, while vector $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$, $x \in \mathbb{R}^n$. From here forward a generic $\mathbf{0}_{n,m}$ represents a matrix of zeros $\in \mathbb{R}^{n\times m}$ and I_n represents a matrix $\in \mathbb{R}^{n\times n}$. The system is subject to hard constraints on state and control

$$x \in \mathbb{X} \subset \mathbb{R}^n,\, u \in \mathbb{U} \subset \mathbb{R}^m$$

Denote a steady state and input of the system as $(x_s,\, u_s)$. We also impose the requirements that the steady state satisfies

$$C\,x_s + D\,u_s = y_s$$

The following equations are satisfied:

$$\underbrace{\begin{bmatrix} A - I_n & B & \mathbf{0}_{p,1} \\ C & D & -I_p \end{bmatrix}}_{M} \begin{bmatrix} x_s \\ u_s \\ y_s \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n,1} \\ \mathbf{0}_{p,1} \end{bmatrix} \tag{3.11}$$

Assumption 3.15. *The pair $(A,\, B)$ is stabilizable and the state is measured at each sampling time.*

Assumption 3.15 is necessary and sufficient to ensure that the set of equations has a no-trivial solution. This one belongs to the kernel of M,

$$\begin{bmatrix} x_s & u_s & y_s \end{bmatrix}^T \in \ker(\,M\,)$$

It can be parametrized as

$$\begin{cases} \begin{bmatrix} x_s & u_s \end{bmatrix}^T = M_\theta \, \theta \\ \\ y_s = N_\theta \, \theta \end{cases} \tag{3.12}$$

where $M_\theta \in \mathbb{R}^{(n+m)\times n_\theta}$, $N_\theta \in \mathbb{R}^{p\times n_\theta}$ and $\theta \in \mathbb{R}^{n_\theta}$ where θ is a parameter vector which characterizes any solution of (3.12). We observe that

$$\begin{bmatrix} M_\theta \\ N_\theta \end{bmatrix}$$

is a matrix in which columns are a kernel base of M . This parametrization allows us to characterize the subspace of steady states and inputs by a minimal number of variables (θ), which simplifies further calculations necessary for the derivation of the proposed controller. If we have a solution (x_s, u_s), we can define the deviation variables as

$$\begin{aligned} \tilde{x} &= x - x_s \\ \tilde{u} &= u - u_s \end{aligned} \tag{3.13}$$

The zero regulation problem applied to the system in deviation variables finds control sequence $\tilde{u}$ that takes $\tilde{x}$ to zero, or, equivalently, which takes x to x_s. In that case, we will have

$$C \, x_s = y_s$$

which is the goal of set-point tracking problem. After solving the regulation problem in deviation variables, we find an optimum input value, u^* . Thus, the input applied to the system is

$$u = u^* + u_s$$

For tracking problem has a solution, we require at least as many inputs as outputs with set points.

In operational situation we have to solve the following steady-state optimal target problem

$$\min_{x_s,u_s} V_{ss}(\cdot) = \|y_s - y_{target}\|_T^2$$
$$s.t. \begin{cases} x_s &= A\,x_s + B\,u_s \\ y_s &= C\,x_s + D\,u_s \\ x_s &\in \mathbb{X} \\ u_s &\in \mathbb{U} \end{cases} \tag{3.14}$$

This formulation can be modelled as the Steady-State Optimization Module of Figure 3.1

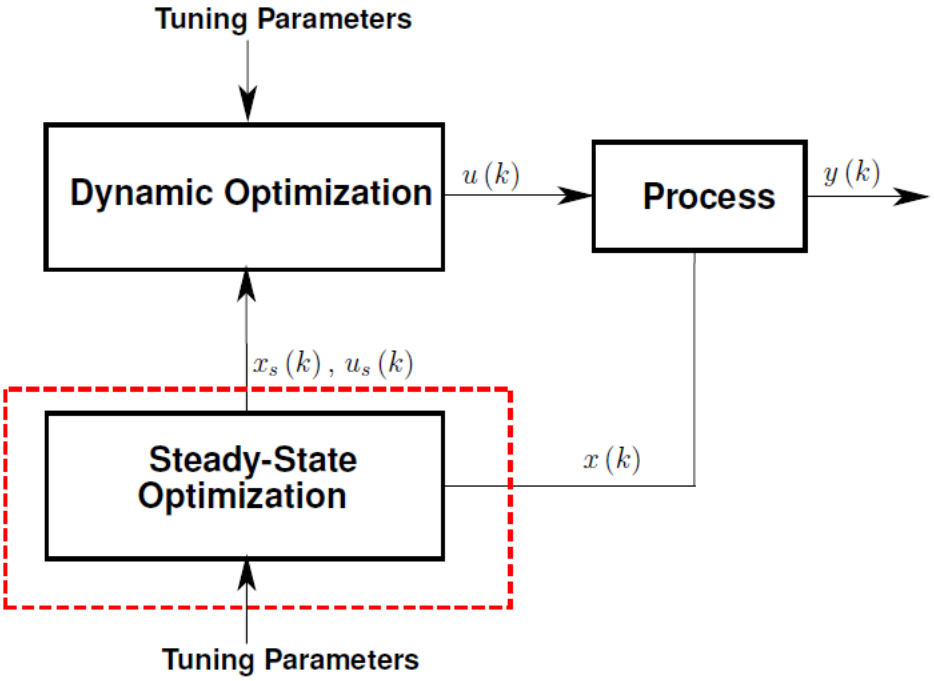

Figure 3.6: Steady State Optimization Module

This optimization compute, for all feasible couples $(x_s,\, u_s)$ the best choice of a set point y_s as close as possible to the target, y_{target} . Given the steady-state solution, we define the following optimal control problem in deviation variables

$$\mathbb{P}_N(\tilde{x}): \qquad V_N^0(\tilde{x}) := \min_{\tilde{\mathbf{u}}} \{V_N(\tilde{x}(0),\, \tilde{\mathbf{u}}) \,|\, \tilde{\mathbf{u}} \in \mathcal{U}_N(\tilde{x})\} \tag{3.15}$$

The moving horizon control law uses the first move of this optimal sequence, $\tilde{u}^0\left(\tilde{x}\left(0\right)\right) = \tilde{\mathbf{u}}^0\left(0\,;\,\tilde{x}\left(0\right)\right)$, so the system input will be

$$u = \tilde{u}^0\left(\tilde{x}\left(0\right)\right) + u_s$$

There are several approaches in literature to implement this formulation. Results and convergence are tied to:

- who is state vector x
- closed-loop stability that is influenced by which type of control approach is used
- the fact if optimization are implemented in two steps or in a single layer structure
  - in two steps formulation the controller are composed by the steady-state-target-optimizer (3.14) and the MPC controller (3.15)
  - in one step formulation, the controller integrates the two previous layer all in one

We expose all these formulations in the next chapter.

CHAPTER 4

Distributed MPC: theory and formulation

In many large-scale control applications, it becomes convenient to break the large plant-wide problem into a set of smaller and simpler sub-problems in which the local inputs are used to regulate the local outputs. The overall plant-wide control is then accomplished by the composite behaviour of the interacting, local controllers. There are many ways to design the local controllers, some of which produce guaranteed properties of the overall plant-wide system. We consider four control approaches in this chapter: decentralized, non-cooperative, cooperative, and centralized predictive control. For ensuring closed-loop stability of a wide class of plant-wide models and decomposition choices, cooperative control emerges as the most attractive option for distributed MPC. Before analyse these four methods we need some basic results. Remember that for us, distributed system is modelled by a graph and the plant-wide behaviour is represented by the overall state vector of the graph, composed by all agent's state vector.

4.1 Introduction and Preliminary Results

In our development of distributed MPC, we require some basic results on two topics: how to organize and solve the linear algebra of linear MPC, and how to ensure stability when using suboptimal MPC.

4.1.1 Least Squares Solution

It proves convenient to see MPC quadratic programming in order to organize the sequence of states and input as a single large linear algebra problem. We consider first the unconstrained LQ problem in 3.1 subject to $x^+ = A\,x + B\,u$ which we solved with dynamic programming. In this case we take the brute-force approach to find the explicit optimal control law directly. So we write the model solution as

$$\begin{bmatrix} x\,(1) \\ x\,(2) \\ \vdots \\ x\,(N) \end{bmatrix} = \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{\mathcal{A}} x\,(0) + \underbrace{\begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}}_{\mathcal{B}} \begin{bmatrix} u\,(0) \\ u\,(1) \\ \vdots \\ u\,(N-1) \end{bmatrix}$$

or in compact form

$$\mathbf{x} = \mathcal{A}\,x\,(0) + \mathcal{B}\mathbf{u}$$

The objective function can be expressed as

$$V\,(x\,(0)\,,\,\mathbf{u}) = \frac{1}{2}\left(x^T\,(0)\,Q\,x\,(0) + \mathbf{x}^T\mathcal{Q}\,\mathbf{x} + \mathbf{u}^T\mathcal{R}\,\mathbf{u}\right)$$

where

$$\mathcal{Q} = \operatorname{diag}\left(\begin{bmatrix} Q & Q & \cdots & P_f \end{bmatrix}\right) \in \mathbb{R}^{Nn \times Nm}$$

$$\mathcal{R} = \operatorname{diag}\left(\begin{bmatrix} R & R & \cdots & R \end{bmatrix}\right) \in \mathbb{R}^{Nm \times Nm}$$

Now, we retain the state sequence and adjoin the model equations as equality constraints. Including state and input in the sequence of unknowns, we define the enlarged vector $\mathbf{z}$ to be

$$\mathbf{z} = \begin{bmatrix} u(0) \\ x(1) \\ u(1) \\ x(2) \\ \vdots \\ u(N-1) \\ x(N) \end{bmatrix}$$

The objective function is

$$\min_{\mathbf{u}} \frac{1}{2} \left(x^T(0) \, Q \, x(0) + \mathbf{z}^T H \, \mathbf{z} \right)$$

in which

$$H = \operatorname{diag} \left(\begin{bmatrix} R & Q & R & Q & \cdots & R & P_f \end{bmatrix} \right)$$

The constraints can be rewritten as

$$D\mathbf{z} = d$$

in which

$$D = - \begin{bmatrix} B & -I & & & & & \\ & A & B & -I & & & \\ & & & \ddots & \ddots & & \\ & & & A & B & -I \end{bmatrix} \qquad d = \begin{bmatrix} A \\ 0 \\ \vdots \\ 0 \end{bmatrix} x(0) \tag{4.1}$$

We should implement an alternative form of 4.1 as we will see later.

4.1.2 Suboptimal MPC and its Stability

There is a practical problem. If the optimal control problem $\mathbb{P}_N(x)$ solved online, is not convex, global minimum cannot be determined in $\mathcal{U}_N(x)$. It is possible to achieve stability without requiring globally optimal solutions of $\mathbb{P}_N(x)$. All that is required is at state x, a feasible solution $\mathbf{u} \in \mathcal{U}_N(x)$ is found giving a cost $V_N(x, \mathbf{u})$ lower than the cost $V_N(w, \mathbf{v})$ at the previous state w due to the early control sequence $v \in \mathcal{U}_N(w)$. Consider then the usual optimal control problem with the terminal cost $V_f(\cdot)$ and terminal constraint set $\mathbb{X}_f$ satisfying 3.7 and 3.8; $\mathbb{X}$ is assumed to be closed and $\mathbb{U}$ to be compact. In addition we assume

that $V_f(\cdot)$ satisfies $\gamma_f(|x|) \leq V_f(x) \leq \alpha_f(|x|)$, $\forall x \in \mathbb{X}_f$ where $\gamma_f(|x|)$ and $\alpha_f(|x|)$ are $\mathcal{K}_\infty$ functions. These conditions are satisfied if, for example, $V_f(\cdot)$ is a positive definite quadratic function and $\mathbb{X}_f$ is a sublevel set of $V_f(\cdot)$ (it's assumed satisfied, $\mathbb{X}_f = \{x | V_f(x) \leq r\}$). Assume the first equation of 3.11 satisfied and let $\mathcal{X}_N$ denote, as before, the set of x for which a control sequence $\mathbf{u}$ exists that satisfies the state, control and terminal constraints. The basic idea behind the suboptimal model predictive controller is simple. Consider the current state is x and that $\mathbf{u} = \{u(0), u(1), \ldots, u(N-1)\}$ is a feasible control sequence. The first element $u(0)$ is applied to the system. Consider now the predicted control sequence $\mathbf{u}^+$ defined by

$$\mathbf{u} = \{u(1), u(2), \ldots, u(N-1), \kappa_f(x(N))\} \tag{4.2}$$

which satisfies 3.7 for all $x \in \mathbb{X}_f$. $\kappa_f(x(N))$ is usually of the form $\kappa_f(x(N)) = Kx$. Then $\mathbf{u}^+ \in \mathcal{U}_N$ satisfies

$$V_N(x^+, \mathbf{u}^+) + l(x, u(0)) \leq V_N(x, \mathbf{u})$$

and, hence

$$V_N(x^+, \mathbf{u}^+) \leq V_N(x, \mathbf{u}) - \alpha_1(|x|) \tag{4.3}$$

Inequality 4.3 is a reminiscent of 3.9 that provides the basis for establishing asymptotic stability of the origin for the controlled system. The obstacle to applying standard Lyapunov theory, is that there is no obvious Lyapunov function because at each x^+ there exists many control sequence u^+ satisfying (4.3). However, global attractivity of the origin in $\mathcal{X}_N$, may be established. The only modification required is when $x \in \mathbb{X}_f$

$$V_N(x, \mathbf{u}) \leq V_f(x) \qquad f(x, u(0)) \in \mathbb{X}_f \tag{4.4}$$

Stability of the origin can be established using (4.3), (4.4) and the properties of $V_f(\cdot)$ as shown subsequently. Inequality (4.4) is achieved quite simply by using the control law $u = \kappa_f(x)$ to generate the control u when $x \in \mathbb{X}_f$. Also, it follows from Assumption (3.7) and the definition $\kappa_f(\cdot)$ that $x^+ = f(x, u(0)) \in \mathbb{X}_f$ if $x \in \mathbb{X}_f$. Thus the two conditions in (4.4) are satisfied by $\mathbf{u}(x; \kappa_f)$. If desidered, $\mathbf{u}(x; \kappa_f)$; may be used for the current control sequence $\mathbf{u}$ or as a "warm start" for an optimization algorithm yielding an improved control sequence. In any case, if (4.4) is satisfied, stability of the origin may be established. When using

distributed MPC it's convenient to implement the control without solving the complete optimization. We consider a specific variation of suboptimal MPC in which a string guess is available from the control trajectory at the previous time. Optimization method is not important, but we restrict the method so that each iteration is feasible and decreases the value of the cost function.

4.1.3 Suboptimal MPC algorithm

Define N_{iter} as number of iteration and $\mathcal{X}_N$ the set of states for which the initial control sequence $\mathbf{h}(x_0)$ is well defined.

Initialize: set current state $x = x_0$, current control sequence, $\mathbf{u} = \mathbf{h}(x_0)$.

Step 1 (State evolution): Apply control $u = u(0)$ to the system. Obtain state at next sample, x^+. For the nominal system $x^+ = f(x, u(0))$.

Step 2 (Warm start): Denote the warm start for the next sample time as $\tilde{\mathbf{u}}^+$. We use

$$\tilde{\mathbf{u}}^+ = \{u(1), u(2), \ldots, u(N-1), 0\}$$

in which $x(N) = \phi(N; x, \mathbf{u})$. The warm start $\tilde{\mathbf{u}}^+$ therefore is a function of $(x, \mathbf{u})$. This warm start is simplified version of 4.2. In MPC is simpler to use zero for the final control move in the warm start.

Step 3 (Iteration of an optimization method): The controllers performs N_{iter} iterations of a feasible path optimization algorithm to obtain an improved control sequence using initial state x^+. The final input sequence $\mathbf{u}^+$ is a function of the state initial condition and the warm start. But these are both function of $(x, \mathbf{u})$, so it can be expressed as

$$\mathbf{u}^+ = g(x, \mathbf{u})$$

Step 4 (Next time step): Update state and input sequence: $x \leftarrow x^+$ and $\mathbf{u} \leftarrow \mathbf{u}^+$.

It's possible to show that system cost function, in this case, satisfies the proprieties for the cost decrease but in this case it depends only from x and the first element of $\mathbf{u}$, $u(0)$. Given a system $x^+ = f(x)$, with equilibrium point at the origin, $f(0) = 0$, denote $\phi(k; x(0))$ as the solution $x(k)$ given the initial state

$x(0)$. We consider exponential stability of the origin (as in Definitions A.7) on a set $\mathbb{X}$ (Theorem A.11).

Lemma 4.1. *Consider a system*

$$\begin{pmatrix} x^+ \\ \mathbf{u}^+ \end{pmatrix} = \begin{pmatrix} f(x, u) \\ g(x, \mathbf{u}) \end{pmatrix} \qquad (x(0), \mathbf{u}(0)) \text{ given}$$

with a steady-state solution $(0, 0) = (f(0, 0), g(0, 0))$*. Assume that the function* $V(\cdot) : \mathbb{R}^n \times \mathbb{R}^{Nm} \to \mathbb{R}_+$ *and input trajectory* $\mathbf{u}$ *satisfies*

$$\begin{aligned} a|(x, \mathbf{u})|^2 \leq V(x, \mathbf{u}) \leq b|(x, \mathbf{u})|^2 \\ V(x^+, \mathbf{u}^+) - V(x, \mathbf{u}) \leq -c|(x, u(0))|^2 \\ |\mathbf{u}| \leq d|x| \quad x \in \mathbb{B}_r \end{aligned}$$

in which $a, b, c, d, r, > 0$*. If* $\mathcal{X}_N$ *is forward invariant for the system* $x^+ = f(x, u)$*, the origin is exponentially stable for all* $x(0) \in \mathcal{X}_N$*.* $\mathbb{B}_r$ *represent a ball of radius* r *arbitrarily small.*

For lemma 4.1 we use the fact that $\mathbb{U}$ is compact. For unbounded $\mathbb{U}$ exponential stability may instead be established by compactness of $\mathcal{X}_N$.

4.2 Definitions and Architectures for Constrained Linear Systems

A common property of the four control approaches is system architecture. We describe it here schematically with no detail on implementation. We explicit them in next Chapter. We start description of distributed MPC algorithms by considering an overall discrete-time linear time-invariant system of the form:

$$\begin{aligned} x^+ &= A\,x + B\,u \\ y &= C\,x \end{aligned} \tag{4.5}$$

in which $x \in \mathbb{R}^n$ and $x^+ \in \mathbb{R}^n$ are, respectively, the system state at a given time and the system state at a successor time, $u \in \mathbb{R}^m$ is the input and $y \in \mathbb{R}^p$ is the output.

We consider that overall system (4.5) is divided into M subsystems, $\mathbb{S}_i$, defined by (disjoint) sets of inputs and outputs (states), and each $\mathbb{S}_i$ is regulated by a *local* MPC. For each $\mathbb{S}_i$, we denote by $y_i \in \mathbb{R}^{p_i}$ its output, by $x_i \in \mathbb{R}^{n_i}$ its state, and by $u_i \in \mathbb{R}^{m_i}$ the control input computed by i-th MPC. Due to interactions among subsystems, the local output y_i (and state x_i) is affected by control inputs computed by (some) other MPCs. Hence, the dynamics of $\mathbb{S}_i$ can be written as:

$$\begin{aligned} x_i^+ &= A_i\,x_i + B_i\,u_i + \sum_{j \in \mathcal{N}_i} B_{ij}\,u_j \\ y_i &= C_i\,x_i \end{aligned} \tag{4.6}$$

in which $\mathcal{N}_i$ denotes the indices of *neighbors* (like adjacent node) of $\mathbb{S}_i$, i.e. the subsystems whose inputs have influence on the states of $\mathbb{S}_i$[1].

Assumption 4.2. *The pair $(A_i,\, B_i)$ is stabilizable and the state is measured at each sampling time.*

Consider a quadratic stage function like in Section 3.2.4 but for only a subsystem $l_i\,(x,\, u) = \frac{1}{2}\left(x^{'} Q_i\,x + u^{'} R_i\,u\right)$ and a terminal cost function $V_{f\,i}\,(x) \triangleq \frac{1}{2} x^{'} P_i\,x$ with $Q_i \in \mathbb{R}^{n_i \times n_i}$, $R_i \in \mathbb{R}^{m_i \times m_i}$ and $P_i \in \mathbb{R}^{n_i \times n_i}$ positive definite. Note that P can be written as the solution of the discrete time algebraic Riccati equation. Let $x_i\,(0)$ be the state of $\mathbb{S}_i$ at the current decision time. Consequently,

[1] We will define explicitly $\mathbb{S}_i$ and $\mathcal{N}_i$ in next Chapter

the finite-horizon cost function associated with $\mathbb{S}_i$ is given by:

$$V_i\left(x_i\left(0\right),\, \mathbf{u}_i,\, \{\mathbf{u}_j\}_{j\in\mathcal{N}_i}\right) \triangleq \sum_{k=0}^{N-1} l_i\left(x_i\left(k\right),\, u_i\left(k\right)\right) + V_{f\,i}\left(x_i\left(N\right)\right) \tag{4.7}$$

in which $\mathbf{u}_i = \left(u_i\left(0\right),\, u_i\left(1\right),\, \ldots,\, u_i\left(N-1\right)\right)$ is a finite-horizon sequence of control inputs of $\mathbb{S}_i$ and $\mathbf{u}_j$ is similarly defined as a sequence of control inputs of each neighbour $j \in \mathcal{N}_i$. Note that (4.7) is a function of neighbours' input sequence $\{u_j\}_{j\in\mathcal{N}_i}$ due to the dynamic (4.6).

Next Figure 4.1 shows a graphical representation of the description above

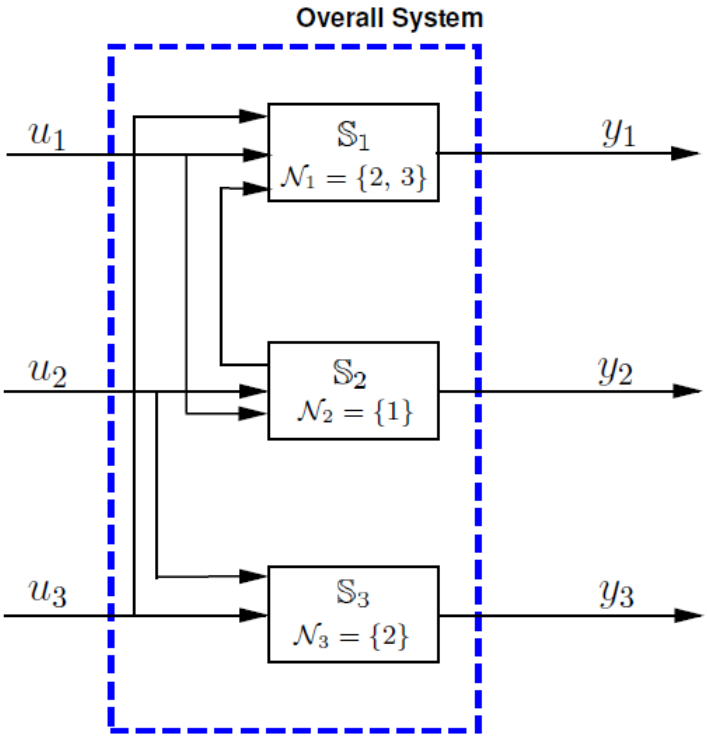

Figure 4.1: Interconnected systems and neighbours definition

4.3 Distributed Control Approaches

Large scale systems (e.g. industrial processing plants, power generation network, etc.) usually comprise several interconnected units may exchange material, energy and information streams. The overall effectiveness and profitability of such large-scale systems depend strongly on the level of local effectiveness and profitability of each unit but also on the level of interactions among units.

An overall optimization goal can be achieved by adopting a single *centralized* model predictive control system in which all control input trajectories are optimized simultaneously to minimize a common objective. This choice is often avoided for several reasons. When the overall number of inputs and states is very large, a single optimization problem may require computational resources (CPU time, memory, etc.) that are no available and/or compatible with system's dynamics. Even if this limitation do no hold, it is often the case that organizational reasons require the use of smaller, local controller, which are easier to coordinate and maintain.

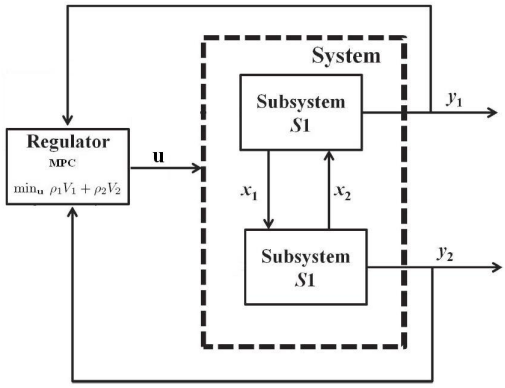

Figure 4.2: Centralized System

Thus, industrial control system are often *decentralized*, i.e. the overall system is divided into subsystems and a local controller is designed for each unit disregarding interaction from/to other subsystems. Due to dynamic coupling is well known that performance of such decentralized systems may be poor and stability properties may be even lost.

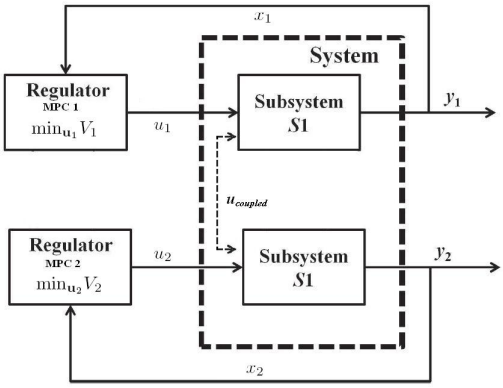

Figure 4.3: Decentralized System, no communication, local objective

Between centralized and decentralized strategies, *distributed* control preserves topology and flexibility of decentralized control and offers a nominal closed-loop stability. This is achieved by two features: the network interactions between subsystems are explicitly modelled and open-loop information, usually input trajectories, is exchanged between subsystem controllers.

Thus, in distributed control there are two main strategies for utilization of the open-loop information. In *non-cooperative* distributed control, each subsystem controller anticipates the effect of network interactions only locally. However if these interactions are strong, non-cooperative control can be destabilize plant and performance can be poorer than decentralized control.

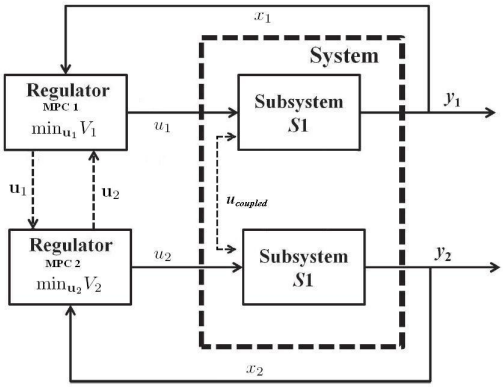

Figure 4.4: Non-Cooperative, communication, local objective

Alternatively, *cooperative* distributed control improves performance by requiring each subsystem to consider the effect of local control actions on all

subsystem in the network. So, each local controller optimize a plant-wide objective function, e.g., the centralized one. However, even if objective function is global, information is transmitted (and received) from any local regulator to a given subset of the others (*partially connected*) algorithms. Distributed optimization algorithms are used to ensure a decrease in the plant-wide objective at each intermediate iterate. Under cooperative control plant-wide performances converge to the Pareto optimum, providing centralized-like performance. Because the optimization may be terminated before convergence, however, cooperative control is a form os suboptimal control for the plant-wide control problem. Hence, stability is deduced from suboptimal control theory.

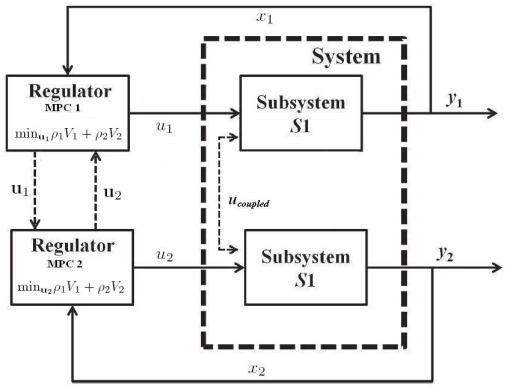

Figure 4.5: Cooperative, communication, global objective

4.4 Cooperative Distributed MPC

Several levels of communications and cooperation can exist among local controllers. We consider now only the *cooperative* MPC architectures. It can be shown that it's a form of suboptimal MPC and establish stability. We recall the finite-horizon cost function for distributed system (4.7)

$$V_i\left(x_i\left(0\right),\,\mathbf{u}_i,\,\{\mathbf{u}_j\}_{j\in\mathcal{N}_i}\right)\triangleq\sum_{k=0}^{N-1}l_i\left(x_i\left(k\right),\,u_i\left(k\right)\right)+V_{f\,i}\left(x_i\left(N\right)\right)$$

associated with $\mathbb{S}_i$ in Section 4.2. In Cooperative MPC architectures each controller optimizes a common plant-wide objective function

$$V\left(x\left(0\right),\,\mathbf{u}\right)\triangleq\sum_{i}^{M}\rho_i V_i\left(x_i\left(0\right),\,\mathbf{u}_i,\,\{\mathbf{u}_j\}_{j\in\mathcal{N}_i}\right)\tag{4.8}$$

in which $\rho_i>0\,\forall\,i$, are given scalar weights, and $\mathbf{u}\triangleq(\mathbf{u}_1,\,\ldots,\,\mathbf{u}_M)$ is the overall control sequence. In particular, given the known value of neighbours subsystems' control input sequences $\{\mathbf{u}_j\}_{j\neq i}$ each local MPC solves the following finite-horizon optimal control problem

$$\begin{aligned}\mathbb{P}_i^{CD_i}:\qquad&\min_{\mathbf{u}_i}V\left(x\left(0\right),\,\mathbf{u}\right)\\ \text{s.t.}\qquad&\mathbf{u}_i\in\mathbb{U}_i^N\end{aligned}\tag{4.9}$$

Where super-script N defines that input control sequence of agent i, with set $\mathbb{S}_i$, belongs to $\mathbb{U}$ for all horizon length; in order to substitute sub-script N in $\mathbb{P}_N^{CD_i}$.

In cooperative scheme the obtained solution can be exchange between subsystems to further iterate an optimization algorithm as in Subsection 4.1.3. Notice that in $\mathbb{P}_i^{CD_i}$, the possible implication of the local controller to other subsystems' objective is taken into account, as well as the effect of the neighbors' sequence $\{\mathbf{u}_j\}_{j\neq i}$ on the local state evolution (4.6). All controllers compute *local* controllers to minimize *global* objective. Convergence of cooperative iterations is guaranteed, and under suitable assumption the converged solution is the centralized Pareto-optimal solution as we'll show later.

4.4.1 Stability of Cooperative DMPC

Given the following first group of Assumption:

Assumption 4.3.

- *Each pair* $(A_i,\, B_i)$ *is stabilizable;*
- *The state of each subsystem* x_i *is assumed known at each decision time;*
- $u_i \in \mathbb{U}_i, \quad i = 1, \ldots, M$ *in which* $\mathbb{U}_i$ *are polyhedrons containing the origin in its interior.*
- *The systems* $(A_i,\, Q_i)$ *are detectable;*
- $N \geq \max_{i \in \mathbb{I}_{1:M}} (n_i^u)$*, in which* n_i^u *is the number of unstable modes of* A_i*, i.e., the number of* $\lambda \in \operatorname{eig}(A_i)$ *such that* $|\lambda| \geq 1$*;*
- $R_i \succeq 0$
- $Q_i \succeq 0$

For an unstable plant, we constrain the unstable modes to be zero at the end of the horizon to maintain closed-loop stability. It is reached with Schur decomposition[2].

Consider that at each iterate $p \geq 0$ the optimization problem (4.9) is solved for each subsystem. Consider $\mathbf{u}_i^*$ as the solution of this problem in which we have considered together with hard input constraints, the stabilizing condition on unstable modes, global state evolution and the Lyapunov stability constraint mentioned in Lemma 4.1 rewritten for a single agent as

$$|\mathbf{u}_i| \leq d_i \sum_{j \in \mathbb{I}_{1:M}} |x_{j\,i}(0)| \quad \text{if } x_{j\,i}(0) \in \mathbb{B}_r \,\forall\, j \in \mathbb{I}_{1:M} \tag{4.10}$$

Given the prior, feasible iterate $(\mathbf{u}_i^p)$ the next iterate is defined to be a convex combination between the optimal solution of 4.9 and the one at previous time instant with a convex step weight w_i such that $\sum_{i=1}^{M} w_i = 1$. The following proprieties follow immediately.

Lemma 4.4. *(Feasibility) Given a feasible initial guess, the iterates satisfy* $\left(\mathbf{u}_1^p,\, \ldots,\, \mathbf{u}_M^p\right) \in \mathbb{U}_1^N \times \ldots \times \mathbb{U}_M^N$ *for all* $p \geq 1$.

Lemma 4.5. *(Convergence) The cost* $V(x(0), \mathbf{u}^p)$ *is non-increasing for each iterate* p *and converges as* $p \to \infty$.

[2] Stewart T. B., Venkat N. A., Rawlings B. J., Wright J. S., Pannocchia G., "Cooperative distributed model predictive control", *Systems & Control Letters*, Vol. 59, 2010, pp. 460-469.

Lemma 4.6. *(Optimality) As $p \to \infty$ the cost $V\left(x\left(0\right), \mathbf{u}^{p}\right)$ converges to the optimal value $V^{0}\left(x\left(0\right)\right)$, and the iterates $\left(\mathbf{u}_{1}^{p}, \ldots, \mathbf{u}_{M}^{p}\right)$ converges to $\left(\mathbf{u}_{1}^{0}, \ldots, \mathbf{u}_{M}^{0}\right)$ that is the Pareto (centralized) optimal soluition.*

We define the steerable set $\mathcal{X}_n$ as before, as the set of all the states x such that there exists a $\mathbf{u} \in \mathbb{U}^{N}$ satisfying the stabilizing condition on unstable modes.

Assumption 4.7. *Given $r > 0$, for all $i \in \mathbb{I}_{1:M}$, d_i is chosen large enough such that there exists a $\mathbf{u}_i \in \mathbb{U}^{N}$ satisfying $|\mathbf{u}_i| \leq d_i \sum_{j \in \mathbb{I}_{1:M}} |x_{ij}|$ and the stabilizing condition on unstable modes $\forall x_{ij} \in \mathbb{B}_r \forall\, j \in \mathbb{I}_{1:M}$.*

Given Assumption 4.7, it can be shown that $\mathcal{X}_N$ is forward invariant.

After these considerations we can now establish stability of the closed-loop system by treating cooperative MPC as a form of suboptimal MPC. We define the warm start

$$\tilde{\mathbf{u}}_{i}^{+} = \left\{u_i\left(1\right), u_i\left(2\right), \ldots, u_i\left(N-1\right), 0\right\}$$

The warm start $\tilde{\mathbf{u}}_{i}^{+}$ is used as the initial condition for the cooperative MPC problem in each subsystems i. We define the functions g_{i}^{p} as the outcome of applying the cooperative control iteration algorithm p times. The system evolution is then given by

$$\begin{pmatrix} x^{+} \\ \mathbf{u}^{+} \end{pmatrix} = \begin{pmatrix} A\,x + B\mathbf{u} \\ g^{p}\left(x,\, \mathbf{u}\right) \end{pmatrix}$$

Theorem 4.8. *(Exponential Stability) Given Assumptions 4.3 and 4.7, the origin of the closed-loop system $x^{+} = A\,x + B\mathbf{u}$ is exponentially stable on the set $\mathcal{X}_N$*

CHAPTER 5

DMPC: Implementation

Because we concentrate on distributed implementation, we will restrict attention on single agent problem. Hence we'll omit subscript N and we change it with i. For example, when we talk about optimal control problem $\mathbb{P}_N^{CD_i}$ we substitute it with $\mathbb{P}_i$. The reference to horizon length N doesn't disappear. It is implicit in the admissible input set definition ($\mathbb{U}_i^N$) as in (4.9).

We have now all tools' theory. We can control our plant-wide system modelled by a graph. We can follow a desired target by a cooperative distributed model predictive control applied on a single agent by modelling interactions among subsystem through arcs.

Let a generic system modelled by a graph. We have a plant-wide cost function. The objective is to find a local control action for each subsystem, in which the general system is divided, to minimize the plant-wide cost function. Each subsystem has a proper state vector and local input represented by

$$x_i^+ = A_i\, x_i + B_i\, u_i$$

As we have exposed in linear MPC example (Sections 3.2.4), we can associate a quadratic stage cost to a single agent

$$l_i(x\,,\,u) = \frac{1}{2}\left[\|x\|_{Q_i}^2 + \|u\|_{R_i}^2\right]$$

where subscripts Q_i and R_i indicate the state and input weight matrices of subsystem i. We associate the terminal cost function

$$V_{f\,i}\left(x\right) = \frac{1}{2}\|x\|_{P_i}^2$$

Let a finite horizon N. We know $x_i\left(k\right)$ be the state x_i at k-th decision time. The cost function on this horizon is (4.7) and we can explicit it with

$$V_i\left(x_i\left(0\right),\, \mathbf{u}_i,\, \{\mathbf{u}_j\}_{j\in\mathcal{N}_i}\right) \triangleq \frac{1}{2}\sum_{k=0}^{N-1}\left[\|x\|_{Q_i}^2 + \|u\|_{R_i}^2\right] + \frac{1}{2}\|x\|_{P_i}^2$$

with $\mathbf{u}_i = \left[u_i\left(0\right),\, u_i\left(1\right),\, \ldots\,,\, u_i\left(N-1\right)\right]$ and $\mathbf{u}_j$ is similarly defined as a sequence of control inputs of each neighbour $j \in \mathcal{N}_i$. Optimal control problem for agent i-th is thus consistency again with $\mathbb{P}_i$ of (4.9) and its solution is

$$\mathbb{P}_i: \quad V^0\left(x\right) := \min_{\mathbf{u}_i}\left\{V\left(x\left(0\right),\, \boldsymbol{u}\right)\right\}$$

Problem $\mathbb{P}_i$ is a parametric optimization problem in which the decision variable is $\mathbf{u}_i$, and both the cost and the constraint set depend on the *parameter* x associated with agent i.

5.1 Node Interaction

At this point we need to explicit $\mathcal{N}_i$ and $\mathbb{S}_i$. The first one, in our case, defines, first of all, the agents defined by *input-star* of i-th agent due to dynamic of x_i^+, as described in (4.6). Nevertheless agent i influences *output-star*'s agents. This influence have to be accounted on cost function: it's important for a cooperative game. Agents of input-star cannot be accounted on cost function because their actions are chosen by themselves. The extended dynamic can be written as

$$\begin{cases} x_i^+ = A_i\, x_i + B_i\, u_i + \sum_{k\,\in\, S_i^{IN}} B_{ik}\, u_k \\ \\ x_j^+ = A_j\, x_j + B_{ji}\, u_i + \left(\sum\limits_{k\,\in\, S_j^{IN}\setminus\{i\}} B_{jk}\, u_k + B_j u_j \right) \quad , j \in S_i^{OUT} \end{cases} \tag{5.1}$$

Interactions are represented by the *constant* matrices $(B_{ik},\, B_{ji},\, B_{jk})$ dependent from from *time variant* input $(u_k,\, u_i)$. More precisely the product, for example, $B_{ik} \cdot u_{ik}$ is the weight of the arc from a node k to another i. These are the adjacent matrix values[1]. So, from here forward, $\mathcal{N}_i$ can be neglected. It's integrated in the above extended dynamic.

If we consider these interaction in stage cost function for all $j \in S_i^{OUT}$ it becomes

$$l_i(x\,,\, u) = \frac{1}{2}\left[\|x\|_{Q_i}^2 + \|u\|_{R_i}^2 + \sum_{j\,\in\, S_i^{OUT}} \|x\|_{Q_j}^2 \right]$$

and final cost function becomes

$$V_{f\,i}\,(x) = \frac{1}{2}\left(\|x\|_{P_i}^2 + \sum_{j\,\in\, S_i^{OUT}} \|x\|_{P_j}^2 \right)$$

As before, this is only the definition of stage e final cost functions without specifying the state value. These considerations lead to a new implementation of V_i where $\{\mathbf{u}_j\}_{j\in\mathcal{N}_i}$ is neglected. A new parameter x_j is now considered as we

[1] Remember that adjacent matrix element (i, j) represent the influence from i to j. Instead in this notation, $B_{ik} \cdot u_k \to (i, k)$ represent the influence from k to i.

have already analysed before. So the finite horizon cost function is implemented as

$$\begin{aligned} V_i(x_i(0),\, \mathbf{u}_i,\, x_j\,(0)) &= \sum_{k=0}^{N-1} \left[l_i \left(\left[\begin{array}{cc} x_i\,(k) & x_j\,(k) \end{array} \right],\, u_i\,(t) \right) \right] \\ &\quad + V_{f\,i} \left(\left[\begin{array}{cc} x_i\,(N) & x_j\,(N) \end{array} \right] \right) \\ &= \left[\|x\|^2_{Q_i} + \|u\|^2_{R_i} + \sum_{j\,\in\, S_i^{OUT}} \|x\|^2_{Q_j} \right] \\ &\quad + \frac{1}{2} \left(\|x\|^2_{P_i} + \sum_{j\,\in\, S_i^{OUT}} \|x\|^2_{P_j} \right) \end{aligned}$$

Plant-wide objective function each local controller must solve remains consistence with previous definition in equation (4.8)

$$V\left(x\,(0)\,,\, \mathbf{u}\right) \triangleq \sum_{i}^{M} \rho_i V_i\left(x_i(0),\, \mathbf{u}_i,\, x_j\,(0)\right)$$

Solution of finite horizon optimal control problem remains as in (4.9)

$$\mathbb{P}_i:\; V^0(x) \;:= \min_{\mathbf{u}_i} \left\{ V\left(x(0),\, \boldsymbol{u}\right) \right\}$$

subject to $\mathbf{u}_i \in \mathbb{U}_i^N$ as before.

We can reformulate this extended formulation in a more compact form by using graph theory by defining what $\mathbb{S}_i$ is exactly. We have seen in equation (5.1) the dynamic interaction of agent i with $j \in S_i^{OUT}$ and $k \in S_i^{IN}$. We define now the cooperative distributed model predictive control extended state vector and matrices associated with the extended dynamic interaction. We define the extended state vector integrating

$$x_i \leftarrow \left[\begin{array}{c} x_i \\ [x_j]_{j \in S_i^{OUT}} \end{array} \right]$$

relative to state matrix

$$A_i \leftarrow diag \left\{ \left[\begin{array}{c} A_i \\ [A_j]_{j \in S_i^{OUT}} \end{array} \right] \right\}$$

, the input matrix

$$B_i \leftarrow \left[\begin{array}{c} B_i \\ [B_{ji}]_{j \in S_i^{OUT}} \end{array} \right]$$

and output matrix

$$C_i \leftarrow diag\left\{\begin{bmatrix} C_i \\ [C_j]_{j\in S_i^{OUT}} \end{bmatrix}\right\}$$

Note that at the extended state vector, correspond an extended output vector

$$y_i \leftarrow \begin{bmatrix} y_i \\ [y_j]_{j\in S_i^{OUT}} \end{bmatrix}$$

Let us define the new input matrix on the new set $\mathbb{S}_i^{IN}$ relative to agent i.

$$\mathbb{S}_i^{IN} \leftarrow S_i^{IN} \cup S_i^{OUT} \cup \left(\bigcup_{j\in S_i^{OUT}} S_j^{IN} \setminus \{i\}\right)$$

Note that by definition, $i \notin \mathbb{S}_i^{IN}$. Then, for a given $k \in \mathbb{S}_i^{IN}$ we have

$$B_{ik} \leftarrow \begin{bmatrix} B_{ik} \\ [B_{jk}]_{j\in S_i^{OUT}} \end{bmatrix}$$

Note that $B_{jj} = B_j$. Thus, considering all the other interactions defined by subset $\mathbb{S}_i^{IN}$ we obtain, for all $k \in \mathbb{S}_i^{IN}$,

$$\bar{B}_i \leftarrow [B_{ik}]_{k\in\mathbb{S}_i^{IN}}^T$$

and

$$\bar{u}_i \leftarrow [u_k]_{k\in\mathbb{S}_i^{IN}}$$

where $[\cdot]^T$ indicates an *horizontal concatenation*. We can expose the extended dynamic interaction in equation (5.1) in a new compact form

$$x_i^+ = A_i\, x_i + B_i\, u_i + \bar{B}_i\, \bar{u}_i \tag{5.2}$$

It's all based on the agent i and factor $\bar{B}_i\, \bar{u}_i$ is a constant parameters that changes at each decision instant. We can reformulate the weight matrices in this new form as

$$P_i \leftarrow diag\left\{\begin{bmatrix} P_i \\ [P_j]_{j\in S_i^{OUT}} \end{bmatrix}\right\}$$

$$Q_i \leftarrow diag\left\{\begin{bmatrix} Q_i \\ [Q_j]_{j\in S_i^{OUT}} \end{bmatrix}\right\}$$

$$T_i \leftarrow diag\left\{\begin{bmatrix} T_i \\ [T_j]_{j\in S_i^{OUT}} \end{bmatrix}\right\}$$

By definition of cooperative model predictive control u_i doesn't change. So due to cost function R_i remains the same.

As an example take a graph with six node. Each node has a numeric identifier. Each node communicates with that ones with next and earlier numeric identifier. By definition of augmented system, we can model, for example, for agents 1 and 2, respectively

$$\mathbb{S}_1 = \{2,\, 3,\, 5,\, 6\} \quad \mathbb{S}_2 = \{1,\, 3,\, 6,\, 4\}$$

This can be graphically represented by

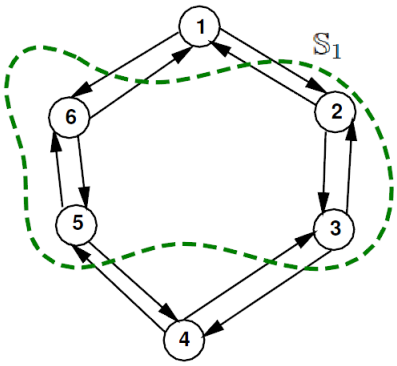

Figure 5.1: Node Interaction Example $\mathbb{S}_1$ set

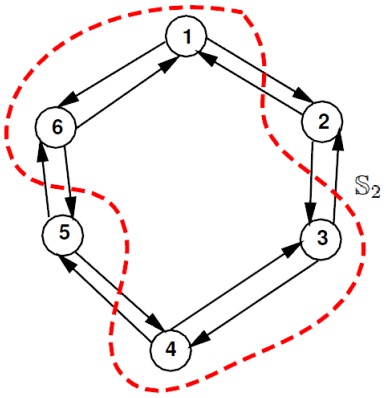

Figure 5.2: Node Interaction Example $\mathbb{S}_2$ set

5.2 Cooperative MPC algorithm

It's time to understand how we can actually control the already defined compact form (5.2)

$$x_i^+ = A_i\, x_i + B_i\, u_i + \bar{B}_i\, \bar{u}_i$$

where $\bar{B}_i\, \bar{u}_i$ can be defined trough adjacent matrix.

We present a streamlined description of a cooperative distributed MPC algorithm, in which each local controller solves $\mathbb{P}_i^{CDi}$ for (5.2), given a previously computed value of all other subsystems' input sequence. For each local controller, the new iterate is defined as a convex combination of the newly computed solution with the previous iteration. A relative tolerance is defined, so that cooperative iterations stop when all local controllers have computed a new iterate sufficiently close to the previous one. A maximum number of cooperative iterations can also be defined, so that a finite bound on the execution time can be established. The convergence to the optimum of the plant-wide cost function can be obtained by a suboptimal MPC algorithm applied to this new form of cooperative MPC described before.

Remember from here forward in this Section due to extended dynamic above, sub-script i represent:

- for vector x, the augmented version of the state, when no differently indicated;
- for vector u, the input sequence of single agent i.

This notation derives from analysed node interaction in Section 5.1 and it is expressed by compact form (5.2).

Algorithm ***Suboptimal Cooperative MPC***

Consider M subsystems with a random state, input and output vector dimensions. Let N horizon length. Consider the cooperative MPC Assumption 4.3. P_i for the single agent, can be computed as the unique stabilizing solution of discrete time algebraic Riccati equation (here, sub-script i is omitted for simplicity)

$$A^T P A - P - A^T P B (B^T P B + R)^{-1} B^T P A + Q = 0$$

Given for each single agent $x_i\,(0)$ and $\mathbf{u}_i = h\ (x_i\,(0))$ for $i = 1 \ldots M$

Step 1 Cooperative optimization First step is the iteration to obtain a local action vector as a convex combination of the newly computed solution and the previous one to optimize the global cost function.

$$\mathbf{u}_i^c \triangleq w_i \mathbf{u}_i^* + (1 - w_i)\mathbf{u}_i^{c-1}$$

where c is the number of iteration. A relative tolerance is defined as

$$e_i \triangleq \frac{||\mathbf{u}_i^c - \mathbf{u}_i^{c-1}||}{1 + ||\mathbf{u}_i^c||}$$

Implementation is based on quadratic programming, instruction *quadprog* in Matlab environment. This instruction require a quadratic linear cost function as mentioned in subsection 4.1.1. In this case we assume also a general augmented initial state $x_i\,(0)$ into augmented state vector $\mathbf{z}$

$$\mathbf{z} = \begin{bmatrix} x_i(0) \\ u_i(0) \\ x_i(1) \\ u_i(1) \\ x_i(2) \\ \vdots \\ u_i(N-1) \\ x_i(N) \end{bmatrix} \tag{5.3}$$

Thus, the objective function is defined as

$$\min_{\mathbf{u}_i} \frac{1}{2}\,\mathbf{z}'\,H\,\mathbf{z}$$

where

$$H = \operatorname{diag}\Big(\begin{bmatrix} Q_i & R_i & Q_i & R_i & Q_i & \cdots & R_i & P_{f\,i} \end{bmatrix}\Big)$$

The constraints can be written again as

$$D\,\mathbf{z} = d$$

In this case we have to considerate interactions, as described in extended dynamic system equation (5.1) or its compact form in equation (5.2). So

$$D = -\begin{bmatrix} I & & & & & & \\ A_i & B_i & -I & & & & \\ & & A_i & B_i & -I & & \\ & & & & \ddots & \ddots & \\ & & & & A_i & B_i & -I \end{bmatrix} \qquad d = \begin{bmatrix} x_i(0) \\ \bar{B}_i\,\bar{u}_i(0) \\ \bar{B}_i\,\bar{u}_i(1) \\ \vdots \\ \bar{B}_i\,\bar{u}_i(N-1) \end{bmatrix}$$

where A_i is the augmented one.

Step 2 Next time step We update each local input action vector. We remember that during cooperative optimization clock is stopped.

Step 3 State evolution We apply every first control action for each subsystem

$$\mathbf{u} = \mathbf{u}(0) = \begin{bmatrix} u_1(0) \\ u_2(0) \\ \vdots \\ u_M(0) \end{bmatrix}$$

We obtain the next time global state system as

$$\mathbf{x}^+ = f\left(\mathbf{x}, \mathbf{u}(0)\right)$$

Step 4 Warm Start Denote the warm start for the next sample time as $\tilde{\mathbf{u}}_i^+$ for single agent. We use

$$\tilde{\mathbf{u}}_i^+ = \left\{u_i\left(1\right), u_i\left(2\right), \ldots, u_i\left(N-1\right), 0\right\}$$

	Algorithm 1 *Cooperative MPC*
1:	Initialize $c \leftarrow 0$ and $e_i \leftarrow 2\epsilon$
2:	**while** $(c < c_{max})$ and $(\exists\, i \,|e_i > \epsilon)$ **do**
3:	$c \leftarrow c + 1$
4:	**for** i = 1 **to** M **do**
5:	Solve $\mathbb{P}_i^{CD_i}$ in (4.9) for (5.2) obtaining $\mathbf{u}_i^*$
6:	**end for**
7:	**for** i = 1 **to** M **do**
8:	Define new iterate: $\mathbf{u}_i^c \triangleq w_i \mathbf{u}_i^* + (1 - w_i)\mathbf{u}_i^{c-1}$
9:	Compute convergence error: $e_i \triangleq \frac{\|\|\mathbf{u}_i^c - \mathbf{u}_i^{c-1}\|\|}{1 + \|\|\mathbf{u}_i^c\|\|}$
10:	**end for**
11:	**end while**
12:	**return** Overall solution $\mathbf{u}^c \triangleq (\mathbf{u}_1^c, \mathbf{u}_2^c, \ldots, \mathbf{u}_M^c)$

Table 5.1: Cooperative MPC Algorithm

Results Cooperative MPC algorithm of *Step 1* verify several nice theoretical and practical properties:

- Feasibility of each iteration: $\mathbf{u}_i^{c-1} \in \mathbb{U}_i^N \Rightarrow \mathbf{u}_i^c \in \mathbb{U}_i^N \, \forall \, i = 1, ..., M$ e $c \in \mathbb{I}_{>0}$
- Cost decrease at each iteration: $V(x(0), \mathbf{u}^c) \leq V(x(0), \mathbf{u}^{c-1}), \, \forall \, c \in \mathbb{I}_{>0}$
- Cost convergence to the centralized optimum $\lim_{c \to \infty} V(x(0), \mathbf{u}^c) = \min_{\mathbf{u} \in \mathbb{U}^N} V(x(0), \mathbf{u})$, where $\mathbb{U} \triangleq \mathbb{U}_1 \times \mathbb{U}_2 \times \ldots \times \mathbb{U}_M$

Resorting to suboptimal MPC theory, the first two above properties can be exploited to show that the origin of closed-loop system:

$$x^+ = A\,x + B\,\kappa^c\,(x)\,, \qquad \kappa^c\,(x) \triangleq u^c\,(0)$$

is *exponentially stable for any finite* $c \in \mathbb{I}_{>}0$.

5.3 Cooperative MPC for Tracking: a novel approach

We focus on this kind of algorithm for tracking. As we have seen before, in tracking MPC, Section 3.3, we have to compute a steady state, input and output of the plant that satisfies (3.11). Steady State Target Optimizer is a function that computes this artificial equilibrium point. However, it can be a stand-alone function as in a two step optimization algorithm or it can be an embedded function in a one step optimization one. In order to study tracking, we define as $\mathbf{z}_{old}$, the old vector $\mathbf{z}$ in equation (5.3) but in deviation variable as defined in (3.13) for a modified dynamic optimization as defined in (3.15)

$$\mathbf{z}_{old} = \left[\begin{array}{ccccccc} \tilde{x}_i\,(0)^T & \tilde{u}_i\,(0)^T & \dots & \tilde{x}_i\,(N-1)^T & \tilde{u}_i\,(N-1)^T & \tilde{x}_i\,(N)^T \end{array} \right]^T$$

5.3.1 Two Steps Algorithms

In two steps algorithms, the equilibrium point $(x_s,\, u_s,\, y_s)$ is computed in decoupled way respect to dynamic optimization. There are two possibilities. One is to compute steady state, input and output once for each **time** iteration, that is in the outer loop. The other possibility is to compute the equilibrium point once for each **cooperative** loop. In both cases we solve the following quadratic programming problem

$$\begin{aligned} &\min_{\mathbf{z}} \tfrac{1}{2}\,\mathbf{z}^T\,H\,\mathbf{z} + f^T\mathbf{z} \\ &s.t. \begin{cases} D\mathbf{z} = d \\ l_b \leq \mathbf{z} \leq u_b \end{cases} \end{aligned} \tag{5.4}$$

Where, in general,

$$\mathbf{z} = \left[\begin{array}{ccc} x_s & u_s & y_s \end{array} \right]$$

What matrices and vectors represent, depends on what kind of optimization is used: centralized or cooperative. That is the value of state vector $\mathbf{z}$. So we compute steady state, input and output through the steady state optimal target problem (3.14) and find the steady variables. Then we solve the dynamic optimization problem (3.15)

SSTO, once for each TIME iteration, Centralized case

In this case Steady State Target Optimizer (SSTO) can be modelled as **Step 0** in MPC algorithm in Section 5.2. The artificial equilibrium point can be computed in centralized mode or in decentralized one. In **first case**, the equations (3.14) applied to centralized model, becomes

$$\min_{x_{s_{cen}},u_{s_{cen}}} V_{ss}(\cdot) = \|y_{s_{cen}} - y_{target_{cen}}\|^2_{T_{cen}}$$

$$s.t. \quad \begin{cases} x_{s_{cen}} = A_{cen}x_{s_{cen}} + B_{cen}u_{s_{cen}} \\ y_{s_{cen}} = C_{cen}x_{s_{cen}} \\ x_{\min_{cen}} \leq x_{s_{cen}} \leq x_{\max_{cen}} \\ u_{\min_{cen}} \leq u_{s_{cen}} \leq u_{\max_{cen}} \end{cases}$$

where

$$x_{s_{cen}} = \begin{bmatrix} x_{s_1} & \dots & x_{s_M} \end{bmatrix}^T \quad \in \mathbb{R}^n$$

$$u_{s_{cen}} = \begin{bmatrix} u_{s_1} & \dots & u_{s_M} \end{bmatrix}^T \quad \in \mathbb{R}^m$$

$$y_{s_{cen}} = \begin{bmatrix} y_{s_1} & \dots & y_{s_M} \end{bmatrix}^T \quad \in \mathbb{R}^p$$

$$y_{target_{cen}} = \begin{bmatrix} y_{target_1} & \dots & y_{target_M} \end{bmatrix}^T \quad \in \mathbb{R}^p$$

$$n = \sum_{i=1}^{M} n_i \quad x_{s_i} \in \mathbb{R}^{n_i}$$

$$m = \sum_{i=1}^{M} m_i \quad u_{s_i} \in \mathbb{R}^{m_i}$$

$$p = \sum_{i=1}^{M} p_i \quad y_{s_i} \in \mathbb{R}^{p_i}$$

$$\forall\, i = 1, \dots, M$$

The sub-subscript cen stands for the centralized version of the variable. They are the plant-wide variables. This problem was solved by quadratic programming instruction *quadprog* as we have seen above in (5.4) where

$$\mathbf{z} = \begin{bmatrix} x_{s_{cen}} & u_{s_{cen}} & y_{s_{cen}} \end{bmatrix}^T$$

and

$$H = \begin{bmatrix} I_n \cdot 10^{-6} & & \\ & I_m \cdot 10^{-6} & \\ & & T_{cen} \end{bmatrix} \qquad f = \begin{bmatrix} \mathbf{0}_{n,1} \\ \mathbf{0}_{m,1} \\ -T_{cen}\, y_{target_{cen}} \end{bmatrix}$$

while

$$D = \begin{bmatrix} A_{cen} - I_n & B_{cen} & \mathbf{0}_{n,p} \\ C_{cen} & \mathbf{0}_{p,m} & -I_p \end{bmatrix} \qquad d = \begin{bmatrix} \mathbf{0}_{n,1} \\ \mathbf{0}_{p,1} \end{bmatrix}$$

Lower and upper bound are defined by

$$l_b = \begin{bmatrix} x_{\min_{cen}} & u_{\min_{cen}} & y_{\min_{cen}} \end{bmatrix}$$

$$u_b = \begin{bmatrix} x_{\max_{cen}} & u_{\max_{cen}} & y_{\max_{cen}} \end{bmatrix}$$

The dimensions of various vectors are described below:

$$\begin{array}{ll} A_{cen} \in \mathbb{R}^{n \times n} & x_{\min_{cen}} \in \mathbb{R}^{n \times 1} \\ B_{cen} \in \mathbb{R}^{n \times m} & u_{\min_{cen}} \in \mathbb{R}^{m \times 1} \\ C_{cen} \in \mathbb{R}^{p \times n} & x_{\max_{cen}} \in \mathbb{R}^{n \times 1} \\ T_{cen} \in \mathbb{R}^{p \times p} & u_{\max_{cen}} \in \mathbb{R}^{m \times 1} \end{array}$$

At the end of optimization we will have the optimum global values for steady state, input and output

$$\mathbf{z}^* = \begin{bmatrix} x^*_{s_{cen}} & u^*_{s_{cen}} & y^*_{s_{cen}} \end{bmatrix}^T$$

In this way we compute the plant-wide optimal steady vector in one shoot. Finally we compute dynamic optimization in deviation variables

$$\tilde{x}_{cen} = x_{cen} - x^*_{s_{cen}}$$
$$\tilde{u}_{cen} = u_{cen} - u^*_{s_{cen}}$$

and with modified upper and lower bound

$$x_{\min_{cen}} - x_{s_{cen}} \leq \tilde{x}_{cen} \leq x_{\max_{cen}} - x_{s_{cen}}$$
$$u_{\min_{cen}} - u_{s_{cen}} \leq \tilde{u}_{cen} \leq u_{\max_{cen}} - u_{s_{cen}}$$

through function (3.15) using $\mathbf{z}_{old}$ vector as optimization vector.

Note that we change the variables outside from this optimization function. This is the easiest way to compute these steady state, input and output in one shot. But it is computational expensive as the plant dimension increase.

SSTO, once for each TIME iteration, Cooperative case

In **second case** we have to solve for SSTO problem for single agent. The equations (3.14) becomes

$$\min_{x_{s_i}, u_{s_i}} V_{ss}(\cdot) = \|y_{s_i} - y_{target_i}\|^2_{T_i}$$

$$s.t. \quad \begin{cases} x_{s_i} = A_i x_{s_i} + B_i u_{s_i} + \bar{B}_i \bar{u}^-_{s_i} \\ y_{s_i} = C_i x_{s_i} \\ x_{\min_i} \leq x_{s_i} \leq x_{\max_i} \\ u_{\min_i} \leq u_{s_i} \leq u_{\max_i} \end{cases}$$

where $\bar{B}_i \bar{u}^-_{s_i}$ account the dynamic interaction at previous cooperative instant and x_{s_i} refers to single agent only. So the optimization vector

$$\mathbf{z} = \begin{bmatrix} x_{s_i} & u_{s_i} & y_{s_i} \end{bmatrix}^T$$

Matrices become

$$H = \begin{bmatrix} I_{n_i} \cdot 10^{-6} & & \\ & I_{m_i} \cdot 10^{-6} & \\ & & T_i \end{bmatrix} \qquad f = \begin{bmatrix} \mathbf{0}_{n_i,1} \\ \mathbf{0}_{m_i,1} \\ -T_i\, y_{target_i} \end{bmatrix}$$

$$D = \begin{bmatrix} A_i - I_{n_i} & B_i & \mathbf{0}_{n_i,p_i} \\ C_i & \mathbf{0}_{p_i,m_i} & -I_{p_i} \end{bmatrix} \qquad d = \begin{bmatrix} -\bar{B}_i \bar{u}^-_{s_i} \\ \mathbf{0}_{p_i,1} \end{bmatrix}$$

While upper and lower bound

$$l_b = \begin{bmatrix} x_{\min_i} & u_{\min_i} & y_{\min_i} \end{bmatrix}$$

$$u_b = \begin{bmatrix} x_{\max_i} & u_{\max_i} & y_{\max_i} \end{bmatrix}$$

We obtain the optimal vector

$$\mathbf{z}^* = \begin{bmatrix} x^*_{s_i} & u^*_{s_i} & y^*_{s_i} \end{bmatrix}^T$$

It refers to a decentralized case so it doesn't take into account dynamic interaction and other agents' input, so these values cannot be take in input for dynamic optimization directly. To find real set-points values we have to find, first of all, an input convex combination

$$u^p_{s_i} = u^*_{s_i} \cdot w_i + u^{p-1}_{s_i} \cdot (1 - w_i)$$

Given the steady equations of state and output at the beginning of this paragraph,

$$x_{s_i} = A_i x_{s_i} + B_i u_{s_i} + \bar{B}_i \bar{u}_{s_i}^-$$
$$y_{s_i} = C_i x_{s_i}$$

we need to solve to find an admissible values for x_{s_i}. At time p we have

$$x_{si}^p = A_i x_{s_i}^p + B_i u_{s_i}^p + \bar{B}_i \bar{u}_{s_i}^{p-1}$$

If we solve for x_{s_i} we obtain

$$x_{si}^p = (I_{n_i} - A_i)^{-1} \left(B_i u_{s_i}^p + \bar{B}_i \bar{u}_{s_i}^{p-1} \right)$$

that we have to compute explicitly.

SSTO, once for each COOPERATIVE iteration

Instead of solve Steady State Target Optimizer once for each time iteration we can do that once for each **cooperative** loop. Implementation is the same as in decentralized case but it's located in the inner loop. As result we will obtain a faster convergence respect to SSTO, once for each TIME iteration, to desidered output value. We expect to converge to SSTO, once for each TIME iteration, at the end of all cooperative ones inside the same time instant.

5.3.2 One Step Algorithms

In one step algorithms, static and dynamic optimization are integrates in a single layer structure. This may be represented by

$$\min_{\tilde{\mathbf{u}}_i} V\left(\tilde{x}\left(0\right),\, \tilde{\mathbf{u}}\right) + \|y_s - y_{target}\|_T^2$$

The main difference with Two Step Algorithms is that steady state, input and output are decisions variables too. By definition of $\mathbf{z}_{old}$ we model all the matrices and vectors defined in Section 5.2 with sub-script old. So we can focus on the new formulation only. In this subsection we compare two version of the same algorithm. In first case we analyse the dynamic optimization quadratic function with a centralized steady vector. In second case we analyse a new formulation with an augmented steady vector. In both cases the main feature is to integrates steady state optimal target problem matrices in the dynamic and constraint ones. We recall here, the general formulation of dynamic optimization using quadratic programming

$$\min_{\mathbf{z}} \frac{1}{2}\mathbf{z}^T H\, \mathbf{z} + f^T \mathbf{z}$$

$$s.t. \quad \begin{cases} G\,\mathbf{z} \leq g \\ D\mathbf{z} = d \\ l_b \leq \mathbf{z} \leq u_b \end{cases}$$

It's obvious vector $\mathbf{z}$ doesn't represent the same as the one that has been used in Two Step Algorithms subsection.

In this case, due to fact that steady state, input and output are decision variables, we have to build selection matrices for the equality and inequality constraints defined above. More precisely we define a state $H_i^{(x)}$, input $H_i^{(u)}$ and input-bar $H_i^{(\bar{u})}$ selection matrices. How these matrices are implemented depends by the added decision variables. It will be clear from context what they are representing.

Centralized steady vector

This first version follow from the SSTO, once for each TIME iteration, Centralized case. The optimization vector $\mathbf{z}$ becomes

$$\mathbf{z} = \begin{bmatrix} x_{s_{cen}} \\ u_{s_{cen}} \\ y_{s_{cen}} \\ \mathbf{z}_{old} \end{bmatrix}$$

and the firsts three diagonals blocks of matrix H, are the same as the matrix H in Centralized case, in Subsection 5.3.1

$$H = \left[\begin{array}{ccccc} I_n \cdot 10^{-6} & & & | & \\ & I_m \cdot 10^{-6} & & | & \\ & & T_{cen} & | & \\ - - - - - & - - - - - & - - - & -- & - - - \\ & & & | & H_{old} \end{array}\right]$$

In this case, for first time in dynamic optimization, vector f is not empty

$$f = \begin{bmatrix} \mathbf{0}_{n,1} \\ \mathbf{0}_{m,1} \\ -T_{cen}\, y_{target_{cen}} \\ - - - - - - - - \\ f_{old} \end{bmatrix}$$

Note that f_{old} is an empty vector. Thus for a right implementation, set it to a zero-vector. Steady state constraints

$$x_{s_{cen}} = A_{cen} x_{s_{cen}} + B_{cen} u_{s_{cen}}$$
$$y_{s_{cen}} = C_{cen} x_{s_{cen}}$$

might be encapsulated inside D_{old} matrix and d_{old} vector. We recall them here

$$D_{old} = - \begin{bmatrix} I & & & & & & \\ A_i & B_i & -I & & & & \\ & & A_i & B_i & -I & & \\ & & & & \ddots & \ddots & \\ & & & & A_i & B_i & -I \end{bmatrix} \qquad d_{old} = \begin{bmatrix} x_i(0) \\ \bar{B}_i\, \bar{u}_i(0) \\ \bar{B}_i\, \bar{u}_i(1) \\ \vdots \\ \bar{B}_i\, \bar{u}_i(N-1) \end{bmatrix}$$

Due to fact that added decisions variables are on the top of the optimization vector $\mathbf{z}$ (See Section 5.2), we have to add three additional columns to D_{old} matrix. Constraint on initial state was

$$x_i(0) = \tilde{x}_i(0)$$

where x_{s_i} was been already computed, thus included in $\tilde{x}_i(0)$. So, due to steady state x_{s_i}, it will becomes

$$x_i(0) + x_{s_i} = \tilde{x}_i(0)$$

and we have to take into account. Remember that dynamic optimization is referred to augmented system. So the old constraints have to be modified on the same principles for initial state. To understand, we analyse the state evolution at first time instant. We have, for augmented system

$$x_i(1) = A_i\, x_i(0) + B_i\, u_i(0) + \bar{B}_i\, \bar{u}_i(0)$$

In case of steady state and input we have

$$x_{s_i} = A_i\, x_{s_i} + B_i\, u_{s_i} + \bar{B}_i\, \bar{u}_{s_i}$$

We find deviation equations as the difference between those ones

$$\tilde{x}_i(1) = A_i\, \tilde{x}_i(0) + B_i\, \tilde{u}_i(0) + \bar{B}_i\, (\bar{u}_i(0) - \bar{u}_{s_i})$$

The last component $-\bar{B}_i\,(\bar{u}_{s_i})$ is not included in the previous formulation of dynamic optimization. So we have to do two things:

- Include this element in dynamic constraints
- Remember we have, $u_{s_{cen}}$
  - we select this sub-vector input relative to i-th agent through selection matrix $H_i^{(\bar{u})}$

After this analysis we conclude that matrix D_{old} has to be modified in this way

$$D = \left[\begin{array}{ccc|ccc} A_{cen} - I_n & B_{cen} & \mathbf{0}_{n,p} & 0 & \cdots & 0 \\ C_{cen} & \mathbf{0}_{p,m} & -I_p & 0 & \cdots & 0 \\ \hline H_i^{(x)} & \mathbf{0}_{p,m} & \mathbf{0}_{n,p} & & & \\ \mathbf{0}_{n,n} & \bar{B}_i H_i^{(\bar{u})} & \mathbf{0}_{n,p} & & & \\ \vdots & \vdots & \vdots & & D_{old} & \\ \mathbf{0}_{n,n} & \bar{B}_i H_i^{(\bar{u})} & \mathbf{0}_{n,p} & & & \end{array}\right]$$

$$d = \left[\begin{array}{c} \mathbf{0}_{n,1} \\ \mathbf{0}_{m,1} \\ \hline d_{old} \end{array}\right]$$

In this case lower and upper bound cannot be expressed by the simple equations

$$l_b \leq \mathbf{z} \leq u_b$$

due to fact that steady state and input are decision variables too. So the previous

$$x_{\min_i} \leq \tilde{x}_i \leq x_{\max_i}$$

$$u_{\min_i} \leq \tilde{u}_i \leq u_{\max_i}$$

where steady variables are already included, become

$$x_{\min_i} \leq \tilde{x}_i + x_{s_i} \leq x_{\max_i}$$

$$u_{\min_i} \leq \tilde{u}_i + u_{s_i} \leq u_{\max_i}$$

It possible to represent these constraints by the system of inequalities

$$G\mathbf{z} \leq g$$

In our case it will be the vertical concatenation of two inequalities: one for lower and one for upper bound

$$\begin{bmatrix} G_{upper} \\ G_{lower} \end{bmatrix} \mathbf{z} \leq \begin{bmatrix} g_{upper} \\ g_{lower} \end{bmatrix}$$

where

$$g_{upper} = \begin{bmatrix} \underbrace{x_{\max_{cen}} \quad u_{\max_{cen}} \quad \dots \quad x_{\max_{cen}}}_{N \times 2+1} \end{bmatrix}$$

$$g_{lower} = \begin{bmatrix} -x_{\min_{cen}} & -u_{\min_{cen}} & \dots & -x_{\min_{cen}} \end{bmatrix}$$

Remember that ee have $N \times 2 + 1$ elements due to horizon length N. Recalling procedure for matrix D we build sub-matrix G_{upper} as

$$G_{upper} = \left[\begin{array}{ccc|ccccc} H_i^{(x)} & \mathbf{0}_{n,m} & \mathbf{0}_{n,p} & I_n & & & \\ \mathbf{0}_{n,n} & H_i^{(u)} & \mathbf{0}_{n,p} & & I_m & & \\ \vdots & \vdots & \vdots & & & \ddots & \\ H_i^{(x)} & \mathbf{0}_{n,m} & \mathbf{0}_{n,p} & & & & I_n \end{array}\right]$$

while $G_{lower} = -G_{upper}$. Matrices $H_i^{(x)}$ and $H_i^{(u)}$ select the part of $x_{s_{cen}}$ and $u_{s_{cen}}$ respectively, composing the inequalities constraints. Consequently l_b and l_u will become empty vectors.

Augmented steady vector

This part is based on Graph Theory and, in particular, on Node Interaction presented in Section 5.1. For continuity of that part and this one, we define as

- n_i the length of the extended state vector

$$x_i \leftarrow \begin{bmatrix} x_i \\ [x_j]_{j \in S_i^{OUT}} \end{bmatrix}$$

- m_i the number of columns of

$$B_i \leftarrow \begin{bmatrix} B_i \\ [B_{ji}]_{j \in S_i^{OUT}} \end{bmatrix}$$

that is equal to m_i used before for single agent

- $\bar{m}_i$ the number of columns of

$$\bar{B}_i \leftarrow [B_{ik}]^T_{k \in \mathbb{S}_i^{IN}}$$

- p_i the length of the extended output vector

$$y_i \leftarrow \begin{bmatrix} y_i \\ [y_j]_{j \in S_i^{OUT}} \end{bmatrix}$$

From here and for all this sub-subsection we define with the simple subscript i matrices and vectors relatives to the *augmented* model. However, the substantial difference between this version and the previous one, is the first part of the optimization vector.

$$\mathbf{z} = \begin{bmatrix} x_{s_i} \\ \begin{bmatrix} u_{s_i} \\ \bar{u}_{s_i} \end{bmatrix} \\ y_{s_i} \\ - - - - - \\ \mathbf{z}_{old} \end{bmatrix}$$

It's possible to note that we refer to an augmented vector for both state and output. Moreover it's necessary to take into account the neighbour dynamic through matrix $[B_{ji}]$ and $\bar{B}_i$. All the formulation above for the centralized steady vector is still consistence, not equal, with this new one. The matrix H becomes

$$H = \begin{bmatrix} I_{n_i} \cdot 10^{-6} & & & | & \\ & \begin{bmatrix} I_{m_i} & I_{\bar{m}_i} \end{bmatrix} \cdot 10^{-6} & & | & \\ & & T_i & | & \\ - - - - - & - - - - - - - - - - & - - - & - - & - - - \\ & & & | & H_{old} \end{bmatrix}$$

For vector f we have

$$f = \begin{bmatrix} \mathbf{0}_{n_i,1} \\ \begin{bmatrix} \mathbf{0}_{m_i,1} \\ \mathbf{0}_{\bar{m}_i,1} \end{bmatrix} \\ -T_i\, y_{target_i} \\ - - - - - - - - - \\ f_{old} \end{bmatrix}$$

Steady state constraints are in this case

$$x_{s_i} = A_i x_{s_i} + B_i u_{s_i} + \bar{B}_i \bar{u}_{s_i}^{-}$$
$$y_{s_i} = C_i x_{s_i}$$

so the old sub-matrix B_{cen} of matrix D becomes the horizontal concatenation of $\begin{bmatrix} B_i & \bar{B}_i \end{bmatrix}$. The principle for dynamic constraints in matrix D are the same as before. Moreover selection matrices are simpler to build due to fact that optimization vector contains only the augmented-model-variables. They will be modelled by

$$\begin{cases} H_i^{(x)} = I_{n_i} \\ H_i^{(u)} = \begin{bmatrix} I_{m_i} & \mathbf{0}_{m_i, \bar{m}_i} \end{bmatrix} \\ H_i^{(\bar{u})} = \begin{bmatrix} \mathbf{0}_{\bar{m}_i, m_i} & I_{\bar{m}_i} \end{bmatrix} \end{cases}$$

So we encapsulate this new formulation in matrix D and we obtain

$$D = \left[\begin{array}{ccc|ccc} A_i - I_{n_i} & \begin{bmatrix} B_i & \bar{B}_i \end{bmatrix} & \mathbf{0}_{n_i, p_i} & 0 & \cdots & 0 \\ C_i & \mathbf{0}_{p_i, m_i + \bar{m}_i} & -I_{p_i} & 0 & \cdots & 0 \\ \hline H_i^{(x)} & \mathbf{0}_{p_i, m_i + \bar{m}_i} & \mathbf{0}_{n_i, p_i} & & & \\ \mathbf{0}_{n_i, n_i} & \bar{B}_i H_i^{(\bar{u})} & \mathbf{0}_{n_i, p_i} & & & \\ \vdots & \vdots & \vdots & & D_{old} & \\ \mathbf{0}_{n_i, n_i} & \bar{B}_i H_i^{(\bar{u})} & \mathbf{0}_{n_i, p_i} & & & \end{array}\right]$$

$$d = \begin{bmatrix} \mathbf{0}_{n_i, 1} \\ \mathbf{0}_{m_i + \bar{m}_i, 1} \\ \hline d_{old} \end{bmatrix}$$

As before in Centralized steady vector, it's not possible to model lower and upper bound simply with

$$l_b \leq \mathbf{z} \leq u_b$$

If we take in to account the new augmented steady vector with selection matrices, it's possible to formulate them with

$$\begin{bmatrix} G_{upper} \\ G_{lower} \end{bmatrix} \mathbf{z} \leq \begin{bmatrix} g_{upper} \\ g_{lower} \end{bmatrix}$$

where

$$g_{upper} = \left[\underbrace{\begin{array}{cccc} x_{\max_i} & u_{\max_i} & \dots & x_{\max_i} \end{array}}_{N*2+1} \right]$$

$$g_{lower} = \begin{bmatrix} -x_{\min_i} & -u_{\min_i} & \dots & -x_{\min_i} \end{bmatrix}$$

Recalling procedure for matrix D we build sub-matrix G_{upper} as

$$G_{upper} = \left[\begin{array}{ccc|ccccc} H_i^{(x)} & \begin{bmatrix} \mathbf{0}_{n_i,m_i} & \mathbf{0}_{n_i,\bar{m}_i} \end{bmatrix} & \mathbf{0}_{n_i,p_i} & I_{n_i} & & & \\ \mathbf{0}_{n_i,n_i} & H_i^{(u)} & \mathbf{0}_{n_i,p_i} & & I_{m_i} & & \\ \vdots & \vdots & \vdots & & & \ddots & \\ H_i^{(x)} & \begin{bmatrix} \mathbf{0}_{n_i,m_i} & \mathbf{0}_{n_i,\bar{m}_i} \end{bmatrix} & \mathbf{0}_{n_i,p_i} & & & & I_{n_i} \end{array} \right]$$

while $G_{lower} = -G_{upper}$. Consequently l_b and l_u are empty vectors.

	Centralized Time Iteration	Single agent Time Iteration	Single agent Cooperative Iter.
SSTO	$\begin{bmatrix} x_{s_{cen}} \\ u_{s_{cen}} \\ y_{s_{cen}} \end{bmatrix}$	$\begin{bmatrix} x_{s_i} \\ u_{s_i} \\ y_{s_i} \end{bmatrix}$	$\begin{bmatrix} x_{s_i} \\ u_{s_i} \\ y_{s_i} \end{bmatrix}$
DYN-OPT	$\mathbf{z} = \mathbf{z}_{old}$	$\mathbf{z} = \mathbf{z}_{old}$	$\mathbf{z} = \mathbf{z}_{old}$

Table 5.2: Two Step Algorithms variables

	Centralized One Step	Augmented One Step
SSTO	/	/
DYN-OPT	$\begin{bmatrix} x_{s_{cen}} \\ u_{s_{cen}} \\ y_{s_{cen}} \\ \mathbf{z}_{old} \end{bmatrix}$	$\begin{bmatrix} x_{s_i} \\ u_{s_i} \\ \bar{u}_{s_i} \\ y_{s_i} \\ \mathbf{z}_{old} \end{bmatrix}$

Table 5.3: One Step Algorithms variables

CHAPTER 6

Application Examples

Prova

6.1 Application 1

Prova 1

6.2 Application 2

Prova 2

APPENDIX A

Stability Theory

We introduce now basic stability theory and we'll focus on discrete time systems. We'll take basic notes on stability analysis and Lyapunov functions. We'll proceed analysing such proprieties relative to closed-loop systems. Finally we'll talk about stability results on MPC.

A.1 Some preliminary definitions

We consider systems of the form 3.4 with $f : \mathbb{R}^n \times \mathbb{R}^m$ continuous. Let $\phi(k;\, x,\, \mathbf{u})$ the solution of 3.4 at instant k considering initial state $x(0) = x_0 = x$ and control sequence $\mathbf{u} = \{u(0),\, u(1), \dots\}$; solution exists and is unique. If we have a state feedback control law $u = \kappa(x)$, we obtain a closed-loop system as $x^+ = f\ (x,\, \kappa(x))$ and solution is $\phi(k;\, x,\, \kappa(x))$.

We would like to be sure that the controlled system is *stable*, i.e., small perturbations of the initial state do not cause large variations in the subsequent behaviour of the system, and the state converges to a desired state or, if this is impossible due to disturbances, to a desired set of states. These objectives are made precise in Lyapunov stability theory; in this theory, the system is assumed

given and conditions ensuring the *stability*, or *asymptotic stability* of a specified state or set are sought; the terms stability and asymptotic stability are defined below. If convergence to a specified state, x^* say, is sought, it is desirable for this state to be an *equilibrium point*:

Definition A.1. ***(Equilibrium Point).*** *A point x^* is an equilibrium point of $x^+ = f\ (x,\, \kappa\,(x))$ if $x\,(0) = x^*$ implies $x\,(k) = \phi\,(k;\, x^*) = x^*$ for each $k \geq 0$. If x^* is an equilibrium point it satisfies $x^* = f\ (x^*)$*

A point x^* is isolated if there are no other equilibrium points in a sufficiently small neighbourhood of x^*. A linear system $x^+ = A\,x + b$ has a single equilibrium point computed by $x^* = (I - A)^{-1}\,b$ if $I - A$ is invertible. If convergence to a set $\mathcal{A}$ is sought, it is desirable for the set $\mathcal{A}$ to be positive invariant:

Definition A.2. ***(Positive Invariant Set).*** *A set $\mathcal{A}$ is positive invariant for the system $x^+ = f\ (x)$, if $x \in \mathcal{A}$ implies $f\,(x) \in \mathcal{A}$.*

Before introducing the concepts of stability and asymptotic stability and their characterization by Lyapunov functions, it is convenient to make a few definitions.

Definition A.3. ***$\mathcal{K}$ functions.***

- *A function $\sigma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is of class $\mathcal{K}$ if it is continuous, zero at zero, strictly increasing;*
- *A function $\sigma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is of class $\mathcal{K}_\infty$ if it is of class $\mathcal{K}$ and not limited;*
- *A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{I}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is of class $\mathcal{K}\,\mathcal{L}$ if it is continuous, for each $t \geq 0$, $\beta\,(\cdot,\, t)$ is of class $\mathcal{K}$ and for each $s \geq 0$, $\beta\,(s,\, \cdot)$ is not increasing and satisfies $\lim_{t\to\infty}\beta\,(s,\, t) = 0$;*
- *A function $\gamma : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is of class $\mathcal{K}\,\mathcal{P}$ (positive definite) if it is continuous and positive everywhere except at the origin.*

A.2 Stability and Asymptotic Stability

Let $x^+ = f\ (x)$ with origin as equilibrium point.

Definition A.4. ***(Local stability).*** *The origin is locally stable if for every $\epsilon > 0$ there exists $\delta > 0$ such that $|x| < \delta$ implies $|\phi\,(k;\, x)| < \epsilon$. In Figure A.1 we note a simple example.*

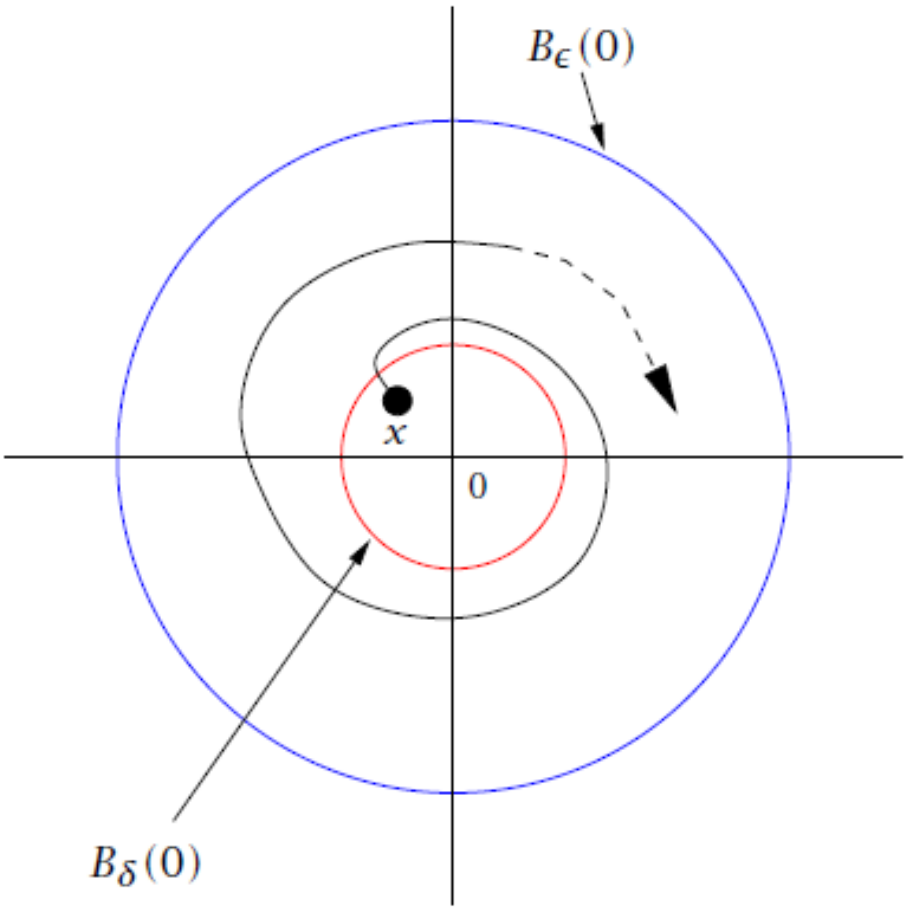

Figure A.1: Stability of the origin

Definition A.5. ***(Global Attraction).*** *The origin is globally attractive for the system* $x^+ = f\ (x)$ *if* $\lim_{k\to\infty} |\phi\,(k;\ x)| = 0\,\forall\, x \in \mathbb{R}^n$

Definition A.6. ***Global Asymptotic Stability (GAS).*** *The origin is globally asymptotically stable for* $x^+ = f\ (x)$ *if it is locally stable and globally attractive*

Definition A.7. ***Global Exponential Stability (GES)*** *The origin is globally exponentially stable if there exists* $c > 0$ *and* $\gamma \in (0,\ 1)$ *such that:* $|\phi\,(k;\ x)| \leq c\,|x|\,\gamma^k$ *for each* $k \geq 0$.

Let $\mathbb{X}$ be positively invariant set for $x^+ = f\ (x)$, such that we have a constrained system in $\mathbb{X}$. The origin is:

- **locally stable** in $\mathbb{X}$ if for each $\epsilon > 0 \, \exists \, \delta > 0$ such that for each $x \in \mathbb{X} \cap \delta \mathbb{B}$ we obtain $|\phi(k;\, x)| < \epsilon \, \forall \, k \geq 0$;
- **attractive** if for each $x \in \mathbb{X}$ we have $\lim_{k \to \infty} |\phi(k;\, x)| = 0$;
- **asymptotically stable** in $\mathbb{X}$ if it is attractive and locally stable;
- Global Asymptotic Stability is equivalent to $|\phi(k;\, x)| \leq \beta(|x|,\, k)$ for each $k \geq 0$, $\beta(\cdot) \in \mathcal{K}\mathcal{L}$.

$\mathbb{X}$ is defined as **region (or domain) of attraction** for the origin.

A.3 Lyapunov Stability Theory

Definition A.8. ***Lyapunov function.*** *A function $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a Lyapunov function for $x^+ = f(x)$ if there exist $\alpha_i \in \mathcal{K}_\infty$, $i = 1,\, 2$ and $\alpha_3 \in \mathcal{P}$, $\mathcal{D}$ such that for all $x \in \mathbb{R}^n$:*

$$\begin{aligned} V(x) &\geq \alpha_1(|x|) \\ V(x) &\geq \alpha_2(|x|) \\ V(f(x)) - V(x) &\leq -\alpha_3(|x|) \end{aligned} \tag{A.1}$$

V decreases during the evolution of the system.

Theorem A.9. ***Lyapunov function and GAS.*** *If $V(\cdot)$ is a Lyapunov function for $x^+ = f(x)$the origin is globally asymptotically stable.*

Theorem (A.9) provides a sufficient condition for global asymptotic stability that might be thought to be conservative.

The appropriate generalization of Theorem (A.9) for the constrained case is described below.

Theorem A.10. ***Lyapunov and GAS for Constrained systems.*** *The origin is asymptotically stable in $\mathbb{X}$ if:*

- *$\mathbb{X}$ is positively invariant for $x^+ = f(x)$;*
- *$V(\cdot)$ is a Lyapunov function for $x^+ = f(x)$.*

Theorem A.11. ***Lyapunov and GES for Constrained systems.*** *The origin of $x^+ = f(x)$ is exponentially stable in $\mathbb{X}$ if:*

- *$\mathbb{X}$ is positively invariant for $x^+ = f(x)$;*

- *there exist* $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ *and positive constants* σ, a_1, a_2, $a_3 > 0$*:*

$$a_1 |x|^\sigma \leq V(x) \leq a_2 |x|^\sigma \tag{A.2}$$

$$V(f(x)) - V(x) \leq -a_3 |x|^\sigma \tag{A.3}$$

We review some facts involving the discrete matrix Lyapunov equation and stability of the linear system $x^+ = A\,x$. The discrete time system is asymptotically stable if and only if the magnitudes of the eigenvalues of A are strictly less than unity. In this case a matrix A is called stable, convergent, or discrete time Hurwitz. The following matrix equation is known as a discrete matrix Lyapunov equation,

$$A^{'} S A - S = -Q \tag{A.4}$$

Lemma A.12. *The properties of solutions to this equation allow one to draw conclusions about the stability of A without computing its eigenvalues.*

- A *is stable;*
- *For each* $Q \in \mathbb{R}^{n \times m}$*, there is a unique solution* S *of (A.4) and if* $Q > 0$*, then* $S > 0$*;*
- *There is some* $S > 0$ *such that* $A^{'} S A - S < 0$*;*
- *There is some* $S > 0$ *such that* $V(x) = x^{'} S x$ *is a Lyapunov function for the system* $x^+ = A\,x$*.*

APPENDIX B

Theory of State Estimation

In most application, the measured variables y are usually, a subset of those used to modelling the system, x. Such measures are corrupted by sensors noise. Even the state evolution is corrupted by process noise. We sould have a good state estimation. In this section will be exposed probability fundamental. It is necessary to have a good estimator: a discrete time model subject to process and sensors noise that is normally distributed is presented (Kalman Filter).

B.1 Linear Systems and Normal Distribution

We denote

$$\begin{aligned} x &\sim N(m, P) \\ p_x(x) &= n(x, m, P) \end{aligned} \tag{B.1}$$

a normally distributed random variable x with mean m and covariance, or simply variance, P. Let it be

$$n(x, m, P) = \frac{1}{(2\pi)^{n/2} (\det P)^{1/2}} \exp\left[-\frac{1}{2}(x-m)' P^{-1} (x-m)\right]$$

the distribution of random variable x. From this formulation derive some following useful results. Let x and y two random variables *(statistically) independent* as follows

$$x \sim N\left(m_x\,,\,P_x\right) \qquad y \sim N\left(m_y\,,\,P_y\right)$$

then, their joint density probability is

$$p_{x,y}\left(x,\,y\right) = n\left(x,\,m_x,\,P_x\right) n\left(y,\,m_y,\,P_y\right)$$

$$\left[\begin{array}{c} x \\ y \end{array}\right] \sim N\left(\left[\begin{array}{c} m_x \\ m_y \end{array}\right], \left[\begin{array}{cc} P_x & 0 \\ 0 & P_y \end{array}\right]\right) \tag{B.2}$$

Let x a random variable like in equation B.1. Let y a linear transformation of x, as $y = A\,x$, then y is distributed as

$$y \sim N\left(Am,\,APA^{'}\right) \tag{B.3}$$

Let x and y two random variables jointly normally distributed, $(x,\,y)$ (not independents)

$$\left[\begin{array}{c} x \\ y \end{array}\right] \sim N\left(\left[\begin{array}{c} m_x \\ m_y \end{array}\right], \left[\begin{array}{cc} P_x & P_{xy} \\ P_{yx} & P_y \end{array}\right]\right)$$

then the conditional density of x given y is also normal

$$p_{x|y}\left(x\,|\,y\right) = n\left(x,\,m,\,P\right) \tag{B.4}$$

in which mean and variance are

$$m = m_x + P_{xy}P_y^{-1}\left(y - m_y\right)$$

$$P = P_x - P_{xy}P_y^{-1}P_{yx}$$

These are the three main results. To derive the optimal estimator, we require the conditional on other additional random variables. Results, similarly to those above are here briefly exposed.

Let $p_{x|z}\left(x\,|\,z\right)$ normal, and let y statistically independent from x and z and normally distributed, as

$$p_{x|z}\left(x\,|\,z\right) = n\left(x,\,m,\,P\right) \qquad y \sim N\left(m_y\,,\,P_y\right)$$

then the conditional probability density function of $(x,\, y)$ given z is

$$p_{x,y|z}\left(x, y \,|\, z\right) = n\left(x,\, m_x,\, P_x\right) n\left(y,\, m_y,\, P_y\right)$$

$$p_{x,y|z}\left(\left[\begin{array}{c} x \\ y \end{array}\right] \middle|\, z\right) = n\left(\left[\begin{array}{c} x \\ y \end{array}\right], \left[\begin{array}{c} m_x \\ m_y \end{array}\right], \left[\begin{array}{cc} P_x & 0 \\ 0 & P_y \end{array}\right]\right) \tag{B.5}$$

If we have a linear transformation, we obtain, similarly to B.3

$$p_{x|z}\left(x \,|\, z\right) = n\left(x,\, m,\, P\right) \quad y = A\, x$$

$$p_{y|z}\left(y \,|\, z\right) = n\left(y,\, Am,\, APA^{'}\right) \tag{B.6}$$

If x and y are jointly and normally distributed (not independents) as

$$p_{x,y|z}\left(\left[\begin{array}{c} x \\ y \end{array}\right] \middle|\, z\right) = n\left(\left[\begin{array}{c} x \\ y \end{array}\right], \left[\begin{array}{c} m_x \\ m_y \end{array}\right], \left[\begin{array}{cc} P_x & P_{xy} \\ P_{yx} & P_y \end{array}\right]\right)$$

then the conditional density of x given y, z is also normal

$$p_{x,y|z} = \left(x|y,\, z\right) = n\left(x,\, m,\, P\right) \tag{B.7}$$

in which

$$m = m_x + P_{xy} P_y^{-1}\left(y - m_y\right)$$
$$P = P_x - P_{xy} P_y^{-1} P_{yx}$$

B.2 Linear Optimal State Estimation

We assume $x\left(0\right)$ be normally distributed

$$x\left(0\right) \sim N\left(\bar{x}\left(0\right),\, Q\left(0\right)\right)$$

In operational environment we don't know $\bar{x}\left(0\right)$ or $Q\left(0\right)$ and we often set initial values of $\bar{x}\left(0\right) = 0$ and large value of $Q\left(0\right)$; it means our lack of prior knowledge little starting information. Measurement $y\left(0\right)$ satisfy

$$y\left(0\right) = C\, x\left(0\right) + v\left(0\right)$$

where $v\left(0\right) \sim N\left(0,\, R\right)$ is system noise. If measurement is quite noisy, then R is large. Given $y\left(0\right)$ we want to obtain the conditional density $p_{x(0)|y(0)}\left(x\left(0\right) \,|\, y\left(0\right)\right)$.

Such, describe our knowledge about the state $x(0)$ after we have measured $y(0)$. These couple of variables satisfy

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} I & 0 \\ C & I \end{bmatrix} \begin{bmatrix} x(0) \\ v(0) \end{bmatrix}$$

Let $x(0)$and $v(0)$ statistically independent; using B.2 we have

$$\begin{bmatrix} x(0) \\ v(0) \end{bmatrix} \sim N\left(\begin{bmatrix} \bar{x}(0) \\ 0 \end{bmatrix}, \begin{bmatrix} Q(0) & 0 \\ 0 & R \end{bmatrix} \right)$$

We know $y(0)$ is a linear transformation of $x(0)$, thus they are not independents; using (B.3) we have

$$\begin{bmatrix} x(0) \\ y(0) \end{bmatrix} \sim N\left(\begin{bmatrix} \bar{x}(0) \\ C\bar{x}(0) \end{bmatrix}, \begin{bmatrix} Q(0) & Q(0)C' \\ CQ(0) & CQ(0)C' + R \end{bmatrix} \right)$$

Given the jointly density probability of it and using B.4 we obtain

$$p_{x(0)|y(0)}(x(0) \mid y(0)) = n(x(0), m, P)$$

in which

$$\begin{aligned} m &= \bar{x}(0) + L(0)(y(0) - C\bar{x}(0)) \\ L(0) &= Q(0)C'\left(CQ(0)C' + R\right)^{-1} \\ P &= Q(0) - Q(0)C'\left(CQ(0)C' + R\right)^{-1} CQ(0) \end{aligned}$$

The optimal state estimation is the value of $x(0)$ that maximizes conditional density $p_{x(0)|y(0)}$. For a normal distribution that is the mean, $\hat{x}(0) = m$. For continuity we denote $P(0) = P$. The change from $P(0)$ to $Q(0)$ is the increasing of information given by measure $y(0)$.

Now we forecast the state evolution and we observe if there is a change in the information gain. State evolution satisfies

$$x(1) = \begin{bmatrix} A & I \end{bmatrix} \begin{bmatrix} x(0) \\ w(0) \end{bmatrix}$$

where $w(0) \sim N(0, Q)$ is the process noise. We want to compute the conditional density $p_{x(1)|y(0)}$. We have to analyse the conditional version of jointly probability density function $(x(0), w(0))$ given $y(0)$. Observing B.5

$$\begin{bmatrix} x(0) \\ w(0) \end{bmatrix} \sim N\left(\begin{bmatrix} \hat{x}(0) \\ 0 \end{bmatrix}, \begin{bmatrix} P(0) & 0 \\ 0 & Q \end{bmatrix}\right)$$

We use then the conditional form of linear transformation such as B.6 to obtain

$$p_{x(1)|y(0)}(x(1) \mid y(0)) = n\left(x(1), \widehat{x}^{-}(1), P^{-}(1)\right)$$

in which mean and variance are

$$\hat{x}^{-}(1) = A\hat{x}(0) \qquad P^{-}(1) = AP(0)A^{'} + Q$$

(Superscripts minus indicates these are statistics before measurements $y(1)$). Once we have obtained measure $y(1)$ at the next instant and being $p_{x(1)|y(0)}$ also normal we can iterate the procedure for all avaialable measurements. If the eigenvalues of A are within the unit circle $AP(0)A^{'}$ will be smaller than $P(0)$ (Q has a positive contribution). This means if system is stable, step by step, variance decreasing, i.e., there is less uncertain on state estimation.

Summarizing, we denote with

$$\mathbf{y}(k) := \{y(0), y(1), \ldots, y(k)\}$$

At step k let the conditional probability density function to $\mathbf{y}(k-1)$ normally distributed as

$$p_{x(k)|\mathbf{y}(k-1)}(x(k) \mid \mathbf{y}(k-1)) = n\left(x(k), \hat{x}^{-}(k), P^{-}(k)\right)$$

Let the initial condition as $\hat{x}^{-}(0) = \bar{x}(0)$ and $P^{-}(0) = Q(0)$, we have a random variable distributed as

$$x \sim N\left(\hat{x}^{-}(k), P^{-}(k)\right)$$

We obtain $y(k)$ that satisfies

$$\begin{bmatrix} x(k) \\ y(k) \end{bmatrix} = \begin{bmatrix} I & 0 \\ C & I \end{bmatrix} \begin{bmatrix} x(k) \\ v(k) \end{bmatrix}$$

As $x(k)$ and $v(k)$ are independent

$$\begin{bmatrix} x(k) \\ y(k) \end{bmatrix} \sim N\left(\begin{bmatrix} \hat{x}^{-}(k) \\ C\hat{x}^{-}(k) \end{bmatrix}, \begin{bmatrix} P^{-}(k) & P^{-}C' \\ CP^{-}(k) & CP^{-}(k)C' + R \end{bmatrix} \right)$$

Denote $\{\mathbf{y}(k-1), y(k)\} = \mathbf{y}(k)$. Using conditional density (B.7) $(x(k) \mid y(k)) \sim N(\hat{x}(k), P(k))$ with

$$\begin{aligned} \hat{x}(k) &= \hat{x}^{-}(k) + L(k)(y(k) - C\hat{x}^{-}(k)) \\ L(k) &= P^{-}(k)C'\left(CP^{-}(k)C' + R\right)' \\ P(k) &= P^{-}(k) - P^{-}(k)C'\left(CP^{-}(k)C' + R\right)^{-1}CP^{-}(k) \end{aligned}$$

Iterating this calculus to next step and following the system dynamic we obtain the density function given by linear transformation similarly B.6

$$p_{x(k+1)|\,\mathbf{y}(k)}(x(k+1) \mid \mathbf{y}(k)) = n\left(x(k+1), \widehat{x}^{-}(k+1), P^{-}(k+1)\right)$$

where

$$\begin{aligned} \hat{x}^{-}(k+1) &= A\hat{x}(k) \\ P^{-}(k+1) &= AP(k)A' + Q \end{aligned}$$

B.3 Moving Horizon Estimation

When using non-linear system or estimation constraints, we cannot compute the conditional density in closed form as in Kalman Filter. Moving horizon estimation remove these difficulties only considering the lasts N measures and find only the lasts N trajectory values. We have to estimate $\mathbf{x}_N(T) = \{x(T-N), \ldots, x(T)\}$ states given $\mathbf{y}_N(T) = \{y(T-N), \ldots, y(T)\}$ measures. We assume temporal window always full. The simplest form of MHE is the following least squares problem,

$$\min_{\mathbf{x}_N(T)} \hat{V}_T(\mathbf{x}_N(T)) \tag{B.8}$$

where

$$\hat{V}_T\left(\mathbf{x}_N\left(T\right)\right) = \frac{1}{2}\left(\sum_{k=T-N}^{T-1} \left|x\left(k+1\right) - A\,x\left(k\right)\right|_{Q^{-1}}^2 + \sum_{k=T-N}^{T-1} \left|y\left(k\right) - C\,x\left(k\right)\right|_{R^{-1}}^2\right) \tag{B.9}$$

We denote circumflex hat to indicate the objective function value considering data from $T-N$ to T rather than the entire vector.

In order to establish convergence, following results on optimal estimator cost function proves useful

Lemma B.1. ***Convergence of state estimator.*** *Given the noise-free measures* $\mathbf{y}\left(T\right) = \left\{C\,x\left(0\right),\, CA\,x\left(0\right),\, \ldots,\, CA^T x\left(0\right)\right\}$*, the optimal estimator cost* $V_T^0\left(\mathbf{y}\left(T\right)\right)$ *converge as* $T \to \infty$*.*

Optimal estimator cost converge without accounting of system observability. But if we want that state estimation converge to the real state we have to restrict the system further.

Lemma B.2. ***Convergence of estimation.*** *For* $(A,\, C)$ *observable,* $Q,\, R > 0$ *and noise-free measures* $\mathbf{y}\left(T\right) = \left\{C\,x\left(0\right),\, CA\,x\left(0\right),\, \ldots,\, CA^T x\left(0\right)\right\}$ *the obtained optimal estimation converge to the state* $\hat{x}\left(T\right) \to x\left(T\right)$ *as* $T \to \infty$*.*

The system restriction can be weakened from observability to detectability. The restriction on the process disturbance weight (variance) Q can be weakened from $Q > 0$ to $Q \geq 0$ and $(A,\, Q)$ stabilizable. The restriction $R > 0$ remains to ensure uniqueness of the estimator.

APPENDIX C

Source Code

Listing C.1: SCRIPT Principale

```
%
% This file set the number of nodes and the horizon
% length and optimize under hard input and output
% constraints through a model that implements a graph.
% Each subsystem is composed by an augmented model
% defined by an adjacent matrix. Then a cooperative
% optimization is performed several times untill a
% certain cost has reached.
%

%%
clear all
clc

%%%%%%%%%%%%%%%%%%%%%%%%    DEFINITIONS %%%%%%%%%%%%%%%%%%%%%%%%
%% Main Parameters
% Nodes number
M = 3;
% Horizon length
```

```
N = 10;
% Number of state variable, input and output for a single
% agent, are generated from the uniform distribution on
% the interval [a, b]through the equation
% floor(a + (b-a).*rand).
a = 2;
b = 4;
% The others parameters depend from these two
init_graph_and_optimization_parameters;
% change interaction: here re-define E
time_iter_max = 60;

%% Implements graph
% Implements all subsystem in a state space model
implements_graph_nodes;

% Augmented Adjacent Matrix: arc (i,j) si constraint
[AdjacentMatrix] = create_adjacent_matrix(V,E,M);

% keyboard
%% Create augmented model
% For each subsystem i, we create the augmented model and
% the modified star that includes the old input-star, the
% old output-star and all input-star of j, i exluded. We
% also create Bi_bar. Only ui_bar changes at every time
% instant.
for i = 1:M
    [model{i} S{i} Bi_bar{i}] = create_augmented_model...
        (i,AdjacentMatrix,V,E,M,N);
end

%%%%%%%%%%%%%%   SUBOPTIMAL MPC algorithm %%%%%%%%%%%%%%

% Cooperative distributed model predictive control can be
% viewed as a form of suboptimal mpc. So we can iterate a
% suboptimal algorithm.

% Define time iteration variable 'time_iter' and
% cooperative iteration variable 'coop_iter'. First
% variable is used in the outer loop. Second one is
% used in cooperative (suboptimal) algorithm
```

```
% 'cooperative_optimization'
time_iter = 1
for i=1:M
    xs{i} = 0*V{i,1}.x0;
    us{i} = 0*V{i,1}.u(:,1);
    ys{i} = 0*ytarget{i};
end

while time_iter <= time_iter_max,

    % Input vector has to converge to centralized one
    % We store at each TIME instant the norm of
    % centralized optimum input vector and that
    % obtained from the single cooperative algorithm
    store_for_stats;
    redefine_targets;

    % Step1: Iteration of an optimization method
    %        Using initial state x+ the final input
    %        sequence u+ is a function of these state
    %        initial condition and the warm start
    %        unext = g(xpre,upre)
    cooperative_optimization;

    % Step2: Next time step, unext ---> upre
    return_overall_solution;

    % Step3: State evolution, xnext = f(xpre,upre),
    %        xnext ---> xpre
    state_evolution;

    % Step4: Warm start unext = {u(1),u(2),...,u(N-1),0}
    warm_start;

    time_iter = time_iter + 1
end

%% Graph
plot_simple_graph;
```

Bibliography

[1] Rawlings B. J., Mayne Q. D., **Model Predictive Control: Theory and Design**, Nob Hill Publishing, 2013.

[2] Maciejowski M. J., **Predictive Control with Constraints**, Prentice Hall, 2000.

[3] Ferramosca A., Limon D., Alvarado I., Camacho E.F., "Cooperative distributed MPC for tracking", *Automatica*, Vol. 49, 2013, pp. 906-914.

[4] Stewart T. B., Venkat N. A., Rawlings B. J., Wright J. S., Pannocchia G., "Cooperative distributed model predictive control", *Systems & Control Letters*, Vol. 59, 2010, pp. 460-469.

[5] Limon D., Alvarado I., Alamo T., Camacho E.F., "MPC for tracking piecewise constant references for constrained linear systems", *Automatica*, Vol. 44, 2008, pp. 2382-2387

[6] Scattolini R., "Architectures for distributed and hierarchical Model Predictive Control - A review", *Journal of Process Control*, Vol. 19, 2009, pp. 723-731.

[7] Trodden P., Richards A., "Cooperative distributed MPC of linear systems with coupled constraints", *Automatica*, Vol. 49, 2013, pp.479-487.

[8] Trodden P., Richards A., "Distributed model predictive control of linear systems with persistent disturbances", *International Journal of Control*, Vol. 83, No. 8, 2010, pp. 1653-1663.

[9] Zhang L., Wang J., Liu Z., Li K., "Distributed MPC for Tracking based Reference Trajectories", *Proceedings of the 33rd Chinese Control Conference*, 2014, pp. 7778-7783.

[10] Conte C., Zeilinger N. M., Morari M., Jones N. C., "Cooperative Distributed Tracking MPC for Constrained Linear Systems: Theory and Synthesis", *52nd IEEE Conference on Decision and Control*, 2013, pp. 3812-3817.

[11] Conte C., Voellmy R. N., Zeilinger N. M., Morari M., Jones N. C., "Distributed synthesis and control of constrained linear system", *Proceedings of the American Control Conference*, 2012, pp. 6017-6022.